

Kieser·Meder

---

# MIKRO- PROZESSOR- TECHNIK

Aufbau  
und Anwendung  
des Mikroprozessor-  
systems U880

---



---

# MIKRO- PROZESSOR- TECHNIK

Aufbau und Anwendung  
des Mikroprozessorsystems U880

---

Dipl.-Ing. Heiko Kieser  
Dr.-Ing. Michael Meder

2., durchgesehene Auflage



VEB VERLAG TECHNIK BERLIN

157 Bilder, 64 Tafeln

© VEB Verlag Technik, Berlin, 1982

Unveränderter Nachdruck 1984

Lizenz 201.370/173/84

DK 681.3-181.4:518.5; DK 681.3.06/.07:62-50 · LSV 3054 · VT 3/5537-2

Lektor: *Karl Belter*

Schutzumschlag: *Kurt Beckert*

Printed in the German Democratic Republic

Gesamtherstellung: Offizin Andersen Nexö, Graphischer Großbetrieb, Leipzig III/18/38

Bestellnummer: 553 094 5

03600

# Vorwort

Die Mikroelektronik, besonders die Technik der Mikroprozessoren, ist gegenwärtig wohl unbestritten das Fachgebiet mit der größten Entwicklungsrate. Die schnelle Verbreitung dieser Technik ist gekennzeichnet durch den immer breiteren Einsatz der Mikroprozessoren und die laufende Erschließung weiterer Anwendungsbereiche. Hierin zeigt sich die zentrale Stellung der Mikroprozessortechnik.

Mit der Vorstellung des Mikroprozessorsystems U880 durch den VEB Funkwerk Erfurt wurde eine breite Basis für die Realisierung der genannten Aufgaben geschaffen. Naturgemäß erwächst daraus ein hoher Bedarf an Informations-, Dokumentations- und Applikationsmaterial zu dieser Schaltkreisfamilie. Das vorliegende Buch soll insbesondere helfen, diese Lücke zu schließen. Deshalb nimmt einerseits die Beschreibung der Systemelemente des Mikroprozessorsystems einen breiten Raum ein; andererseits wurde besonderes Augenmerk auf eine umfangreiche applikative Darstellung gelegt, und die Vielzahl der angegebenen Schaltungsbeispiele und Programmierbeispiele unterstreicht dies. In diesem Zusammenhang sei darauf hingewiesen, daß die angegebenen Dimensionierungen Richtwerte darstellen, die im Laufe des technischen Fortschritts möglicherweise Änderungen erfordern. Deshalb sollten die gültigen Standards bei der Anwendung der Applikationen herangezogen werden.

Prinzipiell werden beim Leser dieses Buches gewisse Grundlagenkenntnisse auf dem Gebiet der Mikroprozessortechnik vorausgesetzt. Es wurde jedoch versucht, den vorliegenden Stoff durch sinnvolle Anordnung und Gliederung verständlich darzustellen, so daß der Leser einen sehr umfassenden und verallgemeinerungsfähigen Einblick in die Technik der gegenwärtigen Mikroprozessoren erhält.

Das Buch ist deshalb besonders für Praktiker bestimmt, also für Ingenieure aus Forschung und Entwicklung, Rationalisierung, Produktion sowie Planung und Leitung. Studierenden gewährt es eine wertvolle Vorbereitung auf den künftigen Beruf.

An dieser Stelle möchten wir uns für die gute Zusammenarbeit mit dem Verlag, vor allem mit unserem Lektor, Herrn *K. Belter*, bedanken.

*H. Kieser M. Meder*

# Inhaltsverzeichnis

<b>1. Einführung in die Thematik</b> .....	11
1.1. Entwicklung der Mikroprozessortechnik .....	11
1.2. Umsetzung von Problemen in Algorithmen .....	13
1.3. Zuordnung von Anwendungsbereichen .....	14
<b>2. Vergleich der ersten und der zweiten Leistungsklasse von Mikroprozessoren</b> .....	16
2.1. Beschreibung des Systems U808 .....	16
2.2. Übersicht über das System U880 .....	18
2.3. Qualitative Gegenüberstellung .....	19
<b>3. Beschreibung der Elemente des Systems U880</b> .....	20
3.1. Zentrale Verarbeitungseinheit .....	20
3.1.1. Einführung .....	20
3.1.2. Struktureller Aufbau .....	20
3.1.2.1. Register .....	21
3.1.2.2. Arithmetik- und Logikeinheit (ALU) .....	23
3.1.2.3. Befehlsdekoder und CPU-Steuerung .....	23
3.1.2.4. Daten- und Adreßbussteuerung .....	23
3.1.3. Erläuterung der Anschlußbelegung .....	23
3.1.4. Zeitverhalten .....	26
3.1.4.1. Befehlscodelesen .....	27
3.1.4.2. Speicherlesen oder -schreiben .....	28
3.1.4.3. Datenein- und -ausgabe .....	30
3.1.4.4. Busanforderung und -bestätigung .....	30
3.1.4.5. Interruptanforderung und -bestätigung .....	31
3.1.4.6. Rücksprung aus HALT .....	33
3.1.5. Funktionelle Befehlsdarstellung .....	33
3.1.5.1. Einführung in die Befehlstypen .....	33
3.1.5.2. Adressierungsarten .....	34
3.1.5.3. Operandenbeschreibung .....	37
3.1.5.4. Mnemotechnische Instruktionsdarstellung .....	39
3.1.5.5. Zusammenfassende Befehlsangabe .....	46
3.1.5.6. Flagoperationen .....	46
3.1.5.7. Analyse der Befehle nach Maschinenzyklen .....	64
3.1.6. Interruptverhalten .....	71
3.1.6.1. Zweck des Interrupts .....	71
3.1.6.2. Sperrung und Freigabe .....	71
3.1.6.3. Verhalten der CPU .....	72
3.1.7. Zusammenfassung der technischen Daten .....	76
3.2. Parallele Ein-/Ausgabe-Einheit .....	78
3.2.1. Einführung .....	78
3.2.2. Struktureller Aufbau .....	79

3.2.3. Erläuterung der Anschlußbelegung .....	81
3.2.4. Auswahl der Betriebsarten .....	84
3.2.4.1. Rücksetzen der PIO .....	84
3.2.4.2. Interruptvektor .....	84
3.2.4.3. Betriebsartenwahl .....	85
3.2.4.4. Interruptsteuerwort .....	87
3.2.4.5. Zusammenfassung .....	89
3.2.5. Beschreibung der Betriebsarten .....	89
3.2.5.1. Byteausgabe .....	89
3.2.5.2. Byteeingabe .....	91
3.2.5.3. Bidirektionaler Betrieb .....	93
3.2.5.4. Bitbetrieb .....	94
3.2.6. Interruptbearbeitung .....	96
3.2.7. Einsatz der PIO in Mikrorechnersystemen .....	98
3.2.7.1. Hardware .....	98
3.2.7.2. Software .....	100
3.2.7.3. Anwendungsmöglichkeiten .....	101
3.2.8. Zusammenfassung der technischen Daten .....	104
3.3. Serielle Ein-/Ausgabe-Einheit .....	105
3.3.1. Einführung .....	105
3.3.2. Struktureller Aufbau .....	106
3.3.3. Erläuterung der Anschlußbelegung .....	107
3.3.4. Beschreibung der Betriebsarten .....	110
3.3.5. Ein-/Ausgabe-Varianten .....	112
3.3.6. Asynchrone Moden .....	114
3.3.6.1. Senden .....	115
3.3.6.2. Empfang .....	115
3.3.7. Synchrone Moden .....	117
3.3.7.1. Monosync .....	117
3.3.7.2. Bisync .....	117
3.3.7.3. Extern-Sync .....	118
3.3.7.4. Senden .....	118
3.3.7.5. Empfang .....	121
3.3.7.6. SDLC-, HDLC-Protokoll .....	123
3.3.8. Programmierung .....	124
3.3.8.1. Schreibregister .....	124
3.3.8.2. Leseregister .....	131
3.3.9. Typische Anwendungen .....	134
3.3.10. Zusammenfassung der technischen Daten .....	139
3.4. Zähler/Zeitgeber .....	139
3.4.1. Einführung .....	139
3.4.2. Struktureller Aufbau .....	140
3.4.3. Erläuterung der Anschlußbelegung .....	142
3.4.4. Auswahl der Betriebsarten .....	144
3.4.4.1. Rücksetzen .....	144
3.4.4.2. Interruptvektor .....	144
3.4.4.3. Kanalsteuerwort .....	145
3.4.4.4. Zeitkonstante .....	147
3.4.4.5. Zusammenfassung .....	148
3.4.5. Beschreibung der Betriebsarten .....	148
3.4.5.1. Zählermode .....	148
3.4.5.2. Zeitgebermode .....	150

3.4.6.	Interruptbearbeitung .....	151
3.4.7.	Einsatz in Mikrorechnersystemen .....	154
3.4.7.1.	Hardware .....	154
3.4.7.2.	Software .....	155
3.4.7.3.	Anwendungsmöglichkeiten .....	155
3.4.8.	Zusammenfassung der technischen Daten .....	161
<b>4.</b>	<b>Eigenschaften des Prozessorsystems U880 .....</b>	<b>162</b>
4.1.	Leistungsbestimmende Parameter .....	162
4.2.	Beschreibung des U880-Interruptsystems .....	163
4.2.1.	Einführung .....	163
4.2.2.	Möglichkeiten der Interruptbearbeitung .....	163
4.2.2.1.	Einordnung im System .....	163
4.2.2.2.	Nichtmaskierbarer Interrupt .....	164
4.2.2.3.	Maskierbarer Interrupt .....	167
4.2.3.	Interrupteigenschaften der Systemelemente .....	172
4.2.3.1.	Priorität .....	172
4.2.3.2.	Anmeldung und Quittierung .....	176
4.2.3.3.	Rückkehr aus Bearbeitung .....	179
4.2.4.	Interruptstruktur der peripheren Systemelemente .....	192
4.2.5.	Programmbearbeitung .....	197
4.3.	Beschreibung der DMA-Eigenschaften .....	199
4.3.1.	Einführung .....	199
4.3.2.	Möglichkeiten der DMA-Bearbeitung .....	199
4.3.2.1.	Einordnung im System .....	199
4.3.2.2.	Schneller Datenzugriff .....	200
4.3.2.3.	Transparenter Betrieb .....	200
4.3.3.	Eigenschaften von DMA-Einheiten .....	202
4.3.3.1.	Priorität .....	202
4.3.3.2.	Busanforderung und -quittierung .....	203
4.3.3.3.	Rückkehr aus DMA-Betrieb .....	204
4.3.4.	Einfluß auf Programmierung .....	204
<b>5.</b>	<b>Ergänzungselemente eines Mikrorechners .....</b>	<b>205</b>
5.1.	Speicherbauelemente .....	205
5.1.1.	Festwertspeicher .....	205
5.1.2.	Schreib/Lese-Speicher .....	211
5.2.	Bipolare Standardbauelemente .....	217
<b>6.</b>	<b>Aufbau des Systems U880 .....</b>	<b>223</b>
6.1.	Minimalkonfiguration .....	223
6.2.	Erweiterungsmöglichkeiten .....	227
6.2.1.	Anforderungen .....	227
6.2.2.	Prozessorbaugruppe .....	231
6.2.3.	Speicherbaugruppe .....	234
6.2.3.1.	Festwertspeicher .....	234
6.2.3.2.	Statische Schreib/Lese-Speicher .....	236
6.2.3.3.	Dynamische Schreib/Lese-Speicher .....	240
6.2.4.	Rechnerperipherie .....	244
6.3.	Anwendervarianten .....	253

<b>7. Anwendung des Systems in einer frei programmierbaren Steuerung</b> .....	256
7.1. Einsatzbereiche und Konfiguration .....	256
7.2. Baugruppenbeschreibung .....	257
7.2.1. Prozessorbaugruppe .....	257
7.2.2. Speicherbaugruppen .....	260
7.2.2.1. Kombinierte Speicherbaugruppe .....	260
7.2.2.2. CMOS-Speicherbaugruppe .....	261
7.2.2.3. Statische RAM-Speicherbaugruppe .....	262
7.2.2.4. Dynamische RAM-Speicherbaugruppe .....	262
7.2.3. Standard-Ein-/Ausgabe-Baugruppen .....	264
7.2.3.1. Parallele Standardschnittstelle .....	264
7.2.3.2. Serielle Standardschnittstelle .....	267
7.2.4. Prozeßnahe Ein-/Ausgabe-Baugruppen .....	269
7.2.4.1. Ausgabeinterface .....	269
7.2.4.2. Eingabeinterface .....	270
7.2.4.3. Digital/Analog-Wandler .....	274
7.2.4.4. Analog/Digital-Wandler .....	275
7.2.5. Ergänzungsbaugruppen .....	276
7.3. Programmierarbeit .....	277
<b>8. Anwendung des Prozessors U880 in einem Lernsystem</b> .....	279
8.1. Aufgabenstellung und Konfiguration .....	279
8.2. Baugruppenbeschreibung .....	281
8.2.1. Prozessorbaugruppe .....	281
8.2.2. Speicherbaugruppe .....	282
8.2.3. Tastatur- und Anzeigebaugruppe .....	284
8.2.4. Rückverdrahtung .....	285
8.2.5. Stromversorgungsbaugruppe .....	286
8.3. Beschreibung der Betriebssoftware .....	286
8.3.1. Systemvereinbarungen .....	286
8.3.2. Aufbau und Arbeitsweise des Monitorprogramms .....	291
8.3.3. Kommandobeschreibung .....	293
8.3.4. Unterprogrammtechnik .....	299
8.3.5. Anwendungshinweise .....	301
8.4. Erweiterungsmöglichkeiten .....	302
8.4.1. PROM-Programmierbaugruppen .....	302
8.4.1.1. Allgemeines .....	302
8.4.1.2. Programmierbaugruppe für U551/U552 .....	303
8.4.1.3. Programmierbaugruppe für U555 .....	306
8.4.2. FPS2-Baugruppen .....	307
8.4.3. Anwendung höherer Programmiersprachen .....	308
<b>9. Begriffserklärung</b> .....	310
<b>Anhang</b> .....	319
A.1. Programmunterlagen zum U808 .....	319
A.2. Programmunterlagen zum U880 .....	322
A.3. Monitorbetriebsprogramm des Lernsystems .....	328

A.4. Unterlagen zur Prozessorbaugruppe des Lernsystems.....	339
A.5. Unterlagen zur Speicherbaugruppe des Lernsystems.....	343
A.6. Unterlagen zur Tastatur/Anzeige-Bedienbaugruppe .....	345
A.7. Unterlagen zum Netzteil des Lernsystems .....	348
<b>Literaturverzeichnis</b> .....	<b>350</b>
<b>Sachwörterverzeichnis</b> .....	<b>351</b>

# 1. Einführung in die Thematik

Die Untersuchung von vielen Steuer- und Regelungsprozessen hat gezeigt, daß deren Automatisierung zweckmäßigerweise nur mit Hilfe von datenverarbeitenden Systemen erfolgen kann. Das ist einer der Gründe, warum die Mikroprozessortechnik in den letzten Jahren zu einem dominierenden Faktor der Mikroelektronik geworden ist. Die entsprechenden Beschlüsse von Partei und Regierung der DDR unterstreichen diese Feststellung. Es ist das Ziel, auf der Basis einer der sich schnell entwickelnden Mikroelektronik rechen-technische Mittel zur Steigerung der Effektivität und Produktivität in der Produktion, der Forschung und Entwicklung sowie der Leitung und Planung einzusetzen. Die konventionelle Steuerungs- und Regelungstechnik, die z. Z. noch in großem Umfang in der Industrie eingesetzt wird, trägt in vielen Fällen nicht zur erforderlichen Verbesserung der Effektivität bei. Vor allem das Problem des Anfahrens von Prozessen sowie die Adaptierung von Reglerparametern zur Prozeßoptimierung verlangen in der konventionellen Regelungstechnik Aufwendungen, die oftmals in keinem akzeptablen Verhältnis zum erzielbaren Nutzen stehen.

Die entwickelte Mikroelektronik, die gegenwärtig für Mikroprozessoren bzw. Mikrorechner dem Anwender zur Verfügung steht, hat einen solchen Stand erreicht, der die notwendigen Automatisierungsleistungen auf der Basis von datenverarbeitenden Anlagen mit vertretbarem Verhältnis von Aufwand und Nutzen ermöglicht. Die Komponenten dazu sind nicht nur leistungsfähig genug, sondern bieten auch genügend Flexibilität bei der Auslegung von Automatisierungseinrichtungen, von Minimalsystemen bis zu Anlagen, die das Gebiet der Prozeßrechen-technik umfassen. Dabei können durch Programmierung die notwendigen Algorithmen implementiert werden. Erst diese Programmiermöglichkeiten gestatten es gegenüber konventionellen Anlagen, sich den bestehenden aktuellen Forderungen fast problemlos anzupassen. Daraus folgt, daß mit standardisierten Hardwaremodulen unterschiedlichste Aufgaben gelöst werden können. Der Entwicklungsaufwand für Steuerungen verschiebt sich somit in zunehmendem Maße in das Gebiet der Softwareentwicklung, wobei hier durch Bereitstellung geeigneter Entwicklungshilfsmittel (das Mikrorechnerentwicklungssystem MRES 20 sowie höhere Programmiersprachen) dem Anwender effiziente Mittel zur kurzfristigen Lösung seiner Problematik zur Verfügung stehen. Als Hardware sind die Mikrorechnersysteme K 1520 bzw. K 1600 vorhanden.

## 1.1. Entwicklung der Mikroprozessortechnik

Seit der Erfindung des Transistors im Jahre 1948 sind über den Spitzentransistor, Legierungstransistor und Drifttransistor wesentliche Fortschritte in der Beherrschung der Mikroelektronik erzielt worden. Durch die flächenhafte Anordnung in der Planartechnik wurden Grundlagen geschaffen, die ein Zusammenfassen mehrerer Transistoren und anderer Einheiten zu einem neuen Bauelement ermöglichten. Damit wurde eine neue Qualität erzielt. Gleichzeitig ermöglichten diese Bauelemente, die als integrierte Schaltungen

bezeichnet werden, ein ständiges Erhöhen der Gebrauchseigenschaften in der Rechen-technik. Da die integrierte Schaltungstechnik aufgrund der hohen Entwicklungskosten für jedes Bauelement eine hohe Produktionsstückzahl erfordert, können i. allg. nur universell verwendbare Schaltungskomplexe ökonomisch vorteilhaft integriert werden. Daher wurden zuerst logische Funktionen, wie UND-Gatter, ODER-Gatter und Negatoren, sowie sequentielle Elemente, z. B. Flipflops, entwickelt. Mit dem Erhöhen des Integrationsgrads, insbesondere durch Einführung der MOS-Technik, ergab sich die Möglichkeit der Herstellung ganzer Funktionskomplexe. Abgeleitet aus den Erfahrungen in der Anwendung der Prozeßrechen-technik und der Herstellung von Tischrechnern wurde als universelle Einheit die *zentrale Verarbeitungseinheit* (CPU) angesehen. Als charakteristische Beispiele seien hier die Entwicklungen der Typen 4004 und 8008 durch die Firma Intel genannt. Es handelt sich dabei um Mikroprozessoren mit 4-bit- bzw. 8-bit-Wortbreite in p-Kanal-Silicon-Gate-Technologie. Die 4-bit-Wortbreite ist vorwiegend gedacht für das Verarbeiten von Zifferinformation, wie sie bei Tischrechnern vorliegt, die 8-bit-Wortbreite für das Verarbeiten von Daten mit ASCII-Zeichen. Die ersten Bauelemente dieser Art waren im Zeitraum 1970/1971 international verfügbar. Für allgemeine Anwendungen hatte sich damals der Typ 8008 durchgesetzt. Der hohe Gesamtaufwand und der fehlende breite Kenntnisstand sowie die relativ geringe Entwicklungsunterstützung verhinderten jedoch den breiten Einsatz dieses Prozessorsystems.

Die weitere Entwicklung der Mikroelektronik bewirkte, daß nicht nur ein Mikroprozessor geschaffen wurde, der mit allgemein vorhandenen Komponenten zu einem Mikrorechner zu ergänzen ist, sondern ganze *Mikroprozessorfamilien*, die aus optimal aufeinander abgestimmten Schaltkreisen bestehen. Neben CPU, RAM und ROM wurde eine Reihe von programmierbaren Ein-/Ausgabe-Bausteinen entwickelt, deren Integrationsgrad teilweise höher als der der CPU ist. Der typische Bereich dafür sind 2000 bis 30000 Transistorfunktionen. Charakteristische Funktionen sind

- programmierbare parallele Ein-/Ausgabe
- programmierbare serielle Ein-/Ausgabe mit unterschiedlichen Übertragungsformaten und -geschwindigkeiten
- programmierbare Zähler und Zeitgeber
- programmierbarer direkter Speicherzugriff zur schnellen Übertragung großer Datenmengen
- Controller für Peripheriegeräte, wie Floppy-Disk, Datensichtgerät, Tastatur, Anzeige usw.

Die Entwicklung der Speicher führte zu höheren Geschwindigkeiten und Bitdichten sowie zur Reduzierung der Anzahl der Versorgungsspannungen, so daß die Bauelemente einfacher anwendbar wurden. Im Jahre 1974 wurde von Intel der Mikroprozessor 8080 vorgestellt, der in der Zwischenzeit international als der Standardtyp eines 8-bit-Mikroprozessorsystems angesehen wird. Da aufgrund des weltweiten Einsatzes eine große Menge von System- und Anwendersoftware für diesen Prozessor entwickelt wurde, sind viele Weiterentwicklungen und Zweitlieferanten in der Zwischenzeit aufgetreten, wie z. B.

- I8080 Intel
- I8085 A Intel
- SAB8080 Siemens
- SAB8085 Siemens
- TMS8080 Texas Instruments
- MPD8080 NEC
- MPD780 NEC

- Z80           Zilog
- MK3880   Mostek
- K580IK80   UdSSR
- NS16008   National Semiconductors
- Am9080    Advanced Micro Devices
- U880       VEB Funkwerk Erfurt
- NSC800    National Semiconductors.

Neben diesen Systemen hat sich in der Klasse der 8-bit-Mikroprozessoren noch das System 6800 der Firma Motorola einen großen Anwenderkreis erworben. Außer dem Grundtyp 6800 sind auch hier Weiterentwicklungen und Zweitlieferanten aufgetaucht z. B.

- 6802    Motorola
- 65XX   MOS-Technologie, Synertek, Rockwell.

Beide Gruppen zeichnen sich dadurch aus, daß seitens der Hersteller eine Vielzahl von Unterstützungen angeboten wird:

- Software zur Systementwicklung
- Softwareentwicklungssysteme
- Hardwareentwicklungssysteme
- OEM-Baugruppen
- Einführungslehrgänge
- höhere Programmiersprachen, wie PLM, PLZ, BASIC, PASCAL
- Anwenderklubs.

Diese Fakten sind oftmals ausschlaggebend bei der Entscheidung für ein bestimmtes Prozessorsystem. Die Leistungsfähigkeit ist sehr ähnlich, so daß meist die bestehende Aufgabe mit einem beliebigen Prozessor dieser Gruppen gelöst werden kann. Da die Entwicklung der erforderlichen Software einen beträchtlichen Kostenfaktor darstellt, wird man i. allg. bei dem System bleiben, für das man sich grundsätzlich entschieden hat.

## 1.2. Umsetzung von Problemen in Algorithmen

Bevor ein Produkt entwickelt wird, ist eine genaue und detaillierte Spezifikation der Eigenschaften des zu entwickelnden Produkts notwendige Voraussetzung für die Verwendung eines Mikrorechners. Das ist auch dann der Fall, wenn diskrete Logik eingesetzt werden soll. Bei Verwendung diskreter Logik, i. allg. TTL, ist lediglich eine Auswahl der entsprechenden Komponenten erforderlich. Die Lösung vieler Aufgaben kann jedoch in Verbindung mit einem Mikroprozessor sowohl hardwaremäßig als auch softwaremäßig erfolgen. Durch komplexe Interfaceschaltkreise wird der Prozessor stark von organisatorischen Aufgaben entlastet. Bei der Produktspezifikation hat man schon deshalb Probleme, weil eine zweckmäßige Aufteilung erst dann möglich ist, wenn die Leistungsdaten der Hardware mit der Software bekannt sind.

Zunächst wird nach der Spezifikation des Produkts ein vorläufiger Entwurf zur Konfiguration des Mikrorechners gemacht. Für bestimmte Probleme wird dann abgeschätzt, ob eine Hardwarelösung oder eine Softwarelösung zweckmäßiger ist (z. B. serielle Schnittstelle mit SIO oder Software). Wenn die Geschwindigkeit des Mikrorechners im *Echtzeitbetrieb* für bestimmte Probleme zu gering ist, können einzelne Aufgaben auch durch diskrete Logik hardwaremäßig gelöst werden. Ebenso ist der Einsatz von zugeordneten Koprozessoren möglich, z. B. schnelle Zusatzarithmetik.

Diskrete Schaltkreise besitzen aufgrund ihrer Konfiguration eine bestimmte Funktion. Die Gesamtheit der Schaltkreise erbringt dann eine komplexe Datenverarbeitung. Durch geeignetes Zerlegen der Problemstellung lassen sich standardisierbare Module finden, die, in geeigneter Weise kombiniert, die Programmstruktur bestimmen. Die *Programmerarbeitung* kann dadurch auf eine höhere Ebene verschoben werden, wenn eine Anzahl derartiger Module vorliegt, deren Eigenschaften bekannt sind. Durch Modifikation lassen sich diese auch noch der konkreten Aufgabenstellung anpassen, also optimieren. Es ist daher um so wichtiger, bei einem bestimmten Prozessortyp zu bleiben, um die einmal erarbeitete Basis auch weiterhin zu nutzen, wenn man sich vergegenwärtigt, daß die Softwareentwicklung einen großen Teil der gesamten Entwicklungskosten ausmacht. Der Prozessor U880 bietet für eine solche Verfahrensweise gute Voraussetzungen, da er sowohl in Minimalsystemen mit geringem Hardwareaufwand als auch in anspruchsvollen Systemen aufgrund seines komfortablen Befehlssatzes günstig einsetzbar ist.

Als wesentlicher Schritt bei der Programmentwicklung ist die Entscheidung anzusehen, in welcher Programmiersprache man arbeiten will. In vielen Fällen wird man von der *Assemblersprache* ausgehen, um eine optimale Speicherbelegung zu erreichen und die volle Leistungsfähigkeit des Mikrorechners auszuschöpfen. Bei Benutzung einer höheren Programmiersprache ist das meist nicht der Fall, da der erzeugte *Objektkode* bei weitem nicht so leistungsfähig ist. Durch die Weiterentwicklung der Compiler ist die Effektivität der Programmierung wesentlich verbessert worden. In zunehmendem Maße scheint ein durch Kombination aus *Assembler* und *Compiler* generiertes *Maschinenprogramm* den besten Lösungsweg darzustellen. Liegen keine geeigneten Compilerbefehle vor, so werden bestimmte Algorithmen in Assemblersprache programmiert. Zusätzlich werden dann noch optimierte Module eingesetzt. Ein verschiebbarer *Binder* (linker) berechnet nach Vorgabe des Programmeintrittspunkts die absoluten Adressen.

Zur Programmerstellung muß ein geeigneter Wirtsrechner vorhanden sein. Im Prinzip kann das jeder beliebige Rechner sein; dann ist jedoch *Crosssoftware* zum Betrieb erforderlich. Da der Mikroprozessor U880 sehr leistungsfähig ist, lassen sich auch komfortable Entwicklungshilfen auf seiner Basis herstellen. Der Prozessor kann dann auch das generierte Maschinenprogramm im Echtzeitbetrieb testen.

Dem Rechner muß in geeigneter Weise das zu übersetzende *Quellprogramm* eingegeben werden. Dazu ist eine Reihe von formalen Regeln zu beachten. Ein *Texteditor* unterstützt diesen Vorgang. Wenn das editierte Programm vorliegt, kann der Assembler es übersetzen, bevor es *gelinkt* wird.

Bei der Umsetzung einer Problemstellung in Algorithmen darf ein Gesichtspunkt nicht außer acht gelassen werden – die *Testbarkeit* von Hardware, Software und Gesamtgerät. Diesem Punkt muß besonders Rechnung getragen werden, da bei Nichtbeachtung wesentliche Zusatzkosten entstehen können. Ein scheinbar bezüglich Hardware und Software optimiertes Gerät läßt sich ohne geeignete Prüfstrategie nur mit großem Aufwand fertigen und reparieren. Daher ist es zweckmäßig, die Algorithmen unter Beachtung der Prüfbarkeit zu modifizieren und die Selbstprüfung (BITE-built-in test equipment) einzuführen.

### 1.3. Zuordnung von Anwendungsbereichen

Der Übergang von der Hardware zur Software ist der Übergang von der diskreten Logik zum Prozessor und wird wesentlich bestimmt durch den Aufbau und die Eigenschaften des Mikroprozessors und der Mikrorechner. In vielen Fällen wird der Mikroprozessor

noch als ein Austauschprodukt für konventionelle Logik betrachtet. Mit Einsatz des Mikrorechners wird jedoch eine neue Qualität erreicht, wenn bei der Programmierung nicht nur von einer einfachen Substitution ausgegangen wird, sondern der Adaptierung an den Prozeß mehr Raum gegeben wird.

In der Mikroprozessortechnik hat sich besonders die 8-bit-Struktur herauskristallisiert. Dafür existiert eine Reihe von Gründen. Die Entwicklung der Mikroprozessoren begann mit 4-bit-Strukturen, da sie in der Anfangsphase vorwiegend nur für Tischrechner zur Verarbeitung von Ziffern vorgesehen waren und weil mit Erhöhung der Wortbreite auch der Integrationsaufwand ansteigt. Auch Steuerungsaufgaben lassen sich mit 4-bit-Prozessoren günstig lösen, so daß diese auch heute noch ihre Berechtigung haben, jedoch jetzt meist in Form von Einchipmikrorechnern. Zur Verarbeitung alphanumerischer Zeichen, überwiegend in ASCII-Verschlüsselung, werden 8 bit benötigt. Dieses Format stimmt auch mit den meisten Lochstreifenlesern überein. Der Datenaustausch mit peripheren Schaltkreisen erfolgt überwiegend mit 8-bit-Wortbreite, auch bei 16-bit-Mikroprozessoren. Die komplexen Peripherieschaltkreise lassen sich daher auch für fast alle Systeme einsetzen.

Für allgemeine Anwendungen werden die 4-bit-Prozessoren in zunehmendem Maße abgelöst. Diese Prozessoren verfügen meist über einen Befehlsspeicher mit 8-bit-Wortbreite, so daß im Schreib-Lese-Speicher keine Befehle untergebracht werden können. Die Anwendung der standardisierten Peripherieschaltkreise bereitet ebenfalls Schwierigkeiten.

Durch die Forderung nach erhöhtem *Datendurchsatz* und die Beherrschung eines höheren Integrationsgrads wird international die Entwicklung in Richtung der 16-bit-Mikroprozessoren gehen, da man hier mit der Adressierung wesentlich variabler wird; denn in den Registern und Speicherstellen lassen sich auch die Adressen unterbringen, für die sich sowohl bei 8-bit-Prozessoren als auch für 16-bit-Prozessoren eine Breite von 16 bit eingebürgert hat.

Wenn umfangreichere Aufgaben zu lösen sind, bietet auch die Verteilung der Aufgaben auf unterschiedliche Prozessoren (distributed processing) eine günstige Variante. Dabei übernehmen die einzelnen Prozessoren bestimmte Arbeiten und werden von einem *Hauptprozessor* koordiniert. Man hat dadurch die Möglichkeit, daß die Teile der Software von verschiedenen Programmierern generiert werden können und auch überschaubarer bleiben als ein größeres Programm. Außerdem werden die Prüfmöglichkeiten und auch die Geschwindigkeit bzw. der Datendurchsatz verbessert, und man kann bei einem Standardprozessor bleiben.

Die vorstehenden Ausführungen zeigten, daß der Produktdefinition in der ersten Entwicklungsphase eine hervorragende Bedeutung beizumessen ist, da in dieser Phase bereits wesentlich die Entwicklungs- und Fertigungskosten einschließlich der Zuverlässigkeit bestimmt werden. Daher läßt sich auch keine allgemeingültige Lösung angeben; denn sie wird entscheidend vom Anwendungsbereich mitbestimmt. Da die Hardware in der Tendenz immer kostengünstiger zur Verfügung stehen wird, ist besonderer Wert zu legen auf komfortable Entwicklungssysteme sowie auf höhere Programmiersprachen und variable Testeinrichtungen für Hardware und Software.

## 2. Vergleich der ersten und der zweiten Leistungsklasse von Mikroprozessoren

Der Vergleich beider Leistungsklassen soll deutlich machen, wie sich in der DDR die Entwicklung der Mikroprozessoren vollzogen hat. Als erster Mikroprozessor der DDR wurde vom VEB Funkwerk Erfurt der Schaltkreis U808D in p-Kanal-Silizium-Gate-Technologie entwickelt. Aufgrund des erforderlichen Aufwands zur Realisierung eines funktionsfähigen Mikrorechners konnte der Einsatz jedoch nur dort erfolgen, wo der hohe Aufwand trotzdem gerechtfertigt war.

Parallel mit dem Übergang von der ersten zur zweiten Leistungsklasse ist der Übergang von der p-Kanal- zur n-Kanal-Technologie zu sehen. Neben einer Vergrößerung der Verarbeitungsgeschwindigkeit wurden damit auch die volle TTL-Kompatibilität und die Reduzierung der Betriebsspannung auf den TTL-Wert von +5 V erreicht. Weiterhin ist charakteristisch, daß in der zweiten Leistungsklasse komplexe periphere Schaltkreise zur Verfügung stehen.

Aus all den Fakten ist ersichtlich, daß die erste Leistungsklasse für Neuentwicklungen keine Bedeutung mehr hat, so daß hier nur der Vollständigkeit halber die Grundelemente kurz dargestellt werden.

### 2.1. Beschreibung des Systems U808

Das Schaltkreissystem der ersten Leistungsklasse umfaßt die MOS-LSI-Elemente

- |          |                           |       |
|----------|---------------------------|-------|
| ● U808   | 8-bit-CPU                 | pSGT  |
| ● U501   | 2-Kbit-(256 bit × 8)-ROM  | MNOS  |
| ● U551   | 2-Kbit-PROM               | pSGT  |
| ● CM8001 | 256-bit-(256 bit × 1)-RAM | pSGT. |

Alle weiteren Baugruppen eines Mikrorechners dieser Leistungsklasse müssen mit Standard-TTL-Bauelementen realisiert werden. Spezielle **Ein-/Ausgabe-** und Interfaceschaltkreise stehen nicht zur Verfügung. Die CPU U808 wird durch folgende Merkmale charakterisiert:

- 8-bit-Einchip-CPU in pSGT
- Basisbefehlssatz mit 48 Befehlen
- maximale Taktfrequenz 500 kHz
- Befehlsausführung in 12 ... 44  $\mu$ s
- typische Befehlsausführungszeit 20  $\mu$ s
- eingangsseitig TTL-kompatibel
- ausgangsseitig low-power-TTL-kompatibel
- 16 Kbyte direkt adressierbare Speicherkapazität durch internen 14-bit-Adreßbus
- interner achtstufiger 14-bit-Adreßstapelspeicher (Stack)
- sieben Datenregister

- Programmunterbrechungsmöglichkeit durch Interrupteingang
- Synchronisationseingang (READY) für Anschluß langsamer Speicher
- 18poliges DIL-Gehäuse.

Im Bild 2.1.1 ist das Blockschaftbild der IS U808 dargestellt. Bild 2.1.2 zeigt die schematische Anschlußbelegung der CPU.

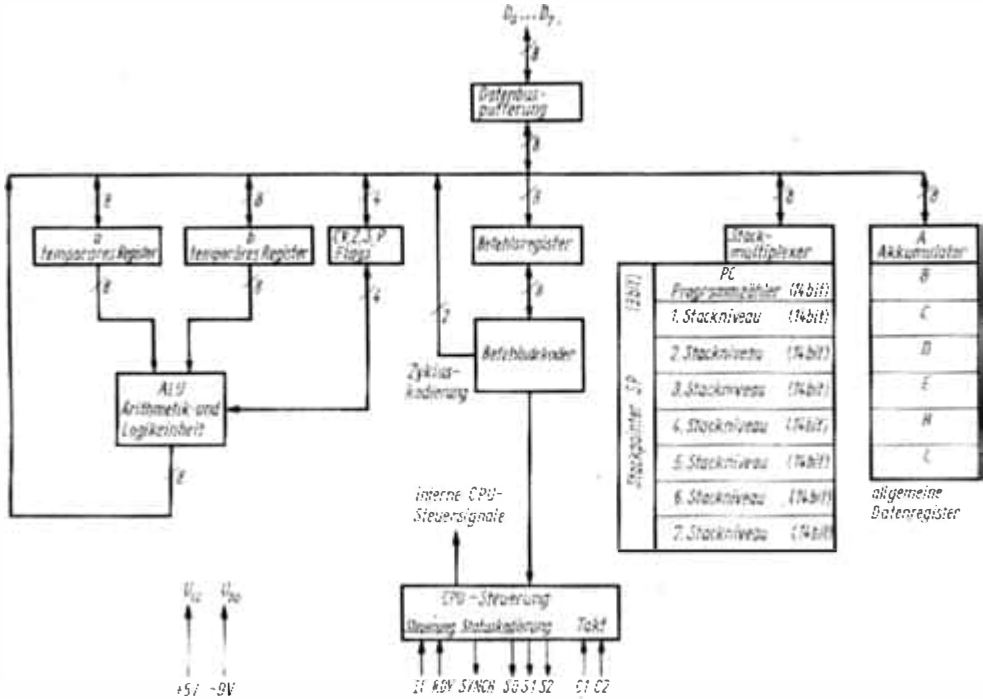


Bild 2.1.1. Blockschaftbild der CPU U808

Die beiden Bits der Zyklus Kodierung werden als  $A_{14}$  und  $A_{15}$  während des Zustands T2 (s. Bild 2.1.2) über den Multiplexbus ausgegeben. Sie haben folgende Bedeutung:

$A_{15}$	$A_{14}$	Bedeutung
0	0	PCI, Befehlslesezyklus (op-code fetch)
0	1	PCC, INPUT- oder OUTPUT-Zyklus
1	0	PCR, Speicherlesezyklus
1	1	PCW, Speicherschreibzyklus

Der 8-bit-bidirektionale Datenbus stellt einen Multiplexbus dar. Über ihn werden sowohl Adressen, Daten als auch Steuerinformationen (Zyklus Kodierung) übertragen. Somit werden allein zwei CPU-Zustände zur Aussendung der 14-bit-Adresse benötigt. Bei einem typischen Befehl (z.B. einem 8-bit-Additionsbefehl) folgt ein Zustand zum Lesen des adressierten Op-Kodes sowie zwei weitere Taktzustände, die der CPU-internen Befehlsausführung dienen.

Da die CPU einen nichtüberlappenden Zweiphasentakt benötigt, dauert ein derartiger CPU-Taktzustand minimal  $4 \mu s$ . Über die Statusausgänge, die mittels SYNC mit dem Systemtakt synchronisiert werden, erfolgt die Zustandskodierung für die Steuerlogik des Mikrorechners.

Der Befehlsspeicher kann aus den pinkompatiblen ROM und PROM aufgebaut werden. Die Zugriffszeit von  $1 \mu s$  reicht für die Anwendung mit der CPU U808 aus, ohne daß Wartezustände eingefügt werden müssen. Die geringe Kapazität von 256 Wörtern mit je 8 bit erfordert in fast allen Fällen, daß mehrere ROM eingesetzt werden müssen. Die ROM

verfügen über Chipauswahleingänge  $\overline{CS}$  und Tristateausgangsstufen, so daß die ROM parallelgeschaltet werden können; ein Dekoder übernimmt die Chipauswahl.

Die RAM CM8001 haben 256 Wörter mit je 1 bit. Daher müssen acht IS in den Adressen, Betriebsspannungen, Schreibsignal und  $\overline{CS}$  parallelgeschaltet werden. Die Datenein- und -ausgänge werden sinngemäß auf die acht Datenleitungen verteilt. Die gesamte Anordnung ergibt ein Äquivalent, bezogen auf den Speicherbereich für ein ROM/PROM. Die Zugriffszeit von  $1,5 \mu s$  reicht ebenfalls für die CPU U808 D bei 500 kHz Taktfrequenz aus.

Aus derzeitiger Sicht hat daher das gesamte System der ersten Leistungsklasse keine Bedeutung mehr und wird nur noch in bereits produzierte Geräte eingesetzt.

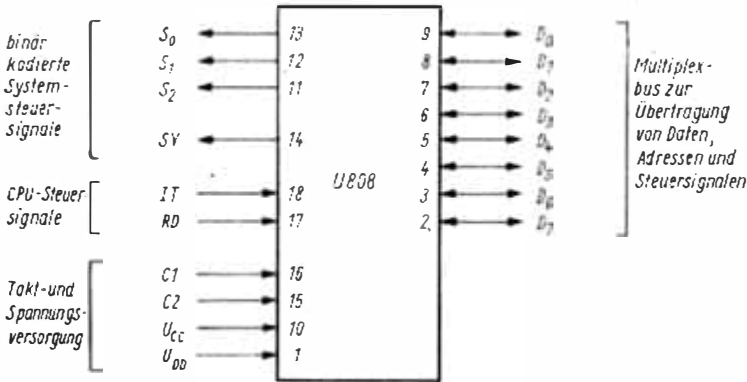


Bild 2.1.2. Schematische Anschlußbelegung der IS U808

- S0 ... S2 binär kodierte Statussignale mit folgender Bedeutung:
- |          |   |
|----------|---|
| S0 S1 S2 | CPU-Zustand   |
| 0 1 0    | T1, Aussenden der niedrigerwertigen Adresse ( $A_0 \dots A_7$ ) über den Datenbus                             |
| 0 1 1    | T11, alternativ zu T1 bei erkannter Interruptforderung, kein Inkrementieren von PC                            |
| 0 0 1    | T2, Aussenden der höherwertigen Adresse ( $A_8 \dots A_{13}$ ) sowie der Zykluskodierung ( $A_{14}, A_{15}$ ) |
| 0 0 0    | T3W, Waitzustand, wenn RD=L   |
| 1 0 0    | T3, Datenverkehr über den Datenbus entsprechend der Zykluskodierung   |
| 1 1 0    | T3S, Haltzustand  |
| 1 1 1    | T4, Befehlsausführung in der CPU  |
| 1 1 0    | T5, Befehlsausführung in der CPU  |
- SY Synchronisationssignal zur Kennzeichnung der Taktzustände des Zweiphasentakts  
 IT Interrupteingang, H-aktiv  
 RD Readyeingang, zur Einfügung von Waitzuständen, H-aktiv  
 C1, C2 Zweiphasentaktsignale  
 $U_{CC}, U_{DD}$  Spannungsversorgung (+ 5 V, -9 V)  
 $D_0 \dots D_7$  Multiplexbus

## 2.2. Übersicht über das System U880

Das System der zweiten Leistungsklasse mit dem Mikroprozessor U880 als bestimmendem MOS-LSI-Schaltkreis erfährt durch die Bereitstellung komplexer Peripherieschaltkreise eine wesentliche Hardwareunterstützung. Zur Schaltkreisfamilie der zweiten Leistungsklasse gehören folgende MOS-LSI-Elemente:

- U880 8-bit-Einchip-CPU, 158 Basisbefehle, 2,5-MHz-Systemtaktfrequenz, typische Befehlsabarbeitung  $1,6 \mu s$
- U855 PIO, parallele Ein-/Ausgabe-Einheit
- U856 SIO, serielle Ein-/Ausgabe-Einheit
- U857 CTC, Zähler/Zeitgeber

- U505 8-Kbit-(1 Kbit  $\times$  8)-ROM, Zugriffszeit 450 ns
- U555 8-Kbit-EPROM, pinkompatibel zu U505
- U202 1-Kbit-(1 Kbit  $\times$  1)-RAM, Zugriffszeit 400 ns.

Die Geschwindigkeiten der Bauelemente sind aufeinander abgestimmt, so daß ebenfalls keine Wartezustände eingefügt werden müssen.

Man erkennt, daß das Schaltkreissystem der zweiten Leistungsklasse wesentlich umfangreicher ist und eine höhere Verarbeitungsgeschwindigkeit aufweist. Durch das Vorhandensein komplexer Peripherieschaltkreise wird die Anzahl der TTL-IS minimiert. Bis auf den EPROM benötigen alle IS als Betriebsspannung nur +5 V. Die Pegel sind bis auf den Takt TTL-kompatibel. Da jedoch nur ein zeitsymmetrischer Einphasentakt mit 5 V eingesetzt wird, ist auch dessen Erzeugung wesentlich einfacher als im Fall der IS U808

### 2.3. Qualitative Gegenüberstellung

Neben der beim System U880 wesentlich besseren Entwicklungsunterstützung durch Software (verbesserte Programmiersprachen, Standardprogramme) und Hardware (leistungsfähige Entwicklungssysteme, Lernsysteme) sind folgende Punkte im direkten Vergleich der Schaltkreisfamilien charakteristisch:

- Der Befehlssatz der CPU U880 umfaßt den mehr als dreifachen Basisbefehlssatz; hinzu kommen effektivere Adressierungsarten und repetierend wirkende Befehle beim U880.
- Die typische Befehlsabarbeitungszeit beträgt bei der IS U880 nur etwa 8 % der Bearbeitungszeit des U808.
- Die parallele Busstruktur der CPU U880 (im Gegensatz zum Multiplexbus des U808) verringert den Hardwareaufwand bei der Einbeziehung von Standardelementen.
- Die ROM- und RAM-Speicherelemente der zweiten Leistungsklasse weisen die vierfache Kapazität auf.
- Eine leistungsfähige hardwaremäßige Peripherieunterstützung ist nur für die Schaltkreisfamilie U880 verfügbar.
- Die CPU U880 nimmt einen automatischen Refresh dynamischer Speicherelemente vor.

Die zweite Leistungsklasse läßt daher einen wesentlich verringerten Hardwareaufwand für die gleiche Problemlösung zu. Des weiteren wird die Programmlänge aufgrund des besseren Befehlssatzes kürzer. Die vielfache Geschwindigkeit ergibt auch Einsatzmöglichkeiten, wo sonst Prozeßrechner erforderlich wären. Qualitativ ist das System U880 dem System U808 in Aufwand (Kosten) und Leistung so überlegen, daß es in der Breite, von Minimalsystemen für einfachste Probleme bis zu Mehrrechnersystemen, effektiv geeignet ist.

## 3. Beschreibung der Elemente des Systems U880

### 3.1. Zentrale Verarbeitungseinheit

#### 3.1.1. Einführung

Die CPU eines Mikroprozessorsystems bestimmt mit ihren Eigenschaften (Wortbreite, Befehlssatz, Befehlsausführungsdauer) in weiten Grenzen die Leistungsfähigkeit eines mit ihr realisierten Mikrorechners bzw. einer Steuerung.

Das Mikroprozessorsystem U880 zeichnet sich aus durch eine hohe Flexibilität sowohl bezüglich der Hardware als auch der Software. Für den Bereich der Hardware gilt, daß das gesamte Typenspektrum der Bauelemente hinsichtlich Signalpegel und Versorgungsspannung vollständig TTL-kompatibel ist; lediglich für die Taktversorgung wird ein zeitsymmetrischer 5-V-Pegel benötigt. Darüber hinaus ist das System auf Standardbauelemente (Festwertspeicher, statische und dynamische Schreib-Lese-Speicher, Dekoder, Treiber u. a.) orientiert. Aufgrund des relativ geringen Aufwands ist das System auch für Anwendungen geeignet, wo es auf minimalen Bauelementeaufwand ankommt. Die Geschwindigkeit und die Softwareeigenschaften des Mikroprozessors U880 lassen einen Einsatz zu, der bis an das untere Ende des typischen Anwendungsgebiets von Minirechnern heranreicht.

Die IS U880 wird im wesentlichen durch folgende Punkte charakterisiert:

- Ein-Chip-Mikroprozessor in n-Kanal-Silicon-Gate-Technologie
- Befehlssatz mit 158 Basisbefehlen einschließlich 16-, 8-, 4- und Einzelbitbefehle
- umfangreiche Adressierungsarten, u. a. indizierte, relative und Bitadressierung
- 17 interne Register, Realisierung eines Tauschregistersatzes
- drei schnelle Interruptbehandlungsarten sowie ein weiterer, nichtmaskierbarer Interrupt
- direkter Anschluß von statischen und dynamischen Standardhalbleiterspeichern möglich
- interne Refreshlogik
- typische Befehlsausführungsdauer  $1,6 \mu\text{s}$   
(bei einer Taktfrequenz von 2,5 MHz)
- alle Anschlüsse, bis auf den Takt, voll TTL-kompatibel
- einfacher, zeitsymmetrischer Takt mit 5-V-Pegel.

#### 3.1.2. Struktureller Aufbau

Das Blockschaltbild der internen Architektur der IS U880 ist im Bild 3.1.1 gezeigt. Dieses Bild zeigt die Hauptelemente der CPU, auf die in den nachfolgenden Ausführungen Bezug genommen wird.

3.1.2.1. Register

Die CPU U880 ist vorwiegend für die Verarbeitung von Wörtern mit 8 bit Breite vorgesehen. Da sich mit 8 bit nur 256 unterschiedliche Speicherstellen adressieren lassen, wurde für den Adreßbus die doppelte Breite, also 2 byte oder 16 bit festgelegt, um so 64 K Speicherstellen adressieren zu können. Mit diesen Vorbemerkungen läßt sich der Registeraufbau leichter erklären; er ist im Bild 3.1.2 dargestellt. Der Prozessor kann für Datenmanipulationen auf insgesamt 208 bit eines internen Schreib-Lese-Speichers zurückgreifen. Dieser Speicher besteht aus achtzehn 8-bit-Registern und vier 16-bit-Registern. Die Register stellen ein statisches RAM dar.

**Hauptregistersatz.** Der Hauptregistersatz besteht aus acht 8-bit-Registern; von diesen sind die Register A und F für spezielle Aufgaben vorgesehen. Die restlichen sechs Register werden für allgemeine Verwendung genutzt.

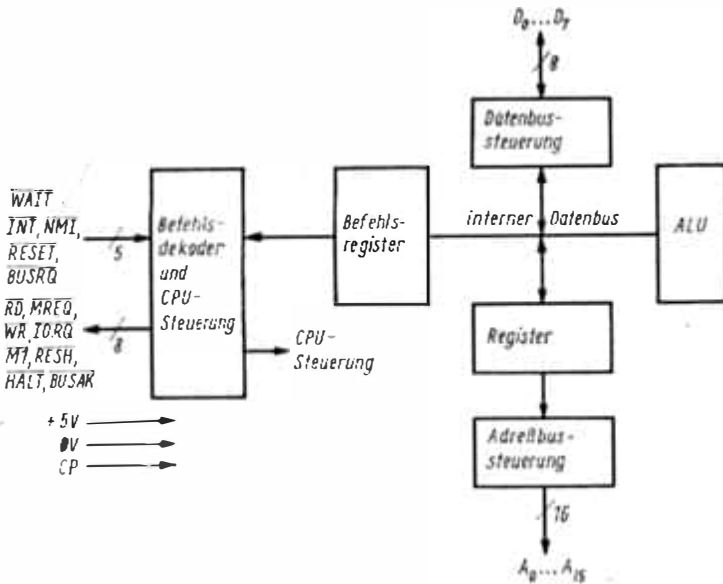


Bild 3.1.1  
Blockschaltbild der CPU

Hauptregistersatz		Tauschregistersatz	
Akkumulator <b>A</b>	Flags <b>F</b>	Akkumulator <b>A'</b>	Flags <b>F'</b>
<b>B</b>	<b>C</b>	<b>B'</b>	<b>C'</b>
<b>D</b>	<b>E</b>	<b>D'</b>	<b>E'</b>
<b>H</b>	<b>L</b>	<b>H'</b>	<b>L'</b>

} allgemeine Register

Interrupt-register <b>I</b>	Refresh-register <b>R</b>
Indexregister <b>IX</b>	
Indexregister <b>IY</b>	
Stapelzeiger <b>SP</b>	
Programanzähler <b>PC</b>	

} Spezialregister

Bild 3.1.2  
Registeranordnung

Das Register A, der Akkumulator, enthält das Ergebnis von 8 bit arithmetischen oder logischen Operationen, wobei das Flagregister F die spezifischen Bedingungen für 8-bit- oder 16-bit-Operationen anzeigt (z. B. ob das Ergebnis einer Operation positiv oder negativ, Null oder ungleich Null ist).

Die weiteren sechs 8-bit-Register lassen sich auch in den Paaren BC, DE und HL als 16-bit-Register verwenden. Auf diese Registerpaare können sich dann auch die 16-bit-Operationen beziehen, da der Akkumulator zur Aufnahme des Ergebnisses zu klein ist.

Die Allzweckregister werden vom Programmierer für einen weiten Anwendungsbereich benutzt. Sie vereinfachen auch die Programmierung, da nicht immer erst auf den externen Speicher über kompliziertere Befehle zurückgegriffen werden muß. In sehr kleinen Systemen kann man auf einen externen RAM verzichten.

**Tauschregistersatz.** Neben dem Hauptregistersatz ist der Tauschregistersatz vorhanden, der identisch zu diesem aufgebaut ist. Der Programmierer hat die Möglichkeit, jeweils den Akkumulator und das zugehörige Flagregister sowie mit einem weiteren Befehl die Allzweckregister auszutauschen. Die Tauschregister werden jeweils mit einem Strich gekennzeichnet (z. B. AF'). In Systemen, in denen eine schnelle Interruptverarbeitung erforderlich ist, kann ein Satz der Allzweckregister und des Akkumulators sowie des Flagregisters für die Durchführung dieser schnellen Interruptroutine reserviert werden. Es braucht nur ein einfacher Austauschbefehl zur Umschaltung zwischen den Routinen ausgeführt zu werden. Dadurch wird beträchtlich Zeit in der Bedienung eines Interrupts eingespart, zumal das Retten und Wiederholen der Registerinhalte in dem externen Stack beim Interrupt und innerhalb der Subroutine entfallen können.

**Spezielle Register.** *Programmzähler (program counter, PC).* Der Programmzähler enthält die 16-bit-Adresse der laufenden Instruktion, die vom Speicher als Befehlskode geholt wurde. Die Bezeichnung Programmzähler ist eigentlich ungenau, da dieser Zähler nicht die Programmschritte zählt, sondern der PC wird automatisch inkrementiert, nachdem sein Inhalt auf den Adreßbus geladen wurde. Da Befehle mit mehreren Bytes des Operationskodes vorhanden sind, wird teilweise bei einzelnen Befehlen mehrfach inkrementiert. Wenn ein Sprung im Programm auftritt, wird der neue Adreßinhalt automatisch in den PC übertragen und das Inkrementieren überschrieben.

*Stapelzeiger (stack pointer, SP).* Der Stackpointer enthält die 16-bit-Adresse der aktuellen Spitze des Stapels, der sich irgendwo frei wählbar in dem externen RAM-Speicher befindet. Der externe Stapelspeicher ist als last-in-first-out (LIFO) organisiert. Das bedeutet, daß die zuletzt eingegebenen Wörter als erste wieder ausgegeben werden. Die Daten können von speziellen CPU-Registern auf den Stapel abgelegt oder vom Stapel in spezifische Register eingelesen werden. Dazu werden die entsprechenden PUSH- und POP-Befehle ausgeführt. Der Stapel ermöglicht die einfache Ausführung von Mehrerebeneninterrupts, unbegrenzte Unterprogrammverzweigung und die Vereinfachung vieler Arten von Datenmanipulationen.

*Indexregister (IX und IY).* Diese 16-bit-Register werden bei der indizierten Adressierung und bei 16-bit-Datenmanipulationen verwendet. In der indizierten Adressierungsmode wird ein Indexregister als Basis benutzt, das einen Pointer für einen Speicherbereich darstellt, in dem Daten gespeichert oder von dem Daten geholt werden sollen. Diese CPU-Befehle enthalten ein zusätzliches Byte, das die Distanz von der Basis bestimmt. Der Wert dieser Distanz ist als Zweierkomplement spezifiziert. Damit ist die indizierte Adressierung besonders für Tabellenarbeiten und für die Adressierung in verschiebbaren Programmteilen geeignet.

*Interruptvektorregister (I).* Dieses CPU-Register stellt in der Interruptbetriebsart IM2 des Prozessors U880 die höherwertigen 8 bit des zur Adressierung der Interruptbedienroutine

benötigten Interruptpointers bereit. Die niederwertigeren Bits dieses Pointers werden hierbei direkt vom interruptanfordernden peripheren Element geliefert. Somit können schnelle und leistungsfähige Interruptreaktionen erreicht werden (s. auch Abschn. 4.).

**Refreshregister (R).** Dynamische Schreib-Lese-Speicher benötigen zum Datenerhalt ein ständiges Auffrischen der in Kapazitäten gespeicherten Information durch Lesen aller Zeilen der Speicher in einem bestimmten Zeitabstand (meist 2 ms). Dadurch können dynamische Speicher ebenso leicht eingesetzt werden wie statische. Das R-Register ist 7 bit breit und wird nach jedem Holen einer Instruktion inkrementiert. Das R-Register kann daher als Zähler für M1-Zyklen ausgenutzt werden. Die Daten des Refreshzählers werden als unterster Teil der Adresse zusammen mit einem Refreshsteuersignal ( $\overline{\text{RFSH}}$ ) ausgesendet. Das geschieht im M1-Zyklus, während die CPU die eingeholte Instruktion dekodiert und ausführt. Diese Refreshmode ist vollständig für den Programmierer transparent und verringert nicht die Systemgeschwindigkeit. Der Inhalt des Registers kann gelesen und durch Schreiben gezielt gesetzt werden.

### 3.1.2.2. Arithmetik- und Logikeinheit (ALU)

Die arithmetischen und logischen Instruktionen mit einer Breite von 8 bit werden von der CPU in der ALU ausgeführt. Innerhalb der CPU kommuniziert die ALU mit den Registern und dem externen Datenbus über den internen CPU-Datenbus. Die ALU kann nachfolgende Funktionsarten ausführen:

- Addition
- Subtraktion
- Inkrementieren
- Dekrementieren
- Bitsetzen
- Bitrücksetzen
- Bittesten
- logisches AND
- logisches OR
- logisches XOR (exklusives OR)
- links und rechts Schieben (arithmetisch und logisch)
- links und rechts Rotieren (bit- und digitweise).

### 3.1.2.3. Befehlsdekoder und CPU-Steuerung

Jeder Befehl wird vom Speicher geholt, im Befehlsregister zwischengespeichert und im Dekoder dekodiert. Die CPU-Steuerung führt die gewünschte Funktion aus und erzeugt die erforderlichen Steuersignale. Damit werden das zeitrichtige Lesen und Schreiben der Daten von oder nach den Registern ermöglicht. Weiterhin wird die ALU gesteuert, und die extern benötigten Steuersignale werden geliefert.

### 3.1.2.4. Daten- und Adreßbussteuerung

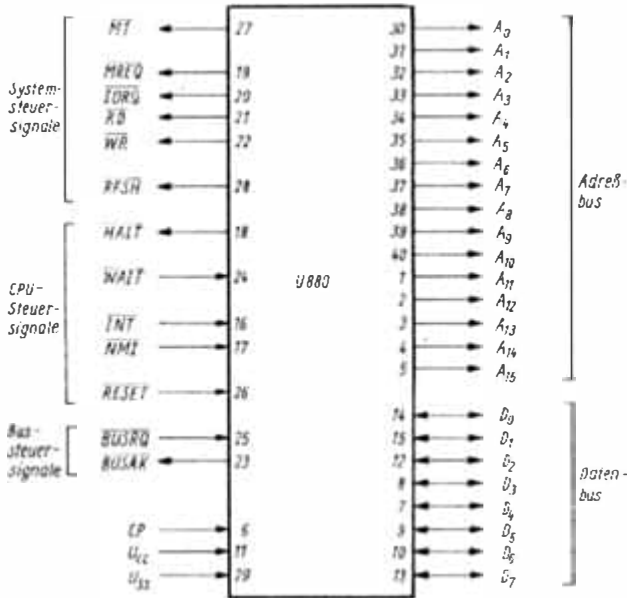
Die Daten- und Adreßbussteuerung ermöglicht die Koordinierung des bidirektionalen Informationsaustauschs. Dadurch wird vermieden, daß gleichzeitig mehrere Signalquellen um eine Informationsleitung kämpfen und u. U. Bauelementedefekte auftreten.

## 3.1.3. Erläuterung der Anschlußbelegung

Die CPU U880 wird in einem 40poligen DIL-Gehäuse geliefert. Im Bild 3.1.3 ist die schematische Anschlußbelegung gezeigt, auf die in den nachfolgenden Ausführungen Bezug genommen wird.

**A<sub>0</sub> ... A<sub>15</sub> Address Bus (Ausgänge, tristate)**

Der 16-bit-Adreßbus der CPU stellt die Adressen für den Datenaustausch mit dem Speicher und den Ein-/Ausgabe-Einheiten bereit. Darüber hinaus wird über den Adreßbus während der Refreshzyklen die Refreshadresse für dynamische RAM ausgesendet (A<sub>0</sub> ... A<sub>6</sub>). Das Tristateverhalten der Adreßlinien gestattet die Busübernahme durch andere Einheiten im DMA-Betrieb. A<sub>0</sub> ist das niederwertigste Adreßbit.



**Bild 3.1.3**  
Schematische Anschlußbelegung  
der CPU

**D<sub>0</sub> ... D<sub>7</sub> Data-Bus (bidirektional, tristate)**

Der 8-bit-Datenbus der CPU dient dem eigentlichen Informationsaustausch zwischen Prozessor einerseits sowie dem Speicher und den Ein-/Ausgabe-Einheiten andererseits. Das Tristateverhalten ermöglicht ebenfalls eine Busübernahme im DMA-Betrieb. D<sub>0</sub> ist das niederwertigste Datenbit.

**CP Clock Pulse (Eingang, 5-V-Pegel)**

Der Systemtakt dient zur Synchronisation der meisten internen Abläufe der CPU U880. Er ist ein zeitsymmetrischer Einphasentakt.

**RESET Reset (Eingang, L-aktiv)**

Das RESET-Signal hat die Aufgabe, den Prozessor in einen Anfangszustand zu bringen. Es erfolgt ein Rücksetzen und eine Anfangsinitialisierung der CPU. Hierzu muß das Signal mindestens drei Taktzustände aktiv sein. Im einzelnen werden folgende Funktionen ausgeführt:

- Rücksetzen des Programmzählers auf 0000H
- Rücksetzen der Interruptfreigabefllops
- Rücksetzen des Registers I auf 00H
- Rücksetzen des Registers R auf 00H
- Setzen der Interruptbetriebsart IM0.

Während des Rücksetzens gehen der Adreßbus und der Datenbus in den hochohmigen sowie die Steuersignalausgänge in ihre inaktiven Zustände. Es wird somit ebenfalls kein Refresh durchgeführt.

**WAIT** Wait (Eingang, L-aktiv)

Dieses Signal gestattet die Einfügung von Wartezuständen in die Lese- und Schreibzyklen der CPU mit dem Speicher und den Ein-/Ausgabe-Einheiten. Somit wird eine Synchronisation der CPU U880 mit langsamen Einheiten ermöglicht.

**MI** Machine-Cycle 1 (Ausgang, L-aktiv)

Dieses Steuersignal der CPU dient zur Kennzeichnung der Befehlsholzyklen. Darüber hinaus nimmt es in Verbindung mit dem aktiven Zustand des  $\overline{\text{IORQ}}$ -Signals die Interruptquittierung vor.

**MREQ** Memory Request (Ausgang, tristate, L-aktiv)

Das  $\overline{\text{MREQ}}$ -Signal zeigt eine Speicheroperation der CPU (Lesen, Schreiben oder Befehlsholen) an.

**IORQ** Input/Output Request (Ausgang, tristate, L-aktiv)

Dieses Steuersignal kennzeichnet den Datenverkehr zwischen Prozessor und dem durch den Adreßbus (meist  $A_0 \dots A_7$ ) adressierten Ein-/Ausgabe-Gerät. In Verbindung mit  $\overline{\text{MI}}$  nimmt es die Interruptquittierung vor.

**RD** Read (Ausgang, tristate, L-aktiv)

Das  $\overline{\text{RD}}$ -Signal zeigt an, daß eine Leseoperation mit dem Speicher oder der Peripherie ausgeführt wird.

**WR** Write (Ausgang, tristate, L-aktiv)

Das  $\overline{\text{WR}}$ -Steuersignal der CPU ist aktiv, wenn der Datenbus gültige Daten für eine Schreiboperation enthält.

Die letztgenannten vier Steuersignale dienen besonders zum direkten Datentransport. Sie weisen ein Tristateverhalten auf, damit im DMA-Betrieb andere Einheiten den Datenverkehr organisieren können.

**RFSH** Refresh (Ausgang, L-aktiv)

Dieses Signal kennzeichnet den Refreshzyklus der CPU U880. In Verbindung mit dem Steuersignal  $\overline{\text{MREQ}}$  können dynamische Speicherelemente aufgefrischt werden.

**HALT** Halt (Ausgang, L-aktiv)

Der aktive Zustand dieses Ausgangs zeigt an, daß der CPU-Befehl HALT ausgeführt worden ist und der Prozessor sich im Haltzustand befindet. Dieser Zustand kann nur durch die Ausführung eines Interrupts bzw. durch  $\overline{\text{RESET}}$  verlassen werden. Die CPU führt automatisch NOP-Befehle aus, um die Refreshfunktion aufrechtzuerhalten.

**INT** Interrupt Request (Eingang, L-aktiv)

Über diesen CPU-Steuereingang erfolgt die Anmeldung von maskierbaren Interrupts. Bei Interruptfreigabe der CPU reagiert der Prozessor mit dem Einschleusen einer Bedienroutine in die Befehlsabarbeitung.

**NMI** Non Maskable Interrupt Request (Eingang, L-aktiv)

Der NMI-Eingang dient zur Anmeldung von nicht maskierbaren Interruptforderungen. Die vom Prozessor in den Befehlsablauf eingefügte Serviceroutine wird i. allg. für Notfunktionen im Mikrorechner benutzt.

**BUSRQ** Bus Request (Eingang, L-aktiv)

Dieses Steuersignal dient zur Anforderung des CPU-Bussystems. Es wird von einer Einheit zur Anmeldung der DMA-Betriebsart geliefert.

**BUSAK** Bus Acknowledge (Ausgang, L-aktiv)

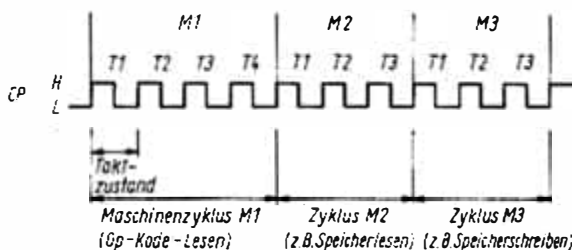
Mit diesem Steuersignal quittiert die CPU U880 die Anmeldung einer Busanforderung an BUSRQ. Während des aktiven Zustands von BUSAK befinden sich der Datenbus, der Adreßbus und die Steuersignale RD, WR, MREQ, IORQ im hochohmigen Zustand. Die anderen Ausgangssignale sind inaktiv. Der Datenverkehr kann somit von einer DMA-Einheit gesteuert werden.

**3.1.4. Zeitverhalten**

Das Zeitverhalten der CPU U880 wird im wesentlichen durch den Systemtakt bestimmt. Die CPU tauscht in Abarbeitung der Befehle ihre Daten über den bidirektionalen Datenbus aus. Ebenso holt die CPU ihre Befehle durch Adressierung des Speichers über den Datenbus ein. Folgende Grundfunktionen werden ausgeführt:

- Befehlskodelesen
- Speicherlesen
- Speicherschreiben
- I/O-Lesen
- I/O-Schreiben
- Interruptbestätigung
- CPU-interne Operation.

Alle Instruktionen werden auf der Basis der o. g. Grundfunktionen ausgeführt und sind teilweise Zusammensetzungen mehrerer Grundfunktionen. Die Gesamtzahl der Takte einer Grundfunktion bilden einen *Maschinenzyklus*; ein oder mehrere Maschinenzyklen bilden einen *Befehls-* oder *Instruktionszyklus*. Grundsätzlich beginnt jeder Instruktionszyklus mit dem Lesen eines Befehlskodes (Op-Kode). Dieser *Maschinenzyklus* wird als *M1-Zyklus* bezeichnet, und von der CPU wird ein Steuersignal M1 ausgesendet. Das M1-Signal wird im System als Synchronisationssignal für Instruktionszyklen verwendet. Da in jedem Instruktionszyklus mindestens ein *M1*-Zyklus enthalten ist, wird ein kontinuierlicher Refresh gesichert. Die Taktperioden werden auch als *T-Zustände* bezeichnet.

**Bild 3.1.4****Aufbau eines Befehlszyklus (Beispiel)**

**M1, M2, M3** Maschinenzyklen der CPU  
**CP** Systemtakt (T1, T2, T3, T4)  
**T1, T2, T3, T4** Taktzustände der CPU

Jede Grundfunktion besteht aus drei bis sechs T-Zuständen, die M1-Zyklen aus vier bis sechs.

Bild 3.1.4 zeigt als Beispiel einen Instruktionszyklus, der aus drei Maschinenzyklen besteht. Bisher wurde vorausgesetzt, daß keine WAIT-Zustände zur Synchronisation der externen Signale eingeschoben wurden, die dann natürlich die Maschinenzyklen entsprechend verlängern.

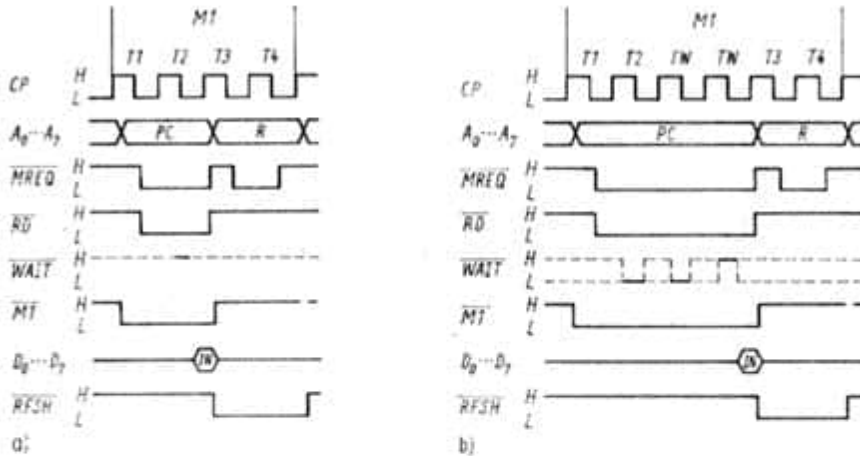


Bild 3.1.5. Befehlscodelesezyklus (M1-Zyklus)

a) ohne Waitzustände

b) mit Einfügung von zwei Waitzuständen

CP Systemtakt (T1, T2, T3, T4 Taktzustände der CPU; TW eingefügte Waitzustände)

$A_0 \dots A_{15}$  Belegung des CPU-Adreßbusses (PC aktueller Programmzählerstand; R Inhalt des Refreshregisters auf  $A_0 \dots A_4$ )

$\overline{MREQ}$ , RD

$\overline{MT}$ , RFSH Systemsteuersignalausgänge der CPU

WAIT CPU-Steuersignaleingang

$D_0 \dots D_7$  Belegung des Systemdatenbusses an der CPU (IN gültige Op-Kode-Daten)

Aus den Grundfunktionen lassen sich die entsprechenden Zeitdiagramme der CPU ableiten. Diese sind in den Bildern 3.1.5 bis 3.1.11 dargestellt. Dabei wurden die Grundfunktionen sowohl mit als auch ohne WAIT-Zustände dargestellt:

- Befehlscodelesezyklus (M1-Zyklus) (Bild 3.1.5)
- Speicherlese- oder -schreibzyklus (Bild 3.1.6)
- Ein-/Ausgabe- (I/O-) Zyklus (Bild 3.1.7)
- Busanforderungs- und -bestätigungszyklus (Bild 3.1.8)
- Interruptquittierungszyklus (Bild 3.1.9)
- NMI-Quittierungszyklus (Bild 3.1.10)
- Rückkehr aus dem Haltzustand (Bild 3.1.11).

### 3.1.4.1. Befehlscodelesen

Bild 3.1.5a zeigt das Zeitverhalten beim Lesen eines Befehlscodes (M1-Zyklus). Dieser M1-Zyklus ist für den größten Teil der Instruktionen vier Taktzustände lang, unter der Voraussetzung, daß kein WAIT-Zustand eingefügt wurde. Der M1-Zyklus läßt sich in zwei Teile untergliedern. Im ersten Teil erfolgt das Lesen des OP-Kodes, im zweiten Teil der Refresh.

Zu Beginn des M1-Zyklus wird der Inhalt des Programmzählers PC auf den Adreßbus gegeben. Nach einer halben Taktperiode werden die Signale  $\overline{MREQ}$  und RD aktiv (L).

Mit diesen Signalen wird das Lesen des Speichers eingeleitet. Das Timing sichert, daß schon bei der fallenden Flanke von  $\overline{\text{MREQ}}$  und  $\overline{\text{RD}}$  der Adreßbus stabil ist. Damit ist es möglich, die fallende Flanke von  $\overline{\text{MREQ}}$  direkt einzusetzen – als Chip-enable- (CE-) Takt für dynamische und für flankenaktivierte Speicher. Das  $\overline{\text{RD}}$ -Signal zeigt der Speicherlogik und der Systemlogik, daß der Datenbus für einen Signalfluß in Richtung CPU zu schalten ist. Mit der fallenden Flanke von T2 wird die WAIT-Leitung abgefragt. Die Daten werden mit der steigenden Flanke von T3, also am Ende von T2, in die CPU übernommen. Es ist damit sichergestellt, daß die CPU den Datenbus abgefragt hat, bevor  $\overline{\text{MREQ}}$  und  $\overline{\text{RD}}$  inaktiv werden.

Die Taktzustände T3 und T4 des Befehlscodelesezyklus werden innerhalb der CPU für die Dekodierung und Ausführung des eingelesenen Befehls benötigt, so daß innerhalb dieser Zustände keine weitere Operation ausgeführt wird. (Daten- und Adreßbus sind von der CPU nicht belegt für die Befehlsausführung.) Während T3 und T4 enthalten die niederwertigsten 7 bit des Adreßbusses den Inhalt des Registers R, d. h. die Refreshadresse.  $\overline{\text{MREQ}}$  und  $\overline{\text{RFSH}}$  dienen dazu, alle dynamischen Schreib-Lese-Speicher gleichzeitig anzusprechen und damit aufzufrischen. Das Lesesignal  $\overline{\text{RD}}$  wird während des Refresh nicht aktiv, um Konflikte am Datenbus durch gleichzeitiges Ansprechen zu vermeiden. Damit infolge der UND-Verknüpfung von  $\overline{\text{MREQ}}$  und  $\overline{\text{RFSH}}$  für das gleichzeitige Ansprechen aller Speicher keine Dekodierspitzen entstehen, wird das  $\overline{\text{RFSH}}$ -Signal jeweils eine halbe Taktperiode breiter gemacht als Anfang und Ende des  $\overline{\text{MREQ}}$ . Das  $\overline{\text{RFSH}}$ -Signal kann daher nicht direkt für den Speicherrefresh benutzt werden, da nicht sicher ist, daß schon zu Beginn des  $\overline{\text{RFSH}}$ -Signals der Adreßbus mit der Refreshadresse eingeschwungen ist.

Bild 3.1.5b zeigt den Befehlscodelesezyklus, wenn während der Abfrage in T2 eine aktive WAIT-Leitung festgestellt wird. Der M1-Zyklus wird dann entsprechend verlängert. Stellt die CPU während T2 ein aktives  $\overline{\text{WAIT}}$  fest, so wird ein zusätzlicher Zustand TW eingeführt. Auch während der fallenden Flanke des Taktes CP in TW wird das  $\overline{\text{WAIT}}$ -Signal abgefragt und bei Aktivität der Maschinenzklus weiter verlängert. Damit ist es möglich, den Lesezyklus so zu verlängern, daß dieser an die Zeitbedingung Zugriffzeit des Speichers (in den meisten Fällen ein ROM) angepaßt werden kann.

#### 3.1.4.2. Speicherlesen oder -schreiben

Aus Bild 3.1.6 werden die Unterschiede zwischen einem M1- oder Befehlscodelesezyklus und einem Speicherlese- oder -schreibzyklus im Zeitverhalten deutlich. Die letzteren Zyklen sind drei Taktperioden lang. Es erfolgt kein Refresh. Im Gegensatz zum M1-Zyklus werden die Daten erst nach 2,5 Taktperioden, vom Aussenden der Adresse ab gerechnet, durch die CPU abgefragt. Die Signale  $\overline{\text{MREQ}}$  und  $\overline{\text{RD}}$  sowie die Adreßaussendung haben das gleiche Zeitverhalten wie im M1-Zyklus zu Beginn des Zyklus.  $\overline{\text{MREQ}}$  und  $\overline{\text{RD}}$  sind um eine halbe Taktperiode, die Adresse ist für einen Takt verlängert.

Für die meisten Halbleiterspeicher ist es erforderlich, daß stabile Adressen und Daten den Schreibimpuls überlappen. Das  $\overline{\text{WR}}$ -Signal erfüllt diese Bedingung und kann daher direkt als Schreibimpuls verwendet werden.

Das Bild 3.1.6b zeigt den gleichen Zyklus mit Verlängerung durch Benutzung des  $\overline{\text{WAIT}}$ -Eingangs der CPU. Die Verhältnisse entsprechen in diesem Punkt denen der Verlängerung des M1-Zyklus durch  $\overline{\text{WAIT}}$ .

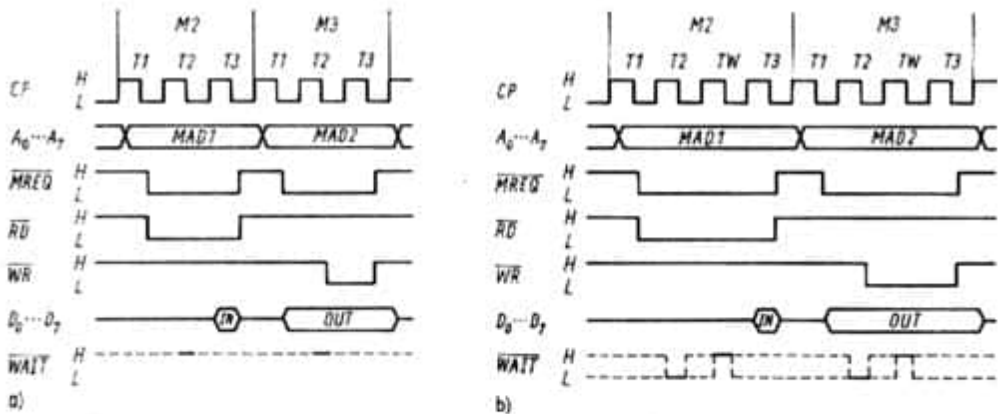


Bild 3.1.6. Speicherlese- und -schreibzyklus

a) ohne Waitzustände

b) mit Einfügung von je einem Waitzustand

M2 Speicherlesezyklus der CPU

M3 Speicherschreibzyklus der CPU

CP Systemtakt (T1, T2, T3 Taktzustände der CPU; TW eingefügte Waitzustände)

 $A_0 \dots A_{15}$  Belegung des CPU-Adreßbusses (MAD1, MAD2 beliebige Speicherplatzadressen) $\overline{MREQ}$ ,  $\overline{WR}$ ,  $\overline{RD}$  Systemsteuersignalausgänge der CPU $D_0 \dots D_7$  Belegung des Systemdatenbusses an der CPU (IN gültige Speicherdaten; OUT gültige CPU-Daten)

WAIT CPU-Steuersignaleingang

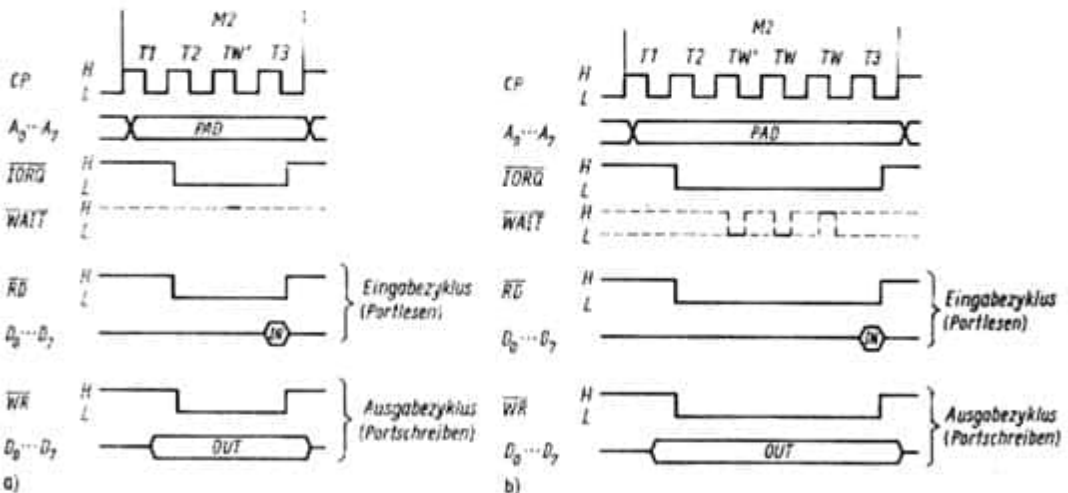


Bild 3.1.7. I/O-Lese- oder -Schreibzyklus

a) ohne zusätzliche Waitzustände

b) mit Einfügung von zwei zusätzlichen Waitzuständen

M2 INPUT- oder OUTPUT-Maschinenzyklus

CP Systemtakt (T1, T2, T3 Taktzustände der CPU; TW' automatisch eingefügte Waitzustände; TW auf Anforderung eingefügte Waitzustände)

 $A_0 \dots A_7$  Belegung des niederwertigen Teil des CPU-Adreßbusses (PAD Portadresse) $\overline{IORQ}$ ,  $\overline{RD}$ ,  $\overline{WR}$  Systemsteuersignalausgänge der CPU $D_0 \dots D_7$  Belegung des Systemdatenbusses an der CPU (IN gültige Portdaten; OUT gültige CPU-Daten)

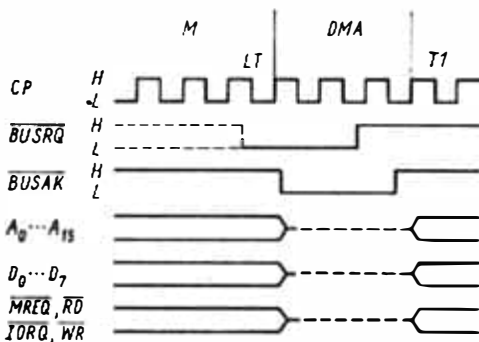
WAIT CPU-Steuersignaleingang

3.1.4.3. Datenein- und -ausgabe

Bild 3.1.7 zeigt die Datenein- und -ausgabe. In diesem Bild sind der Lese- und der Schreibzyklus gleichzeitig dargestellt, obwohl natürlich nur einer der beiden Zyklen auftreten kann. Das entsprechende zweite Steuersignal ( $\overline{WR}$  oder  $\overline{RD}$ ) befindet sich im inaktiven Zustand, und der Datenbus enthält nur die für die Operation adäquaten Daten. Ansonsten gelten die hier aufgeführten Fakten für beide Datentransfers. Man erkennt, daß nach T2 ein  $\overline{WAIT}$ -Zustand eingefügt ist. Dieser Zustand wird von der CPU automatisch eingefügt, damit an die I/O-Baugruppen keine hohen Geschwindigkeitsforderungen gestellt werden müssen. Das betrifft insbesondere die Dekodierlogik, die festlegt, ob aufgrund des Timing zusätzliche  $\overline{WAIT}$ -Anmeldungen zu erfolgen haben. Es müßte dann die  $\overline{WAIT}$ -Leitung zum Zeitpunkt der fallenden Flanke von T2 aktiv sein auf der Basis von  $\overline{IORQ}$  und der Portdekodierung, wofür nur etwa 40 ns zur Verfügung stünden. Der automatisch generierte  $\overline{WAIT}$ -Zustand ermöglicht die Verwendung von Ein-/Ausgabe-Bausteinen in MOS-Technik. Während T2 wird aufgrund der o.g. Wirkungsweise die  $\overline{WAIT}$ -Leitung im Gegensatz zu den Speicheroperationen nicht abgefragt, sondern nur in den  $\overline{WAIT}$ -Zuständen. Bild 3.1.7b zeigt den um zwei extern verursachte  $\overline{WAIT}$ -Zustände verlängerten Zyklus. Wie im Fall der Speicherschreiboperation ist das Timing von Daten und  $\overline{WR}$  so gestaltet, daß die Zeit stabiler Daten die Zeit des aktiven Schreibimpulses  $\overline{WR}$  überlappt. Dadurch kann  $\overline{WR}$  als Takt für ein Ausgabeport verwendet werden. Die Daten können sowohl flankengesteuert mit der steigenden oder fallenden Flanke von  $\overline{WR}$  in das Port eingeschrieben als auch zustandsgesteuert mit  $\overline{WR}$  übernommen werden.

3.1.4.4. Busanforderung und -bestätigung

Bild 3.1.8 zeigt die Zeitverhältnisse für den Vorgang einer Busanforderung und der nachfolgenden Bestätigung. Zu Beginn des letzten Taktzustands eines jeden Maschinenzyklus wird die Leitung  $\overline{BUSRQ}$  durch die CPU abgefragt. Wird ein aktives Signal erkannt, dann erfolgt mit der nächsten steigenden Taktflanke, d.h. also nach Abschluß des laufenden Maschinenzyklus, die Freigabe des Busses durch die CPU und die Aussendung des Signals  $\overline{BUSAK}$  als Bestätigung. Die CPU setzt ihre Daten- und Adreßleitungen sowie die Tristatesteuerleitungen in den hochohmigen Zustand. Dadurch kann eine andere Einheit den Bus steuern und einen Datenaustausch zum Speicher und den I/O-Ports durchführen. Dieser Vorgang wird als DMA (direkter Speicherzugriff) auf der Basis „cycle stealing“ (Zyklusstehlen) bezeichnet. Im Zustand  $\overline{BUSAK}$  aktiv wird mit jeder steigenden



**Bild 3.1.8**  
**Busanforderungs- und -bestätigungszyklus**  
M beliebiger Maschinenzyklus der CPU  
DMA direkter Zugriff auf das Bussystem der CPU möglich  
CP Systemtakt (LT letzter Taktzustand des Maschinenzyklus M; T1 erster Taktzustand des auf M folgenden Maschinenzyklus)  
 $\overline{BUSRQ}$ ,  $\overline{BUSAK}$  Bussteuersignale der CPU  
 $A_0 \dots A_{15}$ ,  $D_0 \dots D_7$ ,  $\overline{MREQ}$ ,  $\overline{IORQ}$ ,  $\overline{RD}$ ,  $\overline{WR}$  Belegung des Mikrorechnerbussystems durch die CPU (während DMA floatend)

Taktflanke die Leitung  $\overline{\text{BUSRQ}}$  abgefragt und bei Inaktivität nach dem nachfolgenden T-Zustand wieder zur normalen Arbeitsweise der CPU übergegangen. Prinzipiell läßt sich die Busanforderung über eine beliebige Zeit ausdehnen. Es muß dann jedoch beachtet werden, daß seitens der CPU kein Refresh dynamischer Speicher mehr erfolgt. Sind derartige Speicher im System vorhanden, muß entweder die Zeit der Busübernahme begrenzt werden, oder das externe busanfordernde Gerät muß die Speicherauffrischung sicherstellen.

Da  $\overline{\text{BUSRQ}}$  eine höhere Priorität als der nichtmaskierbare und der maskierbare Interrupt hat, wird eine Interruptanforderung durch die CPU nicht akzeptiert.

### 3.1.4.5. Interruptanforderung und -bestätigung

Im Gegensatz zur Busanforderung wird der Interrupt nur mit der letzten steigenden Taktflanke des letzten Maschinenzyklus einer Instruktion abgefragt. Dadurch ist sichergestellt, daß jede Instruktion erst vollständig abgearbeitet wird. Bild 3.1.9 zeigt den Zeitablauf, wenn ein  $\overline{\text{INT}}$ -Signal aktiv ist und ein Interrupt bestätigt wird. Voraussetzung dafür ist, daß softwaremäßig durch die Maskierung ein Interrupt freigegeben wurde und kein  $\overline{\text{BUSRQ}}$  aktiv ist. Die Bestätigung des  $\overline{\text{INT}} = \text{L}$  erfolgt durch Erzeugung eines speziellen M1-Zyklus. Dieser M1-Zyklus ist dadurch gekennzeichnet, daß anstelle des  $\overline{\text{MREQ}}$  das  $\overline{\text{IORQ}}$  aktiv wird. Die unterbrechende Einheit kann diese Kombination von  $\overline{\text{M1}}$  und  $\overline{\text{IORQ}}$  dekodieren.  $\overline{\text{M1}}$  und  $\overline{\text{IORQ}}$  werden von der CPU nur bei Interrupt gleichzeitig verwendet, da ansonsten  $\overline{\text{M1}}$  das Lesen eines Befehls kennzeichnet. In dem speziellen  $\overline{\text{M1}}$ -Zy-

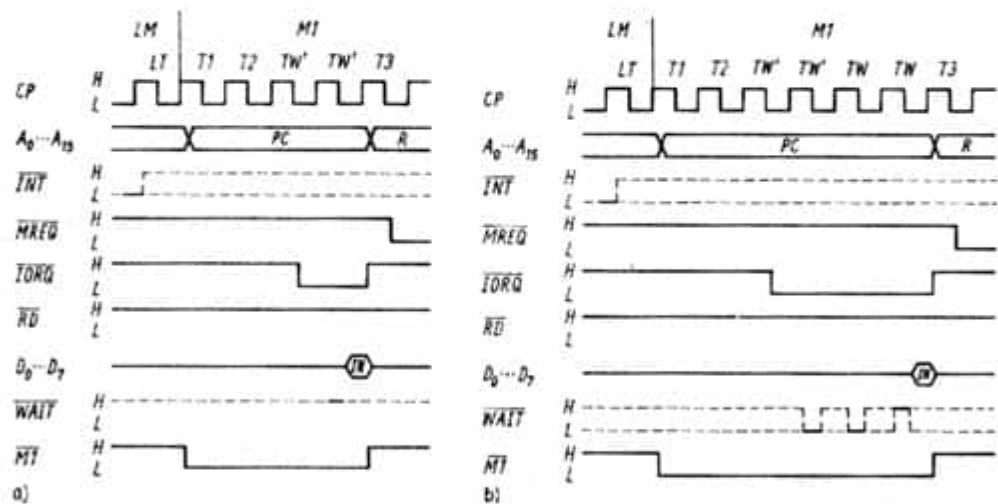


Bild 3.1.9. Interruptquittierungszyklus

a) ohne zusätzliche Waitzustände

b) mit Einfügung von zwei zusätzlichen Waitzuständen

LM letzter Maschinenzyklus des vorangegangenen Befehls

M1 Quittierungszyklus der CPU

CP Systemtakt (LT letzter Taktzustand von LM; T<sub>1</sub>, T<sub>2</sub>, T<sub>3</sub> Taktzustände der CPU; TW' automatisch von der CPU eingefügte Waitzustände; TW auf Anforderung eingefügte Waitzustände)

A<sub>0</sub>...A<sub>15</sub> Belegung des CPU-Adressbusses (PC aktueller, stationärer Programmzählerstand; R Refreshregisterinhalt)

M1, IORQ, MREQ, RD

Systemsteuersignalausgänge der CPU

INT, WAIT CPU-Steuersignaleingänge

D<sub>0</sub>...D<sub>7</sub> Belegung des Systemdatenbusses an der CPU (IN von dem interruptfördernden Gerät in den Interruptmoden IM1 und IM2 auf den Datenbus plazierte Information)

klus werden von der CPU automatisch nach T2 zwei  $\overline{\text{WAIT}}$ -Zustände eingeführt, um der Prioritätskettenschaltung für den Interrupt ein Einschwingen zu ermöglichen. Diese zwei  $\overline{\text{WAIT}}$ -Zustände erlauben den Ein-/Ausgabe-Einheiten festzustellen, welche Einheit die höchste Anmeldepriorität besitzt.

Bild 3.1.9b zeigt, wie das Zeitdiagramm durch Einfügung eines extern erzeugten  $\overline{\text{WAIT}}$ -Zustands gedehnt wird.

Bild 3.1.10 zeigt die Antwort auf einen nichtmaskierbaren Interrupt ( $\overline{\text{NMI}}$ ). Ein  $\overline{\text{NMI}}$ -Signal wird in der CPU zwischengespeichert; es wird ein entsprechendes Flipflop gesetzt, das durch die CPU wie die  $\overline{\text{INT}}$ -Leitung am Ende einer Instruktion abgefragt wird. Das  $\overline{\text{NMI}}$ -Signal hat eine höhere Priorität als  $\overline{\text{INT}}$  und eine geringere als  $\overline{\text{BUSRQ}}$ ; es kann softwaremäßig nicht gesperrt werden. Das ist sinnvoll, um auf besonders wichtige Ereignisse sofort reagieren zu können, z. B. auf eine Havarie, einen Netzausfall.

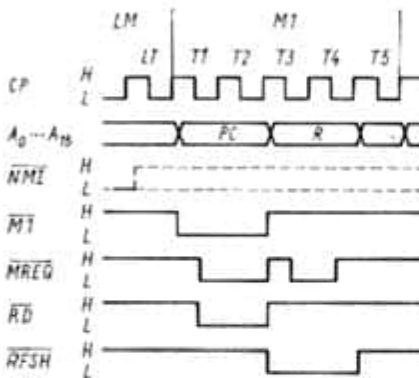


Bild 3.1.10

## NMI-Quittierungszyklus

LM letzter Maschinenzklus des vorangegangenen Befehls

M1 Quittierungszyklus der CPU

CP Systemtakt (LT letzter Taktzustand von LM; T1,T2,T3,T4,T5

Taktzustände der CPU)

 $\overline{\text{NMI}}$  CPU-Steuerleitung

M1, MREQ, RD, RFSH

Systemsteuersignaleingänge der CPU

Die  $\overline{\text{NMI}}$ -Bestätigung erfolgt dadurch, daß die CPU den aktuellen Programmzählerstand im externen Stack ablegt und die weitere Programmabarbeitung an der Stelle 0066H fortsetzt. Dort beginnt das Programm der  $\overline{\text{NMI}}$ -Routine, z. B. ein Rettungsprogramm. Im Bestätigungszyklus werden durch die CPU natürlich keine  $\overline{\text{WAIT}}$  eingefügt, da keine zeitaufwendigen oder zeitkritischen externen Vorgänge aufgrund der festen Startadresse der Interruptroutine ausgeführt werden müssen.

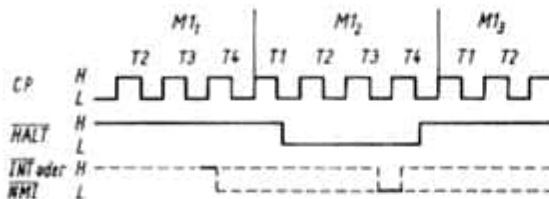


Bild 3.1.11. Rückkehr aus dem Haltzustand

M1<sub>1</sub> Befehlsholeryklus der CPU, in dem der Befehl HALT dekodiert wurde automatisch von der CPU erzeugter Maschinenzklus mit Befehlsabarbeitung NOP zur Aufrechterhaltung der Refreshfunktion (es können beliebig viele derartige Zyklen auftreten, bis eine Interruptlinie oder RESET aktiv werden)

M1<sub>2</sub> INT- bzw. NMI-Quittierungszyklus

M1<sub>3</sub> Systemtakt (T1,T2,T3,T4 Taktzustände der CPU)

CP Systemtakt

$\overline{\text{NMI}}$ ,  $\overline{\text{INT}}$  CPU-Steuerleitung

HALT CPU-Steuerleitung

### 3.1.4.6. Rücksprung aus HALT

Die HALT-Instruktion wird im Programm ausgeführt, wenn der normale Programmablauf beendet ist bzw. wenn die CPU weitere Daten benötigt, um das Programm fortsetzen zu können. Den Haltzustand kann die CPU nur dann verlassen, wenn ein  $\overline{\text{NMI}}$ - oder  $\overline{\text{INT}}$ -Signal bestätigt wird. Während HALT werden von der CPU NOP-Befehle ausgeführt. Der Zweck dieser NOP-Befehle ist der Refresh dynamischer Speicher. Die NOP bestehen aus MI-Zyklen, mit dem Unterschied, daß die Daten vom Speicher durch den Kode für NOP ersetzt werden (00H). Während dieser NOP-Befehle wird von der CPU zur Bestätigung das Signal  $\overline{\text{HALT}}$  ausgegeben. Bild 3.1.11 zeigt das zugehörige Zeitverhalten.

### 3.1.5. Funktionelle Befehlsdarstellung

In diesem Abschnitt sollen die einzelnen CPU-Befehle in ihrer Wirkung erläutert werden. Weiterhin werden Zusammenhänge zwischen Befehlen und Befehlsgruppen dargestellt.

#### 3.1.5.1. Einführung in die Befehlstypen

Die Basisbefehle der CPU U880 lassen sich prinzipiell in folgende Hauptgruppen unterteilen:

- Befehle zur Datenübertragung
- Befehle zur blockweisen Datenbehandlung
- arithmetische und logische Befehle
- Befehle zum Rotieren und Verschieben
- Befehle zur Bitmanipulation
- Sprungbefehle
- Befehle zur Unterprogrammbehandlung
- Befehle zur Ein-/Ausgabe (I/O-Befehle)
- Befehle zur CPU-Steuerung.

Die Befehle zur Datenübertragung bzw. zum Laden stellen die größte Gruppe dar. Aufgrund der vielseitigen Adressierungsvarianten läßt sich eine große Zahl von Op-Kodes ableiten. Die Ladebefehle übertragen Daten in der CPU zwischen Registern oder zwischen Registern und dem externen Speicher. Darüber hinaus besteht die Möglichkeit, daß im Befehl enthaltene Daten (Konstanten) in ein Register oder in den Speicher geladen werden. Da die CPU eine Verarbeitungsbreite von 8 bit hat, sind die meisten Befehle auf 8 bit Breite orientiert. Eine Reihe von Registern läßt sich aber zu Registerpaaren zusammenfassen; andere sind bereits 16-bit-Register, da sie für Adressierungszwecke vorgesehen sind. Weitere Befehle erlauben das Austauschen von Daten der Registerpaare. Im Gegensatz zu den Ladebefehlen, wo die Quelle der Daten unverändert bleibt, ändern sich hier mehrere Register.

Mit der CPU U880 ist die Möglichkeit gegeben, den Datenverkehr auch blockweise zu organisieren. Dabei reicht ein einzelner Befehl, um einen Datenblock von beliebiger Länge innerhalb des Speichers umzuladen. Man kann das als einen softwaremäßigen direkten Speicherzugriff betrachten. Die Blocksuchbefehle werden dazu verwendet, um einen definierten Speicherbereich auf Übereinstimmung von Speicherinhalt mit dem Inhalt des Akkumulators abzusuchen.

Es werden dabei zwei 8-bit-Wörter verglichen, und der Befehl wird abgebrochen, wenn die Bereichsgrenze erreicht ist oder wenn eine Übereinstimmung gefunden wurde.

Die Befehle für blockweise Ausführung schließen auch die Auffrischung (refresh) von dynamischen Speichern ein.

Das ist notwendig, da in Abhängigkeit von der programmierten Blocklänge eine größere Befehlsarbeitungszeit erforderlich sein kann. Das ist auch der Grund, warum eine Unterbrechung des zyklischen Befehlsablaufs durch Interrupt möglich ist.

Die arithmetischen und logischen Befehle beziehen sich auf Registerinhalte oder auf Speicherinhalte. Sie sind wie die Ladebefehle vorwiegend 8-bit-orientiert, einzelne lassen sich mit Registerpaaren oder Doppelregister auf 16 bit anwenden. Im Fall der 8-bit-Befehle wird das Ergebnis im Akkumulator abgelegt, und die Flags werden entsprechend gesetzt.

Mit den Bitmanipulationsbefehlen kann man beliebige Bits im Akkumulator, in einem allgemeinen Register oder im Speicher testen, setzen oder rücksetzen. Dafür wird lediglich ein einzelner Befehl benötigt.

Die Sprung-, Unterprogrammaufruf- und Rücksprungbefehle werden eingesetzt, um den Befehlsablauf bezüglich der Adressen im Programmspeicher zu steuern. Dadurch wird es erst möglich, den Programmspeicher optimal auszunutzen. Mehrfach benötigte Programmteile können aufgerufen werden und benötigen dadurch nur einmal den Platz im Programmspeicher. Die dazu erforderliche Rettung des aktuellen Programmzählers erfolgt automatisch.

Ein weiterer Sprungbefehl ist der Rückstart (Restart), der mit einer 1-byte-Instruktion den Sprung auf acht vorgegebene Adressen ermöglicht. Die neue Adresse des Programmzählers ist kodiert im Op-Kode (3 bit) enthalten. Dieser Befehl ist besonders sinnvoll in Verbindung mit der Interruptmode IM0, wobei das dem Interrupt folgende Wort des Datenbusses als Befehl interpretiert und von der unterbrechenden Einheit geliefert wird.

Weiterhin ist es möglich, daß die Registerinhalte von HL, IX und IY direkt in den Programmzähler geladen werden können.

Dadurch vergrößert sich die Gruppe der Befehle, die eine Programmverzweigung bewirken. Die Vielseitigkeit dieser Befehlsgruppe wird noch dadurch erhöht, daß die Befehlsausführung von verschiedenen Bedingungen (Flags) abhängig gestaltet werden kann.

Die Befehle der I/O-Gruppe lassen unterschiedlichste Organisationen des Datentransfers über I/O-Ports zu. Neben der direkten Angabe der Portadresse im Befehl besteht die Möglichkeit, eine Adressierung des Ports über den Inhalt des Registers C vorzunehmen.

Die zyklischen Portbefehle, die mehrere Bytes transferieren, sind sehr sinnvoll, wenn größere Datenmengen bewegt werden sollen. Das trifft z. B. zu, wenn Folienspeicher, sog. Floppy-Disks, verwendet werden. Hardsektorierte Disketten haben je Sektor einen Dateninhalt von 128 bytes, solche mit „double density“ 256 bytes. Somit ist es möglich, mit einem Befehl einen Block von Daten zu übernehmen. Die Organisation eines Floppy-Disks wird somit softwaremäßig erleichtert.

Für die Steuerung des Prozessors ist eine Gruppe von Befehlen vorhanden, die es gestattet, die CPU in den benötigten Betriebszustand zu bringen. Hierzu gehören das Programmieren der Interruptbetriebsart, das Setzen bzw. Rücksetzen der Interruptfreigabeflipflops sowie die HALT- und NOP-Instruktionen.

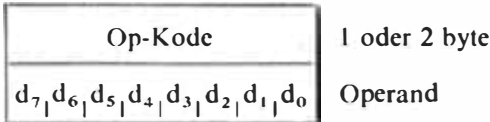
### 3.1.5.2. Adressierungsarten

Ein wichtiges Leistungskriterium für den Befehlssatz eines Mikroprozessors sind Anzahl und Art der möglichen Adressierungsarten. Die unterschiedlichen Adressierungsmoden

sind hierbei einerseits erforderlich, um verschiedene Varianten von Operanden verarbeiten zu können, und andererseits, um eine effiziente Programmgestaltung zu ermöglichen.

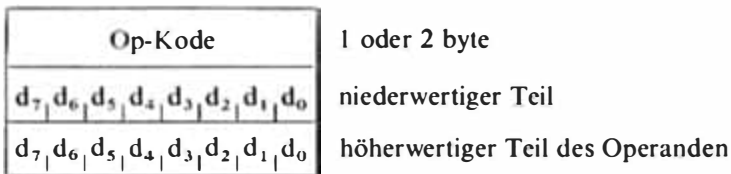
Nachfolgend sollen die Adressierungsarten der CPU U880 detailliert dargestellt werden.

**Unmittelbare Adressierung.** In dieser Adressierungsmode ist der Operand das Byte, das unmittelbar dem Op-Kode folgt. Im Op-Kode ist daher nur ein Hinweis auf diese Adressierungsart enthalten, eine umfangreichere Adreßerzeugung ist nicht erforderlich. Der Operand kann als dem Befehl zugehörig betrachtet werden.



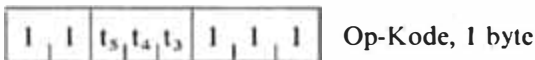
Der Operand wird auch als *Konstante* bezeichnet, da er durch den Befehl in seinem Wert fest vorgegeben ist.

**Erweiterte unmittelbare Adressierung.** Diese Adressierungsmode ist eine Erweiterung der unmittelbaren Adressierung in der Form, daß der Operand auf 2 byte verlängert wird, um so Doppelregister oder Registerpaare laden zu können. Die 2 byte des Operanden folgen unmittelbar dem Op-Kode.



Ein Beispiel für diese Adressierungsart ist das Laden des Indexregisters IX mit einer 16-bit-Konstanten (LD IX,nn).

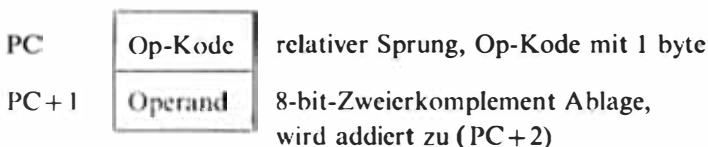
**Modifizierte Adressierung der Seite Null.** Mit einer Seite wird im Zusammenhang mit Mikrorechnern i. allg. ein Speicherbereich von 4 Kbyte bezeichnet. Die Seite Null spezifiziert somit den Bereich 0000H ... 1000H. Die CPU U880 besitzt einen speziellen Ein-bytebefehl, der einen Unterprogrammaufruf zu einer von acht möglichen Adressen in diesem Speicherbereich vornimmt. Die konkrete Rufadresse wird hierbei durch 3 bit des Op-Kode-Bytes definiert. Der Restartbefehl hat somit folgende Struktur:



Die effektive Restartadresse ist

$$0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ t_5\ t_4\ t_3\ 0\ 0\ 0\ B.$$

**Relative Adressierung.** Bei der relativen Adressierung wird das dem Op-Kode folgende Byte als Ablage benutzt für einen relativen Sprung. Damit sowohl vorwärts als auch rückwärts gesprungen werden kann, wird die Ablage im Zweierkomplement angegeben und zum PC addiert.

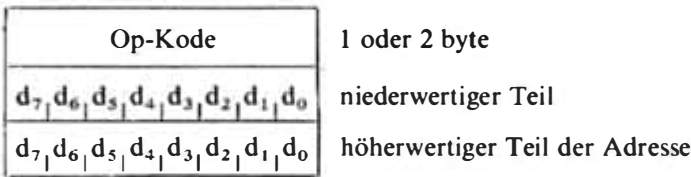


Diese Adressierungsart für einen relativen Sprung kann dann angewendet werden, wenn der Sprungabstand relativ klein ist. Dann hat der Befehl gegenüber absoluter Adressierung zwei Vorteile:

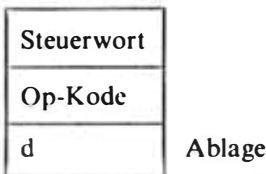
- Der Befehl erspart 1 byte im Befehlsspeicher.
- Die Berechnung einer absoluten Adresse entfällt, damit entfällt bei Verschiebung des Programms eine Befehlsänderung.

In den meisten praktischen Fällen ist die Tatsache gegeben, daß nur eine geringe Ablage erforderlich ist. Die vorzeichenbehaftete Ablage kann einen Bereich einnehmen von +127 bis -128, gerechnet von der Adresse des Op-Kodes plus zwei. Damit kann von +129 bis -126, gerechnet von der Adresse des Op-Kodes des relativen Sprunges, gesprungen werden.

**Erweiterte Adressierung.** Bei der erweiterten Adressierung folgen dem Op-Kode 2 byte (16 bit), die als Konstante dienen. Diese Daten können entweder eine Adresse bei absoluten Sprung- oder Rufbefehlen darstellen, oder sie spezifizieren eine Adresse für einen absoluten Datentransfer mit dem Speicher.



**Indizierte Adressierung.** Bei der indizierten Adressierung wird die Adresse ähnlich wie bei der relativen Adressierung gebildet, nur daß als Basis nicht der PC, sondern eines der Indexregister IX oder IY verwendet wird. Dadurch ist dieser Befehl besonders prädestiniert zur Abarbeitung von Tabellen. Die Werte IX+d oder IY+d dienen als Pointer. Die in den hierdurch adressierten Speicherplätzen lokalisierten Daten stellen die Operanden bei dieser Adressierungsart dar. Da die Inhalte von IX und IY bei diesen Befehlen unverändert bleiben und für d eine Ablage in der Darstellung des vorzeichenbehafteten Zweierkomplements zur Anwendung kommt, läßt sich ein Bereich von +127 bis -128 relativ zu IX oder IY erfassen.



**Registeradressierung.** In vielen Befehlen der CPU U880 werden direkt ein oder zwei CPU-Register als Operanden benutzt. Ein Beispiel hierfür ist die Befehlsgruppe

LD r<sub>1</sub>, r<sub>2</sub>,

wobei mit jeweils 3 bit des Op-Kodes ein Register bestimmt wird.

**Implizierte Adressierung.** Eine Anzahl von Befehlen des U880 bezieht sich mit ihrem Op-Kode grundsätzlich auf bestimmte CPU-Register, die als feste Operanden dienen. Beispiele hierfür sind die Akkumulatorbefehle NEG und CPL.

**Indirekte Registeradressierung.** Dieser Adressierungstyp verwendet ein 16-bit-Registerpaar der CPU (z. B. HL) als Pointer zum Speicher. Er wird häufig für Speicherzugriffe allgemeiner Art verwendet. Beispiele für diese Adressierung sind die Blockübertragungs- und Blocksuchinstruktionen sowie die Memoryadressierung durch das Registerpaar HL.

**Bitadressierung.** Die Bitadressierung wird bei der im Befehlssatz des U880 implementierten Befehlsgruppe zur Bitmanipulation (Bittesten, -setzen und -rücksetzen) angewendet. Hierbei erfolgt die Auswahl des spezifizierten Bits (eins von acht) durch drei entsprechend kodierte Op-Kodebits. Die Operandenadressierung kann mit Hilfe direkter und indirekter Registeradressierung sowie durch indizierte Adressierung erfolgen.

**Gemischte Adressierung.** Eine Vielzahl von Befehlen verwendet mehrere Operanden. Deshalb können in einer Instruktion ebenfalls mehrere Adressierungsarten auftreten, um Datensinke und Datenquelle festlegen zu können. Ein Beispiel hierfür ist die Befehlsgruppe

LD (IX + d), r,

die neben einer Registeradressierung eine indizierte Adressierung enthält.

### 3.1.5.3. Operandenbeschreibung

Durch die dargestellten Adressierungsarten lassen sich innerhalb der Befehlsgruppen verschiedene Operanden spezifizieren. Die nachfolgende Aufstellung enthält die symbolische Vereinbarung der Operanden, die bei der Beschreibung der U880-Assemblersprache bzw. des Befehlssatzes verwendet werden:

- r** spezifiziert mit 3 bit im Op-Kode eines der 8-bit-CPU-Register A, B, C, D, E, H oder L;  
für die Op-Kode-Bits gilt hierbei folgende Zuordnung:
- |     |   |
|-----|---|
| 000 | B |
| 001 | C |
| 010 | D |
| 011 | E |
| 100 | H |
| 101 | L |
| 111 | A |
- M** spezifiziert den Inhalt der Speicherstelle, die durch das Registerpaar HL adressiert wird
- n** spezifiziert einen Einbyteoperanden (8-bit-Konstante oder 8-bit-Portadresse) im Zahlenbereich 00H ... 0FFH (0 ... 255)
- nn** spezifiziert einen Zweibyteoperanden (16-bit-Konstante oder 16-bit-Speicherplatzadresse) im Zahlenbereich 0000H ... 0FFFFH (0 ... 65535)
- d** spezifiziert einen Einbyteoperanden (Abstand, Ablage) im Zahlenbereich -128 ... +127 (vorzeichenbehaftetes Zweierkomplement)
- e** spezifiziert einen Einbyteoperanden (Abstand, Ablage) im Zahlenbereich -126 ... +129; es gilt:  $e = d + 2$
- b** spezifiziert mit 3 bit des Op-Kodes im Binärformat eine Zahl (Nummer) im Bereich 0 ... 7
- dd** spezifiziert mit 2 bit im Op-Kode eines der 16-bit-Register bzw. Registerpaare BC, DE, HL oder SP;  
für die Op-Kode-Bits gilt folgende Zuordnung:
- |    |    |
|----|----|
| 00 | BC |
| 01 | DE |
| 10 | HL |
| 11 | SP |

- qq** spezifiziert ebenfalls mit zwei Op-Kode-Bits eines der Registerpaare BC, DE, HL, AF; es gilt hierbei folgende Zuordnung für die Op-Kode-Bits:
- |    |    |
|----|----|
| 00 | BC |
| 01 | DE |
| 10 | HL |
| 11 | AF |
- s** steht für einen der Operanden r, n, M, (IX + d), (IY + d)
- f** steht für einen der Operanden r, M, (IX + d) oder (IY + d)
- pp** spezifiziert mit 2 bit des Op-Kodes eines der 16-bit-Register bzw. Registerpaare BC, DE, IX, SP; die Zuordnung der Op-Kode-Bits entspricht der des Operanden dd, mit der Ausnahme, daß das Registerpaar HL durch das Indexregister IX ersetzt wird
- rr** spezifiziert eines der Registerpaare bzw. 16-bit-Register BC, DE, IY oder SP; die Zuordnung der Op-Kode-Bits entspricht wiederum der des Operanden dd, jedoch wird HL durch IY ersetzt
- p** spezifiziert mit 3 bit im Op-Kode (Binärdarstellung: t) eine Rückstartadresse aus den Werten 00H, 08H, 10H, 18H, 20H, 28H, 30H oder 38H (dezimal: 0, 8, 16, 24, 32, 40, 48 oder 56)
- (nn)** spezifiziert den Inhalt der Speicherstelle nn (Einbyteoperand) oder die Inhalte der Speicherstellen nn und nn + 1 (Zweibyteoperand) als Operanden
- (HL)** spezifiziert den Inhalt der durch HL adressierten Speicherstelle; der Operand (HL) ist somit mit dem Operanden M identisch; die U880-Assemblersprache läßt beide Schreibweisen zu.

Die Darstellung der Klammerschreibweise läßt sich prinzipiell auch auf andere Ausdrücke anwenden. Die Klammern um einen Operanden drücken aus, daß nicht der Operand selbst, sondern der Inhalt des durch den Operanden adressierten Speichers verwendet wird. Eine Ausnahme stellen hierbei die Befehle JMP (HL), JMP (IX) und JMP (IY) dar. Bei ihnen wird entgegen obiger Vereinbarung direkt der Registerinhalt für die Befehlsausführung benutzt. Die Sprünge erfolgen somit auf die Adresse, die in dem jeweiligen Indexregister bzw. im Registerpaar HL steht.

Die CPU U880 benutzt in ihren Flagregistern jeweils sechs Informationsbits, die entsprechend der Wirkung einer Anzahl von Befehlen (meist arithmetische oder logische Befehle) beeinflußt werden. Vier dieser Bedingungsflags sind testbar, d. h., sie können für Programmverzweigungen eingesetzt werden. Diese Verzweigungen können durch Sprung-, Ruf- oder Rückkehrbefehle erreicht werden. Die entsprechenden Befehle besitzen im Präfix des Mnemoniks einen Bedingungskode (condition code), der nachstehende Bedeutung hat. Der Bedingungskode wird allgemein mit cc abgekürzt. Da durch die vier testbaren Flags acht verschiedene Bedingungen möglich sind, erfolgt die Kodierung über 3 bit des Op-Kodes. Ihre Belegung ist ebenfalls nachstehend mit angegeben.

- |       |    |   |
|-------|----|---|
| 0 0 0 | NZ | nicht Null (non zero); Befehl wird ausgeführt, wenn das Zeroflag rückgesetzt ist; Z = 0 |
| 0 0 1 | Z  | Null (zero); Z = 1  |
| 0 1 0 | NC | kein Übertrag (non carry); CY = 0   |
| 0 1 1 | C  | Übertrag (carry); CY = 1  |
| 1 0 0 | PO | Parität ist ungerade (parity odd) bzw. kein Überlauf; P/V = 0                           |
| 1 0 1 | PE | Parität ist gerade (parity even) bzw. Überlauf; P/V = 1                                 |
| 1 1 0 | P  | Vorzeichen ist positiv (plus); S = 0  |
| 1 1 1 | M  | Vorzeichen ist negativ (minus); S = 1.  |

Eine detailliertere Beschreibung der Flagoperationen enthält Abschn. 3.1.5.6.

In den Mnemoniks und Op-Kodes wird für die Zahlen die hexadezimale Schreibweise angewendet, da sie kürzer und oftmals übersichtlicher ist als eine binäre oder dezimale Schreibweise. In der Rechentechnik wird mit binären Größen gearbeitet. Daraus ergibt sich, daß mit 4 bit eine Zahl dargestellt werden kann, die u. U. einen größeren Wert als eine Dezimalziffer haben kann. Derartige Zahlen werden als *Pseudotetraden* bezeichnet, und die Darstellung erfolgt mit Buchstaben, d. h. nach 9 mit A bis F.

dezimal	hexadezimal	binär
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

#### 3.1.5.4. Mnemotechnische Instruktionsdarstellung

In diesem Abschnitt erfolgt eine detaillierte Beschreibung der einzelnen Befehle bzw. der Befehlsgruppen des Befehlssatzes der CPU U880. Die zusammenfassende Darstellung der Befehlsgruppen in Tafelform vermittelt der Abschn. 3.1.5.5., da diese Übersichten vom Anwender häufig zusammenhängend benötigt werden.

Eine Zusammenfassung aller Flagoperationen wird im Abschn. 3.1.5.6. vorgenommen.  
**8-bit-Ladebefehle.** Die mnemotechnische Abkürzung der Assemblersprache für die ganze Gruppe der Ladebefehle heißt: LD (von load). Diesem Kode folgt die Zielangabe, dann die Quellangabe (LD Ziel, Quelle). Zu beachten ist, daß etliche Kombinationen von Adressierungsarten zulässig sind.

Tafel 3.1.2 (S. 48) zeigt alle Basisbefehle der 8-bit-Ladebefehlsgruppe.

Die meisten Op-Kode lassen sich aus dem Befehl LD  $r_1, r_2$  ableiten. Dieser Befehl ist wie folgt zu verstehen:

Die sieben Register A, B, C, D, E, H und L können sowohl als Quellregister wie auch als Zielregister spezifiziert werden. Daraus ergibt sich der Befehl LD  $r_1, r_2$  mit 49 Op-Kode. Davon stellen sieben Befehle, nämlich LD  $r_1, r_2$  mit  $r_1 = r_2$ , einen Leerbefehl dar, der in der Wirkung identisch mit NOP ist, da praktisch keine Datenübertragung zustande kommt und auch keine Flags geändert werden.

Als Quelle der Datenübertragung kommt auch noch eine Konstante in Betracht, die unmittelbar im Befehl enthalten ist und dem Op-Kode folgt. Der Befehl lautet damit

LD r, n.

Als Quelle und als Ziel kann weiterhin der Speicher dienen; daraus ergeben sich die Befehle LD r,M, LD M,r und LD M,n. Der Speicher kann nicht gleichzeitig Quelle und Ziel der Datenübertragung sein, da die Übertragung immer unter Zwischenschaltung der CPU erfolgt.

Mit Hilfe eines Steuerworts lassen sich die auf den Speicher bezogenen Befehle auch in der indizierten Adressierungsmode verwenden, wobei intern in der CPU eine Umschaltung vorgenommen wird. Dem Op-Kode folgt bei diesen Befehlen die Ablage d. Damit ergeben sich die Befehle

```
LD (IX + d),r
LD (IY + d),r
LD (IX + d),n
LD (IY + d),n
LD r,(IX + d)
LD r,(IY + d).
```

Bei diesen Ladebefehlen wird kein Flag beeinflusst.

Unter Bezug auf den Akkumulator mit indirekter Adressierung aus den Registerpaaren oder der Konstanten (nn) ergeben sich folgende Befehle:

```
LD A,(BC)
LD A,(DE)
LD A,(nn)
LD (BC),A
LD (DE),A
LD (nn),A.
```

Flags werden ebenfalls nicht beeinflusst.

Die Register I und R können als Quelle oder Ziel in Verbindung mit dem Akkumulator verwendet werden:

```
LD A,I
LD A,R
LD I,A
LD R,A.
```

Bei den Befehlen LD A,I und LD A,R werden dabei folgende Flags beeinflusst:

```
H := 0
N := 0
P/V := IFF2 (Interruptfreigabeflipflop der CPU).
```

Diese Gruppe der 8-bit-Ladebefehle umfaßt damit 104 sinnvolle Op-Kode.

**16-bit-Ladebefehle.** In Tafel 3.1.3 (S. 50) sind die zulässigen Befehlstypen dieser Gruppe zusammengefaßt.

Zu beachten ist, daß die Möglichkeit zur erweiterten Adressierung für alle Registerpaare gegeben ist und daß die indirekten Registeroperationen, die den Kellerzeiger betreffen, die PUSH- und POP-Befehle sind. Die mnemotechnischen Abkürzungen für diese Befehle heißen PUSH und POP. Der Unterschied zu gewöhnlichen Ladebefehlen besteht darin, daß der Kellerzeiger automatisch erniedrigt oder erhöht wird, wenn ein Byte gekellert oder aus dem Keller (Stack) geholt wird.

PUSH und POP stellen zueinander inverse Befehle dar. Das gilt ebenfalls für die Reihenfolge des Datentransfers der Bytes. Bei PUSH wird zuerst das höherwertige Byte transferiert, bei POP zuerst das niederwertige Byte.

Bei allen 16-bit-Ladebefehlen inklusive PUSH und POP erfolgt keine Beeinflussung der Flags.

**16-bit-Austauschbefehle.** In dieser Gruppe sind sechs Befehle enthalten, die in Tafel 3.1.4 dargestellt sind. Die mnemotechnische Abkürzung EX bezieht sich auf Exchange (Austausch).

Der Op-Kode 08H ermöglicht es dem Programmierer, zwischen den zwei Paaren von Akkumulator und Flagregistern umzuschalten, wogegen der Op-Kode D9H die Umschaltung zum Duplikatsatz der sechs Allgebrauchsregister bewirkt. Diese Op-Kodes sind nur 1 byte lang, um die Zeit, die für den Austausch benötigt wird, soweit wie möglich herabzusetzen, so daß der Duplikatsatz dazu benutzt werden kann, sehr kurze Interruptantwortzeiten zu realisieren.

Ebenso wie bei den 16-bit-Ladebefehlen wird beim Datenaustausch der Register kein Flag beeinflußt.

**Blockweise Datenübertragung.** In dieser Befehlsgruppe (s. ebenfalls Tafel 3.1.4, S. 52) existieren vier Instruktionen mit folgenden Mnemoniks:

LDI	Laden und Inkrementieren
LDIR	Laden und Inkrementieren bis BC=0
LDD	Laden und Dekrementieren
LDDR	Laden und Dekrementieren bis BC=0.

Die Befehlsausführung beeinflußt folgende Flags:

H := 0  
 N := 0  
 P/V := 0,

wenn das Ergebnis des Dekrementierens von BC Null ist, sonst gilt

P/V := 1.

Daraus folgt, daß bei den zyklischen Befehlen nach dem letzten Datentransport P/V=0 ist.

Der Befehl LDI (LDD) bewirkt, daß der Inhalt des Speicherplatzes, auf den HL zeigt, in den Speicherplatz geladen wird, für den DE als Pointer dient. Danach werden die Registerpaare HL und DE inkrementiert (dekrementiert) und der Bytezähler BC grundsätzlich dekrementiert. Damit ist die Möglichkeit für einen weiteren Datentransfer des nächsten Bytes gegeben. Zwischen den Transfers hat man Gelegenheit, erforderlichenfalls eine Datenverarbeitung vorzunehmen. Das Flag P/V kann als Abbruchbedingung genutzt werden für die blockweise Datenübertragung.

Die Instruktion LDIR (LDDR) zeigt die gleiche Befehlsausführung wie LDI (LDD), nur daß von vornherein zyklisch gearbeitet und P/V=0 automatisch als Abbruchbedingung verwendet wird. Mit einer einzigen Instruktion läßt sich somit ein ganzer Block von Daten innerhalb des Speichers ähnlich einem DMA verschieben. Da sowohl für die Adressierung als auch für den Bytezähler Registerpaare mit 16 bit eingesetzt sind, ist sowohl die Lage des Datenblocks als auch seine Länge beliebig. Die Datenblöcke (Quelle und Ziel) können sich dabei auch überlappen. Lediglich ein Befehl mit HL = DE ist sinnlos, da Quelle und Ziel identisch sind. Irgendwelche Beschränkungen treten nicht auf. Zu

beachten ist, daß in der Befehlsabarbeitung zuerst der Datentransfer ausgeführt wird, erst danach wird BC dekrementiert und auf dieser Basis bei zyklischem Befehl entschieden, ob noch ein weiterer Datentransfer ausgeführt werden soll oder ob der Befehl damit beendet ist. Wenn vor Befehlsausführung  $BC=0$  ist, hat man somit eine Blocklänge von  $2^{16} = 65536$  byte spezifiziert.

**Blockweise Datenüberprüfung.** Die blockweise Datenüberprüfung läßt wie die Datenübertragung vier Instruktionen zu, wie in Tafel 3.1.4 dargestellt ist.

Als Mnemoniks sind

CPI	Vergleichen und Inkrementieren
CPIR	Vergleichen und Inkrementieren bis $BC=0$ oder $A=M$
CPD	Vergleichen und Dekrementieren
CPDR	Vergleichen und Dekrementieren bis $BC=0$ oder $A=M$

zugelassen.

Der Befehl CPI (CPD) wird dadurch ausgeführt, daß vom Akkumulatorinhalt der Inhalt des Speicherplatzes, dessen Adresse in HL steht, subtrahiert wird. Der Bytezähler BC wird bei jeder Befehlsausführung dekrementiert. Das Ergebnis des Vergleichs wird durch entsprechende Beeinflussung der Flags festgehalten. Da das Registerpaar HL inkrementiert (dekrementiert) wird, beginnt der Vergleich an der niederwertigeren (höherwertigeren) Speicherstelle.

Die Instruktion CPIR (CPDR) bewirkt die wiederholte Ausführung des Befehls CPI (CPD). Der Befehl CPI (CPD) wird dabei so lange wiederholt, bis eine der beiden Abbruchbedingungen erfüllt ist. Das ist entweder die gefundene Übereinstimmung oder das Ende des zu untersuchenden Datenblocks. Dementsprechend werden auch die Flags gesetzt:

S	:= 1, wenn das Ergebnis der Subtraktion negativ ist, andernfalls S := 0
Z	:= 1, wenn $A = (HL)$ , andernfalls S := 0
H	:= 1, wenn Übertrag von Bit 4, andernfalls H := 0
P/V	:= 1, wenn $BC = 0$ , andernfalls P/V := 0
N	:= 1.

Die wiederholte Befehlsausführung wird dadurch erreicht, daß bei nichterfüllter Abbruchbedingung ( $P/V = 0$ ) der PC zweimal dekrementiert wird.

**8-bit-Arithmetik- und -Logikbefehle.** Alle Befehle dieser Gruppe sind in Tafel 3.1.1 (S.47) zusammenfassend dargestellt. Sie benutzen grundsätzlich den Akkumulator als Zielregister. Im Befehl muß daher nur die Quelle spezifiziert werden. Insgesamt sind in dieser Befehlsgruppe 108 verschiedene Op-Kode zugelassen, die aus folgenden Mnemoniks resultieren:

ADD s	Addieren
ADC s	Addieren mit Übertrag
SUB s	Subtrahieren
SBC s	Subtrahieren mit Übertrag
AND s	logische UND-Verknüpfung
OR s	logische ODER-Verknüpfung
XOR s	exklusive ODER-Verknüpfung
CMP s	Vergleichen mit Akkumulatorinhalt
INC f	Inkrementieren
DEC f	Dekrementieren.

Die Flags CY, Z, S und H werden entsprechend dem Ergebnis der Operation gesetzt. Das Flag P/V wird durch Überlauf (V) beeinflußt. Flag N entspricht dem Bit  $D_4$  des jeweiligen Op-Kodes und zeigt für die Ausführung des DAA-Befehls an, ob eine Addition oder Subtraktion ausgeführt wurde.

Der Befehl CMP s wird wie SUB s ausgeführt, mit dem Unterschied, daß nur die Flags, aber nicht der Akkumulatorinhalt, beeinflußt werden. Eine Reihe von Befehlen, die auch als Quelle den Akkumulator verwenden, erhalten damit eine besondere Bedeutung.

- ADD A Verdopplung des Akkumulatorinhalts
- ADC A wie ADD A sowie Berücksichtigung des Übertrags
- SUB A lösche A ( $A := 00$ ), setze die Flags  $S := 0$ ;  $Z := 1$ ;  $H := 0$ ;  $P/V := 0$ ;  $N := 1$ ;  $CY := 0$
- SBC A wie SUB A, wenn Übertragsflag  $CY = 0$  vor Befehlsausführung; mit Übertragsflag  $CY = 1$  wird der Akkumulator auf 0FFH gesetzt sowie die Flags auf  $S := 1$ ;  $Z := 0$ ; H ist unbekannt;  $P/V := 1$ ;  $N := 1$ ;  $CY := 1$
- AND A Akkumulatorinhalt bleibt unverändert; Flags werden entsprechend dem Inhalt beeinflußt:  $S := MSB$ ;  $Z := 1$ , wenn  $A = 0$ , sonst  $Z := 0$ ;  $P/V := 1$  bei gerader Parität von A, sonst  $P/V := 0$ ;  $N := 0$ ;  $CY := 0$
- OR A wie AND A
- XOR A lösche Akkumulator ( $A := 00$ ), setze Flags wie AND A
- CMP A Akkumulatorinhalt unverändert, setze Flags wie SUB A.

Da die Befehle AND A und OR A identisch sind, verbleiben in dieser Befehlsgruppe 107 sinnvolle Op-Kode.

**16-bit-Arithmetikbefehle.** Die Befehlstypen dieser Gruppe sind zusammenfassend in Tafel 3.1.5 (S. 54) dargestellt. Die verwendeten Mnemoniks entsprechen prinzipiell denen der 8-bit-Arithmetikbefehle, nur, daß zusätzlich das Ziel der Operation angegeben werden muß. Da der Befehl SUB nicht vorgesehen ist, muß bei Bedarf das Übertragsflag rückgesetzt werden, z. B. durch AND A. Die 16-bit-Arithmetikbefehle lassen sich auf Registerpaare und Doppelregister anwenden.

**Allgemeine Akkumulator- und Flagoperationen.** In dieser Gruppe (Tafel 3.1.6, S. 55) lassen sich fünf Mnemoniks mit folgender Bedeutung einordnen:

- DAA dezimale Anpassung des Akkumulators
- CPL Ergänzung des Akkumulators
- NEG Negation des Akkumulatorinhalts
- CCF negiere Übertragsflag
- SCF setze Übertragsflag.

Die dezimale Anpassung wird im Ergebnis einer Addition oder Subtraktion im gepackten BCD-Format durchgeführt, um wiederum ein gepacktes BCD-Format zu erhalten. Unter diesem Format wird verstanden, daß der 8-bit-Operand als eine zweistellige BCD-Zahl aufzufassen ist. Da die BCD-Ziffer nur die Werte 0 bis 9 zuläßt, müssen die im Ergebnis der Addition oder Subtraktion möglicherweise entstehenden Werte der Ziffer oberhalb 9 wieder in eine gültige BCD-Zahl umgewandelt werden. Die Tetraden oberhalb 9 werden als *Pseudotetraden* bezeichnet, da sie im BCD-Format nicht enthalten sind. Treten z. B. bei einer Addition Pseudotetraden auf, so wird die Ziffer 6 addiert, um den Bereich 10 bis 15 einer Tetrade zu überspringen.

Die Instruktion CPL invertiert den Akkumulatorinhalt (Einerkomplement), die Instruktion NEG negiert den Akkumulatorinhalt (Zweierkomplement). Letzteres entspricht einer Subtraktion des Akkus von Null.

Mit dem Befehl CCF wird der Übertragsflag CY invertiert; mit dem Befehl SCF wird es gesetzt.

**CPU-Steuerbefehle.** Diese Befehlsgruppe (s. ebenfalls Tafel 3.1.6) enthält sechs Steuerbefehle.

Der NOP ist ein Leerbefehl, bei dem keine Operation ausgeführt wird. Der HALT-Befehl unterbricht die Arbeit der CPU, bis ein nachfolgender Interrupt empfangen wird. Die Befehle DI und EI dienen dazu, Interrupts abzuweisen oder zuzulassen. Die drei Interruptartbefehle setzen die CPU in eine der drei möglichen Interruptbetriebsarten. Wenn die CPU einen HALT-Befehl empfangen hat, führt sie kontinuierlich NOP aus, damit ein dynamischer Speicher seine Auffrischung erhält und kein Datenverlust eintritt.

**Rotations- und Verschiebefehle.** Die Befehle dieser Gruppe sind zusammenfassend in Tafel 3.1.7 (S. 56) enthalten.

Diese Befehlsgruppe läßt aufgrund der Möglichkeit der Adressierung aller Register eine Vielfältigkeit der Anwendung zu. Darüber hinaus besteht sogar die Möglichkeit, direkt im Speicher zu verschieben und zu rotieren. Insbesondere bei Multiplikation und Division sind diese Befehle sehr nützlich. Zwei Befehle lassen das Rotieren von Digits im Akkumulator mit einer Speicherstelle zu, die durch das Registerpaar HL adressiert wird. Für BCD-Arithmetik ist diese Art der Rotation zweckmäßig. Die konkreten Verschiebeoperationen sind hierbei aus Tafel 3.1.7 ersichtlich (symbolische Befehlsdarstellung). Die gesamte Gruppe enthält 76 sinnvolle Op-Kode.

**Bitmanipulationsbefehle.** Fast in jedem Programm wird die Möglichkeit genutzt, Bits in einem Register oder auf einem Speicherplatz zu setzen, zu löschen oder zu testen. Diese Bits können Flags in einer allgemein verwendbaren Softwareroutine, Anzeigen von externen Steuerbedingungen oder gepackte Daten im Speicher sein, um die Speicherausnutzung effektiver zu gestalten.

Die CPU U880 ist in der Lage, im Akkumulator, in einem Allgebrauchsregister oder auf einem Speicherplatz mit einem einzigen Befehl ein Bit zu setzen, zu löschen oder zu testen. Die Tafel 3.1.8 (S. 58) enthält die Befehle, die für diesen Zweck zur Verfügung stehen. Die Registeradressierung kann sich auf den Akkumulator oder ein beliebiges Allgebrauchsregister beziehen, mit dem die Operation durchgeführt werden soll. Indirekte Register und indizierte Adressierung stehen für die Arbeit mit den externen Speicherplätzen zur Verfügung. Die Bittestoperation setzt das Nullflag (Z), wenn das getestete Bit eine Null ist (s. auch Abschn. 3.1.5.6.).

Die Gruppe umfaßt 240 sinnvolle Op-Kode.

**Sprungbefehle.** Die Tafel 3.1.9 stellt eine Liste aller Sprung-, Ruf- und Rückkehrbefehle dar, die in der CPU U880 implementiert sind. Ein *Sprung* ist eine *Programmverzweigung*, bei der der Befehlszähler mit dem 16-bit-Wert geladen wird, der durch eine der drei möglichen Adressierungsarten festgelegt wird (erweiterter Direktwert, relative oder indirekte Registeradressierung). Zu beachten ist, daß die Befehle der Sprunggruppe die verschiedensten Bedingungen berücksichtigen können, die entsprechend der Spezifikation erfüllt sein müssen, bevor der Sprung ausgeführt wird. Wenn diese Bedingungen nicht erfüllt sind, wird das Programm mit dem Befehl fortgesetzt, der dem Sprungbefehl als nächster folgt. Die Bedingungen sind alle abhängig von den Daten im Flagregister (s. Abschnitt 3.1.5.6.). Die erweiterte Direktwertadressierung wird benutzt, um zu einem anderen Speicherbereich zu springen. Dieser Befehl erfordert 3 byte (2 davon, um die 16-bit-Adresse zu definieren), wobei das niederwertigere Adressenbyte das erste ist, gefolgt von dem höherwertigen Adressenbyte. Die Sprungbefehle mit relativer Adressierung benötigen nur 2 byte. Das zweite Byte ist das vorzeichenbehaftete Zweierkomplement der Distanz zwischen Zieladresse und dem aktuellen Wert des Befehlszählers (Ablage). Diese

Distanz kann sich in dem Bereich von +127 bis -128 bewegen (bezogen auf die Adresse des Op-Kodes des dem Sprungbefehl folgenden Befehls).

Sprungarten mit indirekter Registeradressierung sind auch vorhanden. Diese Befehle sind so implementiert, daß der Inhalt entweder des HL-Registerpaars oder eines der Indexregister IX und IY direkt in den Befehlszähler geladen wird. Damit sind Programmsprünge in Abhängigkeit von vorangegangenen Berechnungen möglich.

Für die Programmschleifensteuerung kann der Befehl DJNZ e vorteilhaft verwendet werden. Dieser 2 byte lange relative Sprungbefehl erniedrigt das Register B, und es wird gesprungen, solange das Register B ungleich Null ist. Der relative Abstand wird als ein vorzeichenbehaftetes Zweierkomplement ausgedrückt.

In der gesamten Befehlsgruppe bleiben die Flags unbeeinflußt.

**Unterprogrammaufruf- (Ruf-), Rückstart- (Restart-) und Rücksprung- (Rückkehr-) Befehle.** Die dieser Befehlsgruppe zugehörigen Befehle sind zusammenfassend in Tafel 3.1.11 (S. 62) aufgeführt.

Der Unterprogrammaufruf (CALL) ist ein spezieller Sprungbefehl, bei dem die CPU die Adresse des auf den CALL-Befehl folgenden Bytes kellert, bevor der Sprung ausgeführt wird. Der Rückkehrbefehl ist das Gegenteil des CALL-Befehls, weil die Daten aus der obersten Ebene des Stacks direkt in den Befehlszähler geladen werden, um so die Sprungadresse zu bilden. Die Ruf- und Rückkehrbefehle vereinfachen die Unterprogramm- und Interruptbehandlung. Weiterhin sind zwei spezielle Rückkehrbefehle RETI und RETN in der CPU enthalten. Die Rückkehr von einem Interrupt (RETI) und die Rückkehr von einem nicht maskierbaren Interrupt (RETN) werden von der CPU wie *unbedingte* Rückkehrsprünge behandelt. Der Unterschied besteht darin, daß RETI am Ende einer Interruptroutine aufgerufen werden kann, um alle peripheren Chips des Systems durch Dekodierung dieses Befehls an die richtige Steuerung der Interruptbehandlung mit *gestufter* Priorität zu erinnern. Dieser Befehl vereinfacht zusammen mit der Implementierung von peripheren Schaltkreisen des U880-Systems die normale Rückkehr von einem gestuften Interrupt (s. auch Abschn. 4.2.).

Die Instruktionen für Unterprogrammaufruf erfordern aufgrund der unmittelbaren erweiterten Adressierung 3 byte. Neben diesen Befehlen gibt es noch eine spezielle Gruppe, die Rückstarts (RST p). Dafür wird lediglich 1 byte benötigt, da die Rückstartadressen implizit in Op-Kode verschlüsselt sind. Die Flags werden bei dieser Befehlsgruppe ebenfalls nicht beeinflußt.

**Ein- und Ausgabebefehle.** Die CPU U880 hat einen ausgebauten Satz von Ein- und Ausgabebefehlen, der in Tafel 3.1.10 aufgelistet ist. Die Adressierung der Ein- oder Ausgabebefehle kann entweder absolut oder über das Register C indirekt erfolgen. Zu beachten ist, daß die Daten bei der indirekten Registeradressierung zwischen den Ports und einem der internen Register transportiert werden können. Zusätzlich sind acht Blockübertragungsbefehle implementiert worden. Diese Befehle sind den Speicherblocktransportbefehlen ähnlich, mit dem Unterschied, daß sie das Registerpaar HL für die Angabe der Quelladresse (bei Ausgabebefehlen) oder der Zieladresse (bei Eingabebefehlen) benutzen. Das Register B wird als Bytezähler verwendet. Das Register C enthält die Adresse des Kanals, für den der Ein- oder Ausgabebefehl gewünscht ist. Weil das Register B nur eine Länge von 8 bit hat, kann ein Blockübertragungsbefehl bis 256 byte verarbeiten.

In den Befehlen IN n und OUT n erscheint die Adresse des Ports in der unteren Hälfte des Adreßbusses ( $A_0 \dots A_7$ ), und der Akkumulatorinhalt wird in die obere Hälfte des Adreßbusses übergeführt. In allen Befehlen mit indirekter Registeradressierung einschließlich der Blockübertragung wird der Inhalt des Registers C in die untere Hälfte des

Adreßbusses übergeführt (Kanaladresse), wogegen der Inhalt des Registers B in die obere Hälfte des Adressenbusses übergeführt wird.

Die Befehle IN C und OUT C haben eine besondere Bedeutung. Mit der Ausführung von OUT C gibt das mit dem Inhalt des Registers C adressierte Port keine Daten, sondern seine eigene Portadresse aus. IN C bedeutet, daß der Inhalt des Registers C eine Portadresse festlegt und dann durch den Inhalt eben dieses Ports ersetzt wird.

Bei den Befehlstypen IN n, OUT n und OUT r erfolgt keine Beeinflussung der Flags. Die indirekt adressierten Eingabebefehle IN r und INF beeinflussen die Flags entsprechend dem Dateninhalt des Ports. (P/V wird entsprechend der Parität beeinflusst; N := 0.)

Bei den Blockein- und -ausgabebefehlen werden die Flags folgendermaßen beeinflusst:

S	unbekannt
Z := 1	wenn nach Befehlsausführung B=0 gilt, sonst Z := 0
H	unbekannt
P/V	unbekannt
N := 1	
CY	wird nicht verändert.

Die Bedingung Z = 1 wird somit bei den repetierenden Befehlen automatisch als Abbruchbedingung benutzt.

### 3.1.5.5. Zusammenfassende Befehlsangabe

In diesem Abschnitt soll eine zusammenfassende Darstellung des gesamten Befehlssatzes der CPU U880 gegeben werden, unterteilt in Gruppen der Befehlsarten. Die nachstehenden Tafeln 3.1.1 bis 3.1.11 enthalten hierzu folgende Angaben:

- den Mnemonikkode der U880-Assemblersprache,
- die symbolische Operation, die den Datenfluß bzw. die Wirkung der jeweiligen Instruktion darstellt,
- die Beeinflussung des Flagregisters F; hierbei ist die gewählte Darstellungsart identisch mit der, die bei der Zusammenfassung der Flagoperationen im folgenden Abschnitt gewählt wurde (↓ Flag wird durch Befehl beeinflusst; ● Flag wird nicht beeinflusst; X Zustand des Flags ist unbekannt; 0, 1 Flag wird rückgesetzt bzw. gesetzt; P, V P/V-Flag wird durch Parität bzw. durch Überlauf beeinflusst),
- den Op-Kode; der Op-Kode ist dann von Bedeutung, wenn Programme in der Maschinensprache erarbeitet werden (Einarbeitungs- und Lernphase, keine Hilfsmittel wie Assembler od. dgl. vorhanden),
- die Anzahl der Bytes, die durch den jeweiligen Befehl im Programmspeicher belegt werden,
- die Anzahl der Maschinentakte und
- die Anzahl der Taktzyklen, die bei der Befehlsabarbeitung von der CPU benötigt werden.

### 3.1.5.6. Flagoperationen

Jedes der zwei U880-Flagregister enthält sechs Informationsbits, die durch verschiedene CPU-Operationen beeinflusst werden können. Vier dieser Bits können getestet werden, d.h., sie werden als Bedingung für Sprünge, Unterprogrammaufruf- oder Rückkehr-

Tafel 3.1.1. Befehlsübersicht 8-bit-Arithmetik- und Logikbefehle

Mnemonic	Symbolische Operation	Flagregister										Op-Code D <sub>7</sub> D <sub>6</sub> D <sub>5</sub> D <sub>4</sub> D <sub>3</sub> D <sub>2</sub> D <sub>1</sub> D <sub>0</sub>	Hex.	Bytes	M-Zyklen	Takte	Bemerkungen			
		F <sub>7</sub>	F <sub>6</sub>	F <sub>5</sub>	F <sub>4</sub>	F <sub>3</sub>	F <sub>2</sub>	F <sub>1</sub>	F <sub>0</sub>	S	Z							P/V	N	CY
ADD r	A := A+r	↑	↑	X	↑	X	↑	X	V	0	↑	1	0	0	0	r	C6	1	4	
ADD n	A := A+n	↑	↑	X	↑	X	↑	X	V	0	↑	1	1	0	0	0	1	1	0	
ADD M	A := A+M	↑	↑	X	↑	X	↑	X	V	0	↑	1	0	0	0	1	1	0		
ADD (IX+d)	A := A + (IX+d)	↑	↑	X	↑	X	↑	X	V	0	↑	1	1	0	1	0	1			ADD s
ADD (IY+d)	A := A + (IY+d)	↑	↑	X	↑	X	↑	X	V	0	↑	1	1	1	0	1				
ADCs	A := A+s+CY	↑	↑	X	↑	X	↑	X	V	0	↑	1	0	0	1	0				
SUBs	A := A-s	↑	↑	X	↑	X	↑	X	V	1	↑	0	0	1	0					
SBCs	A := A-s-CY	↑	↑	X	↑	X	↑	X	V	1	↑	0	1	1						
ANDs	A := A · s	↑	↑	X	1	X	P	0	0	0	↑	1	0	0						
ORs	A := A + s	↑	↑	X	0	X	P	0	0	0	↑	1	1	0						
XORs	A := A ⊕ s	↑	↑	X	0	X	P	0	0	0	↑	1	0	1						
CMPs	A = s?	↑	↑	X	↑	X	V	1	↑	↑	1	1	1							
INCr	r := r+1	↑	↑	X	↑	X	V	0	-	-	↑	0	0	r	1	0	0			
INCM	M := M+1	↑	↑	X	↑	X	V	0	-	-	↑	0	0	1	1	0	1	0		
INC (IX+d)	(IX+d) := (IX+d) + 1	↑	↑	X	↑	X	V	0	-	-	↑	1	0	1	1	0	1	0		
INC (IY+d)	(IY+d)	↑	↑	X	↑	X	V	0	-	-	↑	1	1	1	1	0	1	0		
DECf	f := f-1	↑	↑	X	↑	X	V	1	-	-	↑	0	0	1	1	0	1	0		

Die hervorgehobenen drei Bits ersetzen 0 0 0 des ADD s-Befehls

Bei dem analogen DEC f-Befehl ist das gekennzeichnete Bit 1

Tafel 3.1.2. Befehlsübersicht 8-bit-Ladebefehle

Mnemonic	Symbolische Operation	Flagregister										Op-Kode								Hex.	Bytes	M-Zyklen	Takte	Bemerkungen							
		F <sub>7</sub>	F <sub>6</sub>	F <sub>5</sub>	F <sub>4</sub>	F <sub>3</sub>	F <sub>2</sub>	F <sub>1</sub>	F <sub>0</sub>	S	Z	H	P/V	N	CY	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>						D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>			
LD r <sub>1</sub> ,r <sub>2</sub>	r <sub>1</sub> := r <sub>2</sub>	-	-	X	-	X	-	-	-	-	-	-	-	-	-	-	-	0	1	r <sub>1</sub>	r <sub>2</sub>	-	-	-	-	-	-	-	1	4	
LD r,n	r := n	-	-	X	-	X	-	-	-	-	-	-	-	-	-	-	-	0	0	r	1	1	0	-	-	-	-	-	2	7	
LD r,M	r := M	-	-	X	-	X	-	-	-	-	-	-	-	-	-	-	-	0	1	r	1	1	0	-	-	-	-	-	2	7	
LD r,(IX+d)	r := (IX+d)	-	-	X	-	X	-	-	-	-	-	-	-	-	-	-	-	1	1	0	1	1	0	1	-	-	-	-	5	19	DD
LD r,(IY+d)	r := (IY+d)	-	-	X	-	X	-	-	-	-	-	-	-	-	-	-	-	0	1	r	1	1	0	-	-	-	-	-	5	19	FD
LD (HL),r	M := r	-	-	X	-	X	-	-	-	-	-	-	-	-	-	-	-	0	1	1	0	r	-	-	-	-	-	-	2	7	
LD (IX+d),r	(IX+d) := r	-	-	X	-	X	-	-	-	-	-	-	-	-	-	-	-	1	1	0	1	1	0	1	-	-	-	-	5	19	DD
LD (IY+d),r	(IY+d) := r	-	-	X	-	X	-	-	-	-	-	-	-	-	-	-	-	0	1	1	0	r	-	-	-	-	-	-	5	19	FD
LD M,n	M := n	-	-	X	-	X	-	-	-	-	-	-	-	-	-	-	-	0	0	1	0	1	1	0	-	-	-	-	3	10	36
LD (IX+d),n	(IX+d) := n	-	-	X	-	X	-	-	-	-	-	-	-	-	-	-	-	1	1	0	1	1	0	1	-	-	-	-	5	19	DD
		-	-	X	-	X	-	-	-	-	-	-	-	-	-	-	-	0	0	1	0	1	1	0	-	-	-	-	5	19	36
		-	-	X	-	X	-	-	-	-	-	-	-	-	-	-	-	d	-	-	-	-	-	-	-	-	-	-			
		-	-	X	-	X	-	-	-	-	-	-	-	-	-	-	-	n	-	-	-	-	-	-	-	-	-	-			

LD (Y+d),n	(Y+d) := n	--	X	X	--	1 1 1 1 1 0 1 0 0 1 1 0 1 1 0	FD 36	4	5	19
LD A, (BC)	A := (BC)	--	X	X	--	0 0 0 0 1 0 1 0	0A	1	2	7
LD A, (DE)	A := (DE)	--	X	X	--	0 0 0 1 1 0 1 0	1A	1	2	7
LD A, (nn)	A := (nn)	--	X	X	--	0 0 1 1 1 0 1 0	3A	3	4	13
LD (BC),A	(BC) := A	--	X	X	--	0 0 0 0 0 0 1 0	02	1	2	7
LD (DE),A	(DE) := A	--	X	X	--	0 0 0 1 0 0 1 0	12	1	2	7
LD (nn),A	(nn) := A	--	X	X	--	0 0 1 1 0 0 1 0	32	3	4	13
LD A,I	A := I	↑ ↓	X	0	X IFF 0	1 1 1 0 1 1 0 1	ED	2	2	9
LD A,R	A := R	↑ ↓	X	0	X IFF 0	0 1 0 1 0 1 1 1	57	2	2	9
LD I,A	I := A	--	X	X	--	0 1 0 1 1 1 1 1	5F	2	2	9
LD R,A	R := A	--	X	X	--	0 1 0 0 0 1 1 1	47	2	2	9
		--	X	X	--	1 1 1 0 1 1 0 1	ED	2	2	9
		--	X	X	--	0 1 0 0 1 1 1 1	4F	2	2	9

Tafel 3.1.3. Befehlsübersicht 16-bit-Ladebefehle

Mnemonic	Symbolische Operation	Flagregister F <sub>7</sub> F <sub>6</sub> F <sub>5</sub> F <sub>4</sub> F <sub>3</sub> F <sub>2</sub> F <sub>1</sub> F <sub>0</sub> S Z H P/V N CY	Op-Kode D <sub>7</sub> D <sub>6</sub> D <sub>5</sub> D <sub>4</sub> D <sub>3</sub> D <sub>2</sub> D <sub>1</sub> D <sub>0</sub>	Hex.	Bytes	M-Zyklen	Takte	Bemerkungen
LD dd,nn	dd := nn	- - X · X · - - - -	0 0 dd 0 0 0 0 1 n		3	3	10	
LD IX,nn	IX := nn	- - X · X · - - - -	1 1 0 1 1 1 0 1 0 0 1 0 0 0 0 1 n	DD 21	4	4	14	
LD IY,nn	IY := nn	- - X · X · - - - -	1 1 1 1 1 1 0 1 0 0 1 0 0 0 0 1 n	FD 21	4	4	14	
LD HL, (nn)	H := (nn+1) L := (nn)	- - X · X · - - - -	0 0 1 0 1 0 1 0 n	2A	3	5	16	
LD dd, (nn)	dd <sub>H</sub> := (nn+1) dd <sub>L</sub> := (nn)	- - X · X · - - - -	1 1 1 0 1 1 0 1 0 1 dd 1 0 1 1 n	ED	4	6	20	
LD IX, (nn)	IX <sub>H</sub> := (nn+1) IX <sub>L</sub> := (nn)	- - X · X · - - - -	1 1 0 1 1 1 0 1 0 0 1 0 1 0 1 0 n	DD 2A	4	6	20	
LD IY, (nn)	IY <sub>H</sub> := (nn+1) IY <sub>L</sub> := (nn)	- - X · X · - - - -	1 1 1 1 1 1 0 1 0 0 1 0 1 0 1 0 n	FD 2A	4	6	20	

LD (nn),HL	(nn+1) := H <sub>H</sub> (nn) := L	0 0 1 0 0 0 1 0	22	3	5	16
LD (nn),dd	(nn+1) := dd <sub>H</sub> (nn) := dd <sub>L</sub>	1 1 1 0 1 1 0 1 0 1 dd 0 0 1 1	ED	4	6	20
LD (nn),IX	(nn+1) := IX <sub>H</sub> (nn) := IX <sub>L</sub>	1 1 0 1 1 1 0 1 0 0 1 0 0 0 1 0	DD 22	4	6	20
LD (nn),IY	(nn+1) := IY <sub>H</sub> (nn) := IY <sub>L</sub>	1 1 1 1 1 1 0 1 0 0 1 0 0 0 1 0	FD 22	4	6	20
LD SP,HL LD SP,IX	SP := HL SP := IX	1 1 1 1 1 0 0 1 1 1 0 1 1 1 0 1	F9 DD	1 2	1 2	6 10
LD SP,IY	SP := IY	1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 1	FD F9	2	2	10
PUSH qq	(SP-2) := qq <sub>L</sub>	1 1 qq 0 1 0 1	F9	1	3	11
PUSH IX	(SP-1) := qq <sub>H</sub> (SP-2) := IX <sub>L</sub>	1 1 0 1 1 1 0 1 1 1 1 0 0 1 0 1	DD E5	2	4	15
PUSH IY	(SP-1) := IX <sub>H</sub> (SP-2) := IY <sub>L</sub>	1 1 1 1 1 1 0 1 1 1 1 1 0 0 1 0	FD E5	2	4	15
POP qq	qq <sub>H</sub> := (SP+1) qq <sub>L</sub> := (SP)	1 1 qq 0 0 0 1	DD	1	3	10
POPIX	IX <sub>H</sub> := (SP+1) IX <sub>L</sub> := (SP)	1 1 0 1 1 1 0 1 1 1 1 0 0 0 0 1	DD E1	2	4	14
POPIY	IY <sub>H</sub> := (SP+1) IY <sub>L</sub> := (SP)	1 1 1 1 1 1 0 1 1 1 1 0 0 0 0 1	FD E1	2	4	14

Tafel 3.1.4. Befehlsübersicht 16-bit-Austauschbefehle, Befehle zur blockweisen Datenübertragung und Überprüfung

Mnemonic	Symbolische Operation	Flagregister													Op-Kode D <sub>7</sub> -D <sub>6</sub> D <sub>5</sub> D <sub>4</sub> D <sub>3</sub> D <sub>2</sub> D <sub>1</sub> D <sub>0</sub>	Hex.	Bytes	M-Zyklen	Takte	Bemerkungen				
		F <sub>7</sub>	F <sub>6</sub>	F <sub>5</sub>	F <sub>4</sub>	F <sub>3</sub>	F <sub>2</sub>	F <sub>1</sub>	F <sub>0</sub>	S	Z	P/V	N	CY										
EX DE,HL	DE := HL	-	-	X	-	X	-	-	-	-	-	-	-	-	1	1	1	0	1	0	1	1	4	
EX AF	AF := AF'	-	-	X	-	X	-	-	-	-	-	-	-	-	1	0	0	0	1	0	0	0	08	
EX X	BC := BC'	-	-	X	-	X	-	-	-	-	-	-	-	-	1	1	0	1	1	0	0	1	D9	
	DE := DE'	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
	HL := HL'	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
EX (SP),HL	H := (SP+1)	-	-	X	-	X	-	-	-	-	-	-	-	-	1	1	1	0	0	0	1	1	E3	19
	L := (SP)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
EX (SP),IX	IX <sub>H</sub> := (SP+1)	-	-	X	-	X	-	-	-	-	-	-	-	-	1	1	0	1	1	0	1	1	DD	23
	IX <sub>L</sub> := (SP)	-	-	-	-	-	-	-	-	-	-	-	-	-	1	1	0	0	0	1	1	1	E3	
EX (SP),IY	IY <sub>H</sub> := (SP+1)	-	-	X	-	X	-	-	-	-	-	-	-	-	1	1	1	1	0	1	0	1	FD	23
	IY <sub>L</sub> := (SP)	-	-	-	-	-	-	-	-	-	-	-	-	-	1	1	0	0	0	1	1	1	E3	
LDI	(DE) := (HL)	-	-	X	0	X	↑	0	-	-	-	-	-	-	1	1	1	0	0	1	0	1	ED	16
	DE := DE+1	-	-	-	-	-	-	-	-	-	-	-	-	-	1	0	1	0	0	0	0	0	A0	
	HL := HL+1	-	-	-	-	-	-	-	-	-	-	-	-	-	1	0	1	0	0	0	0	0	-	
	BC := BC-1	-	-	-	-	-	-	-	-	-	-	-	-	-	1	1	1	0	1	0	1	1	ED	21
LDIR	(DE) := (HL)	-	-	X	0	X	0	0	-	-	-	-	-	-	1	1	0	1	1	0	1	1	B0	wenn BC ≠ 0, wiederhole Befehl; wenn BC = 0;
	DE := DE+1	-	-	-	-	-	-	-	-	-	-	-	-	-	1	0	1	1	0	0	0	0	-	
	HL := HL+1	-	-	-	-	-	-	-	-	-	-	-	-	-	1	0	1	1	0	0	0	0	-	
	BC := BC-1	-	-	-	-	-	-	-	-	-	-	-	-	-	1	1	0	1	1	0	1	1	ED	16
LDD	(DE) := (HL)	-	-	X	0	X	↑	0	-	-	-	-	-	-	1	1	0	1	1	0	1	1	A8	16
	DE := DE-1	-	-	-	-	-	-	-	-	-	-	-	-	-	1	0	1	0	1	0	0	0	-	
	HL := HL-1	-	-	-	-	-	-	-	-	-	-	-	-	-	1	0	1	0	1	0	0	0	-	
	BC := BC-1	-	-	-	-	-	-	-	-	-	-	-	-	-	1	1	1	0	1	1	0	1	ED	21
LDDR	(DE) := (HL)	-	-	X	0	X	0	0	-	-	-	-	-	-	1	1	0	1	1	0	1	1	B8	wenn BC ≠ 0, wiederhole Befehl; wenn BC = 0;
	DE := DE-1	-	-	-	-	-	-	-	-	-	-	-	-	-	1	0	1	1	0	0	0	0	-	
	HL := HL-1	-	-	-	-	-	-	-	-	-	-	-	-	-	1	0	1	1	0	0	0	0	-	
	BC := BC-1	-	-	-	-	-	-	-	-	-	-	-	-	-	1	1	1	0	1	1	0	1	ED	16
CPI	A = (HL)?	↑	↑	X	↑	X	↑	1	-	-	-	-	-	-	1	1	0	1	1	0	1	1	ED	16
	HL := HL+1	-	-	-	-	-	-	-	-	-	-	-	-	-	1	0	1	0	0	0	0	1	A1	4
	BC := BC-1	-	-	-	-	-	-	-	-	-	-	-	-	-	1	0	1	0	0	0	0	1	-	
CPIR	A = (HL)?	↑	↑	X	↑	X	↑	1	-	-	-	-	-	-	1	1	0	1	1	0	1	1	ED	21
	HL := HL+1	-	-	-	-	-	-	-	-	-	-	-	-	-	1	0	1	1	0	0	0	1	B1	wenn A ≠ (HL) und BC ≠ 0, wiederhole Befehl; wenn A = (HL) oder BC = 0;
	BC := BC-1	-	-	-	-	-	-	-	-	-	-	-	-	-	1	0	1	1	0	0	0	1	-	

CPD	A=(HL)? HL := HL-1 BC := BC-1	↑ ↓ ②	X ↑ X	↑ ↓ ①	·	1 1 1 0 1 1 0 1 1 0 1 0 1 0 0 1	ED A9	2	4	16	
CPDR	A = (HL)? HL := HL-1 BC := BC-1	↑ ↓ ②	X ↑ X	↑ ↓ ①	·	1 1 1 0 1 1 0 1 1 0 1 1 1 0 0 1	ED B9	2	5	21	wenn A ≠ (HL) und BC ≠ 0, wiederhole Befehl; wenn A = (HL) oder BC = 0;

① P/V = 0, wenn das Ergebnis von BC-1 = 0, sonst P/V = 1

② Z = 1, wenn Übereinstimmung (A = M), sonst Z = 0

Tafel 3.1.5. Befehlsübersicht 16-bit-Arithmetikbefehle

Mnemonic	Symbolische Operation	Flagregister F <sub>7</sub> F <sub>6</sub> F <sub>5</sub> F <sub>4</sub> F <sub>3</sub> F <sub>2</sub> F <sub>1</sub> F <sub>0</sub> S Z H P/V N CY	Op-Kode D <sub>7</sub> D <sub>6</sub> D <sub>5</sub> D <sub>4</sub> D <sub>3</sub> D <sub>2</sub> D <sub>1</sub> D <sub>0</sub>	Hex.	Bytes	M- Zyklen	Takte	Bemerkungen
ADD HL,dd	HL := HL+dd	- X X X - 0 ↑	0 0 dd 1 0 0 1		1	3	11	
ADC HL,dd	HL := HL+dd+CY	↑ ↑ X X X V 0 ↑	1 1 1 0 1 1 0 1	ED	2	4	15	
SBC HL,dd	HL := HL-dd-CY	↑ ↑ X X X V 1 ↑	0 1 dd 1 0 1 0	ED	2	4	15	
ADD IX,pp	IX := IX+pp	- X X X - 0 ↑	0 1 dd 0 0 1 0	DD	2	4	15	
ADD IY,rr	IY := IY+rr	- X X X - 0 ↑	0 0 pp 1 0 0 1	FD	2	4	15	
INC dd	dd := dd+1	- X - X - - -	0 0 dd 0 0 1 1		1	1	6	
INC IX	IX := IX+1	- X - X - - -	1 1 0 1 1 1 0 1	DD	2	2	10	
INC IY	IY := IY+1	- X - X - - -	0 0 1 0 0 0 1 1	23	2	2	10	
DEC dd	dd := dd-1	- X - X - - -	0 0 1 0 0 0 1 1	FD	2	2	10	
DEC IX	IX := IX-1	- X - X - - -	0 0 dd 1 0 1 1		1	1	6	
DEC IY	IY := IY-1	- X - X - - -	1 1 0 1 1 1 0 1	DD	2	2	10	
			0 0 1 0 1 0 1 1	2B	2	2	10	
			1 1 1 1 1 1 0 1	FD	2	2	10	
			0 0 1 0 1 0 1 1	2B	2	2	10	

Tafel 3.1.6. Befehlsübersicht allgemeine Akkumulator- und Flagbefehle und CPU-Steuerbefehle

Mnemonic	Symbolische Operation	Flagregister F <sub>7</sub> F <sub>6</sub> F <sub>5</sub> F <sub>4</sub> F <sub>3</sub> F <sub>2</sub> F <sub>1</sub> F <sub>0</sub> S Z P/V N CY	Op-Kode D <sub>7</sub> D <sub>6</sub> D <sub>5</sub> D <sub>4</sub> D <sub>3</sub> D <sub>2</sub> D <sub>1</sub> D <sub>0</sub>	Hex.	Bytes	M-Zyklen	Takt	Bemerkungen
DAA	Überführung des A nach Addition oder Subtraktion mit gepackten BCD-Operanden in gepacktes BCD-Format A := $\bar{A}$	$\bar{\uparrow}$ X $\uparrow$ X P - $\uparrow$	0 0 1 0 0 1 1 1	27	1	1	4	Dezimal-korrektur des Akkumulators
CPL	A := $\bar{A}$	- X 1 X - 1 -	0 0 1 0 1 1 1 1	2F	1	1	4	Einer-komplement Zweier-komplement
NEG	A := -A = $\bar{A} + 1$	$\uparrow$ $\uparrow$ X $\uparrow$ X V 1 $\uparrow$	1 1 1 0 1 1 0 1 0 1 0 0 0 1 0 0	ED 44	2	2	8	
CCF	CY := $\bar{CY}$	- X X X - 0 $\uparrow$	0 0 1 1 1 1 1 1	3F	1	1	4	
SCF	CY := 1	- X 0 X - 0 1	0 0 1 1 0 1 1 1	37	1	1	4	
NOP	keine Operation	- X - X - - -	0 0 0 0 0 0 0 0	00	1	1	4	
HALT	CPU im Halt-zustand	- X - X - - -	0 1 1 1 0 1 1 0	76	1	1	4	
DI	IFF <sub>1</sub> := IFF <sub>2</sub> := 0	- X - X - - -	1 1 1 1 0 0 1 1	F3	1	1	4	
EI	IFF <sub>1</sub> := IFF <sub>2</sub> := 1	- X - X - - -	1 1 1 1 1 0 1 1	FB	1	1	4	
IM0	Interruptmode 0	- X - X - - -	1 1 1 0 1 0 1 1 0 1 0 0 0 1 1 0	ED 46	2	2	8	
IM1	Interruptmode 1	- X - X - - -	1 1 1 0 1 1 0 1	ED	2	2	8	
IM2	Interruptmode 2	- X - X - - -	0 1 0 1 0 1 1 0 1 1 1 0 1 1 0 1 0 1 0 1 1 1 1 0	56 ED 5E	2	2	8	

Am Ende der Befehle EI und DI erfolgt keine Abfrage der INT-Linie durch die CPU.

Tafel 3.1.7. Befehlsübersicht Rotations- und Verschiebefehle

Mnemonic	Symbolische Operation	Flagregister F <sub>7</sub> F <sub>6</sub> F <sub>5</sub> F <sub>4</sub> F <sub>3</sub> F <sub>2</sub> F <sub>1</sub> F <sub>0</sub> S Z H P/V N CY	Op-Kode D <sub>7</sub> D <sub>6</sub> D <sub>5</sub> D <sub>4</sub> D <sub>3</sub> D <sub>2</sub> D <sub>1</sub> D <sub>0</sub>	Hex.	Bytes	M-Zyklus	Takte	Bemerkungen
RICA		- - X 0 X · 0 ↑	0 0 0 0 1 1 1	07	1	1	4	
RLA		- - X 0 X · 0 ↑	0 0 0 1 0 1 1	17	1	1	4	
RRCA		- - X 0 X · 0 ↑	0 0 0 0 1 1 1	0F	1	1	4	
RRA		- - X 0 X · 0 ↑	0 0 0 1 1 1 1	1F	1	1	4	
RLC r		↑ ↑ X 0 X P 0 ↑	1 1 0 0 1 0 1 1 0 0 0 0 0 r	CB	2	2	8	
RLC M		↑ ↑ X 0 X P 0 ↑	1 1 0 0 1 0 1 1 0 0 0 0 1 1 0	CB 06	2	4	15	
RLC (IX+d)		↑ ↑ X 0 X P 0 ↑	1 1 0 1 1 1 0 1 1 1 0 0 1 0 1 1 d	DD CB	4	6	23	RLC-s-Befehl
RLC (IY+d)		↑ ↑ X 0 X P 0 ↑	1 1 1 1 1 1 0 1 1 1 0 0 1 0 1 1 d	FD CB	4	6	23	

Operation	Diagram	Op-Code	Op-Code Hex	Op-Code Bin	Op-Code Hex	Op-Code Bin	Op-Code Hex	Op-Code Bin
RLs		01000000	00	0100	0000	0000	0000	0000
RRCs		00000000	00	0000	0000	0000	0000	0000
RRs		01000000	00	0100	0000	0000	0000	0000
SLAs		10000000	00	1000	0000	0000	0000	0000
SRAs		10000000	00	1000	0000	0000	0000	0000
SRLs		11000000	00	1100	0000	0000	0000	0000
RLD		11101101	ED	1110	1101	1101	1101	1101
RRD		01100111	6F	0110	0111	0111	0111	0111

Die ~~gezeichneten~~ drei Bits er-  
von RLCs;  
Zeitverhalten  
und Op-  
Kodes ent-  
sprechen  
denen von  
RLCs

Tafel 3.1.8. Befehlsübersicht Bitmanipulationsbefehle

Mnemonic	Symbolische Operation	Flagregister F <sub>7</sub> F <sub>6</sub> F <sub>5</sub> F <sub>4</sub> F <sub>3</sub> F <sub>2</sub> F <sub>1</sub> F <sub>0</sub> S Z P/V N CY	Op-Kode D <sub>7</sub> D <sub>6</sub> D <sub>5</sub> D <sub>4</sub> D <sub>3</sub> D <sub>2</sub> D <sub>1</sub> D <sub>0</sub>	Hex.	Bytes	M-Zyklen	Takte	Bemerkungen
BIT b,r	$Z := \overline{r_b} \text{ (} b:r \text{)}$	X ↑ X 1 X X 0 -	1 1 0 0 1 0 1 1 0 1 <span style="border: 1px solid black; padding: 0 5px;">b</span> <span style="border: 1px solid black; padding: 0 5px;">r</span>	CB	2	2	8	
BIT b,M	$Z := \overline{M_b}$	X ↑ X 1 X X 0 -	1 1 0 0 1 0 1 1 0 1 <span style="border: 1px solid black; padding: 0 5px;">b</span> 1 1 0	CB	2	3	12	
BIT b,(IX+d)	$Z := \overline{(IX+d)_b}$	X ↑ X 1 X X 0 -	1 1 0 0 1 1 0 1 1 1 0 0 1 0 1 1 <span style="border: 1px solid black; padding: 0 5px;">d</span>	DD CB	4	5	20	
BIT b,(IY+d)	$Z := \overline{(IY+d)_b}$	X ↑ X 1 X X 0 -	0 1 <span style="border: 1px solid black; padding: 0 5px;">b</span> 1 1 0 1 1 1 1 1 0 1 1 1 0 0 1 0 1 1 <span style="border: 1px solid black; padding: 0 5px;">d</span>	FD CB	4	5	20	
SET b,r	$r_b := 1$	- - X - X - - -	0 1 <span style="border: 1px solid black; padding: 0 5px;">b</span> 1 1 0 1 1 0 0 1 0 1 1 1 <span style="border: 1px solid black; padding: 0 5px;">b</span> <span style="border: 1px solid black; padding: 0 5px;">r</span>	CB	2	2	8	
SET b,M	$M_b := 1$	- - X - X - - -	1 1 0 0 1 0 1 1 1 <span style="border: 1px solid black; padding: 0 5px;">b</span> 1 1 0 1 1 0 0 1 0 1 1 <span style="border: 1px solid black; padding: 0 5px;">d</span>	CB	2	4	15	
SET b,(IX+d)	$(IX+d)_b := 1$	- - X - X - - -	1 1 0 0 1 1 0 1 1 1 0 0 1 0 1 1 1 1 0 0 1 0 1 1 <span style="border: 1px solid black; padding: 0 5px;">d</span>	DD CB	4	6	23	SET b,s- Befehl; bei dem analogen RES b,s- Befehl ist das gekenn- zeichnete Bit
SET b,(IY+d)	$(IY+d)_b := 1$	- - X - X - - -	1 <span style="border: 1px solid black; padding: 0 5px;">b</span> 1 1 0 1 1 1 1 1 0 1 1 1 0 0 1 0 1 1 <span style="border: 1px solid black; padding: 0 5px;">d</span>	FD CB	4	6	23	<span style="border: 1px solid black; padding: 0 5px;">0</span>
RES b,s	$s_b := 0$	- - X - X - - -	1 <span style="border: 1px solid black; padding: 0 5px;">b</span> 1 1 0 <span style="border: 1px solid black; padding: 0 5px;">0</span>					

Tafel 3.1.9. Befehlsübersicht Sprungbefehle

Mnemonic	Symbolische Operation	Flagregister										Op-Kode D <sub>7</sub> D <sub>6</sub> D <sub>5</sub> D <sub>4</sub> D <sub>3</sub> D <sub>2</sub> D <sub>1</sub> D <sub>0</sub>	Hex.	Bytes	M- Zyklen	Takte	Bemerkungen	
		F <sub>7</sub> S	F <sub>6</sub> Z	F <sub>5</sub> F <sub>7</sub>	F <sub>4</sub> H	F <sub>3</sub> F <sub>4</sub>	F <sub>2</sub> P/V	F <sub>1</sub> N	F <sub>0</sub> CY									
JMP nn	PC := nn	-	-	X	-	X	-	-	-	-	-	-	1 1 0 0 0 0 1 1	C3	3	3	10	
JPcc nn	wenn Bedingung erfüllt ist: PC := nn, sonst nächster Befehl	-	-	X	-	X	-	-	-	-	-	-	1 1 <span style="border: 1px solid black; padding: 2px;">cc</span> 0 1 0		3	3	10	
JR e	PC := PC + e	-	-	X	-	X	-	-	-	-	-	-	0 0 0 1 1 0 0 0	18	2	3	12	
JRc e	wenn CY = 1: PC := PC + e, sonst nächster Befehl	-	-	X	-	X	-	-	-	-	-	-	0 0 1 1 1 0 0 0	38	2	2	7	wenn CY = 0
JRNc e	wenn CY = 0: PC := PC + e, sonst nächster Befehl	-	-	X	-	X	-	-	-	-	-	-	e-2	30	2	3	12	wenn CY = 1
JRZ e	wenn Z = 1: PC := PC + e, sonst nächster Befehl	-	-	X	-	X	-	-	-	-	-	-	0 0 1 0 1 0 0 0	28	2	2	7	wenn Z = 0
JRNZ e	wenn Z = 0: PC := PC + e, sonst nächster Befehl	-	-	X	-	X	-	-	-	-	-	-	e-2	20	2	3	12	wenn Z = 1
JMP (HL)	PC := HL	-	-	X	-	X	-	-	-	-	-	-	1 1 1 0 1 0 0 1	E9	1	1	4	
JMP (IX)	PC := IX	-	-	X	-	X	-	-	-	-	-	-	1 1 0 1 1 0 1 1	DD	2	2	8	
JMP (IY)	PC := IY	-	-	X	-	X	-	-	-	-	-	-	1 1 1 0 1 0 0 1	E9	2	2	8	
DJNZ e	B := B - 1; wenn B = 0: PC := PC + e, sonst nächster Befehl	-	-	X	-	X	-	-	-	-	-	-	1 1 1 1 1 0 1 1	FD	2	2	8	
		-	-	X	-	X	-	-	-	-	-	-	1 1 1 0 1 0 0 1	E9	2	2	8	wenn B = 0
		-	-	X	-	X	-	-	-	-	-	-	0 0 0 1 0 0 0 0	10	2	3	13	wenn B = 1
		-	-	X	-	X	-	-	-	-	-	-	e-2					

Tafel 3.1.10. Befehlsübersicht Ein- und Ausgabebefehle

Mnemonic	Symbolische Operation	Flagregister F <sub>7</sub> F <sub>6</sub> F <sub>5</sub> F <sub>4</sub> F <sub>3</sub> F <sub>2</sub> F <sub>1</sub> F <sub>0</sub> S Z H P/V N CY	Op-Kode D <sub>7</sub> D <sub>6</sub> D <sub>5</sub> D <sub>4</sub> D <sub>3</sub> D <sub>2</sub> D <sub>1</sub> D <sub>0</sub>	Hex.	Bytes	M- Zyklen	Takte	Bemerkungen
IN n	A := (n)	- X - X - - - -	1 1 0 1 1 0 1 1	DB	2	3	11	
IN r	r := (C)	↑ ↑ X ↑ X P 0 -	1 1 1 0 1 1 0 1	ED	2	3	12	
INF	F wird durch (C) beeinflusst	↑ ↑ X ↑ X P 0 -	0 1 <span style="border: 1px solid black; padding: 0 5px;">r</span> 0 0 0	ED	2	3	12	
INI	M := (C) B := B-1 HL := HL+1	X ↑ X X X 1 - ①	0 1 1 0 1 0 0 0 1 1 1 0 1 0 1 0	70 ED	2	4	16	
INIR	M := (C) B := B-1 HL := HL+1	X 1 X X X 1 -	1 0 1 0 0 0 1 0 1 1 1 0 1 1 0 1	A2 ED	2	5	21	wenn B ≠ 0, wiederhole Befehl; wenn B = 0
IND	M := (C) B := B-1 HL := HL-1	X ↑ X X X 1 - ①	1 1 1 0 1 1 0 1 1 0 1 0 1 0 1 0	ED AA	2 2	4 4	16 16	
INDR	M := (C) B := B-1 HL := HL-1	X 1 X X X 1 -	1 1 1 0 1 1 0 1 1 0 1 1 1 0 1 0	ED BA	2	5	21	wenn B ≠ 0, wiederhole Befehl; wenn B = 0
OUT n	(n) := A	- X - X - - - -	1 1 0 1 0 0 1 1	D3	2	3	11	
OUT r	(C) := r	- X - X - - - -	<span style="border: 1px solid black; padding: 0 5px;">n</span> 1 1 0 1 1 0 1	ED	2	3	12	
OUTI	(C) := M B := B-1 HL := HL+1	X ↑ X X X 1 - ①	0 1 <span style="border: 1px solid black; padding: 0 5px;">r</span> 0 0 1 1 1 1 0 1 1 0 1	ED A3	2	4	16	
OTIR	(C) := M B := B-1 HL := HL+1	X 1 X X X 1 -	1 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1	ED B3	2	5	21	wenn B ≠ 0, wiederhole Befehl; wenn B = 0
					2	4	16	

OUTD	(C) := M B := B - 1 HL := HL - 1	X ↑ ①	X	X	X	X	X	X	1	1	1	0	1	1	0	1	ED AB	2	4	16	
OTDR	(C) := M B := B - 1 HL := HL - 1	X	1	X	X	X	X	X	1	1	1	0	1	1	0	1	ED BB	2	5	21	wenn B ≠ 0, wiederhole Befehl; wenn B = 0

① Z = 1, wenn das Ergebnis von B - 1 = 0, sonst Z = 0

Tafel 3.1.11. Befehlsübersicht Ruf- und Rückkehrbefehle

Mnemonic	Symbolische Operation	Flagregister										Op-Kode								Hex.	Bytes	M-Zyklen	Takte	Bemerkungen						
		F <sub>7</sub>	F <sub>6</sub>	F <sub>5</sub>	F <sub>4</sub>	F <sub>3</sub>	F <sub>2</sub>	F <sub>1</sub>	F <sub>0</sub>	S	Z	P	V	N	CY	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>						D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>		
CALL nn	(SP-1) := PC <sub>H</sub> (SP-2) := PC <sub>L</sub> PC := nn SP := SP-2	X	-	X	-	X	-	-	-	-	-	-	-	-	-	-	-	1	1	0	0	1	1	0	1	CD	3	5	17	
CAcc nn	wenn Bedingung erfüllt ist, wie CALL nn, sonst nächster Befehl	X	-	X	-	X	-	-	-	-	-	-	-	-	-	-	-	1	1	cc	1	0	0	1	0	0	3	3	10	wenn cc falsch ist
RET	PC <sub>L</sub> := (SP) PC <sub>H</sub> := (SP+1) SP := SP+2	X	-	X	-	X	-	-	-	-	-	-	-	-	-	-	-	1	1	0	0	1	0	0	1	C9	1	3	10	wenn cc wahr ist
Rcc	wenn Bedingung erfüllt ist, wie RET, sonst nächster Befehl	X	-	X	-	X	-	-	-	-	-	-	-	-	-	-	-	1	1	cc	0	0	0	0	0		1	1	5	wenn cc falsch ist
RETI	wie RET, jedoch anderer Op-Kode wie RET;	X	-	X	-	X	-	-	-	-	-	-	-	-	-	-	-	1	1	0	1	1	0	1	ED	2	4	14	wenn cc wahr ist für Rückkehr aus INT-ISR	
RETN	IFF <sub>1</sub> := IFF <sub>2</sub>	X	-	X	-	X	-	-	-	-	-	-	-	-	-	-	-	1	1	0	1	1	0	1	4D	2	4	14	für Rückkehr aus NMI-ISR	
RST p	(SP-1) := PC <sub>H</sub> (SP-2) := PC <sub>L</sub> PC <sub>H</sub> := 00 PC <sub>L</sub> := p SP := SP-2	X	-	X	-	X	-	-	-	-	-	-	-	-	-	-	-	1	1	t	1	1	1	1		1	3	11		

befehle verwendet, z. B. kann ein Sprung erwünscht sein, wenn ein spezielles Bit im Flagregister gesetzt ist. Die vier abfragbaren Flagbits sind folgende:

### S (F<sub>7</sub>) Signumflag (Sign)

Dieses Flag ist gedacht für die Verarbeitung von Zahlen mit Vorzeichen. Das Flag wird gesetzt, wenn das Ergebnis der Operation negativ war. Weil das Bit 7 (MSB) das Vorzeichen der Zahl ist (eine negative Zahl trägt eine 1 im Bit 7), speichert dieses Flag den Zustand des Bits 7 des Akkumulators.

Eine Programmverzweigung kann mit Hilfe dieses Flags durch die Bedingungskodes P (Plus, positives Vorzeichen, S = 0) und M (Minus, negatives Vorzeichen, S = 1) erfolgen.

### Z (F<sub>6</sub>) Zeroflag (Zero); Nullflag

Dieses Flag wird gesetzt, wenn im Ergebnis einer Operation im Akkumulator eine Null geladen wird. Andernfalls wird das Bit gelöscht.

Die Programmverzweigung erfolgt über die Bedingungskodes Z (Zero, Ergebnis war Null, Z = 1) und NZ (Non Zero, Ergebnis war nicht Null, Z = 0).

### P/V (F<sub>2</sub>) Paritäts-Überlauf-Flag (Parity/Overflow)

Dieses Flag hat zwei Aufgaben, indem es einerseits die Parität des Ergebnisses im Akkumulator anzeigt, wenn logische Operationen durchgeführt wurden, und andererseits einen Überlauf anzeigt, wenn arithmetische Operationen mit vorzeichenbehafteten Zweierkomplementen durchgeführt wurden. Das U880-Überlaufflag signalisiert, daß ein solches Zweierkomplement fehlerhaft ist, weil es den zulässigen Bereich von max. +127 oder von min. -128, der in der Zweierkomplementschreibweise dargestellt werden kann, überschreitet.

### Beispiele

```

Addition    + 98 = 0 1 1 0 0 0 1 0
             + 76 = 0 1 0 0 1 1 0 0
             -----
CY = 0      1 0 1 0 1 1 1 0 = - 82 falsch!
V = 1.

```

Das Ergebnis ist falsch, obwohl kein Übertragsflag gesetzt wurde.

Für diesen Fall wurde das Überlaufflag gesetzt, so daß per Programm eine entsprechende Korrektur möglich ist.

Das zweite Beispiel zeigt wieder eine Addition, diesmal jedoch von einer positiven und einer negativen Zahl.

```

Addition    + 37 = 0 0 1 0 0 1 0 1
             - 114 = 1 0 0 0 1 1 1 0
             -----
CY = 0      1 0 1 1 0 0 1 1 = - 77 richtig!
V = 0.

```

Bei logischen Operationen (AND, OR, XOR) wird dieses Flagbit 2 gesetzt, wenn das Ergebnis paargig ist, und es wird gelöscht, wenn das Ergebnis unpaargig ist. Programmverzweigungen können bei diesem Flag durch die Bedingungskodes PO (Parity Odd, Parität ist ungerade P/V = 0) und PE (Parity Even, Parität ist gerade P/V = 1) erreicht werden.

### CY (F<sub>0</sub>) Carryflag (Carry); Übertragsflag

Dieses Flag ist der Übertrag vom Akkumulatorbit mit der höchsten Wertigkeit, z. B. wird das Übertragsflag während eines Additionsbefehls gesetzt, wenn ein Übertrag vom höchsten Bit des Akkumulators erzeugt wird. Dieses Flag wird auch gesetzt, wenn bei einer Subtraktion ein negativer Übertrag entsteht. Die Rotations- und Verschiebebefehle beeinflussen dieses Bit ebenfalls.

Die Programmverzweigung erfolgt mit Hilfe der Bedingungskodes C (Carry, Ergebnis hat Übertrag, CY = 1) und NC (Non Carry, Ergebnis hat keinen Übertrag, CY = 0).

Weiterhin sind noch zwei nicht abfragbare Bits im Flagregister vorhanden. Beide werden für die BCD-Arithmetik benutzt. Diese Bits sind:

#### H (F<sub>4</sub>) Halbbyteübertragsflag

Das ist das BCD-Übertragungsergebnis der letzten vier signifikanten Bits der Operation. Wenn der DAA- (Dezimalkorrektur-) Befehl benutzt wird, dient dieses Flag dazu, das Ergebnis der vorausgegangenen gepackten Dezimaladdition oder -subtraktion zu korrigieren.

#### N (F<sub>1</sub>) Additions-Subtraktions-Flag

Weil der Algorithmus für die Korrektur der BCD-Operationen für Addition und Subtraktion unterschiedlich ist, dient dieses Flag der Anzeige der Befehlsart, die ausgeführt wurde, so daß die DAA-Operation die Korrektur entweder für Addition oder Subtraktion durchführen kann.

In Tafel 3.1.12 ist zusammengestellt, wie jedes Flag durch die verschiedenen Befehle beeinflusst wird. In dieser Tafel bedeutet ein ●, daß dieser Befehl das Flag nicht beeinflusst. Ein X heißt, daß das Flag einen unbestimmten Zustand annimmt. Eine 0 bedeutet, daß das Flag gelöscht wird, eine 1, daß das Flag gesetzt wird. Das Symbol ↓ bedeutet, daß das Flag entsprechend der vorangegangenen Beschreibung gesetzt wird.

Ein Befehl, der in dieser Liste nicht erscheint, beeinflusst kein Flag.

In dieser Tafel sind auch einige Spezialfälle enthalten, die der Klarheit wegen beschrieben werden müssen. Der Blocksuchbefehl setzt das Z-Flag, wenn bei der letzten Vergleichsoperation eine Übereinstimmung zwischen den Speicherdaten und dem Akkumulator gefunden wird. Es wird auch das Paritätsflag gesetzt, wenn der Bytezähler (Register BC) ungleich Null ist. Gleichermaßen wird auch das Paritätsflag in den Blocktransportbefehlen verwendet. Ein anderer Spezialfall tritt während der Blockein- und -ausgabebefehle auf. Hier wird das Z-Flag benutzt, um den Zustand des Registers B anzuzeigen, das als Bytezähler benutzt wird. Wenn der I/O-Blocktransport beendet ist, wird das Zeroflag auf Null gesetzt (d. h. B = 0), während im Fall des Blocktransportbefehls das Paritätsflag zum Ende der Operation gelöscht wird. Ein letzter Spezialfall tritt auf, wenn das Auffrisch- oder I-Register in den Akkumulator geladen wird. Dann wird gleichzeitig das Interrupt-freigabeflipflop in das Paritätsflag geladen, so daß der vollständige Zustand der CPU jederzeit gerettet werden kann.

### 3.1.5.7. Analyse der Befehle nach Maschinenzyklen

In diesem Abschnitt werden die Befehle nach Maschinenzyklen, den dabei erfolgten Aufgaben und den entsprechend verwendeten Taktzuständen analysiert. Im Abschn. 3.1.3. ist das Zeitverhalten der unterschiedlichen Maschinenzyklen beschrieben.

Die nachstehende Tafel 3.1.13 gibt für alle Befehlstypen der U880-Assemblersprache einen Überblick über die bei der Befehlsabarbeitung durchgeführten Einzelaktionen.

Tafel 3.1.12. Zusammenfassung der Flagoperationen

Befehlsgruppe	Flagregister								Bemerkungen
	F <sub>7</sub> S	F <sub>6</sub> Z	F <sub>5</sub>	F <sub>4</sub> H	F <sub>3</sub>	F <sub>2</sub> P/V	F <sub>1</sub> N	F <sub>0</sub> CY	
LD A,I; LD A,R	↑	↑	X	0	X	IFF	0	*	P/V := IFF <sub>2</sub>
LDI; LDD	X	X	X	0	X	↓	0	*	
LDIR; LDDR	X	X	X	0	X	0	0	*	Blocktransferbefehle: P/V=1, wenn BC≠0, sonst P/V=0
CPI; CPD; CPIR; CPDR	X	↑	X	X	X	↓	1	*	
ADD s; ADC s	↑	↑	X	↑	X	V	0	↑	8-bit-Arithmetikbefehle
SUB s; SBC s; CMP s; NEG	↑	↑	X	↑	X	V	1	↑	
AND s	↑	↑	X	1	X	P	0	0	logische Befehle
OR s; XOR s	↑	↑	X	0	X	P	0	0	
INC s	↑	↑	X	↑	X	V	0	*	8-bit-Inkrement und Dekrement
DEC s	↑	↑	X	↑	X	V	1	*	
ADD HL,dd; ADD IX,pp; ADD IY,rr			X	X	X		0	↑	16-bit-Arithmetikbefehle
ADC HL,dd	↑	↑	X	X	X	V	0	↑	
SBC HL,dd	↑	↑	X	X	X	V	1	↑	Rotation des Akkumulators
DAA	↑	↑	X	↑	X	P	*	↑	
CPL	*	*	X	1	X	*	1	*	Gruppe der Rotations- und Verschiebefehle
CCF	*	*	X	X	X	*	0	↑	
SCF	*	*	X	0	X	*	0	1	Rotation des Akkumulators
RLCA; RLA; RRCA; RRA	*	*	X	0	X	*	0	↑	
RLC s; RRC s; RL s; RR s; SLA s; SRA s; SRL s	↑	↑	X	0	X	P	0	↑	Gruppe der Rotations- und Verschiebefehle
RLD; RRD	↑	↑	X	0	X	P	0	*	
BIT b,s	X	↑	X	1	X	X	0	*	Digitrotation Z := $\overline{s_b}$
IN r; INF	↑	↑	X	0	X	P	0	*	
INI; IND; OUTI; OUTD	X	↑	X	X	X	X	1	*	blockweise Ein- bzw. Ausgabe: Z=0, wenn B≠0, sonst Z=1
INIR; INDR; OTIR; OTDR	X	1	X	X	X	X	1	*	

Hierzu enthält die Tafel Angaben über

- den jeweiligen Befehlstyp
- die Anzahl der Bytes im Programmspeicher sowie
- die Aktivitäten in den Maschinentakten M1 bis M5.

Die dargestellten Einzelaktivitäten haben hierbei folgende Bedeutung:

- IOP** interne CPU-Operation (Steuersignalausgänge sind inaktiv)  
**SBL** Lesen des Steuerbytes des Op-Kodes (ED, DD, FD, CB)

<b>OKL</b>	Lesen des Op-Kodes
<b>MRD</b>	8-bit-Speicherlesen (Memory Read)
<b>MRH</b>	16-bit-Speicherlesen, höherwertigeres Byte
<b>MRL</b>	16-bit-Speicherlesen, niederwertigeres Byte
<b>MWR</b>	8-bit-Speicherschreiben (Memory Write)
<b>MWH</b>	16-bit-Speicherschreiben, höherwertigeres Byte
<b>MWL</b>	16-bit-Speicherschreiben, niederwertigeres Byte
<b>OPD</b>	Lesen von 8-bit-Operandendaten (Einbyteoperanden)
<b>ODH</b>	Lesen von 16-bit-Operandendaten, höherwertigeres Byte (Zweibyteoperanden, H)
<b>ODL</b>	Lesen von 16-bit-Operandendaten, niederwertigeres Byte (Zweibyteoperanden, L)
<b>PRD</b>	Portlesen; Eingabe (Port Read)
<b>PWR</b>	Portschreiben; Ausgabe (Port Write)
<b>SRH</b>	Stacklesen, höherwertigeres Byte
<b>SRL</b>	Stacklesen, niederwertigeres Byte
<b>SWH</b>	Stackschreiben, höherwertigeres Byte
<b>SWL</b>	Stackschreiben, niederwertigeres Byte
<b>INTA</b>	Interruptquittierungszyklus der CPU.

Die Tafel enthält ferner zusätzliche Angaben:

<b>SPI</b>	Inkrementieren des Stackpointers SP am Ende des Zyklus
<b>SPD</b>	Dekrementieren des Stackpointers SP am Ende des Zyklus
<b>(n)</b>	n gibt die bei der jeweiligen Aktivität benutzten Taktzustände der CPU an.

Der Vollständigkeit wegen sind am Ende der Tafel 3.1.13 die Aktivitäten des Prozessors bei der Interruptquittierung dargestellt.

### Beispiele

Anhand einiger Beispiele soll diese Tafel erläutert werden. Als erstes der Beispiele sei der einfache, aber recht häufig verwendete Befehl des Ladens eines Registers mit dem Inhalt eines Registers  $LD\ r_1, r_2$  analysiert.  $r_1$  und  $r_2$  können jeweils eines der 8-bit-Register A, B, C, D, E, H oder L spezifizieren. Aus der Tafel ist zu entnehmen, daß der Befehl einen Maschinenzklus M1 mit vier Taktzuständen umfaßt. Selbstverständlich erfolgte die gesamte Analyse bezüglich der notwendigen Taktzustände ohne Berücksichtigung möglicher WAIT-Zustände, die entsprechend der Einfügung in den Befehlsablauf zum Zeitaufwand zu addieren sind.

Der nächste betrachtete Befehl sei das Laden eines Registers mit einer Konstanten n, die als unmittelbarer Wert im Befehl enthalten ist und dem Op-Kode folgt. Daher findet man in der Tafel zwei Maschinenzyklen – wiederum den M1-Zyklus zum Op-Kode-Holen und einen M2-Zyklus zum Speicherlesen des Wertes n. Dieser zweite Zyklus erfordert drei Taktzustände. Nach diesen zwei einfachen Befehlen ist u. a. in der Tafel der umfangreichere Befehl  $LD\ r, (IX + d)$  angegeben. Da mit 3 bit des Op-Kodes nur die Register A bis L sowie der Speicher M spezifiziert werden können, wird ein zusätzliches Steuerwort, in diesem Fall DD, erforderlich, um die indizierte Befehlsausführung zu veranlassen. Das Lesen des Steuerworts geschieht wie das Holen des Op-Kodes in einem M1-Zyklus mit zugehörigem Refresh. Ähnlich wie bei der Ausführung von  $LD\ r, n$  wird nach dem Holen des Op-Kodes der Operand d, die Ablage, gelesen. Der nächste Maschinenzklus M3, bestehend aus fünf Taktzuständen, stellt eine interne CPU-Operation dar und dient der Berechnung der aktuellen Speicheradresse  $IX + d$  durch Addition des Inhalts von 16-bit-Register IX und der 8-bit-Ablage d. Anschließend wird als letzter Maschinen-

Tafel 3.1.13. Analyse der Befehlstypen nach Aktivitäten

Befehlstyp	Bytes	M1-Zyklen	M2-Zyklen	M3-Zyklen	M4-Zyklen	M5-Zyklen	Bemerkungen
LD r <sub>1</sub> , r <sub>2</sub>	1	OKL (4)					
LD r, n	2	OKL (4)	OPD (3)				
LD r, M	1	OKL (4)	MRD (3)				
LD M, r	1	OKL (4)	MWR (3)				
LD r, (ii+d)	3	SBL (4); OKL (4)	OPD (3)	IOP (5)	MRD (3)		ii: eines der Indexregister IX oder IY
LD (ii+d), r	3	SBL (4); OKL (4)	OPD (3)	IOP (5)	MWR (3)		
LD M, n	2	OKL (4)	OPD (3)	MWR (3)			
LD A, (BC); LD A, (DE)	1	OKL (4)	MRD (3)				
LD (BC), A; LD (DE), A	1	OKL (4)	MWR (3)				
LD A, (nn)	3	OKL (4)	ODL (3)	ODH (3)	MRD (3)		
LD (nn), A	3	OKL (4)	ODL (3)	ODH (3)	MWR (3)		
LD A, i	2	SBL (4); OKL (5)					
LD i, A	2	SBL (4); OKL (5)					
LD dd, nn	3	OKL (4)	ODL (3)	ODH (3)			
LD ii, nn	4	SBL (4); OKL (4)	ODL (3)	ODH (3)			
LD HL, (nn)	3	OKL (4)	ODL (3)	ODH (3)		MRH (3)	
LD (nn), HL	3	OKL (4)	ODL (3)	ODH (3)	MRL (3)	MWH (3)	
LD dd, (nn)	4	SBL (4); OKL (4)	ODL (3)	ODH (3)	MRL (3)	MRH (3)	
LD (nn), dd	4	SBL (4); OKL (4)	ODL (3)	ODH (3)	MWL (3)	MWH (3)	
LD ii, (nn)	4	SBL (4); OKL (4)	ODL (3)	ODH (3)	MRL (3)	MRH (3)	
LD (nn), ii	4	SBL (4); OKL (4)	ODL (3)	ODH (3)	MWL (3)	MWH (3)	
LD SP, HL	1	OKL (6)					
LD SP, ii	2	SBL (4); OKL (6)					ii: IX oder IY
PUSH qq	1	OKL (5); SPD	SWH (3); SPD	SWL (3)			
PUSH ii	2	SBL (4); OKL (5); SPD	SWH (3); SPD	SWL (3)			ii: IX oder IY
POP qq	1	OKL (4)	SRL (3); SPI	SRH (3); SPI			
POP ii	2	SBL (4); OKL (4)	SRL (3); SPI	SRH (3); SPI			ii: IX oder IY
EX DE, HL	1	OKL (4)					
EX AF; EXX	1	OKL (4)					
EX (SP), HL	1	OKL (4)	SRL (3); SPI	SRH (4)	SWH (3); SPD	SWL (5)	
EX (SP), ii	2	SBL (4); OKL (4)	SRL (3); SPI	SRH (3)	SWH (3); SPD	SWL (5)	ii: IX oder IY
LDI; LDD; CPI; CPD	2	SBL (4); OKL (4)	MRD (3)	MWR (5)			

Tafel 3.1.1.3. (Fortsetzung)

Befehlstyp	Bytes	M1-Zyklen	M2-Zyklen	M3-Zyklen	M4-Zyklen	M5-Zyklen	Bemerkungen
LDIR; LDDR; CPIR; CPDR	2	SBL (4); OKL (4)	MRD (3)	MWR (5)	IOP (5)		IOP (5) in M4 nur, wenn BC≠0 ALU steht für einen der Mnemoniks: ADD, ADC, SUB, SBC, AND, OR, XOR, CMP
ALU r	1	OKL (4)	OPD (3)				
ALU n	2	OKL (4)	MRD (3)				
ALU M	1	OKL (4)	OPD (3)		MRD (3)		
ALU (ii+d)	3	SBL (4); OKL (4)					
INC r; DEC r	1	OKL (4)					
INC M; DEC M	1	OKL (4)	MRD (3)				
INC (ii+d); DEC (ii+d)	3	SBL (4); OKL (4)	OPD (3)		MRD (4)	MWR (3)	ii: IX oder IY
DAA	1	OKL (4)					
CPL	1	OKL (4)					
NEG	2	SBL (4); OKL (4)					
CCF; SCF	1	OKL (4)					
NOP; HALT	1	OKL (4)					
DI; EI	1	OKL (4)					
IM0; IM1; IM2	2	SBL (4); OKL (4)					
ADD HL, dd	1	OKL (4)	IOP (4)				ADD ii, pp steht für ADD IX, pp oder ADD IY, rr
ADCHL, dd; SBC HL, dd; ADD ii, pp	2	SBL (4); OKL (4)	IOP (4)				ii: IX oder IY
INC dd; DEC dd	1	OKL (6)					
INC ii; DEC ii	2	SBL (4); OKL (6)					
RLCA; RRCA; RLA; RRA	1	OKL (4)					
ROT r	2	SBL (4); OKL (4)	MRD (3)				ROT steht für einen der Mnemoniks: RLC, RL, RRC, RR, SLA, SRA, SRL
ROT M	2	SBL (4); OKL (4)	OPD (3)				
ROT (IX+d)	4	SBL (4); SBL (4)	OPD (3)		MRD (4)	MWR (3)	
ROT (IY+d)	4	SBL (4); SBL (4)	OPD (3)		MRD (4)	MWR (3)	
RLD; RRD	2	SBL (4); OKL (4)	MRD (3)				
BIT b, r; SET b, r; RES b, r	2	SBL (4); OKL (4)					
BIT b, M	2	SBL (4); OKL (4)	MRD (4)				
SET b, M; RES b, M	2	SBL (4); OKL (4)	MRD (4)				
BIT b, (ii+d)	4	SBL (4); SBL (4)	OPD (3)				
SET b, (ii+d); RES b, (ii+d)	4	SBL (4); SBL (4)	OPD (3)				ii: IX oder IY

JMP nn; JPcc nn	3	OKL (4)	ODL (3)	ODH (3)	IOP (5) in M3 nur, wenn Bedingung erfüllt ist
JR c	2	OKL (4)	OPD (3)	IOP (5)	ii: IX oder IY
JRC e; JRZ e; JRNC e;	2	OKL (4)	OPD (3)	IOP (5)	IOP (5) nur, wenn B ≠ 0
JRNZ c	1	OKL (4)	OPD (3)	IOP (5)	cc ist erfüllt
JMP (HL)	2	SBL (4); OKL (4)	OPD (3)	IOP (5)	cc ist nicht erfüllt
JMP (H)	2	OKL (5)	OPD (3)	IOP (5)	cc ist erfüllt
DJNZ c	2	OKL (4)	ODL (3)	ODH (4); SPD	cc ist nicht erfüllt
CALL nn	3	OKL (4)	ODL (3)	ODH (4); SPD	
CAcc nn	3	OKL (4)	ODL (3)	ODH (4); SPD	
CAcc nn	3	OKL (4)	ODL (3)	ODH (3)	
RET	1	OKL (4)	SRL (3); SPI	SRH (3); SPI	
Rcc	1	OKL (5)	SRL (3); SPI	SRH (3); SPI	
Rcc	1	OKL (5)	SRL (3); SPI	SRH (3); SPI	
RETI; RETN	2	SBL (4); OKL (4)	SRL (3); SPI	SRH (3); SPI	
RST p	1	OKL (5); SPD	SWH (3)	SWL (3)	
IN n	2	OKL (4)	OPD (3)	PRD (4)	
IN r; INF	2	SBL (4); OKL (4)	PRD (4)	MWR (3)	
INI; IND	2	SBL (4); OKL (5)	PRD (4)	MWR (3)	IOP (5) nur, wenn B ≠ 0
INIR; INDR	2	SBL (4); OKL (5)	PRD (4)	MWR (3)	
OUT n	2	OKL (4)	OPD (3)	PWR (4)	
OUT r	2	SBL (4); OKL (4)	PWR (4)	PWR (4)	
OUTI; OUTD	2	SBL (4); OKL (5)	MRD (3)	PWR (4)	
OTIR; OTDR	2	SBL (4); OKL (5)	MRD (3)	PWR (4)	
NMI	-	OKL (5); SPD	SWH (3); SPD	SWL (3)	Interruptquittierung
INT, Mode 0	-	INTA (6); SPD	SWH (3); SPD	SWL (3)	Beispiel: RST wird als Op-Kode gelesen
	-	INTA (6)	ODL (3)	ODH (4); SPD	Beispiel: CALL wird als Op-Kode gelesen
INT, Mode 1	-	INTA (7); SPD	SWH (3); SPD	SWH (3); SPD	ähnlich RST 38H
INT, Mode 2	-	INTA (7); SPD	SWH (3); SPD	MRL (3)	indirekt adressierter Rufbefehl: CALL (IP)
	-			MRH (3)	IP: Interruptpointer

zyklus M4 ausgeführt, der ein Speicherlesen von  $(IX + d)$  und das Laden des Registers r mit eben diesem erhaltenen Wert bewirkt. Dafür sind nochmals drei Taktzustände erforderlich, so daß die gesamte Befehlsausführung aus 19 Taktzuständen besteht.

Der Befehl LD (nn),HL bewirkt das Laden des Registerpaars HL sukzessive in die Speicherzellen nn und  $nn + 1$ . Dafür sind fünf Maschinentzyklen mit 16 Taktzuständen erforderlich. Der Op-Kode wird wieder im M1-Zyklus geholt. Anschließend erfolgt in zwei Zyklen, M2 und M3, das Lesen der Adresse nn in der Reihenfolge niederwertiger und danach höherwertiger Adreßteil. Danach wird Register L in den Speicher geschrieben, gefolgt von Register H.

Der M1-Zyklus dient dem Op-Kode-Holen und erfordert i. allg. vier Taktzustände, wenn nicht gleichzeitig im Zyklus weitere Aufgaben zu erfüllen sind. Ein Beispiel dafür ist die Instruktion PUSH qq. Im fünften Taktzustand wird der Stackpointer dekrementiert, da der externe Speicher als LIFO (last-in, first-out) arbeitet und der Stackpointer immer auf die aktuelle Spitze des Speicherbereichs zeigt. Durch das Dekrementieren wird ein freier Speicherplatz adressiert. Auf dem Stack werden 16-bit-Registerpaare bzw. Doppelregister abgelegt. Da das Lesen in der Reihenfolge niederwertiger Teil-höherwertiger Teil erfolgen muß, wird das Schreiben in der umgekehrten Reihenfolge durchgeführt, so daß dem Lesen des Op-Kodes zweimal das Schreiben des Stackes mit jeweils drei Taktzuständen folgt.

Eine Reihe von Befehlen läßt sich blockweise anwenden. Dafür wird dem Mnemonik ein R (für repetierend) zugefügt. Aus der Darstellung der Tafel ist zu entnehmen, daß die Maschinenbefehle mit einem Steuerwort beginnen, so daß nach Lesen des Op-Kodes der PC zweifach inkrementiert wurde. Die zyklische Ausführung erfolgt dadurch, daß in einer internen Operation (M4) der PC wieder zweifach dekrementiert und am Ausgangspunkt neu gestartet wird. Im Maschinentzyklus M3 ist es erforderlich, daß das Doppelregister BC dekrementiert und die Entscheidung getroffen wird, ob die zyklische Ausführung zu beenden ist. Aus der üblichen Befehlsausführung fällt die Bearbeitung von Interrupts etwas heraus. Der NMI-Befehl beginnt in der Ausführung mit einem erweiterten M1-Zyklus, ähnlich wie bei PUSH qq. Der gelesene Op-Kode wird ignoriert; der PC wird nach Rettung auf 66H gesetzt als Rückstartinstruktion. Die Interruptquittierung für NMI erfordert 11 Taktzustände.

Ähnlich wie bei NMI arbeitet die Mode IM0 mit extern eingefügtem Rückstartbefehl oder die Mode IM1 mit intern bestimmtem Rückstart ab Adresse 38H. Der Unterschied besteht darin, daß kein Holen eines Op-Kodes, sondern eine Bestätigung des Interrupts ausgeführt wird.

Die Mode IM2 wird genutzt in Verbindung mit den speziellen peripheren Schaltkreisen des Systems U880. Dafür werden folgende Operationen durchgeführt:

Aus der Tafel ist zu entnehmen, daß die Operationen mit einer Interruptbestätigung beginnen. Das ist dadurch gekennzeichnet, daß gleichzeitig  $\overline{IORQ}$  und  $\overline{MI}$  aktiv sind. Dieser Zustand wird von den peripheren Schaltkreisen dekodiert, und der relevante, den Interrupt anmeldende Schaltkreis liefert einen vorgegebenen, d.h. diesem vorher einprogrammierten Interruptvektor auf den Datenbus. Nach der obligatorischen Rettung des PC bei Interrupt wird durch den Pointer, dessen niederwertiger Teil der gelieferte Vektor und dessen höherwertiger Teil das I-Register bilden, der Speicher adressiert und 2 byte gelesen, die als Adresse der Interruptserviceroutine in den PC geladen werden.

### 3.1.6. Interruptverhalten

#### 3.1.6.1. Zweck des Interrupts

Ein Interrupt ist eine Unterbrechung eines gerade ausgeführten Programms, damit der Prozessor auf bestimmte Forderungen reagieren kann. Damit wird ermöglicht, daß mehrere Programme durch eine CPU abgearbeitet werden können, wenn es die äußeren Umstände erfordern. Man lenkt die CPU durch geschachtelte Interrupts und geeignete Priorisierung auf die zum Zeitpunkt jeweils wichtigste Aufgabe. Gegenüber einer zyklischen Abfrage, ob eine Aufgabe zu erfüllen ist, werden Zeit und Speicherraum eingespart, da sofort reagiert werden kann und die Latenzzeit nicht durch die Abfrageperiode bestimmt ist. Die **zyklische Abfrage** wird auch als *Polling* bezeichnet. Die durch die verschiedenen Interruptvarianten gegebenen Möglichkeiten haben einen effektiven Einfluß auf die Leistungsfähigkeit eines Mikroprozessorsystems.

#### 3.1.6.2. Sperrung und Freigabe

Für die Bearbeitung von Unterbrechungsanforderungen sind bei der CPU U880 zwei Interrupteingänge vorhanden. Diese unterscheiden sich durch Priorität und Maskierbarkeit. Der NMI (nicht maskierbarer Interrupt) besitzt die höchste Interruptpriorität und kann per Programm nicht gesperrt werden, so daß die CPU unbedingt ihr laufendes Programm unterbricht, wenn irgendeine periphere Einheit diesen Interrupt anfordert. Im allgemeinen ist dieser Interrupt für sehr wichtige Ereignisse, z. B. Ausfälle, Havarien und andere nichtvorhersehbare Geschehnisse, reserviert.

Der INT (maskierbarer Interrupt) kann per Programm gezielt gesperrt und freigegeben werden. Das kann z. B. notwendig sein, wenn zwar eine Interruptserviceroutine niedriger Priorität abläuft, aber eine Unterbrechung bestimmter Programmteile aufgrund des geforderten Echtzeitverhaltens nicht tragbar ist. Dann wird gerade dieser Programmteil durch Sperrung des maskierbaren Interrupts geschützt. Freigabe bzw. Sperrung werden in einem Flipflop zwischengespeichert. Durch die Befehle EI (ermögliche Interrupt) und DI (sperrt Interrupt) wird dieses Flipflop entsprechend gesetzt bzw. rückgesetzt.

Aus bestimmten noch zu erläuternden Gründen sind in der CPU U880 zwei Freigabe-Flipflop enthalten, die IFF<sub>1</sub> und IFF<sub>2</sub> genannt werden. Dabei bildet IFF<sub>1</sub> das Flipflop, das den Befehlsablauf direkt beeinflußt.

In bestimmten Situationen ist das Zwischenspeichern des aktuellen Inhalts von IFF<sub>1</sub> erforderlich. Dafür ist IFF<sub>2</sub> vorgesehen.

Ein Rücksetzen der CPU bewirkt, daß sowohl IFF<sub>1</sub> als auch IFF<sub>2</sub> zurückgesetzt werden, so daß die Annahme von Interruptanforderungen gesperrt ist. Durch den Befehl EI kann die Freigabe von maskierbaren Interrupts bewirkt werden. Nach Ausführung des Befehls EI wird die Annahme eines angemeldeten Interrupts so lange gesperrt, bis der dem Befehl EI folgende Befehl ausgeführt ist. Wenn ein Interrupt angenommen wird, werden sowohl IFF<sub>1</sub> als auch IFF<sub>2</sub> zurückgesetzt, um weitere Unterbrechungen zu verhindern, bis in der Interruptserviceroutine eine Freigabe durch Ausführung des Befehls EI erfolgt. Der Zweck der Verzögerung der Freigabe für die Ausführung eines Befehls ist, zu sichern, daß nach der Freigabe noch ein Rücksprung aus der Interruptserviceroutine (RETI oder RETN) möglich ist. Damit wird ein Einschachteln mehrerer Interruptserviceroutinen vermieden.

In allen bisherigen Fällen war der Inhalt von IFF<sub>1</sub> und IFF<sub>2</sub> identisch. Bei der Quittierung eines nichtmaskierbaren Interrupts wird einerseits wiederum IFF<sub>1</sub> rückgesetzt, um maskierbare Interruptforderungen zu unterbinden; andererseits wird aber der Inhalt von

IFF<sub>2</sub> nicht verändert. Damit bleibt in der CPU der Interruptfreigabezustand, der vor Einschachtelung der NMI-Bearbeitungsroutine aktuell war, erhalten. Er kann nach Beendigung der NMI-Routine wiederhergestellt werden. Hierzu dient der Rückkehrbefehl RETN, bei dessen Ausführung u. a. IFF<sub>2</sub> in IFF<sub>1</sub> kopiert wird. Eine weitere Möglichkeit der Testung des IFF<sub>2</sub> bieten die CPU-Befehle LD A,I und LD A,R. Bei diesen Befehlen wird der Zustand von IFF<sub>2</sub> in das Paritätsflag P/V geladen. Er ist somit testbar und kann für Programmverzweigungen verwendet werden.

In Tafel 3.1.14 sind zusammenfassend alle Aktionen dargestellt, die einen Einfluß auf den Zustand der Interruptfreigabeflipflops haben.

Tafel 3.1.14. Beeinflussung der Interruptfreigabeflipflops

CPU-Aktion	IFF <sub>1</sub>	IFF <sub>2</sub>	Bemerkungen
Rücksetzen der CPU	0	0	
Befehl DI	0	0	
Befehl EI	1	1	
Befehl LD A, I	*	*	P/V := IFF <sub>2</sub>
Befehl LD A, R	*	*	P/V := IFF <sub>2</sub>
NMI-Quittierung	0	*	
Rückkehrbefehl RETN	IFF <sub>2</sub>	*	IFF <sub>1</sub> := IFF <sub>2</sub>
INT-Quittierung	0	0	
Rückkehrbefehl RETI	*	*	

### 3.1.6.3. Verhalten der CPU

Neben den beiden Interrupteingängen besitzt die CPU U880 den BUSRQ-Steuereingang für die Busfreigabesteuerung, der insbesondere bei DMA-Betrieb zur Anwendung kommt. Die CPU reagiert auf Unterbrechungsfordernngen über diese drei Steuereingänge (NMI, INT, BUSRQ) mit folgender Rangfolge:

1. Busanforderung mittels BUSRQ
2. nichtmaskierbarer Interrupt NMI
3. maskierbarer Interrupt INT.

Eine Busanforderung kann hierbei von der CPU am Ende eines jeden Maschinenzklus bestätigt werden. Demgegenüber erfolgt die Abfrage der Interruptlinien NMI und INT nur jeweils am Ende jedes letzten Maschinenzklus eines Befehls.

Bei der Bestätigung einer Interruptforderung (NMI bzw. INT) fügt der Prozessor einen Quittierungszyklus in die Befehlsbearbeitung ein. Die hierbei durchgeführten Aktivitäten sind am Ende der Tafel 3.1.13 aufgeführt. Bild 3.1.12 zeigt zusammenfassend das CPU-Verhalten bei Unterbrechungsfordernngen im Flußbild.

Das Zeitverhalten des Prozessors bei einer Busanforderung ist im Bild 3.1.8 dargestellt. Die Einbeziehung des DMA-Betriebs in das Gesamtsystem ist im Abschn. 4.3. (S. 199) enthalten. Die CPU-Aktivitäten bei den Interruptbetriebsarten sind nachfolgend zusammengefaßt.

**Nichtmaskierbarer Interrupt.** Ein nichtmaskierbarer Interrupt wird jederzeit durch die CPU anerkannt. Der NMI kann asynchron angemeldet werden. Damit wird intern ein Auffangflipflop gesetzt. Die zeitlichen Darstellungen entsprechen in den Relationen zum Takt ansonsten dem INT und sind auf das Auffangflipflop bezogen. Wenn ein NMI anerkannt wird, führt die CPU statt der gehaltenen Instruktion einen Rückstart zur Adresse

0066H aus. Dieser Rückstart entspricht einem implizit adressierten Unterprogrammaufruf auf spezifische Adressen der Seite 0 des Speichers. Man beachte, daß aber keine der sonst vorhandenen acht Rückstartadressen eingesetzt wird.

**Maskierbarer Interrupt, Mode IM0.** Diese Mode ist in der Ausführung ähnlich wie beim Prozessor U808. Die CPU U880 betrachtet das von der unterbrechenden Einheit auf dem Datenbus plazierte Wort als *Befehl* und führt diesen entsprechend aus. Das bedeutet, daß nach  $\overline{\text{INT}}$ -Annahme nicht der Speicher angesprochen wird, sondern ein spezieller Interruptcontroller. Es kann dabei jeder Befehl auf den Datenbus gebracht werden, jedoch sind natürlich die Einbytebefehle RST p besonders vorteilhaft.

Die CPU fügt in den Bestätigungszyklus automatisch zwei zusätzliche WAIT-Zustände ein, so daß die Befehlsausführung um zwei Taktzustände verlängert ist. Damit ermöglicht die CPU den Aufbau und das Einschwingen einer externen Prioritätskette, unabhängig von den speziellen peripheren Schaltkreisen des Systems U880. Die Ausführung eines  $\overline{\text{RESET}}$

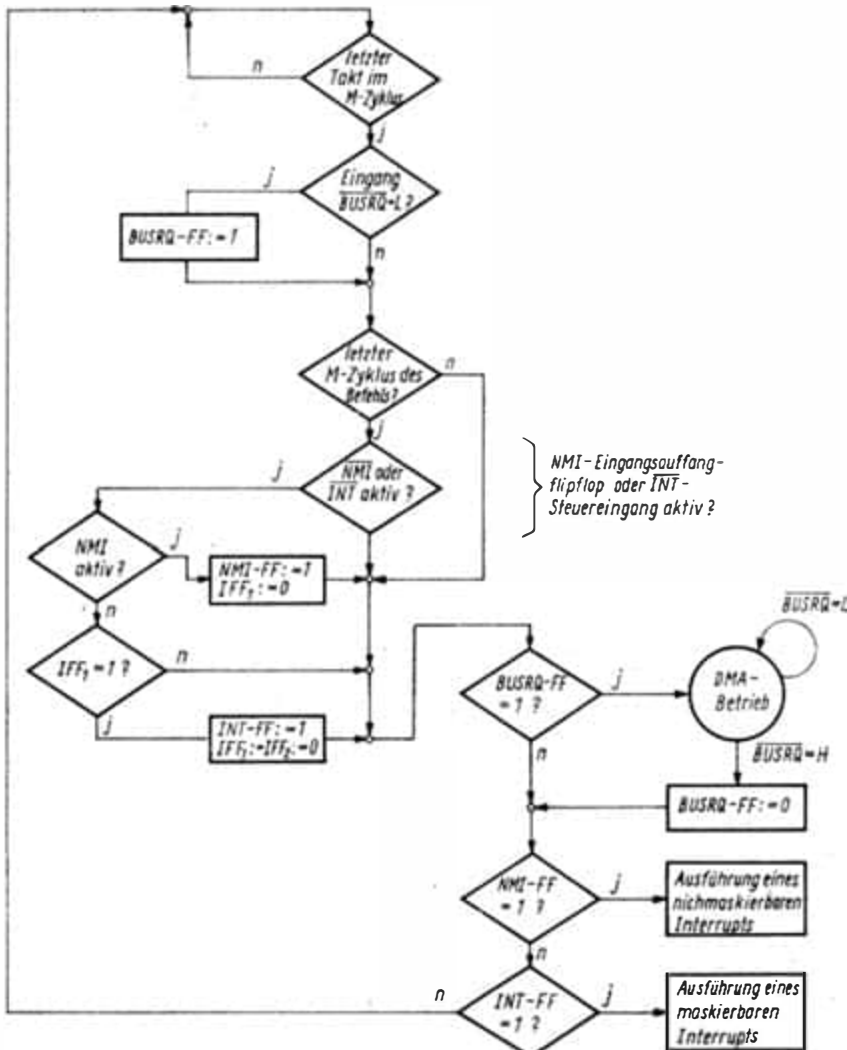


Bild 3.1.12. Flußplan der Unterbrechungsbearbeitung beim U880

durch die CPU veranlaßt diese, automatisch die Interruptmode 0 einzunehmen, da das die allgemeine Mode ist.

**Maskierbarer Interrupt, Mode IM1.** Diese Mode ist besonders zweckmäßig in Minimal-systemen anzuwenden, da ein weiterer hardwaremäßiger Rückstart genutzt werden kann. Der Rückstart erfolgt zur Adresse 0038H. Das bedeutet, daß die Ausführung identisch zum NMI ist – bis auf die Rückstartadresse 0038H statt 0066H und die Verlängerung in der Ausführung um zwei zugefügte WAIT-Zustände.

**Maskierbarer Interrupt, Mode IM2.** Im System U880 ist eine sehr leistungsfähige Mode in Verbindung mit den zugehörigen Peripherieschaltkreisen implementiert. Diese Mode ist die leistungsfähigste, da mit einem einzigen von der unterbrechenden Einheit zu liefernden Byte ein indirekter Unterprogrammaufruf auf einem beliebigen Speicherplatz spezifiziert werden kann.

Bei Anwendung dieser Mode baut der Programmierer im Speicher eine Tabelle auf, in der die Anfangsadressen der Interruptserviceroutinen stehen. Die Startadressen umfassen 16 bit, belegen also zwei Speicherplätze. Deshalb ist das von der unterbrechenden Einheit gelieferte Byte immer geradzahlig ( $d_0 = 0$ ), und es lassen sich nur 128 Startadressen mit

Tafel 3.1.15. Grenzwerte der IS U880

Kenngröße	Kurzzeichen	Einheit	Kleinstwert	Größt-wert	Bemerkung
Betriebsspannung	$U_{CC}$	V	-0,5	7	bei $\vartheta_a = 25^\circ\text{C}$
Eingangsspannung	$U_I$	V	-0,5	7	
Betriebstemperaturbereich	$\vartheta_a$	$^\circ\text{C}$	0 ... 70		
Lagerungstemperaturbereich	$\vartheta_{stg}$	$^\circ\text{C}$	-55 ... 125		
Verlustleistung	$P$	W	-	1,1	

Tafel 3.1.16. Statische Kennwerte der IS U880

Kenngröße	Kurzzeichen	Einheit	Kleinstwert	Größt-wert	Bemerkung
Betriebsspannung	$U_{CC}$	V	4,75	5,25	$\vartheta_a = 0 \dots 70^\circ\text{C}$
Eingangsspannung	$U_{IL}$	V	-0,5	0,8	
Taktingangsspannung	$U_{IH}$	V	2	$U_{CC}$	$\vartheta_a = 0 \dots 70^\circ\text{C}$
	$U_{ILC}$	V	-0,5	0,45	
Ausgangsspannung	$U_{IHC}$	V	$U_{CC} - 0,2$	$U_{CC}$	$\vartheta_a = 0 \dots 70^\circ\text{C}$
	$U_{OL}$	V	-	0,4	bei $I_{OL} = 1,8 \text{ mA}$
	$U_{OH}$	V	2,4	-	$\vartheta_a = 0 \dots 70^\circ\text{C}$ bei $I_{OH} = -0,25 \text{ mA}$ $\vartheta_a = 0 \dots 70^\circ\text{C}$
Eingangsreststrom	$I_{LI}$	$\mu\text{A}$	-	20	bei $U_I = 0$ und 5,25 V
Reststrom des Tristateausgangs im hochohmigen Zustand	$I_{LO}$	$\mu\text{A}$	-	10	
Reststrom des Datenbusses bei Eingabe	$I_{LD}$	$\mu\text{A}$	-	10	$\vartheta_a = 0$ und $70^\circ\text{C}$
Taktkapazität	$C_{CP}$	pF	-	60	bei $\vartheta_a = 25^\circ\text{C}$ und $f = 0,5 \dots 2 \text{ MHz}$
Eingangskapazität	$C_I$	pF	-	5	
Ausgangskapazität	$C_O$	pF	-	10	
Stromaufnahme	$I_{CC}$	mA	-	200	bei $U_{CC} = 5,25 \text{ V}$ und $\vartheta_a = 25^\circ\text{C}$

Tafel 3.1.17. Dynamische Kennwerte der IS U880

Kenngröße	Kurzzeichen	Einheit	Kleinwert	Größtwert
Taktperiode	$t_c$	ns	400	1)
H-Breite des Taktes	$t_{w(CH)}$	ns	180	2000
L-Breite des Taktes	$t_{w(CL)}$	ns	180	2000
Anstiegs-, Abfallzeiten des Taktes	$t_r, t_f$	ns	—	30
Bereitstellungszeit des $\overline{WAIT}$ vor H/L-Flanke des Taktes	$t_{s(WT)}$	ns	70	—
Bereitstellungszeit des $\overline{RESET}$ vor L/H-Flanke des Taktes	$t_{s(RS)}$	ns	90	—
Bereitstellungszeit des $\overline{INT}$ vor L/H-Flanke des Taktes	$t_{s(IT)}$	ns	80	—
Impulsbreite von NMI-Low	$t_{w(NMI)}$	ns	80	—
Bereitstellungszeit des $\overline{BUSRQ}$ vor L/H-Flanke des Taktes	$t_{s(BQ)}$	ns	80	—
Datenbereitstellungszeit bis zur L/H-Flanke des Taktes im M1-Zyklus (Signal: D <sub>0</sub> -D <sub>7</sub> )	$t_{sc(D)}$	ns	50	—
Datenbereitstellungszeit bis zur H/L-Flanke des Taktes von M2 bis M5 (Signal: D <sub>0</sub> -D <sub>7</sub> )	$t_{sc}$	ns	60	—
alle Nachwirkzeiten	$t_H$	ns	0	—
Verzögerungszeit $\overline{RFSH}$ von L/H-Flanke des Taktes bis $\overline{RFSH} = H$	$t_{DH(RF)}$	ns	—	160
Verzögerungszeit $\overline{HALT}$ von H/L-Flanke	$t_{D(HT)}$	ns	—	310
Verzögerungszeit $\overline{BUSAK}$ von L/H-Flanke des Taktes bis $\overline{BUSAK} = L$	$t_{DL(BA)}$	ns	—	130
Verzögerungszeit $\overline{BUSAK}$ von H/L-Flanke des Taktes bis $\overline{BUSAK} = H$	$t_{DH(BA)}$	ns	—	120
Verzögerungszeit $\overline{MREQ}, \overline{IORQ}, \overline{RD},$ $\overline{WR}$ bis Floaten	$t_{F(C)}$	ns	—	110
Verzögerungszeit $\overline{RD}$ von L/H-Flanke des Taktes bis $\overline{RD} = H$	$t_{DHC(RD)}$	ns	—	110
Verzögerungszeit $\overline{RD}$ von H/L-Flanke des Taktes bis $\overline{RD} = H$	$t_{DH\bar{C}(RD)}$	ns	—	120
Verzögerungszeit $\overline{WR}$ von L/H-Flanke des Taktes bis $\overline{WR} = L$	$t_{DLC(WR)}$	ns	—	90
Verzögerungszeit $\overline{WR}$ von H/L-Flanke des Taktes bis $\overline{WR} = L$	$t_{DL\bar{C}(WR)}$	ns	—	100
Verzögerungszeit $\overline{WR}$ von H/L-Flanke des Taktes bis $\overline{WR} = H$	$t_{DH\bar{C}(WR)}$	ns	—	110
Verzögerungszeit $\overline{M1}$ von L/H-Flanke des Taktes bis $\overline{M1} = L$	$t_{DL(M1)}$	ns	—	145
Verzögerungszeit $\overline{M1}$ von H/L-Flanke des Taktes bis $\overline{M1} = H$	$t_{DH(M1)}$	ns	—	145
Verzögerungszeit $\overline{RFSH}$ von L/H-Flanke des Taktes bis $\overline{RFSH} = L$	$t_{DL(RF)}$	ns	—	195

1)  $t_c = t_{w(CH)} + t_{w(CL)} + t_r + t_f$

einem Byte definieren. Dieses Byte bildet als Vektor den niederwertigen Teil der Startadresse. Die CPU setzt aus dem I-Register (höherwertiger Teil der Startadresse) und dem Vektor den 16-bit-Pointer zusammen. Bei der Anwendung der Mode 2 ist zu beachten, daß eine geeignete Initialisierung vorausgehen muß, da beim Auftreten eines  $\overline{\text{RESET}}$  an der CPU die Mode 0 für Interrupts gesetzt und das I-Register auf den Wert 00H gebracht wird. Das erste Byte, das durch den Pointer adressiert wird, ist der niederwertige Teil der Adresse in der Starttabelle für Interruptserviceroutinen.

Für die Ausführung eines Interrupts in dieser Mode werden 19 Taktzustände benötigt. Das teilt sich auf in sieben Taktzustände zum Lesen der 8 bit der unterbrechenden Einheit, sechs, um den Programmzähler zu retten (PUSH) und weitere sechs, um die Sprungadresse zu erhalten.

Die Einbeziehung der Interrupts in das Gesamtsystem, insbesondere die Wirkung der peripheren Elemente, ist ausführlich im Abschn. 4.2. dargestellt.

### 3.1.7. Zusammenfassung der technischen Daten

Nachfolgend sollen die elektrischen Kenngrößen der CPU U880 spezifiziert werden. Als Bezugspotential gilt für alle angegebenen Parameter  $U_{SS} = 0 \text{ V}$ .

Tafel 3.1.15 zeigt die beim Einsatz zu beachtenden Grenzwerte. Tafel 3.1.16 gibt die statischen Kennwerte der IS U880 wieder. In Tafel 3.1.17 sind die dynamischen Kennwerte angegeben. Tafel 3.1.18 enthält die Angabe der Verzögerungszeiten.

Tafel 3.1.18. Verzögerungszeiten der IS U880

Kenngröße	Kurzzeichen	Einheit	Kleinstwert	Größt-wert
Verzögerungszeit $\overline{\text{MREQ}}$ von H/L-Flanke des Taktes bis $\overline{\text{MREQ}} = \text{H}$	$t_{\text{DHC}(\text{MR})}$	ns	—	110
Verzögerungszeit $\overline{\text{MREQ}}$ von L/H-Flanke des Taktes bis $\overline{\text{MREQ}} = \text{H}$	$t_{\text{DHC}(\overline{\text{MR}})}$	ns	—	110
Verzögerungszeit $\overline{\text{IORQ}}$ von L/H-Flanke des Taktes bis $\overline{\text{IORQ}} = \text{L}$	$t_{\text{DLC}(\text{IR})}$	ns	—	100
Verzögerungszeit $\overline{\text{IORQ}}$ von H/L-Flanke des Taktes bis $\overline{\text{IORQ}} = \text{L}$	$t_{\text{DLC}(\overline{\text{IR}})}$	ns	—	120
Verzögerungszeit $\overline{\text{IORQ}}$ von L/H-Flanke des Taktes bis $\overline{\text{IORQ}} = \text{H}$	$t_{\text{DHC}(\text{IR})}$	ns	—	110
Verzögerungszeit $\overline{\text{IORQ}}$ von H/L-Flanke des Taktes bis $\overline{\text{IORQ}} = \text{H}$	$t_{\text{DHC}(\overline{\text{IR}})}$	ns	—	120
Verzögerungszeit $\overline{\text{RD}}$ von L/H-Flanke des Taktes bis $\overline{\text{RD}} = \text{L}$	$t_{\text{DLC}(\text{RD})}$	ns	—	110
Verzögerungszeit $\overline{\text{RD}}$ von H/L-Flanke des Taktes bis $\overline{\text{RD}} = \text{L}$	$t_{\text{DLC}(\overline{\text{RD}})}$	ns	—	140
Adressenausgangsverzögerungszeit	$t_{\text{D}(\text{AD})}$	ns	—	160
Verzögerungszeit bis Floaten	$t_{\text{F}(\text{AD})}$	ns	—	110
Ausgangsverzögerungszeiten: Daten	$t_{\text{D}(\text{D})}$	ns	—	260
Verzögerungszeit bis Floaten bei Schreibzyklus	$t_{\text{F}(\text{D})}$	ns	—	90
Verzögerungszeit $\overline{\text{MREQ}}$ von H/L-Flanke des Taktes bis $\overline{\text{MREQ}} = \text{L}$	$t_{\text{DL}\overline{\text{C}}(\text{MR})}$	ns	—	110

Tafel 3.1.19. Zusätzliche Zeitangaben bei der IS U880

- Adresse vor MREQ stabil, Speicherzyklus  
 $t_{acm} = t_{w(CH)} + t_r - 75 \text{ ns}$
- Adresse vor IORQ, RD oder WR stabil, E/A-Zyklus  
 $t_{aci} = t_c - 80 \text{ ns}$
- Adresse nach RD oder WR  
 $t_{ca} = t_{w(CL)} + t_r - 40 \text{ ns}$
- Adresse nach RD oder WR beim Floaten stabil  
 $t_{caf} = t_{w(CL)} + t_r - 60 \text{ ns}$
- Daten vor WR stabil, Speicherzyklus  
 $t_{dcm} = t_c - 180 \text{ ns}$
- Daten vor WR stabil, E/A-Zyklus  
 $t_{dci} = t_{w(CL)} + t_r - 180 \text{ ns}$
- Daten nach WR stabil  
 $t_{daf} = t_{w(CL)} + t_r - 50 \text{ ns}$
- Impulsbreite von MREQ-Low  
 $t_{w(MREL)} = t_c - 40 \text{ ns}$
- Impulsbreite von MREQ-High  
 $t_{w(MRH)} = t_{w(CH)} + t_r - 30 \text{ ns}$
- Impulsbreite von WR-Low  
 $t_{w(WRL)} = t_c - 40 \text{ ns}$
- M1 vor IORQ stabil (Interruptannahme)  
 $t_{m1} = 2t_c + t_{w(CH)} + t_r - 80 \text{ ns}$

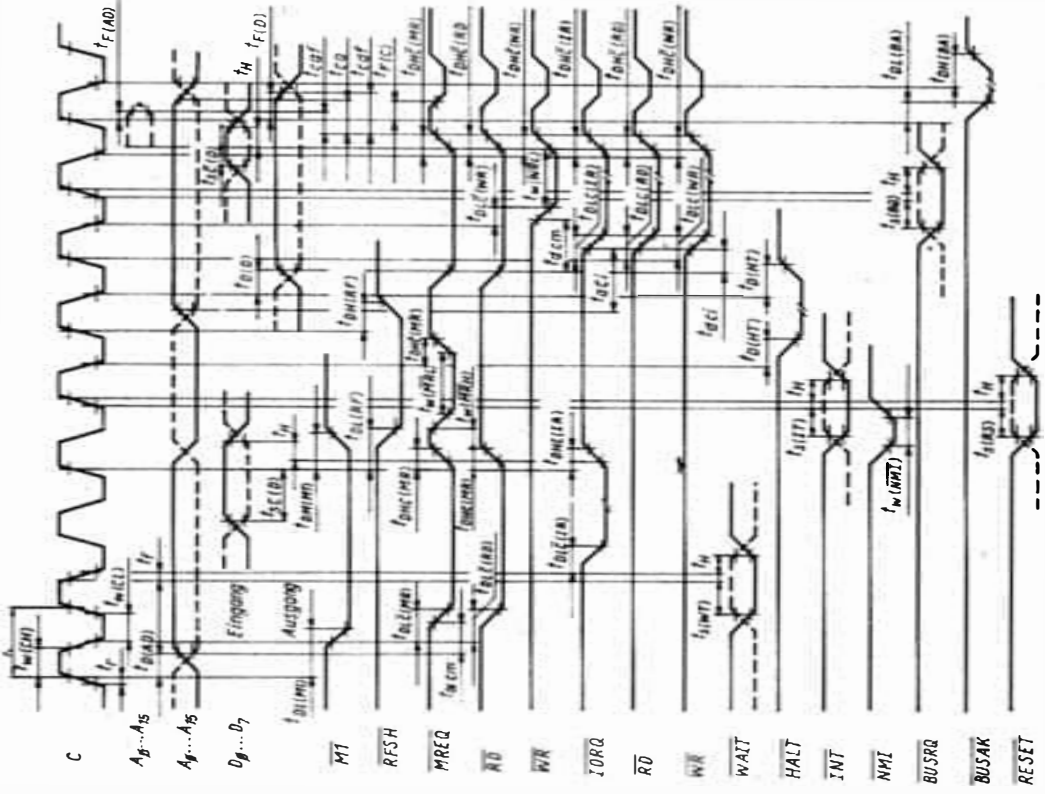


Bild 3.1.13. Darstellung der dynamischen Kennwerte der CPU im Zeitdiagramm

Hierfür gelten folgende Einsatzbedingungen:

$$\begin{aligned} \vartheta_a &= 70^\circ\text{C} \\ U_{CC} &= 4,75\text{ V} \\ U_{IL} &= 0,8\text{ V} \\ U_{IH} &= 2,0\text{ V} \\ U_{ILC} &= 0,45\text{ V} \\ U_{IHC} &= 4,55\text{ V} \\ C_L &= 100\text{ pF.} \end{aligned}$$

In Tafel 3.1.19 sind zusätzliche Zeitangaben aufgeführt. Die zugehörige bildliche Darstellung des Zeitverhaltens ist im Bild 3.1.13 enthalten.

## 3.2. Parallele Ein-/Ausgabe-Einheit

### 3.2.1. Einführung

Die IS U855 ist eine parallele Ein-/Ausgabe-Einheit (PIO), die ihre maximale Leistungsfähigkeit im System U880 erreicht. Sie beinhaltet zwei vollständige, nach außen TTL-kompatible 8-bit-Ein-/Ausgabe-Kanäle und ist in ihren wesentlichen Eigenschaften programmierbar.

Die charakteristischen Merkmale der IS sind:

- n-Kanal-Silicon-Gate-Technologie (nSGT), Depletion Load
- eine Versorgungsspannung 5 V
- Einphasensystemtakt (5 V)
- zwei unabhängige 8-bit-Ein-/Ausgabe-Kanäle mit Handshakelogik (ermöglicht Quittierungsbetrieb)
- vier Betriebsarten können programmiert werden:
  - Byteausgabe
  - Byteeingabe
  - bidirektionaler Bytebetrieb (nur bei Port A)
  - bitorientierter Datenaustausch
- Interruptfähigkeit der PIO ermöglicht reaktionsschnelle und leistungsfähige Systemkonzepte
- Interruptverschachtelungen durch „daisy chain“-Logik möglich (Kaskadierung mehrerer PIO bzw. systemspezifischer peripherer IS)
- automatische Interruptvektorerzeugung
- Kanal B ermöglicht direkte Treibung von Darlingtontransistorstufen (8 Ausgänge)
- TTL-Kompatibilität aller Ein- und Ausgänge
- 40poliges DIL-Gehäuse nach TGL 26713.

Typische Anwendungsfälle für die IS U855 sind parallele Schnittstellen zu Tastaturen, Kontaktabfrageeinheiten, Lochbandlesern und -stanzern, Druckern und PROM-Programmierereinheiten.

Die PIO-Einheit ist hierbei wesentlich leistungsfähiger als beispielsweise ein Interface mit der IS MH3212 (s. auch Abschn. 5.2.), da sie u. a. eine vollständige programmierbare automatische Interruptbearbeitung ermöglicht. Dadurch ist es nicht notwendig, die Mikrorechnerperipherie mit Hilfe zyklischer Abfrage zu überwachen und zu bedienen

(sog. Polling), sondern es ist eine wesentlich zeitgünstigere Überwachung durch Interruptbearbeitung möglich.

Die Zusammenschaltung der IS U855 mit der CPU U880 erfordert keine weiteren zusätzlichen Standardbauelemente, ausgenommen bei großen Systemen Adressendekoder, Kaskadierungslogik und Datenbustreiber.

### 3.2.2. Struktureller Aufbau

Das Blockschaltbild des parallelen Ein-/Ausgabe-Bausteins ist im Bild 3.2.1 dargestellt. Die IS U855 besteht aus einem CPU-Buspuffer, einer internen Steuerlogik, der Logik der beiden Ein-/Ausgabe-Ports und einer Interruptsteuerlogik.

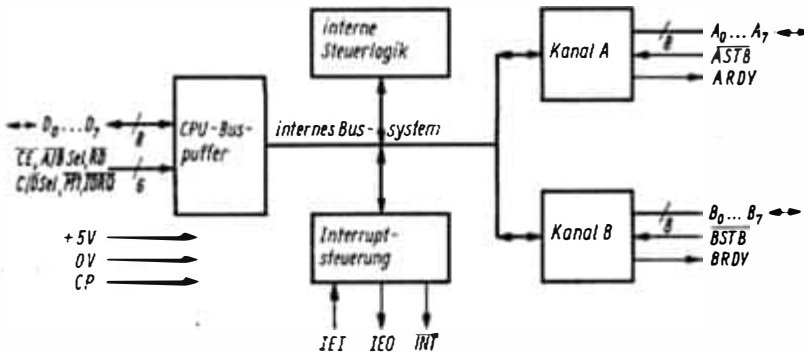


Bild 3.2.1. Blockschaltbild der PIO

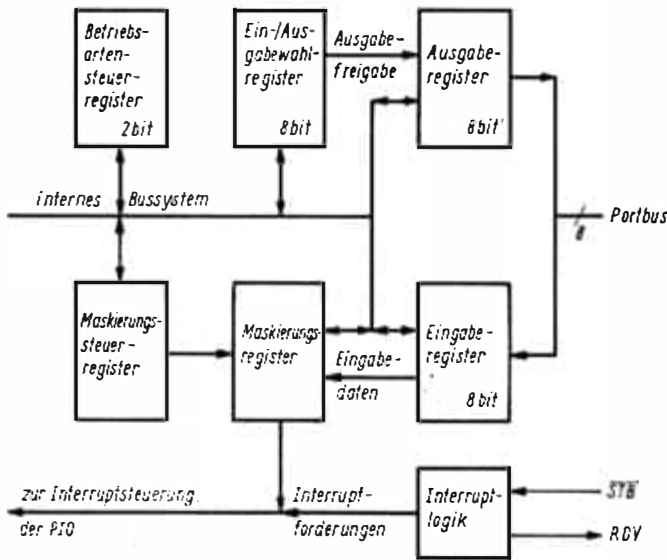


Bild 3.2.2  
Blockschaltbild der Portlogik

Der CPU-Buspuffer dient zur Anpassung der PIO-Daten- und -Steuerinformationen an das Standardzeitverhalten des U880 und ermöglicht eine direkte Zusammenschaltung beider Bauelemente. Die interne Steuerlogik überwacht in Abhängigkeit vom Systemtakt, von den CPU-Steuersignalen und vom Zustand der PIO-Quittierungssignale die Funktion

der Einheit. Die Interruptsteuereinheit übernimmt in Auswertung der in den Portlogiken abgespeicherten Informationen, der an den Kanälen anliegenden Daten, des Zustands der Quittierungssignale und in Abhängigkeit externer Informationen die Anforderung von Interrupts an die CPU.

Bild 3.2.2 zeigt die innere Struktur eines der beiden Ein-/Ausgabe-Ports.

Das Betriebsartenregister ist ein 2-bit-Register, das durch ein Steuerwort (OUT-Befehl) vom Prozessor programmiert wird. Mit ihm wird eine der vier möglichen Betriebsarten des PIO-Ports (Byteausgabe, Byteeingabe, bidirektionaler Bytebetrieb bei Port A, Bitbetrieb) festgelegt.

Das Ausgaberegister dient in der Betriebsart Byteausgabe zur Zwischenspeicherung der Ausgabedaten. Es wird durch die CPU mit Hilfe eines Datenworts (OUT-Befehl) programmiert.

Das Eingaberegister dient analog hierzu in der Betriebsart Byteeingabe zur Zwischenspeicherung der Porteingabedaten. Die Übernahme dieser Daten wird durch die Quittierungslogik (Signal **STB**) gesteuert. Das Register kann vom Prozessor mittels einer IN-Instruktion als Datenwort gelesen werden. In der Betriebsart bidirektionaler Bytebetrieb dienen die Ein- und Ausgaberegister des Ports A zur bidirektionalen Datenübertragung. Die Steuerung hierfür wird von den Quittierungslogiken der Kanäle A (Datenausgabesteuerung) und B (Dateneingabesteuerung) übernommen. Port B muß in diesem Anwendungsfall in der Betriebsart Bitbetrieb verwendet werden, da hierfür keine Quittierungslogik erforderlich ist.

Das Ein-/Ausgabe-Wahlregister ist ein 8-bit-Register, das in der Betriebsart Bitdatenaustausch zur Festlegung der Funktion der einzelnen Bits (Eingabe oder Ausgabe) dient. Es ist durch den Prozessor per Steuerwort programmierbar (OUT-Operation).

Die Portlogik beider Kanäle enthält neben diesen für den eigentlichen Datentransfer nötigen Registern weitere, die eine leistungsfähige gerätebezogene Interruptbearbeitung ermöglichen.

Das Interruptfreigabeflipflop dient zur Sperrung bzw. Freigabe der Interruptfähigkeit des entsprechenden Ports und ist über ein Steuerwort von der CPU setz- oder rücksetzbar.

Das Vektorregister ist ein 8-bit-Register; es ist mit Hilfe eines Steuerworts programmierbar. In der Interruptbetriebsart IM2 des Prozessors U880 wird nach einer Interruptanmeldung und -quittierung eines systemzugehörigen Peripherieschaltkreises automatisch eine 16-bit-Unterprogrammstartadresse gebildet. Diese Startadresse wird aus zwei aufeinanderfolgenden Speicherplätzen entnommen (Low-Byte, High-Byte), die durch den Inhalt des 8-bit-CPU-Registers I und den sieben niederwertigeren Bits des Vektorregisters des unterbrechenden peripheren Geräts adressiert werden (s. Abschn. 3.2.4.2.).

Es ist somit möglich, in Verbindung mit einer Prioritätsfestlegung mittels der „daisy-chain“-Kaskadierung ein sehr leistungsfähiges und automatisch wirkendes Interruptsystem anzuwenden.

In den byteorientierten Betriebsarten des PIO erfolgt eine Interruptanmeldung nach Auswertung der Signale in der Quittierungslogik.

Dagegen erfolgt die Interruptanmeldung in der Bitbetriebsart nach Auswertung der anliegenden Portdaten. In dieser Auswertung dienen die Maskierungs- und Maskierungssteuerregister. Beide Register sind ebenfalls durch Steuerwörter vom Prozessor programmierbar. Mit dem 8-bit-Maskierungsregister wird festgelegt, welche der acht Datenlinien des entsprechenden Ports überhaupt für eine Interruptanmeldung verwendet werden. Das 2-bit-Maskierungssteuerregister entscheidet, welcher Pegel als aktiv definiert wird (H oder L) und ob eine AND- oder eine OR-Verknüpfung zwischen den ausgewählten Datenlinien die Anmeldung auslöst. In dieser Betriebsart können also weitere logische Entscheidungen

(z. B. Auslösung eines Alarms bei einer bestimmten Bitbelegung des Ports) ohne zusätzlichen Hardwareaufwand bzw. zeitliche Belastung des Mikroprozessors gefällt werden.

### 3.2.3. Erläuterung der Anschlußbelegung

Im Bild 3.2.3 ist die schematische Anschlußbelegung des U855 dargestellt. Die Funktion dieser Pins soll im folgenden näher beschrieben werden.

**D<sub>0</sub> ... D<sub>7</sub>** Data Bus (bidirektional, tristate)

Über den Datenbus des Systems U880 erfolgt der eigentliche Informationsaustausch zwischen der CPU und der Ein-/Ausgabe-Einheit; es werden alle Daten- und Steuerwörter übertragen.

**B/ $\bar{A}$  Sel** Port B/ $\bar{A}$  Select (Eingang, H-aktiv)

Mit diesem Pin erfolgt die Auswahl des Ports (A oder B), mit dem der Datenaustausch erfolgen soll. L-Pegel selektiert Port A, H-Pegel Port B. Üblicherweise wird die CPU-Adresse A<sub>0</sub> für diese Funktion verwendet.

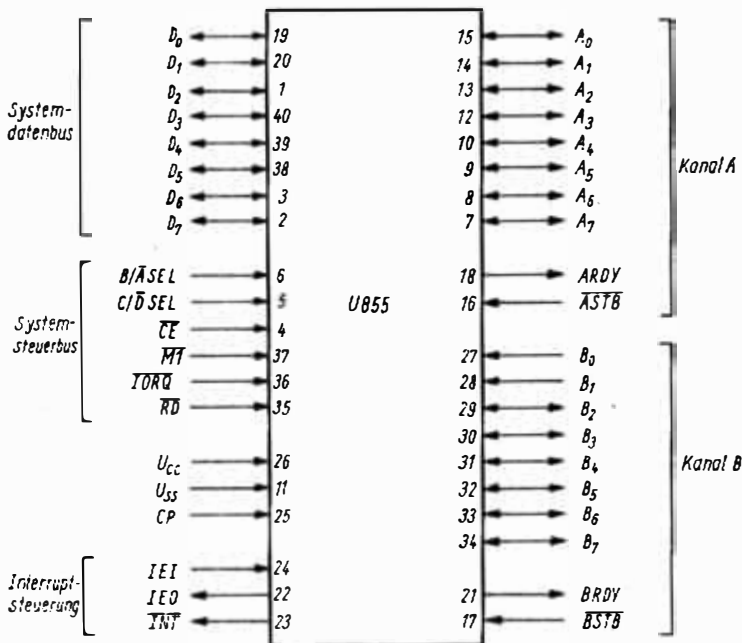


Bild 3.2.3. Schematische Anschlußbelegung der PIO

**C/ $\bar{D}$  Sel** Control/Data Select (Eingang, H-aktiv)

Dieses Signal legt fest, ob im laufenden I/O-Zyklus der CPU das Wort, das in das durch Pin B/ $\bar{A}$  Sel ausgewählte Port eingeschrieben bzw. ausgelesen wird, ein Daten- oder ein Steuerwort darstellt. Bei L-Pegel erfolgt ein Datenaustausch mit dem Eingabe- bzw. Ausgaberegister des Ports; im anderen Fall werden die Steuerregister des Kanals programmiert.

Für diese Funktion wird i. allg. die CPU-Adresse A<sub>1</sub> verwendet.

**$\overline{CE}$**  Chip Enable (Eingang, L-aktiv)

Mit diesem Signal (L-Pegel) erfolgt die Aktivierung der PIO und somit die Ermöglichung des Datenaustauschs mit der CPU. Das  $\overline{CE}$ -Signal wird üblicherweise in einem Adreßdeko-der aus den CPU-Adressen  $A_2 \dots A_7$  gewonnen.

**CP** Clock Pulse (Eingang, 5-V-Pegel)

Der Systemtakt dient zur internen Synchronisation der meisten zeitlichen Abläufe der IS U855.

 **$\overline{M1}$**  Machine Cycle 1 (Eingang, L-aktiv)

Das Signal  $\overline{M1}$  (Maschinenzyklus 1 der CPU) ist ein Steuersignal der CPU, das zur Kennzeichnung des Befehlsholezyklus (op-code fetch) dient. In Verbindung mit dem PIO-Bau-stein hat es die Aufgabe, die Interruptlogik zu synchronisieren (insbesondere bei der Interruptquittierung und bei der Erkennung des RETI-Befehls am Ende eines Unter-programms). Das Pin hat außerdem die folgende zusätzliche Funktion: Ein L-Pegel am  $\overline{M1}$ -Pin in Verbindung mit inaktiven  $\overline{RD}$ - und  $\overline{IORQ}$ -Signalen bewirkt eine Rücksetzung der PIO-Logik. Somit wird ein sonst notwendiges zusätzliches Pin für die Rücksetzung der IS eingespart. Das ist notwendig aufgrund der Beschränkung durch das 40polige Ge-häuse.

 **$\overline{IORQ}$**  In/Out Request (Eingang, L-aktiv)

Dieses Signal ist ebenfalls ein Steuersignal der CPU. Es dient in Verbindung mit den PIO-Signalen  $\overline{CE}$ ,  $C/\overline{D}$  Sel,  $B/\overline{A}$  Sel und  $\overline{RD}$  zur Kennzeichnung des Datenverkehrs zwi-schen CPU und PIO.

Die nachfolgend dargestellte Besonderheit ist bei diesem Signal im Zusammenwirken mit der Ein-/Ausgabe-Einheit U855 zu beachten. Ein L-Pegel an diesem Pin ( $\overline{IORQ}$ ) in Verbindung mit einem aktiven  $\overline{M1}$ -Signal bedeutet, daß die CPU die Anmeldung eines Interrupts durch eines der beiden Ports quittiert.

Im selben Zyklus erfolgt daraufhin die Plazierung des entsprechenden Interruptvektors auf den Datenbus.

 **$\overline{RD}$**  Read Cycle (Eingang, L-aktiv)

Das  $\overline{RD}$ -Signal ist ebenfalls ein Steuersignal der CPU. Es dient zum Einschreiben von Informationen (Daten) in die CPU. In Verbindung mit den aktiven Zuständen der Si-gnale  $\overline{CE}$ ,  $C/\overline{D}$  Sel,  $B/\overline{A}$  Sel und  $\overline{IORQ}$  steuert es den Datentransport von der PIO in Rich-tung CPU.

Das für den Schreibvorgang benötigte Signal  $\overline{WR}$  (write cycle) muß aufgrund der Be-schränkung der Pinanzahl des Gehäuses ebenfalls intern generiert werden:

$$\overline{WR}' = \overline{RD} + \overline{CE} + \overline{IORQ}.$$

**IEI** Interrupt Enable Input (Eingang, H-aktiv)

Dieses Signal dient in Verbindung mit IEO zur Bildung einer systemweiten Interrupt-prioritätsfestlegung durch Kaskadierung aller interruptfähigen I/O-Geräte (PIO, SIO, CTC). H-Pegel an diesem Pin bedeutet, daß kein höherwertigeres Gerät zum aktuellen Zeitpunkt eine Interruptserviceroutine (ISR) durchführt bzw. angemeldet hat; die PIO ist also *interruptfähig*. L-Pegel verbietet die Anmeldung einer ISR bzw. unterbricht eine ge-

rade in Bearbeitung befindliche ISR zugunsten des höherpriorisierten Geräts, sofern der maskierbare Interrupt der CPU freigegeben ist.

#### **IEO** Interrupt Enable Output (Ausgang, H-aktiv)

Dieses Pin führt L-Pegel, wenn die betreffende PIO oder ein höherwertigeres Gerät in der „daisy-chain“ eine ISR in Bearbeitung bzw. angemeldet hat. H-Pegel erlaubt nachfolgenden Geräten die Anmeldung von Interrupts über die  $\overline{\text{INT}}$ -Linie an die CPU.

#### **$\overline{\text{INT}}$** Interrupt Request (Ausgang, open-drain, L-aktiv)

Das Signal  $\overline{\text{INT}}$  dient zur Anmeldung eines Interrupts an die CPU. Die Quittierung dieser Anmeldung erfolgt von der CPU durch die Signale  $\overline{\text{IORQ}}$  und  $\overline{\text{M1}}$ ; die Interruptanmeldung wird daraufhin rückgesetzt.

#### **A<sub>0</sub> ... A<sub>7</sub>** Port A Bus (bidirektional, tristate)

Der 8-bit-Portbus des Kanals A dient zum Datenaustausch mit der Peripherie.

#### **$\overline{\text{A STB}}$** Port A Strobe Pulse (Eingang, L-aktiv)

Die Funktion dieses Eingangs hängt von der gewählten Betriebsart ab.

*Byteausgabe:* Das periphere Gerät quittiert den Empfang von Daten vom PIO-Port mit einer positiven Flanke; diese Flanke kann im Port zur Erzeugung eines Interrupts verwendet werden.

*Byteeingabe:* Wenn das Signal  $\overline{\text{A STB}}$  aktiv ist, werden Daten vom Portbus (also von der Peripherie) in das Eingaberegister gespeichert; Interrupterzeugung ist möglich.

*Bidirektionaler Bytebetrieb:* Während  $\overline{\text{A STB}}$  aktiv ist, werden die Daten aus dem Ausgaberegister des Ports A auf den bidirektionalen Port-A-Bus durchgeschaltet; die positive Flanke des Signals quittiert den Empfang durch das periphere Gerät und kann eine Interruptanmeldung auslösen.

*Bitorientierter Betrieb:* Der Eingang wird nicht abgefragt.

#### **A RDY** Port A Ready (Ausgang, H-aktiv)

Die Bedeutung dieses Ausgangs hängt ebenfalls von der gewählten Betriebsart ab.

*Byteausgabe:* Das Signal wird aktiv, um anzuzeigen, daß Daten zur Ausgabe verfügbar sind und der Portbus einen eingeschwungenen Zustand erreicht hat.

*Byteeingabe:* Das Signal ist aktiv, wenn das Eingaberegister „leer“ ist (also vor der positiven Flanke von  $\overline{\text{A STB}}$ ) und Daten von der Peripherie erwartet.

*Bidirektionaler Betrieb:* Der aktive Zustand des Signals zeigt an, daß Daten vom Ausgaberegister des Ports A auf dem Port-A-Bus verfügbar sind. Wenn  $\overline{\text{A STB}}$  inaktiv ist, wird in dieser Betriebsart der Portbus auf Eingabe umgeschaltet (ist also „hochohmig“).

*Bitbetrieb:* Der Ausgang wird nicht verwendet und ist inaktiv (führt also L-Pegel).

#### **B<sub>0</sub> ... B<sub>7</sub>** Port B Bus (bidirektional, tristate)

Der 8-bit-Portbus des Kanals B dient ebenso wie der Port-A-Bus zum Datenaustausch mit der Peripherie. Zusätzlich können durch dieses Port direkt Darlingtontistorstufen angesteuert werden.

#### **$\overline{\text{B STB}}$** Port B Strobe Puls (Eingang, L-aktiv)

Die Funktion dieses Pins entspricht in den Betriebsarten Byteeingabe, Byteausgabe und Bitbetrieb der des Signals  $\overline{\text{A STB}}$  des Ports A.

Wird der PIO in die Betriebsart bidirektionaler Bytebetrieb für Kanal A und Bitbetrieb für Kanal B gesetzt, so werden die beiden Pins  $\overline{B\ STB}$  und B RDY für die Steuerung des bidirektionalen Datenaustauschs des Ports A verwendet.

Wenn der Eingang  $\overline{B\ STB}$  in diesem Fall aktiv wird, werden Daten vom Port-A-Bus (also von der Peripherie) in das Eingaberegister des Ports A abgespeichert; gleichzeitig ist eine Interruptanmeldung mit der L/H-Flanke des Signals möglich.

**B RDY** Port B Ready (Ausgang, H-aktiv)

Die Funktion dieses Pins entspricht ebenfalls in den Betriebsarten Byteeingabe, Byteausgabe und Bitbetrieb der des Ausgangs A RDY des Ports A.

Im bidirektionalen Bytebetrieb des Ports A (Port B im bitorientierten Betrieb) ist der Ausgang B RDY aktiv, wenn das Eingaberegister des Kanals A neue Daten über den Port-A-Bus von der Peripherie erwartet.

**3.2.4. Auswahl der Betriebsarten**

**3.2.4.1. Rücksetzen der PIO**

Die parallele Ein-/Ausgabe-Einheit U855 nimmt nach Einschalten der Betriebsspannung automatisch einen rückgesetzten Zustand ein. Hierbei werden beide Kanäle in die Betriebsart Byteeingabe (Mode 1) gesetzt, um einen hochohmigen Zustand an den Portdatenleitungen ( $A_0 \dots A_7, B_0 \dots B_7$ ) zu erreichen. Die Readysignale (A RDY, B RDY) sind inaktiv. Außerdem werden bei beiden Kanälen jeweils die Maskierungsregister, die Ausgaberegister und die Interruptfreigabeflipflops rückgesetzt.

Aufgrund der Anschlußbeschränkung des 40poligen Gehäuses besitzt die IS U855 keinen gesonderten Rücksetzanschluß. Die IS läßt sich aber durch die Signalkonfiguration

$$\overline{MI} = L, \quad \overline{RD} = H, \quad \overline{IORQ} = H$$

rücksetzen. Diese Signalbelegung tritt im System U880 nicht auf und kann deshalb für diese Funktion verwendet werden.

Im Bild 3.2.4 ist die Verknüpfung des Rücksetzsignals mit  $\overline{MI}$  dargestellt.

Beim Rücksetzen der IS U855 mit dem Signal  $\overline{MI}$  werden ebenfalls die oben beschriebenen Funktionen ausgeführt; die Inhalte der Vektorregister beider Kanäle bleiben erhalten.

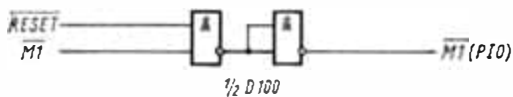


Bild 3.2.4 Rücksetzlogik für die PIO

Neben diesen beiden Möglichkeiten des Rücksetzens kann die PIO auch softwaremäßig in den durch die genannten Bedingungen charakterisierten rückgesetzten Zustand gesetzt werden. Hierbei müssen die entsprechenden PIO-Register einzeln geladen werden. Besondere Bedeutung kommt der Rücksetzung des Interruptbearbeitungszustands zu (s. Abschn. 3.2.4.4.).

**3.2.4.2. Interruptvektor**

In der leistungsfähigen Interruptbetriebsart IM2 der CPU U880 wird von dem unterbrechenden peripheren Gerät (in diesem Fall: die PIO) ein Interruptvektor abgefordert.

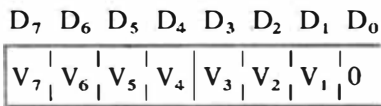
Dieser Vektor wird von der PIO während des von der CPU eingeschobenen Interruptquittierungszyklus auf den Datenbus geschaltet. Er bildet mit dem CPU-internen Register I einen Pointer (das 8-bit-I-Register stellt hierbei den höherwertigen Anteil dar), der auf eine Speicherplatzadresse zeigt. Aus den Inhalten dieses und des nachfolgenden Speicherplatzes wird die Startadresse des durch die Interruptanmeldung aufzurufenden Unterprogramms gebildet.

Da zur Bildung der Startadresse jeweils zwei Speicherplätze benötigt werden, muß der Interruptvektor (V) des peripheren Geräts die Wortlänge von 7 bit bereitstellen.

Das Bit  $V_0$  des Vektors wird bei Aussendung automatisch zu 0 gesetzt. Der hieraus gebildete „gerade“ 16-bit-Pointer (Bit 0 = 0) zeigt auf das niederwertige Byte der Startadresse, der inkrementierte Pointer (Bit 0 = 1) auf das höherwertige Byte.

Der Interruptvektor kann durch einen OUT-Befehl der CPU in den ausgewählten Kanal der PIO als Steuerwort ( $C/\bar{D}$  Sel = 1) eingeschrieben werden.

Dieses Steuerwort hat folgendes Format:

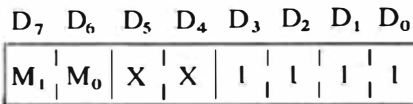


Das Bit  $D_0 = 0$  identifiziert das Steuerwort als Interruptvektor für den PIO-Kanal.

### 3.2.4.3. Betriebsartenwahl

Beide Kanäle der IS U855 können in einer der Betriebsarten Byteausgabe, Byteeingabe, bidirektionaler Bytebetrieb und Bitbetrieb (ausgenommen Port B im bidirektionalen Betrieb) arbeiten. Die Auswahl hierzu wird in der Kanallogik durch das 2-bit-Betriebsartenregister getroffen. Dieses Register des ausgewählten Ports kann mit Hilfe eines OUT-Befehls der CPU mit einem Steuerwort ( $C/\bar{D}$  Sel = 1) geladen werden.

Das hierfür notwendige Steuerwort hat folgendes Format:



Die Belegung der Bits  $D_0$ ,  $D_1$ ,  $D_2$  und  $D_3$  mit 1 identifiziert das Steuerwort als Betriebsartenauswahlwort. Der Wert der Bits  $D_4$  und  $D_5$  wird nicht abgefragt. Die Bedeutung der Bits  $D_6$  und  $D_7$  für die ausgewählte Betriebsart ist in Tafel 3.2.1 dargestellt.

$D_7$	$D_6$	Betriebsart	Mode	Mögliches Steuerwort (hexadezimal)
0	0	Byteausgabe	0	0FH
0	1	Byteeingabe	1	4FH
1	0	Bidirektionaler Betrieb	2	8FH
1	1	Bitbetrieb	3	0CFH

Tafel 3.2.1. Betriebsartenauswahl an der PIO

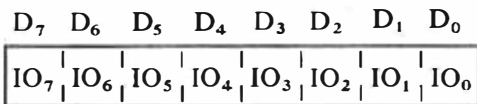
Bei Auswahl der Betriebsart Byteausgabe (Mode 0) können Daten durch einen OUT-Befehl der CPU in das ausgewählte Ausgaberegister der PIO eingeschrieben werden. Der Inhalt des Ausgaberegisters wird direkt auf den zugehörigen Portbus ( $A_0 \dots A_7$

bzw.  $B_0 \dots B_7$ ) durchgeschaltet. Er kann jederzeit unabhängig vom Zustand der Quittierungssignale ( $\overline{A\ STB}$ ,  $A\ RDY$  bzw.  $\overline{B\ STB}$ ,  $B\ RDY$ ) durch eine entsprechende OUT-Operation verändert werden. Die Quittierungssignale dienen zur Ansteuerung der Peripherie ( $A\ RDY$ ,  $B\ RDY$ ) und zur Anmeldung von Interrupts durch die Peripherie ( $\overline{A\ STB}$ ,  $\overline{B\ STB}$ ). Der aktuelle Inhalt des aktivierten Ausgaberegisters kann jederzeit durch eine Leseoperation (IN-Befehl der CPU) abgefragt werden.

Die Auswahl der Betriebsart Byteeingabe (Mode 1) bewirkt ein automatisches Rücksetzen der Quittierungssignale  $A\ RDY$  und  $B\ RDY$ . Das Laden der Eingaberegister der beiden PIO-Kanäle mit den auf den zugehörigen Portbussen anliegenden Daten erfolgt durch die Quittierungssignale  $\overline{A\ STB}$  bzw.  $\overline{B\ STB}$  zustandsgesteuert. Bei Ansteuerung dieser Signale durch die Peripherie (also bei Quittungsbetrieb) muß nach Auswahl der Betriebsart eine Leseoperation (IN-Befehl der CPU) ausgeführt werden, um die Signale  $A\ RDY$  bzw.  $B\ RDY$  zu aktivieren.

In der Betriebsart bidirektionaler Bytebetrieb (Mode 2) des Ports A werden alle vier Quittierungsleitungen zur Richtungssteuerung des bidirektionalen Port-A-Busses benötigt. Deshalb muß vor Auswahl der Mode 2 des Kanals A der Kanal B in die Betriebsart Bitbetrieb gesetzt werden. In dieser Betriebsart (Mode 3) werden die Quittierungsleitungen ( $B\ RDY$ ,  $\overline{B\ STB}$ ) nicht benutzt und an den Kanal A freigegeben. Bei Datenleseoperationen (IN-Befehle der CPU) im bidirektionalen Betrieb werden in Abhängigkeit vom Zustand des Quittierungssignals  $\overline{A\ STB}$  Daten entweder aus dem Ausgaberegister ( $\overline{A\ STB} = L$ ) oder aus dem Eingaberegister ( $\overline{A\ STB} = H$ ) des Kanals A gelesen.

Bei Selektierung der Betriebsart Bitbetrieb (Mode 3) können alle Bits des dem jeweiligen Kanal zugehörigen Portbusses unabhängig voneinander als Eingänge oder als Ausgänge definiert werden. Zu dieser Auswahl muß nach Festlegung der Betriebsart ein Steuerwort ( $C/\overline{D}\ Sel = 1$ ) in den entsprechenden Kanal eingeschrieben werden (OUT-Befehl der CPU). Mit diesem Steuerwort wird das Ein-/Ausgabe-Wahlregister der Portlogik geladen. Es hat das nachfolgend dargestellte Format:



Hierbei bedeutet das Setzen von Bits ( $IO_n = 1$ ), daß die der Wertigkeit dieser Bits entsprechenden Busleitungen ( $A_n$  bzw.  $B_n$ ) des ausgewählten Kanals als Eingänge wirken. Umgekehrt definiert das Rücksetzen ( $IO_m = 0$ ) die zugehörigen Busleitungen ( $A_m$  bzw.  $B_m$ ) als Ausgänge. Das Steuerwort ist nicht besonders als Ein-/Ausgabe-Wahlwort gekennzeichnet (alle 8 bit werden zur Informationsübertragung verwendet), da automatisch nach Auswahl der Mode 3 das nächste in die Kanallogik eingeschriebene Steuerwort zur Ein-/Ausgabe-Wahl verwendet wird.

Bei Datenleseoperationen (IN-Befehl) werden bei dieser Betriebsart die zur CPU übertragenen Daten bitweise entsprechend dem Inhalt des Port-Ein-/Ausgabe-Wahlregisters aus dem Eingabe- oder dem Ausgaberegister des Kanals zusammengestellt. Sind die entsprechenden Portleitungen als Eingänge programmiert, werden die zugehörigen Bits dem Eingaberegister, anderenfalls dem Ausgaberegister des Kanals entnommen. In der Bitbetriebsart werden die Quittierungssignale nicht verwendet; die Ausgänge  $A\ RDY$  bzw.  $B\ RDY$  sind inaktiv (führen also L-Pegel). Eine Ausnahme hiervon bildet der bereits erwähnte Fall, daß Kanal A in der Betriebsart bidirektionaler Betrieb arbeitet.

### 3.2.4.4. Interruptsteuerwort

Bei der IS U855 erfolgt die Auslösung von Interruptanmeldungen in Abhängigkeit von der für den Kanal ausgewählten Betriebsart. Bei den bytebezogenen Betriebsarten (Mode 0, 1 und 2) erfolgt die Anmeldung mit Hilfe der Quittierungslogik. In der Betriebsart Bitbetrieb (Mode 3) erfolgt die Interruptanmeldung durch eine programmierbare logische Auswertung der am Portbus anliegenden Datenkonfiguration.

In allen Betriebsarten hängt aber die Weitergabe der Interruptanmeldung an die CPU vom Zustand des zum PIO-Kanal zugehörigen Interruptfreigabeflipflops ab.

Das nachfolgend dargestellte Interruptsteuerwort dient zum Laden des Interruptfreigabeflipflops der Portlogik und zum Festlegen der in der Mode 3 zur Interruptauslösung benötigten Auswahlkriterien.

Das Steuerwort hat das nachfolgend dargestellte Format:

$D_7$	$D_6$	$D_5$	$D_4$	$D_3$	$D_2$	$D_1$	$D_0$
EI	A/O	H/L	MF	0	1	1	1

Die Belegung der Bits  $D_0$  bis  $D_3$  identifiziert das Steuerwort als Interruptsteuerwort.

Das Bit  $D_7$  des Wortes beeinflusst das Interruptfreigaberegister (Interruptenableflipflop) des programmierten Kanals. Bei Setzen dieses Bits ( $D_7 = 1$ ) erfolgt die Weitergabe von anhängigen Interruptanmeldungen, die durch die PIO-Interruptlogik ausgelöst wurden, an den Steuereingang  $\overline{INT}$  der CPU U880. Bei rückgesetztem Bit  $D_7$  ( $D_7 = 0$ ) wird die Weitergabe dieser Anmeldung an die CPU verhindert. In diesem Fall wird aber eine durch die Peripherie ausgelöste Interruptforderung in der PIO abgespeichert. Wird das Interruptfreigabeflipflop des betreffenden Kanals im späteren Programmablauf wieder gesetzt ( $D_7 = 1$ ), gelangt dieser abgespeicherte Interrupt zur Anmeldung an die CPU. Diese Anmeldung erfolgt auch, wenn seitens der Peripherie dieser Interrupt nicht mehr aktuell ist (also die den Interrupt auslösende Bedingung aufgehoben ist). Die PIO reagiert hierbei unabhängig von der Art der Interruptanmeldung, die durch die Betriebsart des Kanals festgelegt wird. Eine anhängige Anmeldung geht aber verloren, wenn zum Zeitpunkt dieser Interruptforderung die PIO bereits eine Interruptbearbeitung hat (Interruptbearbeitungsflipflop der Kanallogik gesetzt) und das auslösende Ereignis zum Zeitpunkt der Rückkehr aus dieser ISR nicht mehr aktuell ist.

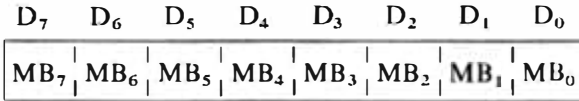
Die Bits  $D_6$ ,  $D_5$  und  $D_4$  werden in der Betriebsart Bitbetrieb zur logischen Auswahl der Kriterien, die eine Interruptanmeldung auslösen sollen, verwendet. Die Interruptanmeldung erfolgt, wenn die logische Verknüpfung (OR oder AND) der als aktiv definierten Pegel (H oder L) von durch eine Maskierung ausgewählten Bits des Portbusses wahr ist. Die durch diese Bedingungen spezifizierte logische Funktion hat (Interruptbearbeitungsflipflop der Kanallogik gesetzt) die gleiche Wirkung wie das  $\overline{STB}$ -Signal in den Bytebetriebsarten.

Hierbei erfolgt mit Bit  $D_6$  die Auswahl der logischen Verknüpfung. Bei Setzen dieses Bits ( $D_6 = 1$ ) wird die logische AND-Funktion, beim Rücksetzen ( $D_6 = 0$ ) die logische OR-Funktion angewendet.

Mit Bit  $D_5$  erfolgt die Festlegung des aktiven Pegels der Portleitungen. Ein Setzen dieses Bits ( $D_5 = 1$ ) definiert H-Pegel als aktiv; umgekehrt ist bei  $D_5 = 0$  L-Pegel das aktive Niveau.

Mit Setzen des Bits  $D_4$  ( $D_4 = 1$ ) erkennt der PIO-Kanal das nachfolgende Steuerwort

als Maskierungswort. Dieses Steuerwort dient zum Laden des 8-bit-Maskierungsregisters der Portlogik. Es hat das nachfolgend dargestellte Format:



Hierbei bedeutet das Rücksetzen der verwendeten Maskierungsbits ( $MB_n = 0$ ), daß die in der Wertigkeit korrespondierenden Bits des Portbusses ( $A_n$  bzw.  $B_n$ ) zur logischen Auswertung bei der Interrupterzeugung herangezogen werden. Bei Setzen der Bits ( $MB_m = 1$ ) werden die zugehörigen Portleitungen ( $A_m$  bzw.  $B_m$ ) nicht bewertet.

Dieses Maskierungswort braucht nicht geladen zu werden ( $D_4 = 0$  des Interruptsteuerworts), wenn alle Bits des Kanals zur Auswertung herangezogen werden sollen und wenn sich der PIO vor der Programmierung im rückgesetzten Zustand befand. Beim Rücksetzen des PIO wird das Maskierungsregister rückgesetzt; alle Maskierungsbits sind  $MB = 0$ .

Das Bit  $D_4$  des Interruptsteuerworts hat eine weitere Bedeutung für bestimmte Systemanwendungen. Das Einschreiben eines Interruptsteuerworts mit gesetztem Bit  $D_4$  ( $D_4 = 1$ ) zu einem beliebigen Zeitpunkt bewirkt, daß in allen Betriebsarten der IS U855 eine anhängige, jedoch nicht durch die CPU quittierte Interruptanforderung rückgesetzt wird. Das bedeutet, daß intern abgespeicherte Anmeldungen gelöscht werden. Die Weitergabe anhängiger Interruptforderungen wird durch die Kanalsteuerlogik bei rückgesetztem Interruptfreigabeflipflop, bei einem quittierten Interruptbearbeitungszustand des Kanals und bei inaktivem Interruptfreigabeeingang IEI ( $IEI = L$ ) verhindert.

Tafel 3.2.2. Programmierung der PIO

Interruptvektor							
$V_7$	$V_6$	$V_5$	$V_4$	$V_3$	$V_2$	$V_1$	0
Betriebsartenauswahl							
$M_1$	$M_0$	X	X	1	1	1	1
$M_1$	$M_0$	Betriebsart					
0	0	Byteausgabe					
0	1	Byteeingabe					
1	0	bidirektional					
1	1	Bitbetrieb					
Ein-/Ausgabe-Wahlwort, wenn Bitbetrieb ( $M_1 = M_0 = 1$ )							
$IO_7$	$IO_6$	$IO_5$	$IO_4$	$IO_3$	$IO_2$	$IO_1$	$IO_0$
Interruptkontrollwert							
EI	A/O	H/L	MF	0	1	1	1
EI	Interruptfreigabe						
A/O	AND- bzw. OR-Verknüpfung der logischen Funktion						
H/L	H- bzw. L-Pegel an den Portlinien sind aktiv (logisch 1)						
MF	Maskierungswort folgt als nächstes Steuerwort						
Maskierungswort, wenn $MF = 1$							
$MB_7$	$MB_6$	$MB_5$	$MB_4$	$MB_3$	$MB_2$	$MB_1$	$MB_0$
Interruptkontrollwert ohne Auswahlinformationen							
EI	X	X	X	0	0	1	1

Andererseits wird eine über die  $\overline{\text{INT}}$ -Linie an die CPU weitergeleitete Interruptanmeldung vom Prozessor nicht quittiert, wenn eine Busanforderung vorliegt ( $\overline{\text{BUSRQ}} = \text{L}$ ) oder wenn das prozessorinterne Interruptfreigabeflipflop  $\text{IFF}_1$  rückgesetzt ist. Für alle diese dargestellten Fälle hat also das Setzen des Bits  $D_4$  des Interruptsteuerworts Bedeutung. Eine mögliche Anwendung dieses Steuerworts besteht also z. B. darin, daß am Ende der ISR eines PIO-Kanals eine erneute (anhängige) Interruptforderung des gleichen Kanals zurückgenommen werden kann.

Nach dem Laden dieses Interruptsteuerworts ( $D_4 = 1$ ) erkennt der PIO-Kanal das nachfolgende Steuerwort unabhängig von der gewählten Betriebsart als Maskierungswort. Es wird das zugehörige PIO-Register geladen.

Die IS U855 bietet eine weitere Möglichkeit zur Beeinflussung des Interruptfreigabeflipflops der PIO-Kanallogik. Das hierzu vorgesehene Steuerwort hat folgendes Format:

$D_7$	$D_6$	$D_5$	$D_4$	$D_3$	$D_2$	$D_1$	$D_0$
EI	X	X	X	0	0	1	1

Durch Bit  $D_7$  kann das Interruptfreigabeflipflop (IFF) beeinflußt werden ( $D_7 = 0$ , IFF rückgesetzt;  $D_7 = 1$ , IFF gesetzt), ohne daß weitere Informationen in die PIO eingeschrieben werden müssen. Dieses Steuerwort, das durch die Bitbelegung  $D_0$  bis  $D_3$  charakterisiert wird, kann deshalb insbesondere zur Beeinflussung der PIO-Interruptfreigabe im Programmablauf benutzt werden.

### 3.2.4.5. Zusammenfassung

In Tafel 3.2.2 ist die Programmierung eines PIO-Kanals zusammenfassend dargestellt.

## 3.2.5. Beschreibung der Betriebsarten

### 3.2.5.1. Byteausgabe

Nach Auswahl der Betriebsart Byteausgabe mit dem entsprechenden Steuerwort nimmt das dem angesprochenen PIO-Kanal entsprechende  $\text{READY}$ -Signal (A RDY oder B RDY) einen inaktiven Zustand ein.

Das Ausgaberegister der Portlogik dieses Kanals ist direkt auf den Portbus (A oder B) geschaltet.

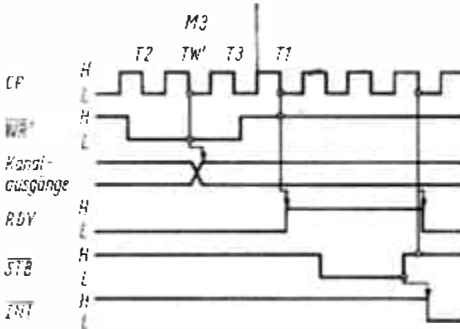
Durch Einschreiben eines Datenworts ( $C/\overline{D}$  Sel = 0) wird nun eine Byteausgabeoperation ausgelöst. Hierbei wird das Ausgaberegister des Kanals geladen. Bei Verwendung der Quittierungslogik durch die Peripherie ergibt sich das im Bild 3.2.5 dargestellte Zeitverhalten am PIO.

Das Einschreiben des Datenworts erfolgt in einem Schreibzyklus, der durch das intern in der IS U855 erzeugte Signal  $\overline{\text{WR}}$  gekennzeichnet ist:

$$\overline{\text{WR}} = \text{RD} + \overline{\text{CE}} + \overline{C/\overline{D}} + \overline{\text{IORQ}}. \quad (3.2.1)$$

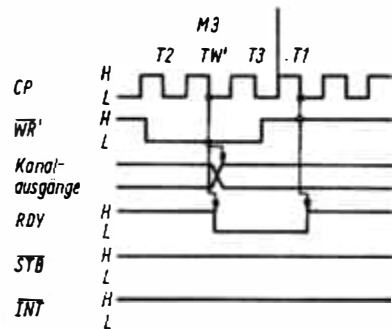
Nachdem dieses Signal inaktiv geworden ist, also nach dessen steigender Flanke, sind die von der CPU übertragenen Daten auf dem entsprechenden Portbus verfügbar. Dieser Zustand wird der Peripherie angezeigt, indem das zugehörige  $\text{READY}$ -Signal (A RDY oder B RDY) nach der nächsten fallenden Flanke des Systemtakts aktiv wird (also H-Pegel führt). Eine positive (steigende) Flanke des in der Peripherie zu erzeugenden Quittierungs-

signals ( $\overline{A\ STB}$  bzw.  $\overline{B\ STB}$ ) bewirkt danach ein Rücksetzen des READY-Signals ebenfalls nach der nächsten fallenden Taktflanke. Gleichzeitig kann von dieser Flanke des  $\overline{STROBE}$ -Signals in Abhängigkeit vom zuvor programmierten Zustand des Interrupt-freigabefllops der Portlogik und abhängig von der aktuellen Priorität des PIO-Kanals eine Interruptanmeldung an die CPU ausgelöst werden.



**Bild 3.2.5. Zeitverhalten der PIO in der Ausgabemode, Quittierungsbetrieb**

M3	OUTPUT-Maschinenzyklus der CPU
CP	Systemtakt (T2, TW', T3, T1 Taktzustände der CPU)
$\overline{WR'}$	internes Schreibsignal der PIO; s. Gl. (3.2.1)
$\overline{RD4}, \overline{STB}$	Quittierungslinien des PIO-Ports
$\overline{INT}$	Interruptausgang der PIO (nur der betreffende PIO-Kanal sei interruptwirksam)



**Bild 3.2.6. Zeitverhalten der PIO in der Ausgabemode,  $\overline{STB}$ -Linie fest beschaltet**

M3	OUTPUT-Maschinenzyklus der CPU
CP	Systemtakt (T2, TW', T3, T1 Taktzustände der CPU)
$\overline{WR'}$	internes Schreibsignal der PIO; s. Gl. (3.2.1)
$\overline{RDY}, \overline{STB}$	Quittierungslinien des PIO-Ports
$\overline{INT}$	Interruptausgang der PIO (nur Wirkung des betreffenden Ports dargestellt)

Die Anwendung dieses unkomplizierten Quittierungsbetriebs erfolgt hauptsächlich bei der Bedienung langsamer Peripheriegeräte und bei der Synchronisierung der Peripherie mit der Mikrorechnerschnittstelle beim Vorhandensein bestimmter zeitlicher Bedingungen.

Sollen Ausgabeoperationen ohne Quittierungsbetrieb ausgeführt werden, kann der  $\overline{STROBE}$ -Eingang des Kanals ( $\overline{A\ STB}$  bzw.  $\overline{B\ STB}$ ) an einen festen Pegel (H oder L) gelegt werden. Es ergibt sich dann an der Portlogik das im Bild 3.2.6 dargestellte Zeitverhalten.

Nach Initialisierung der Betriebsart ist die entsprechende READY-Linie wiederum inaktiv. Das Einschreiben eines Datenworts bewirkt ein Umschalten des Signals (A RDY bzw. B RDY) auf H-Pegel nach Beendigung des Schreibzyklus. Da am  $\overline{STROBE}$ -Eingang jetzt keine steigende Flanke auftreten kann, bleibt das READY-Signal bis zur nächsten Ausgabeoperation des Kanals aktiv. Das Signal (A RDY bzw. B RDY) wird nach der fallenden Taktflanke des automatisch von der CPU bei diesem OUT-Befehl (gekennzeichnet durch  $\overline{WR'} = 0$ ) eingefügten Waitzustands rückgesetzt. Nach Beendigung des Schreibbefehls ( $\overline{WR'} = 1$ ) wird READY wieder aktiv.

Das READY-Signal ist also während der Änderung und des Einschwingens der neuen Information auf dem Portbus (A bzw. B) inaktiv. Seine positive Flanke kann durch die Peripherie zur flankengesteuerten Zwischenspeicherung der Ausgabedaten verwendet werden (z. B. direkte Ansteuerung der Trigger D174 durch dieses Signal). Da am  $\overline{STROBE}$ -Eingang keine positive Flanke auftritt, erfolgt keine Anmeldung von Interrupts.

Eine weitere Möglichkeit der Beschaltung der IS U855 in dieser Betriebsart stellt das Zusammenschalten beider Quittierungssignale des Kanals dar ( $\overline{A\ STB} = A\ RDY$  bzw.  $\overline{B\ STB} = B\ RDY$ ). Das hierbei auftretende Zeitverhalten ist im Bild 3.2.7 dargestellt.

Die READY-Linie wird nach einer OUT-Operation wie in den bereits beschriebenen Fällen aktiv. Die hierdurch ausgelöste positive Flanke am  $\overline{\text{STROBE}}$ -Eingang bewirkt aber sofort ein Rücksetzen von READY nach der nächsten fallenden Flanke des Systemtakts. Es wird somit vom READY-Ausgang ein positiver Impuls mit der Dauer von etwa einer Taktperiode erzeugt, nachdem der Portbus bereits einen eingeschwungenen Zustand erreicht hat.

Obwohl der  $\overline{\text{STROBE}}$ -Eingang eine positive Flanke empfängt, wird keine Interruptanmeldung ausgelöst. Das ist dadurch begründet, daß der auf einen OUT-Befehl folgende Maschinenzyklus systembedingt grundsätzlich ein  $\overline{\text{M1}}$ -Zyklus ist. Bei aktivem  $\overline{\text{M1}}$ -Signal wird aber die interne Interruptlogik aller peripheren Geräte (also auch der PIO) nicht verändert, um Einschwingvorgänge bei der Interruptquittierung zu vermeiden.

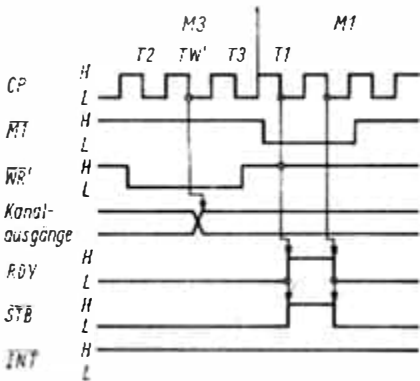


Bild 3.2.7

Zeitverhalten der PIO in der Ausgabemode,  $\overline{\text{STB}}$ -Linie mit RDY-Linie verbunden

- M3 OUTPUT-Maschinenzyklus der CPU
- CP Systemtakt (T2, TW', T3, T1 Taktzustände der CPU)
- M1 CPU-Steuerausgang bzw. PIO-Steuereingang
- $\overline{\text{WR}}$  internes Schreibsignal der PIO; s. Gl. (3.2.1)
- RDY,  $\overline{\text{STB}}$  Quittierungslinien des PIO-Ports
- $\overline{\text{INT}}$  Interruptausgang der PIO (nur Wirkung des betreffenden Ports dargestellt)

In der Peripherie kann der so erzeugte READY-Impuls zur flankengesteuerten (positive oder negative Flanke) oder zur zustandsgesteuerten Zwischenspeicherung der auf den Portbus geschalteten Ausgabedaten verwendet werden (z. B. direkte Ansteuerung von parallelbetriebenen Schieberegistern D195 als 4-bit-Register).

### 3.2.5.2. Byteeingabe

Nach Initialisierung des angesprochenen PIO-Kanals in dieser Betriebsart nimmt die zugehörige READY-Linie (A RDY bzw. B RDY) einen inaktiven Zustand ein. Mit diesem Status wird der Peripherie angezeigt, daß das Eingaberegister keine neuen Daten erwartet. Umgekehrt wird mit aktivem READY-Ausgang signalisiert, daß das Porteingaberegister zuvor von der CPU gelesen wurde.

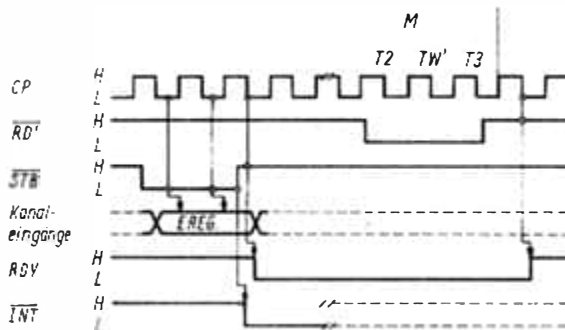


Bild 3.2.8

Zeitverhalten der PIO in der Eingabemode, Quittierungsbetrieb

- M INPUT-Maschinenzyklus der CPU
- CP Systemtakt (T2, TW', T3 Taktzustände der CPU)
- $\overline{\text{RD}}$  internes Leseignal der PIO; s. Gl. (3.2.2)
- RDY,  $\overline{\text{STB}}$  Quittierungslinien des PIO-Ports
- E-REG Zustand des internen Eingaberegisters der PIO-Port-Logik
- $\overline{\text{INT}}$  Interruptausgang der PIO (nur der betreffende PIO-Kanal sei interruptwirksam)

Das Einspeichern der auf dem **entsprechenden** Portbus (A oder B) anliegenden Daten erfolgt mit aktivem Zustand des zugehörigen  $\overline{\text{STROBE}}$ -Eingangs.

Die Verwendung dieses somit ermöglichten einfachen Quittierungsbetriebs kann zur Synchronisation oder zur schnellen Bedienung der Paralleleingabeschnittstelle dienen. Das Zeitverhalten an der IS U855 bei Quittierungsbetrieb ist im Bild 3.2.8 dargestellt.

Bei dieser Anwendung muß nach Initialisierung des Kanals eine blinde (also nicht auszuwertende) Dateneingabeoperation ausgeführt werden, damit die  $\overline{\text{READY}}$ -Linie einen aktiven Zustand einnimmt. Dieser H-Pegel zeigt der Peripherie an, daß Daten in das Eingaberegister der PIO-Kanallogik eingeschrieben werden können. Dieses Einschreiben erfolgt zustandsgesteuert mit aktivem Pegel des zugehörigen  $\overline{\text{STROBE}}$ -Signals. Mit Inaktivwerden dieses Eingangs (also mit der steigenden Flanke) wird ein Rücksetzen des  $\overline{\text{READY}}$ -Ausgangs nach der folgenden fallenden Flanke des Systemtakts ausgelöst.  $\overline{\text{READY}}$  signalisiert also der Peripherie, daß sich bereits neue Daten im Eingaberegister des Ports befinden. Gleichzeitig wird von der positiven  $\overline{\text{STROBE}}$ -Flanke bei freigegebener Interruptlogik des Kanals eine Interruptanmeldung erzeugt. In der dadurch ausgelösten Interruptserviceroutine (ISR) könnte nun beispielsweise das Lesen des Eingaberegisters durch die CPU erfolgen.

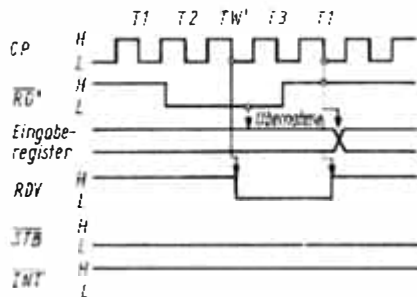
Die Leseoperation der CPU (IN-Befehl) ist durch das in der PIO generierte Signal  $\overline{\text{RD}}'$  gekennzeichnet:

$$\overline{\text{RD}}' = \overline{\text{RD}} + \overline{\text{CE}} + \overline{\text{C/D}} + \overline{\text{IORQ}}. \tag{3.2.2}$$

Nach Beendigung dieser Operation (inaktives  $\overline{\text{RD}}'$ -Signal) erfolgt das erneute Setzen des  $\overline{\text{READY}}$ -Ausgangs nach der nächsten fallenden Taktflanke. Der Peripherie wird angezeigt, daß der PIO neue Daten erwartet. Der Ausgangszustand für einen neuen Einschreibvorgang ist somit erreicht.

Sollen Eingabeoperationen ohne Quittierungsbetrieb ausgeführt werden, muß der entsprechende  $\overline{\text{STROBE}}$ -Eingang mit L-Pegel beschaltet werden. Das Einschreiben der auf dem Portbus anliegenden Daten erfolgt nun direkt in das Porteingaberegister. Bei einem Lesezyklus durch die CPU (IN-Befehl), der wiederum durch aktives  $\overline{\text{RD}}'$ -Signal gekennzeichnet ist, wird die  $\overline{\text{READY}}$ -Leitung nach der negativen Taktflanke des automatisch von der CPU eingeschobenen Waitzustands rückgesetzt. Nach Ende der IN-Operation wird sie, wie bereits beschrieben, wieder aktiv.

Der inaktive Pegel der  $\overline{\text{READY}}$ -Linie kann bei dieser Anwendung in der Peripherie benutzt werden, um die Übergabe von Daten an den Portbus zu unterbinden. Damit ist sichergestellt, daß während einer Einschreibeoperation eingeschwungene Daten in die CPU übernommen werden. Da am  $\overline{\text{STROBE}}$ -Eingang kein Pegelwechsel auftritt, können keine Interruptanmeldungen ausgelöst werden. Bild 3.2.9 zeigt das Zeitverhalten am PIO bei diesem Anwendungsfall.



**Bild 3.2.9**  
Zeitverhalten der PIO in der Eingabemode,  
*STB-Linie ständig aktiv*

- M3 INPUT-Maschinenzyklus der CPU
- CP Systemtakt (T1, T2, TW', T3 Taktzustände der CPU)
- $\overline{\text{RD}}'$  internes Lesesignal der PIO; s. Gl. (3.2.2)
- $\overline{\text{RDY}}, \overline{\text{STB}}$  Quittierungslinien des PIO-Ports
- $\overline{\text{INT}}$  Interruptausgang der PIO (nur Wirkung des betreffenden Kanals dargestellt)

### 3.2.5.3. Bidirektionaler Betrieb

Bei Auswahl dieser Betriebsart für den Kanal A der Ein-/Ausgabe-Einheit muß Kanal B in die Mode Bitbetrieb gesetzt werden. Das ist erforderlich, weil alle vier Quittierungssignale (A RDY,  $\overline{A STB}$ , B RDY,  $\overline{B STB}$ ) für die Steuerung des bidirektionalen Datenaustauschs über den Portbus A durch die Peripherie genutzt werden. Im Bild 3.2.10 ist das Zeitverhalten am PIO bei ausgewählter Betriebsart 2 dargestellt.

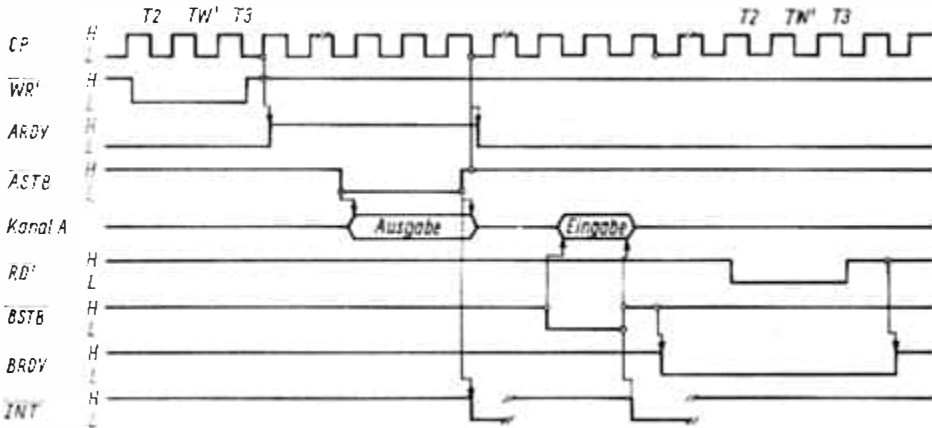


Bild 3.2.10. Zeitverhalten der PIO in der bidirektionalen Mode

M3	OUTPUT- bzw. INPUT-Maschinenzyklus der CPU
CP	Systemtakt (T2, TW', T3 Taktzustände der CPU)
$\overline{WR'}$	internes Schreibsignal der PIO; s. Gl. (3.2.1)
A RDY	Quittierungslinien des PIO-Ports A
$\overline{A STB}$	
$\overline{RD'}$	internes Lesesignal der PIO; s. Gl. (3.2.2)
$\overline{B STB}$ , B RDY	Quittierungslinien des PIO-Ports B
$\overline{INT}$	Interruptausgang der PIO (nur von Quittierungslinien ausgelöste Interruptmeldungen dargestellt)

Die Quittierungssignale des Kanals A der IS U855 (A RDY,  $\overline{A STB}$ ) dienen zur Steuerung des Datentransports in Richtung Peripherie (Ausgabe). Analog zur Betriebsart Byteausgabe wird der Ausgang A RDY nach einer Schreiboperation der CPU (OUT-Befehl) aktiviert.

Er signalisiert somit der Peripherie, daß neue Ausgabedaten im Ausgaberegister des Kanals A verfügbar sind. Diese Daten können nun durch Aktivierung des Eingangs  $\overline{A STB}$  auf den bidirektionalen Port-A-Bus geschaltet und durch die Peripherie empfangen werden.

Durch Rücksetzen dieses Quittungssignals ( $\overline{A STB} = 1$ ) wird der Portbus wieder in einen hochohmigen Zustand versetzt. Gleichzeitig wird mit der nächsten fallenden Flanke des Systemtakts der Ausgang A RDY rückgesetzt (A RDY = 0), um der Peripherie die Beendigung des Ausgabezyklus und die Freigabe des Port-A-Busses anzuzeigen. Mit der steigenden Flanke von  $\overline{A STB}$  wird bei gesetztem Interruptfreigabeflipflop des Kanals A und freigegebener Interruptlogik der PIO ( $IE1 = 1$ ) eine Interruptanmeldung an die CPU ausgelöst. Bei Quittierung dieser Anmeldung durch die CPU wird der Interruptvektor des Kanals A gelesen.

Der Datentransport von der Peripherie zur IS U855 über den Port-A-Bus wird durch

die Quittierungssignale des Kanals B ( $B\text{ RDY}$ ,  $\overline{B\text{ STB}}$ ) gesteuert. Das Zeitverhalten entspricht hierbei dem der Betriebsart Byteeingabe.

Die Daten werden also mit aktivem  $\overline{B\text{ STB}}$ -Signal in das Eingaberegister des Kanals A eingeschrieben. Nach dem Inaktivwerden dieses Signals wird der Ausgang  $B\text{ RDY}$  rückgesetzt, um der Peripherie anzuzeigen, daß der Eingabezyklus beendet ist und sich aktuelle Daten im Eingaberegister befinden. Die positive Flanke von  $\overline{B\text{ STB}}$  löst eine Interruptanmeldung des Kanals B aus, wenn das Interruptfreigabeflipflop dieses Ports gesetzt und die Interruptlogik freigegeben ist. Die Quittierung dieser Anmeldung durch die CPU hat das Aussenden des Interruptvektors des Kanals B zur Folge. Da der Kanal B in der Betriebsart Bitbetrieb arbeitet, können Zweideutigkeiten bei der Interruptanmeldung auftreten, weil in dieser Mode die Anmeldung von Interrupts durch eine maskierbare Auswertung der auf dem Port-B-Bus liegenden Dateninformationen erfolgt.

Der Kanal B kann also bei entsprechender Portbelegung die gleiche Interruptservice-routine aufrufen wie Kanal A bei Quittierung des Eingabebetriebs (in der bidirektionalen Betriebsart). Wenn diese Zweideutigkeit der Interruptanmeldung vermieden werden soll, muß durch Setzen des Maskierungsregisters des Kanals B eine Anmeldung des im Bitbetrieb arbeitenden Ports verhindert werden. Dieser Kanal kann dann durch die CPU mit Polling bedient werden.

Bei der Eingabe im bidirektionalen Betrieb ist eine weitere Besonderheit zu beachten. Bei einer Leseoperation der CPU ( $\overline{RD'} = 0$ ) werden nur dann die Daten des Eingaberegisters gelesen, wenn das Signal  $\overline{A\text{ STB}}$  inaktiv ist. Im anderen Fall ( $\overline{A\text{ STB}} = 0$ ) wird der Inhalt des Ausgaberegisters des Ports A gelesen. Zur Vermeidung dieses Falls kann in der Peripherie  $\overline{A\text{ STB}}$  mit dem negierten  $B\text{ RDY}$ -Signal mit Hilfe OR logisch verknüpft werden.

Das Quittungssignal  $\overline{A\text{ STB}}$  nimmt dann während einer Leseoperation ( $B\text{ RDY} = 0$ ) einen inaktiven Zustand ( $\overline{A\text{ STB}} = 1$ ) ein. Die Eingabedaten werden also dem Eingaberegister des Kanals A entnommen.

#### 3.2.5.4. Bitbetrieb

Bei Anwendung der Betriebsart Bitbetrieb werden die Quittierungssignale des ausgewählten Kanals nicht benutzt. Das READY-Signal ( $A\text{ RDY}$  bzw.  $B\text{ RDY}$ ) führt L-Pegel, ausgenommen  $B\text{ RDY}$ , wenn Kanal A im bidirektionalen Betrieb arbeitet.

Eine Datenschreiboperation der CPU (OUT-Befehl) bewirkt ein Einschreiben der Daten in das Ausgaberegister der Kanallogik der PIO. Mit dem Inhalt des Ein-/Ausgabewahlregisters ist festgelegt, welche Bits dieses Registers direkt auf den Portbus geschaltet sind (also als Ausgänge wirken). Die anderen Bits des Busses wirken als Eingänge. Bei einem auf den PIO-Kanal bezogenen Lesebefehl der CPU (IN-Befehl) erfolgt die Übernahme der auf dem Portbus liegenden Daten flankengesteuert mit der fallenden Flanke des  $\overline{RD}$ -Signals in das Eingaberegister der Kanallogik. Die mit dieser Flanke übernommenen aktuellen Portdaten werden während des laufenden Lesezyklus an den Prozessor gesendet. Dieses Datenwort wird je nach Inhalt des Ein-/Ausgabewahlregisters aus Eingabebits (vom Portbus) und Ausgabebits (aus dem Ausgaberegister) gebildet. Das Zeitverhalten am PIO ist im Bild 3.2.11 dargestellt.

Die Auslösung von Interruptanmeldungen erfolgt in der Betriebsart Bitbetrieb durch logische Auswertung der auf dem Portbus anliegenden aktuellen Bitbelegung. Diese Auswertung bezieht sich also sowohl auf die Bits des Portbusses, die als Eingänge wirken und ihre Belegung von der Peripherie erhalten, als auch auf die als Ausgänge definierten Bits,

die durch das Ausgaberegister belegt werden. Durch den Inhalt des Maskierungsregisters ist bestimmt, welche dieser Bits einen Einfluß auf die logische Auswertung haben. Das 2-bit-Maskierungssteuerregister legt nun fest, mit welchem Pegel (H oder L) und welcher Funktion (OR oder AND) die Auswahl erfolgt. Als logische Funktion der Bitauswertung kann deshalb die AND-, NAND-, OR- oder NOR-Verknüpfung der nichtmaskierten Bits des Portbusses verwendet werden.

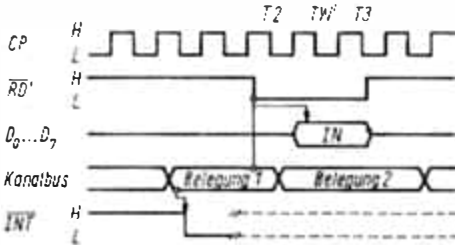


Bild 3.2.11

## Zeitverhalten der PIO in der Bitmode

M3	INPUT-Maschinenzyklus der CPU
CP	Systemtakt (T2, TW, T3 Taktzustände der CPU)
$\overline{RD}$	internes Lesesignal der PIO; s. Gl. (3.2.2)
$D_0 \dots D_7$	Belegung des Systemdatenbusses (IN gültige Eingabedaten)
$\overline{INT}$	Interruptausgang der PIO (nur Wirkung des betreffenden PIO-Ports durch Auswertung der logischen Belegung dargestellt)

Die Interruptanmeldung erfolgt bei freigegebener Interruptlogik der PIO und gesetztem Interruptfreigabeflipflop, wenn diese logische Funktion der ausgewählten Portbusbits erfüllt wird (also bei einem 0/1-Übergang). Bleibt diese Verknüpfung wahr (unabhängig vom Ereignis), wird nach Verlassen des Interruptbetriebs (RETI-Befehl) hierdurch keine neue Anmeldung ausgelöst. Voraussetzung für die Interruptauslösung ist also, daß sich bei freigegebener Interruptlogik der PIO durch Veränderung der Bitbelegung ein Sprung der verwendeten Funktion auf logisch 1 vollzieht. Es erfolgt somit ebenfalls keine neue Interruptanmeldung, wenn der Übergang der logischen Funktion der Bitbelegung auf logisch 1 noch bei blockierter Interruptlogik (durch Bearbeitung der vom Kanal ausgelösten Interruptserviceroutine) ausgelöst wird. Die anhängige Interruptforderung wird jedoch PIO-intern abgespeichert und kann nach dem Verlassen des Interruptbearbeitungszustands (RETI-Befehl) eine Anmeldung auslösen. Voraussetzung hierfür ist aber, daß bei Abarbeitung des RETI-Befehls die logische Funktion noch erfüllt ist.

Da bei aktivem  $\overline{MI}$ -Signal der Interruptzustand der IS U855 nicht verändert werden darf (Einschwingen der Interruptprioritätenkette), erfolgt bei Erfüllung der logischen Funktion während eines  $\overline{MI}$ -Zyklus die Anmeldung des Interrupts nach der Rückflanke des  $\overline{MI}$ -Signals (wiederum vorausgesetzt, daß die logische Funktion dann noch erfüllt ist).

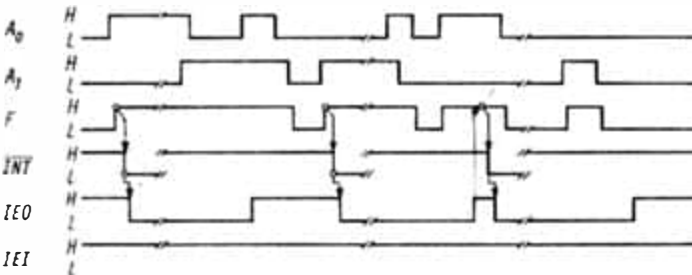


Bild 3.2.12. Interruptanmeldung in Bit-Mode der PIO (Beispiel)

$A_0 \dots A_1$	Aktivitäten der zugehörigen Portlinien der PIO
F	Belegung der intern in der PIO erzeugten logischen Funktion
$\overline{INT}$	Interruptausgang der PIO (nur Wirkungen des Ports A bei zugehörigen Anmeldungen dargestellt)
IEO	Interruptfreigabeausgang der PIO (zeigt Bearbeitungszustand der durch die dargestellten Interruptforderungen ausgelösten ISR an)
IEI	Interruptfreigabeingang der PIO (bei vorliegendem Beispiel ständig aktiv)

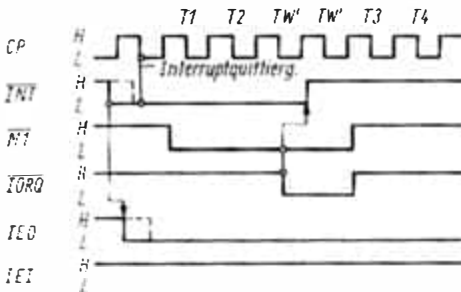
Im Bild 3.2.12 sind noch einmal an einem Beispiel die Bedingungen für die Interruptanmeldung im Bitbetrieb dargestellt. In diesem Beispiel sei das Interruptfreigabeflipflop gesetzt, und die Bits  $A_0$  und  $A_1$  sollen über die logische OR-Funktion (positive Logik) Interrupts auslösen. Das Signal IEO der PIO zeigt den aktuellen Bearbeitungsstatus der vom Kanal aufgerufenen ISR an.

### 3.2.6. Interruptbearbeitung

Die Erzeugung von Interrupts erfolgt in den bytebezogenen Betriebsarten mit Hilfe des Quittierungssignals  $\overline{\text{STROBE}}$  bzw. im Bitbetrieb durch logische Auswertung der aktuellen Portbusbelegung. Voraussetzung hierfür ist, daß die betrachtete PIO (-Kanal) in der Interruptkette (daisy chain) die höchste Priorität aufweist ( $\text{IEI} = 1$ ).

Diese Interruptprioritätenkette wird im System U880 durch Reihenschaltung der peripheren Bauelemente (PIO, SIO, CTC) mit den Interruptfreigabesignalen IEI und IEO gebildet. Ein H-Pegel am Eingang IEI der PIO bedeutet hierbei, daß die interne Interruptlogik freigegeben ist und eine Interruptanmeldung in Abhängigkeit vom Zustand des Interruptfreigabeflipflops des betrachteten Ports an die CPU weitergegeben wird. Ein L-Pegel am Ausgang IEO der PIO bewirkt, daß bei nachfolgenden peripheren IS die Interruptanmeldung blockiert bzw. daß deren in Bearbeitung befindliche ISR bei Interruptfreigabe der CPU (Befehl EI erfolgte) durch das höherwertigere Gerät unterbrochen wird.

Die IS U855 enthält zwei Ein-/Ausgabe-Kanäle, die in bezug auf die Interruptbearbeitung gleichartig wirken (zwei getrennte Interruptfreigabeflipflops, zwei Vektorregister). Hierbei besitzt der Kanal A die höhere Priorität, liegt also in einer Interruptkette weiter vorn als Kanal B. Damit bei der Quittierung einer Interruptanmeldung durch die CPU diese Prioritätenkette einen eingeschwungenen Zustand erreicht, wird die Interruptlogik aller PIO (der Kette und der anderen peripheren IS) während des aktiven  $\overline{\text{M1}}$ -Signals blockiert. Die Interruptquittierung der anmeldenden PIO (die gekennzeichnet ist durch  $\text{IEI} = 1$  und  $\text{IEO} = 0$ ) erfolgt durch gleichzeitiges Auftreten der aktiven Zustände der Signale  $\overline{\text{M1}}$  und  $\overline{\text{IORQ}}$  in einem in den Befehlsablauf von der CPU eingeschobenen Quittierungszyklus. Das entsprechende Zeitverhalten ist im Bild 3.2.13 dargestellt.



**Bild 3.2.13**  
Zeitverhalten der PIO im Interruptquittierungszyklus

- CP            Systemtakt (T1, T2, TW, T3, T4 Taktzustände der CPU)
- $\overline{\text{INT}}$         Interruptausgang der PIO
- $\overline{\text{M1}}, \overline{\text{IORQ}}$     Steuerausgänge der CPU bzw. Steuereingänge der PIO
- IEI            Interruptfreigabeeingang der PIO
- IEO            Interruptfreigabeausgang der PIO

Nach Empfang dieser Quittierung befindet sich der PIO-Kanal im Interruptbearbeitungszustand. Er sendet seinen Interruptvektor an die CPU, und programmäßig wird eine durch diese Interruptanmeldung aufgerufene ISR eingeschoben. Die  $\overline{\text{INT}}$ -Linie wird durch diesen PIO-Kanal wieder freigegeben. Das Ende einer ISR wird durch den CPU-Rückkehrbefehl RETI gekennzeichnet.

Die im Bearbeitungszustand befindliche PIO dekodiert parallel zur CPU die auf dem Datenbus ankommenden Befehle (die gekennzeichnet sind durch  $\overline{M1}$ -Zyklen). Erkennt sie die Befehlsfolge ED, 4D (2-byte-Befehl RETI), so verläßt sie den Interruptbearbeitungszustand und aktiviert ihren Ausgang IEO (vorausgesetzt IEI = 1). Dieses Zeitverhalten ist im Bild 3.2.14 dargestellt.

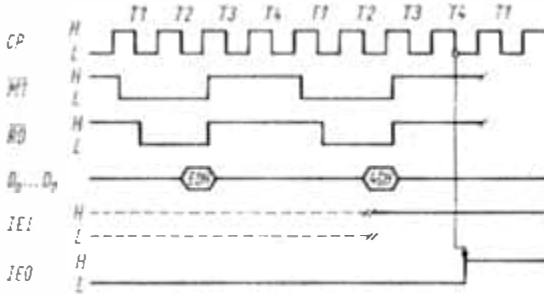


Bild 3.2.14

Zeitverhalten der PIO bei Verlassen des Interruptbearbeitungszustands

CP Systemtakt (T1, T2, T3, T4 Taktzustände der CPU)

$\overline{M1}$ ,  $\overline{RD}$  Steuerausgänge der CPU bzw. Steuereingänge der PIO

$D_0 \dots D_7$  Belegung des Systemdatenbusses (gültige Befehlscode: EDH und 4DH)

IEI Interruptfreigabeeingang

IEO Interruptfreigabeeingang

Bei der Interruptbearbeitung können aber auch die nachfolgend beschriebenen Fälle auftreten. Wenn während der Abarbeitung der ISR des betrachteten PIO-Kanals (der sich im Interruptbearbeitungszustand befindet) eine höherwertige periphere IS (in diesem Fall: PIO) einen Interrupt anmeldet ( $\overline{INT} = 0$ ), dann schaltet dessen Freigabeeingang IEO auf L-Pegel ( $IEO = 0$ ). Die Prioritätenkette gibt dieses Potential an die nachfolgenden IS weiter. Die niederwertigere, in Bearbeitung befindliche PIO, empfängt also auch  $IEI = 0$ .

Ist nun in der laufenden ISR (der niederwertigeren PIO) das Interruptfreigabeflipflop der CPU gesetzt (mit dem Befehl EI der CPU), so unterbricht die Interruptanmeldung des höherwertigen PIO (eigentlich: PIO-Kanal) die laufende ISR. Die Anmeldung wird durch die CPU quittiert (Bild 3.2.13), und programmäßig wird eine ISR des höherwertigen IS eingefügt. Nach Beendigung dieser ISR empfängt die höherwertige PIO die RETI-Instruktion, da ihr Interruptfreigabeeingang H-Pegel führt ( $IEI = 1$ ).

Der Interruptfreigabeeingang wird wieder auf H-Pegel geschaltet (Bild 3.2.14), und die niederwertigere IS empfängt am IEI-Eingang ebenfalls wieder H-Pegel. Die CPU bearbeitet daraufhin die zurückgespeicherte und zuvor unterbrochene ISR der niederwertigen PIO zu Ende. Der daraufhin folgende RETI-Befehl setzt nun den Interruptbearbeitungszustand der niederwertigen PIO zurück. Der Ausgangszustand ist wieder erreicht.

Ist im anderen Fall das Interruptfreigabeflipflop der CPU im Befehlsablauf der ISR der niederwertigeren PIO nicht gesetzt worden, so erfolgt keine Unterbrechung des laufenden Unterprogramms. Der höherwertige PIO-Kanal meldet weiter seinen Interrupt an ( $\overline{INT} = 0$ ); sein Freigabeeingang führt ebenfalls weiter L-Potential ( $IEO = 0$ ). Somit empfängt die niederwertigere IS am Eingang IEI ebenfalls L-Pegel ( $IEI = 0$ ). Damit nun der im Interruptbearbeitungszustand befindliche PIO-Kanal die RETI-Instruktion erkennen kann, muß der IEO-Ausgang der anmeldenden, aber nicht bearbeiteten höherwertigen IS nach Erkennen des Befehlsbytes ED (1. RETI-Byte) für die Dauer eines  $\overline{M1}$ -Zyklus auf H-Pegel geschaltet werden. Dadurch führt der IEI-Eingang bei Empfang des zweiten Befehlsbytes 4D der RETI-Instruktion H-Potential und kann den Interruptbearbeitungszustand verlassen.

Der IEO-Ausgang dieses (niederwertigeren) PIO-Kanals führt aber weiterhin L-Potential, da der höherwertigere PIO-Port weiter seine nichtquittierte Interruptanmeldung aus-

sendet und somit am IEO-Ausgang wieder L-Pegel führt, der durch die Prioritätenkette weitergegeben wird.

Zusammenfassend läßt sich aus diesen Bedingungen die logische Funktion des PIO-Ausgangs IEO darstellen:

$$IEO = IEI \cdot \overline{RF2A} \cdot \overline{RF2B} \cdot (\overline{INTPA} \cdot \overline{INTPB} + RFI). \quad (3.2.3)$$

IEO	Interruptfreigabeausgang des PIO (H aktiv)
IEI	Interruptfreigabeeingang des PIO (H aktiv)
RF2A } RF2B }	Kombinationssignale, die sich aus dem Interruptbearbeitungszustand des Kanals (A oder B) und dem in einem FF zwischengespeicherten dekodierten zweiten Byte des RETI-Befehls zusammensetzen (H aktiv)
INTPA } INTPB }	Interruptanforderungen der Kanäle (A oder B), die noch nicht durch die CPU quittiert wurden (H aktiv)
RFI	Zustand des FF, in dem das dekodierte erste Byte des RETI-Befehls für einen $\overline{M1}$ -Zyklus zwischengespeichert wird (H aktiv).

### 3.2.7. Einsatz der PIO in Mikrorechnersystemen

#### 3.2.7.1. Hardware

Die Signalkonfiguration der IS U855 erlaubt eine direkte Zusammenschaltung mit dem Prozessor U880. In Mikrorechnerminimalsystemen kann deshalb eine direkte Kopplung der Daten- und Steuerbuslinien erfolgen.

Ansteuerseitig können das Freigabesignal  $\overline{CE}$  und die Auswahlsignale  $B/\overline{A}$  Sel,  $C/\overline{D}$  Sel direkt mit Adreßleitungen der CPU beschaltet werden. In größeren Mikrorechnern wird aufgrund der Anzahl peripherer Geräte der Einsatz eines Adreßdekoders notwendig. Üblicherweise wird hierbei aus den Adreßlinien  $A_2 \dots A_7$  das Freigabesignal  $\overline{CE}$  gebildet, während die Auswahlsignale mit den niederwertigen Adressen  $A_0$  und  $A_1$  beschaltet werden. Es ergibt sich somit ein Adressierungsraum für 128 PIO-Kanäle.

Reicht diese Anzahl bei besonderen Anwendungsfällen nicht aus, so erlaubt die Befehlskonfiguration der OUT- bzw. IN-Instruktion die Einbeziehung der höherwertigen Adressen  $A_8 \dots A_{15}$  in die Adreßdekodierung. Es können dann aber Probleme bei der Zuordnung der ISR (ebenfalls durch Interruptvektor auf 128 begrenzt) auftreten.

Die Zeitverhältnisse an der PIO U855 gestatten die Pufferung des Adreß-, Steuer- und Datenbusses sowie des Systemtakts. Üblicherweise erfolgt die Treibung des Adreß- und Steuerbusses prozessorseitig bzw. in Busverlängerungseinheiten, um ausreichende Pegel und Lastfaktoren zur Ansteuerung der Peripherie bereitzustellen. Die Datenbuspufferung erfolgt sowohl CPU-seitig als auch periphereseitig, um Störgrößen und große Leitungskapazitäten (z. B. bei Anschluß großer Standardspeicher) auf dem Bus zu vermeiden. Da der Datenbus bidirektional arbeitet, muß an den Treiberschaltkreisen eine Richtungsumschaltung vorgesehen werden. Periphereseitig muß diese Umschaltlogik auf Schreib- und Lesevorgänge und auf Interruptquittierungen der zugehörigen Peripherie reagieren. Bild 3.2.15 zeigt das Prinzipschaltbild einer Richtungsumschaltung für ein PIO-Bauelement, das über einen Datenbustreiber mit dem Prozessor korrespondiert. Die Umschaltlogik reagiert jeweils nur bei Aktivierung des Peripherielements. Im Interruptquittierungszyklus erfolgt eine Richtungsumschaltung in Abhängigkeit vom Zustand des IEI-Eingangs und des IEO-Ausgangs der PIO. Die Konfiguration  $IEI = 1$  und  $IEO = 0$  zeigt hierbei an, daß das Element eine priorisierte Interruptanmeldung zum Prozessor

gesendet hat. Bei inaktiven PIO muß dennoch der Datenbus übertragen werden, damit die parallele Auswertung des RETI-Befehls durch die Peripherieelemente erfolgen kann.

Der Bustreiber muß deshalb immer aktiviert sein ( $\overline{CS} = L$ ).

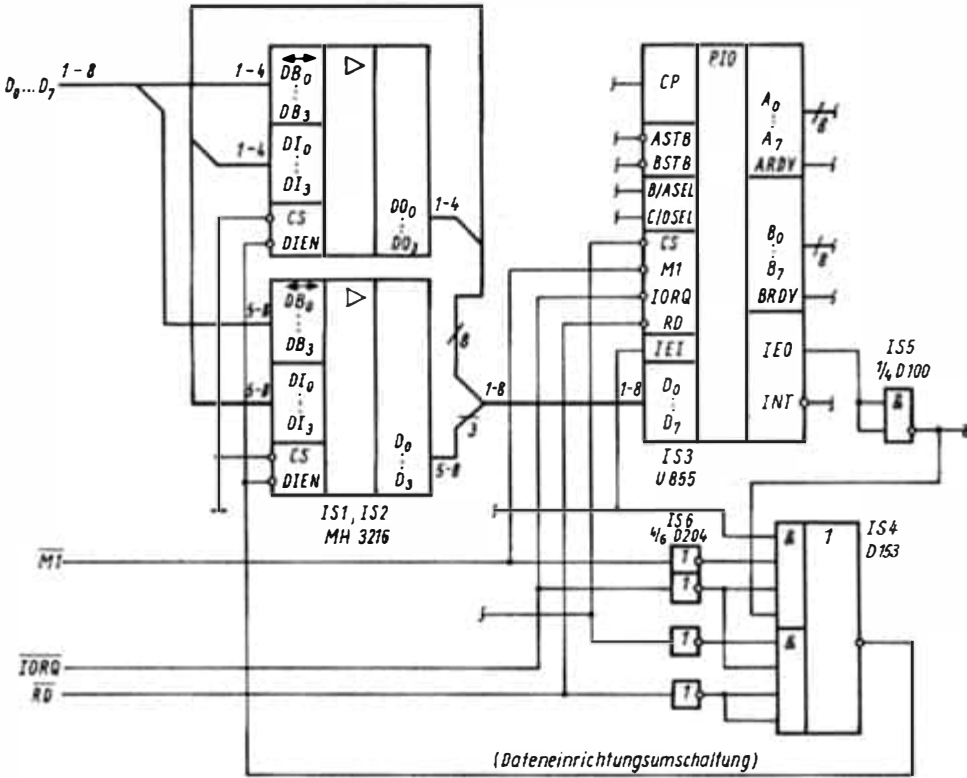


Bild 3.2.15. Datenbuspufferung mit Datenrichtungslogik

Bei der Interruptkaskadierung der PIO mit weiteren peripheren Elementen mit Hilfe der IEI-IEO-Linie wird in großen Rechnersystemen (mehr als sechs Elemente) eine Hardwarelösung benötigt, die die Verzögerung der Weitergabe des L-Pegels in der Prioritätskette kompensiert. Im Bild 3.2.16 ist eine Möglichkeit für eine derartige Logik dargestellt.

Der L-Pegel wird durch eine „vorausschauende“ Logik (look ahead) in der Kette weitergegeben. Die L-Verzögerungszeit der gesamten Kette wird durch die Verzögerung der TTL-Gatter bestimmt (s. Abschn. 4.2.).

Bild 3.2.16  
Einfache IEI-IEO-Umgehungslogik  
(look ahead logic)

3.2.7.2. Software

Die Initialisierung der parallelen Ein-/Ausgabe-Einheit U855 erfolgt üblicherweise zusammen mit den anderen peripheren Systemelementen zu Beginn des Hauptprogramms. Zu dieser Anfangsinitialisierung gehören das Laden der Interruptvektoren, das Festlegen der Betriebsarten und das Laden der Kanalsteuerregister. Nach Einschalten der Betriebsspannung nimmt die PIO automatisch die Betriebsart Eingabe ein. Somit befinden sich alle Portlinien ( $A_0 \dots A_7, B_0 \dots B_7$ ) im hochohmigen Zustand (tristate). Soll nun ein PIO-Kanal in einer Ausgabebetriebsart eingesetzt werden, kann das entsprechende Ausgaberegister vor der Betriebsartenwahl geladen werden. Nach Aktivierung der Portausgänge durch die entsprechende Modeauswahl haben diese Linien dann die vorgegebenen Pegel, und es werden Fehlreaktionen in der Peripherie vermieden.

Bei Anwendung der bidirektionalen Betriebsart muß bei der Initialisierung zuvor Kanal B in die Bitmode gesetzt werden.

Bei Anwendung des Interruptbetriebs (Mode IM2 der CPU) muß im Hauptspeicher eine Interrupttabelle aufgebaut werden. Für die PIO werden hierfür je Kanal zwei aufeinanderfolgende Speicherplätze benötigt, die durch das I-Register der CPU und den jeweiligen Interruptvektor bzw. den um 1 inkrementierten Vektor adressiert werden. Diese Speicherplätze enthalten H- und L-Byte der Startadresse der jeweiligen Kanal-ISR.

Aufgrund des Umfangs der Interruptvektoren (sieben gültige Bits) können insgesamt 128 verschiedene Startadressen für die ISR bei unverändertem I-Registerinhalt gebildet werden.

Da die PIO im nichtinitialisierten Zustand die Eingabebetriebsart einnimmt, kann zu diesem Zeitpunkt bereits durch eine entsprechende Signalkonfiguration an einem der STB-Eingänge eine Interruptanforderung anhängig sein. Der Zustand des RDY-Ausgangs hat keinen Einfluß auf die Annahme einer Interruptforderung über die STB-Linie. Eine solche anhängige Interruptforderung wird bei Setzen des Interruptfreigabeflipflops der Kanallogik an die CPU weitergeleitet. Die PIO-Interruptstruktur beinhaltet aber die Möglichkeit, daß durch Setzen des Bits  $D_4$  im Interruptsteuerwort eine derartige, nicht-quittierte Anmeldung rückgesetzt und damit unwirksam gemacht wird. Nach Setzen des Bits  $D_4$  ist zu beachten, daß das nachfolgende Steuerwort dieses Kanals als Maskierungswort interpretiert wird.

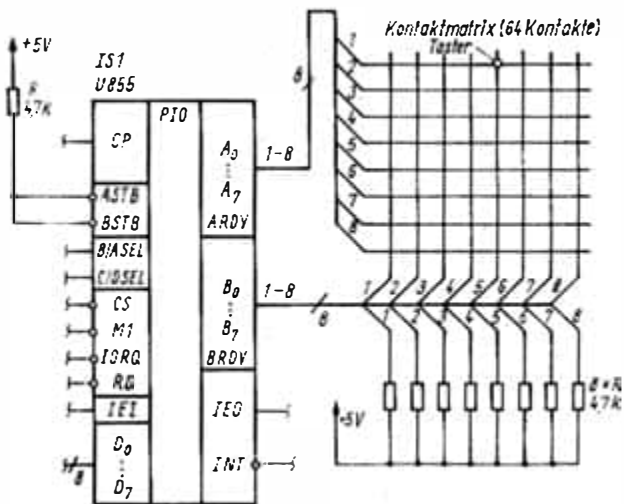


Bild 3.2.17  
Einfache Tastaturabfrageeinheit  
mit einer PIO für die Überwachung  
von 64 Tasten

### 3.2.7.3. Anwendungsmöglichkeiten

Die umfangreichen Programmiermöglichkeiten der IS U855 erlauben einen vielseitigen Einsatz zur Schaffung von digitalen Ein-/Ausgabe-Schnittstellen zwischen Mikrorechner und Peripherie. Das Vorhandensein von Quittierungslinien erleichtert den Anschluß von standardisierten Parallelschnittstellen. Aufgrund der leistungsfähigen Interruptstruktur

Tafel 3.2.3. Grenzwerte der IS U855

Bezugspotential $U_{SS} = 0 \text{ V}$				
Parameter	Symbol	min.	max.	Einheit
Betriebsspannung	$U_{CC}$	-0,5	7	V
Eingangsspannung	$U_I$	-0,5	7	V
Betriebsumgebungs- temperatur	$\theta_a$	0	70	°C
Lagerungstemperatur	$\theta_{stg}$	-55	125	°C
Gehäuseverlustleistung ( $\theta_a' = 25^\circ\text{C}$ )	$P_V$		1,1	W

Tafel 3.2.4. Statische Betriebsbedingungen der IS U855

Parameter	Symbol	min.	max.	Einheit
Betriebsspannung	$U_{CC}$	4,75	5,25	V
Eingangsspannung Low	$U_{IL}$	-0,5	0,8	V
Eingangsspannung High	$U_{IH}$	2	$U_{CC}$	V
Takteingangsspannung Low	$U_{ILC}$	-0,5	0,45	V
Takteingangsspannung High	$U_{IHC}$	$U_{CC}-0,2$	$U_{CC}$	V

Tafel 3.2.5. Statische Kennwerte der IS U855

Parameter	Symbol	Meßbedingung	min.	max.	Einheit
Ausgangsspannung Low	$U_{OL}$	$I_O = 1,8 \text{ mA}$		0,4	V
Ausgangsspannung High	$U_{OH}$	$I_O = -0,25 \text{ mA}$	2,4		V
Stromaufnahme	$I_{CC}$			100	mA
Eingangsreststrom	$I_{LI}$	$U_I = 0 \dots 5,25 \text{ V}$		10	$\mu\text{A}$
Reststrom der Tri- stateausgänge und Interrupt im hochohmigen Zustand; Datenbus bei Eingabe	$I_{LO},$ $I_{LINT},$ $I_{LD}$			10	$\mu\text{A}$
Laststrom der Darlington- Treiber-Ausgänge (nur Kanal B)	$I_{OHD}$	$U_{OH} = 1,5 \text{ V}$ $R_{EXT} = 390 \Omega$	1,5	3,8	mA

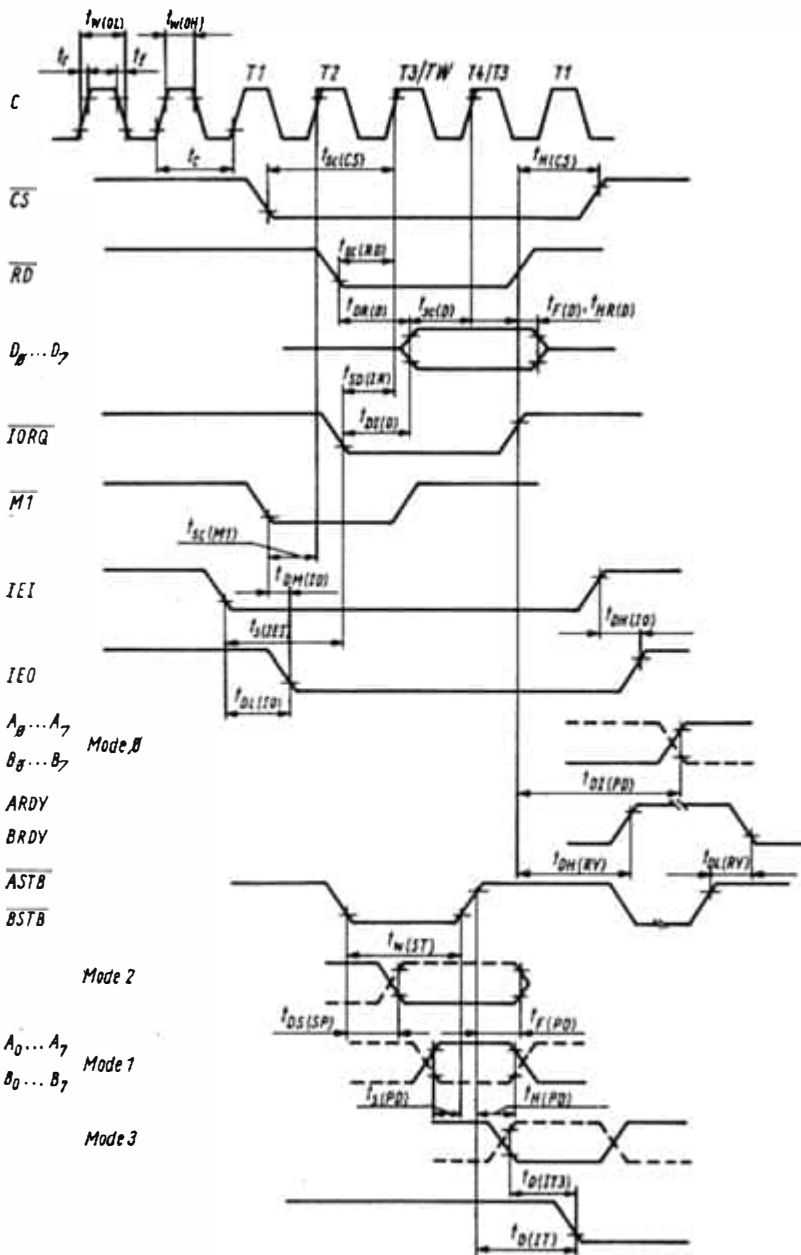


Bild 3.2.18. Darstellung der dynamischen Kennwerte der PIO im Zeitdiagramm

Tafel 3.2.6. Dynamische Kennwerte der IS U855

Dynamische Kennwerte	Symbol	min.	max.	Einheit
Taktperiode	$t_c$	400	1)	ns
H-Breite des Taktes	$t_{w(CK)}$	180	2000	ns
L-Breite des Taktes	$t_{w(CL)}$	180	2000	ns
Anstiegs- und Abfallzeiten des Taktes	$t_r, t_f$		30	ns
Alle Haltezeiten für spezifizierete Setzzeiten D0 bis D7; Datensetzzeit zu L/H des Taktes während eines Schreib- oder M1-Zyklus	$t_{H'}^i$	0		ns
A0 ... A 7, B0 ... B7; Kanaldatensetzzeit zu L/H von $\overline{STROBE}$ (Mode 1)	$t_{s(PD)}$	200		ns
$\overline{CE}, C/D; B/\overline{A}$ : Setzzeit der Steuersignale zu L/H des Taktes während eines Lese- oder Schreibzyklus	$t_{sc(CE)}$	280		ns
IEI-Setzzeit zu H/L von $\overline{IORQ}$ während eines INTA-Zyklus	$t_{s(IEI)}$	140		ns
$\overline{RD}$ -Setzzeit zu L/H von C während eines Lese- oder $\overline{M1}$ -Zyklus	$t_{sc(RD)}$	240		ns
$\overline{IORQ}$ -Setzzeit zu L/H von C während eines Lese- oder Schreibzyklus	$t_{sc(IR)}$	250		ns
$\overline{M1}$ -Setzzeit zu L/H von C während eines INTA- oder $\overline{M1}$ -Zyklus	$t_{sc(M1)}$	210		ns
ASTB, $\overline{BSTB}$ : Impulsbreite, $\overline{STROBE}$	$t_{w(ST)}$	150		ns
Impulsbreite, $\overline{STROBE}$ Mode 2	$t_{w(ST)}$	$t_{s(PD)}$		
Dauer von $\overline{M1}$ für einen RESET-Impuls	$t_{MT(RESET)}$	$2 \cdot t_c$		
Datenausgabe im Lesezyklus zur H/L-Flanke von $\overline{RD}$	$t_{DR(D)}$		440	ns
Datenausgabe während eines INTA-Zyklus zur H/L-Flanke von $\overline{IORQ}$	$t_{DI(D)}$		350	ns
Floaten des Datenbusses IEO, L/H-Flanke von IEI	$t_{F(D)}$		170	ns
IEO, H/L-Flanke von IEI	$t_{DH(IO)}$		220	ns
IEO, H/L-Flanke von IEI	$t_{DL(IO)}$		200	ns
IEO, H/L-Flanke von $\overline{M1}$	$t_{DM(IO)}$		310	ns
Kanaldatenausgang zur H/L-Flanke von $\overline{STROBE}$ in Mode 2	$t_{DS(PD)}$		240	ns
Floaten des Kanals zur L/H-Flanke von $\overline{STROBE}$ in Mode 2	$t_{F(PD)}$		210	ns
Kanaldatenstandzeit bezogen auf L/H-Flanke von $\overline{IORQ}$ während des Schreibzyklus (Mode 0)	$t_{DI(PD)}$		210	ns
INT zur L/H-Flanke von $\overline{STROBE}$	$t_{D(IT)}$		500	ns
$\overline{INT}$ bei Datenübertragung in Mode 3	$t_{D(I13)}$		660	ns
READY zur L/H-Flanke von $\overline{STROBE}$	$t_{DL(RY)}$		$t_c + 410$	ns
READY zur L/H-Flanke von $\overline{IORQ}$	$t_{DI(RY)}$		$t_c + 470$	ns

1)  $t_c = t_{w(CK)} + t_{w(CL)} + t_r + t_f$

ergibt sich neben der zyklischen Bedienung der Peripherie die sehr effektive und zeit-sparende Möglichkeit des Betriebs auf Anforderung. Als Beispiel hierfür ist im Bild 3.2.17 eine mit geringem Aufwand realisierte Tastaturabfrageeinheit für 64 Tasten dargestellt. Port A der PIO arbeitet in der Ausgabebetriebsart und führt im Ruhezustand (keine Tastenbedienung) an allen Portlinien L-Pegel. Der Kanal B arbeitet im Bitbetrieb, wobei alle Portleitungen auf Eingabe programmiert sind. Dieser Kanal dient zur Auslösung einer Interruptanforderung für den Fall, daß an einem der Eingänge ein L-Pegel auftritt. Somit werden alle Bits zur Erzeugung der Anforderung verwendet, L ist der aktive Pegel, und die Verknüpfungsfunktion ist OR. Im Fall einer beliebigen Tastenbedienung erfolgt die Anforderung eines Interrupts. In der zugehörigen ISR erfolgt nun die Abfrage des Tastenfelds nach der auslösenden Taste (den auslösenden Tasten) durch acht Ein-/Ausgabezyklen. Danach ist eine entsprechende Programmverzweigung bzw. das Herstellen des Ausgangszustands (evtl. verzögert) denkbar.

### 3.2.8. Zusammenfassung der technischen Daten

In diesem Abschnitt sollen die elektrischen Kennwerte der parallelen Ein-/Ausgabe-Einheit U855 näher spezifiziert werden. Das Bezugspotential für alle angegebenen Spannungsparameter ist  $U_{SS} = 0\text{ V}$ .

In Tafel 3.2.3 sind die wichtigsten, beim Einsatz zu beachtenden Grenzwerte aufgeführt. Tafel 3.2.4 gibt die statischen Betriebsbedingungen der IS U855 an. Tafel 3.2.5 enthält die statischen Kennwerte der PIO. Diese Kenngrößen sind für den Einsatztemperaturbereich  $\vartheta_a = 0 \dots 70^\circ\text{C}$  definiert.

In Tafel 3.2.6 sind die dynamischen Kennwerte zusammengefaßt. Sie beziehen sich auf folgende Meßbedingungen:

$$\vartheta_a = 70^\circ\text{C}$$

$$U_{CC} = 4,75\text{ V}$$

$$U_{IL} = 0,8\text{ V}$$

$$U_{IH} = 2,0\text{ V}$$

$$U_{ILC} = 0,45\text{ V (Taktingang)}$$

$$U_{IHC} = 4,55\text{ V (Taktingang)}.$$

Die zugehörige bildliche Darstellung des Zeitverhaltens ist im Bild 3.2.18 enthalten.

Tafel 3.2.7 gibt die Pindakapazitäten der IS U855 ( $\vartheta_a = 25^\circ\text{C}$ ;  $f = 0,5 \dots 2\text{ MHz}$ ) an.

Tafel 3.2.7. Pindakapazitäten der IS U855

$\vartheta_a = 25^\circ\text{C}$   
 $f = 0,5 \dots 2\text{ MHz}$

Parameter	Symbol	max.	Einheit
Taktkapazität	$C_{CP}$	14	pF
Eingangskapazität	$C_i$	7	pF
Ausgangskapazität	$C_o$	10	pF

### 3.3. Serielle Ein-/Ausgabe-Einheit

#### 3.3.1. Einführung

Für die Kommunikation mit der realen Umwelt müssen Mikrorechner mit Ansteuereinheiten zur Peripherie ausgerüstet sein. Je nach Anforderung wählt man dabei die Möglichkeiten der *parallelen* oder *seriellen* Datenübertragung. Mit der parallelen Datenübertragung erzielt man höhere Geschwindigkeiten im Datenaustausch als mit seriellen Einrichtungen. Die serielle Übertragung erfordert dagegen weniger Verbindungsleitungen. Eine Reihe peripherer Einheiten arbeitet von Natur aus sequentiell. Als typische Einheiten können Folienspeicher (Floppy-Disk), Plattenspeicher, Datenfernübertragung und die Ansteuerung von Fernschreibern genannt werden. Dafür wurden international bestimmte Übertragungsformate vereinbart. Das älteste und auch heute noch verwendete Verfahren wurde für die Fernschreibtechnik entwickelt. Es handelt sich dabei um ein asynchrones Verfahren mit Start-Stop-Betrieb. Über diesen Anwendungsfall hinausgehende Forderungen werden mit weiterentwickelten Verfahren erfüllt, so daß jetzt Peripherieschaltkreise für den seriellen Betrieb hohen Ansprüchen gerecht werden müssen. Entsprechend umfangreich ist daher die interne Logik. Die IS U856 enthält zwei Kanäle, die voneinander unabhängig arbeiten und für vollen Duplexbetrieb ausgelegt sind.

Die IS U856 hat folgende charakteristische Merkmale:

- n-Kanal-Silicon-Gate-Technologie (nSGT), Depletion-Load
- eine Versorgungsspannung 5 V
- Einphasensystemtakt (5 V)
- zwei unabhängige, voll duplex arbeitende Kanäle
- Datenübertragungsraten in synchronen oder quasisynchronen Betriebsarten bis zu 550 Kbit/s
- Datenempfangsregister vierfach gepuffert (FIFO)
- Datensenderegister zweifach gepuffert
- asynchrone Moden
  - 5, 6, 7 oder 8 bit je Charakter
  - 1, 1<sup>1</sup>/<sub>2</sub> oder 2 Stopbits
  - gerade, ungerade oder unterdrückte Parität
  - Unterbrechungserzeugung und -erkennung (break)
  - Paritäts-, Überlauf- und Formatfehlererkennung
- byteorientierte synchrone Moden
  - interne oder externe Charaktersynchronisation
  - ein oder zwei Synchronisationszeichen in getrennten Registern
  - automatisches Einfügen von Synchronisationscharakteren
  - CRC-Erzeugung und -Testung
- bitorientierte synchrone Moden
  - HDLC- oder IBM-SDLC-Prozeduren
  - automatische Nulleinfügung und -ausblendung
  - automatische Flageinfügung
  - Adreßfeldererkennung
  - Abbruchfolgeerzeugung und -erkennung
  - Schutz von gültig empfangenen Daten gegen Überschreiben
  - CRC-Erzeugung und -Testung

- Modemsteuersignallinien (je Kanal zwei Eingänge und zwei Ausgänge)
- für CRC-Behandlung sind CRC 16- und CRC-CCITT-Verfahren implementiert
- Interruptverschachtelungen durch „daisy chain“-Logik möglich
- automatische Interruptvektorerzeugung (abhängig vom auslösenden Ereignis)
- TTL-Kompatibilität aller Ein- und Ausgänge
- 40poliges DIL-Gehäuse nach TGL 26713.

Die IS U856 ist ein Peripherieschaltkreis des Mikroprozessorsystems U880. Sie ist wie die IS U855 (PIO) kompatibel zum Bus der CPU U880. Die Grundfunktion sind ein Seriell-Parallel-Konverter und ein Parallel-Seriell-Konverter, wobei die konkreten Eigenschaften weitestgehend programmierbar sind und so durch die Systemsoftware dem Anwendungsfall angepaßt werden können. Dabei können asynchrone, synchrone bitorientierte und synchrone byteorientierte Protokolle verwendet werden. Die SIO kann CRC-Kode erzeugen und überprüfen sowie die Datenrichtigkeit in verschiedenen zu vereinbarenden Moden erkennen. Der Schaltkreis besitzt zwei gleichartige Kanäle, die für Modemsteuerung entsprechende Ein-/Ausgänge besitzen. Werden diese Anschlüsse nicht für die Modemsteuerung benötigt, lassen sie sich für allgemeine Anwendung einsetzen.

**3.3.2. Struktureller Aufbau**

Einen Überblick über die interne Architektur der SIO gibt das Blockdiagramm im Bild 3.3.1. Man erkennt, daß die SIO symmetrisch aufgebaut ist und ihre beiden Kanäle über den internen Bus zusammengefaßt werden. Zu jedem dieser Kanäle zugeordnet sind Lese- und Schreibregister (RR0 ... RR2, WR0 ... WR7) sowie die entsprechende Steuerung und Statuslogik, die das Interface zu Modems und anderen externen Einheiten ermöglicht.

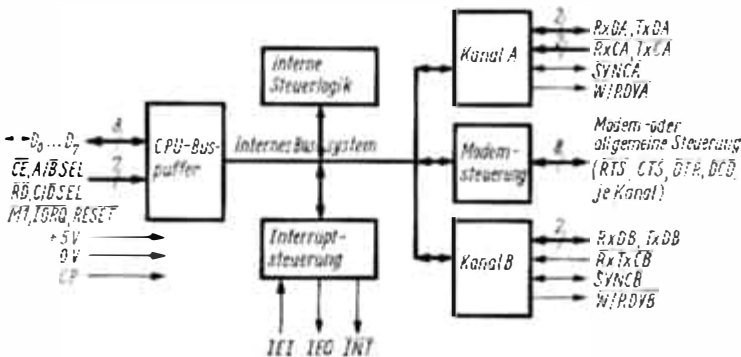


Bild 3.3.1. Blockschaltbild der SIO

Die Gruppe der Lese- und Schreibregister enthält fünf 8-bit-Steuerregister, zwei Register für den Synchronisationscharakter und zwei Statusregister. Zur Bildung des Interruptvektors wird in ein weiteres 8-bit-Register (WR2) des Kanals B der niederwertige Teil der Tabellenadresse der Interruptserviceroutine programmiert. Dieser Wert kann über das Leseregister 2 (RR2) des Kanals B gelesen werden. In beiden Kanälen sind folgende Register vorhanden:

- WR0 ... WR7 Schreibregister 0 ... 7
- RR0 ... RR2 Leseregister 0 ... 2.

Nachfolgend wird die funktionelle Zuordnung der Lese- und Schreibregister jedes Kanals angegeben:

- WR0 Registerpointer, CRC-Initialisierung, Initialisierungskommandos für die verschiedenen Moden usw.
- WR1 sendet Empfangsinterrupt und Definition der Datenübertragungsmode
- WR2 Interruptvektor (nur für Kanal B wirksam)
- WR3 Empfängerparameter und -steuerung
- WR4 verschiedene Parameter und Moden für Sender und Empfänger
- WR5 Sendeparameter und -steuerung
- WR6 Synchronisationscharakter oder SDLC-Adreßfeld
- WR7 Synchronisationscharakter oder SDLC-Flag
- RR0 Status des Sende-/Empfangs-Puffers, Interruptstatus und externer Status
- RR1 spezieller Status der Empfangsbedingungen
- RR2 modifizierter Interruptvektor (nur für Kanal B wirksam).

Die Logik für beide Kanäle beeinflußt das Format, die Synchronisation und die Gültigkeit der Daten, die von und zu dem Kanalinterface zu übertragen sind. Die Eingänge zur Modemsteuerung „clear to send“ (CTS) und „data carrier detect“ (DCD) werden überwacht per Programm durch die diskrete Steuerlogik. Alle Modemsteuersignale können in ihrer Funktion durch Programmierung für beliebige andere Zwecke eingesetzt werden.

Für die automatische Interruptvektorerzeugung bestimmt die Interruptsteuerlogik, welcher Kanal und welche Einheit innerhalb des Kanals die höchste Priorität besitzt. Dabei ist dem Kanal A die höhere Priorität gegenüber dem Kanal B zugeordnet. In jedem Kanal sind Empfänger-, Sender- und externe Statusinterrupts in der genannten Reihenfolge priorisiert.

### 3.3.3. Erläuterung der Anschlußbelegung

Im Bild 3.3.2. ist die schematische Anschlußbelegung der SIO U856 dargestellt. Die Funktion der Pins soll nachfolgend näher beschrieben werden:

**D<sub>0</sub> ... D<sub>7</sub>** Data Bus (bidirektional, tristate)

Über den Datenbus des Systems U880 erfolgt der eigentliche Informationsaustausch zwischen der CPU und der Ein-/Ausgabe-Einheit; es werden alle Daten- und Steuerwörter übertragen.

**B/ $\bar{A}$  Sel** Port B/ $\bar{A}$  Select (Eingang, H-aktiv)

Mit diesem Pin erfolgt die Auswahl des Kanals (A oder B), mit dem der Datenaustausch erfolgen soll; L-Pegel selektiert Port A, H-Pegel Port B. Üblicherweise wird die CPU-Adresse A<sub>0</sub> für diese Funktion verwendet.

**C/ $\bar{D}$  Sel** Control/Data Select (Eingang, H-aktiv)

Dieses Signal legt fest, ob im laufenden I/O-Zyklus der CPU das Wort, das in den durch Pin B/ $\bar{A}$  Sel ausgewählten Port eingeschrieben bzw. ausgelesen wird, ein Daten- oder ein Steuerwort darstellt. Bei L-Pegel erfolgt ein Datenaustausch; in anderen Fall werden die Steuerregister des Kanals programmiert.

Für diese Funktion wird i. allg. die CPU-Adresse A<sub>1</sub> verwendet.

**CE** Chip Enable (Eingang, L-aktiv)

Mit diesem Signal (L-Pegel) erfolgt die Aktivierung der SIO und somit die Ermöglichung des Datenaustauschs mit der CPU. Das  $\overline{CE}$ -Signal wird üblicherweise in einem Adreßdeko- der aus den CPU-Adressen  $A_2 \dots A_7$  gewonnen.

**CP** Clock Pulse (Eingang, 5-V-Pegel)

Der Systemtakt dient zur internen Synchronisation der meisten zeitlichen Abläufe der IS U856.

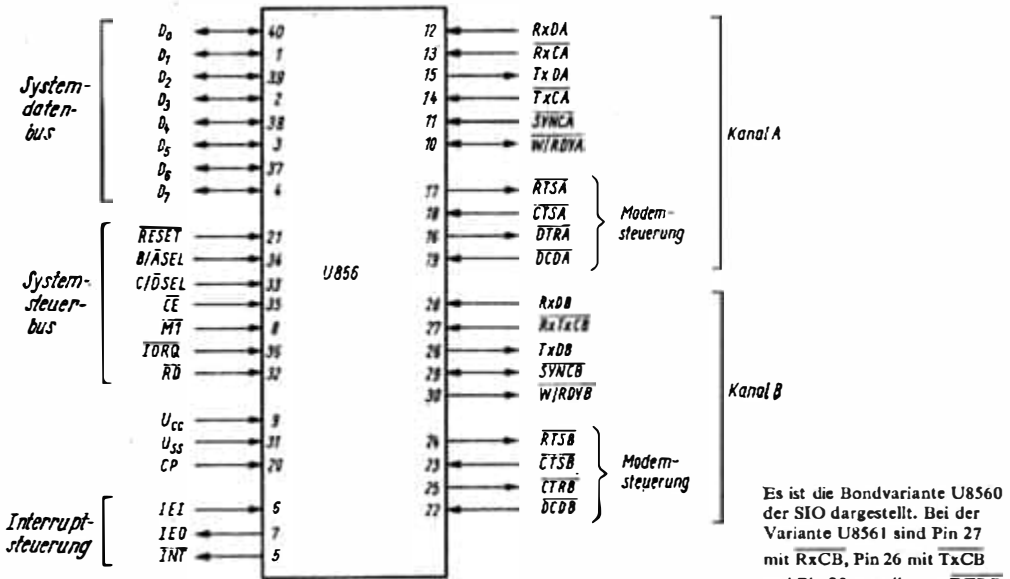


Bild 3.3.2. Schematische Anschlußbelegung der SIO

**MI** Machine Cycle 1 (Eingang, L-aktiv)

Das Signal  $\overline{MI}$  (Maschinenzyklus 1 der CPU) ist ein Steuersignal der CPU, das zur Kennzeichnung des Befehlsholezyklus (op-code fetch) dient. In Verbindung mit dem SIO-Baustein hat es die Aufgabe, die Interruptlogik zu synchronisieren.

**IORQ** In/Out Request (Eingang, L-aktiv)

Dieses Signal ist ebenfalls ein Steuersignal der CPU. Es dient in Verbindung mit den PIO-Signalen  $\overline{CE}$ ,  $C/\overline{D}$  Sel,  $B/\overline{A}$  Sel und  $\overline{RD}$  zur Kennzeichnung des Datenverkehrs zwischen CPU und SIO.

Nachfolgend dargestellte Besonderheit ist bei diesem Signal im Zusammenwirken mit der Systemperipherie (U856) zu beachten. Ein L-Pegel an diesem Pin ( $\overline{IORQ}$ ) in Verbindung mit einem aktiven  $\overline{MI}$ -Signal bedeutet, daß die CPU die Anmeldung eines Interrupts durch einen der beiden Ports quittiert.

Im selben Zyklus erfolgt daraufhin die Plazierung des entsprechenden Interruptvektors auf den Datenbus.

**RD** Read Cycle (Eingang, L-aktiv)

Das  $\overline{RD}$ -Signal ist ebenfalls ein Steuersignal der CPU. Es dient zum Einschreiben von Informationen (Daten) in die CPU. In Verbindung mit den aktiven Zuständen der Signale

$\overline{CE}$ ,  $C/\overline{D}$  Sel,  $B/\overline{A}$  Sel und  $\overline{IORQ}$  steuert es den Datentransport von der SIO in Richtung CPU.

**IEI** Interrupt Enable Input (Eingang, H-aktiv)

Dieses Signal dient in Verbindung mit IEO zur Bildung einer systemweiten Interrupt-prioritätsfestlegung durch Kaskadierung aller interruptfähigen I/O-Geräte (PIO, SIO, CTC).

**IEO** Interrupt Enable Output (Ausgang, H-aktiv)

Dieses Pin führt L-Pegel, wenn die betreffende SIO oder ein höherwertigeres Gerät in der „daisy-chain“ eine ISR in Bearbeitung bzw. angemeldet hat.

**$\overline{W/RDYA}$ ,  $\overline{W/RDYB}$**  Wait/Ready A, B (Ausgang, L-aktiv)

Diese Anschlüsse sind im rückgesetzten Zustand, und wenn sie für  $\overline{WAIT}$  programmiert sind, Open-drain-Ausgänge. Im Fall der READY-Funktion dieser Ausgänge dienen Gegentaktstufen zur Realisierung der Logikpegel.

Die Readyanschlüsse dienen zur Verbindung mit einem DMA-Controller; die Waitanschlüsse synchronisieren die CPU U880 mit der SIO zur Anpassung an die Datenübertragungsrate der SIO, wobei die Refreshbedingungen dynamischer Speicher zu beachten sind.

**$\overline{CTSA}$ ,  $\overline{CTSB}$**  Clear To Send A, B (Eingang, L-aktiv)

Diese Modemeingänge können programmiert werden als automatische Freigabe. Ein L an diesen Eingängen gibt den entsprechenden Sender frei. Andernfalls können diese Eingänge als Allzweckeingänge programmiert werden. Die Eingänge sind mit Schmitt-Trigger ausgerüstet, um Signale mit größeren Anstiegszeiten verarbeiten zu können. Die SIO erkennt Pegelübergänge und gibt an die CPU einen Interrupt bei beiden Flanken. Ein bestimmter Störabstand der Schmitt-Trigger ist nicht spezifiziert.

**$\overline{DCDA}$ ,  $\overline{DCDB}$**  Data Carrier Detect A, B (Eingang, L-aktiv)

Diese Anschlüsse sind Modemeingänge ähnlich  $\overline{CTS}$ . Der Unterschied besteht darin, daß sie zur Freigabe der Empfänger verwendet werden können.

**RxDA, RxDB** Receive Data A, B (Eingang, H-aktiv)

Dateneingänge für die serielle Datenübertragung.

**TxDA, TxDB** Transmitt Data A, B (Ausgang, H-aktiv)

Datenausgänge für die serielle Datenübertragung.

**$\overline{RxCA}$ ,  $\overline{RxCB}$**  Receiver Clock A, B (Eingang, L-aktiv)

Per Programm kann bei der asynchronen Betriebsart bestimmt werden, ob hier die Taktfrequenz 1-, 16-, 32- oder 64mal zur Baudrate anliegen muß. Mit der steigenden Flanke von  $\overline{RxC}$  bzw. der entsprechenden Untersetzung werden die Daten, die an RxD anliegen, in das Empfangsregister geschoben. Die Eingänge können direkt durch den CTC U857 angesteuert werden. Die Eingänge sind mit Schmitt-Trigger versehen, damit keine speziellen Forderungen an die Anstiegszeit bzw. Abfallzeit der Takte gestellt werden müssen.  $\overline{RxCB}$  und  $\overline{TxCB}$  sind parallelgeschaltet.

**$\overline{\text{TxCA}}$ ,  $\overline{\text{TxCB}}$**  Transmitter Clock A, B (Eingang, L-aktiv)

An diesen Takteingängen gelten die gleichen Bedingungen wie beim Empfängertakt. Zu beachten ist, daß für die Taktfrequenz bei den korrespondierenden Sendern und Empfängern der gleiche Multiplikator zur Baudrate programmiert sein muß.  $\overline{\text{RxCB}}$  und  $\overline{\text{TxCB}}$  sind parallelgeschaltet.

**$\overline{\text{RTSA}}$ ,  $\overline{\text{RTSB}}$**  Request To Send A, B (Ausgang, L-aktiv)

Diese Modemsteuerausgänge dienen zur Sendeanforderung. Der  $\overline{\text{RTS}}$ -Ausgang wird L, wenn das RTS-Bit gesetzt ist. Wird in der asynchronen Mode das RTS-Bit rückgesetzt, wird der  $\overline{\text{RTS}}$ -Ausgang H, nachdem das Senderegister leer ist. In der synchronen Mode folgt der  $\overline{\text{RTS}}$ -Ausgang direkt dem Zustand des RTS-Bits.

**$\overline{\text{DTRA}}$ ,  $\overline{\text{DTRB}}$**  Data Terminal Ready A, B (Ausgang, L-aktiv)

Diese Modemausgänge folgen dem Zustand des  $\overline{\text{DTR}}$ -Bits (Datenterminal bereit), wenn sie nicht für allgemeine Anwendung programmiert sind.

**$\overline{\text{SYNCA}}$ ,  $\overline{\text{SYNCB}}$**  Synchronisation A, B (bidirektional, L-aktiv)

Diese Anschlüsse können entweder als Eingänge oder als Ausgänge wirken. In der asynchronen Empfängermoden sind sie Eingänge ähnlich  $\overline{\text{CTS}}$  und  $\overline{\text{DCD}}$ . In dieser Mode beeinflussen die Übergänge der Logikpegel den Zustand des Synchronisations-Verfolgungs-Statusbits des Empfängerregisters RR0. Auch in der Mode mit externer Synchronisation wirken diese Anschlüsse als Eingänge. Dabei muß  $\overline{\text{SYNC}}$  mit der zweiten steigenden Flanke von  $\overline{\text{RxC}}$  auf L geschaltet werden, nachdem das letzte Bit des Synchronisationscharakters mit steigender Flanke von  $\overline{\text{RxC}}$  empfangen wurde. Das bedeutet, daß die externe Logik zwei komplette Empfängertakte nach Erkennen des Synchronisationsmusters warten muß, bevor der  $\overline{\text{SYNC}}$ -Eingang aktiv angesteuert wird. Es ist sinnvoll, den  $\overline{\text{SYNC}}$ -Eingang auf L zu halten, bis die CPU der externen Synchronisationslogik mitteilt, daß die Synchronisation verlorengegangen ist oder daß eine neue Nachricht übertragen werden soll. Die Charakterzusammenstellung beginnt mit der steigenden Flanke von  $\overline{\text{RxC}}$ , der unmittelbar die fallende Flanke des  $\overline{\text{SYNC}}$ -Pegels in der externen Synchronisationsmode vorausgeht.

In der internen Synchronisationsmode, Monosync oder Bisync, wirken diese Anschlüsse als Ausgänge. Der Pegel ist aktiv während des Empfängertakts, mit dem der Synchronisationscharakter erkannt wurde. Der Synchronisationszustand wird nicht gespeichert, so daß diese Ausgänge aktiv sind, wenn immer ein Synchronisationsmuster erkannt wurde, unabhängig von den Charaktergrenzen.

### 3.3.4. Beschreibung der Betriebsarten

Im Bild 3.3.3 ist die Datenverteilung bei Sender und Empfänger für jeden der beiden Kanäle gezeigt. Zusätzlich zum 8-bit-Empfängerschieberegister besitzt der Empfänger in einer FIFO-Anordnung drei 8-bit-Pufferregister, um eine 3-byte-Verzögerung zu erreichen. Durch diese Anordnung wird der CPU zusätzliche Zeit für die Einleitung einer Interruptserviceroutine bei Beginn einer Datenblockübertragung mit hoher Geschwindigkeit zur Verfügung gestellt. Das Fehler-FIFO des Empfängers speichert Paritäts- und Formatfehler sowie andere Statusinformationen für jedes der drei Bytes im Daten-FIFO des Empfängers.

Abhängig von der Mode und der Charakterlänge werden die eingegebenen Daten über einen der verschiedenen Pfade verteilt. In der asynchronen Mode werden die seriellen Daten in den 3-byte-Puffer übernommen, wenn die Charakterlänge 7 oder 8 bit beträgt, oder sie werden in das 8-bit-Schieberegister des Empfängers eingegeben, wenn die Länge 5 oder 6 bit beträgt.

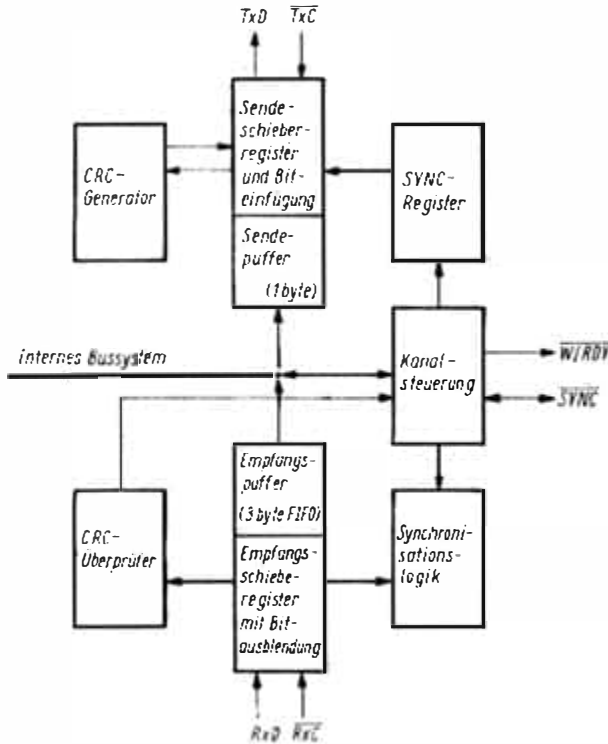


Bild 3.3.3

Blockschaltbild der Kanallogik der SIO

Abhängig von der Phase des Empfangsprozesses ist die Datenverteilung in der synchronen Mode bestimmt. Der synchrone Empfang beginnt mit dem Empfänger in der Verfolgungsphase, während der der Empfänger den einlaufenden Datenstrom nach einem Bitmuster untersucht, das übereinstimmt mit dem vorprogrammierten Synchronisationscharakter (oder den Flags in der SDLC-Mode). Falls die SIO für Monosyncverfolgung programmiert wurde, ist für die Übereinstimmung nur der einzelne Charakter, der im Schreibregister WR7 gespeichert ist, maßgebend. In der Bisyncverfolgung muß die Übereinstimmung mit den zwei Synchronisationscharakteren der Register WR6 und WR7 gegeben sein.

In jedem Fall durchlaufen die eingegebenen Daten das Synchronisationsregister des Empfängers und werden mit den vorprogrammierten Synchronisationscharakteren der Register WR6 und WR7 verglichen. Im Fall der Monosyncmode reicht die Übereinstimmung der eingegebenen Daten mit dem Inhalt des Registers WR7 zur Erreichung der Synchronisation.

In der Bisyncmode werden die eingegebenen Daten jedoch in das Schieberegister des Empfängers eingegeben, während die nächsten 8 bit in das Synchronisationsregister des Empfängers geladen werden. Die Übereinstimmung der eingelaufenen Informationen im Synchronisationsregister des Empfängers mit den vorprogrammierten Charakteren des Registers WR6 und WR7 bewirkt die Synchronisation. Wenn die Synchronisation erfolgt

ist, durchlaufen die eingegebenen Daten nicht mehr das Synchronisationsregister, sondern werden direkt dem Puffer zugeführt.

In der SDLC-Mode durchlaufen die eingegebenen Daten zuerst das Synchronisationsregister des Empfängers, das kontinuierlich den empfangenen Datenstrom überwacht und bei Bedarf die Herausnahme von Nullen bewirkt. Nach Einlauf von fünf aufeinanderfolgenden Einsen wird das sechste Bit überprüft. Wenn dieses Bit eine Null ist, wird es aus dem Datenstrom herausgenommen. Ist das sechste Bit eine Eins, so wird das siebente Bit überprüft. Ist dieses Bit eine Null, so wurde ein Flag empfangen; ist es eine Eins, stellt die empfangene Nachricht einen Abbruch dar. Die so formatierten Daten durchlaufen den Puffer und werden in das Schieberegister des Empfängers weitergeleitet. Es ist zu beachten, daß der SDLC-Empfang auch in der Verfolgungsphase beginnt, während die SIO versucht, eine Übereinstimmung der empfangenen Daten im Schieberegister mit dem Charakter des Registers WR7 zu finden. Wenn der erste Flagcharakter erkannt wurde, werden alle nachfolgenden Daten über den selben Weg gegeben, unabhängig von der Charakterlänge. Die Datenverteilung für die einzelnen Moden ist unterschiedlich, obwohl stets dieselbe Überprüfungsschaltung für CRC, sowohl für SDLC als auch die synchronen Daten benutzt wird. Wird ein Bisyncprotokoll eingesetzt, erfordert die byteorientierte Arbeitsweise, daß die CPU eingesetzt wird, um die Datencharaktere in CRC einzuschließen. Damit der CPU für diese Entscheidung Zeit zur Verfügung steht, ermöglicht die SIO eine 8-bit-Verzögerung der synchronen Daten. In der SDLC-Mode ist keine Verzögerung erforderlich, da die SIO selbst die Bytes bestimmt, von denen CRC zu berechnen ist.

Vom internen Datenbus der SIO wird das 8-bit-Datensenderegister des Senders geladen. Weiterhin besitzt der Sender ein Sendeschieberegister, das von den Registern WR6 und WR7 sowie mit den zu sendenden Daten geladen wird. Die Register WR6 und WR7 enthalten die Synchronisationscharaktere in den Moden Monosync oder Bisync oder das Adreßfeld und das Flag im Fall der SDLC-Mode.

In den Synchronmoden werden zu Beginn der Übertragung die Inhalte der Register WR6 und WR7 in das Sendeschieberegister geladen. Da in den Synchronmoden ständig Nachrichten gesendet werden, werden bei Auftritt der Bedingung, daß keine zu übertragenden Charaktere vorliegen, in die Nachricht zur Auffüllung die Inhalte der Register WR6 und WR7 eingeschoben. In der SDLC-Mode werden die Flags zu Beginn und am Ende der Übertragung in das Sendeschieberegister geladen.

Die Asynchrondaten des Sendeschieberegisters wurden mit den programmierten Start- und Stopbits versehen und werden mit der entsprechenden Geschwindigkeit sowohl zum Sendermultiplexer als auch zum CRC-Generator bei der einfachen Taktrate  $\overline{TxC}$  geschoben.

SDLC- und HDLC-Daten werden über die Einfügungslogik für Nullen hinausgeschoben, wobei diese Logik während des Sendens der Flags wirksam ist. Für alle anderen Fehler (Adreß-, Steuerungs- und Formatüberprüfung) wird eine Null nach fünf aufeinanderfolgenden Einsen in den Datenstrom eingefügt. Das Ergebnis des CRC-Generators für SDLC-Daten wird ebenfalls über die Einfügungslogik für Nullen geleitet.

### 3.3.5. Ein-/Ausgabe-Varianten

Die SIO stellt ein Bindeglied zwischen der CPU und den externen Interfaces dar. In beiden Richtungen bestehen mehrere Möglichkeiten der Arbeitsweise. Die SIO ermöglicht Polling (zyklische Abfrage), Interrupt (vektoriert und nichtvektoriert) und blockweise Über-

tragungsmoden zum Transfer von Daten, Status- und Steuerinformationen zu und von der CPU. Die blockweise Datenübertragung kann sowohl unter CPU- als auch unter DMA-Steuerung erfolgen.

Die Methode des Pollings vermeidet Programmunterbrechungen, erfordert jedoch zyklische Abfragen und bewirkt zeitliche Verzögerungen. Die Statusregister RR0 und RR1 werden zu jedem Zeitpunkt beeinflusst, indem eine Funktion ausgeführt wird. Das bedeutet, daß damit die ursprüngliche Information verlorengeht. Alle Interruptmoden der SIO müssen bei Polling gesperrt werden. Im Polling überprüft die CPU den in den Registern RR0 jeden Kanals gespeicherten Status. Die Bits  $D_0$  und  $D_2$  des Statusregisters zeigen an, daß das Senden oder das Empfangen von Daten erforderlich ist. Ebenso werden Fehler oder andere spezielle Statusbedingungen angezeigt. Es ist nicht erforderlich, daß der in dem Register RR1 enthaltene Status (spezielle Empfangsbedingungen) gelesen wird, da die Statusbits in RR1 mit dem in RR0 enthaltenen Status verbunden sind, wenn ein empfangener Charakter zur Verfügung steht.

Ebenso wie die PIO (U855) bietet die SIO eine umfangreiche Interruptanordnung, um in Echtzeitanwendungen eine schnelle Interruptreaktion zu ermöglichen. Wie schon dargestellt, enthalten die Register WR2 und RR2 des Kanals B den Interruptvektor, der in Verbindung mit dem I-Register der CPU U880 auf den Punkt des Speichers zeigt, wo der Anfang der Interruptserviceroutine angegeben ist. Da ein Interrupt durch die SIO aufgrund unterschiedlichster Bedingungen zustande kommen kann, wird dieser Vektor durch die SIO modifiziert. Damit vermeidet man, daß die CPU vor Ausführung der eigentlichen Interruptserviceroutine erst noch die Statusinformation der SIO abfragen muß. Die SIO kann acht verschiedene Interruptvektoren liefern, wenn sie dafür programmiert wurde. Das Bit  $D_2$  des Registers WR1 von Kanal B sagt aus, ob der Status den Vektor modifizieren soll. Wenn dieses Bit gesetzt wurde, wird der Interruptvektor in WR2 entsprechend der gegebenen Priorität der verschiedenen Interruptbedingungen modifiziert. Die Beschreibung des Schreibregisters WR1 (Abschn. 3.3.8.1., S. 126) zeigt hierbei die Einzelheiten der möglichen Beeinflussungen.

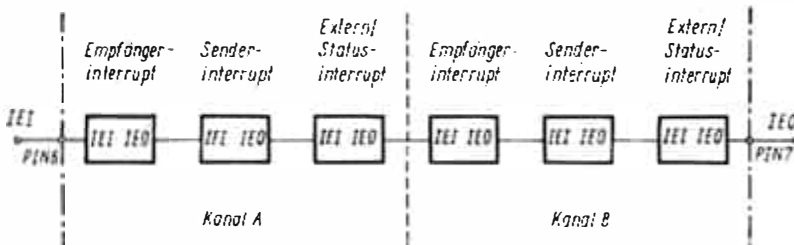


Bild 3.3.4. Interne Interruptprioritätenkette der SIO

Die Hauptursachen von Interrupts sind Empfänger-, Sender- und Extern-/Statusinterrupts (Bild 3.3.4). Jeder dieser Interrupts wird erst durch Programmierung freigegeben, wobei Kanal A die höhere Priorität besitzt und innerhalb der Kanäle die angegebene Reihenfolge der Interrupts auch die Reihenfolge der Priorität darstellt. Wenn der Senderinterrupt freigegeben wurde, wird erst dann ein Interrupt erzeugt, wenn der Senderpuffer leer wird. Das bedeutet, daß vorher ein Charakter geladen werden mußte, damit nicht schon zum Zeitpunkt der Programmierung bei leerem Sendepuffer ein Interrupt ausgelöst wird. Unter der Bedingung der Freigabe kann der Empfänger die CPU aufgrund dreier Möglichkeiten *interrupten*:

- Interrupt beim ersten empfangenen Charakter

- Interrupt bei jedem empfangenen Charakter
- Interrupt bei speziellen Empfangsbedingungen.

Der letztgenannte Interrupt kann nur auftreten, wenn eine der beiden zuerst genannten Moden freigegeben wurde. Der Interrupt beim ersten empfangenen Charakter wird üblicher Weise eingesetzt in Verbindung mit blockweiser Datenübertragung. Der Interrupt bei jedem empfangenen Charakter liefert die Möglichkeit, den Interruptvektor bei Auftreten von Paritätsfehlern zu modifizieren. Der Interrupt bei speziellen Empfangsbedingungen kann bei Auftreten von bestimmten Charakteren oder Nachrichten ausgelöst werden (z. B. Empfängerüberlauf, Formatende bei SDLC-Mode).

Der Extern-/Statusinterrupt dient hauptsächlich zur Überwachung der Signalübergänge der  $\overline{\text{CTS}}$ -,  $\overline{\text{DCD}}$ - und  $\overline{\text{SYNC}}$ -Anschlüsse. Außerdem kann dieser Interrupt bei leerem Sender oder bei der Erkennung einer Unterbrechung (asynchrone Mode) oder Abbruch (SDLC-Mode) ausgelöst werden. Der Interrupt, der bei Erkennung einer Unterbrechung bzw. eines Abbruchs bewirkt wird, kann sowohl zu Beginn als auch am Ende dieser Ursache ausgelöst werden. Das ermöglicht den ordnungsgemäßen Ablauf einer Nachrichtenübertragung, die richtige Initialisierung der nachfolgenden Nachricht und das korrekte Timing zugehöriger Einheiten.

Bei blockweiser Datenübertragung kann die Steuerfunktion dafür sowohl von der CPU als auch von einem DMA-Controller übernommen werden. Bei Blocktransfer werden die Wait-Readyausgänge in Verbindung mit den Wait-/Readybits des Schreibregisters WR1 verwendet. Der Wait-Readyausgang kann durch Software definiert werden als Readyleitung bei Steuerung durch DMA. Der Readyausgang der SIO zeigt an, daß diese für die Übertragung von Daten zum oder vom Speicher bereit ist. Demgegenüber zeigt der Waitausgang, daß die SIO für die Datenübertragung noch nicht bereit ist und die CPU daher den Ein-/Ausgabe-Zyklus zu verlängern hat. Mit der Beschreibung des Registers WR1 werden die Programmierung der Bits 5, 6 und 7 sowie der Wait-/Readyausgang genauer definiert.

### 3.3.6. Asynchrone Moden

Die Arbeitsweise der SIO ist bestimmt durch den Inhalt der Steuerregister. Diese müssen programmiert werden, bevor irgendeine Operation durch die SIO ausgeführt werden kann. Diesen Vorgang nennt man *Initialisierung*. Einige Kommandos und Moden können während des Betriebs verändert werden. Die Statusregister der SIO können jederzeit gelesen werden. Mit der Initialisierung werden die Parameter Charakterlänge, Taktrate, Anzahl von Stopbits, gerade, ungerade oder keine Parität, Interruptmode und Empfänger- oder Senderfreigabe festgelegt. Diese Parameter werden in die Schreibregister mit Hilfe des Systemprogramms geladen. Dabei muß der Inhalt des Registers WR4 vor dem Register WR1, WR3 und WR5 definiert werden.

Sollen die Daten über einen Modem oder eine serielle Schnittstelle (z. B. RS232C) gesendet werden, müssen die Ausgänge „Sende-anforderung“ ( $\overline{\text{RTS}}$ ) und „Datenterminal bereit“ ( $\overline{\text{DTR}}$ ) zusammen mit dem Freigabebit gesetzt werden. Die Sendung kann nicht vor dem Setzen des Freigabebits beginnen.

Die Möglichkeit der selbständigen Freigabe erlaubt dem Programmierer, den ersten Charakter einer Nachricht einzuschreiben, ohne auf  $\overline{\text{CTS}}$  zu warten. Wenn die selbständige Freigabe gewählt wurde, wartet die SIO darauf, daß der  $\overline{\text{CTS}}$ -Pegel auf L geht, bevor die Sendung beginnt. Benutzt man den  $\overline{\text{CTS}}$ -Eingang für einen anderen Zweck,

muß das Bit für die selbständige Freigabe auf Null gesetzt werden. Bild 3.3.5 zeigt das Format bei asynchroner Datenübertragung. In asynchroner Betriebsart werden die Register WR6 und WR7 nicht benutzt, da diese die Synchronisationscharaktere für die synchrone Mode enthalten.

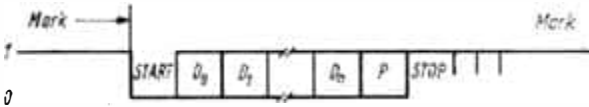


Bild 3.3.5. Datenformat bei der asynchronen Betriebsart

START	Startbit
$D_0 \dots D_n$	Datenbits ( $n = 5, 6, 7$ oder $8$ )
P	Paritätsbit (gerade oder ungerade Parität bzw. entfällt)
STOP	Stoppbit (1, $1\frac{1}{2}$ oder 2 Bits)

Die Empfängerports sind vierfach gepuffert, d. h., es sind drei Speicherregister zusätzlich zum Eingangsschieberegister vorhanden. Die Fehlerflags sind ebenfalls vierfach gepuffert und werden zur gleichen Zeit wie der Charakter geladen. Die Empfängerüberlauf- und Paritätsfehlerflags werden nicht rückgesetzt, bevor ein Fehlerrücksetzkommando ausgeführt wird. Formatende und CDC-Formatfehler zeigen den Zustand des Charakters, der gerade im Puffer vorhanden ist, wenn sie nicht rückgesetzt wurden.

### 3.3.6.1. Senden

Der Datenausgang des Senders wird auf H gehalten (Zeichen, Mark), wenn der Sender keine zu übertragenden Daten enthält. Per Programm kann dieser Ausgang durch den Sendeunterbrechungsbefehl (WR5,  $D_4$ ) auf L gehalten werden, bis dieses Kommando wieder aufgehoben wird.

Zu dem zu sendenden Charakter fügt die SIO automatisch das Startbit, das Paritätsbit und die Anzahl der programmierten Stoppbits hinzu. Wenn die Charakterlänge 6 oder 7 bit beträgt, werden die nicht benutzten Bits durch die SIO nicht berücksichtigt. Ist die Charakterlänge kleiner oder gleich 5 bit, so werden die Bits  $D_5$  und  $D_6$  des Schreibregisters rückgesetzt.

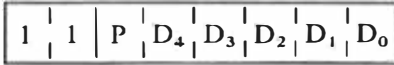
Die seriellen Daten des Anschlusses  $\overline{\text{TxD}}$  werden mit der Taktrate des Sendertakts ( $\overline{\text{TxC}}$ ) oder einem Teil von diesen ( $1/16$ ,  $1/32$  oder  $1/64$ ) geliefert. Die seriellen Daten werden mit der fallenden Flanke von  $\overline{\text{TxC}}$  hinausgeschoben. In der Mode mit Extern-/Statusinterrupt wird der Zustand der Anschlüsse  $\overline{\text{DCD}}$ ,  $\overline{\text{CTS}}$  und  $\overline{\text{SYNC}}$  während der gesamten Sendung einer Nachricht überwacht. Wenn diese Eingänge ihren Zustand für eine minimale spezifizizierte Zeit ändern, wird ein Interrupt ausgelöst.

### 3.3.6.2. Empfang

Für den asynchronen Empfang wird das Empfängerfreigabebit gesetzt. Wenn die Möglichkeit der selbständigen Freigabe gewählt wurde, muß auch  $\overline{\text{DCD}}$  L sein. Ein L (Pause, Space) am Empfängerdateneingang RxD zeigt ein Startbit an. Falls dieses L mindestens eine halbe Bitzeit anliegt, wird angenommen, daß ein gültiges Startbit vorhanden ist, und der Dateneingang wird in der Mitte eines jeden Bits abgefragt, bis der gesamte Charakter erfaßt wurde. Diese Methode der *Startbiterkennung* ermöglicht eine gute Stör- unterdrückung.

Wenn die Taktfrequenz am Eingang  $\overline{\text{RxC}}$  gleich der Baudrate ist, muß eine externe

Synchronisation erfolgen. Die empfangenen Daten werden mit der steigenden Flanke von  $\overline{\text{RxC}}$  erfaßt. Der Empfänger fügt Einsen ein, wenn eine Charakterlänge unterschiedlich von 8 bit gewählt wurde. Ist die Parität freigegeben, so wird das Paritätsbit aus dem empfangenen Charakter nicht herausgenommen. In diesem Fall setzt der Empfänger einen Charakter aus der benötigten Anzahl von Datenbits einschließlich des Paritätsbits und Einsen für die nichtbesetzten Bits zusammen. Am Beispiel des Empfangs eines 5-bit-Baudotcharakters sei das erläutert. Es ergibt sich folgendes Format:



Wenn ein Charakter empfangen wurde, wird dieser auf die folgenden Fehlerbedingungen überprüft:

- Wenn Parität freigegeben ist, wird das Paritätsfehlerbit (RR1, D<sub>4</sub>) gesetzt, wenn das Paritätsbit des empfangenen Charakters nicht mit der programmierten Parität übereinstimmt. Ist dieses Bit gesetzt, bleibt es erhalten, bis der Befehl zum Fehlerrücksetzen (WR0) gegeben wird.
- Das Formatfehlerbit (RR1, D<sub>6</sub>) wird gesetzt, wenn der Charakter ohne Stopbit zusammengesetzt ist. Im Gegensatz zum Paritätsfehlerbit wird dieses Bit nicht gespeichert und ist nur für die Zeit des empfangenen Charakters wirksam.
- Wenn die CPU keinen Charakter liest, obwohl mehr als drei Charakter empfangen wurden, wird das Empfängerüberlaufbit (RR1, D<sub>5</sub>) gesetzt.

Falls dieser Zustand auftritt, ersetzt der vierte empfangene Charakter den dritten Charakter im Empfangspuffer. Wie beim Paritätsfehler kann dieses Bit ausschließlich rückgesetzt werden durch einen Befehl der CPU zum Fehlerrücksetzen. Beide Fehler bewirken einen Interrupt mit einem Interruptvektor, der eine spezielle Empfangsbedingung anzeigt (aber nur, wenn programmiert wurde, daß der Status den Vektor zu beeinflussen hat).

Da die Folgen für Paritätsfehler und Empfängerüberlauf gespeichert werden, zeigt der gelesene Fehlerstatus einen Fehler des empfangenen Wortes im Empfangspuffer sowie jeden Paritäts- oder Überlauffehler seit dem letzten Befehl zum Fehlerrücksetzen. Um Übereinstimmung zwischen dem Zustand des Fehlerpuffers und dem Inhalt des Empfangsdatenpuffers zu halten, muß das Fehlerstatusregister vor den Daten gelesen werden. Das wird leicht dadurch erreicht, indem Vektorinterrupt eingesetzt wird, da unter diesen Bedingungen ein spezieller Interruptvektor erzeugt wird.

Wenn der Extern-/Statusinterrupt freigegeben ist, bewirkt das Erkennen einer Unterbrechung einen Interrupt, und das entsprechende Statusbit (RR0, D<sub>7</sub>) wird gesetzt.

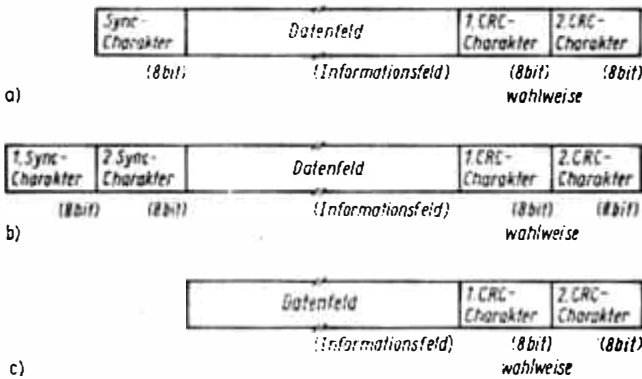
Die SIO überwacht den Empfängerdateneingang und wartet darauf, daß die Unterbrechung beendet wird, wobei dann die SIO die CPU mit dem Rücksetzen des Breakstatus zu 0 unterbricht. Die CPU muß daraufhin wieder den Befehl zum Rücksetzen des Extern-/Statusinterrupts in der Serviceroutine ausgeben, um die Erkennungslogik einer Unterbrechung erneut zu initialisieren.

Falls programmiert wurde, daß für jeden Charakter ein Interrupt erfolgen soll, ist der erzeugte Interruptvektor dann verändert, wenn ein Fehlerstatus im Register RR1 vorliegt. Beim Auftreten eines Empfängerüberlaufs wird der zuletzt empfangene Charakter in den Puffer geladen, und der vorhergehende Charakter geht verloren. Das Empfängerüberlaufbit wird gesetzt, wenn der Charakter gelesen wird, der die anderen Charaktere überschrieben hat, und ein Interruptvektor für die spezielle Empfangsbedingung wird ausgelöst, wenn freigegeben wurde, daß der Status den Vektor zu beeinflussen hat.

Im Polling muß das Bit „Empfangener Charakter verfügbar“ (Register RR0, Bit D<sub>0</sub>) überwacht werden, damit die CPU weiß, wann sie einen Charakter lesen kann. Dieses Bit wird automatisch beim Lesen der Empfangspuffer rückgesetzt. Um in der Pollingmode ein Überschreiben von Daten zu vermeiden, muß der Status des Senderpuffers überprüft werden, bevor ein neuer Charakter in den Sender eingeschrieben wird. Das Bit „Senderpuffer leer“ ist auf 1 gesetzt, wenn der Senderpuffer leer ist.

### 3.3.7. Synchrone Moden

Die IS U856 (SIO) verfügt neben der asynchronen Betriebsart über die Möglichkeit, eine synchrone serielle Datenübertragung zu organisieren. Die asynchrone Betriebsart arbeitet im Start-Stop-Betrieb, um so den Anfang eines Charakters im seriellen Datenstrom zu kennzeichnen. Bei synchroner Datenübertragung dagegen erfolgt die Synchronisation durch einen speziellen Vorgang für einen Block von Daten. Dafür sind drei Möglichkeiten implementiert: Monosync, Bisync und Extern-Sync. Bei synchroner Betriebsart wird der an  $\overline{RxC}$  und  $\overline{TxC}$  anliegende Takt nicht in der Frequenz geteilt, so daß die Baudrate gleich der Taktfrequenz ist. Der Empfänger erfaßt die Daten mit der steigenden Flanke des Empfängertakts  $\overline{RxC}$ ; der Sender gibt die Datenänderung mit der fallenden Flanke des Sendertakts  $\overline{TxC}$  aus.



**Bild 3.3.6**  
**Datenformat bei den synchronen Betriebsarten (außer HDLC/SDLC)**  
 a) Monosyncmode (8-bit-Synchronisation)  
 b) Bisyncmode (16-bit-Synchronisation)  
 c) Extern-Sync-Mode (externe Synchronisation)

Die unterschiedlichen Moden Monosync, Bisync und Extern-Sync sind durch die Art der Synchronisation des ersten Charakters gekennzeichnet. Die Mode muß bestimmt werden, weil die Register in den unterschiedlichen Moden verschieden benutzt werden. Bild 3.3.6 zeigt alle drei Formate der synchronen Moden.

#### 3.3.7.1. Monosync

Beim Empfang bewirkt die Übereinstimmung eines einzelnen Synchronisationscharakters (8-bit-Synchronisationsmode) mit dem im Register WR7 enthaltenen Charakter die Synchronisation und gibt die Datenübertragung frei.

#### 3.3.7.2. Bisync

Für die Charaktersynchronisation ist in dieser Mode die Übereinstimmung zweier aufeinander folgender Synchronisationscharaktere (16-bit-Synchronisationsmode) mit den programmierten Synchronisationscharakteren, die in den Registern WR6 und WR7 ge-

speichert sind, maßgebend. In beiden genannten Moden wird  $\overline{\text{SYNC}}$  als ein Ausgang benutzt und ist aktiv für den Teil des Empfängertakts, der den Synchronisationscharakter entdeckt.

### 3.3.7.3. Extern-Sync

In dieser Mode wird die Charaktersynchronisation extern vorgenommen.  $\overline{\text{SYNC}}$  ist ein Eingang, dem mitgeteilt wird, daß die externe Charaktersynchronisation erreicht wurde. Nachdem das Synchronisationsmuster erkannt wurde, muß die externe Logik zwei volle Empfängertaktzyklen warten, um den Synchronisationseingang zu aktivieren. Der Synchronisationseingang muß dann auf L bleiben, bis die Charaktersynchronisation verlorengegangen ist. Die Charakterzusammensetzung beginnt mit der steigenden Flanke des Empfängertakts  $\overline{\text{RxC}}$ , der der fallenden Flanke von  $\overline{\text{SYNC}}$  vorausgeht.

Nach dem Rücksetzen ist der Empfänger in jedem Fall in der Verfolgungsphase, in der die SIO die Charaktersynchronisation sucht. Die Verfolgung kann beginnen, wenn der Empfänger freigegeben ist, und entsprechend kann die Datenübertragung nun beginnen, wenn die Charaktersynchronisation erreicht wurde. Ging diese Synchronisation verloren, kann die Verfolgungsphase erneut aufgenommen werden, und zwar durch Einschreiben des erforderlichen Steuerworts mit gesetztem Bit „Nimm Verfolgungsphase auf“ (Register WR3, D<sub>4</sub>). In der Sendemode wird immer die programmierte Anzahl von Synchronisationsbits (8 oder 16) gesendet. In der Monosynchronmode wird der Inhalt des Registers WR6 gesendet, und der Empfänger vergleicht den empfangenen Charakter mit dem Inhalt des Registers WR7. In allen synchronen Moden erfolgt die Zusammensetzung der empfangenen Daten kontinuierlich, bis die SIO zurückgesetzt oder bis der Empfänger gesperrt wird (durch Befehl oder durch  $\overline{\text{DCD}}$  in der Mode mit selbsttätiger Freigabe) oder bis die CPU das Bit „Nimm Verfolgungsphase auf“ setzt.

Nachdem die Synchronisation erreicht wurde, ist die Ausführung der Monosync-, Bisync- und Extern-Sync-Betriebsarten sehr ähnlich. Im Abschn. 3.3.8.1. ist angegeben, wie die Schreibregister WR3, WR4 und WR5 in den synchronen Betriebsarten verwendet werden. WR0 bildet einen Zeiger auf die anderen Register und enthält außerdem einige Befehle; WR1 legt die Interruptmoden fest. Im Register WR2 wird der Interruptvektor gespeichert, und in den Registern WR6 und WR7 sind die Synchronisationscharaktere enthalten.

### 3.3.7.4. Senden

**Initialisierung.** Zur ordnungsgemäßen Arbeitsweise des Senders ist es erforderlich, daß dieser durch das Systemprogramm mit den folgenden Parametern initialisiert wird:

- gerade oder ungerade oder keine Parität
- x1-Taktmode
- 8-bit- oder 16-bit-Synchronisationscharakter
- CRC-Polynom
- Senderfreigabe
- Senderforderung ( $\overline{\text{RTS}}$ )
- Datenterminal bereit ( $\overline{\text{DTR}}$ )
- Interruptmoden
- Sendecharakterlänge.

Bei dieser Initialisierung muß beachtet werden, daß die Parameter des Registers WR4 spezifiziert werden, bevor die Register WR1, WR3, WR5, WR6 und WR7 mit Parametern

oder Befehlen beschrieben werden. In den synchronen Moden können zwei Polynome gewählt werden, CRC16 ( $X^{16} + X^{15} + X^2 + 1$ ) oder CRC-CCITT ( $X^{16} + X^{12} + X^5 + 1$ ). In beiden Fällen (SDLC-Moden nicht gewählt) werden der CRC-Generator und der CRC-Überprüfer rückgesetzt (alles Null). Bei der Senderinitialisierung wird der CRC-Generator durch Ausführung des Befehls „Setze Sender-CRC-Generator zurück“ (Register WR0) initialisiert. Sowohl der Sender als auch der Empfänger benutzen dasselbe Polynom. Zur Datenübertragung kann die Sendeinterruptfreigabe oder Wait/Readyfreigabe gewählt werden. Die Mode Extern-/Statusinterrupt wird benutzt, um sowohl den Status des  $\overline{\text{CTS}}$ -Eingangs als auch den Speicher für Senderunterlauf/Nachrichtenende zu überwachen. Zusätzlich besteht die Möglichkeit, mit Hilfe der selbsttätigen Freigabe den Sender freizugeben, wenn  $\overline{\text{CTS}}$  aktiv ist. Die erste Datenübertragung zur SIO kann erfolgen, wenn der Extern-/Statusinterrupt auftritt ( $\overline{\text{CTS}}$ -Statusbit gesetzt) oder, unmittelbar dem Befehl zur Senderfreigabe folgend, wenn die Mode zur selbsttätigen Freigabe gesetzt ist. Der Datenausgang des Senders wird auf Zeichen (Mark) geschaltet, und zwar nach einem Rücksetzen oder wenn der Sender nicht freigegeben ist.

Durch Programm kann eine Unterbrechung bewirkt werden, die sofort erfolgt, wenn das Bit „Sendeunterbrechung“ gesetzt ist. Wenn der Sender vollständig initialisiert und freigegeben ist, ist die Grundsituation, daß 8-bit- oder 16-bit-Synchronisationscharaktere gesendet werden.

**Datenübertragung.** Falls das Freigabebit für Sendeinterrupt (Register WR1,  $D_1$ ) gesetzt ist, wird ein Interrupt jedesmal dann erzeugt, wenn der Senderpuffer leer wird. Darauf kann reagiert werden, indem entweder ein weiterer Charakter in den Sender eingeschrieben oder der Anmeldespeicher für Sendeinterrupt durch das entsprechende Rücksetzkommando rückgesetzt wird (Register WR0, Befehl 5). Wenn der Interrupt durch diesen Befehl quittiert wird und keine weiteren Charaktere in den Sender eingeschrieben werden, kann kein weiterer Interrupt aufgrund des leeren Senderpuffers auftreten, da nur infolge des Leerwerdens ein Interrupt ausgelöst werden kann. Dieser Vorgang bewirkt einen *Senderunterlauf*.

Zur Aufrechterhaltung der Synchronisation werden in der Bisynctriebsart bei Auftreten eines Sendeunterlaufs (Sender hat keine zu sendenden Daten zur Verfügung) Füllcharaktere in den Datenstrom eingefügt. Dafür stehen der SIO zwei Möglichkeiten zur Verfügung. Sie kann Synchronisationscharaktere einfügen, oder sie sendet die zuletzt generierten CRC-Charaktere, gefolgt von Synchronisationscharakteren. Die Auswahl dafür erfolgt durch Steuerung mit dem Befehl „Setze Senderunterlauf/Nachrichtenende zurück“ (Register WR0). Da mit dem Rücksetzen des Chips oder eines Kanals das Statusbit Senderunterlauf/Nachrichtenende (Register RR0,  $D_6$ ) gesetzt ist, erfolgt das automatische Einfügen von Synchronisationscharakteren, wenn keine zu sendenden Daten zur Verfügung stehen. Für diese automatisch eingefügten Synchronisationscharaktere wird CRC nicht berechnet. Wenn die CPU das Ende einer Nachricht entdeckt, kann der Befehl zum Rücksetzen des Statusbits Senderunterlauf/Nachrichtenende ausgeführt werden. Das gestattet dem Sender, CRC zu senden, wenn keine weiteren Daten zur Verfügung stehen. Die SIO sendet in diesem Fall CRC, gefolgt von Synchronisationscharakteren, um die Nachricht abzuschließen.

Wenn ein Rücksetzen gegeben wird, nachdem der erste Charakter geladen wurde, wird das 16-bit-CRC-Wort ausgesendet und nachfolgend Synchronisationscharakter, wenn der Sender das erstmal keine Daten zu senden hat. Wegen des Senderunterlaufs wird ein Extern-/Statusinterrupt erzeugt, wenn das Bit Senderunterlauf/Nachrichtenende gesetzt wird. Im Fall der Einfügung von Synchronisationscharakteren wird ein Interrupt

erzeugt, aber nur nach den ersten eingefügten Synchronisationscharakteren. Der Status zeigt an, daß die Bits für Senderunterlauf/Nachrichtenende und leeren Sendepuffer gesetzt sind.

Im Fall der CRC-Einfügung ist das Bit Senderunterlauf/Nachrichtenende gesetzt, und das Bit „Sendepuffer leer“ ist zurückgesetzt, während CRC gesendet wird. Wenn CRC vollständig gesendet wurde, ist das Statusbit „Sendepuffer leer“ gesetzt, und der CPU wird durch Interrupt mitgeteilt, daß eine andere Nachricht beginnen kann. (Dieser Interrupt tritt auf, weil CRC gesendet und die Synchronisation geladen wurde.) Falls keine weiteren Nachrichten zu senden sind, kann das Programm die Übertragung durch Rücksetzen von  $\overline{\text{RTS}}$  und Sperrung des Senders (Registers WR5, D<sub>3</sub>) abschließen.

Sollen Füllcharaktere gesendet werden, so ist das dadurch zu erreichen, indem die CPU in die Mode für 8 bit je Sendecharakter gesetzt und 0FFH in den Sendepuffer eingegeben, während CRC gesendet wird. Eine andere Möglichkeit besteht darin, die Synchronisationscharaktere als Füllcharaktere zu bestimmen. Das soll folgendermaßen verdeutlicht werden:

Die IS U856 interruptet, wenn das Bit „Sendepuffer leer“ gesetzt ist. Durch Überwachung des internen Programmstatus erkennt die CPU, daß der letzte Charakter der Nachricht schon zur SIO gesendet wurde. Damit die SIO CRC sendet, gibt die CPU den Befehl zum Rücksetzen des Bits Senderunterlauf/Nachrichtenende (Register WR0) und bedient den Interrupt durch Rücksetzen der Anmeldung für Senderinterrupt. (Dadurch wird erreicht, daß die SIO keine weiteren Daten anfordert.) Da diese Ausführung einen Senderunterlauf bewirkt, beginnt die SIO mit der Aussendung der CRC-Information. Ebenso bewirkt die SIO einen Extern-/Statusinterrupt, wenn das Bit Senderunterlauf/Nachrichtenende gesetzt ist. Die CPU genügt diesem Interrupt, indem sie Füllcharaktere in den Sendepuffer legt und das Bit für Extern-/Statusinterrupt rücksetzt.

Durch diesen Ablauf wird erreicht, daß dem CRC anstatt der Synchronisationscharaktere die Füllcharaktere hinzugefügt werden. Dabei ist zu beachten, daß die SIO mit einem Interrupt für leeren Sendepuffer antwortet, wenn CRC vollständig gesendet ist, und daß Füllcharaktere in das Senderschieberegister geladen werden. Danach kann die CPU weitere Füllcharaktere oder Synchronisationscharaktere aussenden.

Durch Setzen des Bits für die Freigabe des Sender-CRC (Register WR5, D<sub>0</sub>) wird die Berechnung von CRC initialisiert, wenn durch das Programm der erste Charakter zur SIO gesendet wird. Es ist ratsam, vor einer Nachricht, d. h., bevor die Freigabe für Sender-CRC erfolgt, mehrere Synchronisationscharaktere zu senden, obwohl die SIO automatisch ein oder zwei Synchronisationscharaktere einfügt. Dadurch wird sichergestellt, daß der Empfänger ordnungsgemäß synchronisiert.

Damit bestimmte Charaktere von der Berechnung für CRC ausgeschlossen oder in diese Berechnung einbezogen werden, kann das Bit für die Freigabe der Sender-CRC jederzeit geändert werden. Wenn der zu sendende Charakter vom Senderdatenpuffer in das Senderschieberegister geladen wird, dann muß zu diesem Zeitpunkt der gewünschte Zustand des Bits für die Freigabe der Sender-CRC vorliegen. Damit das erreicht wird, muß die CPU das Bit für Freigabe des Sender-CRC festlegen, bevor Daten zur SIO geschickt werden. Im Bisyncprotokoll ist eine transparente Mode enthalten, die dadurch ermöglicht wird, daß das Sender-CRC jederzeit freigegeben bzw. gesperrt werden kann und weiterhin 16-bit-Synchronisationscharaktere eingefügt werden können.

Der Abschluß einer Sendung kann mit der SIO in einer besonderen Weise erfolgen; damit kann die Gültigkeit der Daten gesichert werden. Wenn der Sender während der Übertragung von Daten oder Synchronisationscharakteren gesperrt wird, wird die Sendung dieses Charakters ordnungsgemäß abgeschlossen. Nachfolgend wird der Sender-

ausgang auf Zeichen (Mark) geschaltet, anstatt daß CRC oder Synchronisationscharaktere gesendet werden. Wird der Sender gesperrt, verbleibt der letzte Charakter im Senderpuffer. Wenn die Sperrung während der Sendung von CRC erfolgt, wird die Sendung der 16-bit-Information ordnungsgemäß abgeschlossen, jedoch folgen dann Synchronisationscharaktere anstelle der CRC. Sobald eine Unterbrechung programmiert wird durch Einschreiben in das entsprechende Steuerregister, wird diese auch wirksam. Die Information im Senderpuffer und im Senderschieberegister geht verloren.

In allen Moden erfolgt zuerst die Aussendung des niederwertigsten Bits (LSB). Das erfordert, daß die zu sendenden Daten bei Datenwörtern mit weniger als 8 bit nach rechts geschoben werden, so daß das LSB an die richtige Stelle gelangt. Wenn die Wortlänge 5 bit oder weniger beträgt, muß eine entsprechende Programmierung mit Schreibregister 5 erfolgen.

Der Zustand der nicht benutzten Bits im Datenwort ist irrelevant außer in der zuletzt genannten Mode.

Wenn das Bit für die Freigabe des Extern-/Statusinterrupts gesetzt ist, erzeugen die Bedingungen

- Beginn der Sendung von CRC
- Beginn der Sendung von Synchronisationscharakteren
- Änderung der Sendebereitschaft ( $\overline{CTC}$ )

Interrupts mit zugehörigen Vektoren, wenn das Bit für die Beeinflussung des Vektors durch den Status gesetzt ist.

Alle Interrupts können gesperrt werden, wenn im Polling gearbeitet werden soll oder wenn aufgrund der Programmausführung ein Interrupt zu vermeiden ist.

### 3.3.7.5. Empfang

Der synchrone Empfang wird bezüglich seiner Varianten ähnlich durchgeführt wie das synchrone Senden. Dafür werden die Register zur Spezifikation der Mode in den meisten Fällen sowohl für Empfänger als auch zum Senden benutzt. Der Empfänger hat die Aufgabe, die richtige Synchronisation zum ausgesendeten Zeichen durchzuführen und den gesendeten Datenstrom eindeutig zur Weiterverarbeitung bereitzustellen. Unter anderem sind die Parameter Parität (gerade, ungerade), Synchronisationscharakter (8 bit, 16 bit), CRC-Polynom und Charakterlänge in der Phase der Initialisierung zu bestimmen. Die Synchronisationscharaktere sind wieder in die Schreibregister WR6 und WR7 zu laden. Es ist zu beachten, daß der Empfänger erst dann freigegeben werden darf, wenn alle erforderlichen Empfangsparameter spezifiziert sind. Dabei muß zuerst WR4 geladen werden, bevor in die Register WR1, WR3, WR5, WR6 und WR7 Informationen eingeschrieben werden.

Nach der Charaktersynchronisation werden die im Empfängerschieberegister zusammengesetzten Charaktere zum Empfängerstapelspeicher (FIFO) übertragen. Von dort können sie durch die CPU gelesen werden. Das geschieht durch Polling (zyklische Abfrage) oder in vielen Fällen durch Interrupt. Dazu stehen drei Interruptmoden zur Verfügung.

- Interrupt nur beim ersten Charakter
- Interrupt bei jedem Charakter
- Interrupt bei speziellen Empfangsbedingungen.

Zur Übertragung von Datenblöcken wird man i.allg. die erste Möglichkeit wählen, da der Zeitpunkt des Eintreffens eines neuen Charakters vorausbestimmbar ist. Treten sporadische Nachrichten auf, wählt man die zweite Möglichkeit, da damit auf das Ein-

laufen jedes einzelnen Charakters reagiert werden kann. Die dritte Möglichkeit muß man in Verbindung mit einer der erstgenannten sehen.

Zur Gewährleistung der gewünschten CRC-Berechnung muß der Zustand des entsprechenden Freigabebits eindeutig sein, bevor der Charakter in den Schieberegisterpuffer geladen wird. Die SIO berechnet CRC mit einem Zeitverzug von 8 Bitzeiten, damit der CPU genügend Zeit zur Verfügung steht, auf bestimmte übertragene Charaktere zu reagieren. Der Zeitverzug bezieht sich auf das Laden des Charakters in den Empfängerpuffer.

In den Moden Monosync, Bisync und Extern-Sync enthält das Bit für CRC- bzw. Formatfehler (Leseregister RR1, D<sub>6</sub>) das Vergleichsergebnis, jedoch um 16 Bitzeiten verzögert (8-bit-Verzögerung und 8-bit-Schiebezeit für CRC), und zwar bezogen auf die Übertragung vom Empfängerschieberegister zum Puffer. Das Ergebnis des Vergleichs sollte Null sein, um anzuzeigen, daß kein CRC- oder Übertragungsfehler vorliegt.

In der SDLC-Mode beginnt die Datenübertragung mit dem ersten Charakter, der kein Flag darstellt, nachdem mindestens ein Flag (7EH, 0111110B) empfangen wurde, wenn die Adreßsuchmode gesperrt ist. Wenn diese Mode freigegeben ist, muß einem Flag entweder die programmierte Adresse oder die allgemeine Adresse (FFH, 1111111B) folgen, bevor der Datenverkehr beginnen kann.

- Wenn Interrupts gesperrt sind, z. B. beim Polling, kann das Vorhandensein von Charakteren im Empfängerpuffer durch Überwachung des entsprechenden Bits im Leseregister RR0 erfolgen.
- Wenn CRC programmiert wurde, wird das üblicherweise zur Einleitung einer Datenblockübertragung erfolgen.
- Wenn die Länge der Informationsübertragung vor der Ausführung nicht bekannt ist, kann ein Interrupt am Nachrichtenende programmiert werden, um aus der Software-Schleife herauszukommen. Bevor ein Interrupt durch den ersten Charakter eines nachfolgenden Datenblocks erzeugt werden kann, muß durch Befehl der Interrupt beim ersten Charakter zurückgesetzt werden.
- Flags werden nicht übertragen. Die in die Übertragung zusätzlich eingefügten Nullen werden automatisch herausgelassen.
- Wenn mindestens sieben Einsen vorhanden sind, wird ein Abbruch erkannt und ein Statusinterrupt erzeugt, wenn dafür die Freigabe erfolgt ist.

Dabei wird das Bit Unterbrechung/Abbruch im Leseregister RR0 gesetzt. Nach Ausführung des Befehls „Setze Extern-/Statusinterrupt zurück“ kann ein weiterer Interrupt auftreten, wenn das andauernde Auftreten von Einsen beendet ist.

In der SDLC-Mode erfolgt die Steuerung des CRC-Überprüfers des Empfängers automatisch. Das Byte, das das Bit für Nachrichtenende enthält, umfaßt das Ergebnis der CRC-Überprüfung. Ist das Bit für CRC-Formatfehler nicht gesetzt, zeigt CRC eine gültige Nachricht an. Eine spezielle Überprüfung wird zur SDLC-Kontrolle wegen des Auftretens von nur Einsen benutzt. Das Endergebnis muß 1D0FH bzw. 0001110100001111 sein.

Die Charakterlänge kann jederzeit geändert werden. Falls die Adreß- und Steuerinformationen als 8-bit-Charaktere vereinbart sind, kann der Empfänger auf eine geringe Charakterlänge während der Zeit der Zusammensetzung des ersten Informationscharakters umgeschaltet werden. Jedoch muß diese Umschaltung so schnell erfolgen, daß sie wirksam ist, bevor die gewünschte Anzahl der Bits bereits zusammengesetzt ist.

Wurde die Adreßsuchmode nicht verwendet oder haben die Nachrichten Mehrbyteadressen, ist es nicht erforderlich, daß die CPU die gesamte Nachricht liest. Wenn erfaßt

wurde, daß die Nachricht nicht benötigt wird, bewirkt das Einschreiben des Bits „Nimm Verfolgungsphase auf“, daß der Empfang ausgesetzt wird, bis eine neue, durch ein Flag gekennzeichnete Nachricht empfangen wurde.

Wurde das Abschlußflag empfangen, wird ein Interrupt mit einem speziellen Vektor ausgelöst, wenn dieser freigegeben wurde. Dieser Interrupt zeigt an, daß das Byte mit dem Bit für „Nachrichtenende gesetzt“ empfangen wurde. Zusätzlich zum Ergebnis der CRC-Überprüfung enthält das Leseregister RRI den Ergebniskode, der zu dieser Zeit gültig ist. In den Fällen, in denen die Anzahl der Bits des Datenfelds (I-Feld) kein ganzzahliges Vielfaches der verwendeten Charakterlänge ist, zeigen diese Bits die Grenze zwischen den Bits der CRC-Überprüfung und denen des Datenfelds.

Nur wenn 5 ... 7 bit je Charakter sowie die halbduplex Mode gewählt wurden, kann eine Paritätsüberprüfung stattfinden. Da für die Festlegung der Parität für Sender und Empfänger keine getrennten Möglichkeiten vorgesehen sind, lassen sich auch nicht unterschiedliche Moden einstellen.

### 3.3.7.6. SDLC-, HDLC-Protokoll

Zwischen diesen beiden synchronen Datenübertragungsmoden bestehen nur wenige Unterschiede. Daher wird in den folgenden Ausführungen vorrangig SDLC beschrieben und auf die Besonderheiten bei HDLC hingewiesen.

Gegenüber dem synchronen Bisyncprotokoll unterscheidet sich die SDLC-Mode beträchtlich, da sie bitorientiert ist (anstatt der Charakterorientierung des Bisync). Die SIO enthält mehrere Möglichkeiten zur Verwendung unterschiedlicher Nachrichtenlängen.

Die SDLC-Nachricht wird eingeschlossen in Flags, die ähnlich sind den Synchronisationscharakteren im Bisyncprotokoll. Das Register WR0 adressiert die anderen Register und gibt verschiedene Befehle aus. Register WR1 definiert die Interruptmoden, Register WR2 enthält den Interruptvektor, Register WR6 die Prüfadresse, Register WR7 den Flagcharakter. Ein Rücksetzen des CRC-Generators bedeutet in der SDLC-Mode, daß dieser in allen Bits auf 1 gesetzt wird. Nach einem Reset wird der Senderausgang auf Mark gehalten; eine Unterbrechung kann programmiert werden, so daß der Senderausgang auf Space geht. Wenn der Sender vollständig initialisiert und freigegeben ist, werden dauernd Flags gesendet.

Durch Ausführung eines Befehls zum Abbruch der Sendung (Register WR0, Befehl 1) wird eine Abbruchfolge erzeugt. Diese besteht aus mindestens acht, höchstens 13 Einsen, die ausgesendet werden, bevor weiterhin dauernd Flags folgen. Mit der Abbruchfolge gehen alle gerade gesendeten und im Senderpuffer vorhandenen Daten verloren. Wenn erforderlich, wird automatisch eine zusätzliche Null nach fünf aufeinanderfolgenden Einsen in den Datenstrom eingefügt. Das trifft aber nicht zu für Flags und Abbruchfolgen.

Für die WAIT/READY-Funktion und Interrupts gibt es eine Reihe von Kombinationen in der SDLC-Mode. Wenn das Bit für die Freigabe von Senderinterrupts gesetzt ist, wird immer dann ein Interrupt erzeugt, wenn der Senderpuffer leer wird. Nach dem Interrupt kann der SIO die Folge Flags, 8-bit-Adreßfeld, Steuerfeld und Informationsfeld eingegeben werden. Unter Benutzung der Eigenschaft des Senderunterlaufs sendet die SIO die Formatprüffolge.

Wenn die WAIT/READY-Funktion der SIO zur zeitlichen Steuerung gewählt wurde, zeigt ein WAIT der CPU, daß die SIO für eine Datenübernahme noch nicht bereit ist und die CPU daher den I/O-Zyklus zu verlängern hat. Einem DMA-Steuerschaltkreis zeigt das READY an, daß der Senderpuffer leer ist und die SIO daher den nächsten Charakter übernehmen kann. Wenn dann das Senderschieberegister leer ist und der SIO kein neuer

Charakter angeboten wurde, geht die SIO in den Zustand Senderunterlauf. Die weitere Funktion kann dann erfolgen wie bei einem Interrupt.

SDLC-ähnliche Protokolle besitzen nicht die Vorkehrungen, um in die Nachricht Füllcharaktere einzufügen. Daher schließt die SIO ordnungsgemäß mit dem SDLC-Format ab, wenn sowohl Senderpuffer als auch Senderschieberegister leer sind. Das erfolgt dadurch, daß zwei Bytes CRC und nachfolgend mindestens ein Flag gesendet werden. Somit wird erreicht, daß die CPU nicht gezwungen ist, schnell auf ein Nachrichtenende zu reagieren, so daß sehr hohe Übertragungsgeschwindigkeiten mit DMA oder blockweiser Übertragung durch CPU-Steuerung möglich sind.

Die Verfahrensweise der SIO bezüglich Senderunterlauf hängt ab vom Zustand des Bits für Senderunterlauf/Nachrichtenende. Dieses Statusbit ist nach einem Reset gesetzt und verhindert damit, daß CRC-Charaktere eingefügt werden, wenn noch keine zu sendenden Daten zur Verfügung stehen. Daher werden nur Flagcharaktere gesendet. Sobald Daten in den Senderpuffer eingeschrieben werden, beginnt die SIO mit der eigentlichen Sendung. Zwischen dem ersten Einschreiben von Daten und dem Ende der Nachricht muß der SIO der Befehl zum Rücksetzen des Bits Senderunterlauf/Nachrichtenende gegeben sein. Das Aussenden des CRC setzt dann wieder dieses Bit.

Wenn der Extern-/Statusinterrupt gesetzt ist, CRC gesendet wird, ist das Bit Senderunterlauf/Nachrichtenende gesetzt und das Bit „Senderpuffer leer“ rückgesetzt. Damit wird kenntlich gemacht, daß das Senderregister mit CRC-Daten gefüllt ist. Ist dann die CRC-Information vollständig ausgesendet, ist das Bit „Senderpuffer leer“ gesetzt, und ein Interrupt informiert die CPU, daß eine weitere Nachrichtenübertragung beginnen kann. Das Auftreten dieses Interrupts ist bedingt durch den Abschluß der Sendung von CRC und durch das Lesen des Flags. Durch Rücksetzen von  $\overline{\text{RTS}}$  und Sperrung des Senders kann die Sendung abgeschlossen werden.

Zu Beginn jedes Blockes muß der CRC-Generator rückgesetzt werden (alles Einsen), damit die Berechnung von CRC beginnen kann. Die Berechnung beginnt dann, wenn das Programm das Adreßfeld sendet (acht Einsen). Es ist zweckmäßig, per Programm zu Beginn der Nachricht mehrere Flagcharaktere zu senden, obwohl die SIO automatisch einen solchen Charakter sendet. Damit wird eine ordnungsgemäße Charaktersynchronisation sichergestellt. Die Freigabe für Sender-CRC sollte erfolgen, bevor das Adreßfeld gesendet wird. Alle Charaktere, die von den Flags eingeschlossen werden, sind in die CRC-Berechnung einbezogen.

Jeder Charakter wird vollständig ausgesendet, auch wenn während seiner Sendung der Sender gesperrt wird. Danach wird der Sender auf Mark an seinem Ausgang geschaltet.

### 3.3.8. Programmierung

Zur Festlegung der konkreten Betriebsart der IS U856 ist es erforderlich, daß die SIO in geeigneter Weise eine Reihe von *Kommandowörtern* zur Initialisierung zugeführt bekommt. Dabei wird zuerst die grundsätzliche Betriebsart eingestellt und dann die speziellen Bedingungen der ausgewählten Mode. Bevor weitere andere Parameter eingeschrieben werden, muß das Register WR4 spezifiziert werden, da die Bedeutung der anderen Parameter davon abhängig ist.

#### 3.3.8.1. Schreibregister

Für jeden Kanal sind acht Schreibregister vorgesehen. Die SIO wird über vier Portadressen angesprochen. Die Auswahl erfolgt über die Eingänge Kanalauswahl  $B/\overline{A}$  und

Steuerung  $C/\bar{D}$  ( $B/\bar{A} = 0$  Kanal A,  $B/\bar{A} = 1$  Kanal B,  $C/\bar{D} = 0$  Daten,  $C/\bar{D} = 1$ , Steuerung/Status).

Alle Schreibregister bis auf WR0 benötigen 2 byte zur ordnungsgemäßen Programmierung. Die niederwertigen 3 bit des ersten Bytes spezifizieren als Adresse das Register, in das das folgende Steuerwort einzuschreiben ist.

Eine Ausnahme bildet Schreibregister WR0. Ein RESET, intern oder extern ausgelöst, initialisiert die SIO zum Zugriff auf Register WR0. Alle Basisbefehle und die CRC-Steuerung können über 1 byte in Verbindung mit Register WR0 festgelegt werden. Grundsätzlich erfolgt jeder Zugriff zu den Registern WR1 ... WR7 über die Adressierung mit WR0. Soll WR0 nicht modifiziert werden, ist für die Bits  $D_7 \dots D_3$  00000B einzusetzen (Nullkode). Ist nach dem Zugriff auf WR0 kein Zugriff auf ein anderes Register vorgesehen, ist als Registeradresse 000B für  $D_2 \dots D_0$  einzusetzen.

**Schreibregister WR0**



*Registeradressierung:*

0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0
0	1	1	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0

*Kanalbefehle:*

0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0
0	1	1	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0

*CRC-Befehle:*

0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0
1	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0

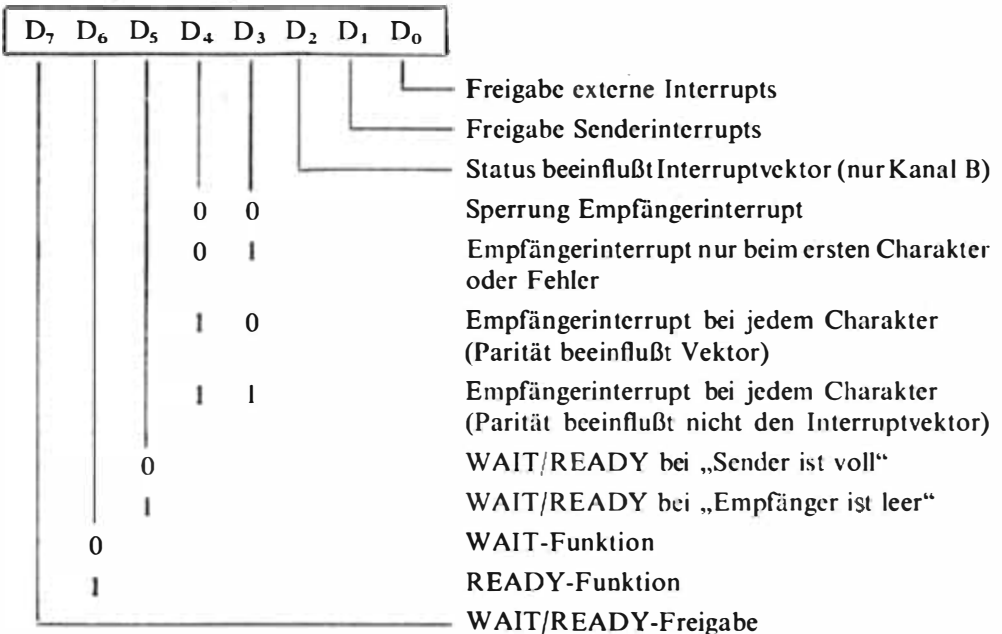
Die Bits  $D_0 \dots D_2$  bilden einen Zeiger zu dem Register, in das das folgende Byte eingeschrieben wird. Die Bits  $D_3 \dots D_5$  ergeben binär kodiert acht mögliche Kanalsteuerbefehle entsprechend den nachstehenden Erläuterungen.

Befehl 0: Dieser Befehl hat keine Auswirkung. Er wird normalerweise eingesetzt zur Verwendung des Bytes als Zeiger zu einem anderen Register.

- Befehl 1: Das Senden eines Abbruchs wird ausschließlich in der SDLC-Mode benutzt, um eine Folge von 8 ... 13 Einsen zu erzeugen.
- Befehl 2: Nach einem externen oder Statusinterrupt werden die Statusbits des Leseregisters RR0 abgespeichert. Dieser Befehl bewirkt eine erneute Freigabe und ermöglicht so das Auftreten von Interrupts.
- Befehl 3: Dieser Befehl bewirkt das gleiche wie ein externes Rücksetzen, jedoch nur für den ausgewählten Kanal. Nach Ausgabe dieses Befehls muß mindestens vier Taktzyklen gewartet werden, bevor weitere Information in den Kanal eingeschrieben wird. Nach Befehlausführung müssen alle Steuerregister des Kanals erneut beschrieben werden. Das Rücksetzen von Kanal A bewirkt außerdem ein Rücksetzen der Interruptprioritätslogik.
- Befehl 4: Damit eine neue Nachricht einen weiteren Interrupt auslösen kann, ist bei Programmierung der Mode „Interrupt nur beim ersten empfangenen Charakter“ ein Rücksetzen der Logik nach Vollendung einer Nachricht erforderlich.
- Befehl 5: Der Sender gibt einen Interrupt aus, wenn er leer wird und die Mode „Interrupt bei jedem Charakter“ gewählt wurde. Falls dann keine weiteren Charaktere zu senden sind, verhindert dieser Befehl weitere Senderinterrupts.
- Befehl 6: Paritäts- und Überlauffehler werden bis zum Rücksetzen durch diesen Befehl im Register RRI festgehalten. Daraus ergibt sich, daß Fehler in einer blockweisen Datenübertragung erst am Ende des Blockes ermittelt werden können.
- Befehl 7: Dieser Befehl, der in den Kanal A eingegeben werden muß, wird durch die SIO in gleicher Art und Weise ausgeführt wie ein RETI-Befehl auf dem Datenbus (EDH, 4DH). Dadurch kann die interne Prioritätskette auch in Systemen ohne externe Prioritätskette Verwendung finden.

Die Bits D<sub>6</sub> und D<sub>7</sub> bilden die Kodierung zum CRC-Rücksetzen. Dabei bildet die Kombination 00 einen Nullcode, der keine Auswirkungen hat und bei Verwendung als Zeiger zu anderen Registern benutzt wird.

**Schreibregister WR1**



Das Schreibregister WR1 enthält die Steuerbits der verschiedenen Moden von Interrupt und WAIT/READY.

- $D_0 = 1$ : Externe Interrupts sind freigegeben, treten auf als Funktion von Logikübergängen der Leitungen  $\overline{DCD}$ ,  $\overline{CTS}$  und  $\overline{SYNC}$  oder als Ergebnis des Beginns der Sendung von CRC- oder Synchronisationscharakteren sowie einer Unterbrechungsbedingung.
- $D_1 = 1$ : Senderinterrupts sind freigegeben und treten dann auf, wenn der Senderpuffer leer wird.
- $D_2 = 0$ : Der Status beeinflußt den Interruptvektor nicht, so daß dieser genauso ausgegeben wird, wie er eingeschrieben wurde. Das Bit  $D_2$  wird nur in das Schreibregister WR1 des Kanals B eingeschrieben. Die Mode gilt dann für beide Kanäle.
- $D_2 = 1$ : Der im Fall eines Interrupts durch die SIO ausgegebene Interruptvektor wird durch den Status folgendermaßen modifiziert:

$V_7$	$V_6$	$V_5$	$V_4$	$V_3$	$V_2$	$V_1$	$V_0$	
				0	0	0		Kanal B: Senderpuffer ist leer
				0	0	1		Kanal B: Extern-/Statusänderung
				0	1	0		Kanal B: empfangener Charakter ist verfügbar
				0	1	1		Kanal B: spezielle Empfangsbedingung
				1	0	0		Kanal A: Senderpuffer ist leer
				1	0	1		Kanal A: Extern-/Statusänderung
				1	1	0		Kanal A: empfangener Charakter ist verfügbar
				1	1	1		Kanal A: spezielle Empfangsbedingung

Die speziellen Empfangsbedingungen können sein: Paritätsfehler, Empfängerüberlauf, CRC-/Formatfehler, Formatende (SDLC-Mode).

Die Bits  $D_3$  und  $D_4$  des Registers WR1 spezifizieren die Interruptsmode des Empfängers, wie schon dargestellt.

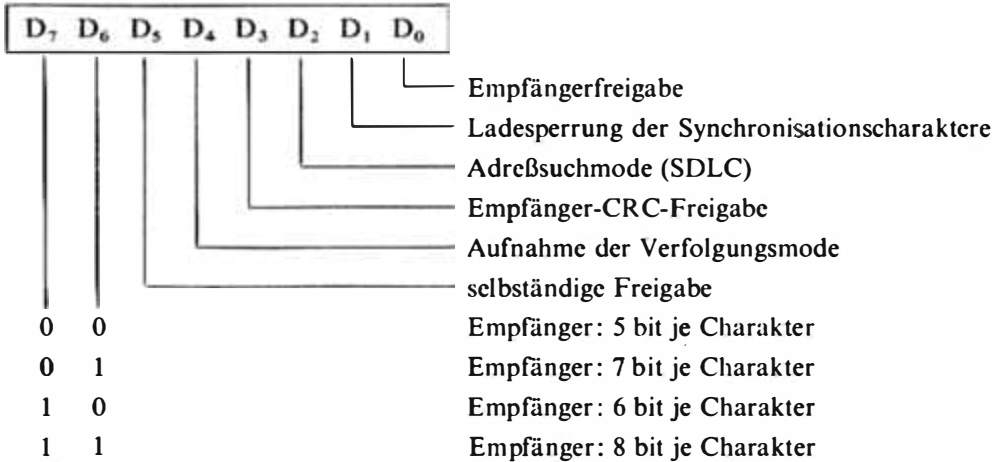
- $D_5 = 0$ : Wenn der Senderpuffer voll ist und die  $\overline{WAIT/READY}$ -Leitung freigegeben ist, wird diese aktiv.
- $D_5 = 1$ : Wenn der Empfänger leer ist und die  $\overline{WAIT/READY}$ -Leitung freigegeben ist, wird diese aktiv.
- $D_6 = 0$ : Wenn die SIO in Verbindung mit dem  $\overline{WAIT}$ -Eingang benutzt werden soll, wird mit diesem Bit die WAIT-Funktion der SIO ausgewählt.
- $D_6 = 1$ : Die Leitung  $\overline{W/RDY}$  wird in Verbindung mit einer DMA-Einheit als  $\overline{READY}$ -Leitung benutzt. Die READY-Funktion kann jederzeit auftreten, unabhängig davon, ob die SIO adressiert ist oder nicht. Diese Funktion ist aktiv H und ist abgeleitet von der positiven Flanke des Taktes.
- $D_7 = 0$ : Sperrung der Funktion WAIT/READY. Die Leitung ist H in der READY-Mode oder hochohmig in der WAIT-Mode.

### Schreibregister WR2

$D_7$	$D_6$	$D_5$	$D_4$	$D_3$	$D_2$	$D_1$	$D_0$	
$V_7$	$V_6$	$V_5$	$V_4$	$V_3$	$V_2$	$V_1$	$V_0$	Interruptvektor

Das Register WR2 ist das Vektorregister. Es ist nur im Kanal B vorhanden. Die Bits  $V_0$  und  $V_4 \dots V_7$  werden immer genauso ausgesendet, wie sie in die SIO eingeschrieben wurden. Für die Bits  $V_3 \dots V_1$  trifft das nur dann zu, wenn Bit  $D_2$  des Registers WR1 rückgesetzt ist ( $D_2 = 0$ ).

**Schreibregister WR3**



Das Register WR3 enthält Steuerbits für die Empfängerlogik.

$D_0 = 1$ : Empfängerfunktion kann beginnen.

$D_1 = 1$ : Die Synchronisationscharaktere, die einer Nachricht vorausgehen, werden nicht in den Empfängerpuffer geladen. Die CRC-Berechnung wird durch die Herausnahme der Synchronisationscharaktere nicht unterbrochen.

$D_2 = 1$ : In der SDLC-Mode bedeutet das Setzen dieses Bits, daß Nachrichten mit Adressen, die nicht mit der programmierten Adresse oder der Globaladresse (1111111 B) übereinstimmen, unterdrückt werden. Das bedeutet, daß kein Interrupt auftritt, bevor eine Adreßübereinstimmung erfolgte.

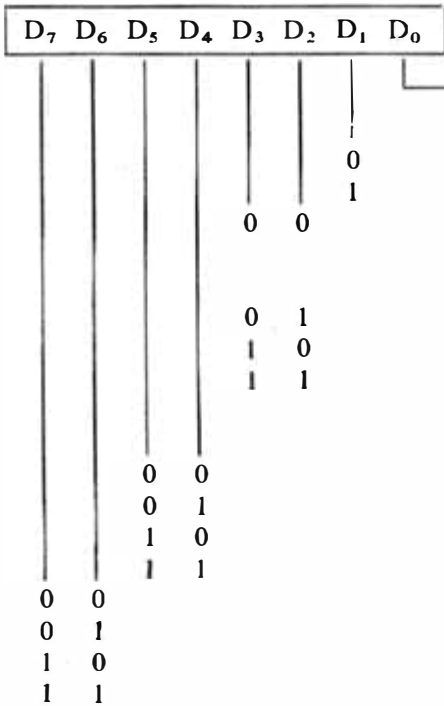
$D_3 = 1$ : Die Berechnung des CRC beginnt oder wird fortgesetzt beim Start des letzten Charakters, der vom Empfängerregister zum Puffer übertragen wurde, unabhängig von der Anzahl der Charaktere vom Puffer.

$D_4 = 1$ : Die Verfolgungsmode wird erneut aufgenommen, so daß eine verlorengangene Charaktersynchronisation erreicht werden kann.

$D_5 = 0$ : Die Eingänge  $\overline{DCD}$  und  $\overline{CTS}$  sind nur Eingänge zu den entsprechenden Bits des Leseregisters RR0.

$D_5 = 1$ : Damit werden die Signale an den Eingängen  $\overline{DCD}$  und  $\overline{CTS}$  zu den entsprechenden Freigaben für Empfänger und Sender.

Die Bits  $D_6$  und  $D_7$  bestimmen die Anzahl der im Empfänger seriell einlaufenden Bits, die zu einem Charakter zusammengefaßt werden. Solange die Anzahl der eingelaufenen Bits kleiner als die programmierte Charakterlänge ist, kann jederzeit diese Charakterlänge geändert werden.

**Schreibregister WR4**

Paritätsfreigabe (Paritätsbit wird erzeugt/  
erwartet)

ungerade Parität

gerade Parität

synchrone Betriebsart (s. auch D<sub>5</sub>, D<sub>4</sub>)

*Asynchrone Betriebsarten:*

1 Stopbit je Charakter

1<sup>1</sup>/<sub>2</sub> Stopbit je Charakter

2 Stopbit je Charakter

*Synchrone Betriebsarten:*

Monosyncmode

Bisyncmode

SDLC-Mode

Extern-Sync-Mode

Taktmode x1

Taktmode x16

Taktmode x32

Taktmode x64

D<sub>0</sub> = 1: Ein zusätzliches Bit zu den spezifizierten Bits je Charakter enthält die gewählte Parität.

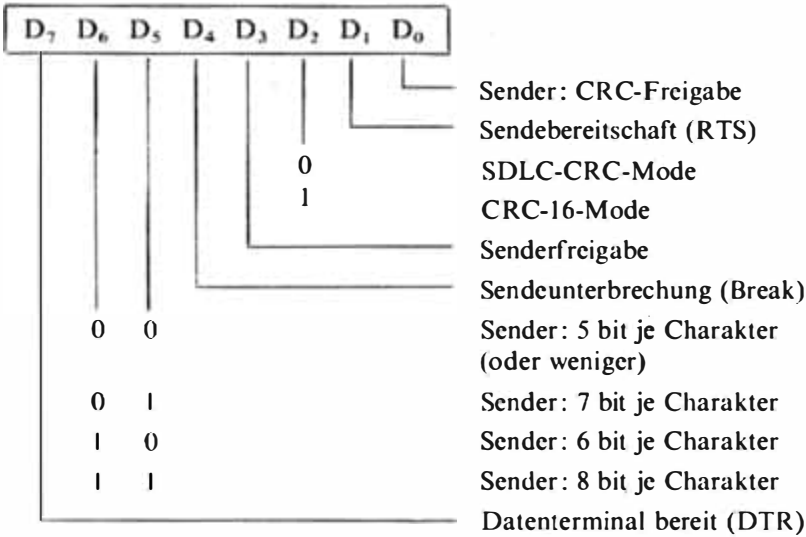
D<sub>1</sub> = 0: Es wird ungerade Parität gesendet und empfangene Charaktere daraufhin überprüft.

D<sub>1</sub> = 1: Es wird gerade Parität gesendet und empfangene Charaktere daraufhin überprüft.

Die Bits D<sub>2</sub> und D<sub>3</sub> bestimmen die Anzahl der Stopbits, die zu jedem gesendeten asynchronen Charakter hinzugefügt werden. Der Empfänger überprüft grundsätzlich nur, ob mindestens ein Stopbit vorhanden ist.

Die Bits D<sub>6</sub> und D<sub>7</sub> spezifizieren den Multiplikator der Datenrate zur Taktrate. Bei synchronen Moden müssen Datenrate und Taktrate übereinstimmen (x1); bei asynchronen Moden ist eine beliebige Auswahl möglich. Es ist zu beachten, daß der gleiche Multiplikator sowohl für den Sender als auch für den Empfänger gilt. In allen Moden darf die Datenrate höchstens Systemtaktfrequenz dividiert durch 4,5 betragen. In der Mode x1 muß die Bitsynchronisation extern erfolgen. Es können nicht 1<sup>1</sup>/<sub>2</sub> Stopbits programmiert werden.

**Schreibregister WR5**



D<sub>0</sub> = 1: Damit wird CRC vom gerade ausgesendeten Charakter berechnet. CRC wird dann gesendet, wenn dieses Bit gesetzt ist und Senderunterlauf herrscht.

D<sub>1</sub> = 0: RTS-Bit (Sendeaufforderung) zurückgesetzt.

Damit wird das Pin  $\overline{\text{RTS}}$  zu H, jedoch in der asynchronen Betriebsart erst, nachdem alle Bits des Charakters gesendet wurden und der Senderpuffer leer ist. In der synchronen Betriebsart entspricht der Logikpegel des Anschlusses  $\overline{\text{RTS}}$  dem Zustand des Bits.

D<sub>1</sub> = 1: RTS-Bit gesetzt,  $\overline{\text{RTS-Anschluß}}$  wird L.

D<sub>2</sub> = 0: Auswahl des SDLC-Polynoms, das dann gleichzeitig für Sender und Empfänger gilt. Das SDLC-CRC-Polynom muß ausgewählt werden, wenn die SDLC-Mode benötigt wird. Der CRC-Generator und -Überprüfer wird in allen Bits auf Eins gesetzt. Das SDLC-Polynom ist  $X^{16} + X^{12} + X^5 + X^1$ .

D<sub>2</sub> = 1: Auswahl des CRC16-Polynoms

Wenn die SDLC-Mode nicht gewählt wurde, wird der CRC-Generator und -Überprüfer auf Null gesetzt. Das CRC16-Polynom ist  $X^{16} + X^{15} + X^2 + 1$ .

D<sub>3</sub> = 0: Wenn der Sender nach Beginn einer Sendung gesperrt wird, werden die gerade gesendeten Daten oder Synchronisationscharaktere vollständig ausgegeben. Wird der Sender während der Übertragung eines CRC-Charakters gesperrt, erfolgt die Aussendung von Synchronisationscharakteren oder Flags.

D<sub>3</sub> = 1: Senderfreigabe; bevor dieses Bit gesetzt ist, werden keine Daten gesendet, und der Datenausgang des Senders bleibt auf Zeichen (Mark) geschaltet.

D<sub>4</sub> = 1: Mit dem Setzen dieses Bits wird der Datenausgang des Senders sofort auf Pause (Space) geschaltet.

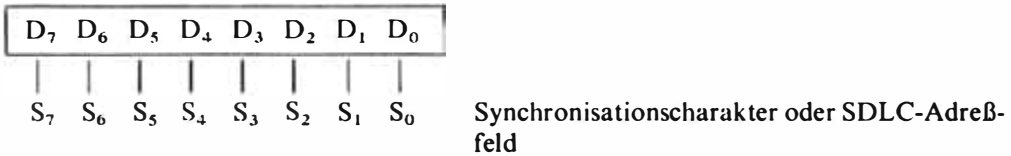
Die Bits  $D_5$  und  $D_6$  spezifizieren die Anzahl der Bits, die der Sender mit einem Charakter zu übertragen hat. Die zu sendenden Bits müssen auf den rechten Rand des Wortes bezogen sein, wobei das niederwertigste Bit ganz rechts steht. In einer Mode mit bis 5 bit je Charakter muß die CPU den Datencharakter entsprechend nachfolgender Zuordnung formatieren:

$D_7$	$D_6$	$D_5$	$D_4$	$D_3$	$D_2$	$D_1$	$D_0$	Charakterlänge
1	1	1	1	0	0	0	D	1 Datenbit (D)
1	1	1	0	0	0	D	D	2 Datenbits
1	1	0	0	0	D	D	D	3 Datenbits
1	0	0	0	D	D	D	D	4 Datenbits
0	0	0	D	D	D	D	D	5 Datenbits

$D_7=0$ : Steuerbit  $\overline{\text{DTR}}$ -Pin H (inaktiv).

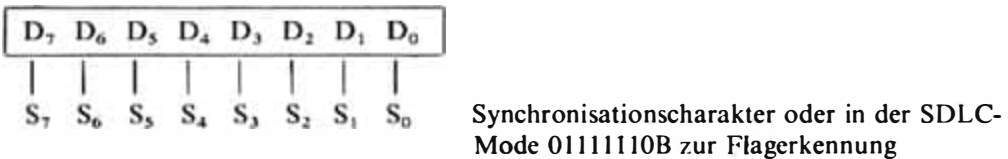
$D_7=1$ : Steuerbit  $\overline{\text{DTR}}$ -Pin L (aktiv).

### Schreibregister WR6



Der Inhalt dieses Registers ist der Synchronisationscharakter des Senders in der Monosynchronmode oder die ersten 8 bit eines 16-bit-Synchronisationscharakters in der Bisynchronmode. In der SDLC-Mode enthält das Register das Adreßfeld, das mit dem Inhalt des Adreßfelds des SDLC-Formates verglichen wird.

### Schreibregister WR7

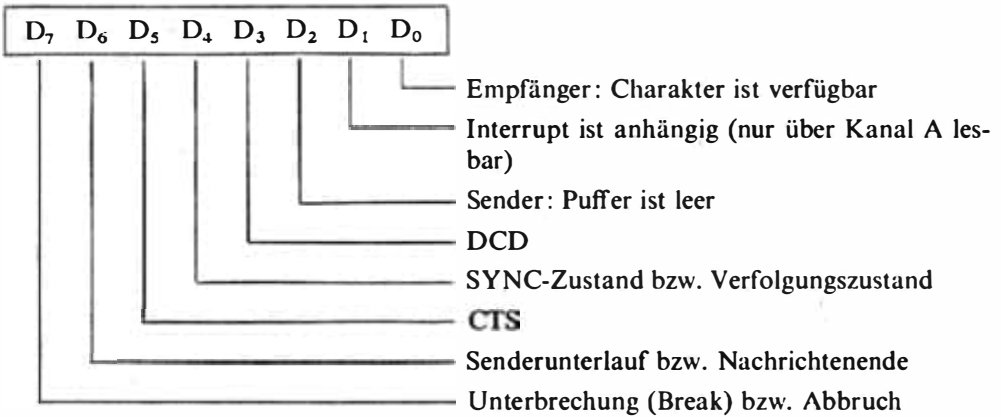


Der Inhalt dieses Registers ist der Synchronisationscharakter des Empfängers in der Monosynchronmode, die letzten 8 bit eines 16-bit-Synchronisationscharakters in der Bisynchronmode oder der Flagcharakter in der SDLC-Mode.

#### 3.3.8.2. Leseregister

Die IS U856 (SIO) enthält je Kanal zwei bzw. drei Leseregister, die Statusinformationen beinhalten. Eines dieser Register dient zum Lesen des Interruptvektors. Es wird von beiden Kanälen benutzt, ist aber dem Kanal B zugeordnet. Die Adressierung der Leseregister erfolgt gleichermaßen wie die der Schreibregister. Der einzige Unterschied besteht in der Ausführung einer Eingabeoperation durch die CPU anstelle der Ausgabe. Das Lesen von RR2 erfordert somit das Einschreiben des entsprechenden Registerpointers (Schreibregister WR0), gefolgt vom Lesevorgang der CPU für RR2.

**Leseregister RR0**



Der Inhalt dieses Registers kennzeichnet den Status der Empfänger- und Senderpuffer, die  $\overline{DCD}$ -,  $\overline{CTS}$ - und  $\overline{SYNC}$ -Eingänge, die Speicher für Senderunterlauf/Nachrichtenende und Unterbrechung/Abbruch.

- $D_0 = 1$ : Mindestens ein Charakter ist im Empfangspuffer vorhanden.
- $D_1 = 1$ : Jede Interruptbedingung beider Kanäle bewirkt das Setzen dieses Bits. Jedoch ist dieses Bit ausschließlich durch Kanal A lesbar. In Kanal B gilt immer  $D_1 = 0$ .
- $D_2 = 1$ : Dieses Bit wird beim Leerwerden des Senderpuffers gesetzt, außer wenn in einer synchronen Betriebsart ein CRC-Charakter gesendet wird. Nach dem Rücksetzen ist dieses Bit ebenfalls gesetzt.

Das Bit  $D_3$  zeigt den Zustand des  $\overline{DCD}$ -Eingangs zum Zeitpunkt der letzten Änderung eines der fünf Extern-/Statusbits. Jede Änderung des Logikpegels am  $\overline{DCD}$ -Eingang bewirkt das Auslösen eines Extern-/Statusinterrupts. Das Lesen des aktuellen Zustands des DCD-Bits muß direkt einem Befehl zum Rücksetzen des Extern-/Statusinterrupts folgen.

Die Bedeutung des Bits  $D_4$  ist von der eingestellten Betriebsart abhängig. In den asynchronen Moden wirkt das Bit ähnlich  $D_3$ , mit dem Unterschied, daß der  $\overline{SYNC}$ -Eingang überwacht wird. In den synchronen Betriebsarten ist dieses Bit rückgesetzt, wenn eine Charaktersynchronisation erreicht wurde. Es wird durch „Aufnahme der Verfolgungs-mode“ ( $D_4$  von WR3) gesetzt. Der Zustand des Bits  $D_4$  von RR0 ist hierbei unabhängig von  $\overline{SYNC}$ .

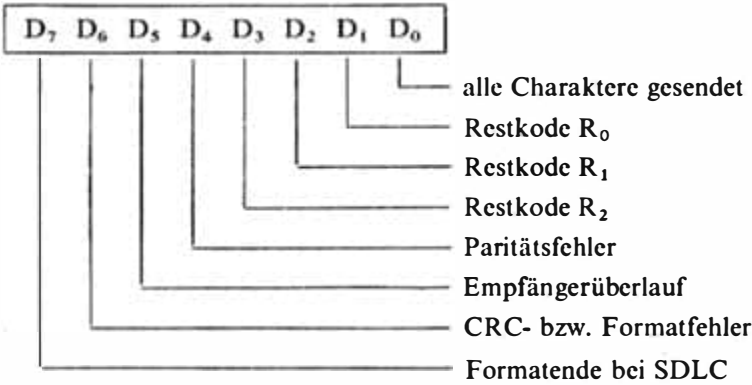
Das Bit  $D_5$  überwacht den  $\overline{CTS}$ -Eingang. Die Funktion entspricht hierbei der des Bits  $D_3$ .

Das Bit  $D_6$  zeigt mit seinem gesetzten Zustand an, daß bei der synchronen Datenübertragung erstmalig der Senderpuffer leer ist. Es werden somit nachfolgend CRC- oder Synccharakter ausgesendet. Abhängig vom aktiven Zustand dieses Bits ( $D_6 = 1$ ) erfolgt die Erzeugung von Extern-/Statusinterrupts, falls eine entsprechende Freigabe vorliegt. Wenn  $D_6 = 1$  und  $D_2 = 0$  ist, wird nachfolgend ein CRC-Charakter ausgesendet. Gilt  $D_6 = 1$  und  $D_2 = 1$ , folgt ein Synchronisationszeichen.

Das Bit  $D_7$  wird nur in den asynchronen Betriebsarten und in der SDLC-Mode verwendet. Im asynchronen Betrieb ist dieses Bit gesetzt, wenn eine Unterbrechung (Break) erkannt wurde. Bei entsprechender Freigabe werden bei gesetztem Bit wiederum Extern-/Statusinterrupts erzeugt. Nach einer erneuten Interruptfreigabe (Befehl 2 des WR0) bleibt

$D_7$  gesetzt, bis der Break aufgehoben wird. In der SDLC-Mode wird das Bit  $D_7$  analog zum Break der asynchronen Mode durch eine erkannte Abbruchfolge (sieben oder mehr Einsen) beeinflusst.

### Leseregister RR1



Das Bit  $D_0$  wird in den asynchronen Moden gesetzt, wenn alle Datenbits den Sender verlassen haben. In den synchronen Betriebsarten ist  $D_0$  immer gesetzt. Es werden jedoch keine Interrupts durch dieses Bit bewirkt.

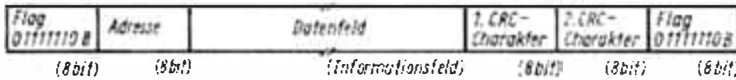


Bild 3.3.7. Datenformat bei der SDLC-Betriebsart

Die Bits  $D_3 \dots D_1$  bilden in der SDLC-Mode einen Restkode, der nach Setzen des Formatendebits ( $D_7$  von RR1), also nach dem Erkennen des Schlußflags, gültig ist. Die Bits kennzeichnen die Grenzen des Datenfelds (Informationsfelds) in den Fällen, wo die Anzahl der Datenbits kein ganzzahliges Vielfaches der verwendeten Charakterlänge ist (Bild 3.3.7). Sie dienen somit der Abgrenzung des Datenfelds vom CRC-Feld.

### Beispiel

Für einen Empfänger, der auf ein Datenformat von 8 bit je Charakter eingestellt ist, haben die Restkodes folgende Bedeutung:

$R_2$	$R_1$	$R_0$	Datenbits im vorletzten Byte	Datenbits im letzten Byte
1	0	0	3	0
0	1	0	4	0
1	1	0	5	0
0	0	1	6	0
1	0	1	7	0
0	1	1	8	0
1	1	1	8	1
0	0	0	8	2

Bei anderen Formaten kann jeweils eine ähnliche Übersicht konstruiert werden. Der Ausgangspunkt hierbei ist der Restkode, der bei Übereinstimmung der Grenze des Daten- und CRC-Feldes (also bei Ganzzahligkeit des o. g. Vielfachen) gilt:

$$\begin{matrix} R_2 & R_1 & R_0 \\ 0 & 1 & 1 \end{matrix}$$

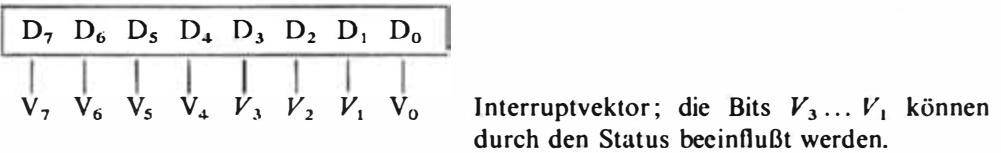
Das Bit  $D_4$  zeigt bei der Paritätsüberprüfung (durch WR4 ausgewählt) mit dem aktiven Zustand ( $D_4 = 1$ ) an, daß ein Paritätsfehler vorliegt. Dieser Zustand wird zwischengespeichert, bis ein entsprechendes Rücksetzen (Befehl 6 des WR0) erfolgt.

Das Bit  $D_5$  wird gesetzt, wenn mehr als vier Zeichen empfangen wurden, ohne daß Daten von der CPU gelesen wurden. Dieses Bit wird nur dann beeinflußt, wenn das Datenwort von der CPU gelesen wird, das zuvor überschrieben wurde. Danach bleibt das Bit zwischengespeichert, bis Befehl 6 durch WR0 ausgeführt wurde. Bei entsprechender Freigabe erfolgt die Aussendung eines Empfängerinterrupts mit dem Vektor für spezielle Empfangsbedingungen.

Das Bit  $D_6$  zeigt in den asynchronen Betriebsarten, daß ein Formatfehler erkannt wurde. Das Bit wird nicht zwischengespeichert. Das Erkennen eines Formatfehlers bewirkt das Einfügen einer halben Bitzeit, so daß dieser Fehler nicht als neues Startbit interpretiert wird. In den synchronen Moden kennzeichnet dieses Bit das Auftreten eines CRC-Fehlers.

Das Bit  $D_7$  dient in der SDLC-Mode zur Anzeige, daß ein gültiges Endflag empfangen wurde und somit die richtigen Fehler- und Restkode gelesen werden können.

**Leseregister RR2**



Dieses Register enthält exakt den in Register WR2 eingeschriebenen Interruptvektor, wenn der Status den Vektor nicht beeinflussen soll ( $D_2 = 0$  von WR1). Wenn dieses Steuerbit jedoch gesetzt ist, enthält das Register RR2 zum Zeitpunkt des Lesens genau den Vektor, den die SIO bei einem Interrupt liefern würde. Liegt in einem solchen Fall keine Interruptforderung vor, gelten  $V_3 = 0$ ,  $V_2 = 1$  und  $V_1 = 1$ . Dieses Register ist nur dem Kanal B zugeordnet und kann somit nur über diesen Kanal ausgelesen werden.

**3.3.9. Typische Anwendungen**

Die SIO U856 ist aufgrund der Fülle von Programmiermöglichkeiten in der Lage, eine Vielzahl von seriellen Datenübertragungsformaten zu bedienen. In der Praxis hat sich gezeigt, daß insbesondere die *asynchrone Datenübertragung* eine hervorragende Rolle spielt, da sehr viele Datensinken und Datenquellen über eine asynchrone Schnittstelle verfügen. Die Geschwindigkeit ist dabei meist in weiten Grenzen wählbar, so daß man sich unterschiedlichen Forderungen anpassen kann.

Die *serielle Datenübertragung* hat gegenüber paralleler Übertragung den Vorteil, daß nur sehr wenige Übertragungsleitungen benötigt werden. Geht man von den Ursprüngen der Fernschreibtechnik aus, dann stellt dieses serielle Verfahren wohl das erste Daten-

übertragungsverfahren dar, daß in größerem Umfang Anwendung gefunden hat und auch noch heute weltweit genutzt wird.

Zum Betrieb eines Rechners sind Datenein- und -ausgabegeräte erforderlich. Typische Vertreter sind

- Lochstreifenleser
- Lochstreifenstanzer
- Datensichtgerät
- alphanumerische Tastatur
- Fernschreiber
- Drucker
- unter- oder übergeordnete Rechner.

Als standardisierte Schnittstelle hat sich weitestgehend RS232C eingebürgert. Das ist eine Spannungsschnittstelle mit positivem und negativem Pegel. In der Fernschreibtechnik ist die Stromschleife mit unterschiedlichen Strömen (20, 40, 60 mA) sowie Einfach- und Doppelstrom üblich. Dadurch lassen sich mehrere Geräte, Empfänger und Sender in Serie schalten.

Tafel 3.3.1. Grenzwerte der IS U856

Kenngröße	Einheit	Kleinstwert	Größt- wert	Bemerkung
Betriebsspannung $U_{CC}$	V	-0,5	7	
Eingangsspannung $U_I$	V	-0,5	7	
Betriebstemperaturbereich $\theta_a$	°C	0 ... 70		
Lagerungstemperaturbereich $\theta_{stg}$	°C	-55 ... 125		
Verlustleistung $P$	W		1,1	bei $\theta_a = 25^\circ\text{C}$

Tafel 3.3.2. Statische Kennwerte der IS U856

Kenngröße	Kurzzeichen	Einheit	Kleinstwert	Größt- wert	Bemerkung
Betriebsspannung	$U_{CC}$	V	4,75	5,25	
Eingangsspannung	$U_{IL}$	V	-0,5	0,8	
	$U_{IH}$	V	2	$U_{CC}$	
Takteingangsspannung	$U_{ILC}$	V	-0,5	0,45	
	$U_{IHC}$	V	$U_{CC} - 0,2$	$U_{CC}$	
Ausgangsspannung	$U_{OL}$	V	-	0,4	bei $I_{OL} = 1,8 \text{ mA}$
	$U_{OH}$	V	2,4	-	bei $I_{OH} = -0,25 \text{ mA}$
Eingangsreststrom	$I_{L1}$	$\mu\text{A}$	-	10	
Reststrom des Tristateausgangs im hochohmigen Zustand	$I_{LO}$	$\mu\text{A}$	-	10	bei $U_I = 0 \text{ V}$ ... $U_{CC}$
Reststrom des Datenbusses bei Eingabe	$I_{LD}$	$\mu\text{A}$	-	10	
Taktkapazität	$C_{CP}$	pF	-	60	bei $\theta_a = 25^\circ\text{C}$
Eingangskapazität	$C_I$	pF	-	5	und $f = 0,5$ ... 2 MHz
Ausgangskapazität	$C_O$	pF	-	10	
Stromaufnahme	$I_{CC}$	mA	-	140	bei $U_{CC} = 5,25 \text{ V}$

Tafel 3.3.3. Dynamische Kennwerte der IS U856

Parameter	Symbol	Einheit	Kleinstwert	Größt- wert
Taktperiode	$t_c$	ns	400	2000
Taktimpulsweite High	$t_{w(CH)}$	ns	170	2000
Taktimpulsweite Low	$t_{w(CL)}$	ns	170	2000
Taktanstiegs- und Abfallzeiten	$t_r, t_f$	ns	0	30
Steuersignalhaltezeit von steigender C-Flanke	$t_{HCS}$	ns	0	
Steuersignal-Voreinstellzeit von steigender C-Flanke	$t_{SC(CS)}$	ns	160	
Datenausgangsverzögerung von steigender C-Flanke während Readyzyklus	$t_{DR(D)}$	ns		480
Daten-Voreinstellzeit von steigender Flanke während Write- oder M1-Zyklus	$t_{SC(D)}$	ns	50	
Datenhaltezeit von steigender C-Flanke während Write- oder M1-Zyklus	$t_{HC(D)}$	ns	0	
Datenausgangsverzögerung von fallender $\overline{IORQ}$ -Flanke während INTA-Zyklus	$t_{DI(D)}$	ns		340
Verzögerung bis zum floatenden Bus von steigender $\overline{IORQ}$ -Flanke während INTA-Zyklus	$t_{FIM(D)}$	ns		230
Verzögerung bis zum floatenden Bus von steigender $\overline{RD}$ -Flanke während INTA-Zyklus	$t_{FR(D)}$	ns		230
Verzögerung bis zum floatenden Bus von fallender IEI-Flanke während INTA-Zyklus	$t_{FI(D)}$	ns		230
IEO-Verzögerungszeit von fallender IEI-Flanke	$t_{DL(IO)}$	ns		150
IEO-Verzögerungszeit von steigender IEI-Flanke	$t_{DH(IO)}$	ns		250
IEO-Verzögerungszeit von fallender $\overline{M1}$ -Flanke (wenn Interrupt genau vor $\overline{M1}$ -Flanke erfolgte)	$t_{DC(IO)}$	ns		300
$\overline{M1}$ -Voreinstellzeit von steigender C-Flanke während READ- oder WRITE-Zyklus	$t_{SWC(M1)}$	ns	210	
$\overline{M1}$ -Voreinstellzeit von steigender C-Flanke während INTA- oder $\overline{M1}$ -Zyklus	$t_{SRC(M1)}$	ns	210	
$\overline{M1}$ -Haltezeit von steigender C-Flanke	$t_{HC(M1)}$	ns	0	
$\overline{RD}$ -Voreinstellzeit von steigender C-Flanke während Write- oder INTA-Zyklus	$t_{SWC(RD)}$	ns	240	
$\overline{RD}$ -Haltezeit von steigender C-Flanke während INTA-Zyklus	$t_{HC(RD)}$	ns	0	
$\overline{RD}$ -Voreinstellzeit von steigender C-Flanke während READ- oder $\overline{M1}$ -Zyklus	$t_{SRC(RD)}$	ns	240	

Tafel 3.3.3 (Fortsetzung)

Parameter	Symbol	Einheit	Kleinstwert	Größt-wert
RD-Haltezeit von steigender C-Flanke während Write-Zyklus	$t_{HWC(RD)}$	ns	0	
RD-Haltezeit von steigender C-Flanke während M1-Zyklus	$t_{HMC(RD)}$	ns	0	
INT-Verzögerungszeit von der Mitte des Empfangsdatenbits	$t_{DRx(IT)}$	Taktperioden	10	13
INT-Verzögerungszeit von der Mitte des Sendedatenbits	$t_{DTx(IT)}$	Taktperioden	5	9
INT-Verzögerung von der steigenden C-Flanke	$t_{DC(IT)}$	ns		200
W/R-Verzögerungszeit von IORQ oder CE im WAIT-Mode	$t_{DIC(W/R)}$	ns		180
W/R-Verzögerungszeit von der fallenden C-Flanke	$t_{DHC(W/R)}$	ns		150
WAIT/READY-High, WAIT-Mode	$t_{DHC(W/R)}$	ns		150
W/R-Verzögerungszeit von der Mitte des Empfangsdatenbits, Readymode	$t_{DRx(W/R)}$	Taktperioden	10	13
W/R-Verzögerungszeit von der Mitte des Sendedatenbits, Readymode	$t_{DTx(W/R)}$	Taktperioden	5	9
W/R-Verzögerungszeit von der steigenden C-Flanke, WAIT/READY-Low, Readymode	$t_{DLC(W/R)}$	ns		120
Minimale Impulslänge zum Einspeichern des Status im Register und Erzeugen eines Interrupts	$t_{w(RH)}$	ns	200	
Minimale Low-Impulslänge zum Einspeichern des Status im Register und Erzeugen eines Interrupts	$t_{w(PL)}$	ns	200	
Syncimpulsverzögerungszeit von der Mitte des Empfangsdatenbits, Ausgang	$t_{DL(SY)}$	Taktperioden	4	7
Syncimpulsvoreinstellzeit von der steigenden Rx-C-Flanke, Eingang (Ext.-Sync-Mode)	$t_{SL(SY)}$	ns	100	
SYNC-Impulslänge (Ext.-Sync-Mode)	$t_{w(SY)}$	Taktperioden	3	
Sendertaktperiode	$t_{C(TxC)}$	ns	400 <sup>1)</sup>	
Sendertaktimpulsweite High	$t_{w(TCH)}$	ns	180	
Sendertaktimpulsweite Low	$t_{w(TCL)}$	ns	180	
TxD-Ausgangsverzögerung von fallender Tx-C-Flanke (x1-Taktfaktor)	$t_{D(TxD)}$	ns		400
Empfängertaktperiode	$t_{D(RxC)}$	ns	400 <sup>1)</sup>	
Empfängertaktimpulsweite High	$t_{w(RCH)}$	ns	180	
Empfängertaktimpulsweite Low	$t_{w(RCL)}$	ns	180	

1) In allen Moden muß die Systemtaktfrequenz mindestens vereinfach größer als die maximale Datenrate sein.

Der Takt für die Steuerung der Datenübertragungsrate läßt sich direkt von einem CTC ableiten. Dadurch ist eine einfache Programmiermöglichkeit für die Geschwindigkeit gegeben. Für die Taktfrequenz von Mikrorechnern auf der Basis U880 hat sich daher eine Frequenz durchgesetzt, die ein duales Vielfaches der Datenübertragungsrate asynchroner Schnittstellen darstellt. Die CPU U880 ist für eine maximale Taktfrequenz von 2,5 MHz spezifiziert. Dieser Frequenz kommt am nächsten 2,4576 MHz ( $38,4 \text{ kHz} \times 64$ ), und

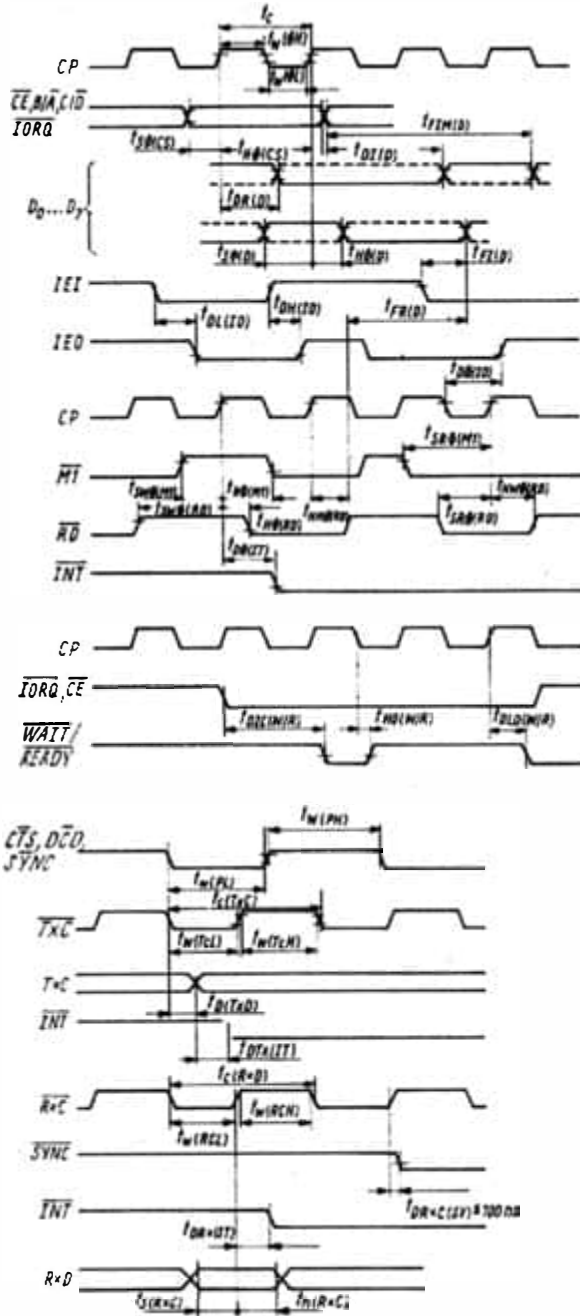


Bild 3.3.8  
Darstellung der dynamischen  
Kennwerte der SIO im Zeitdiagramm

diese Frequenz wird auch meist eingesetzt, da sich außerdem mit Hilfe eines CTC ebenfalls leicht eine Echtzeituhr ableiten läßt:

$$\begin{aligned} 2,4576 \text{ MHz} : 256 \text{ (Vorteiler CTC)} &= 9600 \text{ Hz} \\ 9600 \text{ Hz} : 96 \text{ (Kanal CTC)} &= 100 \text{ Hz (10-ms-Intervalle)} \\ 100 \text{ Hz} : 100 \text{ (Kanal CTC)} &= 1 \text{ Hz (1-s-Intervalle)}. \end{aligned}$$

Da die synchronen Moden 4,5 Takte je zu übertragendem Bit erfordern, läßt die Taktfrequenz 2,5 MHz 555 555 Bit/s zu. Mit dieser Rate lassen sich auch Floppy-Disk mit einfacher und doppelter Schreiddichte ansteuern und gleichzeitig die automatische CRC-Erzeugung und -Überprüfung ausnutzen.

### 3.3.10. Zusammenfassung der technischen Daten

An dieser Stelle sollen die elektrischen Kennwerte der seriellen Ein-/Ausgabe-Einheit U856 spezifiziert werden. Als Bezugspotential gilt für alle Spannungsparameter  $U_{SS} = 0 \text{ V}$ .

Die beim Einsatz der SIO zu beachtenden Grenzwerte sind in Tafel 3.3.1 aufgeführt. In Tafel 3.3.2 sind die statischen Kennwerte (bei 0 ... 70°C) enthalten. Tafel 3.3.3 faßt die dynamischen Kennwerte der IS U856 zusammen. Hierbei gilt für alle Verzögerungszeiten:

$$\begin{aligned} \vartheta_a &= 70^\circ\text{C} \\ U_{CC} &= 4,75 \text{ V} \\ U_{IL} &= 0,8 \text{ V} \\ U_{IH} &= 2 \text{ V} \\ U_{ILC} &= 0,45 \text{ V} \\ U_{IHC} &= 4,55 \text{ V}. \end{aligned}$$

Für alle anderen Kennwerte gilt:

$$\begin{aligned} \vartheta_a &= 0 \dots 70^\circ\text{C} \\ U_{CC} &= 4,75 \dots 5,25 \text{ V}. \end{aligned}$$

Die bildliche Darstellung der dynamischen Kennwerte zeigt Bild 3.3.8.

## 3.4. Zähler/Zeitgeber

### 3.4.1. Einführung

Die IS U857 ist ein Zähler/Zeitgeber-Schaltkreis (CTC), die bezüglich Interruptverhalten und Leistungsfähigkeit auf das System U880 zugeschnitten ist. Sie umfaßt vier vollständige Zähler/Zeitgeber-Kanäle mit einem Zählumfang von jeweils 8 bit sowie 8-bit-Vorteilern, die in der Zeitgeberbetriebsart zum Herunterteilen des Systemtakts dienen. Die Einheit ist in ihren Eigenschaften in weiten Grenzen programmierbar (Zeitkonstanten, Vorteiler, Triggerflanken, Interruptverhalten usw.).

Die charakteristischen Merkmale der IS sind

- n-Kanal-Silicon-Gate-Technologie, Depletion-Load
- eine Versorgungsspannung von 5 V
- Einphasensystemtakt (5 V)
- vier unabhängige 8-bit-Kanäle mit Eingängen für externe Takt/Triggerflanke und Ausgängen für Nulldurchgang/Zeitgeberausgang (programmierbar)
- zwei Betriebsarten
  - Zählermode (8-bit-Zähler, Zählvorgang mit maximal halber Systemtaktfrequenz)
  - Zeitgebermode (8-bit-Rückwärtszähler mit rückladbarem Zeitkonstantenregister und programmierbaren 8-bit-Vorteilern)
- Interruptfähigkeit bei Nulldurchgang der Zähler ermöglicht schnelle und leistungsfähige Reaktion der CPU
- Interruptkaskadierung durch „daisy-chain“-Logik
- automatische Interruptvektorerzeugung
- TTL-Kompatibilität aller Ein- und Ausgänge
- 28poliges DIL-Gehäuse nach TGL 26 713.

Typische Anwendungsfälle für den CTC sind Zeitgeber für Anzeigeroutinen und Echtzeituhren, Ereigniszähler, Frequenzzähler und integrierende DA-Wandler. Die IS ist hierbei aufgrund der Vielzahl der programmierbaren Eigenschaften sehr anpassungsfähig. Zur Realisierung von Zählerumfängen größer als 8 bit können die CTC-Kanäle mittels ihrer Ein- und Ausgänge zur Erreichung des notwendigen Zählumfangs kaskadiert werden. Die Zusammenschaltung der IS U857 mit der CPU U880 erfordert keine weiteren Zusatzbauelemente. Ausgenommen hiervon sind Standardbauelemente in großen Mikrorechnersystemen, die zur Bustreibung, Adreßdekodierung und Kaskadierung dienen.

### 3.4.2. Struktureller Aufbau

Das Blockschaltbild vom Zähler/Zeitgeber ist im Bild 3.4.1 dargestellt. Der CTC besteht aus einem CPU-Buspuffer, einer internen Steuerlogik, einer Interruptsteuerlogik und der Logik der vier unabhängigen Kanäle.

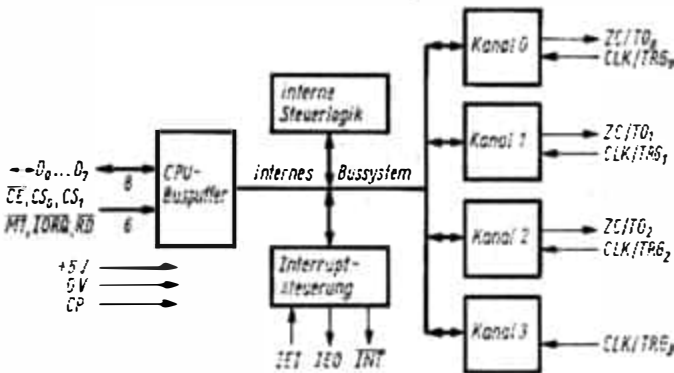


Bild 3.4.1  
Blockschaltbild des CTC

Der CPU-Buspuffer dient zur Anpassung der Daten- und Steuerinformationen an das Standardzeitverhalten des Systems U880 und ermöglicht eine direkte Zusammenschaltung des CTC mit der CPU U880.

Die interne Steuereinheit überwacht in Abhängigkeit vom Systemtakt und von den

CPU-Steuersignalen die Funktion der Einheit. Sie steuert den Datenaustausch zwischen CPU und dem CTC-Kanal, der durch die Kanalauswahleingänge CS0 und CS1 binär kodiert angesprochen wird.

Die Interruptsteuerlogik dient zur Zwischenspeicherung, Anmeldung, Quittierung und Rücksetzung von in den Kanälen durch Zählernulldurchgänge erzeugten Interruptanforderungen. Sie steuert im CTC die Interruptpriorität und erzeugt in Abhängigkeit vom anmeldenden Kanal und vom Inhalt des Vektorregisters den aktuellen Interruptvektor im Interruptquittierungszyklus der CPU.

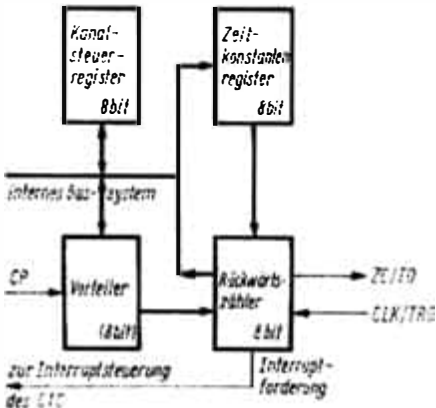


Bild 3.4.2  
Blockschaltbild der Kanallogik

Im Bild 3.4.2 ist die innere Struktur eines der vier Zähler/Zeitgeber-Kanäle dargestellt. Alle Kanäle sind funktionsmäßig völlig identisch. Jedoch ist der Zählerausgang des Kanals 3 aufgrund der Anschlußbeschränkung des 28poligen Gehäuses nicht vorhanden. Die Kanallogik besteht aus einem Kanalsteuerregister, einem Zeitkonstantenregister, einem 8-bit-Rückwärtszähler, einem 8-bit-Vorteiler und entsprechender Ansteuerlogik.

Das Kanalsteuerregister dient zur Festlegung der Betriebsart und der programmierbaren Eigenschaften des jeweils angesteuerten Kanals. Es kann von der CPU durch einen Ausgabebefehl (OUT-Operation) mit einem 8-bit-Kanalsteuerwort geladen werden. Dieses Kanalsteuerwort dient zur Auswahl der Betriebsart, zum Einstellen des Vorteilers, zur Festlegung der auslösenden Flanke des Zähleingangs, zur Auswahl des Triggerzeitpunkts, zum Laden einer Zeitkonstante in das Zeitkonstantenregister, zum Rücksetzen der Zählbedingungen und zur Freigabe bzw. Sperrung von Interruptanmeldungen durch Laden des Interruptfreigabeflupps.

Der Vorteiler ist ein voreinstellbarer 8-bit-Zähler, der den Systemtakt entweder um den Faktor 16 oder 256 herunterteilt. Er stellt in der Betriebsart Zeitgeber den Eingangstakt für den 8-bit-Rückwärtszähler bereit.

Das Zeitkonstantenregister ist ein 8-bit-Zwischenspeicher, das durch einen Ausgabebefehl (OUT-Operation der CPU) geladen werden kann. Es dient in beiden Betriebsarten des CTC-Kanals zum Voreinstellen des Rückwärtszählers. Der Inhalt des Zeitkonstantenregisters wird automatisch beim Nulldurchgang des Rückwärtszählers und bei der Initialisierung des Kanals in den Zähler eingeschrieben.

Der Rückwärtszähler ist ein 8-bit-Zähler, dessen aktueller Zählerstand durch einen Eingabebefehl (IN-Operation) von der CPU gelesen werden kann. Er wird in beiden Betriebsarten durch das Zeitkonstantenregister voreingestellt. In der Betriebsart Zähler wirkt der Kanaleingang CLK/TRG als Takteingang des Rückwärtszählers. In der Zeitgebermode erfolgt die Taktung des Zählers durch den Ausgang des 8-bit-Vorteilers. Bei

einem Nulldurchgang des Rückwärtszählers wird am Kanalausgang ZC/TO ein positiver Impuls erzeugt. Gleichzeitig kann durch dieses Ereignis in Abhängigkeit von der Interruptfreigabe eine Interruptanforderung ausgelöst werden. Der Rückwärtszähler des CTC-Kanals 3 besitzt keinen Ausgang ZC/TO.

### 3.4.3. Erläuterung der Anschlußbelegung

Im Bild 3.4.3 ist die schematische Anschlußbelegung der IS U857 dargestellt. Die Funktion der einzelnen Pins soll im folgenden näher beschrieben werden.

**D<sub>0</sub> ... D<sub>7</sub>** Data Bus (bidirektional, tristate)

Über den Datenbus des Systems U880 erfolgt der Datenaustausch (Steuerwörter, Interruptvektoren, Zählerstände) zwischen der CPU und dem Zähler/Zeitgeber.

**$\overline{\text{CE}}$**  Chip Enable (Eingang, L-aktiv)

Mit diesem Freigabesignal (L-Pegel) erfolgt die Aktivierung des CTC und somit das Ermöglichen des Informationsaustauschs mit der CPU. Das  $\overline{\text{CE}}$ -Signal wird üblicherweise in einem Adreßdekoder aus den CPU-Adressen A<sub>2</sub> ... A<sub>7</sub> erzeugt.

**CS<sub>0</sub>, CS<sub>1</sub>** Channel Select (Eingang, H-aktiv)

Diese Freigabesignale dienen zur binären Auswahl des anzusprechenden CTC-Kanals. Hierbei gilt folgende Zuordnung:

	CS <sub>0</sub>	CS <sub>1</sub>
Kanal 0	0	0
Kanal 1	0	1
Kanal 2	1	0
Kanal 3	1	1

Als Kanalauswahlsignale werden üblicherweise die CPU-Adressen A<sub>0</sub> und A<sub>1</sub> verwendet.

**CP** Clock Pulse (Eingang, 5V-Pegel)

Der Systemtakt dient zur internen Synchronisation der zeitlichen Abläufe der IS U857.

**$\overline{\text{M1}}$**  Machine Cycle 1 (Eingang, L-aktiv)

Das Signal  $\overline{\text{M1}}$  (Maschinenzyklus 1 der CPU) ist ein Steuersignal von der IS U880, das zur Kennzeichnung des Befehlsholezyklus (data fetch) dient. In bezug auf den CTC-Baustein dient es zur Quittierung und zur Rücksetzung der Interruptanmeldungen vom Zähler/Zeitgeber.

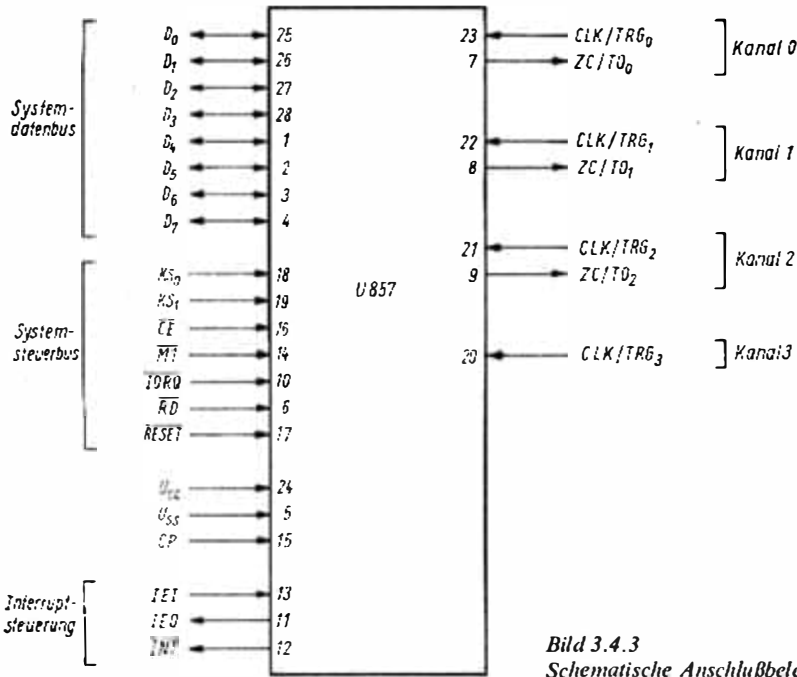
**$\overline{\text{IORQ}}$**  In/Out Request (Eingang, L-aktiv)

Dieses Steuersignal der CPU dient zur Kennzeichnung des Datenverkehrs zwischen CTC und CPU.

**$\overline{\text{RD}}$**  Read Cycle (Eingang, L-aktiv)

Dieses Steuersignal der CPU ist bei Lesevorgängen der CPU (Datenverkehr in Richtung CPU) aktiv. An der IS U857 steuert dieses Signal in Verbindung mit den Signalen  $\overline{\text{CE}}$ ,

CS0, CS1,  $\overline{\text{IORQ}}$  den Datentransport vom ausgewählten CTC-Kanal in Richtung CPU (Lesen der Kanalzählerstände). Das den Schreibvorgang (Datentransport von der CPU zur Peripherie) kennzeichnende Signal  $\overline{\text{WR}}$  (write cycle) wird aufgrund der Anschlußbeschränkung des Gehäuses intern in der IS erzeugt ( $\overline{\text{WR}} = \text{RD} + \overline{\text{CE}} + \overline{\text{IORQ}}$ ).



**IEI** Interrupt Enable Input (Eingang, H-aktiv)

Der Interruptfreigabeeingang dient in Verbindung mit dem Ausgang IEO zur Bildung einer systemweiten Interruptprioritätskette (daisy chain) durch die Kaskadierung aller interruptfähigen Peripheriegeräte (PIO, SIO, CTC). L-Pegel an diesem Eingang verbietet die Anmeldung einer Interruptanforderung eines CTC-Kanals an die CPU.

**IEO** Interrupt Enable Output (Ausgang, H-aktiv)

Der Interruptfreigabeausgang führt L-Pegel, wenn einer der CTC-Kanäle einen Interrupt anfordert bzw. eine ISR bearbeitet oder wenn der Eingang IEI L-Pegel führt.

**$\overline{\text{INT}}$**  Interrupt Request (Ausgang, open-drain, L-aktiv)

Das Signal  $\overline{\text{INT}}$  dient zur Anmeldung eines Interrupts an die CPU. Die Quittierung und Rücksetzung dieser Meldung erfolgt durch gleichzeitiges Aktivieren der CPU-Ausgänge  $\overline{\text{MI}}$  und  $\overline{\text{IORQ}}$ .

**$\overline{\text{RESET}}$**  Reset (Eingang, L-aktiv)

Mit aktivem  $\overline{\text{RESET}}$ -Signal wird der Zähler/Zeitgeber rückgesetzt. Hierbei werden folgende Vorgänge ausgelöst:

- Unterbrechung der Zählvorgänge aller Kanäle
- Rücksetzen der Interruptfreigabeflipflops aller Kanalsteuerregister

- Rücksetzen der aktuellen Interruptzustände aller Kanäle
- Ausgang  $\overline{\text{INT}}$  wird inaktiv
- Ausgänge ZC/TO werden inaktiv
- Datenbuslinien (D<sub>0</sub> bis D<sub>7</sub>) werden hochohmig.

Der CTC-Eingang  $\overline{\text{RESET}}$  wird üblicherweise mit dem systemweiten Rücksetzsignal verbunden.

**CLK/TRG** External Clock/Timer-Trigger (Eingang, aktiver Pegel durch Kanalsteuerwort programmierbar)

Diese vier Takt- und Triggereingänge wirken in Verbindung mit der zugehörigen CTC-Kanallogik. Ihre Funktion ist von der im entsprechenden Kanal selektierten Betriebsart abhängig.

*Zählermode:* Eine aktive Signalflanke (Auswahl durch Kanalsteuerwort, ob H/L- oder L/H-Flanke auslösend ist) bewirkt ein Dekrementieren des Rückwärtszählers.

*Zeitgebermode:* Die aktive Signalflanke (Auswahl durch Kanalsteuerwort) löst die Zeitgeberfunktion aus.

**ZC/TO** Zero Count/Time Out (Ausgang, H-aktiv)

Diese drei Ausgänge werden durch die zugehörigen Kanäle angesteuert. Kanal 3 des CTC (CS1 = 1, CS0 = 1) hat aufgrund der Anschlußbegrenzung des 28poligen Gehäuses keinen ZC/TO-Ausgang. Die Ausgänge führen in beiden Betriebsarten H-Potential, wenn der zugehörige Rückwärtszähler einen Nulldurchgang ausführt. Die Kanalausgänge sind zur Treibung von nachgeschalteten Darlingtonttransistoren ausgelegt.

### 3.4.4. Auswahl der Betriebsarten

#### 3.4.4.1. Rücksetzen

Nach Zuschalten der Betriebsspannung nimmt der Zähler/Zeitgeber einen undefinierten Zustand ein. Das Anlegen eines  $\overline{\text{RESET}}$ -Impulses an den zugehörigen Pin bewirkt ein Unterbrechen der Zählvorgänge aller Kanäle, ein Rücksetzen der Interruptbearbeitungszustände der Ports, die Inaktivierung aller Kanalausgänge sowie das Abschalten der Datenbusleitungen. Dieses hardwaremäßige Rücksetzen erfolgt üblicherweise in Mikrorechnern systemweit nach Einschalten der Betriebsspannung durch ein „power-on-reset“. Eine weitere Möglichkeit des Rücksetzens von Zähler/Zeitgeber-Kanälen ist softwaremäßig durch Setzen des Bits D<sub>1</sub> der Kanalsteuerregister gegeben. Hiermit kann ebenfalls eine Unterbrechung des Zählvorgangs erreicht werden (s. auch Abschn. 3.4.4.3.).

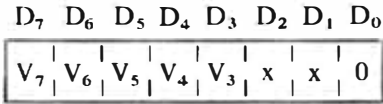
Zur Initialisierung der Kanäle des CTC als Zähler oder Zeitgeber müssen durch die CPU die entsprechenden Kanalsteuerregister und die Zeitkonstantenregister geladen werden. Um Interruptanmeldungen zu ermöglichen, muß das Interruptvektorregister durch ein entsprechendes Steuerwort, daß in den Kanal 0 geschrieben wird, definiert werden. Das Zählen bzw. die Zeitgeberfunktion beginnt daraufhin mit Erreichen der vorgewählten Auslösebedingungen.

#### 3.4.4.2. Interruptvektor

Bei Anwendung der leistungsfähigen Interruptbetriebsart IM2 der CPU U880 wird von dem den Interrupt anmeldenden peripheren Gerät ein Interruptvektor im Interruptquittierungszyklus der CPU abgefordert. Dieser 8-bit-Vektor bildet in Verbindung mit dem

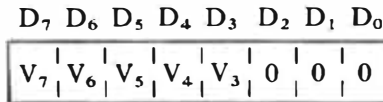
CPU-internen Register I einen 16-bit-Pointer, der im Hauptspeicher auf die Startadressentabelle der Interruptbearbeitung zeigt.

Beim CTC wird der Interruptvektor durch Einschreiben eines Steuerworts (OUT-Operation der CPU) in den Kanal 0 geladen. Dieses Steuerwort hat folgendes Format:

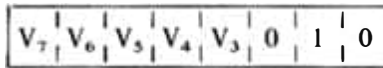


Das Bit D<sub>0</sub> = 0 identifiziert das Steuerwort als Interruptvektor. Beim Einschreiben dieses Vektors werden die beiden Bits D<sub>2</sub> und D<sub>1</sub> nicht berücksichtigt und können deshalb einen beliebigen Wert haben.

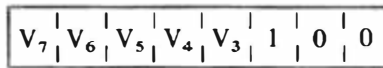
Bei Aussendung eines Interruptvektors nach vorausgegangener Interruptanmeldung und Quittierung durch die CPU geben diese Bits (D<sub>2</sub> und D<sub>1</sub>) binär kodiert die Kanalnummer des interruptauslösenden CTC-Kanals an. Somit können vom Zähler/Zeitgeber folgende Vektoren ausgesendet werden:



(Kanal 0 wird bearbeitet)



(Kanal 1 wird bearbeitet)



(Kanal 2 wird bearbeitet)

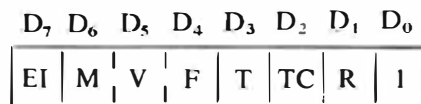


(Kanal 3 wird bearbeitet)

Bei Freigabe aller Interruptfreigabefllops können somit im Zusammenhang mit dem I-Register der CPU vier verschiedene Pointer gebildet werden, die in der Startadressentabelle auf vier unterschiedliche ISR-Startadressen hinweisen. Damit kann jeder Zähler/Zeitgeber-Kanal mit einer eigenen ISR bedient werden.

### 3.4.4.3. Kanalsteuerwort

Jeder der vier Kanäle der IS U857 kann entweder als Zähler oder als Zeitgeber arbeiten. Die Auswahl hierzu wird im Kanalsteuerregister vorgenommen. Ebenso werden in diesem Register die Vorteilerwahl, die Definition der Auslösebedingungen (Trigger), das Laden des Zeitkonstantenregisters, das Rücksetzen des Kanals und die Interruptfreigabe festgelegt. Das Steuerregister des entsprechenden Kanals (ausgewählt durch die Select-signalen CS0 und CS1) wird durch eine OUT-Operation der CPU mit dem Kanalsteuerwort geladen. Dieses Steuerwort hat folgendes Format:



Die Belegung des Bits  $D_0 = 1$  identifiziert das Steuerwort als Kanalsteuerwort. Es wird somit vom Interruptvektor des Kanals 0 unterschieden.

Das Bit  $D_7$  beeinflusst das Interruptfreigabeflipflop des ausgewählten Kanals. Bei Setzen des Bits  $D_7$  ( $D_7 = 1$ ) erfolgt die Weitergabe von Interruptanforderungen, die durch einen Nulldurchgang des Rückwärtszählers ausgelöst wurden, an den Steuereingang  $\overline{\text{INT}}$  der CPU. Voraussetzung für das Einschreiben von  $D_7 = 1$  in das Kanalsteuerregister ist, daß zuvor der Interruptvektor in den Kanal 0 eingeschrieben wurde. Bei rückgesetztem Bit ( $D_7 = 0$ ) wird vom ausgewählten Kanal beim Nulldurchgang des Rückwärtszählers kein Interrupt ausgesendet. Eine noch nicht bearbeitete anhängige Interruptanmeldung des Kanals wird zurückgenommen. Bei erneutem Setzen des Bits  $D_7$  durch ein neues Kanalsteuerwort erfolgt keine Interruptaussendung durch die vor dem Einschreiben dieses Steuerworts vorangegangenen Nulldurchgänge des Zählers.

Mit dem Bit  $D_6$  wird die Betriebsartenauswahl des CTC-Kanals vorgenommen. Bit  $D_6 = 1$  setzt den Kanal in die Zählerbetriebsart. Der Rückwärtszähler wird mit jeder Triggerflanke des am CLK/TRG-Eingang anliegenden externen Taktsignals dekrementiert. Mit Bit  $D_6 = 0$  wird die Zeitgebermode des CTC-Kanals ausgewählt. Der Rückwärtszähler wird durch den um den Vorteilerwert geteilten Systemtakt dekrementiert und erzeugt durch seine Nulldurchgänge Zeitintervalle mit einer Periode  $t_p$ :

$$t_p = t_c \cdot VT \cdot TC; \quad (3.4.1)$$

$t_c$  Periode des Systemtakts, ( $t_{c\min} = 400 \text{ ns}$ )

TC Wert des Zeitkonstantenregisters (TC = 1 ... 256)

VT Wert des Vorteilers (VT = 16, 256).

Das Bit  $D_5$  wird nur in der Betriebsart Zeitgeber ausgewertet. Es dient zur Auswahl des Vorteilerfaktors. Mit Bit  $D_5 = 1$  wird der Wert des Vorteilers auf  $VT = 256$  gesetzt; anderenfalls ( $D_5 = 0$ ) wird der Wert  $VT = 16$  eingestellt.

Mit Bit  $D_4$  wird in beiden Betriebsarten die aktive Flanke des am Kanaleingang CLK/TRG anliegenden Takt- bzw. Triggersignals definiert. In der Zählerbetriebsart erfolgt mit  $D_4 = 1$  das Dekrementieren des Rückwärtszählers bei jeder positiven Flanke bzw. bei  $D_4 = 0$  mit jeder negativen Flanke des externen Taktsignals. In der Betriebsart Zeitgeber erfolgt unter der Voraussetzung, daß der Zeitgebervorgang durch ein externes Triggersignal gestartet wird ( $D_3 = 1$ ), die Auslösung bei  $D_4 = 1$  mit der positiven bzw. bei  $D_4 = 0$  mit der negativen Flanke dieses Signals.

Das Datenbit  $D_3$  wird ebenfalls nur bei Auswahl der Betriebsart Zeitgeber ausgewertet. Es dient zur Definition des Auslösezeitpunkts des Zeitgebervorgangs. Bei gesetztem Bit  $D_3$  ( $D_3 = 1$ ) erfolgt der Start des Zeitgebervorgangs nach Eintreffen der mittels Bit  $D_4$  ausgewählten gültigen Flanke des am Eingang CLK/TRG anliegenden Triggersignals. Dieses Triggersignal wird erst nach dem Einschreiben der Zeitkonstante gültig. Der Vorteiler wird hierbei bei Einhalten der Voreinstellzeit (setup time) nach zwei Taktzyklen des Systemtakts erstmalig dekrementiert (anderenfalls nach drei Zyklen). Bei rückgesetztem Bit  $D_3$  ( $D_3 = 0$ ) beginnt die Zeitgeberoperation mit der steigenden Flanke des T2-Maschinenzustands (M1-Cycle, instruction fetch), der dem Laden der Zeitkonstante (port write cycle) folgt.

Mit Bit  $D_2$  erfolgt an die Steuerlogik des Zähler/Zeitgeber-Kanals eine Voranmeldung, ob mit der nächsten Portoperation (OUT-Befehl der CPU) zum betreffenden Kanal eine Zeitkonstante in das entsprechende Register geladen wird. Hierbei wird bei gesetztem Bit  $D_2$  ( $D_2 = 1$ ) das nächste Datenwort, das in den Kanal eingeschrieben wird, in das Zeitkonstantenregister geladen. Diese Aktion ist bei jeder Neuinitialisierung des CTC-Ports erforderlich. Wird in einen bereits initialisierten und sich in Betrieb befindenden Kanal

ein Steuerwort mit Bit  $D_2 = 1$  und ein nachfolgendes Zeitkonstantendatenwort eingeschrieben, so beendet der CTC-Kanal den laufenden Zyklus mit der alten Zeitkonstante. Nach dem Nulldurchgang des Rückwärtszählers wird dann die neue Zeitkonstante rückgeladen und im folgenden auch weiter benutzt. Auf diese Art und Weise kann im laufenden Zähler- oder Zeitgeberbetrieb die Zeitkonstante geändert werden. Wenn das Bit  $D_2$  nicht gesetzt ist ( $D_2 = 0$ ), erfolgt kein Einschreiben eines nachfolgenden Datenworts in das Zeitkonstantenregister des Kanals. Diese Bedingung ist nur dann sinnvoll, wenn das betreffende Port bereits initialisiert ist und mit dem neuen Datenwort andere Bedingungen (z. B. Interruptfreigabe, Betriebsart oder Triggerflanke) neu festgelegt werden sollen. Der CTC-Kanal arbeitet dann immer mit der alten Zeitkonstante weiter.

Das Datenbit  $D_1$  des Kanalsteuerworts dient zum softwaremäßigen Zurücksetzen oder zum Unterbrechen des ablaufenden Zähler- oder Zeitgebervorgangs. Bei gesetztem Bit  $D_1$  ( $D_1 = 1$ ) unterbricht der betreffende Kanal das Dekrementieren des Rückwärtszählers. Es werden hierbei aber keine Bits des Kanalsteuerregisters und somit keine aktuellen CTC-Bedingungen verändert. Wenn außerdem das Bit  $D_2$  gesetzt ist ( $D_2 = 1$ ), führt der CTC-Kanal nach Einschreiben einer Zeitkonstante die Operation in der vorgewählten Betriebsart fort. Ist Bit  $D_2$  rückgesetzt ( $D_2 = 0$ ), besteht in der Neuinitialisierung des Ports durch Einschreiben eines neuen Kanalsteuerworts die einzige Möglichkeit, den Kanal wieder in Betrieb zu nehmen. Bei rückgesetztem Steuerwortbit  $D_1$  ( $D_1 = 0$ ) wird der Zähler- bzw. Zeitgeberbetrieb nicht unterbrochen. Diese Möglichkeit wird angewendet, wenn Bedingungen im Kanal verändert werden sollen (Verändern des Interruptfreigabefllops, Laden einer neuen Zeitkonstante u. dgl.), ohne dabei den bearbeiteten Vorgang zu unterbrechen.

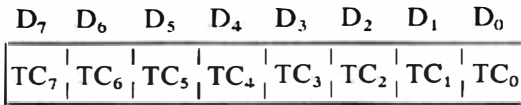
### 3.4.4.4. Zeitkonstante

Nach der Initialisierung eines Zähler/Zeitgeber-Kanals mit Hilfe eines entsprechenden Kanalsteuerworts kann das CTC-Port erst nach Laden eines Datenworts in das Zeitkonstantenregister des ausgewählten Kanals mit der programmierten Operation beginnen. Das Laden des Zeitkonstantenregisters erfolgt durch eine OUT-Operation der CPU, wenn zuvor in den Kanal ein Kanalsteuerwort mit gesetztem Bit  $D_2$  ( $D_2 = 1$ ) eingeschrieben wurde. Das hiermit gesetzte Bit  $D_2$  des Kanalsteuerregisters bewirkt, daß die Kanalsteuerlogik dieses nachfolgende Datenwort von der CPU als Zeitkonstante interpretiert. Deshalb werden keine besonderen Bitbelegungen des Zeitkonstantenworts zur Kenn-

Tafel 3.4.1. Programmierung des CTC

Interruptvektor							
V <sub>7</sub>	V <sub>6</sub>	V <sub>5</sub>	V <sub>4</sub>	V <sub>3</sub>	X	X	0
Kanalsteuerwort							
EI	M	V	F	T	TC	R	I
EI	Interruptfreigabe			T	Triggerzeitpunkt		
M	Betriebsart			TC	Zeitkonstante folgt		
V	Vorteilerfaktor			R	Rücksetzen		
F	Triggerflanke						
Zeitkonstantenwort, wenn TC=1 im Kanalsteuerwort							
TC <sub>7</sub>	TC <sub>6</sub>	TC <sub>5</sub>	TC <sub>4</sub>	TC <sub>3</sub>	TC <sub>2</sub>	TC <sub>1</sub>	TC <sub>0</sub>

zeichnung benötigt, somit sind alle 8 bit für die Informationsübertragung gültig. Das Zeitkonstantenwort hat folgendes Format:



Das Bit D<sub>7</sub> dieser Zeitkonstante ist das höchstwertige Bit (MSB). Aufgrund des Umfangs von 8 bit kann der Wert der Zeitkonstante zwischen 1 und 256 liegen. Hierbei ist dem hexadezimalen Wert TC = 00H die Zeitkonstante TC = 256D zugeordnet.

**3.4.4.5. Zusammenfassung**

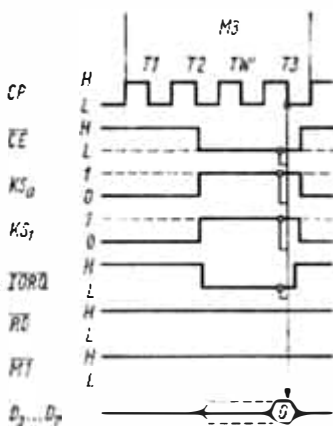
In der Tafel 3.4.1 ist die Programmierung vom Zähler/Zeitgeber U857 zusammenfassend dargestellt.

**3.4.5. Beschreibung der Betriebsarten**

**3.4.5.1. Zählermode**

Der Zähler/Zeitgeber U857 hat vier Kanäle, die unabhängig voneinander betrieben werden können. Ein Kanal des CTC nimmt die Betriebsart Zähler ein, wenn in sein Kanalsteuerregister ein Datenwort mit gesetztem Bit D<sub>6</sub> geladen wird. Bei einer Initialisierung, also bei zuvor rückgesetztem Port, muß zur Inbetriebnahme außerdem eine Zeitkonstante in das entsprechende Register dieses Kanals eingeschrieben werden. Der Rückwärtszähler wird dann bei jedem Aktivwerden bzw. sinngemäß mit jeder aktiven Flanke am Kanaleingang CLK/TRG dekrementiert. Hierbei wird durch Bit D<sub>4</sub> im Steuerregister des Kanals festgelegt, welcher Pegel an diesem Eingang aktiv ist (H oder L).

Wird einer der Kanäle (unabhängig von der gewählten Betriebsart) zur Erzeugung von Interrupts benutzt, muß vor der Inbetriebnahme dieses Kanals (mit gesetztem Interrupt-freigabeflipflop) das Vektorregister des Kanals 0 geladen werden.



**Bild 3.4.4**  
**Zeitverhalten des CTC im Schreibzyklus**  
 M3 Outputmaschinenzyklus der CPU  
 CP Systemtakt (T1, T2, TW, T3 Taktzustände der CPU)  
 EE Bausteinfreigabesignal  
 KS<sub>0</sub>, KS<sub>1</sub> Kanalauswahlsignale zur Anwahl der CTC-Kanäle 0 ... 3  
 TORQ, MI, RD CPU-Steuersignalausgänge bzw. Steuersignaleingänge des CTC  
 D<sub>0</sub> ... D<sub>7</sub> Belegung des Systemdatenbusses (0 gültige Übernahmedaten für den CTC)

Das Einschreiben dieser Datenwörter in die Kanalsteuerregister und die Zeitkonstantenregister der CTC-Kanäle sowie in das Vektorregister erfolgt über Ausgabebefehle des Prozessors (OUT-Operation der CPU). Das hierbei auftretende Zeitverhalten am Zähler/Zeitgeber U857 ist im Bild 3.4.4 dargestellt. Aus diesen Zeitbedingungen ist zu ent-

nehmen, daß die IS U857 direkt mit dem Prozessor U880 zusammengeschaltet werden kann.

Der aktuelle Stand des Rückwärtszählers kann jederzeit durch einen Eingabebefehl (IN-Operation) der IS U880 abgefragt werden. Das zugehörige Zeitverhalten vom Zähler/Zeitgeber ist im Bild 3.4.5 verdeutlicht. Der mit der steigenden Flanke des CPU-Systemtaktzyklus T2 in diesem Lesevorgang gültige Inhalt des Rückwärtszählers des durch die binären Auswahlsignale CS0 und CS1 angesprochenen Kanals wird auf den Datenbus ausgegeben.

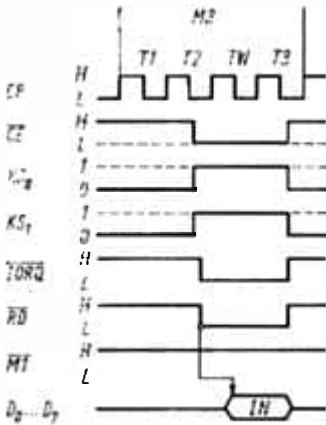


Bild 3.4.5  
Zeitverhalten des CTC im Lesezyklus

- M3 Outputmaschinenzyklus der CPU
- CP Systemtakt (T1, T2, TW', T3 Taktzustände der CPU)
- CE Bausteinfreigabe
- CS<sub>0</sub>, CS<sub>1</sub> Kanalauswahlsignale zur Anwahl der CTC-Kanäle 0 ... 3
- TORQ, M1, RD CPU-Steuersignalausgänge bzw. Steuersignaleingänge des CTC
- D<sub>0</sub> ... D<sub>7</sub> Belegung des Systemdatenbusses (IN Ausgabedaten des CTC: Zählerstand des Rückwärtszählers)

Beim Nulldurchgang des Rückwärtszählers wird automatisch die Zeitkonstante aus dem entsprechenden Kanalregister in den Zähler rückgeladen. Dabei erfolgt keine Unterbrechung des Zählvorgangs. Wurde zuvor ein neues Datenwort in das Zeitkonstantenregister durch entsprechende Vorankündigung mit einem Kanalsteuerwort ( $D_6 = 1, D_2 = 1, D_1 = 0, D_0 = 1$ ) geschrieben, so erfolgt nach dem Nulldurchgang das Rückladen dieser neuen Zeitkonstante.

Gleichzeitig wird bei jedem Nulldurchgang des Rückwärtszählers am Kanalausgang ZC/TO ein positiver Impuls mit der Länge von etwa anderthalb Systemtaktperioden ausgesendet. Dieses Signal kann beispielsweise zur Kaskadierung von CTC-Kanälen verwendet werden. Bei Verwendung dieses Übertragsimpulses zur Taktung eines nachfolgenden Zählerkanals können leistungsfähige Zähler mit großem Zählumfang (z. B. 16-bit-Zähler bei Verwendung von zwei Ports) realisiert werden. Aufgrund der Pinbeschränkung des 28poligen DIL-Gehäuses besitzt jedoch der Kanal 3 keinen ZC/TO-Ausgang. Dieser Kanal kann also nur dann eingesetzt werden, wenn keine hardwaremäßige Kaskadierung bzw. Auswertung des Übertrags notwendig ist.

Vom Übertrag der Rückwärtszähler wird außerdem in allen vier CTC-Kanälen bei entsprechend gesetzten Freigabeflipflops in den Kanalsteuerlogiken eine Interruptanforderung ausgelöst. Die Anwendung der leistungsfähigen Interruptbearbeitung des Systems (Interruptmode IM2 der CPU U880) kann aber nur erfolgen, wenn das Vektorregister des CTC geladen wurde.

Im Bild 3.4.6 ist zusammenfassend das Zeitverhalten eines CTC-Kanals dargestellt.

Während des Betriebs können Zählbedingungen für den Kanal verändert werden, indem ein neues Steuerwort in das Kanalsteuerregister eingeschrieben wird. In diesem Datenwort muß Bit  $D_1$  rückgesetzt sein, damit der Dekrementiervorgang des Rückwärtszählers nicht unterbrochen wird. Es ist somit möglich, eine neue Zeitkonstante anzukün-

digen, die Zählflanke zu ändern, das Interruptfreigabeflapp zu beeinflussen oder in die Betriebsart Zeitgeber überzuwechseln. Ein Rücksetzen des Kanals kann softwaremäßig durch Setzen des Bits  $D_1$  des Kanalsteuerregisters oder hardwaremäßig durch Anlegen eines RESET-Impulses erfolgen.

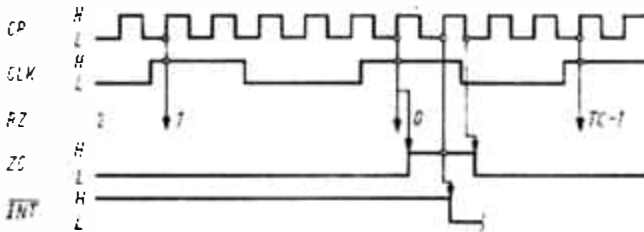


Bild 3.4.6. Zeitverhalten des CTC im Zählerbetrieb

CP	Systemtakt
CLK	CTC-Kanaleingang CLK/TRG in Zählermode
RZ	aktueller Inhalt des internen Kanalrückwärtszählers (TC geladene Zeitkonstante)
ZC	CTC-Kanalausgang ZC/TO in Zählermode
INT	Interruptausgang des CTC (nur Wirkung des betreffenden Kanals dargestellt)

### 3.4.5.2. Zeitgebermode

Das Einstellen dieser CTC-Betriebsart erfolgt ebenfalls durch Laden eines Steuerworts in das entsprechende Kanalsteuerregister. Das Bit  $D_6$  dieses Datenworts ist hierbei rückgesetzt. Der Zeitgebervorgang beginnt in Abhängigkeit vom Wert des Bits  $D_3$  dieses Steuerworts entweder direkt nach dem Laden der Zeitkonstante in das betreffende Kanalregister oder nachdem bei bereits geladenem Zeitkonstantenregister am Eingang CLK/TRG eine vorgewählte Triggerbedingung eintritt. Das Laden der Kanalsteuerregister und Zeitkonstantenregister erfolgt wie in der Zählermode mit dem im Bild 3.4.4 dargestellten Zeitverhalten. Auch das Auslesen des Inhalts der Rückwärtszähler erfolgt wie im Abschn. 3.4.5.1. bereits beschrieben (Bild 3.4.5).

Der Rückwärtszähler des Zähler/Zeitgeber-Kanals wird durch den Vorteiler getaktet. Hierbei erfolgt ein Dekrementieren des Zählers je nach Wert des Bits  $D_5$  des Kanalsteuerregisters mit jeder 16. bzw. mit jeder 256. Systemtaktperiode. Die Zeitkonstante des Zeitgebervorgangs setzt sich somit aus dem Produkt der Werte des Vorteilers und der Zeitkonstante sowie der Taktperiode zusammen (Gl. 3.4.1). Der Wert  $TC = 0$  des Zeitkonstantenregisters entspricht hierbei einer Zeitkonstante von 256 D. Es lassen sich also je Kanal Zeitgeber mit einem Zählumfang von 16 bit, ausgehend von dem Systemtakt, erzielen. Dieser Zählumfang eines Kanals kann um jeweils 8 bit vergrößert werden, wenn eine Kaskadierung mit Hilfe des Kanalausgangs ZC/TO erfolgt. An diesem Ausgang wird bei jedem Übertrag des Rückwärtszählers ein positiver Impuls ausgesendet. Dieses Signal kann zur hardwaremäßigen Kaskadierung verwendet werden, indem es zur Ansteuerung eines nachfolgenden Kanals, der in der Zählermode betrieben werden muß, benutzt wird.

Beim Nulldurchgang des Rückwärtszählers erfolgt ebenso wie in der Betriebsart Zähler unter den Interruptfreigabebedingungen die Aussendung einer Interruptanmeldung. Voraussetzung für die Anwendung der Interruptbetriebsart IM2 der CPU U880 ist wiederum, daß das Vektorregister der IS U857 geladen wurde.

Das Zeitverhalten eines CTC-Kanals in der Zeitgebermode ist zusammenfassend im Bild 3.4.7 dargestellt.

Ebenso wie in der Betriebsart Zählermode können während eines laufenden Zeitgebervorgangs alle im Kanalsteuerregister festgelegten Bedingungen geändert werden. Mit

rückgesetztem Bit  $D_1$  ( $D_1 = 0$ ) wird gesichert, daß der Zeitgebervorgang nicht unterbrochen wird. Das Rücksetzen des Kanals erfolgt mit Setzen des Bits  $D_1$  im Kanalsteuerwort bzw. hardwaremäßig durch Auslösen eines RESET-Impulses.

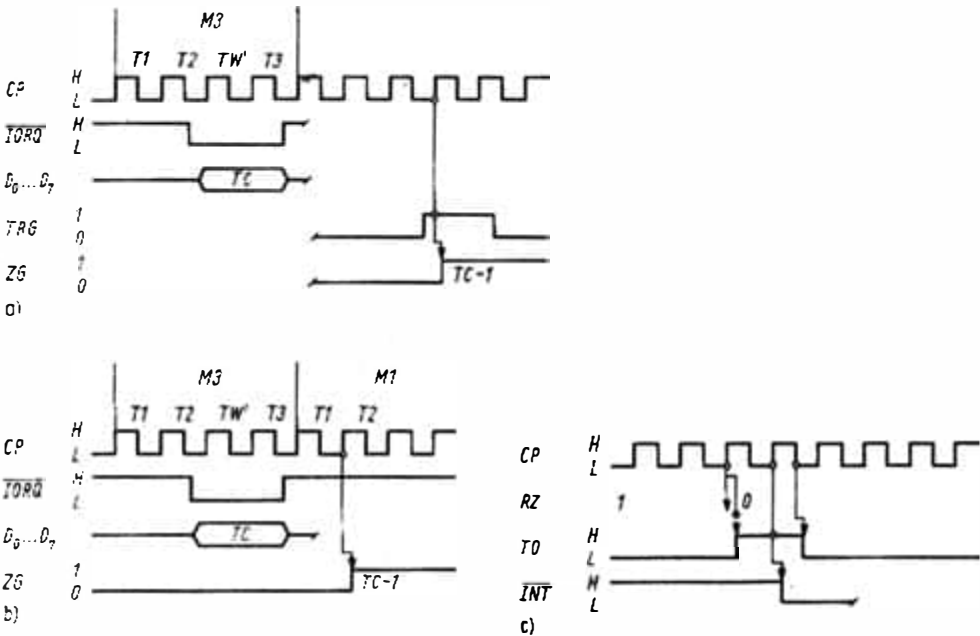


Bild 3.4.7. Zeitverhalten des CTC im Zeitgeberbetrieb

- a) Auslösung des Zeitgebervorganges durch Triggersignal
- b) Auslösung nach Laden der Zeitkonstante
- c) Aktivitäten beim Nulldurchgang des Rückwärtszählers

M3	Outputmaschinenzyklus der CPU
M1	M1-Maschinenzyklus der CPU (Befehlslesen)
CP	Systemtakt (T1, T2, TW', T3 Taktzustände der CPU)
$\overline{IORQ}$	Steuersignalausgang der CPU bzw. Steuersignaleingang des CTC
$D_0 \dots D_7$	Belegung des Systemdatenbusses (TC gültige Ausgabedaten der CPU; Zeitkonstante für den CTC-Kanal)
TRG	CTC-Kanaleingang CLK/TRG in Zeitgebermode (aktiver Pegel programmierbar)
ZG	Beginn des Zeitgebervorganges durch erstmaliges Dekrementieren des Kanalrückwärtszählers
RZ	aktueller Inhalt des internen Rückwärtszählers
TO	CTC-Kanalausgang ZC/TO in Zeitgebermode
$\overline{INT}$	Interruptausgang des CTC (nur Wirkung des betreffenden Kanals dargestellt)

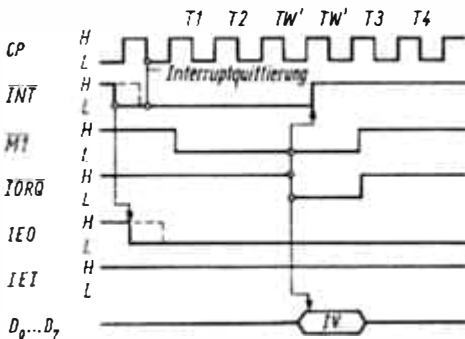
### 3.4.6. Interruptbearbeitung

Die Auslösung von Interruptanmeldungen eines Zähler/Zeitgeber-Kanals erfolgt unabhängig von der gewählten Betriebsart durch die Nulldurchgänge des Rückwärtszählers. Voraussetzungen für die Weitergabe dieser Anmeldungen an die CPU sind, daß sowohl das Interruptfreigabeflipflop der Kanalsteuerlogik gesetzt ist als auch der betrachtete Kanal die höchste aktuelle Interruptpriorität im Mikrorechnersystem aufweist (also am Eingang IEI H-Potential führt).

Diese Interruptpriorität wird im System U880 in der Interruptprioritätenkette ermittelt. Die Kette entsteht durch systemweites Zusammenschalten der peripheren Schaltkreise (PIO, SIO, CTC) mit Hilfe der Interruptsignale IEI und IEO. Ein H-Potential am Eingang einer IS U857 bedeutet somit, daß der z. Z. höchstwertige Kanal zur Weitergabe seiner vorliegenden Interruptanmeldung über die  $\overline{INT}$ -Leitung zur CPU berechtigt ist.

Bei einer solchen Interruptanmeldung zum Prozessor wird der Interruptausgang IEO rückgesetzt (IEO = L), um Interruptanmeldungen von nachfolgenden (also niederwertigeren) Peripherieelementen zu verhindern. Die Interruptanmeldung und Rücksetzung des IEO-Ausgangs wird jedoch während eines aktiven  $\overline{MI}$ -Signals der CPU für die Dauer dieses Signals verzögert. Diese Maßnahme ist erforderlich, damit die Interruptprioritätenkette im Interruptquittierungszyklus (gekennzeichnet durch  $\overline{MI}$ ) einen eingeschwungenen Zustand aufweist.

Der Zähler/Zeitgeber U857 hat vier Kanäle, die in bezug auf das Interruptverhalten völlig gleichwertig wirken. Jeder Kanal besitzt in seiner Steuerlogik ein Interruptfreigabeflipflop und antwortet im Fall einer Interruptanmeldung im Quittierungszyklus der CPU mit einem eigenen Interruptvektor. Dieser Interruptvektor wird dadurch erzeugt, daß der für alle Kanäle gleichzeitig zuvor in das Interruptvektorregister des Kanals 0 geladene Vektor durch Hinzufügen von 2 bit spezifiziert wird. Diese Bits ( $D_1$  und  $D_2$  des Vektors) enthalten die binär kodierte Kanalnummer des aussendenden Ports. Sie ermöglichen somit, daß der Prozessor auf jeden Kanalinterrupt mit einer zugehörigen ISR reagieren kann. Die Interruptpriorität unter den vier Zähler/Zeitgeber-Kanälen wird durch die Reihenfolge der Kanalnummern festgelegt. Der Kanal 0 hat hierbei die höchste Priorität. Er kann somit bei Einhaltung der Interruptfreigabebedingungen die Serviceroutinen aller anderen Kanäle unterbrechen.



**Bild 3.4.8**  
Zeitverhalten des CTC im Interruptquittierungszyklus der CPU

- CP Systemtakt (T1, T2, TW', T3, T4 Taktzustände der CPU)
- $\overline{INT}$  Interruptausgang des CTC (nur Quittierung eines Kanalinterrupts dargestellt)
- $\overline{MI}, IORQ$  Steuersignalausgänge der CPU bzw. Steuerungseingänge des CTC
- IEI, IEO Interruptfreigabelinien des CTC
- $D_0 \dots D_7$  Belegung des Systemdatenbusses (IV vom quittierten CTC-Kanal ausgesendeter Interruptvektor)

Bei einem Nulldurchgang des Rückwärtszählers eines CTC-Kanals sendet der CTC unter den Voraussetzungen, daß das Interruptfreigabeflipflop dieses Ports gesetzt war und der Kanal die höchste aktuelle Interruptpriorität aufweist, eine Interruptanmeldung an den Prozessor. Die CPU fragt nach jedem Befehl die Interruptanmeldungslinie ( $\overline{INT}$ -Eingang der CPU) ab und reagiert bei freigegebenem CPU-Interruptflipplop (Freigabe z. B. nach Befehl EI) mit einem Interruptquittierungszyklus auf eine erkannte Anmeldung. Dieser Quittierungszyklus unterbricht die bisherige Programmabarbeitung und bewirkt eine Abfrage des Interruptvektors des anmeldenden Kanals. Im Bild 3.4.8 ist das für diesen Zyklus charakteristische Zeitverhalten des Zähler/Zeitgebers U857 dargestellt. In dem Interruptquittierungszyklus erfolgt automatisch das Abspeichern des aktuellen Befehlszählerstands im Stapelspeicher (stack). Gleichzeitig erfolgt die Bildung des neuen Befehlszählers, der auf die Anfangsadresse der eingeschobenen ISR zeigt. Der neue 16-bit-Befehlszählerstand (PC) wird aus zwei aufeinanderfolgenden Speicherplätzen entnommen (Low-Byte, High-Byte), die von einem 16-bit-Pointer (Zeiger) von der CPU adressiert werden. Dieser Interruptpointer setzt sich aus dem I-Register der CPU (8 bit, High-Byte) und dem Interruptvektor des anmeldenden CTC-Kanals (8 bit, Low-Byte) zusammen.

Da Bit  $D_0$  des Vektors den Wert  $V_0 = 0$  aufweist, zeigt der Pointer auf eine „gerade“ Speicherplatzadresse. Diese Zelle enthält das Low-Byte des neuen Befehlszählers. Der um eins inkrementierte Pointer weist dann auf den Speicherplatz, der das High-Byte enthält. Nach dem Ende des eingeschobenen Interruptquittierungszyklus bearbeitet der Prozessor dann den ersten Befehl der durch den interruptenden Kanal aufgerufenen ISR.

Hierbei wird zunächst automatisch das Einschleichen weiterer ISR von höherwertigen Kanälen bzw. peripheren Geräten, die spätere Interruptanmeldungen an die CPU weitergaben, durch systembedingtes Zurücksetzen des CPU-Interruptfreigabefllops verhindert. Erfolgt aber im Befehlsablauf der ISR eine Interruptfreigabe (durch Befehl EI), so wird erneut das laufende Programm unterbrochen und die ISR des höherwertigen Kanals (bzw. Geräts) eingeschoben. Eine laufende Serviceroutine wird durch den Befehl RETI abgeschlossen. Hierbei erfolgt automatisch das Rückspeichern des letzten aktuellen Standes des Befehlszählers (vor Beginn der abgeschlossenen ISR) aus dem Stapelspeicher (stack). Gleichzeitig erfolgt die Rücksetzung des Interruptbearbeitungszustands des Kanals, der diese beendete ISR anmeldete. Dieser Kanal ist dadurch gekennzeichnet, daß er der höchstwertig quitierte Kanal ist. Die Kanallogik muß hierzu parallel zum Prozessor den Datenverkehr in den Befehlsholzyklen der CPU (data fetch) überwachen, um den Befehl RETI dekodieren zu können. Im Bild 3.4.9 ist das Zeitverhalten am Zähler/Zeitgeber während des RETI-Befehls verdeutlicht.

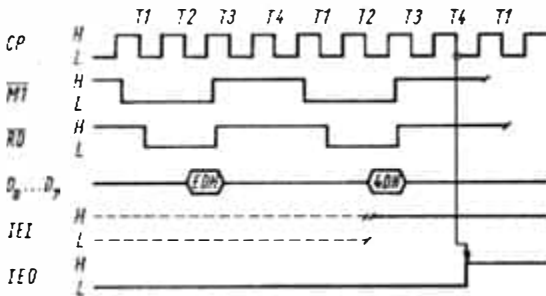


Bild 3.4.9

Zeitverhalten des CTC bei Verlassen des Interruptbearbeitungszustands

CP	Systemtakt (T1, T2, T3, T4 Taktzustände der CPU)
$\overline{MI}$ , $\overline{RD}$	Steuersignalausgänge der CPU bzw. Steuersignaleingänge des CTC
$D_0 \dots D_7$	Belegung des Systemdatenbusses (EDH, 4DH Befehlskodaten des RETI-Befehls aus Speicher)
IEI, IEO	Interruptfreigabelinien des CTC

Folgende Besonderheit ist bei der Abarbeitung des RETI-Befehls zu beachten:

Zwei CTC-Schaltkreise liegen in der Interruptprioritätenkette. Ein Kanal vom niedrigerwertigen Zähler/Zeitgeber meldet eine Interruptanforderung an den Prozessor an. Diese Anmeldung wird in einem Interruptquittierungszyklus anerkannt, und die ISR dieses Kanals wird in die Befehlsfolge eingeschoben. In dieser ISR erfolgt keine weitere Interruptfreigabe der CPU mit Hilfe eines EI-Befehls. Daraufhin löst ein Port des höherwertigen CTC ebenfalls eine Interruptanmeldung aus. Der CTC-Ausgang IEO führt L-Potential, das in der Prioritätenkette durchgeschaltet wird. Die Anmeldung wird jedoch wegen der obengenannten Bedingung nicht quitiert. Am Ende der durch den niedrigerwertigen CTC-Kanal eingeschobenen ISR erfolgt die Ausführung des RETI-Befehls. Dieser Befehl wird parallel zum Prozessor wiederum von beiden Zählern/Zeitgebern mitdekodiert. Hierbei gibt nun der höherwertigere CTC-Kanal nach Dekodierung des ersten RETI-Bytes (EDH) die IEO-Linie für die Dauer dieses Befehls frei (IEO = H). Die Freigabe der Kaskadierungslinie IEI – IEO durch alle höherwertig anmeldenden, aber nicht durch die CPU quitierten Kanäle nach Empfang des RETI-Bytes EDH ist notwendig, damit der in Bearbeitung befindliche höchstwertige Kanal bei Empfang des zweiten RETI-Bytes (4DH) am Eingang IEI H-Potential besitzt. Er ist somit als höchstwertiger bearbeiteter Kanal identifiziert (Eingang IEI = H, Ausgang IEO = L) und verläßt den Inter-

ruptbearbeitungszustand. Der Prozessor arbeitet nun die zuvor unterbrochene Programmfolge, die durch den rückgespeicherten Befehlszähler gekennzeichnet wird, weiter ab. Nach Interruptfreigabe im weiteren Programmablauf durch den Befehl EI (evtl. auch schon am Ende der ISR) erfolgt die Quittierung des noch anstehenden höherwertigen Kanalinterrupts sowie das Einschleichen der zugehörigen ISR.

Erfolgt bei gesetztem Interruptfreigabeflipflop eines CTC-Kanals ein Nulldurchgang des Rückwärtszählers, so geschieht bei L-Potential am Eingang IEI keine Weitergabe der Interruptanmeldung an die CPU. Die Anmeldung wird jedoch im CTC-Kanal zwischengespeichert und nach Aktivwerden des Eingangs IEI (H-Pegel) zum Prozessor weitergeleitet.

### 3.4.7. Einsatz in Mikrorechnersystemen

#### 3.4.7.1. Hardware

Die Zusammenschaltung des Zählers/Zeitgebers mit dem Prozessor U880 erfordert signalmäßig keine Anpassung. Somit kann in Minimalsystemen der CTC daten- und steuerbusmäßig direkt mit der CPU verbunden werden. Ansteuerseitig kann ebenfalls auf einen Adreßdekoder für die CTC-Auswahlsignale ( $CS_0$ ,  $CS_1$ ,  $\overline{CE}$ ) verzichtet werden und eine Zusammenschaltung dieser Eingänge mit drei Adreßlinien ( $A_0$  und  $A_1$  für  $CS_0$  und  $CS_1$ ) erfolgen. Bei einem Einsatz in großen Mikrorechnern erlauben die Zeitbedingungen sowohl des Prozessors als auch des CTC die Verwendung von Datenbustreibern, Adreßdekodern und Puffern für die Steuersignale. Üblich ist in Mikrorechnersystemen der Einsatz von Datenbustreibern prozessorseitig und periphereseitig, um Störgrößen und Leitungskapazitäten zu vermindern. Der Steuer- und Adreßbus wird prozessorseitig getrieben, um die in den peripheren Baugruppen auf diesen Linien auftretenden Lastfaktoren versorgen zu können. Weitere Signalverzögerungen können in Adreßdekodern und Systemtakttreibern periphereseitig auftreten.

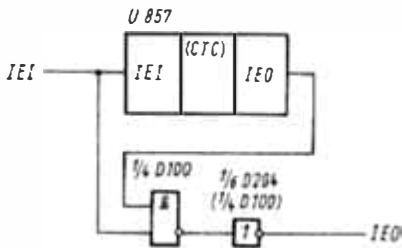


Bild 3.4.10  
Einfache IEI-IEO-Umgehungslogik

Die Interruptkaskadierung des CTC mit weiteren peripheren Schaltkreisen des Systems mit Hilfe der IEI-IEO-Linie erfordert in größeren Mikrorechnern (mehr als vier periphere Elemente) eine hardwaremäßige Umgehungsschaltung (look ahead). Bei der Interruptanmeldung eines höherwertigen peripheren Geräts geschieht die Weitergabe der Interruptsperrung an die nachfolgenden Peripherieelemente mit dem Durchlauf des L-Pegels in der dem auslösenden Gerät nachgeschalteten Interruptkaskadierungskette. Dieses Durchlaufen des L-Pegels kann aber durch jedes periphere Gerät um etwa 190 ns verzögert werden. Damit nun im Interruptquittierungszyklus des CPU mit Aussenden des  $\overline{IORQ}$ -Signals ein eingeschwungener Zustand vorliegt, muß ein schnelles Durchschalten der Kette auf L-Pegel durch eine Umgehungsschaltung erzielt werden. Im Bild 3.4.10 ist eine Möglichkeit der Umgehungsschaltung für den CTC dargestellt.

### 3.4.7.2. Software

Die Initialisierung des Zählers/Zeitgebers erfolgt üblicherweise mit den anderen peripheren Geräten am Anfang des Hauptprogramms. Zu dieser Anfangsinitialisierung gehören das Festlegen des CTC-Interruptvektors, das Laden des I-Registers der CPU (bei Anwendung des Interruptbetriebs) und die Auswahl der Betriebsarten durch Laden der Kanalregister. Das Einschreiben der Zeitkonstanten in die entsprechenden Kanalregister kann an geeigneter Stelle im Programmablauf erfolgen und somit als zusätzliche Triggerbedingung benutzt werden.

Bei Anwendung des Interruptbetriebs (Mode IM2 der IS U880) muß im Hauptspeicher des Mikrorechners eine Vektortabelle enthalten sein, die adressenmäßig zu den einzelnen Kanälen korreliert. Diese Vektortabelle enthält die Startadressen der ISR und wird durch einen im Interruptquittierungszyklus aus dem I-Register der CPU und dem Interruptvektor gebildeten Pointer adressiert. Bei Interruptbereitschaft aller vier Kanäle eines CTC müssen acht aufeinanderfolgende Speicherplätze zur Bildung der zugehörigen Tabelle verwendet werden. Im Mikrorechner können aus den 8-bit-Interruptvektoren der peripheren IS bei unverändertem I-Register der CPU somit 128 unterschiedliche Startadressen der ISR gebildet werden.

### 3.4.7.3. Anwendungsmöglichkeiten

Aufgrund der umfangreichen Programmiermöglichkeiten ergeben sich für den Zähler/Zeitgeber U857 vielfältige Anwendungsmöglichkeiten. Nachfolgend sollen deshalb einige wichtige Einsatzgebiete aufgezeigt werden.

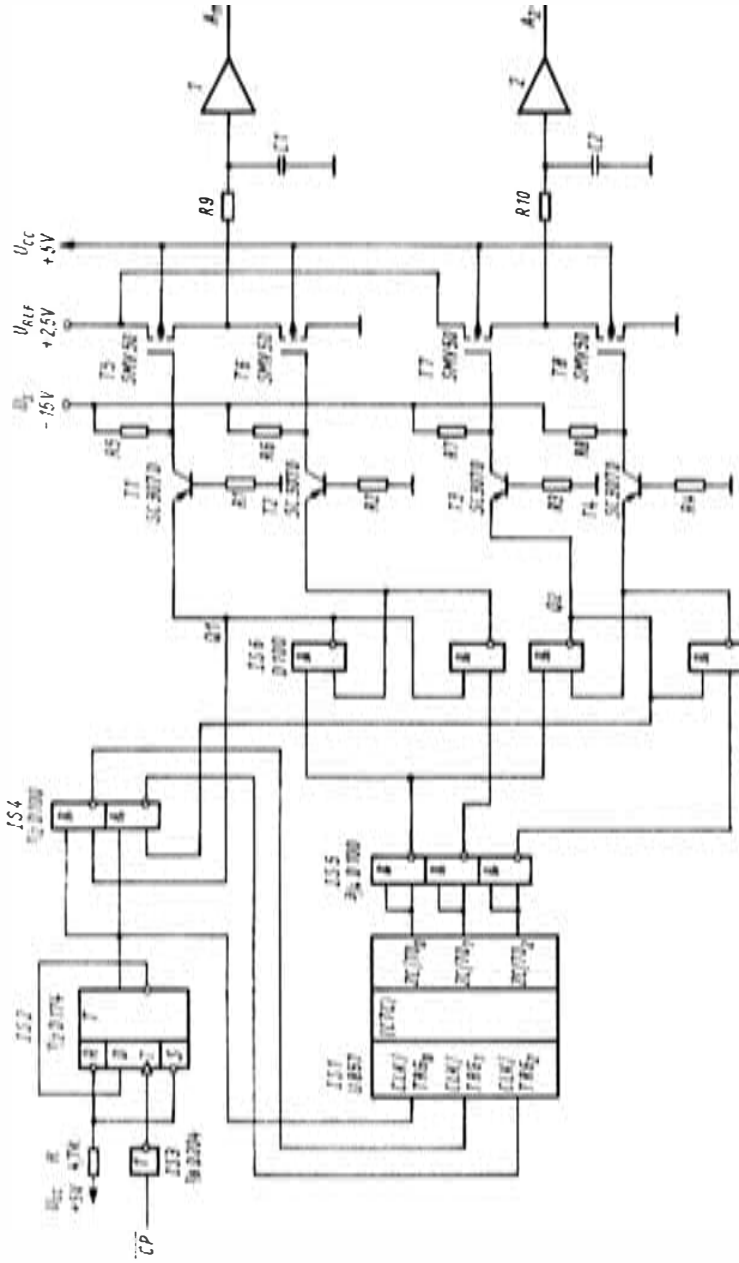
In der Betriebsart Zeitgeber werden durch automatisches Dekrementieren des Rückwärtszählers zeitgleiche Impulse erzeugt, die als Taktsignale für verschiedenste Anwendungsfälle verwendet werden können. Ein Zähler/Zeitgeber-Kanal in dieser Mode kann z. B. zur Empfangs- und Sendetakterzeugung von seriellen Ein-/Ausgabe-Geräten (beispielsweise SIO U856) benutzt werden. Hierdurch ergibt sich im Mikrorechner die Möglichkeit der Programmierung verschiedener Datenübertragungsraten durch Verändern der Zeitkonstante im CTC-Kanal. Die automatische Erkennung und Wahl von Übertragungsraten wird ebenfalls ermöglicht. In Mikrorechnern mit derartig getakteten seriellen Geräten ist es zur Erreichung der üblichen und genormten Datenübertragungsraten (110 baud; 1,2 kbaud; 9,6 kbaud u. ä.) notwendig, angepaßte Systemtaktfrequenzen zu verwenden, um die auftretenden Fehler kleinzuhalten. Bei einer Übertragungsrate von 1200 baud und einem Vorteilerfaktor  $VT = 16$  ergibt sich mit einer Zeitkonstante  $TC = 128$  nach Gl. (3.4.2) eine Systemtaktfrequenz von  $f_c = 2,4576$  MHz.

$$f_c = DR \cdot VT \cdot TC; \quad (3.4.2)$$

$f_c$  Systemtaktfrequenz  
 DR Datenübertragungsrate  
 VT Vorteilerfaktor  
 TC Zeitkonstante.

Diese Systemtaktfrequenz kann auch zur Erzeugung der Übertragungsrate  $DR = 110$  baud verwendet werden ( $VT = 256$ ;  $TC = 87$ ), wenn das serielle Ein-/Ausgabe-Gerät in einer asynchronen Betriebsart arbeitet. Der auftretende Zeitgeberfehler von etwa 3‰ wird hierbei durch die Benutzung von Start- und Stopbits kompensiert.

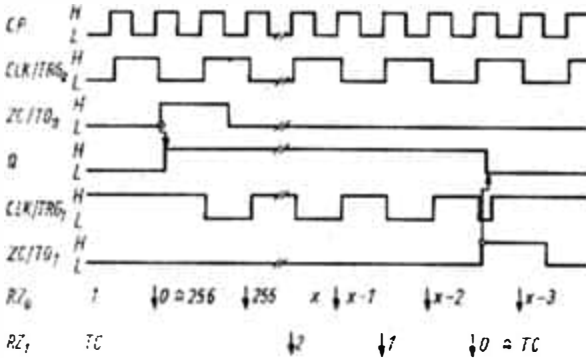
In der Zeitgebermode kann der CTC zur Kontrolle von Programmabläufen eingesetzt werden. Hierbei können durch hochpriorisierte Zeitgeberinterrupts wichtige Programmabschnitte als ISR in den Programmablauf eingeschoben werden. Typische Anwendungs-



**Bild 3.4.1.1. Wirkschaltung eines 2 x 8-bit-Digital/Analog-Wandlers mit einem CTC (Ausschnitt)**  
 1, 2 Verstärkerelemente oder Spannungs-Strom-Wandler

beispiele hierfür sind Tastatur- und Anzeigebedienroutinen, bei denen es auf eine zyklische Bedienung in bestimmten Zeitabschnitten ankommt.

Als weiterer typischer Anwendungsfall läßt sich aus zwei CTC-Kanälen und wenigen Zusatzbauelementen ein einfacher, zum Mikrorechner statisch wirkender (also ohne zyklische Softwareunterstützung arbeitender) 8-bit-Digital-Analog-Wandler (DAC) realisieren. Der DA-Wandler arbeitet nach einem integrierenden Verfahren; seine Prinzipschaltung ist im Bild 3.4.11 wiedergegeben. Hierbei wird von einem CTC-Kanal ein Referenztakt mit einer Zeitkonstante von  $TC = 256$  erzeugt. In einem weiteren Kanal wird als Zeitkonstante der binäre Wert des zu bildenden Analogwerts geladen. Im Bild 3.4.12 ist das dabei entstehende Zeitverhalten dargestellt. Die Auflösung des auf diese Art und Weise realisierten DAC beträgt 8 bit; die erreichbare Umsetzzeit liegt in der Größenordnung von etwa 20 ms.



**Bild 3.4.12. Zeitverhalten des im Bild 3.4.11 dargestellten DAC (nur Zeitverhalten eines Analogkanals dargestellt)**

- CP Systemtakt
- CLK/TRG<sub>0</sub>, ZC/TO<sub>0</sub> Takteingang und Übertragsausgang des in Zählermode betriebenen Kanals 0
- Q Signal am Ausgang der IS6/I
- CLK/TRG<sub>1</sub>, ZC/TO<sub>1</sub> Takteingang und Übertragsausgang des in Zählermode betriebenen Kanals 1
- RZ<sub>0</sub>, RZ<sub>1</sub> aktuelle Inhalte der internen Kanalrückwärtszähler des CTC ( $TC_0 = 0$ ;  $TC_1$  auszugebender Analogwert)

**Tafel 3.4.2. Grenzwerte der IS U857**

Parameter	Symbol	min.	max.	Einheit
Betriebstemperatur	$\theta_a$	0	70	°C
Lagertemperatur	$\theta_{stg}$	- 55	+ 125	°C
Betriebsspannung	$U_{CC}$	- 0,5	+ 7	V
Eingangsspannung	$U_i$	- 0,5	+ 7	V
Gehäuseverlustleistung ( $\theta_a = 25^\circ\text{C}$ )	$P_V$		0,7	W

**Tafel 3.4.3. Statische Betriebsbedingungen der IS U857**

Parameter	Symbol	min.	max.	Einheit
Betriebsspannung	$U_{CC}$	4,75	5,25	V
Eingangsspannung Low	$U_{iL}$	- 0,5	0,8	V
Eingangsspannung High	$U_{iH}$	2,0	$U_{CC}$	V
Takteingangsspannung Low	$U_{iLc}$	- 0,5	0,45	V
Takteingangsspannung High	$U_{iHc}$	$U_{CC} - 0,2$	$U_{CC}$	V

Tafel 3.4.4. Statische Kennwerte der IS U857

Parameter	Symbol	Meßbedingung	min.	max.	Einheit
Ausgangsspannung Low	$U_{OL}$	$I_O = 1,8 \text{ mA}$		0,4	V
Ausgangsspannung High	$U_{OH}$	$I_O = -100 \mu\text{A}$	2,4		V
Stromaufnahme	$I_{CC}$	$t_c = 400 \text{ ns}$	80		mA
Eingangsreststrom	$I_{LI}$	$U_I = 0 \dots 5,25 \text{ V}$	10		$\mu\text{A}$
Reststrom der Tristateausgänge im hochohmigen Zustand	$I_{LD}$ $I_{LINT}$		10		$\mu\text{A}$
Laststrom der Darlington-Treiber-Ausgänge	$I_{OHD}$	$U_{OH} = 1,5 \text{ V}$ $R_{eM} = 390 \Omega$	1,5	3,8	mA

Tafel 3.4.5. Dynamische Kennwerte der IS U857

Parameter	Symbol	min.	max.	Einheit
High-Breite des Taktes	$t_{w(CH)}$	170	2000	ns
Low-Breite des Taktes	$t_{w(CL)}$	170	2000	ns
Anstiegs- und Abfallzeiten des Taktes	$t_r, t_f$		30	ns
Datensetzzeit bis zur steigenden Flanke von C während des Schreib- oder M1-Zyklus	$t_{sc(D)}$	80		ns
Datenhaltezeit von der Anstiegsflanke von C während des Schreib- oder M1-Zyklus	$t_{HC(D)}$	0		ns
Kanalauswahlsetzzeit bis zur Anstiegsflanke von C während des Lese- oder Schreibzyklus	$t_{sc(KS)}$	140		ns
Kanalauswahlhaltezeit von der Anstiegsflanke von C während des Schreibzyklus	$t_{HC(KS)}$	$t_c + 50$		ns
Verzögerung des Datenausgangs von der Anstiegsflanke von C während des Lesezyklus	$t_{DR(D)}$		480	ns
Datenausgangsverzögerung von der fallenden Flanke von $\overline{IORQ}$ während des INTA-Zyklus	$t_{DI(D)}$		340	ns
Verzögerung bis zum Floaten des Busses von der Anstiegsflanke von $\overline{IORQ}$ oder M1 während des INTA-Zyklus	$t_{FIM(D)}$		210	ns
Verzögerung bis zum Floaten des Busses von der Anstiegsflanke von $\overline{CS}$ , $\overline{IORQ}$ oder RD während des Lesezyklus	$t_{FR(D)}$		210	ns
Verzögerung bis zum Floaten des Busses von der abfallenden Flanke von IEI während des INTA-Zyklus	$t_{FI(D)}$		230	ns
IEO-Verzögerungszeit von der fallenden Flanke von IEI	$t_{DL(IO)}$		190	ns
IEO-Verzögerungszeit von der Anstiegsflanke von IEI	$t_{DH(IO)}$		220	ns
IEO-Verzögerungszeit von der abfallenden Flanke von C während RETI	$t_{DC(IO)}$		185	ns
$\overline{INT}$ -Verzögerungszeit von der ansteigenden Flanke von C/TRG (Betriebsart: Zähler)	$t_{DCK(IT)}$		$2t_c + 210$	ns
$\overline{INT}$ -Verzögerungszeit von der Anstiegsflanke von C (Betriebsart: Zeitgeber)	$t_{DC(IT)}$		$t_c + 210$	ns

Tafel 3.4.5 (Fortsetzung)

Parameter	Symbol	min.	max.	Einheit
Für beide Betriebsarten:				
ZC/TO-Verzögerungszeit von der Anstiegsflanke von C ZC/TO High	$t_{DH(ZC)}$		200	ns
ZC/TO-Verzögerungszeit von der Anstiegsflanke von C ZC/TO Low	$t_{DL(ZC)}$		200	ns
Chipfreigabezeit bis zur Anstiegsflanke von C während des Lese- oder Schreibzyklus	$t_{sc(CE)}$	160		ns
Chipfreigabezeit von der Anstiegsflanke von C während des Schreibzyklus	$t_{HC(CE)}$	0		ns
$\overline{IORQ}$ -Setzzeit bis zur Anstiegsflanke von C während des Lese- oder Schreibzyklus	$t_{sc(IR)}$	250		ns
$\overline{IORQ}$ -Haltezeit von der Anstiegsflanke von C während des Schreibzyklus	$t_{HC(IR)}$	0		ns
$\overline{M1}$ -Setzzeit bis zur Anstiegsflanke von C während des Lese- oder Schreibzyklus	$t_{swc(M1)}$	210		ns
$\overline{M1}$ -Setzzeit bis zur Anstiegsflanke von C während des INTA oder des M1-Zyklus	$t_{src(M1)}$	115		ns
$\overline{M1}$ -Haltezeit von der Anstiegsflanke von C	$t_{HC(M1)}$	0		ns
$\overline{RD}$ -Setzzeit bis zur Anstiegsflanke von C während des Schreib- oder INTA-Zyklus	$t_{swc(RD)}$	115		ns
$\overline{RD}$ -Haltezeit von der Anstiegsflanke von C während des INTA-Zyklus	$t_{HC(RD)}$	0		ns
$\overline{RD}$ -Setzzeit bis zur Anstiegsflanke von C während des Lese- oder M1-Zyklus	$t_{src(RD)}$	240		ns
$\overline{RD}$ -Haltezeit von der Anstiegsflanke von C während des Schreibzyklus	$t_{HWC(RD)}$	0		ns
$\overline{RD}$ -Haltezeit von der Anstiegsflanke von C während des M1-Zyklus	$t_{HMC(RD)}$	0		ns
Taktperiode (Betriebsart: Zähler)	$t_{C(CK)}$	$2t_C$		
Taktsetzzeit bis zur Anstiegsflanke von C für einen anliegenden Zählimpuls (Betriebsart: Zähler)	$t_{s(CK)}$	210		ns
Triggersetzzeit bis zur Anstiegsflanke von C für die Freigabe des Vorteilers auf den zweiten folgenden C (Betriebsart: Zeitgeber)	$t_{s(TR)}$	210		ns
Takt- und Trigger-Anstiegs- und Abfallzeiten (in beiden Betriebsarten)	$t_r, t_f$		50	ns
High-Pulsbreite von Takt- und Triggersignal	$t_{w(CTH)}$	200		ns
Low-Pulsbreite von Takt- und Triggersignal	$t_{w(CTL)}$	200		ns

Tafel 3.4.6. Pinkapazitäten der IS U857

$f_a = 25^\circ\text{C}$

$\vartheta = 0,5 \dots 2 \text{ MHz}$

Parameter	Symbol	max.	Einheit
Taktkapazität	$C_0$	25	pF
Eingangskapazität	$C_I$	7	pF
Ausgangskapazität	$C_O$	14	pF

In der Zählerbetriebsart kann ein CTC-Kanal als Ereigniszähler betrieben werden. Die hierbei erreichbare Zählfrequenz entspricht der halben Systemtaktfrequenz. Mit den Kanalein- bzw. -ausgängen CLK/TRG und ZC/TO können mehrere Zählerkanäle kaskadiert werden, um größere Zählerumfänge zu realisieren.

Gleichermaßen können einem Zeitgeberkanal Zählerkanäle nachgeschaltet werden, um größere Zeitintervalle am letzten Zähler dieser Kette zu erzeugen.

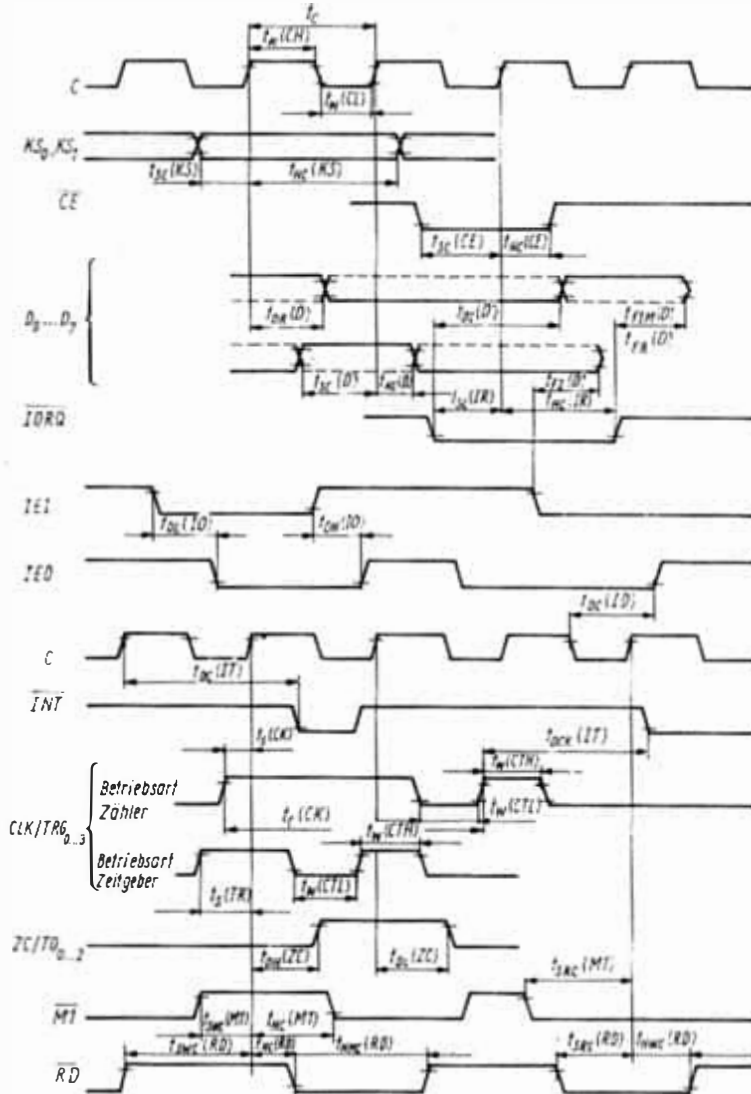


Bild 3.4.13. Darstellung der dynamischen Kennwerte des CTC im Zeitdiagramm

Im Zählerbetrieb arbeitende CTC-Kanäle können auch benutzt werden, um durch einzelne Signalleitungen Interruptanmeldungen zur CPU auszulösen. Diese Möglichkeit kann dann angewendet werden, wenn das komfortable Interruptsystem des Prozessors zu schnellen und leistungsfähigen Reaktionen auf einzelne Signale genutzt werden soll und der Einsatz einer PIO-IS U855 nicht lohnt bzw. im Mikrorechner unbenutzte CTC-

Kanäle hierzu verwendet werden können. Der Kanal kann im Zählermode mit der Zeitkonstante  $TC = 1$  arbeiten, die den Interrupt auslösende Triggerflanke am CLK/TRG-Eingang ist programmierbar.

### 3.4.8. Zusammenfassung der technischen Daten

In diesem Abschnitt erfolgt die Spezifikation der elektrischen Kennwerte vom Zähler/Zeitgeber U857.

In Tafel 3.4.2 sind die Grenzwerte der IS U857 aufgeführt. Tafel 3.4.3 enthält die statischen Betriebsbedingungen, Tafel 3.4.4 die statischen Kennwerte. Diese Kenngrößen sind für den Umgebungstemperaturbereich  $\vartheta_a = 0 \dots 70^\circ\text{C}$  festgelegt.

Die Tafel 3.4.5 faßt die dynamischen Kennwerte des CTC zusammen. Sie beziehen sich auf die Meßbedingungen:

$$\vartheta_a = 70^\circ\text{C}$$

$$U_{CC} = 4,75 \text{ V}$$

$$U_{IL} = 0,8 \text{ V}$$

$$U_{IH} = 2,0 \text{ V}$$

$$U_{ILC} = 0,45 \text{ V (Taktingang)}$$

$$U_{IHC} = 4,55 \text{ V (Taktingang)}.$$

Bild 3.4.13 enthält die zugehörige Darstellung des Zeitverhaltens.

In Tafel 3.4.6 sind die Pinkapazitäten der IS U857 aufgeführt.

## 4. Eigenschaften des Prozessorsystems U880

### 4.1. Leistungsbestimmende Parameter

Die Leistungsfähigkeit eines Mikrorechners wird vorrangig durch die Systemeigenschaften des Prozessors und der peripheren Elemente bestimmt. Diese Faktoren bestimmen somit die Haupteinsatzgebiete sowie die Grenzen der Anwendbarkeit von Mikrorechnersystemen. Dabei werden in den grundlegenden Einsatzbereichen, insbesondere also der Datenverarbeitungstechnik und der Steuer- und Regelungstechnik, unterschiedliche und verschieden gerichtete Forderungen an Mikrorechnerkonzepte gestellt.

Für den Mikrorechnereinsatz in der Datentechnik ergeben sich hauptsächlich Anforderungen an einen zur Realisierung von arithmetischen und logischen Funktionen ausgelegten Befehlssatz und an die Ausführungszeiten dieser Befehle. Des weiteren wird ein relativ großer Adressierungsraum für externe Speicherelemente (ROM, RAM) benötigt. Der Einsatz großer Schreib-/Lese-Speicherbereiche, die vorteilhaft mit dynamischen RAM-Bauelementen aufgebaut werden, wird durch eine prozessorseitig vorgenommene automatische Refreshgenerierung begünstigt. Leistungsfähige periphere Systemelemente werden zur Gestaltung von standardisierten parallelen und seriellen Ein-/Ausgabe-Einrichtungen benötigt.

In der Steuer- und Regelungstechnik bestehen dagegen vor allem Forderungen zur Realisierung vielfältiger Ein-/Ausgabe-Einrichtungen und nach möglichst kurzen Programmbearbeitungszeiten. Signaländerungen in der Peripherie erfordern eine möglichst schnelle Reaktion des Rechners. Weitere Gesichtspunkte beim Mikrorechnereinsatz sind die Möglichkeit des Echtzeitbetriebs, die Schaffung von Datenbahnkopplungen zu hierarchisch angeordneten Rechnern und die Anwendbarkeit eines leistungsfähigen Interruptsystems.

Der Mikroprozessor U880 und die zugehörigen peripheren Elemente PIO, SIO, CTC erfüllen in weiten Grenzen die Forderungen dieser Haupteinsatzgebiete. Mit der maximalen Systemtaktfrequenz von  $f_c = 2,5$  MHz ergibt sich eine typische Befehlsbearbeitungszeit von  $t = 1,6 \mu\text{s}$ , bezogen auf die Ausführung eines hierzu üblicherweise betrachteten 8-bit-Additionsbefehls. Der Prozessor besitzt 158 Basisbefehle. Aus diesem Basisbefehlssatz können durch Spezifizierung der CPU-Register 695 verschiedene Befehle (Op-Kode) gebildet werden. Die Leistungsfähigkeit dieser Befehle ist beispielsweise gekennzeichnet durch die Adressierungsmöglichkeiten, das Flagsystem, die Unterprogrammabarbeitung, die Anwendung von Einzelbit-, Nibble- (4-bit-), Byte- (8-bit-) und Wort- (16-bit-) Verarbeitung und die Anzahl der arithmetischen und logischen Befehle. Der Befehlssatz beinhaltet auch sich automatisch wiederholende (repetierende) Befehle, die zur Übertragung von großen Datenblöcken vorteilhaft angewendet werden können. Der Tauschregistersatz der CPU kann in Interruptbedienroutinen zur Rettung der aktuellen Registerinhalte benutzt werden und ermöglicht damit auch die schnelle Reaktion auf Interruptanmeldungen.

Die Eigenschaften der peripheren Elemente des Mikrorechnersystems bestimmen weitgehend die Möglichkeiten der Anpassung des Rechners an äußere Signale. Sie haben deshalb besondere Bedeutung für Einsatzfälle in der Steuer- und Regelungstechnik. Die in

verschiedenen Betriebsarten und Einsatzbedingungen programmierbaren Elemente parallele Ein-/Ausgabe-Einheit, serielle Ein-/Ausgabe-Einheit und Zähler/Zeitgeber gestatten aufgrund ihrer Komplexität die Organisation eines den Mikroprozessor zeitlich nur sehr gering belastenden Datenverkehrs mit der Peripherie.

Als wesentliche Systemeigenschaften, die die Reaktionszeit und Leistungsfähigkeit des Mikrorechnersystems bestimmen, sollen in den nachfolgenden Abschnitten die Interruptbearbeitung und das Verhalten bei direktem Speicherzugriff des Systems U880 dargestellt werden.

## 4.2. Beschreibung des U880-Interruptsystems

### 4.2.1. Einführung

Die Aufgabe eines Interruptsystems besteht darin, auf vorliegende Interruptanmeldungen aus der Peripherie durch Einschleichen einer zugehörigen Programmfolge (ISR) in das laufende Programm zu reagieren. Die Interruptbearbeitung von peripheren Geräten stellt die Alternative zur programmgestützten Peripheriebearbeitung durch zyklische Abfrage (Polling) dar. Sie ermöglicht eine Bearbeitung auf Anforderung und belastet den Prozessor zeitlich geringer. Die Einsatzmöglichkeiten werden durch die Länge des einzuschleibenden Unterprogramms und durch Art und Anzahl der interruptfähigen Peripherieelemente bestimmt. Damit der Prozessor möglichst schnell auf eine Interruptanforderung aus der Peripherie reagieren kann, besitzt die CPU unterschiedlich priorisierte Interrupteingänge. Es ist die Anmeldung von nichtmaskierbaren Interrupts (NMI) und von maskierbaren Interrupts (INT) möglich. Ein NMI unterbricht grundsätzlich jede Programmabarbeitung, während ein INT softwaremäßig gesperrt werden kann und somit für die Dauer der Sperrung unwirksam bleibt. Die CPU hat die Möglichkeit der Auswertung eines Interruptvektors, der vom peripheren Systemelement im Interruptquittierungszyklus erzeugt wird und mit dessen Hilfe die Startadresse der ISR gebildet werden kann. Hierdurch kann im Programmablauf auf die Abfrage aller Peripherieelemente nach der Quelle der Interruptauslösung verzichtet werden. Die Auswahl des zur Anmeldung kommenden Interrupts bei mehreren Anforderungen durch verschiedene Geräte erfolgt in der Peripherie durch Bildung einer *Interruptprioritätenkette*. Diese Prioritätenkette wird hardwaremäßig durch Kaskadierung der interruptfähigen Elemente gebildet. Der Prozessor wird also auch bei der Festlegung der Interruptwertigkeit nicht zusätzlich belastet. Die Leistungsfähigkeit des Interruptbetriebs wird außerdem durch komfortable CPU-Befehle verbessert, die den Tausch der Registersätze bzw. Registerrettungsoperationen in den Stackbereich des Hauptspeichers bewirken. Somit kann nach Ablauf der eingeschobenen ISR der alte Programmbearbeitungszustand wiederhergestellt werden.

### 4.2.2. Möglichkeiten der Interruptbearbeitung

#### 4.2.2.1. Einordnung im System

Der Prozessor U880 reagiert auf externe Anforderungen zur Unterbrechung der laufenden Programmfolge mit der nachfolgend dargestellten Priorität:

1. Busanforderung durch  $\overline{\text{BUSRQ}}$
2. Interruptanmeldung durch  $\overline{\text{NMI}}$
3. Interruptanmeldung durch  $\overline{\text{INT}}$ .

Die Anforderung des Prozessorbusses durch das Steuersignal  $\overline{\text{BUSRQ}}$  dient zur Ausführung einer DMA-Operation (s. Abschn. 4.3.). Dieser direkte Speicherzugriff kann nach jedem abgearbeiteten Maschinenzyklus in die Programmbearbeitung eingeschoben werden, wenn das interne BUSRQ-Flipflop zuvor gesetzt wurde. Der Zustand dieses Flipflops wird in Abhängigkeit vom Signal am  $\overline{\text{BUSRQ}}$ -Eingang der CPU während der steigenden Flanke des letzten Taktes eines jeden Maschinenzyklus ermittelt. Im Gegensatz hierzu erfolgt die Auswertung der Zustände der Interruptflipflops (NMI-Flipflop, INT-Flipflop) nur am Ende des letzten Maschinenzyklus eines vollständig abgearbeiteten Befehls. Falls zu diesem Zeitpunkt ebenfalls eine Busanforderung vorliegt, wird diese vorrangig abgearbeitet.

Das Setzen des NMI-Flipflops erfolgt zustandsgesteuert durch das Aktivwerden des entsprechenden Signals am NMI-Eingang des Prozessors. Zum Anmelden eines nicht-maskierbaren Interrupts ist somit nur ein Impuls mit einer bestimmten Impulsbreite als NMI-Signal notwendig. Der Zustand des INT-Flipflops wird dagegen durch Auswertung des INT-Eingangs mit der letzten steigenden Systemtaktflanke des letzten Maschinenzyklus jedes Befehls bestimmt, da die Interruptanmeldung nochmals durch die peripheren Systemelemente zwischengespeichert wird. Die CPU-interne Auswertung der Interruptflipflops erfolgt ebenfalls nur zu diesem Zeitpunkt und bewirkt bei einer erkannten Anmeldung das Einschleusen der entsprechenden Interruptquittierungszyklen. Die Maskierung des INT-Eingangs erfolgt durch die beiden CPU-internen Interruptfreigabeflipflops IFF1 und IFF2. Das Freigabeflipflop IFF1 wirkt hierbei als Maskierungsflipflop, und Flipflop IFF2 dient zur Protokollierung des Zustands von IFF1 in NMI-Serviceroutinen. Beide Flipflops können softwaremäßig durch die CPU-Befehle EI und DI gesetzt bzw. rückgesetzt werden.

#### 4.2.2.2. Nichtmaskierbarer Interrupt

Die Ausführung einer NMI-Anmeldung kann programmäßig nicht verhindert werden. Die laufende Programmabarbeitung wird nach Setzen des internen NMI-Flipflops sofort im nächsten Befehl durch die Quittierung des Interrupts unterbrochen, vorausgesetzt, es lag keine BUSRQ-Anmeldung vor. Die Anwendung dieser höchstwertigen Interruptanmeldung ist deshalb für Rechnerfunktionen vorbehalten, die eine sehr schnelle Reaktion des Prozessors erfordern. Beispiele hierfür sind das Reagieren auf bevorstehende Ausfälle der Rechnerversorgungsspannung und die Bearbeitung von vorrangigen Informationen aus der Peripherie (z. B. Alarmsignale).

Nach Annahme einer NMI-Anmeldung schiebt der Prozessor einen NMI-Quittierungszyklus in die Befehlsabarbeitung ein. Das entsprechende Zeitverhalten an der CPU ist im Bild 4.2.1 dargestellt. Der Prozessor rettet in diesem Quittierungszyklus den aktuellen Programmzählerstand (PC), verhindert automatisch die Anmeldung von maskierbaren Interrupts und setzt den Programmzähler auf die Restartadresse 0066H.

Im M1-Zyklus der NMI-Quittierung erfolgt auf dem Adreßbus die Ausgabe des letzten, aktuellen Programmzählerstands. Der dadurch im Hauptspeicher adressierte Befehlscode wird aber nicht mehr durch den Datenbus des Prozessors übernommen. Der Befehlsholezyklus (op-code fetch) wird somit von der CPU ignoriert. Anstelle dieses nicht bearbeiteten Befehls wird prozessorintern ein Restartbefehl zur Programmspeicherzelle 0066H ausgeführt. In den beiden nachfolgenden Maschinenzyklen (Speicherschreibzyklen M2, M3) erfolgt deshalb die Auslagerung des zuvor ausgesendeten aktuellen Programmzählerstands (PC) in den Stackbereich des Hauptspeichers. Als neuer PC-Stand wird die Restartadresse 0066H gebildet, die im Befehlsholezyklus des nachfolgenden

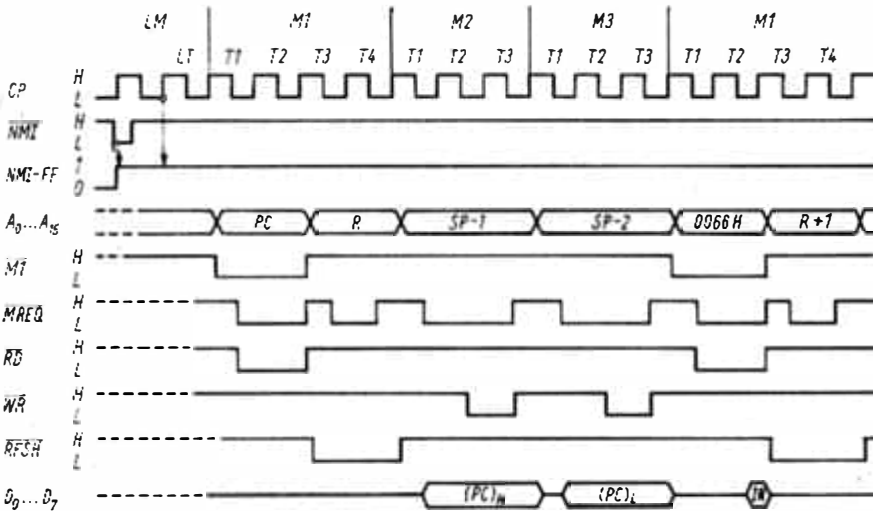


Bild 4.2.1. Zeitverhalten der CPU im NMI-Quittierungszyklus

- LM, LT     letzter Maschinenzklus, letzter Taktzustand des vorangegangenen Befehls
- M1, M2, M3   Maschinenzyklen der CPU
- CP        Systemtakt (T1, T2, T3, T4 Taktzustände der CPU)
- NMI       NMI-Anmeldeeingang der CPU
- NMI-FF     internes Anmeldeflipflop der CPU
- A<sub>0</sub> ... A<sub>15</sub>   Belegung des CPU-Adreßbusses (PC aktueller Programmzählerstand; R Inhalt des Refreshregisters; SP Stackpointerstand; 66H Startadresse der NMI-Bedienroutine)
- M1, MREQ, RD, WR, RFSH   CPU-Steuerausgänge
- D<sub>0</sub> ... D<sub>7</sub>    Belegung des Systemdatenbusses an der CPU ((PC)<sub>H</sub>, (PC)<sub>L</sub> höher- bzw. niederwertiger Teil des PC; IN gültige Op-Kode-Daten)

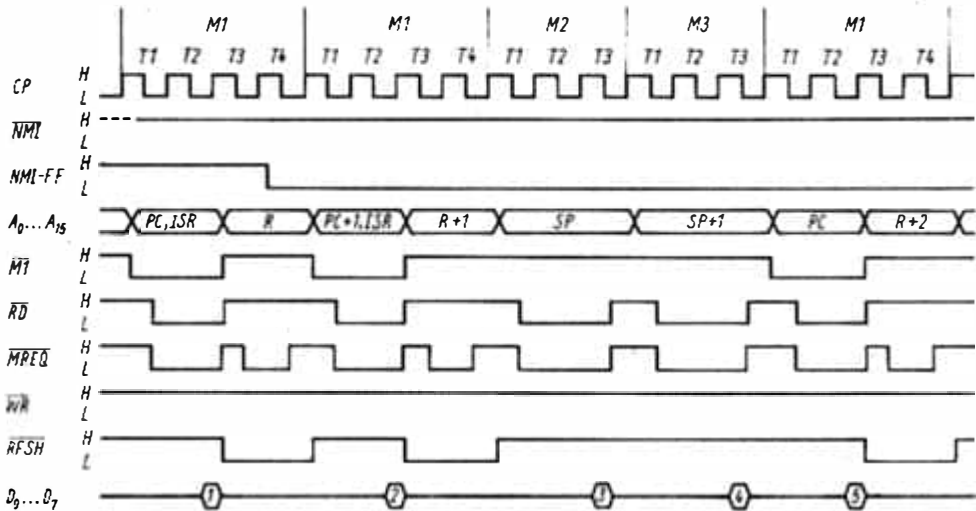
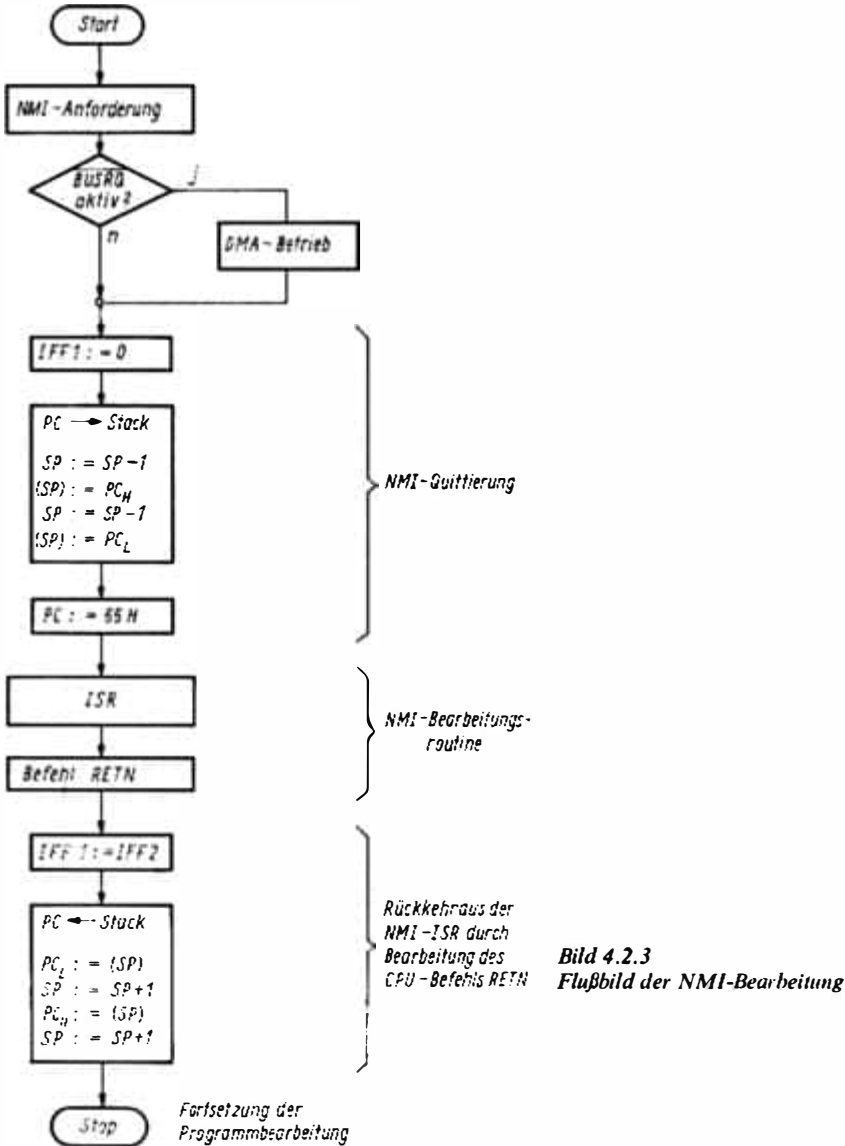


Bild 4.2.2. Zeitverhalten der CPU bei Befehlsabarbeitung RETN

- CP        Systemtakt (T1, T2, T3, T4 Taktzustände der CPU)
- NMI       NMI-Anmeldeeingang der CPU
- NMI-FF     internes Anmeldeflipflop der CPU
- A<sub>0</sub> ... A<sub>15</sub>   Belegung des CPU-Adreßbusses (PC, ISR Programmzählerstand in der Bearbeitungsroutine; R Inhalt des Refreshregisters; SP Inhalt des Stackpointers; PC neuer Programmzählerstand aus Stack)
- M1, MREQ, RD, WR, RFSH   Steuerausgänge der CPU
- D<sub>0</sub> ... D<sub>7</sub>    Belegung des Systemdatenbusses an der CPU  
 (1, 2 Übernahme der RETN-Befehlsbytes; 3 Einlesen des PC, niederwertiges Byte; 4 Einlesen des PC, höherwertiges Byte; 5 Übernahme des Opkodes des folgenden Befehls)

Befehls auf dem Adreßbus ausgegeben wird. Gleichzeitig erfolgt im NMI-Quittierungszyklus die Rücksetzung des CPU-internen Interruptfreigabeflupps IFF1, um die Anmeldung und Ausführung von maskierbaren Interrupts zu verhindern. Der Zustand des Freigabeflupps IFF2 wird hierbei nicht verändert. Dieses Flipflop dient zur Speicherung des Zustands, den IFF1 vor der NMI-Anmeldung einnahm. Der Zustand des Freigabeflupps IFF2 kann softwaremäßig durch Ausführung entweder des Befehls LD A, I oder des Befehls LD A, R ermittelt werden. Nach Abarbeitung eines dieser Befehle beinhaltet der Wert des Paritätsflags (Bit F<sub>2</sub>) den Zustand von IFF2.



Die Rückkehr aus der NMI-Bearbeitungsroutine (ISR) in die unterbrochene Programmbearbeitung erfolgt durch den Befehl RETN. Das CPU-Zeitverhalten bei der Ausführung dieses Befehls ist im Bild 4.2.2 dargestellt. Der Prozessor stellt bei der RETN-Befehl

fehlsabarbeitung in Hinsicht auf die Inhalte des Programmzählers und des Stackpointers sowie auf den Interruptbearbeitungszustand der CPU den Programmbearbeitungszustand wieder her, der vor Anmeldung und Ausführung des nichtmaskierbaren Interrupts vorlag. Hierzu speichert er den vor der NMI-Anmeldung aktuellen Programmzählerstand aus dem Stackbereich des Hauptspeichers in die CPU zurück. Durch zweimaliges Inkrementieren des Stackpointers wird ebenfalls der ursprüngliche Wert des SP erreicht. Der PC-Stand in der NMI-Serviceroutine geht verloren. Der Inhalt des Interruptfreigabeflipflops IFF2 wird in das Flipflop IFF1 kopiert. Da das Freigabeflipflop IFF2 durch die Ausführung der NMI-Quittierung und -Bearbeitung nicht automatisch verändert wird, ergibt sich der Interruptbearbeitungszustand, der in der unterbrochenen Programmfolge aktuell war. Voraussetzung hierfür ist jedoch, daß in der NMI-Serviceroutine kein EI- bzw. DI-Befehl erfolgte, da diese Befehle beide Freigabeflipflops beeinflussen. Zur erneuten Freigabe der NMI-Anmeldeleitung erfolgt außerdem die Rücksetzung des CPU-internen NMI-Flipflops.

Die NMI-Bearbeitung ist zusammenfassend im Bild 4.2.3. als Flußbild dargestellt.

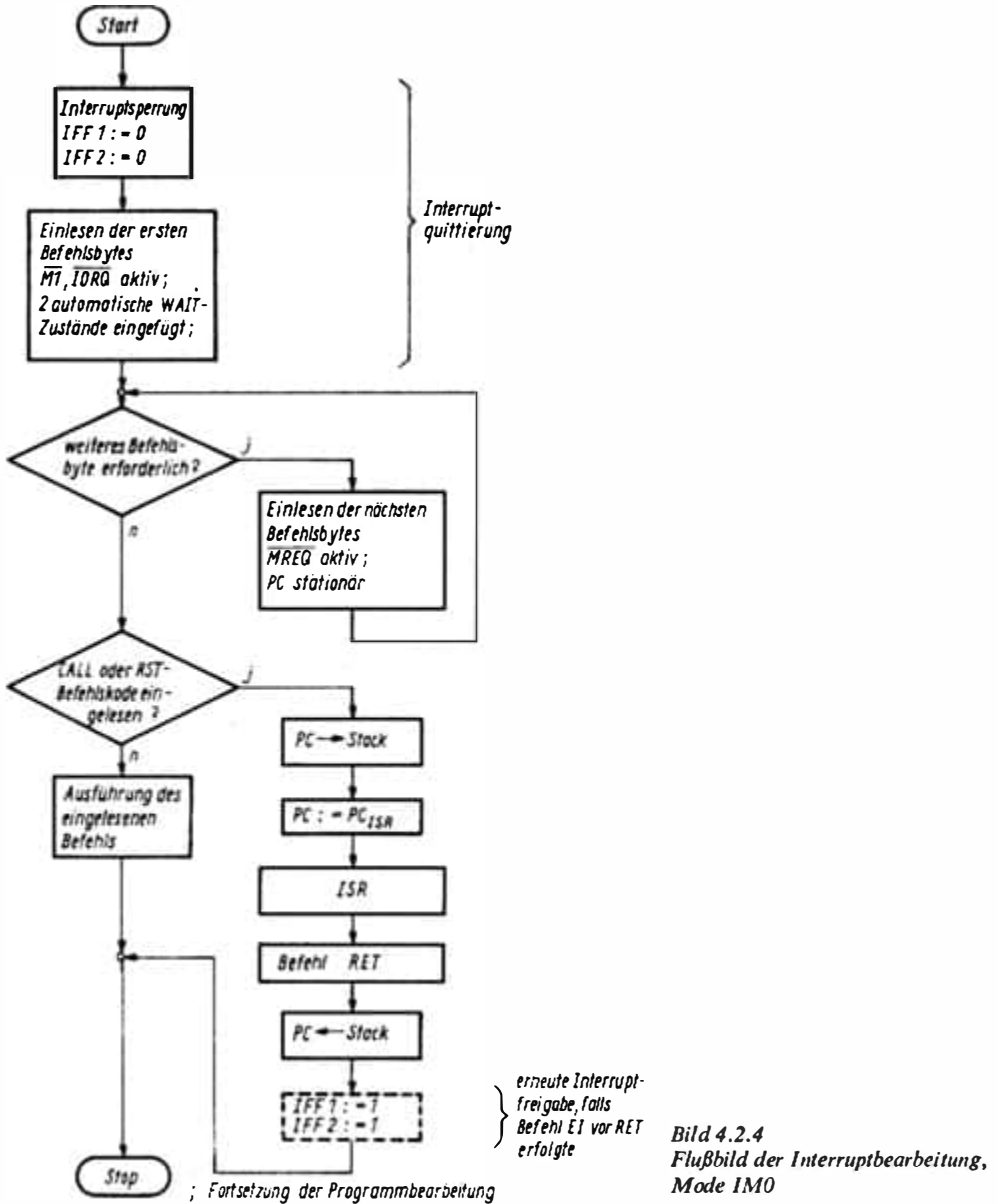
#### 4.2.2.3. Maskierbarer Interrupt

Die Ausführung einer durch einen maskierbaren Interrupt (INT) angeforderten Programmunterbrechung kann programmäßig gesperrt werden. Diese Maskierung geschieht durch Beeinflussung der CPU-internen Interruptfreigabeflipflops IFF1 und IFF2. Beide Freigabeflipflops werden durch den Befehl EI gesetzt und erlauben somit die Ausführung der am  $\overline{\text{INT}}$ -Eingang der Prozessors anstehenden Interruptanmeldung. Das Rücksetzen der Flipflops kann softwaremäßig durch den Befehl DI erfolgen. Während der Abarbeitung dieser beiden Befehle wird prozessorintern die Interruptfreigabe gesperrt. Das bedeutet, daß nach Ausführung dieser Befehle kein Interruptquittierungszyklus in die Programmfolge eingeschoben werden kann. Der Sinn dieser Maßnahme besteht darin, daß nach Interruptfreigabe (Befehl EI) am Ende eines Unterprogramms (z. B. ISR) noch der nachfolgende Rücksprungbefehl (RET, RETI oder RETN) zum höheren Programm-niveau ausgeführt wird. Eine etwaige Interruptanmeldung wird dann erst in diesem Programmteil wirksam. Dadurch werden Verschachtelungen der Interruptbedienroutinen vermieden und der Stackbereich des Speichers weniger belastet.

Die Programmunterbrechung erfolgt ansonsten bei Interruptfreigabe sofort nach dem Befehl, an dessen Ende die Interruptanmeldung vom Prozessor erkannt wurde. Die CPU reagiert daraufhin durch Einschleusen eines Interruptquittierungszyklus. Voraussetzung hierfür ist, daß keine Bus- oder NMI-Anforderung vorliegt, die sonst zuvor bearbeitet werden. Der Prozessor U880 kann in drei Interruptbetriebsarten (IM0, IM1, IM2) arbeiten, die sich im wesentlichen durch unterschiedliche Wirkung und Leistungsfähigkeit unterscheiden.

Die Anwendung des INT-Interruptsystems ist vor allem für die schnelle und leistungsfähige Bedienung der peripheren Systemelemente vorgesehen. Die Systemelemente (PIO, SIO, CTC) sind hardwaremäßig für die Interruptbearbeitung zugeschnitten. Sie beinhalten eine automatische Interruptprioritätenauswahl und erzeugen für die Bearbeitung in der leistungsfähigsten Interruptmode (IM2) einen Interruptvektor. Für die Interruptverarbeitung von Mikrorechnern, die ohne die peripheren Systemelemente realisiert werden, besitzt der Prozessor U880 zwei weitere Interruptbetriebsarten (IM0, IM1). Diese Interruptmoden sind insbesondere für die Verwendung spezieller Interruptschaltkreise (interrupt controller) bzw. für die Anwendung in kleinen, sehr speziellen Mikrorechnern vorgesehen.

Die Annahme einer Interruptanforderung über die INT-Linie erfolgt durch den Prozessor, wenn das interne Interruptfreigabeflipflop IFF1 gesetzt ist und wenn an der CPU keine NMI- oder Busanforderung ansteht. Die Bearbeitung eines DMA-Betriebs oder einer nichtmaskierbaren Interruptanmeldung wird ansonsten zuvor durchgeführt. Der



Prozessor U880 unterbricht nach dem Erkennen der Interruptanmeldung unter den genannten Bedingungen sofort die Programmbearbeitung und schiebt einen Interruptquittierungszyklus in die Befehlsfolge ein. Hierbei erfolgt automatisch das Rücksetzen der CPU-Freigabeflipflops (IFF1 und IFF2) und somit die Sperrung weiterer Interruptforderungen der Peripherie. Der erste Maschinenzklus der Interruptquittierung ist durch

das gleichzeitige Auftreten aktiver  $\overline{M1}$ - und  $\overline{IORQ}$ -Signale gekennzeichnet und wird so von dem üblichen CPU-Maschinenzyklus der Befehlsbearbeitung unterschieden. Die peripheren Elemente können somit die Interruptquittierung erkennen und entsprechend reagieren (die Systemelemente setzen das interne Interruptbearbeitungsflipflop und senden einen Interruptvektor aus). Die CPU fügt in diesen Maschinenzyklus der Quittierung automatisch zwei WAIT-Zustände ein, die ein verzögertes Aussenden des  $\overline{IORQ}$ -Signals bewirken. Hierdurch wird erreicht, daß in der Peripherie eine hardwaremäßig realisierte Interruptprioritätenkette bei Auftreten des Signals  $\overline{IORQ}$  einen eingeschwungenen Zustand erreicht. In Abhängigkeit vom Auftreten dieses Signals im Interruptquittierungszyklus erfolgt vom ausgewählten, priorisierten Systemelement das Belegen des Datenbusses mit dem Interruptvektor. Bei anders realisierten Peripherieelementen kann in Abhängigkeit vom  $\overline{IORQ}$ -Signal ebenfalls die Übernahme des Datenbusses vorgenommen werden. Die Voraussetzung für die Belegung des Datenbusses ist aber, daß durch alle peripheren Elemente bei aktivem  $\overline{M1}$ -Signal der Interruptanmeldungsstatus nicht verändert werden darf. Die weitere Reaktion des Prozessors U880 in diesem Zyklus ist nun von der vorgewählten Interruptbetriebsart abhängig. Diese Interruptmoden können softwaremäßig durch die Befehle IM0, IM1 und IM2 gewählt werden. Nach Anlegen eines  $\overline{RESET}$ -Signals an die CPU nimmt die IS U880 automatisch die Betriebsart IM0 ein. Das Einstellen der Interruptbetriebsart erfolgt üblicherweise im Initialisierungsteil des Programms. Aufgrund der unterschiedlichen CPU-Reaktion und der spezifischen Peripherie bei den verschiedenen Interruptmoden kann i. allg. die Interruptbetriebsart im Programmablauf nicht geändert werden.

Die Interruptbetriebsart IM0 der CPU U880 dient zur Ausführung eines Befehlskodes, den das interruptanmeldende periphere Element im Interruptquittierungszyklus auf den Datenbus legt. Das erste Byte (op-code fetch) der Instruktion wird hierbei in dem durch aktive  $\overline{M1}$ - und  $\overline{IORQ}$ -Signale gekennzeichneten Maschinenzyklus der Interruptquittierung vom Datenbus übernommen. Erfordert die Befehlsausführung weitere Befehls- bzw. Datenzyklen, werden diese als gewöhnliche Maschinenzyklen (memory read, memory write, input, output) ausgeführt. Sie haben jedoch die Besonderheit, daß der Befehlszählerstand (PC) jeweils nicht erhöht wird. Der PC-Stand vor der Interruptanmeldung bleibt somit erhalten. Da dieser ungültige Programmzählerstand auf dem Adreßbus ausgegeben wird, muß hardwaremäßig abgesichert werden, daß keine Fehlreaktionen im Hauptspeicher bzw. in der Peripherie ausgelöst werden. Bei Anwendung dieser Interruptbetriebsart bietet sich der RST-Befehl (restart) zur Ausführung besonders an, da er ein sehr leistungsfähiger Einbytebefehl ist. Bei hardwaremäßiger Realisierung der Datenbusübernahme im Interruptquittierungszyklus bleibt bei Verwendung dieses Befehls der Aufwand in Grenzen. Zudem können durch Spezifizierung der Bits  $D_3$ ,  $D_4$  und  $D_5$  des Befehlskodes acht unterschiedliche Unterprogrammstartadressen (0H, 8H, 10H, 18H, 20H, 28H, 30H, 38H) aufgerufen werden. Die Interruptbearbeitung in dieser Mode (IM0) ist zusammenfassend im Bild 4.2.4 dargestellt. Sie wirkt ähnlich wie die Interruptbearbeitung des Prozessors U808 (s. Abschn. 2.1.). Diese Interruptbetriebsart ermöglicht deshalb die Nutzung evtl. vorhandener peripherer Einheiten des Systems U808.

Die Interruptbetriebsart IM1 des Prozessors U880 realisiert eine einfache, ohne periphere Unterstützung wirkende Interruptbearbeitung durch Ausführen eines Programmrücksprungs zur Speicherplatzadresse 0038H. Hierbei wird automatisch der aktuelle Programmzählerstand durch Laden in den Stackbereich des Speichers gerettet. Im Bild 4.2.5 ist das Zeitverhalten des Interruptquittierungszyklus in dieser Interruptbetriebsart dargestellt. Der Prozessor reagiert in der Mode IM1 ähnlich wie bei der Bearbeitung

einer NMI-Anforderung. Der Programmzähler wird jedoch auf den Wert PC = 38H gesetzt. Die Anwendung dieser Interruptbetriebsart ist deshalb in Mikrorechnern denkbar, die ohne periphere Systemelemente realisiert werden und die nur zwei Interruptebenen benötigen. Als erste Ebene könnte dann der nichtmaskierbare Interrupt verwendet werden, während der in der Mode IM1 betriebene maskierbare Interrupt die zweite Interruptebene darstellt. Die Interruptbearbeitung in der Betriebsart IM1 ist zusammenfassend im Bild 4.2.6 dargestellt.

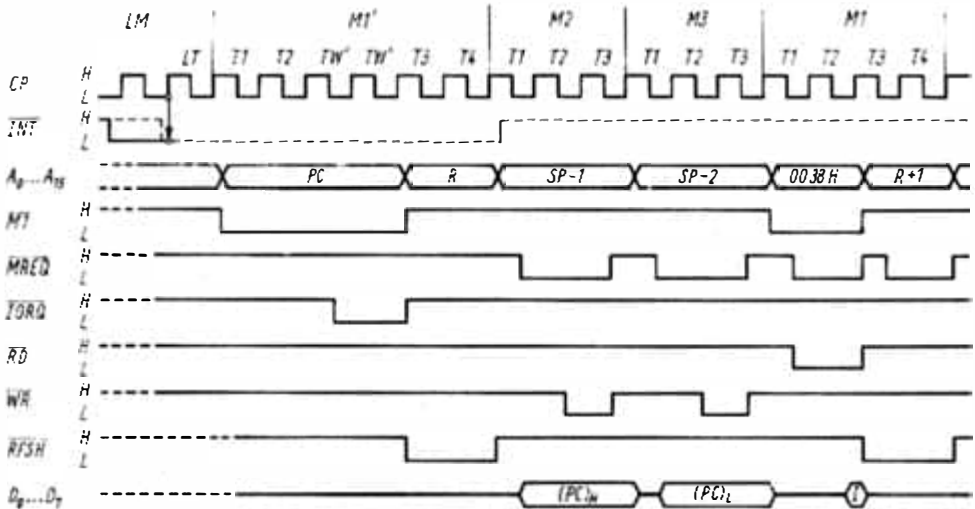


Bild 4.2.5. Zeitverhalten der CPU im Interruptquittierungszyklus der Mode IM1

- LM, LT      letzter Maschinenzyklus, letzter Taktzustand des vorangegangenen Befehls
- M1', M2, M3    Maschinenzyklen der Quittierung
- CP          Systemtakt (T1, T2, TW', T3, T4 Taktzustände der CPU; TW' automatisch von der CPU eingefügte WAIT-Zustände)
- INT        INT-Anmeldeeingang der CPU
- A0 ... A15    Belegung des CPU-Adreßbusses (PC Programmzählerstand; R Inhalt des Refreshregisters; SP Stackpointerstand; 38H Startadresse der ISR)
- MI, MREQ, IORQ, RD, WR, RFSH    Steuersignalausgänge der CPU
- D0 ... D7    Belegung des Systemdatenbusses an der CPU [(PC)<sub>H</sub>, (PC)<sub>L</sub> höher- bzw. niederwertiger Teil des PC; I gültige Op-Kode-Daten]

Die Interruptbetriebsart IM2 der CPU U880 ist die leistungsfähigste der drei genannten Moden. Sie führt einen indirekt adressierten CALL-Befehl zu einer beliebigen Programmspeicherzelle im Adressierungsraum aus. Der zur indirekten Adressierung benötigte 16-bit-Pointer wird hierbei aus dem I-Register der CPU (H-Byte) und einem Interruptvektor (L-Byte) gebildet. Dieser Interruptvektor wird im M1-Maschinenzyklus der Interruptquittierung vom Prozessor gelesen. Er muß deshalb zu diesem Zeitpunkt von dem priorisierten, interruptanmeldenden peripheren Gerät auf den Systemdatenbus gelegt werden. Das entsprechende Zeitverhalten ist im Bild 4.2.7 dargestellt. Die Leistungsfähigkeit dieser Interruptmode ist vor allem dadurch gegeben, daß die peripheren Systemelemente U855 (PIO), U856 (SIO) und U857 (CTC) den angeforderten Interruptvektor automatisch bei der Quittierung ihrer Interruptanmeldungen aussenden. Somit ist hierfür kein zusätzlicher Hardwareaufwand notwendig. Des weiteren gestattet die Interruptmode IM2 die Zuordnung von Interruptbearbeitungsroutinen (ISR) zu den einzelnen interruptfähigen peripheren Systemelementen. Wegen der Spezifizierung der Interruptvektoren durch die Peripherie können in einem Rechnersystem bei unverändertem I-Register der CPU bis zu 128 verschiedene ISR aufgerufen werden. Hierdurch wird Pro-

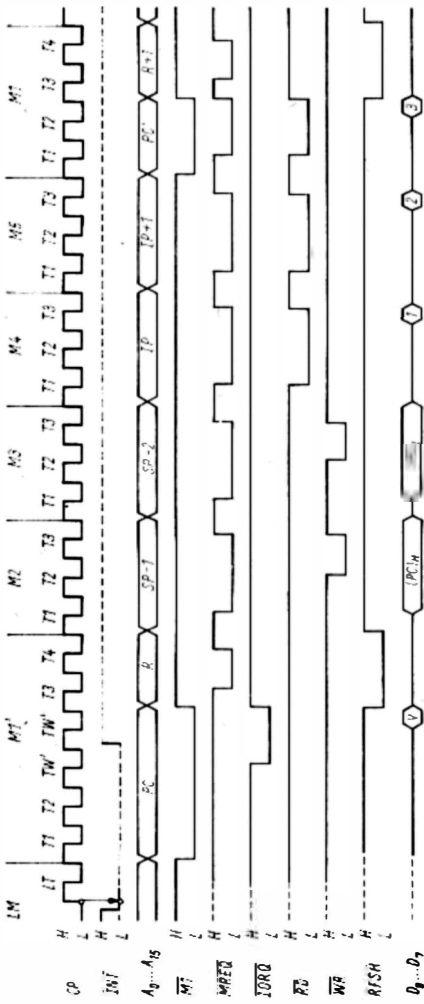


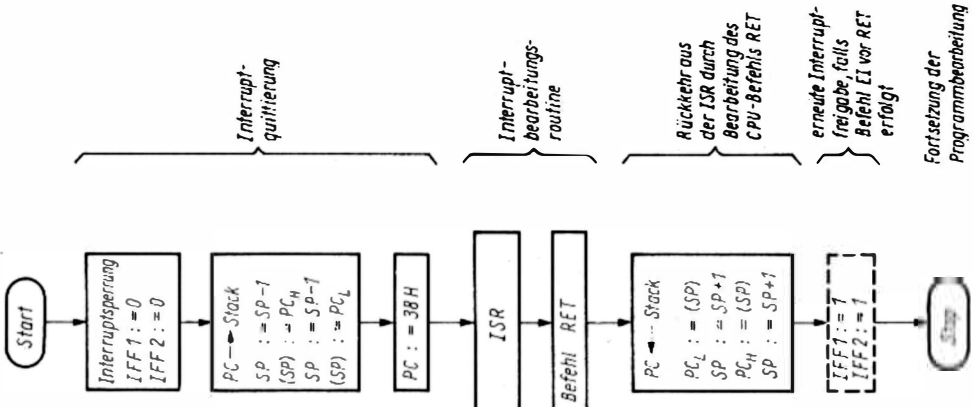
Bild 4.2.7. Zeitverhalten der CPU im Interruptzyklus der Mode IM2

LM, LT letzter Maschinenzklus, letzter Taktzustand des vorangegangenen Befehls  
 M1', M2, M3, M4, M5 Maschinenzklen der Quittierung  
 CP Systemtakt (T1, T2, TW, T3, T4 Taktzustände der CPU; TW automatisch von der CPU eingefügte WAIT-Zustände)

**INT** Interruptanmeldesignal der CPU  
**A<sub>0</sub> ... A<sub>15</sub>** Belegung des CPU-Adreßbusses (PC Programmzählerstand; R Inhalt des Refreshregisters; SP Stackpointerstand; IP Inhalt des intern in der CPU gebildeten Interruptpointers; PC rückgespeicherter Programmzählerstand)

**M1, MREQ, IORQ, RD, WR, RFSH** Steuerungsalgänge der CPU  
**D<sub>0</sub> ... D<sub>7</sub>** Belegung des Systemdatenbusses an der CPU IV Übernahme des Interruptvektors vom quitierten Systemelement; (PC)<sub>H</sub> höher- bzw. niederwertiger Teil des PC; J, 2 Lesen des nieder- und höherwertigen Teils des Programmzählerstandes PC' der ISR; J gültige Op-Kode-Daten des ersten ISR-Befehlsbytes

Bild 4.2.6  
 Flußbild der Interruptbearbeitung, Mode IM1



Fortsetzung der  
 Programmbearbeitung

Interrupt-  
 quittierung

Interrupt-  
 bearbeitungs-  
 routine

Rückkehr aus  
 der ISR durch  
 Bearbeitung des  
 CPU-Befehls RET

erneute Interrupt-  
 freigebe, falls  
 Befehl EI vor RET  
 erfolgt

grammaufwand zur Feststellung des interruptanmeldenden Geräts vermieden. Die Reaktion des Prozessors auf die Interruptanmeldung erfolgt deshalb sehr schnell und leistungsfähig. Die Bearbeitung dieser Interruptbetriebsart durch die CPU ist zusammenfassend im Bild 4.2.8 als Flußbild dargestellt. Da die effektive Anwendung der Interruptmode IM2 durch die Reaktionen der Systemperipherie unterstützt wird, werden die Haupteigenschaften der peripheren Elemente bei der Interruptbearbeitung in dieser Interruptbetriebsart im nachfolgenden Abschnitt näher beschrieben.

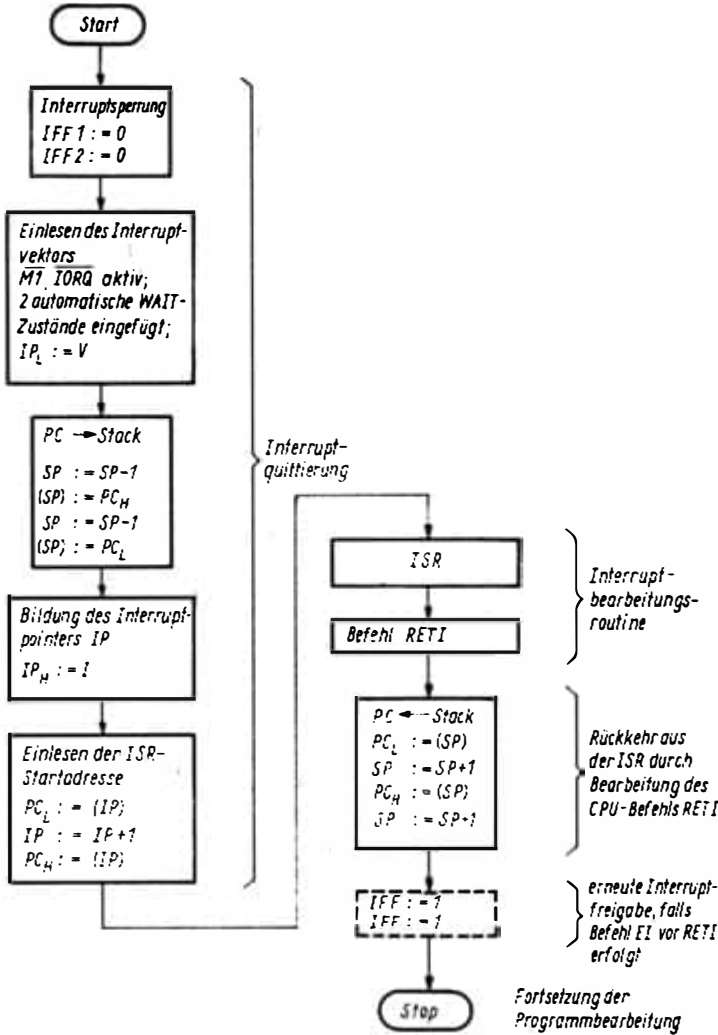


Bild 4.2.8  
Flußbild der Interruptbearbeitung, Mode IM2

### 4.2.3. Interrupteigenschaften der Systemelemente

#### 4.2.3.1. Priorität

Die Interruptstrukturen aller peripheren Systemelemente beinhalten eine automatisch wirkende Logik zur Auswahl der höchstwertigen Interruptanforderung. Hierzu werden sämtliche interruptfähige Peripherielemente mit Hilfe ihrer IEI- und IEO-Anschlüsse in Reihe geschaltet. In dieser Kaskadierungskette (daisy chain) erhält das vorderste Ele-

ment H-Pegel am IEI-Eingang. Dieser Pegel setzt sich in der gesamten Kette fort, vorausgesetzt, daß kein peripheres Gerät eine Interruptforderung aus der Peripherie empfängt. Sinngemäß besteht die Kaskadierungskette auch intern in den peripheren Systemelementen, indem die einzelnen Kanäle über eine IEI-IEO-Linie in Reihe geschaltet sind. Die Priorität der peripheren Systemelemente bzw. der Kanäle ist in der IEI-IEO-Kette somit durch die Lage dieser Elemente festgelegt. Weiter vorn liegende Geräte haben immer die höhere Anmeldepriorität. Der höchstwertige Kanal (Port A bei PIO und SIO, Kanal 0 des CTC) des ersten Elements in der Kette kann durch eine freigegebene Interruptanforderung die gesamte Prioritätenkette blockieren, da sein IEI-Eingang immer an H-Potential liegt.

Die Kaskadierungskette wirkt H-aktiv. Somit kann ein peripheres Systemelement, das am IEI-Eingang einen H-Pegel empfängt, eine anstehende Interruptforderung bei gesetztem Interruptfreigabeflipflop des Elements an die CPU weiterleiten. Es aktiviert seinen INT-Ausgang. Der IEO-Ausgang wird gleichzeitig inaktiv und bewirkt das Durchschleifen des L-Pegels in der nachfolgenden Kette. Dieser L-Pegel bewirkt nun bei den nachfolgenden Peripherielementen, daß eine anstehende Interruptforderung nicht zum Prozessor weitergeleitet wird. Nichtpriorisierte Elemente können somit Programmabläufe (ISR) von höherwertigen Peripherielementen nicht unterbrechen. Die Interruptquittierungslogik der peripheren Systemelemente wirkt abhängig vom Interruptanmelde- und Interruptquittierungszustand des jeweiligen Geräts. Ein interruptanmeldendes niederwertigeres Element wird durch die Interruptanforderung eines höherwertigeren Peripherielements veranlaßt, seine Anmeldung zurückzunehmen. Die Interruptanmeldung des in der Prioritätenkette weiter vorn liegenden Geräts wird bei Interruptfreigabe des Prozessors wirksam; die CPU schiebt die zugehörige ISR in den Programmablauf ein. Nach Rückkehr des Prozessors aus dieser Routine in das zuvor aktuelle Programmniveau erfolgt dann die erneute Anmeldung des niederwertigeren (zwischengespeicherten) Geräteinterrupts, da der Programmbearbeitungszustand des priorisierten Elements rückgesetzt wurde. Das niederwertigere Peripherielement empfängt H-Pegel am Freigabeeingang IEI. Andererseits kann ein höherwertigeres peripheres Systemelement bei einer quittierten Interruptanmeldung eines niederwertigeren seine Interruptanmeldung an die CPU weitergeben und die Interruptprioritätenkette durch Durchschleifen eines L-Signals blockieren. Diese höherpriorisierte Interruptanmeldung kann aber nur wirksam werden, wenn in der bearbeiteten ISR die Interruptfreigabe des Prozessors vorgenommen wurde (durch Befehl EI). In diesem Fall wird die ISR des höherwertigeren Peripherielements in den Programmablauf eingeschoben. Es erfolgt eine Verschachtelung der Interruptbearbeitungsroutinen. Nach Beendigung dieser (höherwertigen) ISR schließt sich die weitere Bearbeitung der unterbrochenen niederwertigeren Bearbeitungsroutine an. Anderenfalls aber (prozessorseitige Interruptsperrung in der ISR des niederwertigeren Geräts) wird zuerst die niederwertigere Serviceroutine beendet. Nach Rückkehr in die zuvor unterbrochene Programmabarbeitung kann bei dortiger Interruptfreigabe des Prozessors die nach wie vor anstehende Interruptanmeldung des höherwertigeren peripheren Elements wirksam werden. Damit in diesem Fall das niederwertigere Peripherielement dennoch beim Rückkehrbefehl RETI als aktives, interruptbearbeitendes Element ermittelt werden kann, müssen höherpriorisierte Peripherielemente mit nichtquittierten Interruptanmeldungen ihren IEO-Ausgang während der Ausführung des RETI-Befehls (Befehlsbytefolge EDH, 4DH) freigeben.

Die Interruptkaskadierungskette dient also zur systemweiten Auswahl des jeweils höchstwertigen interruptanfordernden peripheren Systemelements. Hierbei ist dieses höchstwertige Element während der Interruptquittierung des Prozessors durch die Signalverhältnisse am IEI-Eingang (H-Pegel) und am IEO-Ausgang (L-Pegel) gekennzeichnet.

Von diesen Pegelverhältnissen wird im Interruptquittierungszyklus das Setzen des Interruptbearbeitungsflipflops des Systemelements abgeleitet sowie die Aussendung des Interruptvektors ausgelöst. Bei der Rückkehr aus der Interruptbearbeitungsroutine wird mit Hilfe der Kaskadierungskette das höchstwertige in der Interruptbearbeitung befindliche (Interruptbearbeitungsflipflop gesetzt) Peripherielement ermittelt. Dieses Element ist während der Abarbeitung des RETI-Befehls durch H-Pegel am IEI-Eingang und L-Pegel am IEO-Ausgang gekennzeichnet. Es kann somit seinen Bearbeitungszustand zurücksetzen und die nachfolgende IEI-IEO-Linie freigeben.

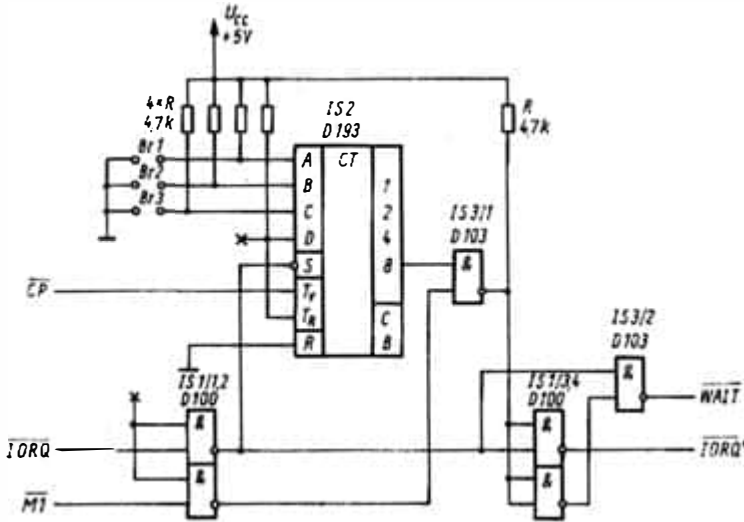


Bild 4.2.9. WAIT-Generator zur Verlängerung der L-Einschwingzeit der Interruptprioritätenkette

$\overline{IORQ}$  Steuersignalausgang der CPU

$\overline{IORQ}^*$  Steuersignaleingang der peripheren Systemelemente

In Mikrorechnern mit vielen peripheren Systemelementen (mehr als vier) treten beim Einschwingvorgang der Kaskadierungskette im Interruptquittierungszyklus Zeitprobleme auf. Für das Einschwingen der Kette steht die Zeit zwischen dem Aktivwerden des  $\overline{MI}$ -Signals und dem Aktivwerden des  $\overline{IORQ}$ -Signals zur Verfügung. Die Zeit zwischen dem Aktivwerden beträgt bei der maximalen Systemtaktfrequenz ( $f_c = 2,5 \text{ MHz}$ ) und unter Worst-case-Bedingungen  $t_n = 870 \text{ ns}$  (mögliche Verzögerung durch Steuerbustreiber nicht berücksichtigt). Demgegenüber tritt an einem Peripherielement, das interruptmäßig passiv ist (keine Anmeldung, keine Bearbeitung), beim Durchschleifen eines L-Pegels von IEI nach IEO eine Verzögerungszeit von  $t_{DL(IO)} = 190 \text{ ns}$  auf. Beim interruptanmeldenden peripheren Element kann weiterhin das Zurücksetzen seines Interruptfreigabeausgangs IEO gegenüber dem Aktivwerden des  $\overline{MI}$ -Signals um maximal  $t_{DM(IO)} = 300 \text{ ns}$  verzögert werden, wenn die Interruptforderung gerade kurz vor Eintreffen des  $\overline{MI}$ -Signals im Element ausgelöst wurde. Des Weiteren ist eine Voreinstellzeit am Eingang IEI des letzten peripheren Systemelements in der Kette vor Eintreffen des aktivierten  $\overline{IORQ}$ -Signals zu berücksichtigen, damit eine evtl. anliegende niederwertigere Interruptforderung dieses Elements noch gesperrt werden kann. Diese Voreinstellzeit beträgt  $t_{s(IEI)} = 140 \text{ ns}$ . Die wirksame Zeit für den Einschwingvorgang an den passiven Peripherielementen beträgt deshalb nur  $t = 430 \text{ ns}$ . In dieser Zeit muß das Durchschleifen des L-Pegels durch  $n - 2$  Elemente ( $n$  Anzahl aller in der Kette liegenden Peripherielemente

des Mikrorechners) gesichert werden. Für große Mikrorechner muß deshalb eine Hardwarelösung geschaffen werden, die ein korrektes Einschwingen der IEI-IEO-Linie gewährleistet. Prinzipiell kann hierbei entweder die wirksame Zeit für den Einschwingvorgang vergrößert werden, oder die Verzögerungszeit der peripheren Elemente muß verringert werden. Im Bild 4.2.9 ist die Logik eines WAIT-Generators dargestellt, der im M1-Maschinenzyklus der Interruptquittierung durch Hardwareprogrammierung (Brücken) bis zu acht WAIT-Zustände erzeugt. Gleichzeitig wird durch Torung ein  $\overline{\text{IORQ}}$ -Signal für die Peripherie gebildet, das gegenüber dem CPU-Signal  $\overline{\text{IORQ}}$  um  $t_D = 400$  bis 3200 ns (bei maximaler Systemtaktfrequenz,  $f_C = 2,5$  MHz) verzögert wird. Die Zeit für den Einschwingvorgang der Kaskadierungskette kann somit entsprechend der vorhandenen Anzahl von Peripheriebauelementen erweitert werden. In Tafel 4.2.1 ist der Zusammenhang zwischen Lage der Brücken, Zahl der hierdurch erzeugten WAIT-Zustände und Anzahl der einzusetzenden interruptfähigen Peripherieelementen ausgewiesen. Der Vorteil der im Bild 4.2.9 dargestellten Hardwaremodifikation besteht darin, daß der notwendige Schaltungsaufwand nur einmal, üblicherweise auf der CPU-Leiterkarte eines

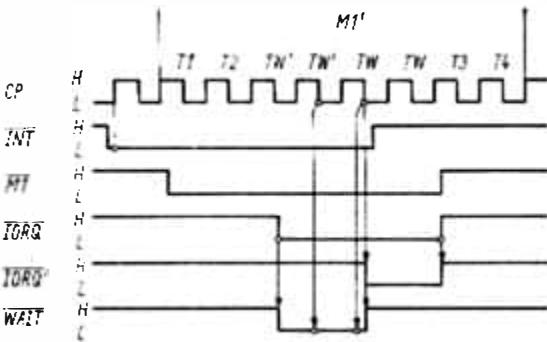


Bild 4.2.10. Zeitverhalten der WAIT-Logik nach Bild 4.2.9 im Interruptquittierungszyklus der CPU

- M1' M1-Maschinenzyklus der Interruptquittierung
- CP Systemtakt (T1, T2, TW', TW, T3, T4 Taktzustände der CPU; TW' automatisch von der CPU eingefügte WAIT-Zustände)
- $\overline{\text{INT}}$  INT-Anmeldeeingang der CPU
- $\overline{\text{MT}}, \overline{\text{IORQ}}$  Steuersignalausgänge der CPU
- $\overline{\text{IORQ}'}$  Steuersignal  $\overline{\text{IORQ}}$  für periphere Systemelemente
- $\overline{\text{WAIT}}$  WAIT-Steuersignaleingang der CPU

Tafel 4.2.1. Wertigkeit der Brücken in der WAIT-Logik nach Bild 4.2.9

Lage der Brücken	Belegung an IS2			Anzahl von TW	Zeit $t_{w1}$ ns	Anzahl von PIO, CTC, SIO
	A	B	C			
—	1	1	1	1	830	6
Br 1	0	1	1	2	1230	8
Br 2	1	0	1	3	1630	10
Br 1,2	0	0	1	4	2030	12
Br 3	1	1	0	5	2430	14
Br 1,3	0	1	0	6	2830	16
Br 2,3	1	0	0	7	3230	19
Br 1,2,3	0	0	0	8	3630	21

TW durch die Logik nach Bild 4.2.9 erzeugte (nicht automatisch eingefügte) WAIT-Zustände  
 $t_{w1}$  wirksame Zeit für den L-Einschwingvorgang der Kette an den interruptmäßig passiven Systemelementen

Rechners, für das gesamte System erforderlich ist. Die Einfügung der WAIT-Zustände beeinflußt die Leistungsfähigkeit des Mikrorechners selbst bei vielen peripheren Systemelementen und bei vielen Interruptreaktionen i. allg. nur wenig. Bild 4.2.10 zeigt das Zeitverhalten des angepaßten M1-Zyklus der Interruptquittierung.

Die alternative Lösung für die Absicherung des Einschwingvorgangs der Interruptkaskadierungskette besteht in der Verminderung der Durchlaufzeiten des L-Pegels bei den peripheren Elementen. Da die Kette H-aktiv wirkt, muß zur „vorausschauenden“ Weitergabe des L-Pegels die AND-Verknüpfung zwischen den IEI- und IEO-Linien angewendet werden. Im Bild 4.2.11 ist diese einfachste Möglichkeit der Realisierung einer Umgehungslogik (look ahead logic) dargestellt. Die Verzögerungszeit beim Durchschleifen des L-Pegels wird nur noch durch die Durchlaufzeiten der Gatter bestimmt und beträgt  $t_D = 30$  ns (1 Gatter D100, 1 Inverter D204) bzw.  $t_D = 35$  ns (2 Gatter D100). Bei der Systemtaktfrequenz  $f_C = 2,5$  MHz können also hierbei ohne Anwendung weiterer Maßnahmen bis zu 16 Peripherielemente eingesetzt werden (14 bei Verwendung von 1/2 D100). Im Bild 4.2.12 ist eine etwas veränderte Schaltungsanordnung der Umgehungslogik angegeben, die den Einsatz von maximal 29 Systemelementen erlaubt. Die Schaltungsverzögerung wird durch die Zusammenfassung der Elemente in Vierergruppen verringert.

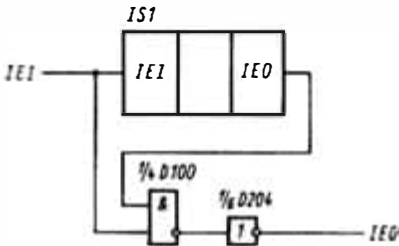
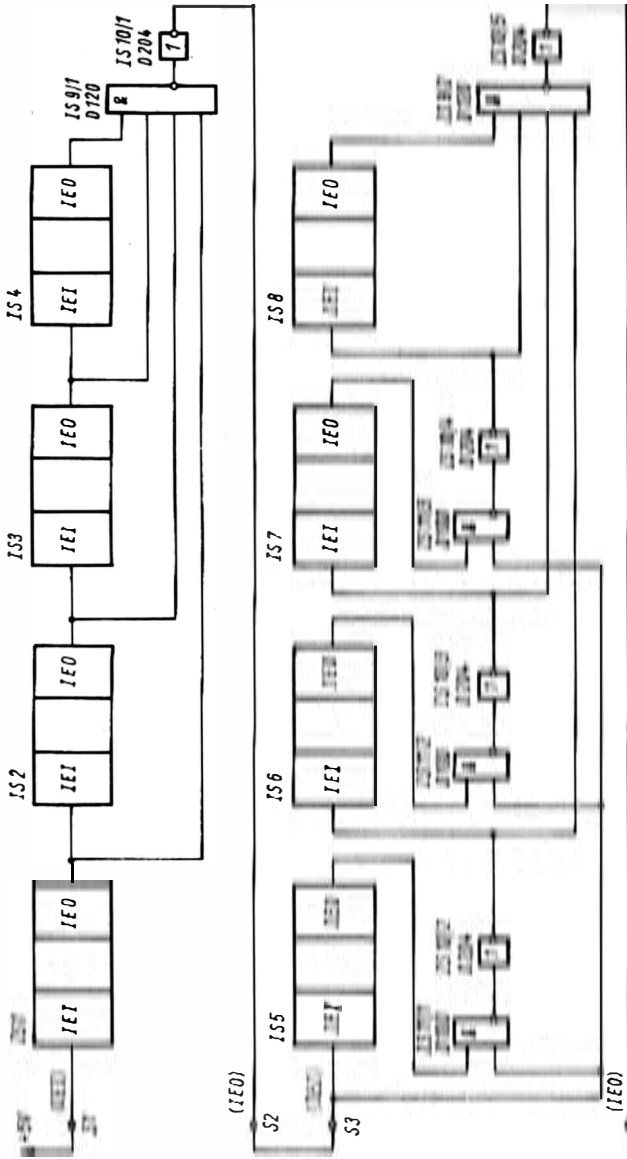


Bild 4.2.11  
Einfache Umgehungslogik  
für IEI-IEO-Prioritätenkette  
IS1 peripheres Systemelement (PIO, SIO, CTC)

Bei der praktischen Realisierung eines Mikrorechners bzw. eines -rechnersystems ist unter diesen Gesichtspunkten abzuschätzen, wie viele periphere Elemente eingesetzt werden müssen. Hierbei ist zu berücksichtigen, daß z. B. die Zusammenfassung der Elemente in Vierergruppen meist nicht sinnvoll ist. Im Vordergrund steht die funktionelle Zusammenfassung der Elemente, da wegen der Verdrahtungsstruktur der Mikrorechner (Busstruktur; einheitliche gedruckte Rückseitenverdrahtung od. dgl.) für Steckeinheiten je ein IEI- und IEO-Anschluß vorteilhaft ist. Der Einsatz einer kombinierten Hardwarelösung ist deshalb beim Aufbau sehr großer Mikrorechner ebenfalls denkbar.

#### 4.2.3.2. Anmeldung und Quittierung

Ein peripheres Systemelement setzt bei einer Interruptforderung ein internes Anmeldeflipflop. Der aktive Zustand dieses Flipflops wird unter Interruptfreigabebedingungen des Elements (Interruptfreigabeflipflop des Peripherielements gesetzt, IEI-Linie aktiviert) über die Interruptanmeldeleitung  $\overline{INT}$  an den Prozessor weitergeleitet. Bei Interruptfreigabe der CPU erfolgt die Auswertung des  $\overline{INT}$ -Eingangs am Ende des in Bearbeitung befindlichen Befehls. Daraufhin wird ein Interruptquittierungszyklus in den Befehlsablauf eingeschoben. In der Zeit zwischen Auswertung der  $\overline{INT}$ -Linie durch die CPU (steigende Systemtaktflanke des letzten T-Zustands) und dem Aktivwerden des  $\overline{M1}$ -Signals des Quittierungszyklus (s. Bild 4.2.7) können aber noch weitere periphere Elemente eine Interruptanmeldung an die CPU aussenden. Die den Quittierungszyklus anfordernde und die in diesem Zyklus quittierte Interruptanmeldung müssen daher nicht



(zu weiteren Vierergruppen peripherer Elemente)

**Bild 4.2.12. Erweiterte Umgebungslogik für Interruptprioritätenkette**

- IS1 ... IS8 periphere Systemelemente (PIO, SIO, CTC)
- SI ... S4 mögliche Schnittstellen zwischen Steckeinheiten

identisch sein. Die Auswahl, welche Anmeldung quittiert wird, erfolgt durch Feststellung der Priorität in der Kaskadierungskette. Beim Aktivwerden des  $\overline{\text{IORQ}}$ -Steuersignals der Interruptquittierung hat diese Prioritätenkette einen eingeschwungenen Zustand. Das höchstwertige, interruptanmeldende Peripherielement weist die Signalkonfiguration  $\text{IEI} = \text{H}$  und  $\text{IEO} = \text{L}$  auf. Diese Konfiguration bewirkt im Quittierungszyklus, daß das Element die Interruptanmeldung auf der  $\text{INT}$ -Linie zurücknimmt, ein internes Interruptbearbeitungsflipflop setzt und das entsprechende Interruptvektorregister auf den Datenbus schaltet.

Die Freigabe der  $\overline{\text{INT}}$ -Anmeldelinie durch das quittierte Peripherielement geschieht durch Rücksetzung des internen Interruptanmeldeflipflops. Das erneute Setzen dieses Flipflops wird aber im Element für die Zeitdauer verhindert, in der es sich im Interruptbearbeitungszustand befindet, also sein Bearbeitungsflipflop gesetzt ist. Somit wird verhindert, daß ein Peripherielement bei einer erneuten Interruptforderung nicht die eigene ISR unterbrechen kann (vorausgesetzt, der Befehl  $\text{EI}$  erfolgte in dieser ISR). Es sind nur Verschachtelungen der ISR durch interruptmäßig höherpriorisierte Elemente möglich. Erst nach Verlassen des Interruptbearbeitungszustands kann ein peripheres Systemelement eine anhängige Interruptforderung wieder durch Setzen des Anmeldeflipflops an den CPU-Eingang  $\overline{\text{INT}}$  weitergeben. Es erfolgt bei Interruptfreigabe des Prozessors dann ein erneutes Einschleusen eines Interruptquittierungszyklus. Ist das betrachtete Element nach wie vor das höchstwertige, so erfolgt wiederum der Aufruf seiner ISR.

Das Setzen des internen Interruptbearbeitungsflipflops bei der Interruptquittierung des priorisierten Peripherielements bewirkt somit eine Sperrung neuer anhängiger Interruptforderungen dieses Elements. Gleichzeitig wird vom Zustand dieses Flipflops das Ausenden des L-Pegels am Freigabeausgang  $\text{IEO}$  abgeleitet. Dieser Pegel bleibt für die gesamte Dauer der Interruptbearbeitung des Elements erhalten.

Eine weitere Funktion des Interruptquittierungszyklus der CPU besteht darin, den Interruptvektor des anmeldenden und des quittierten Systemelements anzufordern. Das Element (Signalkonfiguration  $\text{IEI} = \text{H}$ ,  $\text{IEO} = \text{L}$ ) sendet den Vektor nach Empfang des aktiven  $\overline{\text{IORQ}}$ -Signals auf dem Datenbus aus. Die prozessorseitige Auswertung der Busbelegung geschieht mit der steigenden Flanke des  $\text{T3}$ -Systemtaktzustands im  $\text{M1}$ -Maschinenzyklus der Quittierung (Bild 4.2.7). Da in großen Mikrorechnern üblicherweise sowohl CPU-seitig als auch periphereseitig Datenbustreiber eingesetzt werden, muß eine ordnungsgemäße Richtungsumschaltung dieser Treiber im Interruptquittierungszyklus vorgesehen werden. Treten bei dieser entsprechenden Selektierung Zeitprobleme auf, so kann das Lesen des Vektors durch den Prozessor verzögert werden, indem ein  $\text{WAIT}$ -Zustand (bzw. mehrere Zustände) im Interruptquittierungszyklus eingefügt wird. Bild 4.2.13 zeigt einen entsprechenden Schaltungsvorschlag.

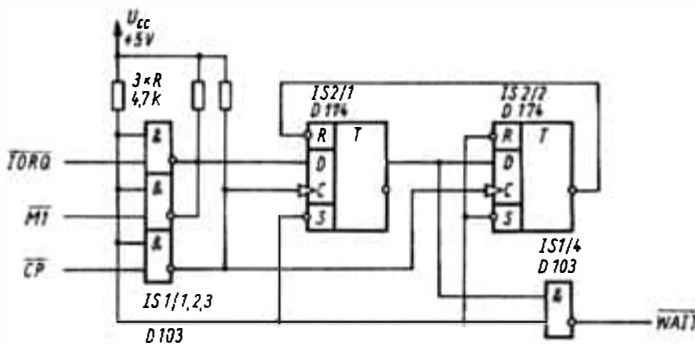


Bild 4.2.13  
WAIT-Generator zur Verlängerung des Interruptvektorlesezyklus

Nach dem Lesen des Interruptvektors im M1-Maschinenzyklus der Interruptquittierung führt die CPU die Retteaktion des Programmzählerstands aus. Hierzu arbeitet sie zwei Speicherschreibzyklen (M2, M3) ab, in denen H- und L-Byte des PC in den durch SP-1 und SP-2 adressierten Stackbereich des Schreib/Lese-Speichers abgelegt werden. Im weiteren Ablauf des Quittierungszyklus folgen dann zwei weitere Maschinenzyklen (M4, M5; Speicherlesezyklen), die zur Bildung der Unterprogrammstartadresse dienen. In diesen M-Zyklen liest der Prozessor L- und H-Byte der ISR-Startadresse. Zur Adressierung dieser beiden Speicherzellen sendet die CPU im M4-Zyklus auf dem niederwertigeren Adreßbus ( $A_0 \dots A_7$ ) den Interruptvektor und auf dem höherwertigeren Bus ( $A_8$  bis  $A_{15}$ ) den Inhalt des I-Registers aus. Im nachfolgenden Zyklus (M5) erfolgt die Ausendung des inkrementierten Pointers. Prinzipiell wirkt also der Interruptquittierungszyklus wie ein CALL-Befehl, mit der Ausnahme, daß der neugebildete Stand indirekt adressiert wird. Mit diesem Programmzählerstand erfolgt dann das Lesen des nächsten Befehlsbytes. Die ISR des interruptauslösenden peripheren Systemelements wird bearbeitet.

#### 4.2.3.3. Rückkehr aus Bearbeitung

Nach Abarbeitung der Interruptbearbeitungsroutine (ISR) eines peripheren Systemelements erfolgt der Rücksprung in das zuvor unterbrochene Programmniveau durch Ausführung des Befehls RETI. Prozessorseitig wird hierdurch der aktuelle, vor der Programmunterbrechung gültige Programmzählerstand (PC) aus dem Stackbereich des Schreib/Lese-Speichers rückgespeichert. Der Programmbearbeitungszustand vor Einschachtelung der ISR wird somit bezüglich PC- und SP-Stand wiederhergestellt. Das entsprechende CPU-Zeitverhalten hierbei ist im Bild 4.2.14. dargestellt. Die Ausführung des RETI-Befehls hat also für den Prozessor die gleiche Wirkung wie die Abarbeitung des

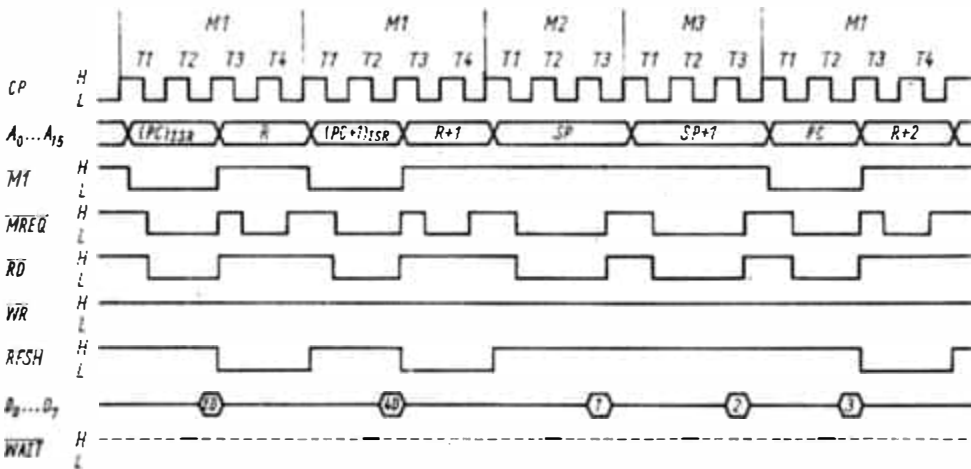


Bild 4.2.14. Zeitverhalten der CPU bei Befehlsbearbeitung RETI

M1, M2, M3 Maschinenzyklen der CPU

CP Systemtakt (T1, T2, T3, T4 Taktzustände der CPU)

$A_0 \dots A_{15}$  Belegung des CPU-Adreßbusses [(PC)<sub>ISR</sub> Programmzählerstand in der ISR; R Inhalt des Refreshregisters; SP Stackpointerstand; PC Programmzählerstand aus Stack]

$\overline{M1}$ ,  $\overline{MREQ}$ ,  $\overline{RD}$ ,  $\overline{WR}$ , RFSH

Steuersignalausgänge der CPU

$D_0 \dots D_7$  Belegung des Systemdatenbusses an der CPU (ED, 4D Befehlsbytes von RETI; 1 Einlesen des PC, niederwertiges Byte, 2 Einlesen des PC, höherwertiger Teil; 3 gültiges Op-Kode-Byte des folgenden Befehls)

WAIT Steuersignaleingang der CPU

Rückkehrbefehls RET. Die Unterschiede der beiden Befehle sind die verschiedenen Op-Kode und die Befehlsausführung (Einbytebefehl RET, Zweibytebefehl RETI).

In der Peripherie bewirkt der Rückkehrbefehl RETI, daß das Systemelement mit der zuletzt quittierten Interruptanmeldung seinen Interruptbearbeitungszustand verläßt. Hierzu dekodieren alle Elemente parallel die auf dem Datenbus ankommenden Befehlsbytes. Diese Bytes (op-codes) werden in den peripheren Elementen durch die aktiven  $\overline{M1}$ - und  $\overline{RD}$ -Signale erkannt und mit steigender Systemtaktflanke (Zustand T3) abgespeichert. Da der RETI-Befehl ein Zweibytebefehl ist, müssen sein Steuerbyte (EDH) und sein Befehlsbyte (4DH) von anderen, gleichen Bytefolgen unterschieden werden. Hierzu dekodieren die Peripherieelemente zusätzlich das Steuerbyte CBH. Das Erkennen dieses Bytes bewirkt, daß das nachfolgende Befehlsbyte (im nächsten M1-Maschinenzyklus) in der Auswertelogik der Systemelemente ignoriert wird. Die Bytefolge CBH, EDH, 4DH kann also nicht als RETI-Befehl interpretiert werden, da das Steuerbyte CBH keinen eigenen Befehlskode darstellt und daher das Byte EDH zur Spezifizierung verwendet (siehe Abschn. 3.1.).

Nach Dekodierung der korrekten Bytefolge des RETI-Befehls wird das Interruptbearbeitungsflippflop des höchstwertigen sich in Bearbeitung befindlichen peripheren Systemelements rückgesetzt. Die während der Interruptbearbeitung blockierte IEO-Linie wird hierdurch vom Element freigegeben, d. h., der logische Pegel an IEO folgt dem Pegel am Freigabeeingang IEI. In der Peripherie erfolgt die Auswahl, welches Peripherieelement während der RETI-Ausführung den höchstwertigen Interruptbearbeitungszustand aufweist, ähnlich wie bei der Interruptquittierung durch einen Einschwingvorgang der IEI-IEO-Prioritätenkette. Hierbei muß dieses höchstwertige bearbeitete Systemelement wieder die Signalkonfiguration IEI = H und IEO = L aufweisen.

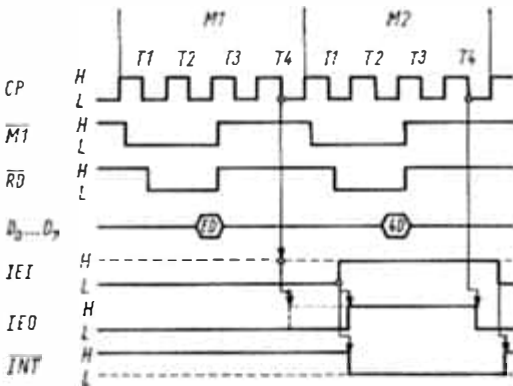


Bild 4.2.15

Zeitverhalten der nichtquittierten interruptfordernden Systemelemente bei Befehlsabarbeitung RETI

- M1 Maschinenzyklen der CPU
- CP Systemtakt (T1, T2, T3, T4 Taktzustände der CPU)
- $\overline{M1}$ ,  $\overline{RD}$  Steuersignalausgänge der CPU bzw. -eingänge der peripheren Systemelemente
- D<sub>0</sub> ... D<sub>7</sub> Beladung des Systemdatenbusses an der CPU (ED, 4 D Befehlsbytes von RETI)
- IEI, IEO Interruptfreigabelinien des nichtquittierten interruptfordernden Systemelements
- $\overline{INT}$  INT-Anmeldeausgang des Systemelements

In der durch den RETI-Befehl abgeschlossenen ISR bestand aber die Möglichkeit, daß höchstwertigere periphere Elemente Interruptanmeldungen an die CPU weitergeben, die nicht quittiert wurden und somit keine weiteren Programmverschachtelungen bewirkten.

Diese höherwertigeren Anmeldungen werden vom Prozessor dann nicht quittiert, wenn die Interruptsperrung der CPU noch vorliegt (Befehl EI erfolgte nicht) oder, anderenfalls, wenn die Interruptforderung erst kurz vor dem RETI-Befehl (nach der letzten steigenden Systemtaktflanke des letzten Befehls vor RETI) an der CPU eintraf. Derartige nichtquittierte Interruptanmeldungen sperren aber die nachfolgende Prioritätenkette, und das höchstwertige quittierte Peripherieelement empfängt L-Pegel am Freigabeeingang IEI. Damit nun dieses Element am Ende des RETI-Befehls die korrekte Signalkonfiguration (IEI = H, IEO = L) empfangen und seinen Interruptbearbeitungszustand ver-

lassen kann, müssen alle nichtquittierten interruptfordernenden Systemelemente die IEI-IEO-Prioritätenkette für die Dauer von RETI freigeben. Der aktive H-Pegel kann dann in der Kette bis zum IEI-Eingang des höchstwertigen quittierten Elements durchgeschaltet werden. Damit für diesen Einschwingvorgang der Kette Zeit gewonnen wird, geschieht die Freigabe der Prioritätenkette durch die nichtquittierten interruptfordernenden Elemente nach Erkennen des RETI-Steuerbytes EDH. Da weitere U880-Befehle dieses Steuerbyte aufweisen, erfolgt der H-Einschwingvorgang nicht nur bei Ausführung des RETI-Befehls. Nach Auswertung des nachfolgenden (von EDH verschiedenen) Befehlsbytes werden die Freigabeausgänge der nichtquittierten Elemente wieder auf L-Pegel geschaltet. Das zugehörige Zeitverhalten ist im Bild 4.2.15 dargestellt. Folgt auf das Steuerbyte EDH das Befehlsbyte 4DH, so verläßt das höchstwertige, quittierte Systemelement den Interruptbearbeitungszustand, indem sein internes Interruptbearbeitungsflipflop zurückgesetzt wird. Das Zeitverhalten der peripheren Systemelemente bei Rückkehr aus der Interruptbearbeitung zeigt Bild 4.2.16.

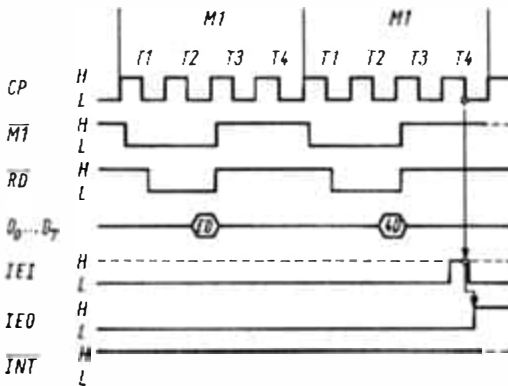


Bild 4.2.16  
Zeitverhalten der peripheren Systemelemente bei Verlassen des Interruptbearbeitungszustands nach Befehlsabarbeitung RETI  
Signale s. Bild 4.2.15

In Mikrorechnern mit mehreren interruptfähigen peripheren Systemelementen können Zeitprobleme des Einschwingvorgangs der Prioritätenkette bei Verlassen des Interruptbearbeitungszustands auftreten. Für den Fall, daß höherwertigere Peripherieelemente nichtquittierte Interruptforderungen aufweisen, müssen diese Elemente nach Empfang des ersten RETI-Bytes EDH die Blockierung ihrer IEO-Freigabeausgänge aufheben. Der aktive H-Pegel kann sich daraufhin bis zum Eingang IEI des höchstwertigen quittierten Elements fortsetzen. Dieses sich noch in Interruptbearbeitung befindende Systemelement muß aber das Signal  $IEI = H$  spätestens kurz vor der internen Dekodierung des zweiten RETI-Bytes 4DH empfangen, damit es als höchstwertiges bearbeitetes Element gekennzeichnet ist und sein Bearbeitungsflipflop zurücksetzt. Die zulässige Zeit für den IEI-IEO-Einschwingvorgang bei RETI beträgt systembedingt unter Worst-case-Bedingungen  $t_{w2} = 1000 \text{ ns}$ . Sie setzt sich nach Gl.(4.2.1) aus der Zeitdifferenz zwischen der elementeinternen Dekodierung der beiden RETI-Bytes, der Verzögerungszeit des IEO-Signals am Ausgang des nichtquittierten Elements und der Voreinstellzeit des eintreffenden Prioritätssignals am Freigabeeingang IEI des bearbeiteten Peripherielements zusammen:

$$t_{w2} = (4 + m) t_C - t_{DH2} - t_{S2}; \quad (4.2.1)$$

$t_{w2}$  Zeit des Einschwingvorgangs der Kette bei RETI

$t_C$  Systemtaktperiode ( $t_C = 400 \text{ ns}$  bei  $f_C = 2,5 \text{ MHz}$ )

$t_{DH2}$  IEO-Signalverzögerung auf H bei EDH ( $t_{DH2} = 400 \text{ ns}$ )

$t_{S2}$  IEI-Voreinstellzeit bei RETI ( $t_{S2} = 200 \text{ ns}$ )

$m$  Zahl der WAIT-Zustände im M1-Maschinenzyklus bei EDH (üblicherweise  $m = 0$ ).

Diese Einschwingzeit reicht je nach Realisierung des L-Einschwingens der Prioritätenkette (bei Interruptquittierung) und abhängig vom Typ der eingesetzten peripheren Systemelemente für Kettenlängen bis sechs Peripherieelemente aus. Diese Anzahl der im Mikrorechner ohne Zusatzlogik einsetzbaren Elemente kann mit Gl. (4.2.2) ermittelt werden:

$$t_{w2} > (n - 2) (t_{DH(10)} + t_{DTTL}); \quad (4.2.2)$$

$n$  Anzahl der Elemente in der IEI-IEO-Kette

$t_{DH(10)}$  IEO-Verzögerungszeit auf H von IEI ( $t_{DH(10)} = 210$  ns bei PIO, CTC;  $t_{DH(10)} = 250$  ns bei SIO)

$t_{DTTL}$  TTL-Verzögerungszeit bei Verwendung einer Umgehungslogik ( $t_{DTTL} = 30 \dots 35$  ns)

$t_{w2}$  Zeit für H-Einschwingvorgang.

Beim Einsatz einer größeren Anzahl peripherer Elemente in U880-Mikrorechnern ist somit eine Zusatzlogik erforderlich, die ein korrektes Einschwingen der Prioritätenkette bei der Rücksetzung der Interruptbearbeitung gewährleistet. Prinzipiell können Schaltungen angewendet werden, die entweder die zulässige H-Einschwingzeit der Kette vergrößern oder die Durchlaufzeiten in den höherwertigeren Systemelementen vermindern. Bei dem peripheren Systemelement SIO besteht neben der dargestellten systemeigenen Interruptrückkehr durch den RETI-Befehl eine weitere, spezifische Rückkehrmöglichkeit. Diese besteht darin, daß in der speziellen der jeweiligen SIO zugehörigen ISR ein Steuerbefehl an dieses Element ausgesendet wird, der unabhängig vom Zustand des IEI-Eingangs der SIO, aber in Auswertung der SIO-internen Prioritätenkette das höchstwertige gesetzte Interruptbearbeitungsflipflop rücksetzt. Die Programmrückkehr muß danach durch den RET-Befehl erfolgen. Voraussetzung für diese Art der *Interruptrückkehr* ist aber, daß die CPU in Mode IM2 betrieben wird, da gerätezugehörige ISR (zumindest bei der SIO) erforderlich sind (vektorierte Interrupts). Die Anwendung dieser SIO-spezifischen Rückkehr kann den Hardwareaufwand in mittleren Mikrorechnern verringern, wenn insgesamt weniger als sechs PIO- bzw. CTC-Elemente eingesetzt werden. Aufgrund der unterschiedlichen Forderungen von Mikrorechnerkonzepten sollen nachfolgend die wichtigsten Möglichkeiten der Realisierung der genannten Zusatzlogik diskutiert werden.

Im Bild 4.2.17 ist eine Schaltungskonzeption dargestellt, die eine hardwaremäßig programmierbare Anzahl von WAIT-Zuständen in den Befehlslesezyklus (M1-Zyklus) des zweiten RETI-Bytes (4DH) einfügt. Hierzu wird der Systemdatenbus des Mikrorechners ( $D_0 \dots D_7$ ) bezüglich des Auftretens der Befehlsbytes CBH, EDH und 4DH überwacht. Das hierbei verwendete Zeitverhalten bezieht sich auf die Verwendung von Standard-ROM/PROM-Bauelementen im Programmspeicher des Rechners, die aufgrund ihrer Zugriffszeit ( $t_{ACC} \leq 450$  ns) Befehlslesevorgänge ohne Einfügung zusätzlicher WAIT-Zustände ermöglichen. Bei Einsatz langsamerer Speicher (z. B. U551/U552) muß die Zeitsteuerung der Logik geändert werden. Die Abspeicherung der durch die IS3 ... IS5 dekodierten Dateninformationen erfolgt entsprechend dem Zeitverhalten des Prozessors mit steigender Flanke des T3-Systemtaktzustands in den M1-Maschinenzyklen (nur op-code fetches). Nach Dekodierung und Speicherung des Befehlssteuerbytes CBH erfolgt die Sperrung des EDH-Flipflops (IS 8/1), damit die Bytefolge CBH, EDH, 4DH nicht als RETI-Befehlscode erkannt werden kann. Im anderen Fall bewirkt das Einspeichern des dekodierten EDH-Bytes (korrektes erstes RETI-Byte) im Trigger IS 8/1, daß im nachfolgenden M1-Maschinenzyklus automatisch zunächst ein WAIT-Zustand eingefügt wird. Hierdurch wird Zeit für die Auswertung und Speicherung des zweiten, auf EDH folgenden Befehlsbytes gewonnen. Mit der steigenden Flanke des eingefügten WAIT-Zustands liegt die Information vor, ob das zweite Byte den Op-Kode 4DH aufweist. In diesem Fall liegt der RETI-Befehl vor, und es erfolgt die Anforderung weiterer WAIT-Zustände, die die wirksame Zeit für den H-Einschwingvorgang der Prioritätenkette gemäß



Gl. (4.2.1) verlängern. Der WAIT-Generator nach Bild 4.2.17 erzeugt bei Bearbeitung des Befehls RETI hardwaremäßig programmierbar bis zu neun zusätzliche WAIT-Zustände. Aufgrund der hierdurch erweiterten Einschwingzeit kann je nach Typ der eingesetzten peripheren Systemelemente und abhängig von der Realisierung des L-Einschwingens der Kette bei Interruptquittierung eine Kettenlänge mit max. 23 Elementen erreicht werden (bei einer Systemtaktfrequenz  $f_C = 2,5 \text{ MHz}$ ). Die Logik ist bezüglich der Anzahl der erzeugten WAIT-Zustände erweiterbar, so daß die H-Einschwingzeit auch noch weiter vergrößert werden kann. Der wesentlichste Vorteil dieser Lösung besteht aber darin, daß die Logik in vorhandenen Mikrorechnern, deren Peripherie erweitert werden soll, *nachgerüstet* werden kann.

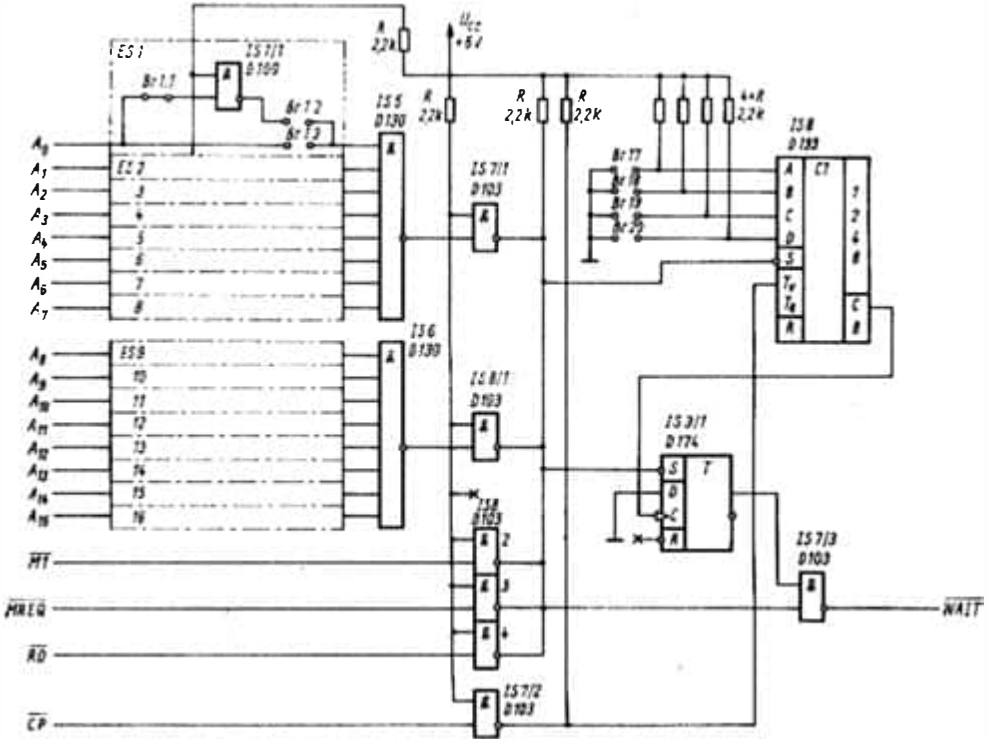


Bild 4.2.18. WAIT-Generator zur Verlängerung der H-Einschwingzeit der Interruptprioritätenkette durch Dekodieren des Programmzählerstands

ES1 ... ES16 gleichartige, programmierbare Eingangsstufen (evtl. auch festverdrahtet oder ersetzt durch zwei  $256 \times 1$ -organisierte bipolare PROM)

Die WAIT-Logik verwendet nur Signale des Systemdaten- und Systemsteuerbusses und kann deshalb als unabhängige Steckeinheit konzipiert werden. Bei ihrem Einsatz sind des weiteren keine Programmänderungen notwendig. Nachteilig erweist sich jedoch die Einfügung der zusätzlichen WAIT-Zustände auf die Leistungsfähigkeit des Mikrorechners. Die Befehlsabarbeitungszeit des Rechners wird vor allem durch den in jeden Befehl mit dem Steuerbyte EDH eingefügten WAIT-Zustand negativ beeinflusst, da insbesondere diese U880-Befehle besonders leistungsfähig sind (z. B. Gruppe der repetierend wirkenden Befehle).

Im Bild 4.2.18 ist die Schaltungsvariante eines WAIT-Generators dargestellt, der den letztgenannten Nachteil der Lösung nach Bild 4.2.17 vermeidet. Voraussetzung für die

Anwendung der Schaltung ist aber, daß die Programmspeicherzelle des zweiten RETI-Bytes bekannt und daß der eigentliche RETI-Befehl nur einmal im Programmspeicher vorhanden ist. Die Rückkehr aus einer ISR darf somit nicht durch Ausführung des Befehls RETI bzw. der Befehlsfolge EI; RETI erfolgen, sondern muß durch einen Sprung auf eine feste Programmspeicherzelle  $nn + 1$  bzw.  $nn$  eingeleitet werden. Auf der Speicherzelle  $nn$  muß nun das drei Befehlsbyte umfassende Hilfsprogramm EI; RETI beginnen, das die Rückkehr aus allen ISR ausführt. Anstelle eines 3-byte-Sprungbefehls (JP  $nn$ ) kann auch ein Rufbefehl verwendet werden (z. B. Einbytebefehl RSTp), um notwendige Änderungen vorhandener Programme evtl. zu vereinfachen. Des weiteren muß die Eingangsstufe der Logik (Bild 4.2.18) für die Dekodierung des Programmzählerstands  $nn + 2$  (Speicherzelle des zweiten RETI-Bytes) durch entsprechende Brücken vorprogrammiert sein. Die Logik fügt bei Einhaltung der genannten Voraussetzungen nach Dekodierung des entsprechenden Programmzählerstands des zweiten RETI-Bytes (4DH) eine ebenfalls durch Brücken voreinstellbare Anzahl von WAIT-Zuständen in den Befehlslesezyklus (M1-Maschinenzyklus) dieses Bytes ein. Die wirksame H-Einschwingzeit der IEI-IEO-Kette wird somit wiederum verlängert und gestattet den Einsatz von max. 35 Systemelementen bei der Taktfrequenz  $f_c = 2,5$  MHz. Der Zeitverlust durch die Einfügung der WAIT-Zustände ist gegenüber Variante 1 (Bild 4.2.17) weniger bedeutungsvoll, da diese Taktzustände nur bei Auftreten des zweiten RETI-Bytes erzeugt werden. Sie beeinflussen aber dennoch die Leistungsfähigkeit des Rechners. Da die Schaltung nach Bild 4.2.18 nur auf Signale des Systemsteuer- und des Adreßbusses zurückgreift, kann sie ebenfalls in bereits vorhandenen Mikrorechnern nachgerüstet werden. Nachteilig ist bei dieser Variante, daß die Programmierung der ISR-Rücksprünge hardwarespezifisch ist und somit die Anwendung vorhandener Programme (die ohne die genannten Bedingungen erstellt wurden) erschwert wird.

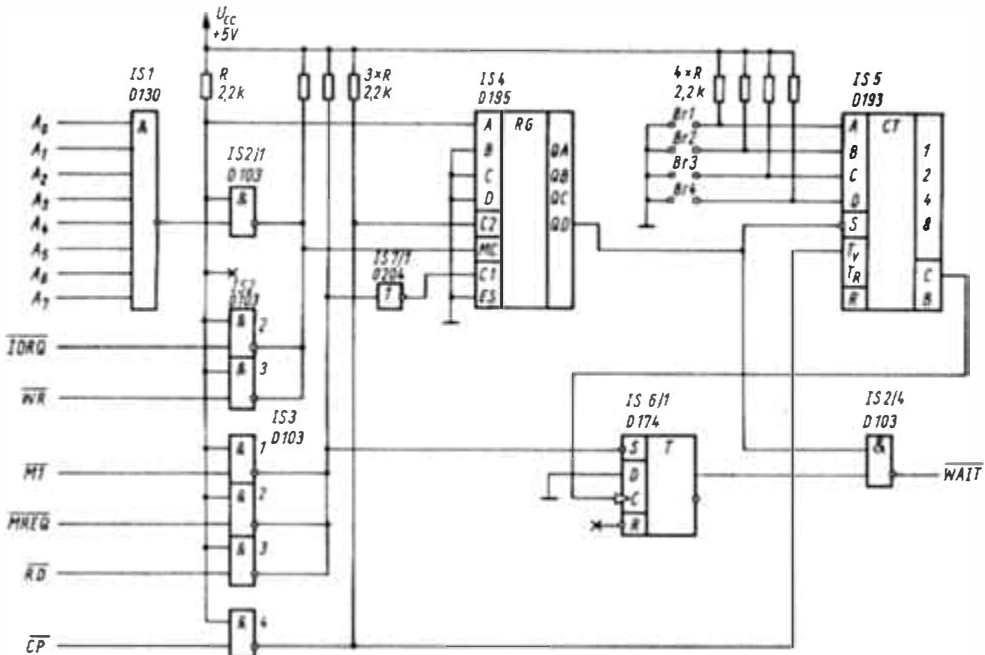


Bild 4.2.19. WAIT-Generator zur Verlängerung der H-Einschwingzeit der Interruptprioritätenkette durch Ausführen eines Vorankündigungsbefehls

Die WAIT-Logik benutzt als Portadresse (Portschreiben) OFFH

Im Bild 4.2.19 ist eine hierzu alternative Schaltungskonzeption dargestellt, die ebenfalls nur bei Ausführung des zweiten RETI-Bytes WAIT-Zustände in den M1-Zyklus (4DH-Lesezyklus) einfügt. Die Auslösung des WAIT-Generators erfolgt aber in diesem Fall in Abhängigkeit von einer Vorankündigung des RETI-Befehls. Die Rückkehr aus einer ISR muß somit durch die Befehlsfolge...; OUT 0FFH; EI; RETI; bzw....; OUT 0FFH; NOP; RETI; ausgeführt werden. Der OUT-Befehl dient zur Vorankündigung des RETI-Befehls und bewirkt das Laden des Schieberegisters IS4. Im dritten Befehlslesezyklus nach dieser Ankündigung wird dann die wiederum vorprogrammierbare Anzahl von WAIT-Zuständen eingeschoben. Der Vorteil dieser Schaltungsanordnung besteht in dem besonders geringen hardwaremäßigen Realisierungsaufwand. Auch diese Variante ist in vorhandenen Rechnern nachrüstbar. Da eine (in diesem Fall die höchstwertige) Portadresse (0FFH) für die Vorankündigung verwendet wird, darf diese ansonsten im Rechner nicht als Ausgabeadresse benutzt werden. Zwar können auch andere Befehle, z. B. LD (nn),A, zur Ankündigung verwendet werden, jedoch erhöht sich hierbei der Schaltungsaufwand. In speziellen Mikrorechnersystemen können auch transparent wirkende Befehle (z. B. LD A,A oder LD B,B u. dgl.) zur Vorankündigung dekodiert werden. Der Nachteil der Variante nach Bild 4.2.19 besteht aber neben der zusätzlichen zeitlichen Belastung der CPU wiederum darin, daß die ISR-Rücksprünge eine hardwareorientierte Programmierung erfordern.

In den zuvor beschriebenen drei Schaltungsvarianten wurde die H-Einschwingzeit gemäß Gl. (4.2.1) durch Einfügen von WAIT-Zuständen (Parameter  $m$ ) vergrößert. Die Schaltungsvariante nach Bild 4.2.20 verlängert diese Einschwingzeit, indem direkt der Systemtakt für die peripheren Elemente nach Dekodierung des zweiten RETI-Bytes 4DH gestreckt wird. Hierzu erzeugt die dargestellte Logik einen neuen Systemtakt  $CP'$ , der den peripheren Systemelementen zugeführt wird. Dieser Takt  $CP'$  folgt dem Systemtakt  $CP$ , wenn kein RETI-Befehl in Bearbeitung ist. Das Zeitverhalten bei Ausführung von RETI ist im Bild 4.2.21 dargestellt. Hierbei wird die fallende Flanke des T4-Taktzustands des Peripherietakts  $CP'$  gegenüber dem Systemtakt  $CP$  bei der Taktfrequenz  $f_c = 2,5$  MHz um  $t_D = 2,4 \mu s$  verzögert. Daraus ergibt sich eine erreichbare Kettenlänge von max. 18 Peripherieelementen. Die Vorteile dieser Variante bestehen vor allem darin, daß keine zusätzlichen WAIT-Zustände die Leistungsfähigkeit des Mikrorechners vermindern und die RETI-Programmierung keinerlei Bedingungen der Hardware unterliegt. Die Schaltungsvariante ist aber in vorhandenen Mikrorechnern nur mit Einschränkung nachrüstbar, da die Taktversorgung der Peripherie verändert werden muß. Die ersten sechs Systemelemente der Kette können direkt mit  $CP$  beschaltet werden, da hier die normale H-Einschwingzeit bei RETI ausreichend ist. Unter diesen Elementen müssen sich alle im Zeitgeberbetrieb arbeitenden IS U857 (evtl. auch U856) befinden, da sonst Fehler bei der Echtzeitbearbeitung auftreten würden. Die im Bild 4.2.20 dargestellte Schaltungskonzeption ist bei Systemtaktfrequenzen  $f_c = 1,75 \dots 2,5$  MHz einsetzbar, da alle Peripherieelemente des Systems U880 eine maximale Halbperiodendauer des Systemtakts von  $t_{W(CP)} = 2 \mu s$  aufweisen. Bei anderen Taktfrequenzen ( $f_c < 1,75$  MHz) ist die Schaltungsanordnung entsprechend zu verändern. Eine weitere Dehnung des Peripherietakts ist nicht möglich, da die Möglichkeit besteht, daß zwei RETI-Befehle (bei Verschachtelungen) aufeinander folgen können und dann die Zeitbedingungen an den Systemelementen nicht mehr gewährleistet wären.

Eine weitere Möglichkeit, das korrekte Einschwingen der Interruptprioritätenkette bei RETI-Bearbeitung zu gewährleisten, besteht darin, die H-Durchlaufzeit ( $t_{DH(O)}$ ) der peripheren Systemelemente gemäß Gl. (4.2.2) zu vermindern. Da die IEI-IEO-Kette aber H-aktiv ist, können nur „vorausschauende“ Umgehungslogiken für den L-Pegel einge-

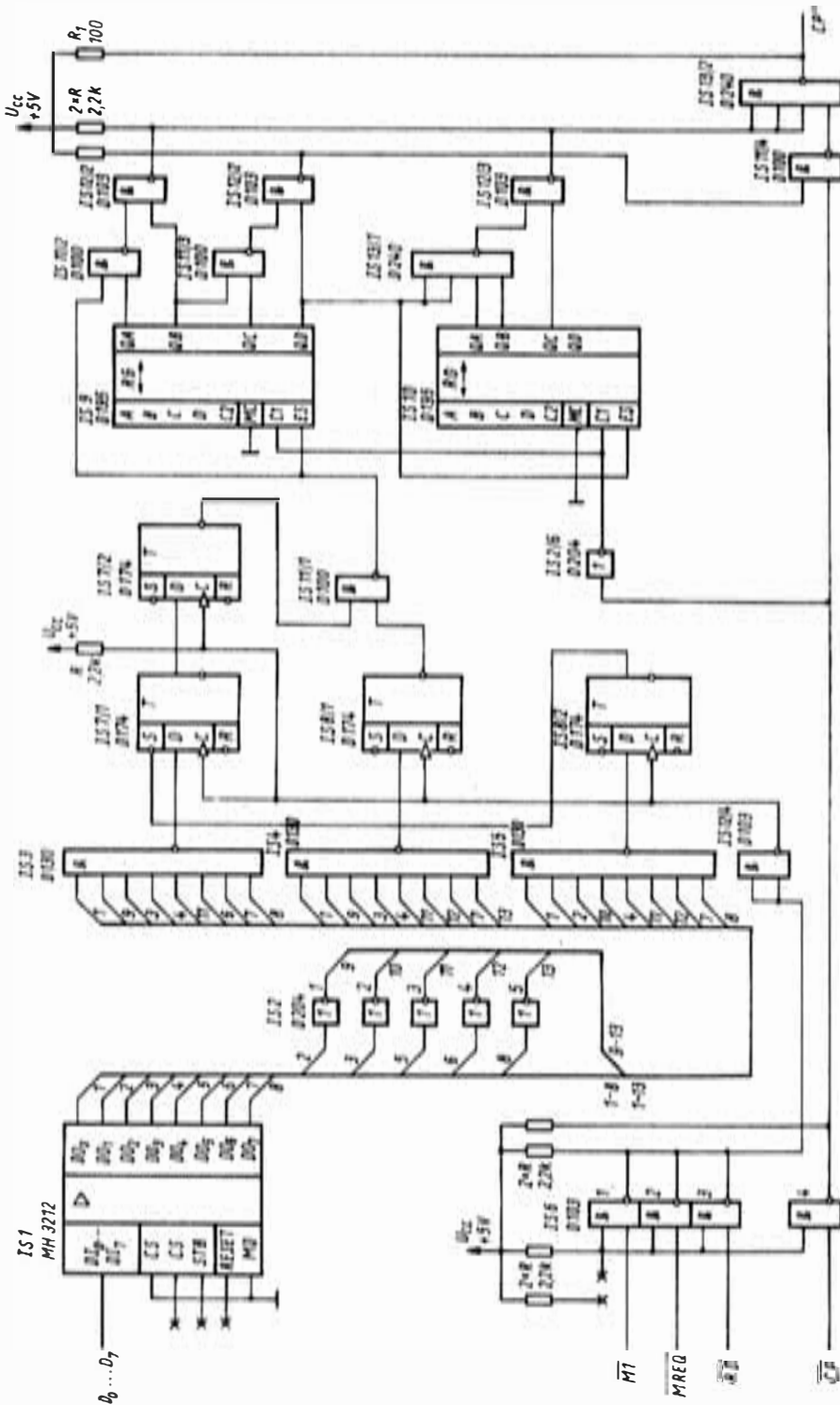


Bild 4.2.20. Logik zur Verlängerung der H-Einschwingzeit der Interruptprioritätenkette durch Streckung des Systemtakt durch peripheren Elemente CP. Systemtakt für peripherie Systemelemente (außer den ersten sechs in der Prioritätenkette)

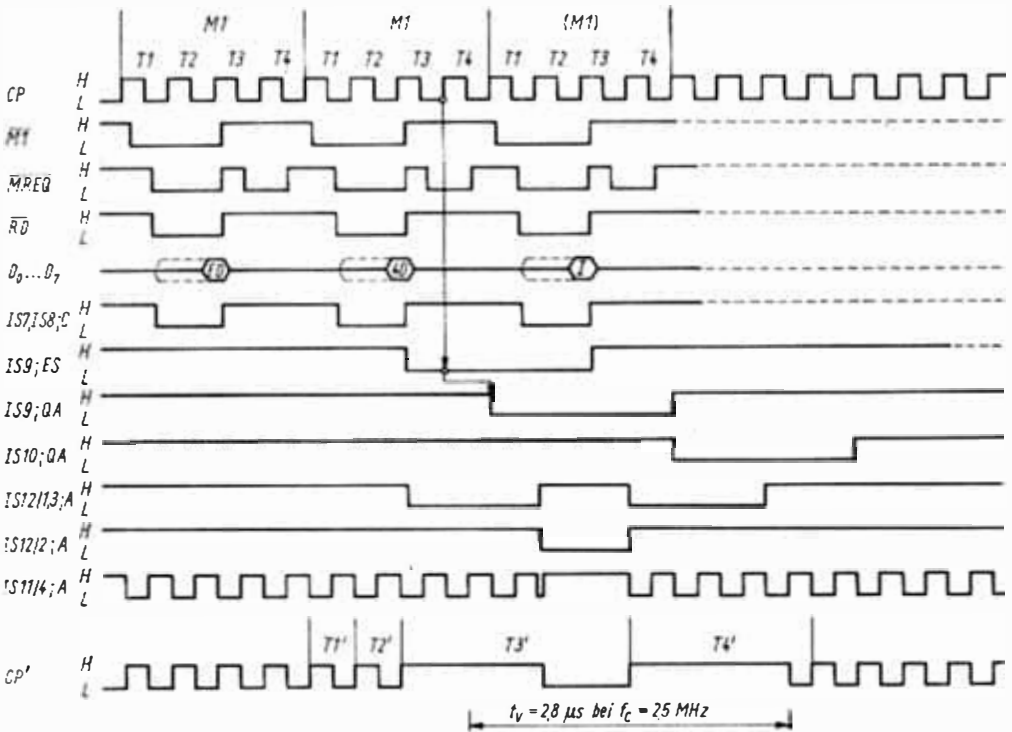


Bild 4.2.21. Zeitverhalten der im Bild 4.2.20 dargestellten Logik

- M1 Maschinenzyklen der CPU [anstelle (M1) kann DMA-Betrieb eingeschoben sein]
- CP Systemtakt (T1, T2, T3, T4 Taktzustände der CPU)
- $\overline{M1}, \overline{MREQ}, \overline{RD}$  Steuersignalausgänge der CPU
- D<sub>0</sub> ... D<sub>7</sub> Belegung des Systemdatenbusses (ED, 4D Befehlsbytes von RETI; I gültiges Op-Kode-Byte des folgenden Befehls)
- CP' von der Logik nach Bild 4.2.20 erzeugter modifizierter Systemtakt

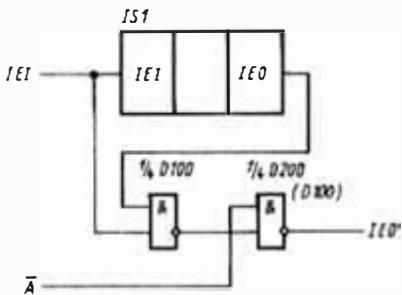


Bild 4.2.22  
Einfache Umgehungslogik mit Steueranschluß  $\bar{A}$   
IS1 peripheres Systemelement (PIO, SIO, CTC)

setzt werden. Eine solche Logik kann bei der Interruptquittierung angewendet werden, um die L-Einschwingzeit zu vermindern. Im Bild 4.2.22 ist eine derartige vereinfachte Umgehungslogik dargestellt. Im Unterschied zur Schaltungsanordnung nach Bild 4.2.11 weist sie die zusätzliche Möglichkeit auf, die Freigabeeingänge IEI der peripheren Elemente mit Hilfe des Signals  $\bar{A}$  zwangsweise auf H-Potential schalten zu können. Wenn nach Erkennen des Steuerbytes EDH (korrektes erstes RETI-Byte) dieses  $\bar{A}$ -Signal aktiv wird ( $\bar{A} = L$ ), dann kann der H-Pegel sich nach dem parallelen Durchlauf durch alle nicht-quittierten Elemente in der gesamten Kette fortsetzen. Im Gegensatz zum korrekten

eingeschwungenen Zustand der Prioritätenkette empfangen dann aber auch Elemente H-Pegel am IEI-Eingang, die sich in der Kette hinter dem höchstwertigen quittierten Element befinden. Nach dem Inaktivwerden des  $\bar{A}$ -Signals ( $\bar{A} = H$ ) schließt sich automatisch ein L-Einschwingvorgang der IEI-IEO-Kette an, an dessen Ende die Prioritätenkette den gültigen eingeschwungenen Zustand aufweist. Im Bild 4.2.23 ist eine Schaltungs-

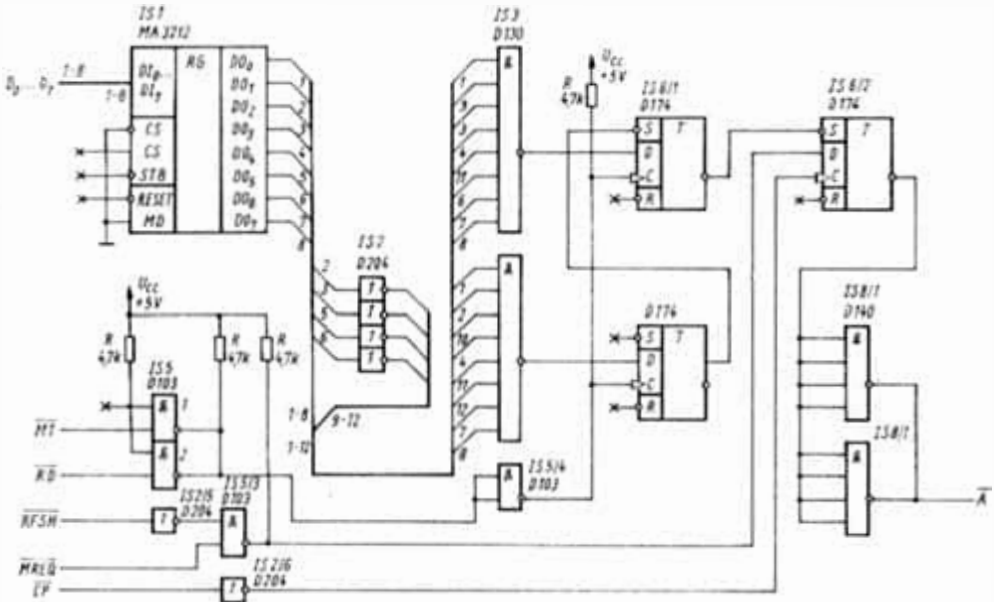


Bild 4.2.23. Steuerlogik zur Verlängerung der H-Einschwingzeit der Interruptprioritätenkette durch Steuersignal  $\bar{A}$

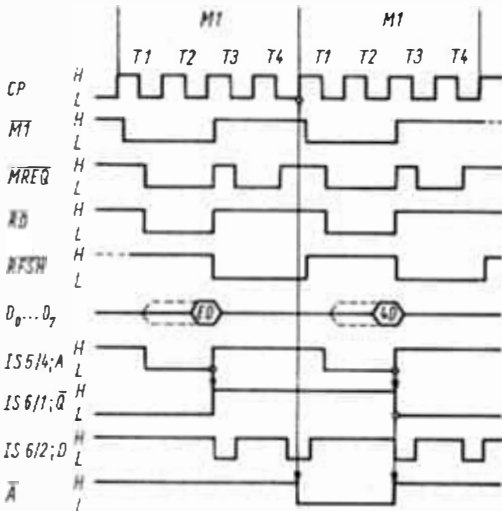


Bild 4.2.24

Zeitverhalten der Steuerlogik nach Bild 4.2.23

- M1 Maschinenzyklen der CPU bei RETI
- CP Systemtakt (T1, T2, T3, T4 Taktzustände der CPU)
- M1, MREQ, RD, RFSH Steuersignale der CPU
- D0 ... D7 Belegung des Systemdatenbusses (ED, 4D Befehlsbytes von RETI)
- $\bar{A}$  Steuersignal zum zwangsweisen H-Schalten der Prioritätenkette

variante angegeben, die ein solches Steuersignal  $\bar{A}$  bei Dekodierung des Befehlsbytes EDH erzeugt. Das zugehörige Zeitverhalten ist im Bild 4.2.24 dargestellt. Da durch das parallele Weitergeben des H-Pegels bei aktivem  $\bar{A}$ -Signal beliebig viele periphere Elemente in der Prioritätenkette freigegeben werden können, wird die Anzahl der in der Kette liegen-

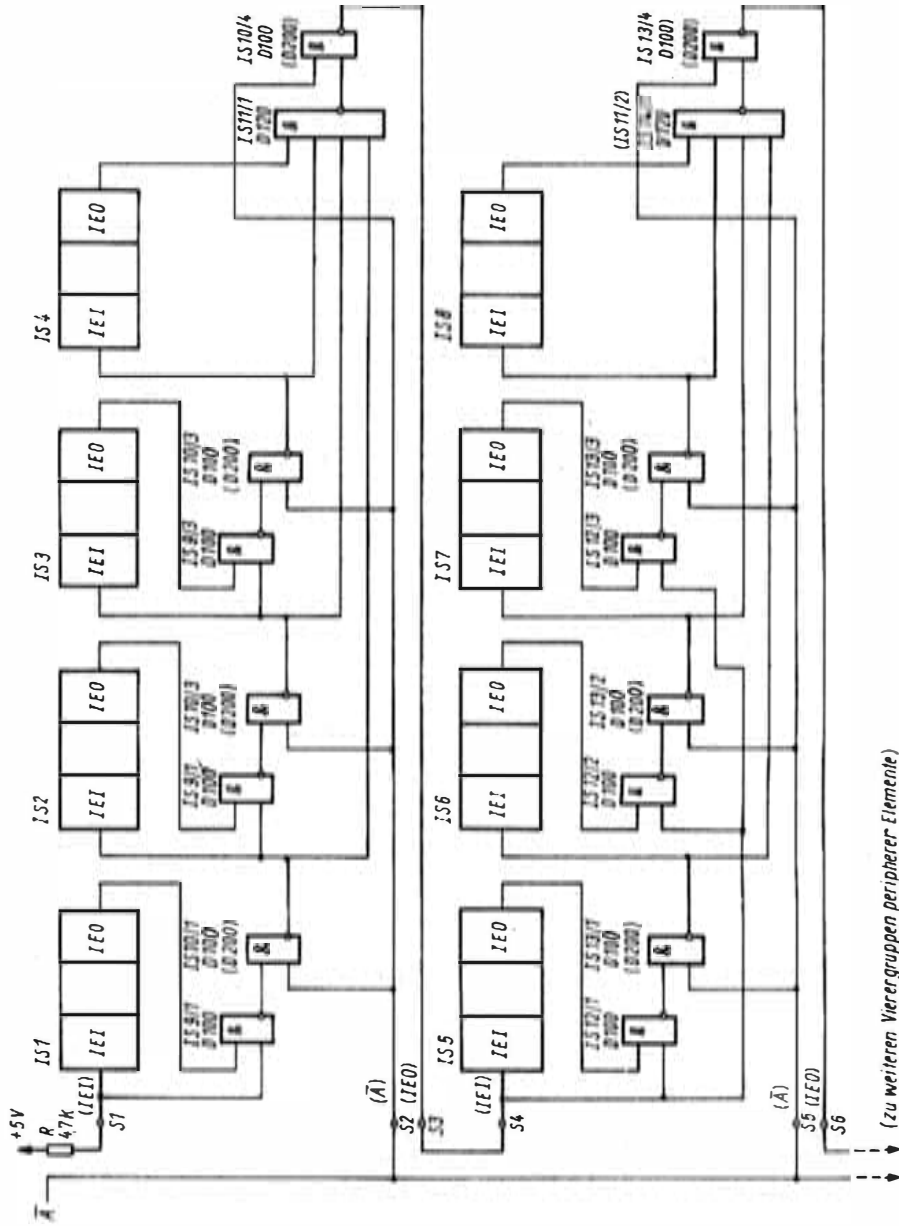


Bild 4.2.25. Erweiterte Umgehungslogik mit Steueranschluß A

JS1 ... JS8 periphere Systemelemente (PIO, SIO, CTC)

S1 ... S6 mögliche Schnittstellen zwischen Steckeinheiten

(zu weiteren Vierergruppen peripherer Elemente)

den Elemente nur durch die L-Verzögerungszeit der Umgehungslogik bestimmt. Nach der Rückflanke des  $\bar{A}$ -Signals ( $\bar{A} = H$ ) steht der Prioritätenkette noch eine Zeit von  $t = 600$  ns für den L-Einschwingvorgang zur Verfügung. Es können also bis zu 19 Systemelemente in der Kette liegen ( $t_{DTTL} = 35$  ns). Da aber die wirksame Zeit für den L-Einschwingvorgang bei der Interruptquittierung (s. Abschn. 4.2.3.2.) ohne Einfügung eines WAIT-Zustands kürzer ist ( $t_{w1} = 430$  ns), wird diese Zeit für die Berechnung der Kettenlänge entscheidend. Somit können bei Anwendung der Umgehungslogik nach Bild 4.2.22 nur max. 14 Elemente in der IEI-IEO-Kette liegen. Bei Anwendung der erweiterten Umgehungslogik nach Bild 4.2.25 erhöht sich diese Zahl auf 25 Elemente (Systemtaktfrequenz  $f_C = 2,5$  MHz). Der Vorteil dieser Variante besteht darin, daß keine zusätzlichen WAIT-Zustände die Leistungsfähigkeit des Mikrorechners beeinflussen und daß die RETI-Programmierung wiederum hardwareunabhängig ist. Die Nachrüstbarkeit in bereits vorhandenen Mikrorechnern ist nur bezüglich der Steuerlogik (s. Bild 4.2.23) gegeben. In der Peripherie wird der Steuereingang  $\bar{A}$  parallel an den Gattern der Umgehungslogik benötigt. Eine zusätzliche Leitung des Steuerbusses ist notwendig.

Die fünf dargestellten Varianten dienen alle zur Lösung des Zeitproblems beim H-Einschwingen der Prioritätenkette während RETI. Sie weisen aber alle unterschiedliche Vor- und Nachteile auf. Die Anwendung eines der Schaltungsvorschläge hängt somit von den konkreten Einsatzbedingungen ab. In Tafel 4.2.2. sind zusammenfassend die Eigenschaften der beschriebenen Varianten dargestellt.

Tafel 4.2.2. Variantenvergleich zur Lösung des Zeitproblems beim H-Einschwingvorgang der Prioritätenkette

Variante	Vorteile	Nachteile
Einfügung von 1 bzw. $n$ WAIT-Zuständen nach Dekodierung des RETI-Bytes 0EDH (Bild 4.2.17)	<ul style="list-style-type: none"> <li>● Nachrüstbarkeit für bestehende Rechner ist gegeben</li> <li>● keine Veränderung des Programms notwendig</li> </ul>	<ul style="list-style-type: none"> <li>● definierte Verlängerung der Abarbeitungszeit bei allen Befehlen mit erstem Op-Kode-Byte 0EDH</li> </ul>
Einfügung von $n$ WAIT-Zuständen nach Dekodierung der RETI-Programmzelle (Bild 4.2.18)	<ul style="list-style-type: none"> <li>● definierte, unbedeutende Verlängerung der Abarbeitungszeit nur bei RETI-Befehl</li> <li>● Nachrüstbarkeit für bestehende Rechner ist gegeben</li> </ul>	<ul style="list-style-type: none"> <li>● hardwarespezifische Programmierung bzw. Programmüberarbeitung notwendig</li> </ul>
Einfügung von $n$ WAIT-Zuständen nach Dekodierung einer Vorankündigung des RETI-Befehls (Bild 4.2.19)	<ul style="list-style-type: none"> <li>● definierte, unbedeutende Verlängerung der Abarbeitungszeit nur bei RETI-Befehl</li> <li>● Nachrüstbarkeit für bestehende Rechner ist gegeben</li> <li>● geringer Schaltungsaufwand</li> </ul>	<ul style="list-style-type: none"> <li>● hardwarespezifische Programmierung bzw. Programmüberarbeitung notwendig</li> <li>● Eingriff (meist unbedeutend) in Systemkonzept des Rechners (Port- oder Memory-adresse)</li> </ul>
Streckung des Systemtakts (Bild 4.2.20)	<ul style="list-style-type: none"> <li>● keine Spezifik bei Programm-erstellung</li> <li>● keine Verlängerung der Abarbeitungszeit bei RETI-Befehl</li> </ul>	<ul style="list-style-type: none"> <li>● Veränderung der Taktversorgung des Rechners (Peripherie)</li> <li>● mittlerer Schaltungsaufwand</li> </ul>
Zwangsweises Freigeben der Prioritätenkette nach Dekodierung des RETI-Bytes EDH (Bilder 4.2.22, 4.2.23)	<ul style="list-style-type: none"> <li>● keine zeitliche Beeinflussung bei RETI-Bearbeitung</li> <li>● keine Spezifik bei Programm-erstellung</li> </ul>	<ul style="list-style-type: none"> <li>● Nachrüstbarkeit in der Peripherie bestehender Rechner i. allg. nicht gegeben (Nachrüstbarkeit der Steuerlogik ist möglich)</li> </ul>

### 4.2.4. Interruptstruktur der peripheren Systemelemente

Die internen Interruptstrukturen aller peripheren Elemente (PIO, SIO, CTC) des Prozessorsystems U880 stimmen in ihren wesentlichen Funktionsgruppen überein. Geringfügige Unterschiede, vor allem bezüglich der softwaremäßigen Überwachung und Steuerung, sind bei den Peripherielementen funktionell bedingt (s. Abschn. 3). Die Interruptstruktur einer interruptfähigen Einheit besteht aus einer Interruptanmeldelogik, einer Quittierungslogik, der RETI-Logik und einer Zeitsteuerung. Die Funktionsgruppen für die Interruptanmeldung und Interruptquittierung mit der zugehörigen Prioritätenlogik (IEI-IEO-Kaskadierung) sind in den Systemelementen entsprechend der Anzahl der selbständigen Interruptmöglichkeiten prinzipiell mehrfach enthalten. Bei den Elementen PIO und CTC entspricht diese Anzahl der Zahl der vorhandenen Kanäle; bei der SIO besitzt jeder Kanal selbst jeweils drei unterschiedliche Interruptquellen. Die Signale der RETI-Logik und der Zeitsteuerung werden im Element für alle interruptfähigen Einheiten gleichzeitig benutzt.

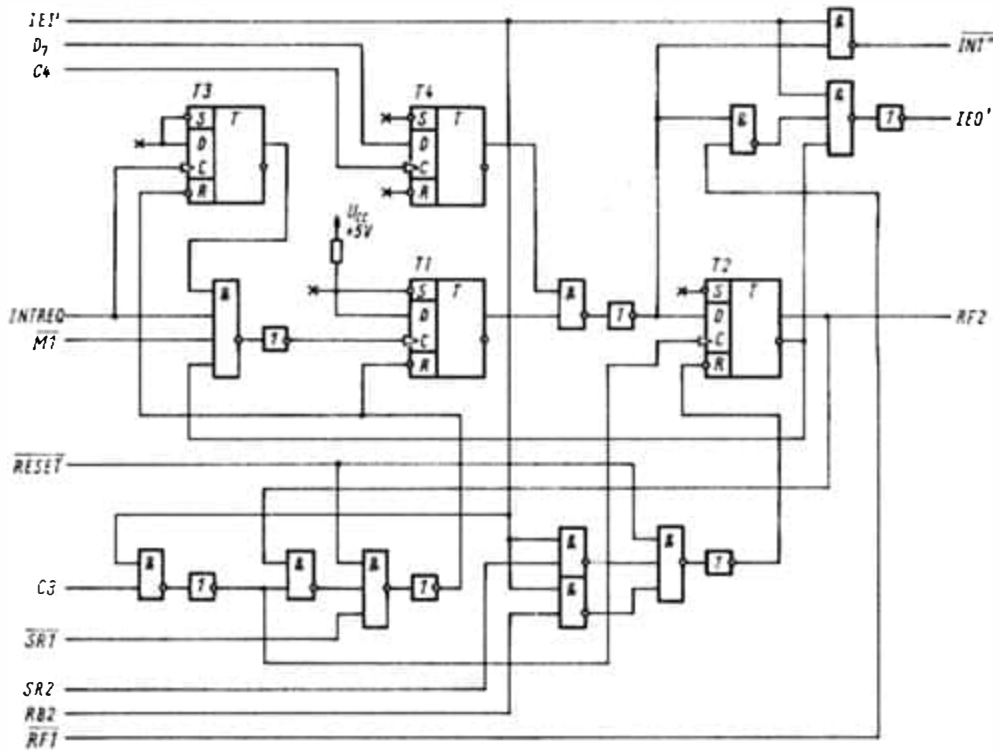


Bild 4.2.26. Schaltungsanordnung zur Nachbildung der Interruptanmelde- und -quittierungslogik der peripheren Systemelemente

IEI', IEO', INT' Interruptsignale eines Kanals eines peripheren Elements

Im Bild 4.2.26 ist eine Schaltungsanordnung dargestellt, die die prinzipielle Wirkung der Interruptanmeldungs- und der Quittierungslogik mit diskreten Schaltkreisen nachbildet. Der Trigger 1 stellt hierbei das Interruptanmeldeflipflop dar. Der Takteingang dieses Flipflops wird durch den aktiven Zustand des M1-Signals und durch Setzen des Triggers 2 (Interruptbearbeitungsflipflop) gesperrt. Eine anhängige Interruptforderung

(Signal INTREQ) wird somit während eines M1-Maschinenzyklus nicht angenommen, damit die Interruptprioritätenkette im Interruptquittierungszyklus (M1-Zyklus) Zeit für den Einschwingvorgang hat. Des weiteren wird eine Anforderung nicht akzeptiert, wenn die betreffende Einheit bereits eine Interruptbearbeitung durchführt. Damit wird gewährleistet, daß eine in Bearbeitung befindliche ISR nicht durch die gleiche Routine unterbrochen werden kann. Bei der parallelen Ein-/Ausgabe-Einheit (PIO) erfolgt eine zusätzliche Sperrung der Interruptanmeldung (Trigger 3), wenn kein Pegelwechsel am internen Interrupteingang (z. B. STB-Eingang) auftritt. Nach dem Setzen des Interruptanmeldeflipflops wird die anhängige Interruptanforderung bei softwaremäßig gesetztem Interruptfreigabeflipflop (Trigger 4) und aktivem Freigabeeingang IEI' des Kanals (bzw. der Einheit bei SIO) über den Ausgang  $\overline{\text{INT}}$  zur CPU weitergeleitet. Das Rücksetzen des Interruptanmeldeflipflops erfolgt bei Aktivierung des  $\overline{\text{RESET}}$ -Signals (wird bei der PIO intern erzeugt) und bei der Interruptquittierung dieser anhängigen Forderung. Das Flipflop kann aber auch softwaremäßig durch Einschreiben eines Steuerworts rückgesetzt werden. Bei der PIO geschieht dieses Rücksetzen nach Einschreiben eines Interruptsteuerworts mit gesetztem Bit D<sub>4</sub> (Maske wird angekündigt). Bei der Quittierung der Interruptanmeldung erfolgt das Setzen des Interruptbearbeitungsflipflops (Trigger 2), wenn die Interrupteinheit die höchste Wertigkeit nach dem Einschwingen der Prioritätenkette aufweist (IEI' = H). Abhängig vom gesetzten Zustand dieses Bearbeitungsflipflops wird die nachfolgende Interruptprioritätenkette durch inaktives IEO'-Signal für die Dauer der Bearbeitung gesperrt. Das Rücksetzen des Interruptbearbeitungsflipflops erfolgt nach dem Erkennen des zweiten RETI-Bytes durch die RETI-Logik des Peripherielements unter der Bedingung, daß die Einheit zum Zeitpunkt der Abfrage des IEI'-Eingangs (fallende Flanke des Taktzustands T4) die höchstwertige quittierte Interruptanmeldung aufweist (IEI' = H). Das Ausgangssignal des Triggers 2 stellt somit ein Kombinations-signal dar, das eine Information über das Aufrufen (Quittierung der Interruptforderung) und das Verlassen (durch zugehörigen RETI-Befehl) der kanalspezifischen ISR beinhaltet. Bei Aktivierung des  $\overline{\text{RESET}}$ -Signals des entsprechenden Elements erfolgt ebenfalls das Rücksetzen des die Interruptbearbeitung anzeigenden Triggers. Das Systemelement SIO hat des weiteren eine softwaremäßige Möglichkeit, mit Hilfe eines Steuerworts (im Bild 4.2.26 durch Signal SR2 dargestellt) das Interruptbearbeitungsflipflop rückzusetzen. Ein derartiges Steuerwort hat somit die gleiche Wirkung für dieses Element, als sei ein RETI-Befehl auf dem Datenbus erkannt worden. Damit ist beispielsweise die Möglichkeit gegeben, daß eine bearbeitete ISR nochmals durch die gleiche oder durch niederwertigere Routinen unterbrochen werden kann. Dagegen kann ein derartiges Steuerwort, wie bereits erwähnt, zur Rücksetzung des höchstwertigen bearbeiteten internen Bearbeitungsflipflops dienen und somit zu einer Rückkehr aus der ISR verwendet werden. Hierbei entfällt dann der zeitkritische Einschwingvorgang in der Interruptprioritätenkette, weil der SIO-Eingang IEI nicht bewertet wird. Des weiteren erleichtert diese Maßnahme die Anwendung der sehr komplexen und leistungsfähigen seriellen Ein-/Ausgabe-Einheit (SIO) in Mikroprozessorsystemen, die keine systemweite Prioritätenkette aufweisen (z. B. System 8080 [4]). Bei einem derartigen Einsatz könnte dann die wirkungsvolle interne Interruptprioritätenkette trotzdem angewendet werden.

Der Interruptanmeldeausgang  $\overline{\text{INT}}$ ' einer Einheit wird bei Vorhandensein einer freigegebenen zwischengespeicherten Interruptforderung aktiv. Da innerhalb der Systemelemente die einzelnen interruptfähigen Einheiten (Kanäle) ebenfalls in der (schaltkreisintern weitergeführten) Interruptprioritätenkette liegen, ergibt sich für den Anmeldeausgang  $\overline{\text{INT}}$  des Peripherielements eine Verschachtelung der Kanalmeldungen. In

Gl. (4.2.3) ist die logische Funktion des  $\overline{INT}$ -Ausgangs eines Elements mit zwei interruptfähigen Kanälen dargestellt.

$$\overline{INT} = \overline{INTA} \cdot IEI + \overline{INTB} \cdot IEOA; \tag{4.2.3}$$

$\overline{INT}$  Interruptausgang des Systemelements

$\overline{INTA}$  Interruptforderungen der Kanäle

$\overline{INTB}$  Interruptforderungen der Kanäle

IEI Freigabeeingang des Systemelements

IEOA Freigabeausgang des Kanals A (IEOA = IEIB) (Kanal A hat somit höhere Priorität als Kanal B).

Das Interruptfreigabesignal am Ausgang IEO' einer Einheit folgt der in Gl. (4.2.4) dargestellten logischen Funktion:

$$IEO' = IEI' \cdot RF2 \cdot (\overline{INT'} + RF1); \tag{4.2.4}$$

IEO' Freigabeausgang der Einheit

IEI' Freigabeeingang der Einheit

RF2 Kombinationssignal, das Interruptbearbeitungszustand der Einheit anzeigt

$\overline{INT'}$  Interruptforderung (nichtquittiert) der Einheit

RF1 Signal, das die Dekodierung eines ersten RETI-Bytes (EDH) anzeigt.

Für den Fall, daß in der Einheit eine nichtquittierte Interruptanmeldung zum Prozessor vorliegt (angezeigt durch aktives  $\overline{INT'}$ ), wird der IEO'-Ausgang nach Erkennen des auf dem Datenbus ankommenden Steuerbytes EDH in Abhängigkeit vom Freigabeeingang IEI' für die Dauer dieses Maschinenzklus aktiviert. Die Bedingung zur Freigabe der Interruptprioritätenkette durch höherwertigere, aber nichtquittierte Einheiten bei der Rückkehr aus Interruptbearbeitungsroutinen wird hierdurch erfüllt. Da in einem peripheren Systemelement mehrere Interrupteinheiten vorhanden sind, müssen bei der Darstellung der logischen Funktion des Systemelementausgangs IEO [s. Gl. (4.2.5)] die angemeldeten bzw. quittierten Interruptzustände der einzelnen Einheiten konjunktiv verknüpft werden:

$$IEO = IEI \cdot \overline{RF2A} \cdot \overline{RF2B} \cdot (\overline{INTA} \cdot \overline{INTB} + RF1); \tag{4.2.5}$$

IEO Freigabeausgang des Systemelements

IEI Freigabeeingang des Systemelements

$\overline{RF2A}$  } Interruptbearbeitungszustände der Kanäle

$\overline{RF2B}$  }

$\overline{INTA}$  } Interruptanmeldungen der Kanäle

$\overline{INTB}$  }

RF1 Signal, das Dekodierung von EDH anzeigt.

Die in Gl. (4.2.5) dargestellte logische Funktion bezieht sich auf ein Systemelement mit zwei interruptfähigen Kanälen (PIO).

Die in der Schaltungsnachbildung der Anmelde- und Quittierungslogik (Bild 4.2.26) verwendeten Steuer- und Taktsignale werden in den peripheren Systemelementen in einer RETI-Logik und einer Interruptzeitsteuerung gebildet. Die Erzeugung der RETI-Signale (RB2, RF1) geschieht hierbei prinzipiell wie in der im Bild 4.2.27 gezeigten TTL-Schaltungsanordnung. Mit Hilfe des Dekoders wird die Auswertung der auf dem Systemdatenbus liegenden Befehlsbytes vorgenommen. Die Abspeicherung der Dateninformation erfolgt jeweils mit steigender Systemtaktflanke während der Befehlslesezyklen (M1 Maschinenzklus, op-code fetch). Damit wird gesichert, daß die gültigen Daten (eingeschwungener Zustand des Busses) wie beim Prozessor mit der steigenden Flanke des T3-Zustands abgefragt werden. Mit einem in der Interruptzeitsteuerung gebildeten Takt erfolgt danach die Abspeicherung der dekodierten Befehlsbytes EDH, 4DH, CBH in

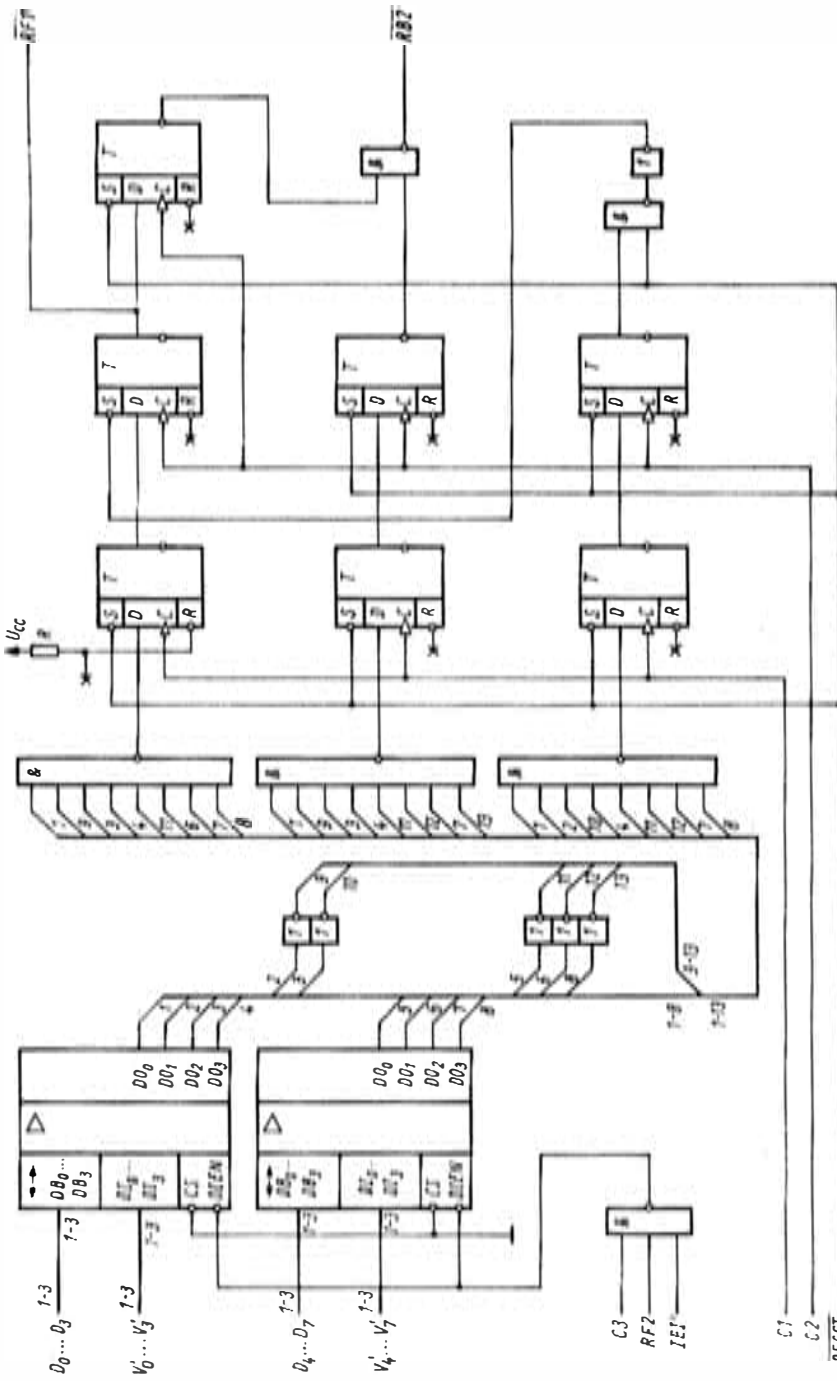


Bild 4.2.27. Schaltungsanordnung zur Nachbildung der peripherielementeinternen RETI-Logik und Datenbussteuerung

\$V\_0 \dots V\_7\$, Inhalt des kanalinternen Interruptvektorregisters  
(Trigger 15 bis 21: Nummerierung von oben nach unten, von links nach rechts; Gatter 9: 3-Eingangs-NAND)

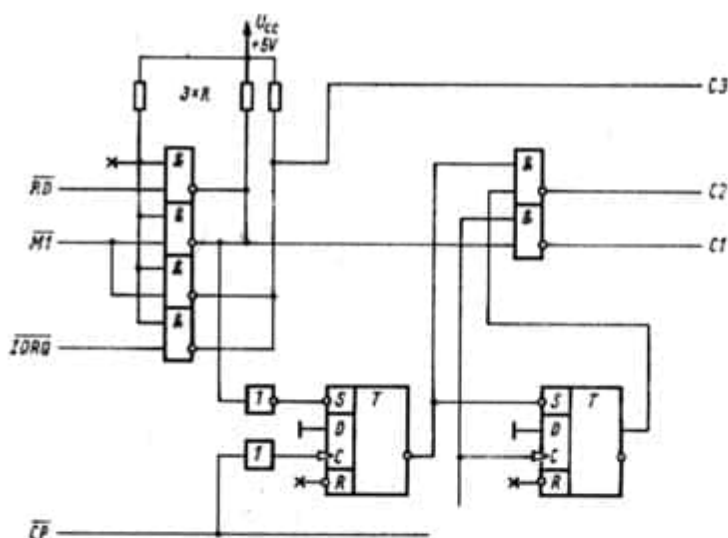


Bild 4.2.28. Schaltungsanordnung zur Nachbildung der elementeinternen Interruptzeitsteuerung

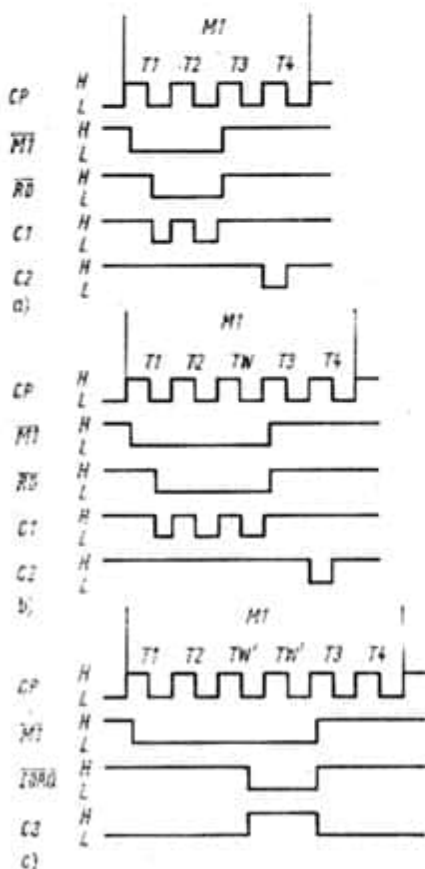


Bild 4.2.29

Zeitverhalten der Zeitsteuerung nach Bild 4.2.28

a) im M1-Befehlslesesyklus

b) im M1-Befehlslesesyklus mit eingefügten zusätzlichen WAIT-Zuständen (bzw. -Zustand)

c) im M1-Interruptquittierungszyklus

M1 Maschinenzklus der CPU

CP Systemtakt (T1, T2, TW, TW', T3, T4 Taktzustände der CPU, TW' automatisch von der CPU eingefügter WAIT-Zustand)

$\overline{MI}$ ,  $\overline{RD}$ ,  $\overline{IORQ}$  Steuersignaleingänge des peripheren Elements

C1, C2, C3 Taktsignale der schaltkreisinternen Interruptsteuerung

den Triggern 18 ... 21. Bei Rücksetzen des Triggers 18 (Byte EDH wurde dekodiert) wird das  $\overline{\text{RF1}}$ -Signal aktiviert. Dieses Signal ermöglicht das Freigeben der Prioritätenkette durch interruptfordernde, nichtquittierte Einheiten entsprechend dem Zeitverhalten nach Bild 4.2.15 (s. Abschn. 4.2.3.3.). Folgt im nächsten Befehlslesezyklus auf das dekodierte Befehlsbyte EDH das Byte 4DH, so wird der Trigger 19 rückgesetzt. Gleichzeitig wird, ausgelöst vom Byte EDH, der Trigger 21 rückgesetzt. Das dadurch aktivierte Signal  $\overline{\text{RB2}}$  zeigt damit an, daß ein RETI-Befehl auf dem Datenbus lag und somit die Rückkehr aus der aktuell höchstwertigen ISR erfolgt. Das dekodierte zwei RETI-Befehlsbyte ( $\overline{\text{RB2}} = \text{L}$ ) setzt daraufhin in Verbindung mit dem aktiven IEI-Signal einer Interrupteinheit ( $\text{IEI}' = \text{H}$ ) das entsprechende Interruptbearbeitungsflippflop zurück. Das Zeitverhalten der Systemelemente gemäß Bild 4.2.16 (s. Abschn. 4.2.3.3.) wird damit erfüllt.

Im Bild 4.2.27 ist zusätzlich prinzipiell die Übernahme des Datenbusses durch ein gerade quittiertes Peripherieelement im Interruptquittierungszyklus dargestellt (Gatter 9). Diese Datenbusübernahme dient zum Aussenden des aktuellen Interruptvektors.

Die zur Ansteuerung der in den Schaltungsvarianten der Interruptlogik verwendeten Taktsignale (C1, C2, C3) werden in den Peripherieelementen, z. T. in komplexen Schaltungsteilen, in Verbindung mit weiteren Steuersignalen erzeugt. Die im Bild 4.2.28 dargestellte Zeitsteuerung bildet die entsprechenden ausschließlich zu Interruptsteuerungszwecken verwendbaren Taktsignale C1 ... C3 in TTL-Schaltungstechnik nach. Das Zeitverhalten dieser Signale zeigt Bild 4.2.29.

#### 4.2.5. Programmbearbeitung

Die ISR-Programmierung stellt im Zusammenhang mit dem Zeitaufwand der Registerrettung, der Verzögerung der Bedienung des interruptfordernenden Elements, dem Verschachtelungsgrad der ISR und der Struktur des Hauptprogramms einen leistungsbestimmenden Parameter der gesamten Interruptbearbeitung dar.

Die Rettung der im Hauptprogramm benötigten Arbeitsregister wird durch die Struktur und den Befehlssatz der CPU U880 wirkungsvoll unterstützt. Diese Registerrettung in den ISR ist in Programmen notwendig, in denen die Einschachtelung einer oder mehrerer ISR an einer beliebigen Stelle erfolgen kann, und bewirkt, daß nach Rückkehr aus der Bearbeitungsroutine der alte Programmbearbeitungszustand erreicht wird. Das zuvor unterbrochene Programm kann somit mit den korrekten Registerinhalten weiter bearbeitet werden.

Prinzipiell sind zwei unterschiedliche Möglichkeiten zur Registerrettung im System U880 vorhanden. Zunächst besteht die Möglichkeit, zu Beginn einer ISR (das gilt grundsätzlich für alle Unterprogrammeinschachtelungen) alle im laufenden Programm benötigten Register (also auch die alternative Registerbank der CPU) in den Schreib/Lese-Speicher des Mikrorechners auszulagern. Der Befehlssatz des Prozessors U880 besitzt hierzu die leistungsfähige Gruppe der PUSH- und POP-Befehle, mit denen je Operation jeweils ein 16-bit-Register (bzw. ein 8-bit-Registerpaar) in den Stackbereich des Speichers ausgelagert bzw. rückgespeichert werden kann. Diese Variante der Registerrettung muß dann angewendet werden, wenn eine Programmverschachtelung in mehreren Ebenen (z. B. durch sich unterbrechende ISR) erfolgen kann. Der Zeitaufwand für die Rettungsoperationen ist relativ gering, läßt aber keine unmittelbare, unverzügliche Bedienung des interruptfordernenden Elements zu. Die andere prinzipielle Möglichkeit der Registerrettung beruht auf der Ausnutzung der *alternativen* CPU-Registerbank für die allgemein verwendbaren Registerpaare BC, DE, HL sowie für den Akkumulator A und das Flag-

register F. Diese alternative Registerbank der CPU U880 kann durch zwei Einbytebefehle (EXX; EXAF) gegen die Hauptregisterbank ausgetauscht werden. Es wird somit eine beinahe unmittelbare Bedienung der Interruptanforderung ermöglicht. Voraussetzungen für die Anwendung dieser Registerrettungsart ist, daß einerseits im Hauptprogramm bei Interruptfreigabe die alternative Registerbank keine zu rettenden Informationen beinhaltet und daß andererseits keine weiteren ISR bzw. Unterprogramme, die diese Variante der Registerrettung verwenden, in die bearbeitete ISR eingeschaltet werden können (Interruptsperrung während Bearbeitung).

Keine Registerrettung ist in ISR notwendig, wenn durch die Struktur des Hauptprogramms (z. B. Lage der Interruptfreigabe durch Befehl EI) gesichert ist, daß keine Registerinformationen verlorengehen.

Ein wichtiger Gesichtspunkt bei der interruptbezogenen Programmerstellung ist der Aufbau des Hauptprogramms. Im folgenden sollen deshalb zwei prinzipielle Strukturvarianten genannt werden. Nach der Programminitialisierung, die i. allg. unter Interruptsperrung erfolgt (Initialisierung der peripheren Elemente, Bestimmung von Programmkonstanten, Setzen von Variablen, Reservierung von Speicherbereichen, Setzen des SP, der Interruptmode u. dgl.), kann sich eine Programmbearbeitung (z. B. zyklische Bedienung langsamer Peripherie, Polling) oder letztlich eine HALT-Warteschleife anschließen. Im ersteren Fall kann die Interruptfreigabe grundsätzlich oder in bestimmten Teilprogrammen erfolgen. Die durch Vektorinterrupts erzeugten ISR bedienen die schnellen peripheren Schnittstellen. In der zweiten Variante – das Programm endet in einer HALT-Schleife – müssen die gesamten Mikrorechnerfunktionen durch Interruptanmeldungen angefordert werden. Das beinhaltet sowohl die Bedienung der Peripherie als auch die Aufrechterhaltung bestimmter, notwendiger Rechnerfunktionen. Diese nicht peripheriebezogenen Anforderungen müssen durch Zeitgeberinterrupts befriedigt werden. Die Rechenzeit des Prozessors wird somit optimal ausgelastet.

Bei der Programmerarbeitung ist der Aufbau der eigentlichen ISR grundsätzlich von den vorstehenden Bedingungen abhängig. Hinsichtlich der Interruptfreigabe in den ISR und den sich daraus ergebenden ISR-Verschachtelungsmöglichkeiten sind drei grundlegende ISR-Programmiervarianten denkbar. Erfolgt in der ISR (zu Beginn oder während der Programmbearbeitung der ISR) der Befehl EI, so ist eine Verschachtelung durch höherwertige Elemente möglich. Angewendet wird diese Programmiervariante, wenn interruptmäßig hochpriorisierte periphere Elemente eine unverzügliche ISR-Bedienung erfordern und niederwertigere ISR unterbrechen müssen (z. B. Bedienung eines SIO-Kanals bei leerem Ausgangs-FIFO im Sendebetrieb). Nach Rückkehr in das Hauptprogramm ist interruptmäßig der Ausgangszustand (Interruptfreigabe) wiederhergestellt. Wenn der Befehl EI erst vor dem Rückkehrbefehl RETI in der ISR erfolgt, dann kann die ISR nicht unterbrochen werden. Eine ISR-Verschachtelung ist nicht möglich, da die Wirkung des EI-Befehls (Interruptfreigabe) systembedingt um einen Befehl verzögert wird. Dadurch wird erreicht, daß der Stackbereich des Schreib/Lese-Speichers nicht unnötig durch die Verschachtelungen belastet wird. Nach der Rückkehr ins Hauptprogramm ist wiederum der Interruptfreigabezustand vorhanden. Als dritte Variante besteht weiterhin die Möglichkeit, keinen EI-Befehl in die ISR einzubauen. Damit ist wiederum keine ISR-Verschachtelung möglich. Andererseits ist nach Rückkehr ins Hauptprogramm die Interruptanmeldung gesperrt. Beispielsweise können an dieser Stelle dann wichtige Hauptprogrammteile, die nicht unterbrochen werden dürfen (z. B. um Registerrettung in den ISR zu vereinfachen), eingeschoben werden, bevor eine erneute Interruptfreigabe durch einen Befehl EI erfolgt.

Weitere Interruptprogrammiermöglichkeiten, insbesondere system- und hardware-spezifische ISR-Varianten (z. B. in Minimalsystemen ohne RAM-Speicher, Vereinfachung

der Einschwingvorgänge der Prioritätenkette bei der SIO), sind denkbar. Ebenso können verschiedene der genannten Programmierungsarten miteinander verkoppelt werden. Insgesamt bietet das System U880 eine Vielzahl von leistungsfähigen problemorientierten Möglichkeiten zur Interruptbedienung und -bearbeitung.

### 4.3. Beschreibung der DMA-Eigenschaften

#### 4.3.1. Einführung

Die Aufgaben des DMA-Betriebs (DMA: direct memory access) bestehen darin, den Zugriff auf Daten des Speichers bzw. auf Peripheriedaten unter Umgehung des Mikroprozessors zu ermöglichen. Da hierbei die Befehlsabarbeitung der CPU entfällt, kann ein sehr schneller Zugriff auf die Daten erreicht werden. Der DMA-Betrieb gestattet deshalb die Ausführung von zeitkritischen Datenzugriffen und die zeitgünstige Übertragung von großen Datenmengen.

Während der Bearbeitung einer DMA-Anforderung durch eine entsprechende (Hardware-) Einheit muß der Systembus des Rechners (Adreß-, Daten- und Steuerbus) für diese DMA-Einheit verfügbar sein. Der Prozessor bzw. die CPU-Baugruppe des Mikrorechners gibt zu diesem Zweck nach Empfang der Anforderung ( $\overline{\text{BUSRQ}}$ -Eingang der CPU) den Systembus frei, d. h., die entsprechenden Buslinien floaten ( $\overline{\text{Tristateausgänge}}$ ). Dieser Zustand wird vom Prozessor durch den aktiven Pegel am Bestätigungsausgang  $\overline{\text{BUSAK}}$  angezeigt. Nach Empfang dieser Quittierung kann die DMA-Einheit den Datenzugriff mit ihrem eigenen, zeitgünstigen Zeitverhalten vornehmen. Da der Steuerbus des Rechners verfügbar ist, können Daten (z. B. aus dem Speicher) auch über die peripheren Schnittstellen des Mikrorechners (z. B. PIO, SIO) übertragen werden.

Weitere leistungsfähige Anwendungen des DMA-Betriebs ergeben sich in Mikrorechnern, die aufgrund des Einsatzes von statischen RAM-Speicherelementen die automatisch von der CPU erzeugten Refreshzyklen (in den M1-Maschinenzyklen) nicht benutzen. Da während dieser Refreshzeiten keine Operationen im Rechner ausgeführt werden, kann der Systembus der CPU-Baugruppe (Buspuffer sind in diesem Fall erforderlich) bei aktivem  $\overline{\text{RFSH}}$ -Signal in den floatenden Zustand gebracht werden. Die DMA-Einheit kann daraufhin für diese Zeit (aktives  $\overline{\text{RFSH}}$ ) einen Datenzugriff organisieren. Dieser während der Refreshzyklen wirkende DMA-Betrieb (refresh cycle stealing) wirkt im Mikrorechnersystem transparent, d. h., er belastet die Rechenzeit des Prozessors prinzipiell nicht.

Hauptanwendungsfälle des DMA-Betriebs sind der Transfer von großen Datenblöcken, der beispielsweise beim Nachladen von schnellen Bildwiederhol Speichern der evtl. am Mikrorechner angeschlossenen Monitore (Bildschirmeinheiten) auftritt, und die Organisation des Zugriffs auf einen Datenspeicher durch mehrere CPU in Mehrprozessorsystemen (multiprocessing).

#### 4.3.2. Möglichkeiten der DMA-Bearbeitung

##### 4.3.2.1. Einordnung im System

Die CPU U880 reagiert auf eine Anmeldung einer Busanforderung über den Prozessor-eingang  $\overline{\text{BUSRQ}}$  mit einer Unterbrechung der laufenden Programmbearbeitung. Diese Programmunterbrechung hat gegenüber Interruptanmeldungen über die  $\overline{\text{NMI}}$ - und  $\overline{\text{INT}}$ -

Linien eine höhere Priorität, d. h., sie wirkt vorrangig. Die Wirkung des DMA-Betriebs – schneller Zugriff auf Rechnerdaten – wird somit durch die Struktur der CPU unterstützt.

Ein DMA-Betrieb kann beim Prozessor U880 nach jedem vollständig bearbeiteten Maschinenzklus eingeschoben werden. Die Busanforderungslinie  $\overline{\text{BUSRQ}}$  der CPU wird hierzu prozessorintern mit jeder steigenden Flanke des letzten Systemtaktzustands aller Maschinenzyklen abgefragt. Der aktive Zustand dieses Prozessoreingangs ( $\overline{\text{BUSRQ}} = \text{L}$ ) bewirkt, daß das CPU-interne  $\text{BUSRQ}$ -Flipflop zu diesem Zeitpunkt gesetzt wird. Vom Zustand dieses Flipflops abhängig erfolgt die Ausgabe des Quittierungssignals  $\overline{\text{BUSAK}}$  über den zugehörigen CPU-Ausgang und das Floaten des Systembusses (Daten und Adreßbus, Steuersignale  $\overline{\text{MREQ}}$ ,  $\overline{\text{IORQ}}$ ,  $\overline{\text{WR}}$ ,  $\overline{\text{RD}}$ ,  $\overline{\text{RFSH}}$ ) nach Beendigung des letzten Maschinenzklus. Bei gesetztem  $\text{BUSRQ}$ -Flipflop erfolgt dann die weitere Abfrage des Zustands der Anmeldeinie  $\overline{\text{BUSRQ}}$  mit jeder steigenden Systemtaktflanke. Nach Auswertung eines inaktiven Signals am  $\overline{\text{BUSRQ}}$ -Eingang wird im Prozessor das Busanforderungsflipflop wieder rückgesetzt. Nach der folgenden fallenden Taktflanke wird der Quittierungsausgang  $\overline{\text{BUSAK}}$  wieder inaktiv, und im nächsten Taktzustand setzt der Prozessor die zuvor unterbrochene Programmfolge fort. Geschah die Einschachtelung des DMA-Betriebs nach Abschluß eines vollständigen Befehls, so werden bei der Rückkehr zunächst die CPU-internen Interruptanmeldeflipflops abgefragt und, abhängig von ihren Zuständen, eine Programmverschachtelung (Interruptbearbeitung) durchgeführt.

Ein transparenter DMA-Betrieb kann von einer DMA-Einheit ausgeführt werden, ohne daß eine Busanforderung an die CPU gesendet wird. Die Quittierung und somit die Information, daß der Systembus verfügbar ist, müssen für die entsprechende DMA-Einheit durch den aktiven Zustand des Signals  $\overline{\text{RFSH}}$  erfolgen.

#### 4.3.2.2. Schneller Datenzugriff

Nach der Quittierung einer Busanforderung durch das Bestätigungssignal  $\overline{\text{BUSAK}}$  der CPU U880 kann eine DMA-Einheit auf den Systembus des Rechners zugreifen. Ein solcher DMA-Betrieb kann zu einer in bezug auf Datenquelle, Datensenke und Blocklänge wahlfreien Datenübertragung dienen. Die Suche nach einer bestimmten, maskierbaren Bytefolge im Speicher des Mikrorechners wäre eine hierzu alternative Möglichkeit des DMA-Betriebs.

In allen diesen Varianten muß die DMA-Einheit den Datenverkehr im Rechner durch Erzeugung der entsprechenden Steuersignale ( $\overline{\text{MREQ}}$ ,  $\overline{\text{IORQ}}$ ,  $\overline{\text{RD}}$ ,  $\overline{\text{WR}}$ ) organisieren. Das entsprechende Zeitverhalten muß den Zugriffszeiten der Speicherelemente und der peripheren Elemente angepaßt sein. Es entspricht deshalb i. allg. dem CPU-Zeitverhalten. Für Datenübertragungen muß die DMA-Einheit den Systembus sowohl für den Sender (Datenquelle) als auch für den Empfänger (Datensenke) nachbilden können. Die DMA-Einheit kann dann beispielsweise auch Daten des Speichers über vorhandene Standardschnittstellen (z. B. PIO) übertragen. Bei speziellen DMA-Einheiten kann je nach der notwendigen Übertragungsrichtung die Sender- oder Empfängerschnittstelle der externen Peripherie direkt angepaßt werden.

#### 4.3.2.3. Transparenter Betrieb

In dieser DMA-Betriebsart (refresh cycle stealing) gibt es prinzipiell keine Busanforderung an die CPU. Die Freigabe des Systembusses durch den Prozessor muß hardware-

mäßig bei aktivem Zustand des CPU-Steuersignals  $\overline{\text{RFSH}}$  realisiert werden. Hierzu müssen die für den DMA-Betrieb notwendigen Buslinien der CPU gepuffert werden. Im Bild 4.3.1 ist eine entsprechende Schaltungsanordnung dargestellt. Die DMA-Einheit empfängt das Steuersignal RFSH als Bestätigungssignal ( $\overline{\text{BAO}}$ ) für den floatenden Zustand der Buslinien. Sie kann daraufhin den Systembus übernehmen und einen eigenen Datenverkehr organisieren. Aufgrund der begrenzten Zugriffszeit der DMA-Einheit (jeweils im Refreshzyklus) kann bei Verwendung des Standardzeitverhaltens der CPU je DMA-Zugriff nur ein Datenzugriff erfolgen, vorausgesetzt, im Mikrorechner sind Standard-Speicherbauelemente mit einer Zugriffszeit von  $t_{\text{ACC}} = 450 \text{ ns}$  und die peripheren Systemelemente (PIO, SIO, CTC) eingesetzt. Anderenfalls (bei Einsatz langsamer Speicher, z. B. CMOS-RAM) kann der DMA-Zugriff durch eine Busanforderung ( $\overline{\text{BUSRQ}} = \text{L}$ ) erweitert werden. Ein derartiger DMA-Betrieb arbeitet dann nicht mehr vollständig transparent; dennoch weist er im Mikrorechnersystem bei Datenübertragungen eine große Leistungsfähigkeit auf.

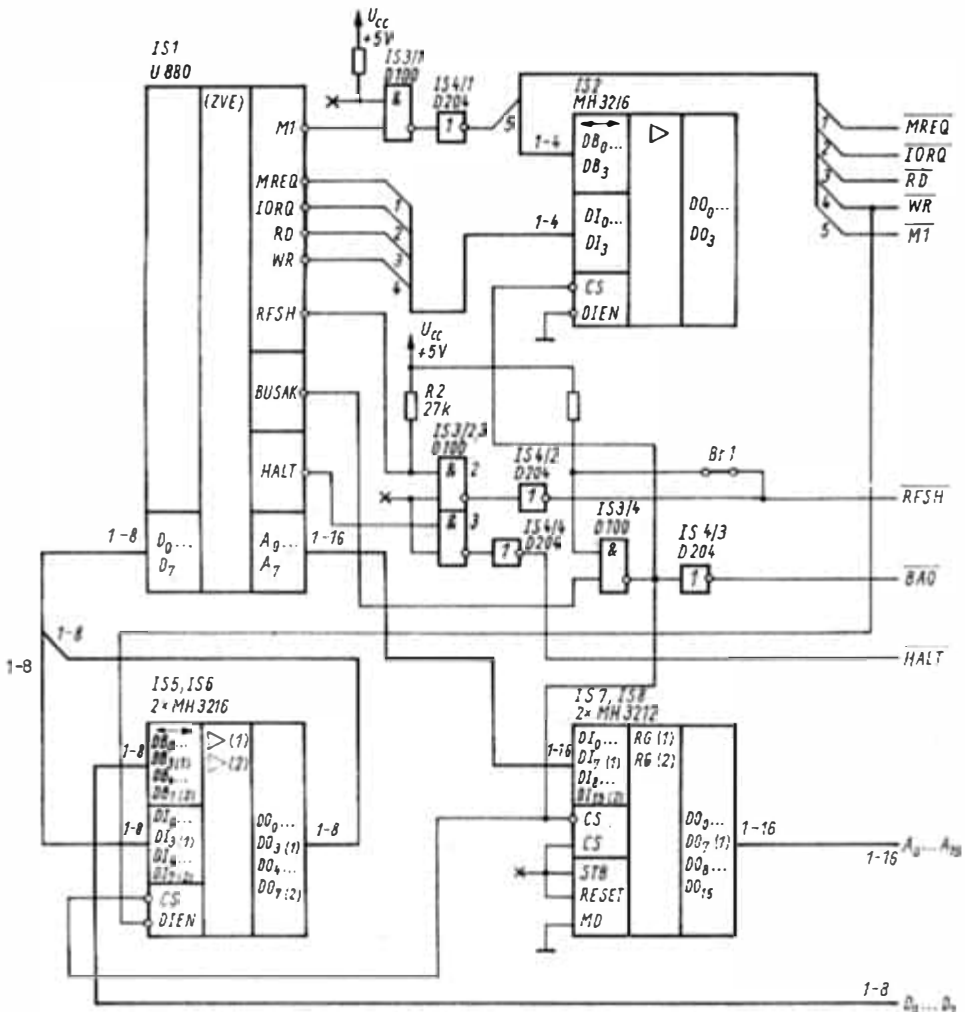


Bild 4.3.1. CPU-Buspufferung für DMA-Betrieb

Br 1 realisiert die Steuerung für die transparente Betriebsart

Der transparente DMA-Betrieb wird i. allg. nur vorgenommen, wenn im Rechner keine dynamischen RAM-Speicherzellen verwendet werden, die die Refreshzyklen der CPU benötigen. Bei speziellen Anwendungen (transparente, aber nicht zeitkritische Übertragung von Daten) ist auch der Einsatz eines externen Refreshzählers denkbar, der nach erfolgtem Auffrischen der dynamischen Speicherzellen die RFSH-Zyklen zeitweise für den DMA-Betrieb freigibt.

### 4.3.3. Eigenschaften von DMA-Einheiten

#### 4.3.3.1. Priorität

Beim Einsatz mehrerer DMA-Einheiten in einem Mikrorechner muß bezüglich der Busanforderung und -übernahme eine Rangfolge festgelegt werden. Hierzu bietet sich ähnlich wie bei der Interruptbearbeitung eine Kaskadierungskette an. Da ein DMA-Betrieb nur nach einer entsprechenden Quittierung ( $\overline{\text{BUSAK}}$  der CPU;  $\overline{\text{RFSH}}$  bei transparentem Betrieb) erfolgen kann, bieten sich diese Signale als Freigabesignal für die DMA-Einheiten an. Im Bild 4.3.2 ist das Blockschaltbild einer derartigen Anordnung dargestellt. Die höchstwertige Einheit in dieser L-aktiven  $\overline{\text{BAI}}-\overline{\text{BAO}}$ -Kaskadierungskette empfängt zuerst das Quittierungssignal  $\overline{\text{BAO}}$  der CPU-Baugruppe. Bei einer DMA-Anforderung in dieser Einheit wird die zugehörige DMA-Bearbeitung vorgenommen. Anderenfalls wird das aktive L-Signal zum Ausgang  $\overline{\text{BAO}}$  des ersten Elements durchgeschaltet und gibt die nachfolgende DMA-Einheit frei. Prinzipiell kann auf diese Weise das aktive Busbestätigungssignal in der gesamten Kaskadierungskette durchgeschaltet werden.

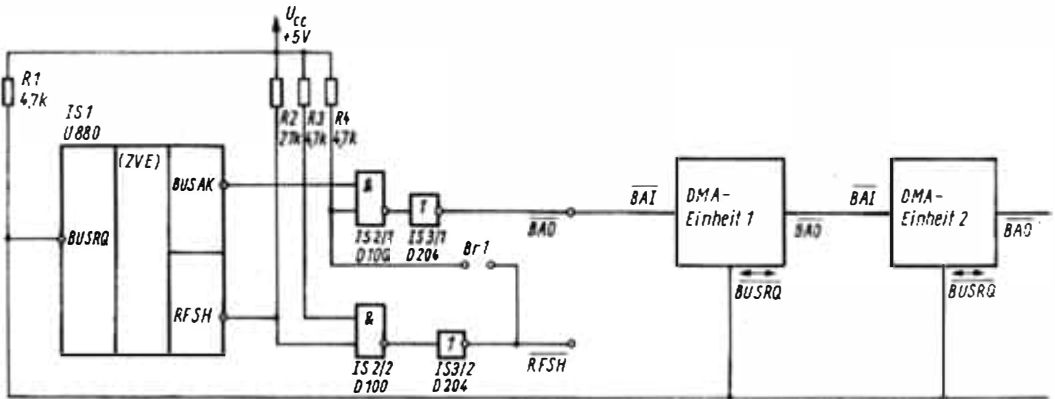


Bild 4.3.2. Blockschaltbild einer DMA-Prioritätenkette

Damit die DMA-Bearbeitung einer niederwertigeren Einheit nicht durch eine höherwertigere DMA-Einheit unterbrochen wird, kann in den DMA-Baugruppen ebenfalls der Zustand der Anmeldeleine  $\overline{\text{BUSRQ}}$  überwacht werden. Hierdurch ist es möglich, wahlweise eine Verschachtelung von DMA-Bearbeitungen zuzulassen.

Da bei einer DMA-Einheit mit transparenter Betriebsart die nutzbare Zugriffszeit bereits durch die Dauer des Refreshzyklus begrenzt ist, sollte diese die höchste Priorität in der Kette aufweisen, um weitere Verzögerungen beim Durchschalten des Quittierungssignals in der Kette zu vermeiden. Da diese Einheit sowieso transparent wirkt, treten keine Beeinflussungen in der Prioritätsauswahl auf.

4.3.3.2. Busanforderung und -quittierung

In den nicht transparent wirkenden DMA-Betriebsarten geschieht eine Busanforderung durch eine DMA-Einheit über die Anmeldelinie  $\overline{BUSRQ}$  zur CPU-Baugruppe. Diese Anforderung einer Einheit kann bei Vorhandensein einer DMA-Kaskadierungskette abhängig von der Priorität und den Verschachtelungsbedingungen verzögert werden. Der Prozessor U880 fragt den Zustand des Busanforderungseingangs  $\overline{BUSRQ}$  mit der steigenden Flanke jedes letzten Systemtaktzustands der Maschinenzyklen ab. Bei Erkennung einer Anmeldung ( $\overline{BUSRQ} = L$ ) wird ein prozessorinternes Flipflop gesetzt, der CPU-Ausgang  $\overline{BUSAK}$  aktiviert ( $\overline{BUSAK} = L$ ) und der CPU-Systembus nach Beendigung dieses letzten Taktzustands hochohmig geschaltet. Das entsprechende CPU-Zeitverhalten ist im Bild 4.3.3 dargestellt. In Mikrorechnern, die CPU-Buspuffer aufweisen, müssen Puffer mit Tristateausgangsstufen eingesetzt werden (in **kleinen** Systemen evtl. auch Open-collector-Stufen), die mit Hilfe des Bestätigungssignals  $\overline{BUSAK}$  ebenfalls in einen inaktiven Zustand gebracht werden können. Somit wird gesichert, daß die CPU-Baugruppe den Rechnersystembus bei quittierter Busanforderung ( $\overline{BUSAK} = \overline{BAO} = L$ ) freigibt.

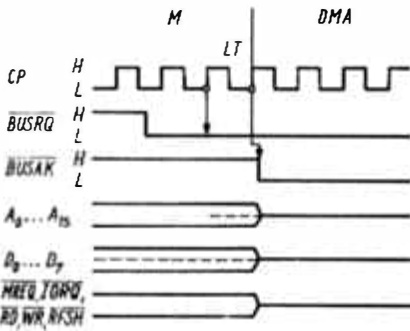


Bild 4.3.3

Zeitverhalten der CPU bei Busfreigabe (DMA-Betrieb)

- M beliebiger Maschinenzyklus der Programmbearbeitung
- DMA Busfreigabe durch die CPU und Möglichkeit des DMA-Betriebs
- CP Systemtakt
- $\overline{BUSRQ}$  Busanforderungseingang der CPU
- $\overline{BUSAK}$  Quittierungsausgang der CPU
- $A_0 \dots A_{15}$  Belegung des Adreßbusses durch die CPU
- $D_0 \dots D_7$  Belegung des Datenbusses durch die CPU
- $\overline{MREQ}, \overline{TORQ}, \overline{RD}, \overline{WR}, \overline{RFSH}$  Belegung der Steuersignale der CPU

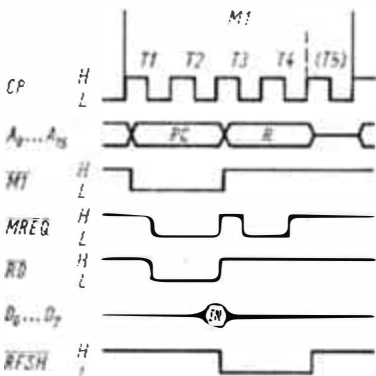


Bild 4.3.4

Standardzeitverhalten der CPU in M1-Befehlslesezyklen

- M1 Befehlslesezyklus der CPU
- CP Systemtakt (T1, T2, T3, T4, T5 Taktzustände der CPU, T5-Taktzustand dient zu internen logischen Entscheidungen – z.B. Überprüfen eines Flags – und wird nur bei einigen Befehlen – z.B. RST, RNZ, DJNZ – eingenommen)
- $A_0 \dots A_{15}$  Belegung des Adreßbusses der CPU (PC Programmzählerstand; R Inhalt des Refreshregisters, während T5 floatet  $A_0 \dots A_{15}$ )
- $\overline{M1}, \overline{MREQ}, \overline{RD}, \overline{RFSH}$  Steuersignalausgänge der CPU
- $D_0 \dots D_7$  Belegung des Systemdatenbusses an der CPU (IN gültige Op-Kode-Daten) aus der Speicherbaugruppe des Rechners

In der transparent wirkenden DMA-Betriebsart (refresh cycle stealing) erfolgt unabhängig von dem Bearbeitungszustand der DMA-Einheit automatisch in jedem Refreshzyklus eine hardwaremäßig erzeugte Busanforderung. Für diese Busanforderung eignet sich direkt das CPU-Steuersignal  $\overline{RFSH}$ , mit dem die externen Buspuffer der CPU-Baugruppe in den hochohmigen (floatenden) Zustand geschaltet werden. Das Signal  $\overline{RFSH}$  wirkt somit gleichzeitig als Quittierungssignal für die DMA-Einheit ( $\overline{RFSH} = \overline{BAO}_{CPU}$ ), indem es den freigegebenen Zustand der CPU-Bustreiber anzeigt. Bild 4.3.4 zeigt das

CPU-Standardzeitverhalten bei M1-Maschinenzyklen. Reicht bei bestimmten Anwendungsfällen die Zeit des RFSH-Zyklus für den DMA-Zugriff nicht aus, so kann mit einer  $\overline{\text{BUSRQ}}$ -Forderung die Zugriffszeit erweitert werden. Diese Möglichkeit besteht, da der Refreshzyklus am Ende des M1-Maschinenzyklus erfolgt und somit den letzten Taktzyklus ( $\overline{\text{BUSRQ}}$ -Anmeldung) umfaßt. Der letzte Taktzyklus stellt also eine interne Operation dar, bei der ebenfalls der Systembus verfügbar ist. Der DMA-Betrieb wirkt in diesem Fall in bezug auf die Ausführung nicht mehr transparent.

#### 4.3.3.3. Rückkehr aus DMA-Betrieb

In den mit der  $\overline{\text{BUSRQ}}$ -Anforderung eingeleiteten DMA-Betriebsarten erfolgt prozessorseitig die Abfrage der Anmeldelinie ( $\overline{\text{BUSRQ}}$ -Eingang der CPU) bei laufender DMA-Bearbeitung mit jeder steigenden Systemtaktflanke. Erkennt die CPU einen inaktiven Pegel an ihrem  $\overline{\text{BUSRQ}}$ -Eingang ( $\overline{\text{BUSRQ}} = \text{H}$ ), so folgt nach der nächsten fallenden Taktflanke das Rückstellen des Bestätigungsausgangs ( $\text{BUSAK} = \text{H}$ ), und im nächsten Taktzustand übernimmt die CPU wieder den Systembus und setzt die zuvor unterbrochene Programmbearbeitung fort. Der DMA-Betrieb ist damit beendet. Das hierbei auftretende CPU-Zeitverhalten ist im Bild 4.3.5 dargestellt.

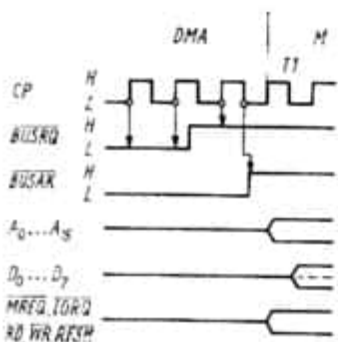


Bild 4.3.5

Zeitverhalten der CPU bei Übernahme der Busse  
(Beendigung eines DMA-Betriebs)

M	in Befehlsabarbeitung folgender Maschinenzyklus
CP	Systemtakt (TI Taktzustand der CPU)
$\overline{\text{BUSRQ}}$	Busanforderungseingang der CPU
$\overline{\text{BUSAK}}$	Quittierungsausgang der CPU
$A_0 \dots A_{15}$	Belegung des Adreßbusses durch die CPU
$D_0 \dots D_7$	Belegung des Datenbusses durch die CPU
$\overline{\text{MREQ}}, \overline{\text{IORQ}}, \overline{\text{RD}}, \overline{\text{WR}}, \overline{\text{RFSH}}$	Belegung (Gültigkeit) der Steuersignale der CPU

In der transparenten Betriebsart wird der Systembus nach dem Inaktivwerden des  $\overline{\text{RFSH}}$ -Signals wieder für die CPU-Baugruppe freigegeben. Der DMA-Zugriff auf den Bus muß zu diesem Zeitpunkt beendet sein, damit nicht mehrere Sender (CPU-Baugruppe, DMA-Einheit) auf dem Systembus gegeneinander arbeiten. Im folgenden Maschinenzyklus der CPU können in bezug auf Aussendung der Adreßsignale CPU-Baugruppe Abweichungen im Zeitverhalten gegenüber dem Standardzeitverhalten auftreten, da die H-Verzögerungszeit des  $\overline{\text{RFSH}}$ -Signals ( $t_{\text{DIRRF}} = 150 \text{ ns}$ ) ein verzögertes Aussenden der Adreßsignale bewirken kann. Dieses abweichende Zeitverhalten muß im Systemkonzept des Mikrorechners berücksichtigt werden.

#### 4.3.4. Einfluß auf Programmierung

Ein DMA-Betrieb kann prinzipiell durch eine externe oder interne programmgestützte Anforderung ausgelöst werden. Bei einem durch die Programmbearbeitung aufgerufenen DMA-Zyklus sind i.allg. die zeitlichen Bedingungen der Programmunterbrechung im Rechner bekannt. Der DMA-Betrieb kann somit direkt durch die Programmierung des Rechners gesteuert werden. Bei externen DMA-Anforderungen (z. B. bei der Datenüber-

nahme durch eine weitere CPU in Mehrprozessorsystemen) sind diese Bedingungen durch die auslösende Einheit zu berücksichtigen. Eine externe DMA-Forderung könnte aber auch indirekt über eine Interruptforderung aus der Peripherie erfolgen. In der zugehörigen ISR könnte dann durch die Rechner-CPU die den aktuellen Bedingungen entsprechende Programmierung und Freigabe der DMA-Einheit ausgeführt werden.

Interruptanmeldungen können aber auch hierzu alternativ von der DMA-Einheit ausgelöst werden, um dem Prozessor nach Beendigung eines DMA-Betriebs anzuzeigen, daß das Programm unterbrochen war und ein Blockaustausch im Speicher stattfand, bzw. daß andere Aktivitäten von der DMA-Einheit im Rechner ausgeführt wurden. Die DMA-Einheit muß in diesem Fall die Interruptstruktur der peripheren Systemelemente aufweisen (z. B. durch Einsatz eines CTC in der DMA-Einheit als Interruptgenerator).

In der transparenten DMA-Betriebsart erfolgt seitens der DMA-Einheit keine Beeinflussung des Programmablaufs durch zeitliche Unterbrechungen. Rückmeldungen über DMA-Aktivitäten (evtl. auch durch den Prozessor vorprogrammiert) können wieder über die Anmeldung von Interrupts vorgenommen werden.

## 5. Ergänzungselemente eines Mikrorechners

Unter dieser Bezeichnung sollen alle die Bauelemente eines typischen Mikrorechners verstanden werden, die nicht zu den eigentlichen Elementen der Schaltkreisfamilie U880 gehören. Art und Anzahl dieser Ergänzungselemente sind hierbei wesentlich von der Konfiguration des jeweiligen Rechners abhängig.

In diesem Abschnitt sollen insbesondere mikrorechnerspezifische Ergänzungsbaulemente beschrieben werden. Hierzu gehören Standardspeicher, Bustreiber und Chip-select-Dekoder. Die darüber hinaus in Mikrorechnern eingesetzten Bauelemente sind typisch TTL-IS der Reihen D10 und D20. Für die Zukunft ist abzuschätzen, daß weitere Funktionsgruppen der Mikrorechner zu spezifischen Ergänzungselementen zusammengefaßt werden (z. B. Taktgeneratoren, Takttreiber).

Festwertspeicher (ROM) werden als Programm- und Konstantenspeicher verwendet, da ihr Inhalt bei Ausfall der Versorgungsspannung nicht verlorengeht. Schreib/Lese-Speicher dienen zur Zwischenspeicherung von größeren Datenmengen (falls die Register der CPU nicht ausreichen). Darüber hinaus kann im RAM die Organisation des CPU-externen Stack (Kellerspeicher) erfolgen. Je nach Anwendungsfall können für diese Aufgaben statische, dynamische oder CMOS-Speicherelemente eingesetzt werden.

Da die MOS-Elemente eines Mikrorechners ausgangsseitig üblicherweise nur über eine begrenzte Steuerfähigkeit (fan-out) verfügen, müssen in größeren Systemen bipolare Interfacelemente eingesetzt werden, die dann die Daten-, Adreß- und Steuersignalverteilung im Rechner vornehmen. Hierzu müssen diese Treiber und Zwischenspeicher am Ausgang Tristateverhalten aufweisen.

Zu den mikrorechnerspezifischen Ergänzungselementen sollen ebenfalls Dekoderbauelemente gerechnet werden, die typisch in größeren Speicherbaugruppen zur Chipauswahl verwendet werden.

### 5.1. Speicherbauelemente

Zur zweiten Leistungsklasse Mikroprozessoren (System U880) existiert eine Reihe von Standardspeichern. Diese Elemente sind meist in nSGT hergestellt und bezüglich Pegel und Geschwindigkeit an die Anforderungen des Prozessors U880 angepaßt.

#### 5.1.1. Festwertspeicher

Als Festwertspeicher sind die beiden pinkompatiblen ROM U505D und EPROM U555C vorgesehen. Sie weisen folgende charakteristischen Merkmale auf:

- Speicherkapazität 8 Kbit, byteorganisiert zu  $1\text{ K} \times 8$
- Zugriffszeit  $t_{\text{ACC}} \leq 450\text{ ns}$
- nSGT-MOS-Technologie
- U505 eine Betriebsspannung +5 V

- U555 drei Betriebsspannungen + 5 V, - 5 V, + 12 V
- Tristateausgangsstufen
- TTL-Kompatibilität aller Ein- und Ausgänge
- 24poliges DIL-Gehäuse nach TGL 26713.

In der Anwendung unterscheiden sich die Typen dadurch, daß für die IS U555 neben der Betriebsspannung + 5 V die Spannungen - 5 V und + 12 V zusätzlich zur Verfügung gestellt werden müssen. Dafür hat man den Vorteil, daß man die IS mehrfach programmieren und mit UV-Licht löschen kann. Die Spannung - 5 V dient als Substratvorspannung

Tafel 5.1.1. Grenzwerte der IS U505 und U555

Kenngröße	Kurzzeichen	min.	max.	Einheit
Betriebsspannung <sup>1)</sup>	$U_{CC}$	-0,3	7	V
Betriebsspannungen <sup>2)</sup>	$U_{DD}$	-0,3	20	V
	$U_{CC}$	-0,3	15	V
	$U_{SS}$	-0,3	15	V
Programmierspannung <sup>2)</sup>	$U_{PR}$	-0,3	32	V
Eingangsspannung <sup>1)</sup>	$U_I$	-0,3	7	V
Eingangsspannung <sup>2)</sup>	$U_I$	-0,3	15	V
Betriebstemperatur	$\vartheta_a$	0	70	°C
Lagertemperatur	$\vartheta_{Lg}$	-55	125	°C

<sup>1)</sup> nur für U505; Spannungen auf  $U_{SS}$  bezogen

<sup>2)</sup> nur für U555; Spannungen auf  $U_{BB}$  bezogen

Tafel 5.1.2. Statische Kennwerte der IS U505 und U555

Kenngröße	Kurzzeichen	min.	typ.	max.	Einheit
Betriebsspannung	$U_{CC}$	4,75	5	5,25	V
Betriebsspannung <sup>2)</sup>	$-U_{BB}$	4,75	5	5,25	V
Betriebsspannung <sup>2)</sup>	$U_{DD}$	11,4	12	12,6	V
H-Eingangsspannung <sup>1)</sup>	$U_{IH}$	2,4			V
H-Eingangsspannung ( $A_n, O_n, \overline{CS}$ ) <sup>2)</sup>	$U_{IH1}$	3,0		$U_{CC} + 0,5$	V
H-Eingangsspannung ( $\overline{CS}$ ) <sup>2)</sup>	$U_{IH2}$	11,4		12,6	V
L-Eingangsspannung	$U_{IL}$	-0,3		0,8	V
H-Ausgangsspannung bei $I_{OH} = -0,4 \text{ mA}^1)$ bzw. $I_{OH} = -1,0 \text{ mA}^2)$	$U_{OH}$	2,4			V
L-Ausgangsspannung bei $I_{OL} = 1,8 \text{ mA}^1)$ bzw. $I_{OL} = 1,6 \text{ mA}^2)$	$U_{OL}$			0,4	V
Stromaufnahme <sup>1)</sup>	$I_{CC}$			120	mA
Stromaufnahme <sup>2)</sup>	$-I_{BB}$			45	mA
	$I_{CC}$			10	mA
	$I_{DD}$			65	mA
Programmierstromaufnahme bei $U_{PR} = 26 \pm 1 \text{ V}^2)$	$I_{PR1}$			20	mA
Programmierstromaufnahme bei $U_{PR} = 1 \text{ V}^2)$	$-I_{PR2}$			3	mA
Eingangskapazität <sup>1)</sup>	$C_I$			10	pF
Eingangskapazität <sup>2)</sup>	$C_I$			6	pF
Ausgangskapazität <sup>1)</sup>	$C_O$			15	pF
Ausgangskapazität <sup>2)</sup>	$C_O$			12	pF

<sup>1)</sup> für U505

<sup>2)</sup> für U555

Alle Spannungen auf  $U_{SS}$  bezogen.

und steuert die MOS-Transistoren in den Zustand als Anreicherungstransistoren. Deshalb darf diese Spannung ( $U_{BB}$ ) nicht später als 10 ms gegenüber den anderen beiden Betriebsspannungen ( $U_{CC}$ ,  $U_{DD}$ ) zugeschaltet bzw. nicht früher als 10 ms als die anderen abgeschaltet werden. Damit werden große interne Querströme vermieden, die sonst zur Zerstörung der IS führen würden.

Tafel 5.1.3. Dynamische Kennwerte der IS U505 und U555

Kenngröße	Kurzzeichen	max.	Einheit
Zugriffszeit ( $A_0 \dots A_9$ )	$t_{ACC}$	450	ns
Selektionszeit ( $\overline{CS}$ )	$t_{CO}$	120	ns
Deselektionszeit ( $\overline{CS}$ ) <sup>1)</sup>	$t_{OD}$	100	ns
Deselektionszeit <sup>2)</sup>	$t_{OD}$	120	ns

1) für U505  
 2) für U555  
 Meßbedingungen:  
 $U_n = U_{nmin}$  (Betriebsspannungen)  
 $U_{III} = U_{IIImin}$   
 $U_{IL} = U_{ILmax}$   
 $C_L = 120 \text{ pF}$  (Lastkapazität auf den Ausgangslinien)  
 $\theta_a = 70^\circ\text{C}$

Die ROM-Version U505 ist als Festwertspeicher dann einzusetzen, wenn der Programm-inhalt feststeht und ein Gerät in großen Stückzahlen (mehrere tausend) produziert wird. Dann ist der große Aufwand der Herstellung einer zugeschnittenen Maske zur Programmierung gerechtfertigt.

Die Festwertspeicher sind byteorganisiert, so daß immer eine 8-bit-Speicherzelle ausgewählt wird. Die Datenausgabe erfolgt über Tristateausgangsstufen. Deshalb können zur Erreichung einer größeren Speicherkapazität als 1 Kbyte mehrere Schaltkreise in allen Anschlüssen bis auf die Chipauswahl ( $\overline{CS}$ ) parallel geschaltet werden. Es ist dabei abzusichern, daß immer nur ein Eingang  $\overline{CS}$  aktiv wird, damit nicht mehrere parallelgeschaltete Ausgänge gegeneinander arbeiten.

Im Bild 5.1.1 ist das Blockschaltbild der IS U555 dargestellt. Es zeigt die interne Organisation des Festwertspeichers. Die Pinbelegung der Speicherbauelemente U505 und U555 ist im Bild 5.1.2 enthalten.

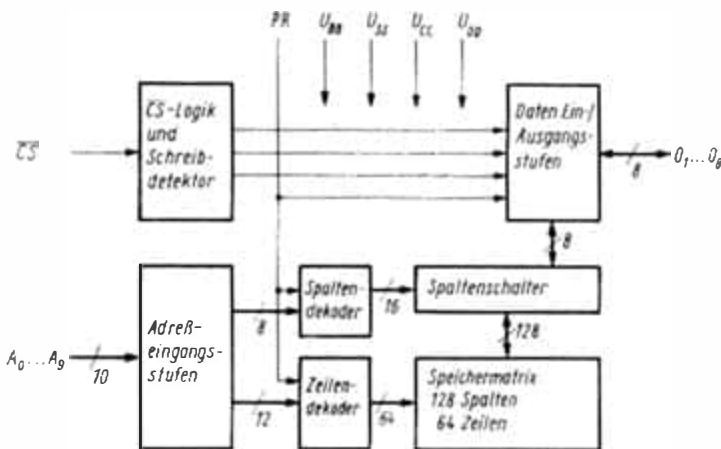
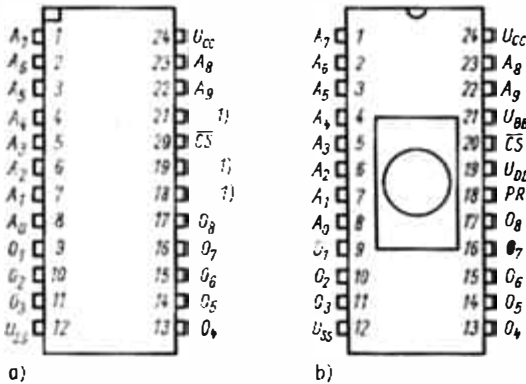


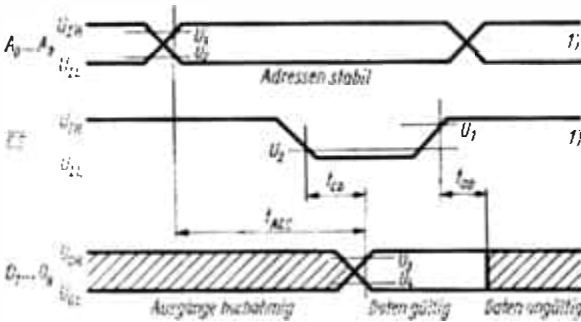
Bild 5.1.1  
 Blockschaltbild der IS U555

Die Tafeln 5.1.1 bis 5.1.3 vermitteln eine Zusammenfassung der elektrischen Kenngrößen der beiden Speicher. Die zugehörige bildliche Darstellung der dynamischen Kennwerte ist im Bild 5.1.3 wiedergegeben.



**Bild 5.1.2**  
Anschlußbelegung  
des 8-Kbit-Festwertspeichers

- a) ROM U505D
  - b) EPROM U555C
- <sup>1)</sup> Die Anschlüsse können mit Potentialen  $-5\text{ V} \leq U \leq 12\text{ V}$  belegt werden



**Bild 5.1.3**  
Zeitverhalten der IS U505  
und U555 im Lesebetrieb

- $U_1 = U_{II\min} - 0,2\text{ V}$
  - $U_2 = U_{II\max} + 0,2\text{ V}$
  - $U_3 = 2,4\text{ V}$
  - $U_4 = 0,8\text{ V}$
- <sup>1)</sup>  $t_{LH}, t_{HL} \leq 1\text{ ns/V}$  (U505)  
 $t_{LH}, t_{HL} \leq 10\text{ ns}$  (U555)

Die Zugriffszeit über den Freigabeeingang  $\overline{CS}$  ( $t_{CO}$ ) ist geringer als die Zugriffszeit über die Adressen ( $t_{ACC}$ ), so daß für die externe Dekoderlogik vor dem  $\overline{CS}$ -Eingang genügend Zeit zum Einschwingen zur Verfügung steht.

Die Dateninformation wird beim EPROM U555 in isolierten Gates (floating gates) gehalten. Die Ladungsträger können in die Gates durch einen Durchbrucheffekt (Avalancheeffekt) gelangen, indem ein Impuls mit relativ hoher Spannung (+26 V; Programmierimpuls an Pin PR) angelegt wird.

Der Schaltkreis U555 wird in den Programmierbetrieb geschaltet, indem am  $\overline{CS}$ -Eingang der Pegel  $U_{III2}$  (+12 V) angelegt wird. Die Spannungsversorgung und Adressierung geschehen hierbei in gleicher Weise wie im Lesebetrieb. Ebenso erfordern die zur Dateneingabe benutzten Anschlüsse  $O_1 \dots O_8$  nur TTL-Pegel. Nachdem gültige Daten- und Adreßinformationen anliegen, erfolgt das Anschalten des kurzzeitigen Programmierimpulses. Jede Speicherzelle muß integral mindestens 50 ms programmiert werden; jedoch darf ein einzelner Programmierimpuls nicht länger als 1 ms anliegen. Deshalb erfolgt das Programmieren zyklisch über den gesamten Speicherbereich. Das zyklische Programmieren ist notwendig, damit bei dem Durchbruchvorgang keine Speicherzellen infolge lokaler Erwärmung zerstört werden. Üblicherweise geschieht das Programmieren der EPROM U555 mit Hilfe spezieller Programmiergeräte.

Die elektrischen Bedingungen für die Programmierung der IS U555 sind zusammenfassend in Tafel 5.1.4 angegeben. Bild 5.1.4 zeigt das hierbei geltende Zeitverhalten.

Das Löschen der EPROM geschieht mit UV-Licht. Die Speicher U555 haben ein Gehäuse mit Quarzglasfenster, damit Licht auf den Chip gelangen kann. Durch intensive Lichteinwirkung (z. B. Sonnenstrahlung) können bereits Ladungsträger aus einzelnen Gates abfließen und so einen Informationsverlust bewirken. Es hat sich daher bewährt, die Quarzglasfenster mit einem Aufkleber zu verschließen. Dieser Aufkleber kann gleichzeitig zur Programmkennzeichnung verwendet werden.

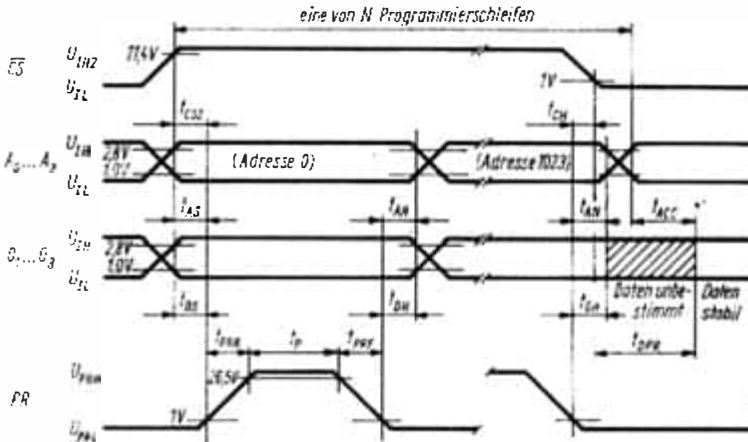


Bild 5.1.4. Zeitverhalten der IS U555 im Programmierbetrieb

Tafel 5.1.4. Kennwerte der IS U555 im Programmierbetrieb

Kenngröße	Kurzzeichen	min.	typ.	max.	Einheit
Betriebsspannung	$U_{CC}$	4,75	5	5,25	V
	$-U_{BB}$	4,75	5	5,25	V
	$U_{DD}$	11,4	12	12,6	V
Betriebstemperatur	$\vartheta_{aP}$	20	25	30	°C
Programmierimpuls					
H-Pegel	$U_{PRH}$	25	26	27	V
L-Pegel	$U_{PRL}$	0		1	V
Programmierimpulsbreite	$t_P$	0,1		1	ms
Programmierimpulsanstiegszeit	$t_{PRR}$	0,5		2	µs
Programmierimpulsabfallzeit	$t_{PRF}$	0,5		2	µs
Programmierzykluszeit	$t_{p \cdot N}$	50			ms
Adreßbereitstellzeit	$t_{AS}$	10			µs
CS-Bereitstellzeit	$t_{CSS}$	10			µs
Datenbereitstellzeit	$t_{DS}$	10			µs
Adreßhaltezeit <sup>1)</sup>	$t_{AH}$	1			µs
CS-Haltezeit <sup>1)</sup>	$t_{CH}$	0,5			µs
Datenhaltezeit	$t_{DH}$	1			µs
Datenverzögerung nach Programmier/ Lese-Umschaltung	$t_{DPR}$			10	µs

<sup>1)</sup> Der Übergang  $U_{IH2}$  auf  $U_{IL}$  an  $\overline{CS}$  am Ende der Programmierzyklen muß nach dem Übergang  $U_{PRH}$  auf  $U_{PRL}$  an PR erfolgen, jedoch vor dem nächsten Adreßwechsel (Adresse 1023 auf Adresse 0).

Im Anfangszustand sind die ausgelieferten EPROM vollständig gelöscht. In diesem Zustand und nach jedem Löschen bewirken alle Bits am Ausgang einen H-Pegel. Die Information wird dadurch eingeschrieben, daß selektiv Nullen programmiert werden. Diese

Nullen können nur dadurch verändert werden, indem der gesamte Chip gelöscht wird. Hierzu wird eine UV-Lichtquelle benötigt, die eine Wellenlänge von etwa 254 nm (Quecksilberdampfampe) bei einem Strahlungsfluß von  $10 \dots 15 \text{ mW/cm}^2$  aufweist. Die Schaltkreise sollten beim Löschen einen Abstand von  $2 \dots 3 \text{ cm}$  von der Lampe haben. Die Löschzeit beträgt dann etwa  $12 \dots 30 \text{ min}$ . Wenn der Abstand zur Lampe verdoppelt wird, so erhöht sich jeweils die erforderliche Löschzeit um das Vierfache.

### 5.1.2. Schreib/Lese-Speicher

Schreib/Lese-Speicher (RAM) dienen i. allg. zur Speicherung von Daten bzw. Variablen. Sie können ebenfalls als Programmspeicher eingesetzt werden, z. B. bei der Programmtestung und Programmearbeitung (in Entwicklungssystemen) oder wenn Programme von externen Datenträgern geladen werden (z. B. von Lochstreifen, Lochkarten, Magnetbändern, Folienspeichern).

Allen diesen Anwendungsfällen der RAM können spezielle Einsatzbedingungen zugeordnet werden, die die Verwendung von statischen, dynamischen oder CMOS-RAM-Bauelementen bedingen. Applikative Angaben hierzu sind in den Abschnitten 6. und 7. enthalten.

Für den Anwender stehen in den genannten Gruppen drei typische Schaltkreise mit folgenden charakteristischen Merkmalen zur Verfügung:

#### U202 bzw. K565RU2A (UdSSR)

- statisches RAM
- Speicherkapazität 1 Kbit, bitorganisiert
- Zugriffszeit  $t_{ACC} \leq 400 \text{ ns}$
- eine Betriebsspannung  $+5\text{V}$
- Schlafzustand durch Absenken der Betriebsspannung möglich (Leistungsaufnahme  $\leq 60 \text{ mW}$ )
- Tristateausgangsstufen
- TTL-Kompatibilität an allen Ein- und Ausgängen
- 16poliges DIL-Gehäuse nach TGL 26713.

#### U256 bzw. K565RU3A (UdSSR)

- dynamisches RAM, Refreshzeit  $\leq 2 \text{ ms}$ , 128 Refreshzyklen
- Speicherkapazität 16 Kbit, bitorganisiert
- Zugriffszeit von  $\overline{\text{RAS}}$   $t_{RAC} \leq 200 \text{ ns}$
- Zugriffszeit von  $\overline{\text{CAS}}$   $t_{CAC} \leq 135 \text{ ns}$
- drei Betriebsspannungen  $-5 \text{ V}$ ,  $+5 \text{ V}$ ,  $+12 \text{ V}$
- Betriebsarten: Lesen, Schreiben, Lesen/Schreiben, Auffrischen (refresh), seitenweises Lesen, seitenweises Schreiben
- Tristateausgangsstufen
- TTL-Kompatibilität an allen Ein- und Ausgängen
- 16poliges DIL-Gehäuse.

#### MH1902 (Tesla, ČSSR)

- CMOS-RAM
- Speicherkapazität 1 Kbit, bitorganisiert
- Zugriffszeit  $t_{ACC} \leq 450 \text{ ns}$

- eine Betriebsspannung + 5 V
- Schlafzustand durch Absenken der Betriebsspannung möglich (Leistungsaufnahme etwa 250  $\mu$ W)
- Tristateausgangsstufen
- Adreßeingangsregister (latch)
- TTL-Kompatibilität an allen Ein- und Ausgängen
- pinkompatibel und eingeschränkt signalkompatibel zu U202
- 16poliges DIL-Gehäuse.

In typischen Mikrorechnerkonfigurationen werden statische RAM als Notizspeicher für Daten eingesetzt, da hierbei in den meisten Fällen nur ein begrenzter Speicherumfang erforderlich ist. Der dynamische Speicher erfordert ein zyklisches Auffrischen der Information, da die Daten in einem Kondensator gespeichert sind. Dazu ist notwendig, daß innerhalb 2 ms mindestens ein Zugriff zu den Speicherzellen erfolgt. Dieser Zugriff kann ein Lesen, Schreiben oder ein Auffrischen (refresh) sein. Der Vorteil dynamischer RAM gegenüber statischen RAM ist die größere Speicherkapazität und Geschwindigkeit. Nachteilig ist der zusätzliche Schaltungsaufwand für den Refresh. CMOS-RAM haben aufgrund der Tatsache, daß im Ruhezustand kein leitender Strompfad vorhanden ist, nur eine äußerst geringe Leistungsaufnahme. Sie ermöglichen zusätzlich im Stand-by-Betrieb ein Absenken der Betriebsspannung zum Datenerhalt, so daß ein leistungsarmer Schlafbetrieb möglich ist.

Zwischen Schreib/Lese-Speicher und Festwertspeicher bestehen i. allg. in der Ansteuerung nur geringe Unterschiede. Schreib/Lese-Speicher haben zusätzlich zu den Festwertspeichern einen Steuereingang, der die Richtung des Datenflusses festlegt. Dieser Eingang wird i. allg. beim Schreiben L-aktiv und heißt Schreibfreigabe ( $\overline{WE}$ , write enable). Für das Einschreiben steht bei den genannten Bauelementen der Dateneingang DI zur Verfügung.

Im Bild 5.1.5 ist das Blockschaltbild des U202 dargestellt. Es zeigt die interne Organisation des Schaltkreises. Bild 5.1.6 enthält die Anschlußbelegung des U202.

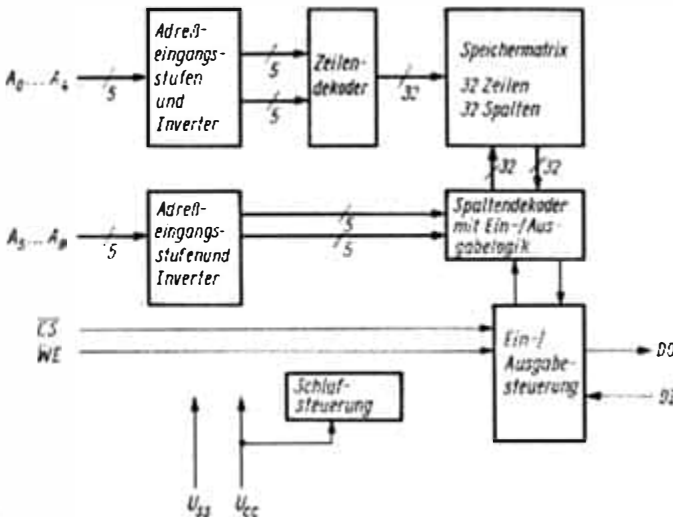


Bild 5.1.5. Blockschaltbild der IS U202

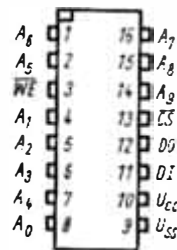


Bild 5.1.6. Anschlußbelegung des 1-Kbit-statischen Schreib/Lese-Speichers U202D

Tafel 5.1.5. Grenzwerte der IS U202

Kenngröße	Kurzzeichen	min.	max.	Einheit
Betriebsspannung	$U_{CC}$	-0,5	7	V
Eingangsspannung	$U_I$	-0,5	7	V
Ausgangsspannung	$U_O$	-0,5	7	V
Verlustleistung des Gehäuses	$P_V$		1	W
Betriebstemperatur	$\vartheta_a$	0	70	°C
Lagertemperatur	$\vartheta_{stg}$	-65	125	°C

Alle Spannungen auf  $U_{SS}$  bezogen.

Tafel 5.1.6. Statische Kennwerte der IS U202

Kenngröße	Kurzzeichen	min.	max.	Einheit
Eingangsreststrom	$I_I$		10	$\mu A$
Ausgangssperrstrom	$I_O$		10	$\mu A$
Stromaufnahme	$I_{CC}$		45	mA
Schlafstromaufnahme	$I_{CCS}$		30	mA
L-Eingangsspannung	$U_{IL}$	-0,5	0,8	V
H-Eingangsspannung	$U_{IH}$	2	$U_{CC}$	V
L-Ausgangsspannung (bei $I_{OL} = 2,1 \text{ mA}$ )	$U_{OL}$		0,4	V
H-Ausgangsspannung (bei $I_{OH} = -100 \mu A$ )	$U_{OH}$	2,4		V
Schlafspannung	$U_{CCS}$	2		V
Eingangskapazität	$C_I$		5	pF
Ausgangskapazität	$C_O$		10	pF

Tafel 5.1.7. Dynamische Kennwerte der IS U202

Kenngröße	Kurzzeichen	min.	max.	Einheit
Dauer des Lesezyklus	$t_{RC}$	400		ns
Zugriffszeit	$t_{ACC}$		400	ns
Ausgangsverzögerung	$t_{CO}$		200	ns
Gültigkeitsdauer der DO-Information nach Adreßänderung	$t_{OH1}$	40		ns
Gültigkeitsdauer der DO-Information nach $\overline{CS}$ -Änderung	$t_{OH2}$	0		ns
Dauer des Schreibzyklus	$t_{WC}$	400		ns
Adreßbereitstellungszeit	$t_{AW}$	20		ns
Schreibimpulsbreite	$t_{WP}$	300		ns
Adreßhaltezeit	$t_{WR}$	0		ns
Datensetzzeit	$t_{DW}$	300		ns
Datenhaltezeit	$t_{DH}$	0		ns
$\overline{CS}$ -L-Zeit im Schreibzyklus	$t_{CW}$	300		ns
Schlafzustand:				
Einschaltverzögerung	$t_{SE}$	0		ns
Ausschaltverzögerung	$t_{SA}$	400		ns

Bei diesem statischen RAM sind prinzipiell vier Betriebszustände zu unterscheiden. Der Ruhezustand ist durch die Bedingung  $\overline{CS} = H$  gekennzeichnet. Hierbei ist die Datenein- bzw. -ausgabe gesperrt, und der Ausgang DO ist im hochohmigen Zustand. Entsprechend der anliegenden Adresse an  $A_0 \dots A_9$  ist eine Speicherzelle intern ausgewählt.

Der Lesezustand wird durch die Bedingungen  $\overline{CS} = L$  und  $\overline{WE} = H$  eingeleitet. Nach Ablauf der Zugriffszeit ( $t_{ACC}$  vom Adreßwechsel,  $t_{CO}$  von der Aktivierung  $\overline{CS}$ ) steht die Information der ausgewählten Speicherzelle am Ausgang DO zur Verfügung. Beim Schreiben ( $\overline{CS} = L, \overline{WE} = L$ ) wird umgekehrt die Information am Eingang DI in die selektierte Zelle geschrieben. Ein Schlafzustand kann während des Ruhezustands durch Absenken der Betriebsspannung  $U_{CC}$  auf den Wert  $U_{CCS}$  erreicht werden. Durch eine interne Indikatorschaltung wird die Spannungsabsenkung erkannt und der Signalweg zwischen Adreßeingangsschaltung und Zeilendekoder unterbrochen. Dadurch sind alle Speicherzellen isoliert und gegen Störungen geschützt. Die Leistungsaufnahme des Schaltkreises ist aufgrund der fehlenden Speicherzellenauswahl und der geringeren Versorgungsspannung stark reduziert.

Die genauen Betriebsbedingungen der IS U202 sind in den Tafeln 5.1.5 bis 5.1.7 dargestellt. Hierbei gibt Tafel 5.1.5 die Grenzwerte des U202 an. Tafel 5.1.6 enthält die statischen Kennwerte, und Tafel 5.1.7 beinhaltet die dynamischen Kennwerte des U202. Bild 5.1.7 gibt das den dynamischen Kenngrößen zugehörige Zeitverhalten wieder.

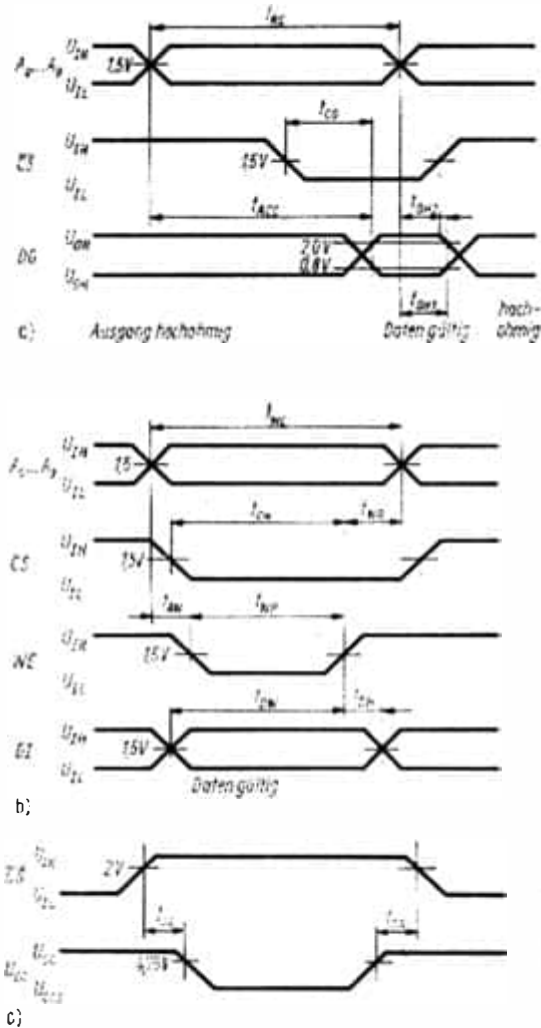


Bild 5.1.7  
Zeitverhalten der IS U202  
a) Lesezyklus  
b) Schreibzyklus  
c) Schlafzustand

Bild 5.1.8 zeigt das Blockschaltbild des dynamischen RAM U256. Die zugehörige Anschlußbelegung ist im Bild 5.1.9 enthalten.

Da die 16-K-dynamischen RAM eine besondere Schaltungstechnik erfordern, sollen hier einige nähere Erläuterungen folgen. Der Speicher ist matrixartig aufgebaut. Zur Adressierung einer Speicherzelle ist eine 14-bit-Adresse erforderlich. Zur Einsparung von Anschlüssen wird die Adresse im Multiplexverfahren eingegeben. Das ist ohne Vergrößerung der Zugriffszeit möglich, da der Speicher intern über 128 Leseverstärker verfügt, so daß eine gesamte Zeile gelesen wird. Die Spezifizierung der Spaltenadresse kann erfolgen, während die Leseverstärker einschwingen. Die Zeilenadresse wird zwischengespeichert. Mit dem Ansprechen einer Zeile werden daher auch gleichzeitig 128 Speicherzellen aufgefrischt, da die gelesene Information wieder eingespeichert und so ein Ladungsverlust ausgeglichen wird. Die 128 Zeilenadressen müssen daher innerhalb von 2 ms mindestens einmal angesprochen werden. Die Zeilenadresse wird über den Takteingang  $\overline{\text{RAS}}$  (row address strobe) intern übernommen. Zur Leistungseinsparung kann der Refresh daher so erfolgen, daß nur  $\overline{\text{RAS}}$  aktiv (L-Pegel) wird. Die Ansteuerung von  $\overline{\text{CAS}}$  (column address strobe) entfällt in diesem Fall, da keine Spezifizierung einer Spalte der Matrix erforderlich ist.

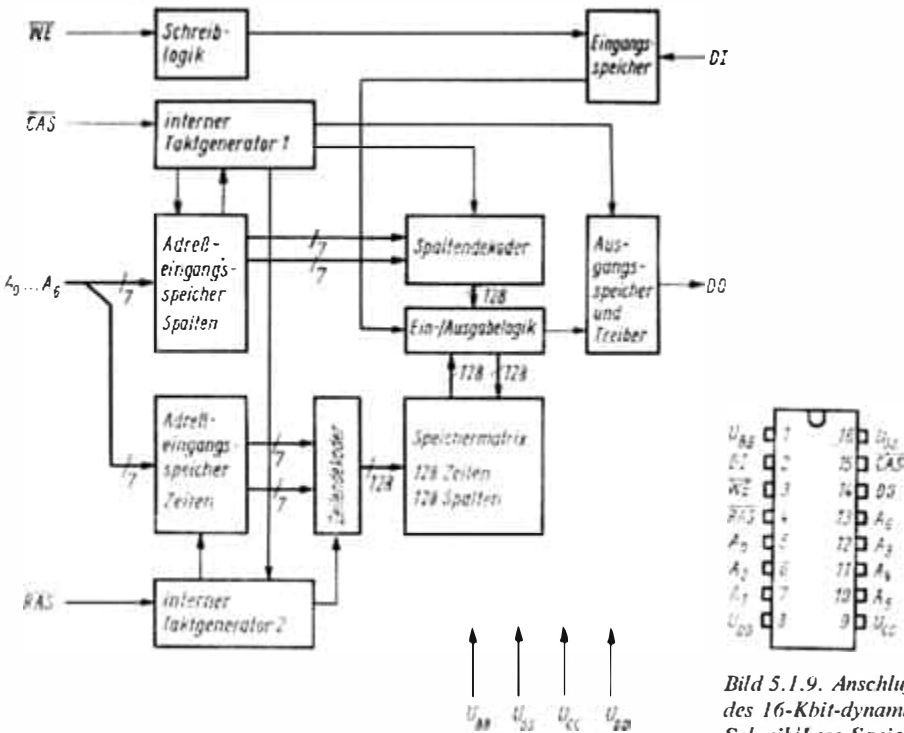


Bild 5.1.8. Blockschaltbild der IS U256

Bild 5.1.9. Anschlußbelegung des 16-Kbit-dynamischen Schreib/Lese-Speichers U256

Die Signale  $\overline{\text{RAS}}$  und  $\overline{\text{CAS}}$  steuern über zwei Taktgeneratoren eine Folge von schaltkreisinternen Ereignissen. Diese werden sequentiell durch unterschiedliche Verzögerungen der internen Takte übertragen. Die zwei Taktfolgen, die durch Aktivwerden von  $\overline{\text{RAS}}$  und  $\overline{\text{CAS}}$  ausgelöst werden, sind logisch miteinander verbunden. Dadurch wird erreicht, daß das Multiplexen der Adresse nicht zeitkritisch wird. Das  $\overline{\text{CAS}}$ -Signal wird innerhalb

des Schaltkreises erst dann wirksam, wenn aufgrund einer Abarbeitung der  $\overline{\text{RAS}}$ -Taktfolge eine interne Freigabe gegeben ist. Das bedeutet, daß extern der  $\overline{\text{CAS}}$ -Takt anliegen darf, sobald die gesicherte Übernahme der  $\overline{\text{RAS}}$ -Adresse erfolgt ist und eine stabile  $\overline{\text{CAS}}$ -Adresse anliegt. Der  $\overline{\text{CAS}}$ -Takt kann auch später anliegen, ohne die Zugriffszeit zu erhöhen. Der Bereich erstreckt sich von dem Zeitpunkt, wo die Zeilenadresse noch zu halten ist, bis zu dem Zeitpunkt, an dem schaltkreisintern die Freigabe des  $\overline{\text{CAS}}$ -Taktes erfolgt. Danach ist die Zugriffszeit nicht mehr determiniert vom  $\overline{\text{RAS}}$ -Takt, sondern vom  $\overline{\text{CAS}}$ -Takt.

Die Daten werden in eine ausgewählte RAM-Zelle eingeschrieben, wenn der  $\overline{\text{WE}}$ -Eingang und der  $\overline{\text{CAS}}$ -Takt aktiv sind und der  $\overline{\text{RAS}}$ -Takt vorher aktiv geworden ist und noch gehalten wird. Die Information des Dateneingangs wird intern zwischengespeichert, so daß sie nicht während des gesamten Schreibzyklus gültig sein muß. Wenn das Schreibsignal ( $\overline{\text{WE}}$ ) vor dem  $\overline{\text{CAS}}$ -Takt aktiv ist (frühes Schreiben), ist gesichert, daß der Datenausgang nicht aktiv wird und im hochohmigen Zustand bleibt. Dadurch ist es möglich, daß Datenausgang und Dateneingang parallelgeschaltet werden.

Der Datenausgang wird nur dann aktiv, wenn  $\overline{\text{RAS}}$  und  $\overline{\text{CAS}}$  aktiv sind und  $\overline{\text{WE}}$  inaktiv zu Beginn von  $\overline{\text{CAS}}$  (Lesen). Das bedeutet, daß man für die Chipauswahl sowohl  $\overline{\text{RAS}}$  als auch  $\overline{\text{CAS}}$  verwenden kann. Im Fall der Selektion über  $\overline{\text{RAS}}$  spricht man von *X-Dekodierung*, im Fall  $\overline{\text{CAS}}$  von *Y-Dekodierung*, mit  $\overline{\text{RAS}}$  und  $\overline{\text{CAS}}$  von *XY-Dekodierung*. Da mit Einleitung eines  $\overline{\text{RAS}}$ -Zyklus bereits Leistung verbraucht wird, ist der Dekodierung über  $\overline{\text{RAS}}$  der Vorzug zu geben.

Durch unterschiedliche Steuerung der Eingänge  $\overline{\text{RAS}}$ ,  $\overline{\text{CAS}}$  und  $\overline{\text{WE}}$  ergeben sich für den 16-K-RAM folgende Betriebsarten:

- Lesen
- Schreiben
- Lesen/Schreiben
- $\overline{\text{RAS}}$ -Refresh
- seitenweises Lesen
- seitenweises Schreiben.

Das gleichzeitige Lesen und Schreiben kann erreicht werden, wenn  $\overline{\text{WE}}$  erst nach einer bestimmten Zeit von  $\overline{\text{RAS}}$  aktiv wird.

Für zeitkritische Verarbeitung großer Datenmengen bieten sich die seitenweisen Betriebsarten an. Diese sind dadurch gekennzeichnet, daß nach Aktivwerden von  $\overline{\text{RAS}}$  dieser Takt für mehrere  $\overline{\text{CAS}}$ -Zyklen aktiv bleibt. Die eingespeicherte Zeilenadresse des RAM bleibt daher konstant, und die Zugriffe finden nur in einer bestimmten Zeile statt. Über die Spaltenadresse läßt sich eine von 128 Speicherzellen ansprechen. Soll diese Anzahl vergrößert werden, können mehrere Speicherschaltkreise parallelgeschaltet und jeweils über  $\overline{\text{CAS}}$  ausgewählt werden. Die Zugriffszeit wird dadurch verringert, daß für jedes Lesen und Schreiben nur ein  $\overline{\text{CAS}}$ -Zyklus stattfinden muß und so die  $\overline{\text{CAS}}$ -Zugriffszeit zum Ansatz gebracht werden kann. Diese Betriebsart bietet sich dann an, wenn nacheinander ähnlich wie bei einem Schieberegister der Zugriff erfolgen muß. Ein typisches Beispiel dafür ist der Bildwiederholpeicher eines grafischen Displays.

Um eine Zerstörung des Bauelements zu vermeiden, muß ähnlich wie beim EPROM U555 die Reihenfolge des Zuschaltens der Betriebsspannungen eingehalten werden. Hier-

bei sollte ebenfalls die Bulkspannung  $U_{\text{BN}}$  als erste eingeschaltet und als letzte ausgeschaltet werden. Die Spannung  $U_{\text{BN}}$  darf nicht positiver als  $U_{\text{SS}}$  werden, wenn  $U_{\text{DD}}$  positiver als  $U_{\text{SS}}$  ist.

Im Bild 5.1.10 ist das Blockschaltbild des statischen CMOS-RAM MH1902 dargestellt. Hieraus ist die unterschiedliche interne Organisation gegenüber dem pinkompatiblen Speicherbauelement U202 zu erkennen. Ein für die Anwendung wesentlicher Unterschied zwischen diesen beiden RAM besteht darin, daß beim MH1902 Adreßzwischenpeicher für die Eingänge  $A_0 \dots A_9$  vorhanden sind. Diese Zwischenpeicher bewirken einerseits, daß bezüglich des Schaltkreises die Leistungsbilanz besser ist, als wenn die Speicher-matrix direkt angesteuert wird (wie beim U202). Andererseits muß vom Anwender berücksichtigt werden, daß die Zugriffszeit  $t_{\text{ACC}}$  erst mit der Aktivierung von  $\overline{\text{CS}}$  und der dadurch bewirkten Adreßübernahme beginnt. Die im Systemkonzept U880 notwendigen Konsequenzen bei dieser Speicheradressierung sind näher im Abschn. 6. erläutert.

Die Anschlußbelegung des MH1902 entspricht der des U202 (s. Bild 5.1.6).

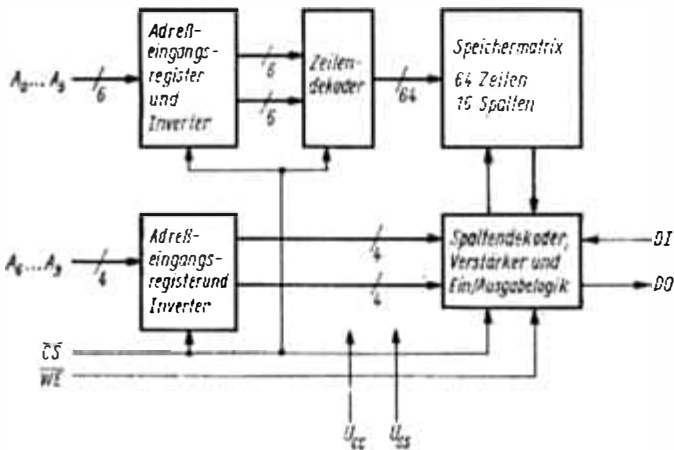


Bild 5.1.10  
Blockschaltbild  
der IS MH1902

## 5.2. Bipolare Standardbauelemente

An dieser Stelle sollen insbesondere die mikrorechnerspezifischen Ergänzungselemente DS8205, DS8212 und DS8216 (VEB Halbleiterwerk Frankfurt/Oder) näher beschrieben werden. Diese IS werden ebenfalls von anderen Herstellern angeboten, u.a. als MH3205, MH3212 und MH3216 von der Firma Tesla (ČSSR). Sie wurden ursprünglich für die Mikrorechnersysteme 8080 und 3000 (bipolares Bit-slice-System mit 2-bit-Prozessorelementen) entwickelt.

Die genannten Bauelemente werden in einer Schottky-TTL-Technologie hergestellt. Ihre charakteristischen Merkmale sind folgende:

### DS8205 oder MH3205

- schneller 1-aus-8-Dekoder
- einfache Erweiterungsmöglichkeit über drei CE-Eingänge
- vollständig TTL-kompatibel
- Verzögerungszeit von den Eingängen zu den Ausgängen max. 18 ns (bei  $\vartheta_a = 25^\circ \text{C}$ )
- 16poliges DIL-Gehäuse.

**DS8212 oder MH3212**

- 8-bit-Datenregister und -treiber mit Zusatzlogik
- verwendbar als 8-bit-Ein-/Ausgabe-Port
- internes Bearbeitungsflipflop für 8080-Interrupterzeugung
- vollständig TTL-kompatibel
- Tristateausgangsstufen
- Ausgangsspannung  $U_{OH} \geq 3,65 \text{ V}$  zur Ansteuerung von nicht TTL-kompatiblen MOS-Bauelementen (z. B. U555)
- Verzögerungszeit von den Dateneingängen DI, von STB oder von  $\overline{CE}_1 \cdot CE_2$  zu den Datenausgängen DO max. 40 ns (bei  $\vartheta_a = 25^\circ\text{C}$ )
- Rücksetzzeit der Datenausgänge DO max. 55 ns (bei  $\vartheta_a = 25^\circ\text{C}$ )
- 24poliges DIL-Gehäuse.

**DS8216 oder MH3216**

- bidirektionaler 4-bit-Bustreiber
- getrennte Anschlüsse für Datenein- und -ausgang
- vollständig TTL-kompatibel
- Ausgangsspannung  $U_{OH} \geq 3,65 \text{ V}$  am Datenausgang DO
- Verzögerungszeit von dem Datenbusanschluß DB zum Datenausgang DO max. 25 ns (bei  $\vartheta_a = 25^\circ\text{C}$ )

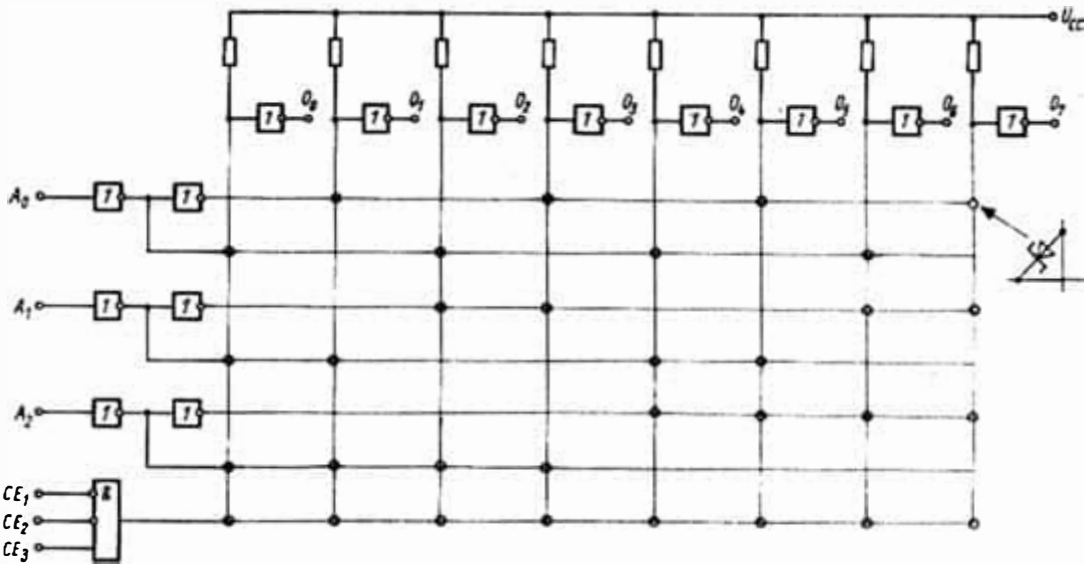


Bild 5.2.1. Innenschaltung der IS DS8205

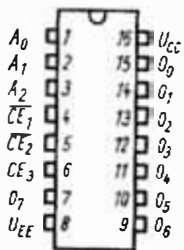


Bild 5.2.2  
Anschlußbelegung  
des Dekoders DS8205

- Verzögerungszeit vom Dateneingang DI zu DB max. 30 ns (bei  $\vartheta_a = 25^\circ\text{C}$ )
- Ausgangsfreigabezeit über die Eingänge  $\overline{\text{CS}}$  und  $\overline{\text{DIEN}}$  max. 65 ns (bei  $\vartheta_a = 25^\circ\text{C}$ )
- Tristateausgangsstufen
- 16poliges DIL-Gehäuse.

Die Innenschaltung des Dekoderschaltkreises DS8205 ist im Bild 5.2.1 wiedergegeben. Die IS verfügt über zwei L-aktive und einen H-aktiven Freigabeeingang. Hierdurch wird in Mikrorechnern eine einfache Konfigurierung und eventuelle Erweiterung des Dekoders unterstützt. Tafel 5.2.1 enthält die wichtigsten statischen Kennwerte der IS DS8205. Im Bild 5.2.2 ist die Anschlußbelegung des Dekoders dargestellt.

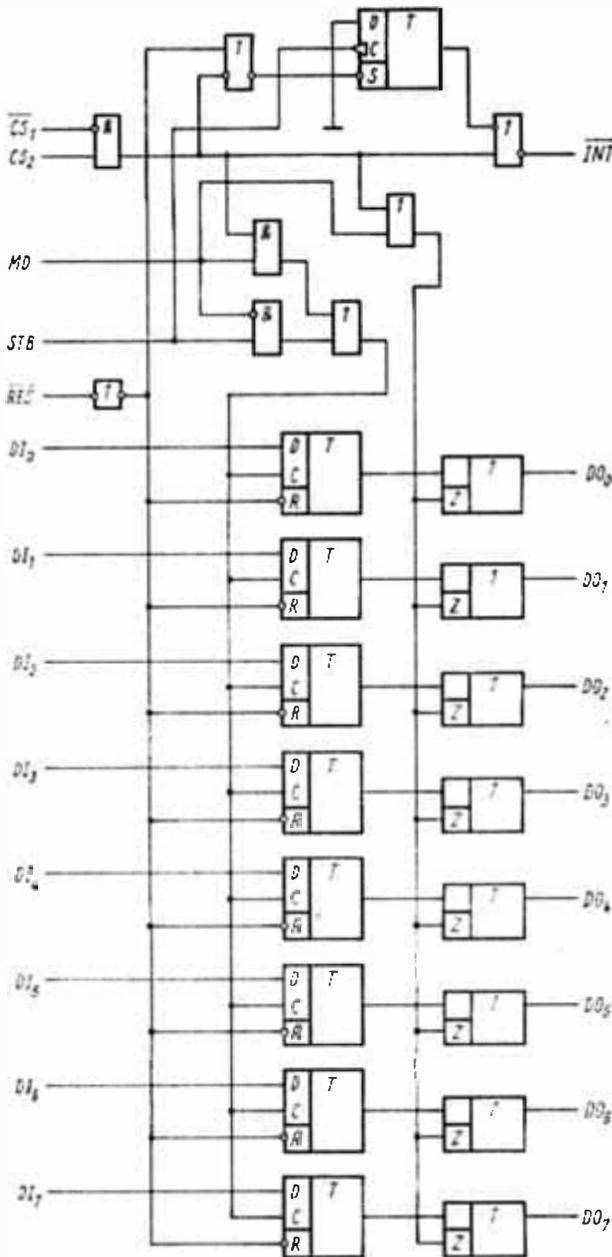


Bild 5.2.3  
Innenschaltung der IS DS8212

Tafel 5.2.1. Statische Kennwerte der IS DS8205

Kenngröße	Kurzzeichen	min.	max.	Einheit
H-Eingangsspannung	$U_{IH}$	2,0		V
L-Eingangsspannung	$U_{IL}$		0,85	V
H-Ausgangsspannung	$U_{OH}$	2,4		V
L-Ausgangsspannung bei $I_{OL} = 10 \text{ mA}$	$U_{OL1}$		0,45	V
bei $I_{OL} = 40 \text{ mA}$	$U_{OL2}$		0,8	V
H-Eingangsstrom	$I_{IH}$		10	$\mu\text{A}$
L-Eingangsstrom	$-I_{IL}$		0,25	mA
Ausgangskurzschlußstrom <sup>1)</sup>	$-I_{OS}$	40	120	mA
Stromaufnahme	$I_{CC}$		70	mA
Eingangskapazität	$C_1$		4	pF

<sup>1)</sup> Nur ein Ausgang belastet.

Die spezifizierten Werte gelten im Betriebstemperaturbereich  $\vartheta_a = 0 \dots 70^\circ\text{C}$ .

Mit dem Schaltkreis MH3212 steht ein universelles Ein-/Ausgabe-Port zur Verfügung. Die Innenschaltung der IS ist im Bild 5.2.3 enthalten. Die acht Flipflops des Datenspeichers sind D-Flipflops, die mit Hilfe des Eingangs  $\overline{RES}$  asynchron rückgesetzt werden können. Die Datenausgänge DO können in den hochohmigen Zustand gesteuert werden; dadurch ist eine Anwendung des DS8212 als Multiplexer oder Buspuffer möglich.

Die Steuerlogik der IS hat die Eingänge  $\overline{CS}_1$ ,  $CS_2$ , MD und STB. Diese Steuereingänge dienen der Chipauswahl sowie zur Steuerung des Datenspeichers, des Zustands der Ausgangspuffer und des Flipflops zur Interruptforderung.

Wenn  $\overline{CS}_1 = L$  und  $CS_2 = H$  gilt, ist die IS selektiert. Damit werden die Ausgangspuffer freigegeben und das Interruptflipflop asynchron gesetzt.

Der Modeneingang MD steuert ebenso wie  $\overline{CS}_1 \cdot CS_2$  die Ausgangspuffer und bestimmt, ob STB oder die Freigabeeingänge zur Datenübernahme benutzt werden. Mit  $MD = H$  (Ausgabemodus) sind die Ausgangspuffer freigegeben, und die Eingangsdaten an DI werden mit  $\overline{CS}_1 \cdot CS_2$  in den Datenspeicher übernommen. Da dieser Speicher zustands-gesteuert ist, läßt sich die IS als Buspuffer (Bustreiber) verwenden. Mit zwei Schaltkreisen DS8212 kann entsprechend ein bidirektionaler Bustreiber aufgebaut werden. Eine weitere Anwendung der IS in dieser Mode ist der Einsatz als Befehlsquelle bei der Quittierung einer Interruptforderung in der Interruptbetriebsart IM0 der CPU U880. Die IS wird in diesem Fall am Eingang ( $DI_0 \dots DI_7$ ) fest mit dem Befehlskode (meist RST-Befehl) beschaltet. Beim RST-Befehl können darüber hinaus drei Op-Kode-Bits vom interruptfordernden Gerät selbst spezifiziert werden.

Bei  $MD = L$  (Eingabemodus) werden die Ausgangspuffer mit  $\overline{CS}_1 \cdot CS_2$  freigegeben, und die Datenübernahme erfolgt mit STB. Somit können mit der IS DS8212 Eingabe- und Ausgabeports realisiert werden, die Handshakefunktionen und Interrupterzeugung ausführen. Im System U880 können diese Ports besonders in Verbindung mit den Interruptbetriebsarten IM0 und IM1 angewendet werden. In der Mode IM2 müßte eine weitere IS DS8212 zur Erzeugung des Interruptvektors eingesetzt werden.

In Tafel 5.2.2 sind zusammenfassend die wichtigsten statischen Kennwerte der IS angegeben. Bild 5.2.4 enthält die Anschlußbelegung des DS8212.

Der Schaltkreis DS8216 stellt einen nichtinvertierenden bidirektionalen Bustreiber dar. Seine Innenschaltung ist im Bild 5.2.5 enthalten. Die IS besteht aus vier Treibereinheiten, die jeweils zwei in Kette geschaltete Treiber aufweisen, sowie aus der Steuerlogik.

Tafel 5.2.2. Statische Kennwerte der IS DS8212

Kenngröße	Kurzzeichen	min.	max.	Einheit
H-Eingangsspannung	$U_{IH}$	2,0		V
L-Eingangsspannung	$U_{IL}$		0,85	V
H-Ausgangsspannung bei $-I_{OH} = 1,0$ mA	$U_{OH}$	3,65		V
L-Ausgangsspannung bei $I_{OL} = 15$ mA	$U_{OL}$		0,45	V
H-Eingangsstrom Eingänge STB, $\overline{CS}_2$ , $\overline{RES}$ , $DI_0 \dots DI_7$ Eingang MD Eingang $\overline{CS}_1$	$+I_{IH1}$ $+I_{IH2}$ $+I_{IH3}$		10 30 40	$\mu$ A
L-Eingangsstrom Eingänge STB, $\overline{CS}_2$ , $\overline{RES}$ , $DI_0 \dots DI_7$ Eingang MD Eingang $\overline{CS}_1$	$-I_{IL1}$ $-I_{IL2}$ $-I_{IL3}$		0,25 0,75 1,0	mA
Ausgangskurzschlußstrom <sup>1)</sup>	$-I_{OS}$	15	75	mA
Stromaufnahme	$I_{CC}$		130	mA
Ausgangsreststrom im hochohmigen Zustand	$ I_{OZ} $		20	$\mu$ A
Eingangskapazität Eingänge $\overline{CS}_1$ , MD andere Eingänge	$C_{11}$ $C_{12}$		12 9	pF
Ausgangskapazität	$C_O$		12	pF

<sup>1)</sup> Nur ein Ausgang belastet!

Alle Werte gelten für  $\theta_a = 0 \dots 70^\circ\text{C}$ .

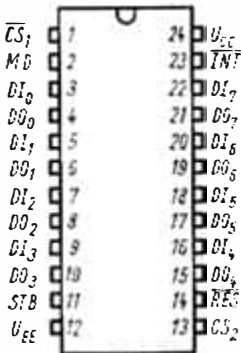


Bild 5.2.4. Anschlußbelegung  
des Datenregisters DS8212

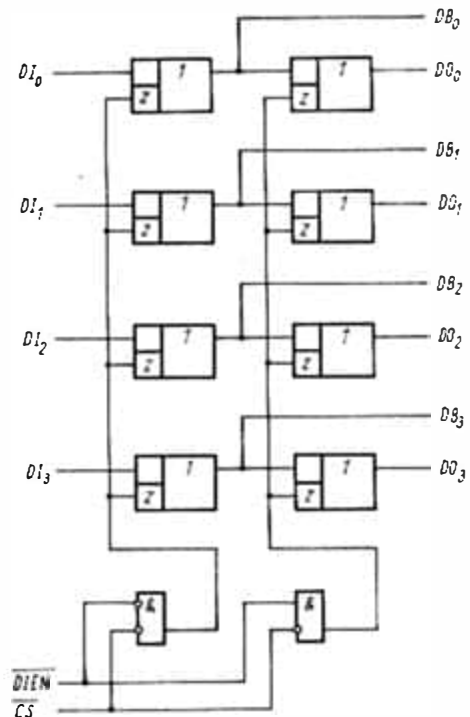


Bild 5.2.5  
Innenschaltung der IS DS8216

Tafel 5.2.3. Statische Kennwerte der IS DS8216

Kenngröße	Kurzzeichen	min.	max.	Einheit
H-Eingangsspannung	$U_{IH}$	2,0		V
L-Eingangsspannung	$U_{IL}$		0,95	V
H-Ausgangsspannung bei $-I_{OH} = 1,0$ mA	$U_{OH1}$	3,65		V
bei $-I_{OH} = 10$ mA	$U_{OH2}$	2,4		V
L-Ausgangsspannung Ausgänge DO bei $I_{OL} = 15$ mA und Ausgänge DB bei $I_{OL} = 25$ mA	$U_{OL1}$		0,45	V
Ausgänge DB bei $I_{OL} = 50$ mA	$U_{OL2}$		0,6	V
H-Eingangsstrom Eingänge $\overline{DIEN}$ , $\overline{CS}$	$I_{IH1}$		80	$\mu A$
Eingänge DI	$I_{IH2}$		40	$\mu A$
L-Eingangsstrom Eingänge $\overline{DIEN}$ , $\overline{CS}$	$-I_{IL1}$		0,5	mA
Eingänge DI, DB	$-I_{IL2}$		0,25	mA
Kurzschlußstrom <sup>1)</sup> Ausgänge DO	$-I_{OS1}$	15	65	mA
Ausgänge DB	$-I_{OS2}$	30	120	mA
Stromaufnahme Ausgangsreststrom im hochohmigen Zustand	$I_{CC}$		120	mA
Ausgänge DO	$ I_{OZ1} $		20	$\mu A$
Ausgänge DB	$ I_{OZ2} $		100	$\mu A$
Eingangskapazität	$C_1$		6	pF
Ausgangskapazität Ausgänge DO	$C_{O1}$		10	pF
Ausgänge DB	$C_{O2}$		18	pF

<sup>1)</sup> Nur ein Ausgang belastet.  
Alle Kennwerte gelten im Betriebstemperaturbereich  $\theta_a = 0 \dots 70^\circ C$ .

Es ist somit möglich, den Datenbus (DB) auf getrennte Datenquellen (DI) und Daten-senken (DO) zu schalten. Hierbei kann immer nur der Treiber aktiv werden, der durch den Datenrichtungseingang  $\overline{DIEN}$  der Steuerlogik ausgewählt ist. Über den Steuereingang  $\overline{CS}$  können beide Treiber in den hochohmigen Zustand gebracht werden ( $\overline{CS} = H$ ).

Durch Parallelschaltung des Dateneingangs DI und des Datenausgangs DO gleichwertiger Elemente wird ein bidirektionaler Bustreiber gebildet.

Die statischen Kennwerte der IS sind in Tafel 5.2.3 angegeben. Bild 5.2.6 zeigt die Anschlußbelegung des bidirektionalen Bustreibers DS8216.

Pinkompatibel zum Bustreiber DS8216 ist die IS MH3226. Im Gegensatz zum DS8216 arbeiten bei dieser IS alle Treiberstufen invertierend. Daraus resultieren etwas günstigere Verzögerungszeiten. Die IS MH3226 wird ebenfalls zur Buspufferung in Mikrorechner-systemen eingesetzt. Aufgrund der invertierenden Arbeitsweise (Quelle und Senke müssen i.allg. grundsätzlich gepuffert werden) ist sie aber meist großen Systemen vorbehalten.



Bild 5.2.6  
Anschlußbelegung  
des Bustreibers DS8216

## 6. Aufbau des Systems U880

### 6.1. Minimalkonfiguration

Ein Mikrorechnersystem besteht i.allg. aus einer CPU-Baugruppe, den Speicherbaugruppen und der Mikrorechnerperipherie. Die CPU-Baugruppe realisiert hierbei die ansteuerseitige Versorgung des Prozessors (Systemtakt, Steuersignale) und die Weitergabe der Systembusinformation an die anderen Baugruppen des Rechners [Bustreibung, Busfreigabe (DMA), Datenrichtungssteuerung]. Der Mikrorechnerspeicher beinhaltet in seinen Baugruppen prinzipiell den Programmspeicher (mit der Rechnersoftware) und einen Datenspeicher. Der Programmspeicher wird hauptsächlich durch Festwertspeicherbauelemente (ROM, PROM, EPROM), der Datenspeicher meist von Schreib/Lese-Speicher-Elementen (RAM) gebildet. Die Mikrorechnerperipherie dient zur Kommunikation des Rechners mit externen Geräten (Steuerungs- und Überwachungsaufgaben) bzw. mit dem Bediener (Monitore, Rechnerüberwachung). Weitere periphere Baugruppen des Mikrorechners können zur Überwachung der Programmbearbeitung („timer interrupt“ durch CTC zur Unterprogrammerzeugung), zur Steuerung von direkten Speicherzugriffen (DMA-Einheiten) oder für andere Zusatzfunktionen eingesetzt werden.

Ein arbeitsfähiger Mikrorechner kann somit konzipiert werden, indem die drei Rechnerbestandteile (CPU, Speicher, Peripherie) in aufeinander abgestimmten Ausbaustufen zusammengeschaltet werden. Die Minimalkonfiguration eines Rechnersystems ergibt sich hierbei aus der Kopplung der möglichst stark abgerüsteten Varianten dieser Baugruppen. Im Bild 6.1.1 ist eine Minimalkonfiguration des Prozessorsystems U880 dargestellt. Diese Konfiguration weist aufgrund der Systemeigenschaften der CPU und der Systemperipherie (in dieser Variante: PIO) trotz des **sehr** geringen Bauelementeaufwands bereits eine relativ hohe Leistungsfähigkeit auf.

Die abgerüstete CPU-Baugruppe besteht in diesem Minimalsystem nur aus dem Prozessor U880, einem Taktgenerator, einer RESET-Logik und der Beschaltung der Interruptanmeldeeingänge der CPU. Der Taktgenerator dient zur Erzeugung des Systemtakts CP. Er ist in dieser Anwendung als Quarzoszillator mit nachfolgenden Teilerstufen ausgeführt. Hierdurch kann die Minimalkonfiguration mit der maximalen Taktfrequenz betrieben werden ( $f_c = 2,5$  MHz, symmetrisches Tastverhältnis). Bei geringeren Anforderungen ( $f_c = 1,5$  MHz) kann aber ein schaltungstechnisch wesentlich einfacherer RC- oder LC-Taktgenerator eingesetzt werden. Der Bauelementeaufwand könnte somit weiter gesenkt werden. Der eigentlichen Takterzeugerschaltung ist eine Taktformerstufe nachgeschaltet, die aus einem TTL-Gatter mit aktivem pull-up besteht. Sie dient zur Erzeugung des notwendigen Spannungshubs am CP-Eingang und zur Erreichung der vorgeschriebenen Taktanstiegs- und -abfallzeiten. Bei Einsatz eines 330- $\Omega$ -Widerstands (passives pull-up) als Taktformer werden ebenfalls die geforderten Spannungspegel am Takteingang erreicht; jedoch können die An- und Abfallgeschwindigkeit des Taktes aufgrund auftretender dynamischer Ströme außerhalb der zulässigen Toleranz liegen. Die einfache RESET-Logik (R1, C1) bewirkt, daß nach Zuschalten der Rechnerversorgungsspannung der Prozessor U880 (und evtl. periphere Elemente, z. B. CTC) rückgesetzt wird. Damit wird er-

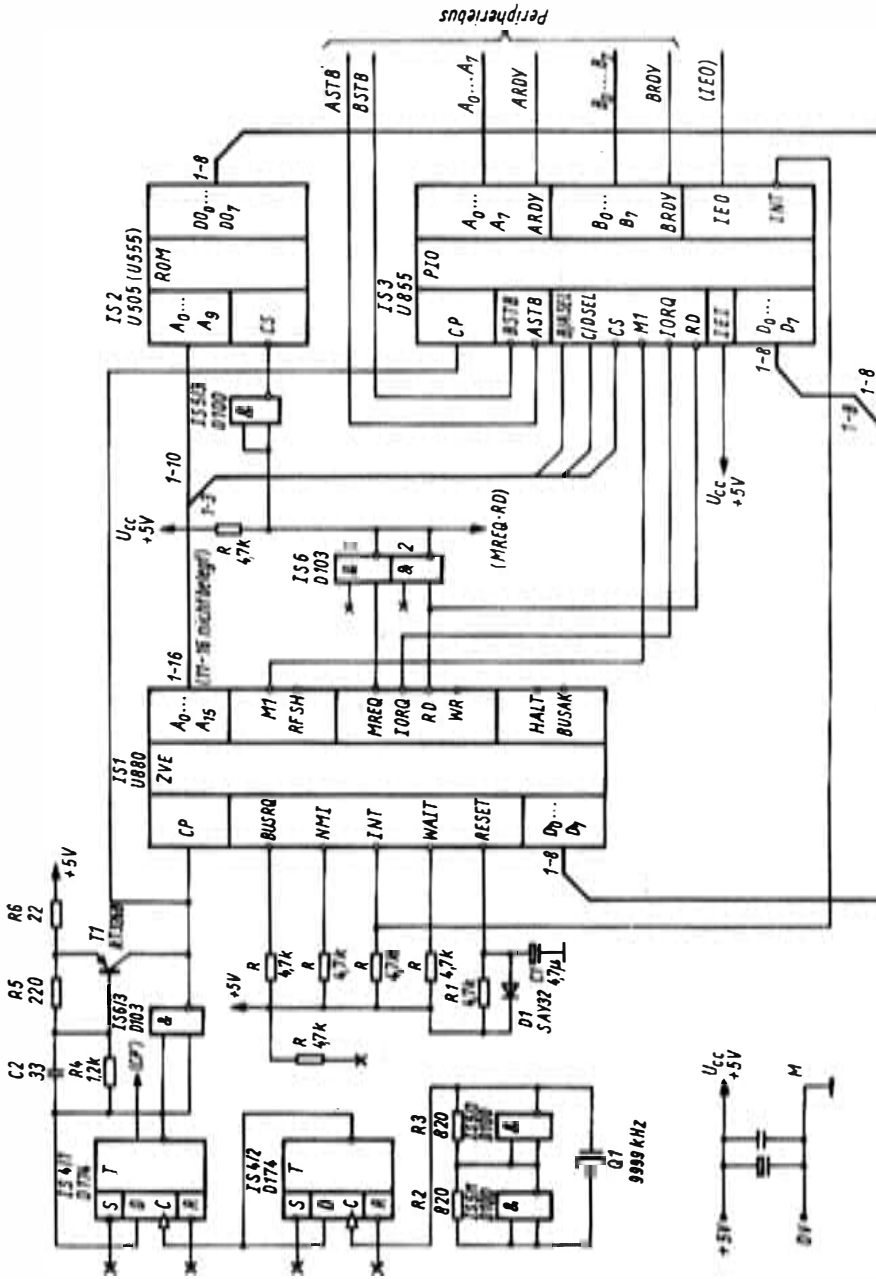


Bild 6.1.1. Minimalconfiguration des Mikroprozessorsystems U880

reicht, daß die Programmbearbeitung der CPU bei der Speicherplatzadresse  $PC = 0000H$  beginnt. Die Beschaltung der CPU-Eingänge  $\overline{INT}$  und  $\overline{NMI}$  ermöglicht bereits in einem kleinen Prozessorsystem die Realisierung von systemweiten Interruptanmeldelinien. Die komfortable Interruptstruktur des Prozessors U880 kann somit bereits im Minimalsystem genutzt werden. Hierbei sind aber programmtechnische Besonderheiten zu beachten, falls kein Stackspeicher (im RAM-Bereich lokalisiert) vorhanden ist. Die Minimalvariante der CPU-Baugruppe erfordert ferner keinerlei Schaltungsaufwand in bezug auf die Verteilung der Systembusinformation. Der Prozessor U880 weist ein zeitlich paralleles Bussystem auf (im Gegensatz zum Multiplexbus des U808), das alle notwendigen Adreß-, Daten- und Steuerinformationen voll dekodiert zur Verfügung stellt. Alle Ausgänge können mit einem Standard-TTL-Eingang belastet werden ( $I_{OL} \leq 1,8 \text{ mA}$ ). Die Eingänge sind bezüglich der Eingangsspannungen ebenfalls TTL-kompatibel (ausgenommen Eingang CP).

Die Speicherbaugruppe enthält im dargestellten Minimalsystem praktisch nur das eigentliche Speicherbauelement. Sonst übliche Bustreibereinheiten, Adreßdekoder, Datenrichtungsumschalteinrichtungen und evtl. notwendige  $\overline{WAIT}$ -Generatoren können weitgehend entfallen. Als Speicherelement bietet sich im einfachsten Fall ein ROM-Speicher U505 an. Dieser Festwertspeicher hat eine Kapazität von 8 Kbit (1 Kbyte) und ermöglicht somit die Unterbringung von kleineren bis mittleren Anwenderprogrammen. Die IS U505 weist des weiteren eine Zugriffszeit von  $t_{ACC} \leq 450 \text{ ns}$  auf und entspricht somit den Systemanforderungen bei der maximalen Systemtaktfrequenz ( $f_C = 2,5 \text{ MHz}$ ). Gegenüber dem sonst bezüglich der technischen Daten gleichwertigen EPROM-Speicherbauelement U555 hat er nur eine Versorgungsspannung  $U_{CC} = 5 \text{ V}$  (U555:  $U_{CC} = 5 \text{ V}$ ,  $U_{DD} = 12 \text{ V}$ ,  $U_{BB} = -5 \text{ V}$ ) und hält somit auch den Netzteilaufwand der Minimalconfiguration gering. Der Einsatz des älteren PROM-Speicherelements U551 (EPROM-Element U552) ist ebenfalls möglich. Diese Bauelemente erfordern aber aufgrund der schlechteren technischen Daten Anpassungseinrichtungen. Die geringere Speicherkapazität (2 Kbit,  $1/4$  Kbyte) macht bereits im Minimalsystem den Einsatz mehrerer Speicherbauelemente und somit den Aufbau eines einfachen Adreßdekoders notwendig. Die geringere Zugriffszeit der IS U551/U552 erfordert den Einsatz eines WAIT-Generators, der den Speicher mit dem Zeitverhalten des Prozessors synchronisiert, oder die maximale Systemtaktfrequenz muß bei dieser Anwendung auf  $f_C < 1,5 \text{ MHz}$  herabgesetzt werden. Die Herstellungstechnologie der IS U551/U552 (p-Kanal MOS) erlaubt des weiteren ausgangsseitig keine direkte Zusammenschaltung der Speicherelemente mit den anderen n-Kanal-MOS-IS (Prozessor U880, periphere Systemelemente U855, U856, U857), da negative Ausgangsspannungen auftreten können. Es wird somit zusätzlich ein Buspuffer bzw. eine entsprechende Klemmschaltung am Datenbus notwendig. Im Bild 6.1.2 ist der zusätzliche Schaltungsaufwand der Speicherbaugruppe bei Verwendung der Bauelemente U551/U552 dargestellt.

In der Speicherbaugruppe der Minimalvariante des Prozessorsystems U880 kann auf den Einsatz eines Schreib/Lese-Speichers (RAM-Speicher) verzichtet werden, da einerseits die CPU U880 aufgrund ihres umfangreichen Doppelregistersatzes bereits für kleine Anwendungsfälle intern ausreichende Speichermöglichkeiten hat und andererseits die Datenregister der peripheren Systemelemente (PIO; evtl. SIO, CTC) weitere Speicherfunktionen ausführen. Die Implementierung eines RAM-Speichers erfordert zudem einen relativ großen Aufwand, wenn nur 1-bit-organisierte RAM-Bauelemente (z. B. U202) verfügbar sind. Da in diesem Fall aber auch kein Stackspeicher im Rechner vorhanden ist, müssen die CALL- und RET-Befehle des Prozessors programmäßig ersetzt oder entsprechend spezifiziert werden (z. B. Setzen des Stackpointers auf eine ROM-Adresse vor

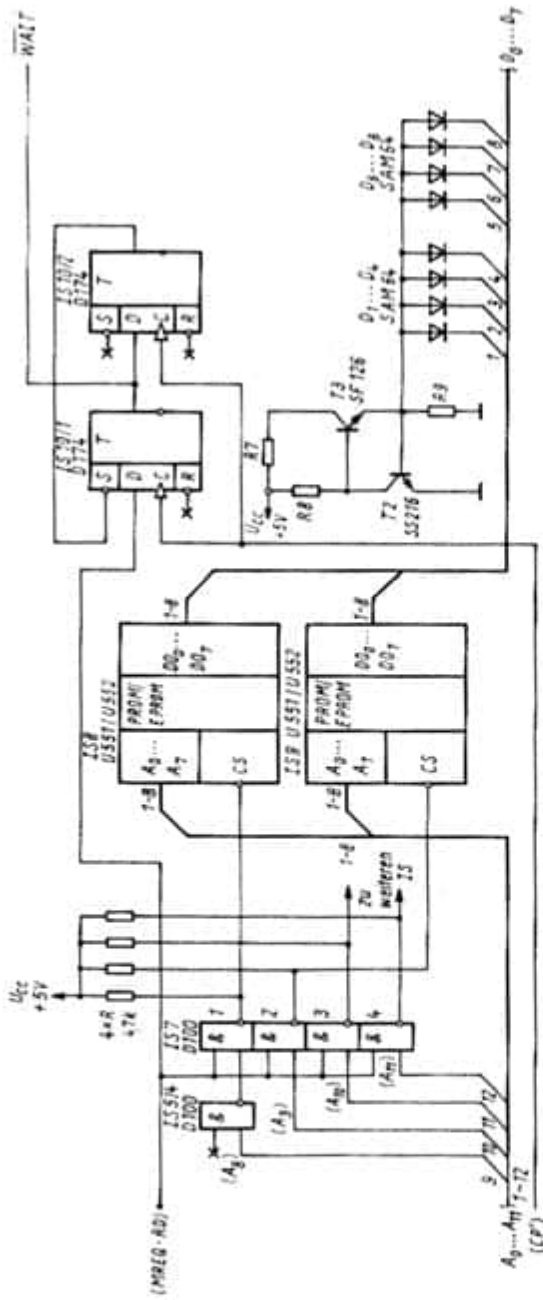


Bild 6.1.2. Einbeziehung von Festwertspeichern der Typen US01, US51 bzw. US52 im Minimalssystem nach Bild 6.1.1.

Ausführung von RET, RETN, RETI). Die Eigenschaften und technischen Daten der in der Speicherbaugruppe einsetzbaren Speicherbauelemente sind im Abschn. 5.1. näher erläutert.

Die Mikrorechnerperipherie erfordert im Minimalsystem rechnerseitig ebenfalls keinerlei Adaptiereinrichtungen. Die Signalkonfiguration der peripheren Systemelemente ist direkt dem Systembus angepaßt. Sie erlaubt den direkten Anschluß von maximal sechs Peripherieelementen ohne Einsatz eines Adreßdekoders und weiterer Hardwarelogik. Die Interruptkaskadierungslinie ermöglicht hierbei die Zusammenschaltung von vier Elementen im Minimalsystem, um zusätzlichen Schaltungsaufwand für Umgehungs- und Zusatzlogik zu vermeiden. Die peripherieeitigen Ein- und Ausgänge sind TTL-kompatibel und können z. T. Darlingtontistorstufen treiben (Port B der PIO, ZC/TO-Ausgänge des CTC).

Die dargestellte Minimalkonfiguration des Systems U880 weist somit bereits wesentliche Leistungsparameter der Systemelemente (CPU, PIO; evtl. CTC, SIO) auf. Sie kann deshalb bereits zur Lösung kleinerer Anwenderprobleme eingesetzt werden. Der besondere Vorteil einer solchen Variante besteht im geringen Bauelementebedarf und in der guten Übersichtlichkeit des Systems. Anwendungsgebiete können deshalb einfache Steuerungsaufgaben (geringer Programmieraufwand bei unkomplizierten Aufgaben), Konsumgüterelektronik (ökonomisch relativ günstige, verfügbare Variante; z. B. für Haushaltwaschmaschine, Grill, Kfz) oder Qualifizierungsaufgaben (Einarbeitung in Mikrorechnerhardware und Programmierung) sein.

Die Minimalvariante stellt eine Grundaustaufstufe dar und bildet somit die Basis für die Erweiterung des Mikrorechnersystems. In ihrer Struktur entspricht sie schon weitgehend einem Einplatinenrechner (SBC, single board computer). Weitere Ausbaustufen sind dann ein modulares System (ein solches System ist durch eine Baugruppenstruktur universell aufrüstbar und erweiterbar) und hierarchische Mikroprozessor- (multiprocessing) und Mikrorechnersysteme. Die Leistungsfähigkeit solcher hierarchischen Rechnerstrukturen erreicht das Niveau, das bisher Mini- und Kleinrechnern vorbehalten war.

## **6.2. Erweiterungsmöglichkeiten**

### **6.2.1. Anforderungen**

Die Konzeption von Mikrorechnersystemen erfordert die Anpassung von Speicher- und Peripheriebaugruppen an den Prozessor bzw. die Prozessorbaugruppe. Hierbei sind vor allem die Anforderungen der Systemelemente (z. B. bezüglich Zeithalten) und bestimmte Einsatzbedingungen des Rechnerkonzepts zu berücksichtigen. Das Systemkonzept eines Mikrorechners bestimmt somit wesentlich dessen Leistungsfähigkeit. Wichtige Parameter dabei sind die Universalität des Rechners in bezug auf seine Anwendungsmöglichkeiten, die Aufrüstbarkeit und Anpaßbarkeit an bestehende Aufgabenstellungen sowie der ökonomische Einsatz von Bauelementen (Preis, Verfügbarkeit).

Die Erweiterung der Prozessorbaugruppe in größeren Mikrorechnern erfolgt vor allem hinsichtlich der Verteilung der Systeminformation (Bussystem). Hierzu werden Bustreiber-schaltkreise eingesetzt, die einerseits die Lastfaktoren der Buslinien vergrößern und somit den Anschluß weiterer Baugruppen (Speicher, Peripherie, Zusatzlogik) an die CPU U880 ermöglichen und andererseits Leitungstreiberfunktionen ausüben, um Störimpulse und Parasitärkapazitäten auf den Linien unwirksam zu machen. Je nach Anwendungsgebiet des Mikrorechners bestehen weitere Erweiterungsmöglichkeiten der CPU-Baugruppe,

besonders in bezug auf Einsatz von Zusatzlogiken, die die Besonderheiten im Zeitverhalten des Prozessors U880 beeinflussen (Interruptbetrieb, DMA, Rücksetzen). Die Realisierung eines Speicherbankbetriebs stellt eine weitere Ergänzung dar und dient zur Erweiterung des Adressierungsbereichs der CPU.

Bei der Anpassung von Speicherbauelementen an die Zentraleinheit (CPU-Baugruppe) bestehen die Aufgaben, den Speicherzugriff dem CPU-Zeitverhalten anzupassen, Adreßdekoder und Adreßbereichsumschalter (Adreßbereichsvorwahl) zu realisieren und datenbusseitig eine Bustreibung mit Richtungssteuerung vorzunehmen. In den Speicherbaugruppen kann je nach Anwendungsfall eine Vielzahl verschiedener Speicherbauelemente eingesetzt werden, die aufgrund ihrer Herstellungstechnologien und inneren Organisation unterschiedliche zeitliche Forderungen (z. B. Zugriffszeit, Einschreibzeit) aufweisen. Diese Zeitbedingungen müssen durch die Speicherbaugruppen dem CPU-Zeitverhalten in den Speicherzyklen (Befehlslesezyklus, M1; Datenlesezyklus, Datenschreibzyklus) angepaßt werden. Zur Synchronisation des Zeitverhaltens dienen, falls erforderlich, in diesen CPU-Zyklen WAIT-Zustände, die zweckmäßigerweise mit Hilfe einer Zusatzlogik von den Speicherbaugruppen über eine systemweite Anmeldeleitung ( $\overline{\text{WAIT}}$ -Eingang der CPU, „wired-or“ verknüpft) angefordert werden können. Bei den Speicherbauelementen muß in bezug auf Zeitverhalten und Adressierungsart zwischen zwei Gruppen unterschieden werden.

Die erste Gruppe (z. B. EPROM U555, RAM U202) verwendet die auf dem Adreßbus des Speicherelements ankommenden Informationen direkt zur Ansteuerung der internen Speichermatrix. Diese Auswahl erfolgt unabhängig vom Anliegen des Freigabesignals  $\overline{\text{CS}}$  des Speicherelements. In Abhängigkeit von den Steuersignalen des Elements (z. B. Freigabesignal  $\overline{\text{CS}}$ , Schreib/Lese-Umschaltung  $R/\overline{W}$ , Ausgangsfreigabe OE) erfolgt dann die Ausführung des eigentlichen Datenverkehrs. Der Speicherzugriff beginnt somit, sobald gültige Informationen auf dem CPU-Adreßbus ausgegeben werden. Die Matrix der Speicherbauelemente wird durch die Adreßlinien aktiviert. Prinzipiell ergibt sich nun die maximale Zugriffszeit der im System U880 einsetzbaren Speicherbauelemente, die eine derartige interne Adreßsteuerung aufweisen, aus folgender Gleichung:

$$t_{\text{ACC}} \leq (n \cdot t_{\text{C}}) - t_{\text{D(AD)}} - t_{\text{DTTL}} + m \cdot t_{\text{C}}; \quad (6.2.1)$$

$t_{\text{ACC}}$  Zugriffszeit der Speicherelemente

$t_{\text{C}}$  Periode des Systemtakts ( $t_{\text{Cmin}} = 400 \text{ ns}$ )

$n \cdot t_{\text{C}}$  CPU-Zykluszeit bis Datenübernahme ( $n = 2$  für Befehlslesezyklen;  $n = 2,5$  für Datenzyklen)

$t_{\text{D(AD)}}$  Adreßverzögerungszeit der CPU ( $t_{\text{D(AD)}} = 145 \text{ ns}$ )

$t_{\text{DTTL}}$  TTL-Verzögerungszeit (Adreßtreiber, Datenbuspuffer)

$m$  Anzahl der eingefügten WAIT-Zustände (falls erforderlich).

Adressenaktivierte Speicherelemente in n-Kanal-MOS-Technologie mit Zugriffszeiten von  $t_{\text{ACC}} \leq 450 \text{ ns}$  können somit im U880-System ohne Einfügung von WAIT-Zuständen direkt angesteuert werden. Bei der maximalen Systemtaktfrequenz ergeben sich hierbei mögliche TTL-Verzögerungszeiten bis  $t_{\text{DTTL}} \leq 205 \text{ ns}$  im Übertragungsweg, die den Einsatz von Datenbustreibern (üblicherweise prozessor- und baugruppenseitig) und Adreßpuffern (üblicherweise prozessorseitig) gestatten.

Die zweite Gruppe der Speicherbauelemente (z. B. dynamisches RAM K565RU3A, CMOS-RAM MH1902) hat interne Adreßregister, die abhängig von der Chipauswahl (Signal  $\overline{\text{CS}}$ ) die Adreßinformationen für die Speichermatrix zwischenspeichern. Die hierdurch erzielten Vorteile sind eine günstigere Leistungsbilanz der Speicherelemente (Zugriff nur bei Chipauswahl) und die Möglichkeit der Adreßmultiplexung zur Verminderung der Anschlüsse des Gehäuses (bei großen dynamischen Speichern). Der Zugriff bei dieser

chipaktivierten Gruppe von Speicherbauelementen beginnt also erst nach Anliegen des Kombinationssignals  $\overline{CS}$ . Dieses Signal wird üblicherweise aus dem CPU-Steuersignal  $\overline{MREQ}$ , durch logische Verknüpfung mit den höherwertigen, in den Adreßbereichsdekodern der Baugruppe verwendeten CPU-Adreßinformationen gebildet. Das Prozessorzeitverhalten erfordert somit kürzere wirksame Zugriffszeiten bei diesen Speicherbauelementen. Die maximale Zugriffszeit eines chipaktivierten Elements ergibt sich allgemein nach

$$t_{ACC} \leq (n \cdot t_C) - t_{DL(MR)} - t_{DTTL} + (m \cdot t_C); \quad (6.2.2)$$

$t_{ACC}$  Zugriffszeit der Speicher

$t_C$  Periode des Systemtakts

$n \cdot t_C$  CPU-Zykluszeit zwischen Aktivierung von  $\overline{MREQ}$  und Datenübernahme ( $n = 1,5$  für Befehlslesezyklus;  $n = 2$  für Datenzyklen)

$t_{DL(MR)}$  Verzögerungszeit von  $\overline{MREQ}$  der CPU auf L-Pegel

$t_{DTTL}$  TTL-Verzögerungszeit (Adreßtreiber, Datenbuspuffer, Adreßdekoder)

$m$  Anzahl möglicher WAIT-Zustände (falls erforderlich).

Die TTL-Verzögerungszeiten bestehen hierbei aus den Durchlaufzeiten der Daten- und Adreßpuffer und zusätzlich aus den Verzögerungszeiten, die im Adreßdekoder und bei der Verknüpfung des  $\overline{MREQ}$ -Signals auftreten. Unter Worst-case-Bedingungen können also im System U880 chipaktivierte Speicherelemente mit Zugriffszeiten  $t_{ACC} \leq 300$  ns ohne Einfügung von zusätzlichen WAIT-Zuständen eingesetzt werden. Bei größeren Zugriffszeiten (z. B.  $t_{ACC} = 450$  ns) muß jeweils ein WAIT-Zustand in die CPU-Speicherzyklen (evtl. nur M1-Zyklus) eingefügt werden. Das Zeitverhalten der Speicherbauelemente bei Einschreibvorgängen (RAM-Speicherelemente) muß ebenfalls auf das CPU-Zeitverhalten abgestimmt werden. Es gelten prinzipiell ähnliche Zusammenhänge wie beim Lesevorgang (Zugriffszeit).

Die Erweiterung von Speicherbaugruppen in bezug auf den Einsatz von Datenbusstreibern (bidirektional bei RAM-Elementen) ist in größeren Mikrorechnern notwendig, um einerseits den Systembus hinsichtlich des Lastfaktors treiben zu können und um andererseits hohe Kapazitäten der Ein-/Ausgänge der Speicherelemente (z. B. parallelgeschaltete Datenlinien von 8-bit-orientierten Elementen) vom Systembus fernzuhalten. Der Einsatz von Datenbusstreibern ist bei bestimmten Speicherbauelementen (z. B. U202) auch aufgrund des Zeitverhaltens notwendig. Bidirektionale Bustreiber (bei Schreib/Lese-Speichern) benötigen zusätzlich eine Richtungslogik, die, abhängig von den CPU-Steuer-signalen  $\overline{RD}$ ,  $\overline{WR}$ , die Richtung des Datenverkehrs bestimmt und Buskonflikte vermeidet. Eine Adreßdekodierung ist in Mikrorechnern erforderlich, um den gesamten Adressierungsraum der CPU ausnutzen zu können. Hierzu müssen sämtliche Adreßlinien (auch die für einen evtl. Speicherbankbetrieb neugebildeten höchstwertigen Linien) in der Speicherbaugruppe verarbeitet werden. Die niederwertigen Adressen (z. B.  $A_0 \dots A_9$  bei U202, U555) werden direkt zur Ansteuerung der Speichermatrix verwendet und so mit den Adreßlinien der Bauelemente verbunden. Adressentreiber sind nicht unbedingt erforderlich (nur in sehr großen Systemen, um kapazitive Lasten und Störpegel zu vermeiden). Weitere Adreßlinien können in Verbindung mit weiteren Steuer- und Freigabesignalen zur Chipauswahl der eingesetzten Speicherelemente (z. B. 8-bit-organisierten Elementen; U555) dienen. Es bietet sich hierfür der Einsatz von Dekoderschaltkreisen (z. B. MH3205; s. Abschn. 5.2.) an. Die übrigen höchstwertigen Adreßleitungen werden zur Speicherblockauswahl benutzt. Sie bilden, abhängig von einer Kodierung, ein Freigabesignal, das zur Ansteuerung der gesamten Baugruppe verwendet werden kann. Die Auslastung des gesamten CPU-Adressierungsraums ist somit gewährleistet, da mehrere

(gleichartige) Speicherbaugruppen aufgrund einer unterschiedlichen Kodierung in verschiedenen Adressierungsbereichen arbeiten können. Die Speicherblockauswahl kann z. B. mit Äquivalenzgattern (TL 7486) oder mit anderen TTL-Standardelementen realisiert werden; zur Adreßbereichskodierung kommen Kodierschalter (z. B. in DIL-Gehäusen), Kodiersteckverbinder (mit Kurzschlußsteckern), bipolare PROM-Elemente oder Drahtbrücken in Betracht.

Bei der Anpassung von Peripheriebaugruppen an die Prozessoreinheit bestehen neben den das Zeitverhalten und die Adressierung betreffenden Anforderungen weitere, die insbesondere Systemeigenschaften wie Interruptverhalten, DMA-Verhalten und die Realisierung von Rechnerschnittstellen betreffen. Das CPU-Zeitverhalten ist auf den Datenaustausch mit den systemzugehörigen Peripherieelementen (PIO, SIO, CTC) zugeschnitten und erfordert keine Anpassungen. Bei anderen peripheren Elementen (z. B. AD-Wandler-Elemente, UART MHB1012) können, falls erforderlich, WAIT-Zustände zur Synchronisierung des Zeitverhaltens eingefügt werden. Das System U880 weist prinzipiell einen Peripherieadressierungsraum von 8 bit auf. Durch die Adreßdekoder der Peripheriebaugruppen können somit 256 verschiedene Kanäle adressiert werden. Da die peripheren Systemelemente aber die niedrigstwertigen Adressen ( $A_0$ ,  $A_1$ ) direkt für Steuerzwecke benötigen (Kanalsteuerung, Kanalauswahl), besteht im System ein Adressierungsraum für max. 64 Systemelemente. Dieser Adressierungsraum ist für spezielle Anwendungen (z. B. bei Verwendung von symbolischen, interpreterbezogenen Rechnersprachen bzw. -sprachvereinbarungen in Ablaufsteuerungen) durch die Einbeziehung der höherwertigen Adreßlinien ( $A_8 \dots A_{15}$ ) erweiterbar.

Der Einsatz von Datenbustreibern (bidirektional bei Eingabegeräten, Anwendung der Interruptbetriebsart IM2) ist wie in den Speicherbaugruppen erforderlich, um in bezug auf Lastfaktoren und kapazitive Belastung einen Datenverkehr mit dem Systembus zu realisieren. Bei der Gestaltung der Datenrichtungsumschaltung muß berücksichtigt werden, daß die Systemelemente PIO, SIO, CTC die Informationen auf dem Systembus während der MI-Maschinenzyklen (Befehlslesezyklen) überwachen können, damit die interne RETI-Dekodierung durchgeführt werden kann.

Weitere Systemanforderungen an die peripheren Baugruppen bestehen bezüglich der Interrupt- und DMA-Eigenschaften. Hierzu gehören die Realisierung der Interruptkaskadierungskette mit einer den speziellen Anforderungen genügenden Umgehungs- und Zusatzlogik (s. Abschn. 4.2.) und die Gestaltung der Systembusübernahme bei DMA-Betrieb einer entsprechenden peripheren Baugruppe zur Vermeidung von Buskonflikten und floatenden Buslinien (s. Abschn. 4.3.). Die Anwendung der leistungsfähigen Interruptbetriebsart IM2 des Prozessors kann bei Einsatz von nichtsystemzugehörigen Peripherieelementen (z. B. UART MHB1012) erreicht bzw. unterstützt werden, indem deren Interruptlinien nicht auf die systemweite  $\overline{\text{INT}}$ -Anmeldelinie geschaltet, sondern durch ein Systemelement (z. B. CTC als Interruptgenerator) überwacht werden. Dieses Element gibt dann eine Interruptforderung an die CPU weiter und liefert, abhängig von seiner aktuellen Priorität, den Interruptvektor.

Weitere Gesichtspunkte bei der Systemkonzeption eines Mikrorechners bzw. -systems sind die Realisierung der Busverdrahtung (z. B. gedruckte Rückverdrahtung), die Schaffung von Busverstärkern zur Ankopplung weiterer Baugruppen, die Einbeziehung von Test- und Prüfmöglichkeiten des Rechners (interner/externer Systemtakt, Emulatoranschlüsse, Bedieneinheiten, Einzelschrittbetrieb, Systembusüberwachung u. dgl.), die Anpassung des verwendeten Leiterkartenformats an die Einsatzforderungen des Systems sowie ökonomische Beziehungen wie Verfügbarkeit und Lieferzeiten von Bauelementen.

### 6.2.2. Prozessorbaugruppe

Die CPU-Baugruppe hat im Mikrorechner insbesondere die Aufgabe, den Prozessor ansteuerseitig zu versorgen und Systeminformationen im Rechner zu verteilen. Sie organisiert somit den Datenverkehr, die Steuerung sowie weitere Zusatzfunktionen des Mikrorechners. Die Systembusverteilung erfolgt in größeren Rechnern über Bustreiberschaltkreise (z. B. DS8212, LS-TTL-Technologie; bezüglich technischer Daten s. Abschn. 5.2.), die den Anschluß vieler weiterer Baugruppen an die Prozessoreinheit gestatten. Die Datenbuslinien werden hierbei durch bidirektionale Bustreiber (z. B. MH3216) gepuffert; die Datenrichtungsschaltung erfolgt abhängig von den CPU-Steuersignalen  $\overline{RD}$ ,  $\overline{WR}$ .

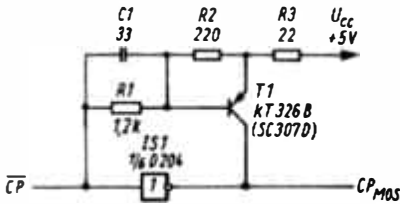


Bild 6.2.1  
MOS-Takttreiberstufe  
mit aktivem pull-up

Die Prozessorbaugruppe beinhaltet einen Taktgenerator, der den Systemtakt für die CPU und die anderen Baugruppen (z.B. Peripherie) bereitstellt. Die gewählte Taktfrequenz ( $f_{C_{max}} = 2,5$  MHz) kann den Einsatzanforderungen des Mikrorechners angepaßt und z. B. zur Erreichung bestimmter Datenübertragungsgeschwindigkeiten (Baudraten) in den peripheren Baugruppen modifiziert werden. Eine hierfür übliche Systemtaktfrequenz beträgt  $f_C = 2,4576$  MHz. Die taktmäßige Ansteuerung der Systemelemente (CPU, PIO, SIO, CTC) erfordert eine MOS-Takttreiberanordnung, die einerseits den notwendigen H-Pegel ( $U_{CC} - 0,2$  V) und andererseits die im Standard spezifizierten Taktanstiegs- und -abfallzeiten ( $t_r, t_f \leq 30$  ns bzw.  $t_r, t_f \leq 20$  ns für  $f_C = 2,5$  MHz) einhält. Eine derartige Schaltungsanordnung ist im Bild 6.2.1 dargestellt. Die MOS-Takttreiberschaltung arbeitet mit einem aktiven pull-up (gebildet durch T1) und erfüllt die gestellten Forderungen im gesamten zulässigen Frequenzbereich des Systemtakts. Ihr Nachteil besteht vor allem darin, daß sie eingangsseitig einen TTL-Standardlastfaktor von  $N = 4$  aufweist. Seitens der Prozessorbaugruppe müßte somit gewährleistet werden, daß der zum Systembus wirkende TTL-Taktverstärker (z. B. realisiert durch Leistungsgatter D140 oder durch parallelgeschaltete Gatter D204) entsprechend viele derartige MOS-Takttreiberanordnungen (in den peripheren Baugruppen zur Versorgung der peripheren Systemelemente) ansteuern kann. Zur Vermeidung von TTL-Verzögerungszeiten bietet es sich an, auf dem Systembus das invertierte Systemtaktsignal  $\overline{CP}$  zu verteilen, da auch der MOS-Takttreiber wieder invertierend wirkt.

Weitere alternative MOS-Takttreiberanordnungen sind im Bild 6.2.2 dargestellt. Die Schaltung nach Bild 6.2.2a ist mit einem passiven Pull-up-Element (Widerstand) ausgestattet und erlaubt nur die Treibung einer begrenzten Anzahl von Systemelementen. Ihr Vorteil besteht im geringen Schaltungsaufwand. Der Takttreiber nach Bild 6.2.2b ist mit diskreten Elementen realisiert. Er ist prinzipiell mit der Anordnung aus Bild 6.2.1 vergleichbar.

Eine wichtige Funktion der Prozessorbaugruppe besteht in der Bereitstellung bzw. Erzeugung von Systemsteuersignalen. Bei der Generierung des  $\overline{RESET}$ -Signals ist hierbei folgende Besonderheit der CPU U880 zu berücksichtigen. Erfolgt die Aktivierung des  $\overline{RESET}$ -Eingangs der CPU während des T3-Taktzustands in einem M1-Maschinenzyklus,

so nimmt das  $\overline{MREQ}$ -Steuersignal des Prozessors für einen der folgenden Taktzustände einen undefinierten Logikpegel ein. Hierdurch wird ein unvollständiger Speicherlesezyklus in den Speicherbaugruppen ausgelöst, der in dynamischen RAM-Speicherelementen (z. B. U256) das Löschen einer Zeile der RAM-internen Matrix hervorrufen kann. Beim **Einsatz** dynamischer Speicherbauelemente muß deshalb eine Synchronisation des externen  $\overline{RESET}$ -Signals mit dem  $\overline{M1}$ -Steuersignal vorgenommen werden. Im Bild 6.2.3 ist eine Schaltung gezeigt, die diese Funktion ausführt und außerdem beim Zuschalten der Rechnerversorgungsspannung ( $U_{CC}$ ) ein Power-on-Rücksetzen ausführt.

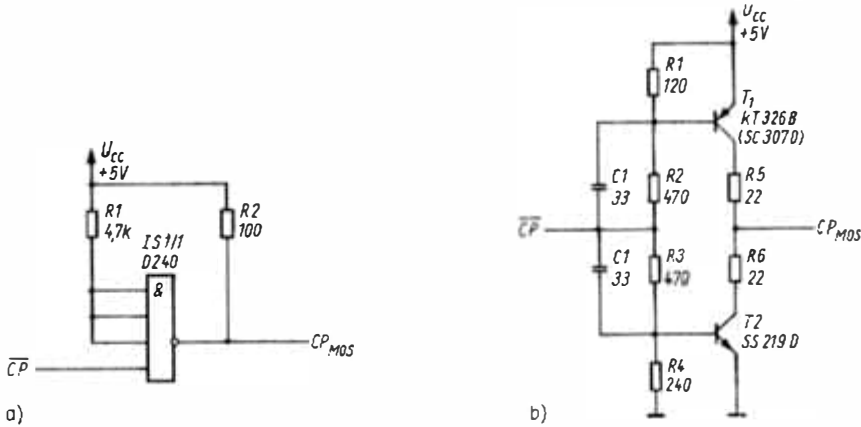


Bild 6.2.2. MOS-Takttreiberstufen

- a) Einsatz eines Hochgeschwindigkeits-Leistungsgatters mit passivem pull-up
- b) Gegentakttreiberstufe mit pnp- und npn-Transistoren

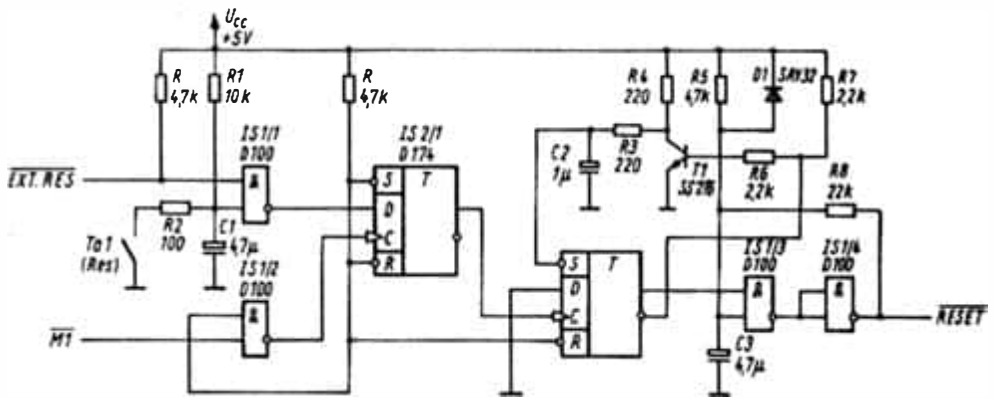


Bild 6.2.3. Erweiterte RESET-Logik zum Einsatz in Mikrorechnersystemen mit dynamischen Speicherelementen

Erweiterungsmöglichkeiten der Prozessorbaugruppe bestehen auch hinsichtlich der Verbesserung des Leistungsniveaus der CPU U880. Für bestimmte Anwendungsfälle (z. B. Vorhandensein großer Datenmengen bei relativ geringem Datendurchsatz) ist es erforderlich, den Adressierungsraum der CPU im Rechner zu erweitern. Hierzu können höchstwertige Adreßlinien ( $A_{16}$  und weitere) beispielsweise durch ein PIO-Element (U855) erzeugt werden, das durch Ausgabeoperationen geladen wird. Die PIO muß durch das  $\overline{RESET}$ -Signal rücksetzbar sein, damit nach der Power-on-Initialisierung die CPU auf



### 6.2.3. Speicherbaugruppe

#### 6.2.3.1. Festwertspeicher

Der Prozessor U880 gestattet in bezug auf Zeitverhalten die direkte Einbeziehung der in n-MOS-Technologie gefertigten Festwertspeicherelemente U505 und U555 in die Speicherbaugruppen des Systems. Die CPU-Speicherzyklen (insbesondere der M1-Befehlslesezyklus) weisen hierbei genügende Zeitreserven auf, um TTL-Verzögerungszeiten der Adreß- und Datenbustreibererlemente sowie der Adreßdekoder ( $t_D \leq 155$  ns, bezogen auf die Adreßlinien;  $t_D \leq 330$  ns, bezogen auf die Bildung von  $\overline{CS}$  der Speicherelemente) auszugleichen. Im Bild 6.2.5 ist die Schaltung einer Speicherbaugruppe dargestellt, die acht ROM/EPROM-Elemente U505/U555 enthält. Die Chipauswahl erfolgt über einen 1-aus-8-Dekoder MH3205; der Adressierungsraum beträgt 8 Kbyte. Aufgrund der vollen Dekodierung aller Adreßlinien des Prozessors und der mit Hilfe der Kodierstecker freien Adreßbereichswahl können bis zu acht derartiger Baugruppen im Mikrorechnersystem U880 angeordnet werden.

Das ROM-Speicherelement U505 ist zum EPROM U555 vollständig pinkompatibel, d. h., es kann eine unmittelbare Substitution erfolgen. Während jedoch der U505 nur eine Versorgungsspannung benötigt ( $U_{CC} = 5$  V,  $U_{SS} = 0$  V), müssen beim EPROM U555 drei Spannungen ( $U_{CC} = 5$  V,  $U_{DD} = 12$  V,  $U_{BB} = -5$  V,  $U_{SS} = 0$  V) bereitgestellt werden. Beim Einsatz dieser EPROM-Elemente ist es also erforderlich, entweder diese Betriebsspannungen im Mikrorechnernetzteil zu erzeugen und über das Verdrahtungssystem zu den Speichern zu leiten oder die zusätzlichen Spannungen  $U_{DD}$  und  $U_{BB}$  mittels Gleichspannungswandler aus der Rechnerversorgungsspannung  $U_{CC}$  direkt in der Speicherbaugruppe zu erzeugen. Die dezentrale Betriebsspannungserzeugung verringert den Aufwand im Mikrorechnernetzteil und bietet optimale Bedingungen für den Einsatz von Schaltnetzteilen. Der Schaltungsaufwand für die Gleichspannungswandler, die zur Erzeugung von  $U_{BB}$  und  $U_{DD}$  notwendig wären, ist aufgrund des Strombedarfs ( $I_{BB} \leq 45$  mA,  $I_{DD} \leq 65$  mA je Speicherelement) jedoch relativ hoch. Er stellt das Kriterium für den Einsatz dieser Stromversorgungsvarianten dar. Grundsätzlich erscheint die Anwendung von Gleichspannungswandlern in den Rechnerbaugruppen nur dann sinnvoll, wenn nicht weitere Bauelemente mit drei Betriebsspannungen (z. B. RAM U256) im Rechner eingesetzt sind oder wenn nicht sowieso im Netzteil mehrere Betriebsspannungen erzeugt werden müssen.

Die Anwendung der in p-MOS-Technologie gefertigten (und somit zum System U808 passenden) langsamen Speicherelemente U501D (maskenprogrammiertes ROM), U551D (elektrisch programmierbares PROM) und U552C (EPROM) in Speicherbaugruppen des Systems U880 erfordert aufgrund des schlechteren Zeitverhaltens der Speicherelemente eine WAIT-Logik. Prinzipiell sind diese Elemente untereinander wiederum vollständig pinkompatibel, weisen jedoch im Zeitverhalten (insbesondere Chipauswahl) geringe Unterschiede auf. Um die direkte Austauschbarkeit zu gewährleisten, sollte die Anpassung im Zeitverhalten unter Worst-case-Bedingungen (auch innerhalb der Gruppe U501, U551, U552) erfolgen. Hierzu ist es erforderlich, in den Speicherzugriffszyklen (zu diesen Elementen) jeweils zwei WAIT-Zustände einzubauen (ähnlich zum Lösungsvorschlag bezüglich Minimalsystem; Bild 6.1.2). Diese Speicherbauelemente beeinflussen somit die Befehlsarbeitungszeiten des Prozessors U880 negativ und setzen die gesamte Leistungsfähigkeit des Rechnersystems herab. Die geringere Speicherkapazität (U501: 2 Kbit, U505: 8 Kbit) vergrößert zudem den Hardwareaufwand des Mikrorechners.

Die Speicherelemente U501, U551, U552 benötigen zwei Versorgungsspannungen

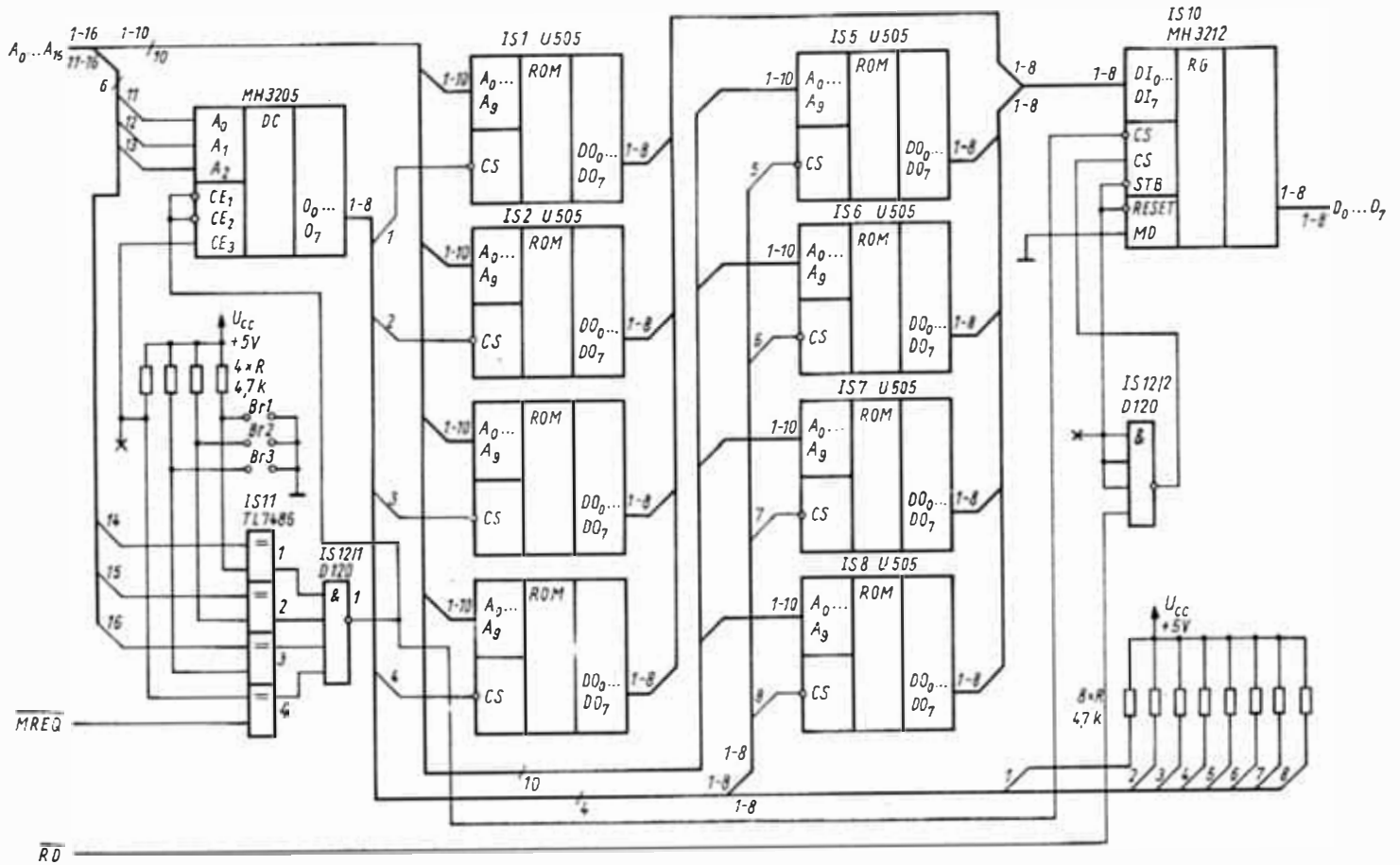


Bild 6.2.5. 8-Kbyte-Festwertspeicherbaugruppe mit U505 bzw. U555

( $U_{CC} = 5 \text{ V}$ ,  $U_{DD} = U_{GG} = -9 \text{ V}$ ). Der Strombedarf für  $I_{DD}$  (auch im Zusammenhang mit der geringen Speicherkapazität und, daraus resultierend, der großen Anzahl von einzusetzenden Bauelementen) erfordert i. allg. eine eigenständige Netzteillösung. Zur Verringerung des  $I_{DD}$ -Strombedarfs können die p-Kanal-Speicherelemente mit getakteter Versorgungsspannung  $U_{GG}$  ( $U_{GG} = +5 \text{ V}$ , Chip inaktiv, Schlafzustand;  $U_{GG} = -9 \text{ V}$ , Chip aktiv) betrieben werden. Das Umschalten der Spannung verschlechtert aufgrund der notwendigen Vorbereitungszeiten aber nochmals das Zeitverhalten der Elemente. Das Einfügen von mindestens einem weiteren WAIT-Zustand beim Speicherzugriff wird erforderlich. Im Bild 6.2.6 ist eine Schaltungsanordnung gezeigt, die eine 2-Kbyte-Speicherbaugruppe mit getaktet betriebenen Standardspeichern U501, U551, U552 darstellt. Hierbei ist die Verwendung des Datenbustreibers (MH3212) unbedingt notwendig, um negative L-Spannungen ( $U_{aL} \leq -1 \text{ V}$ ) vom Systembus zu trennen. Eingangsseitig erfordern die ROM-, PROM- und EPROM-Elemente einen H-Pegel von minimal  $U_{eH} \geq U_{CC} - 2 \text{ V}$ , so daß TTL-Standardelemente mit Pull-up-Widerständen versehen werden müssen (bei Bustreiber-elementen, z. B. DS8205, nicht erforderlich).

### 6.2.3.2. Statische Schreib/Lese-Speicher

In den Speicherbaugruppen des Systems U880 können die statischen RAM-Elemente U202 (n-MOS-Technologie) ohne zeitliche Anpassung direkt eingesetzt werden. Der U202 ist ein voll dekodierter RAM ( $1024 \times 1$ -organisiert) und hat getrennte Linien für die Datenein- und -ausgabe (Pins DI und DO). Die RAM-Blöcke mit dem U202 können erweitert werden, indem diese Ein-/Ausgänge der Blöcke untereinander parallelgeschaltet werden. Die Zusammenschaltung gleichwertiger Eingänge  $DI_n$  mit den Ausgängen  $DO_n$  ist aber wegen des internen Verhaltens nicht möglich. Der Einsatz bidirektionaler Datenbustreiber ist somit grundsätzlich erforderlich. Im Bild 6.2.7 ist die Schaltungsanordnung einer Speicherbaugruppe dargestellt, die einen 2-Kbyte-RAM-Speicher, realisiert mit U202, beinhaltet.

Zur Senkung des Leistungsbedarfs besitzt der RAM U202 einen Schlafzustand, der während eines Ruhezustands ( $\overline{CS}$  inaktiv) durch Absenken der Versorgungsspannung  $U_{CC}$  auf  $U_{CC} \geq 2 \text{ V}$  erreicht wird. Die Leistungsaufnahme eines Chips kann hierdurch von etwa  $P_V = 225 \text{ mW}$  auf  $P_{VS} = 70 \text{ mW}$  herabgesetzt werden. Die Anwendung dieser Schlafmode des U202 verschlechtert aber einerseits das Zeitverhalten des RAM (ein WAIT-Zustand in allen Speicherzyklen wird erforderlich), andererseits ist der Schaltungsaufwand zur Realisierung der Betriebsspannungsumschaltung aufgrund der zu schaltenden Ströme relativ hoch. Denkbar wäre der Einsatz eines umschaltbaren Gleichspannungswandlers (5 V auf 5 V bzw. 5 V auf 2 V). Die Anwendung dieser Schaltungsvariante ist wegen des relativ hohen Aufwands deshalb für Mikrorechner vorbehalten, die entweder einen großen statischen RAM-Bereich benötigen (anwendungsspezifisch bedingt) oder deren Schreib/Lese-Speicher durch externe Spannungsquellen (z. B. Akku) eine Datensicherung gegen Ausfall der Rechnerbetriebsspannung erfordern.

Eine Datensicherung gegen Spannungsausfall ist aber aufwandsgünstiger durch die Verwendung von statischen Schreib/Lese-Speichern in CMOS-SGT-Technologie zu erreichen. Als RAM-Bauelement bietet sich hierfür das zum U202 pinkompatible Element MH1902 an. Bei Einsatz dieses Bauelements in den Speicherbaugruppen des Systems U880 muß zur Anpassung an das Zeitverhalten der CPU ein WAIT-Zustand in den M1-Befehlslezugriff einbezogen werden. Zusätzlich kann ein weiteres Steuersignal zur Chipauswahl der RAM-Bauelemente herangezogen werden, das eine Information über den aktuellen Zustand der Rechnerversorgungsspannung beinhaltet. Damit kann ge-

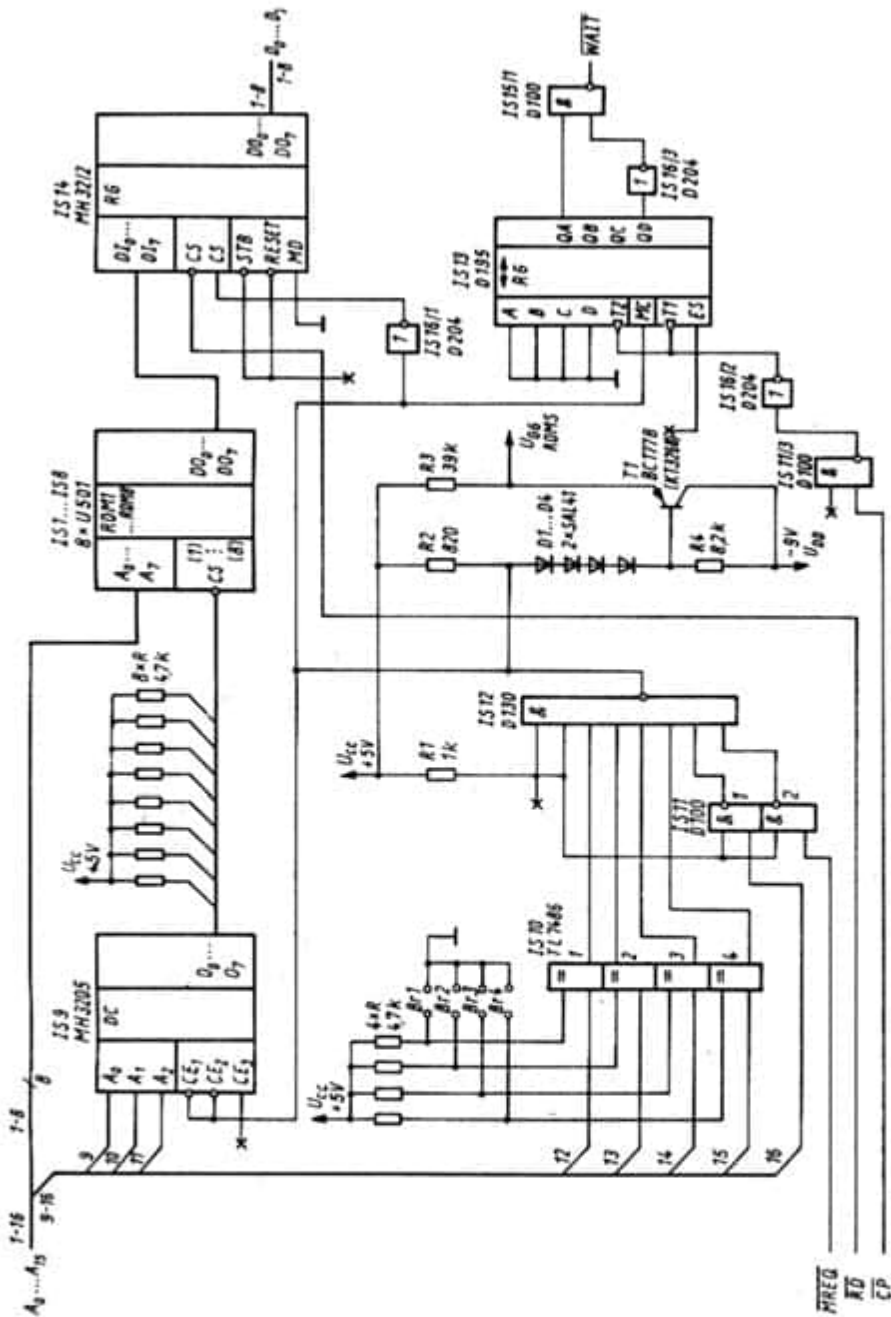


Bild 6.2.6. 2-Kbyte-Festwert-Speicherbaugruppe mit p-MOS-Elementen US01, US11 bzw. US12 und getaktet betriebener Spannungsversorgung der Speicher-elemente

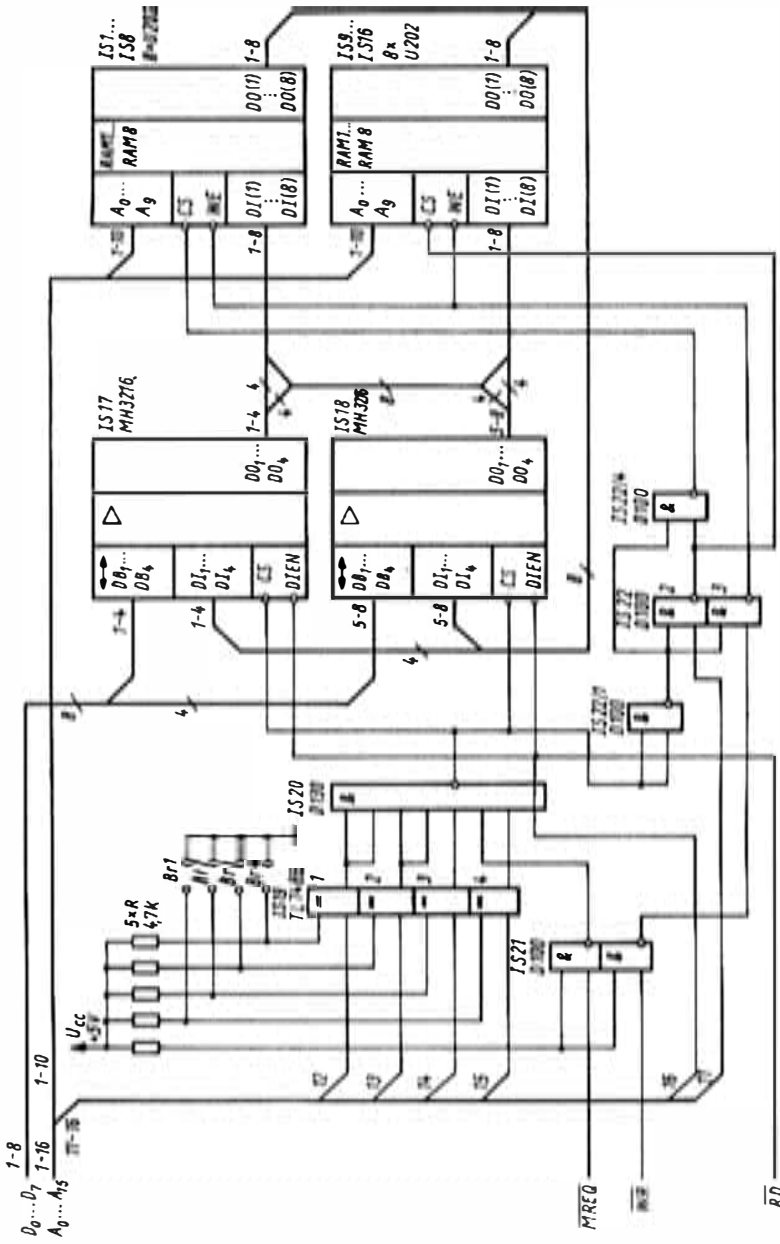


Bild 6.2.7. 2-Kbyte-Operativspeicherbaugruppe mit U202

währleistet werden, daß einerseits bei einem Ausfall der Versorgungsspannung nur noch der bearbeitete Speicherzugriff beendet und somit die  $\overline{CS}$ -Set-up-Zeit zum Absenken der Speicherversorgungsspannung eingehalten wird sowie andererseits nach Zuschalten der Versorgungsspannung stabile Verhältnisse bis zum ersten Datenzugriff gesichert sind. Im Bild 6.2.8 ist eine Schaltungsanordnung gezeigt, die eine frühzeitige Indikation eines Spannungsausfalls durch das Signal VG (Versorgungsspannung gültig bei H-Pegel) vornimmt. Dieses Signal kann aber auch in der Speicherbaugruppe durch Überwachung der Rechnerbetriebsspannung  $U_{CC}$  erzeugt werden, wenn gewährleistet wird, daß die RAM-Versorgungsspannung durch Stützkondensatoren für etwa  $1 \mu\text{s}$  gehalten wird, um ein korrektes Abschalten des  $\overline{CS}$ -Signals zu ermöglichen.

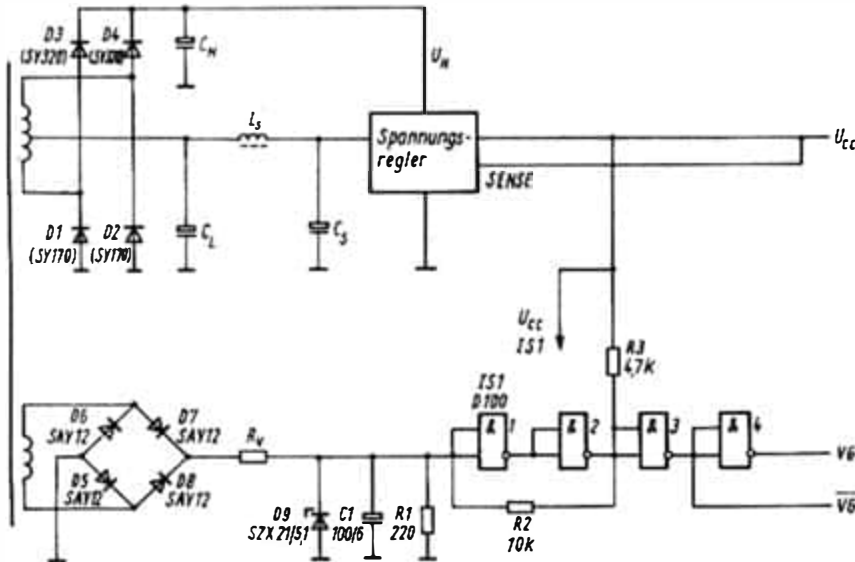


Bild 6.2.8. Schaltungsanordnung zur frühzeitigen Indikation von Netzausfällen in einer Netzteilbaugruppe

Ein weiteres wichtiges Kriterium für den Einsatz einer Datensicherung für einen RAM-Bereich ist die Auswahl einer geeigneten Spannungsquelle, die die Versorgung der CMOS-RAM bei Netzausfall sichert. Folgende Aspekte sind hierbei zu beachten:

- Kapazität der Spannungsquelle (bestimmt die maximale Dauer der Datensicherung bei Netzausfall)
- Einsatz wiederaufladbarer Elemente oder Primärelemente (abhängig von dem Anwendungsfall)
- nominale Zellenspannung (bestimmt Anzahl der einzusetzenden Zellen)
- Zellenendspannung, Entladespannung, Entladeverlauf (abhängig vom Typ der Zelle, erfordert evtl. Einsatz eines Transverters)
- Lebensdauer der Spannungsquelle (Serviceprobleme)
- Abmessungen, Masse (Anordnung auf der Leiterkarte oder extern)
- Kapazitäts-Größen-Verhältnis
- Verfügbarkeit, Preis.

Der Einsatz von Primärelementen ist Anwendungsfällen vorbehalten, bei denen Spannungsausfälle selten auftreten (z. B. Einsatz von billigen Zink-Kohle-Zellen; R 14 u. dgl.) oder bei denen es auf hohe Batteriekapazitätswerte bei kleinsten geometrischen Abmes-

sungen ankommt (z. B. Einsatz von Silberoxid-Uhrenzellen). Üblicherweise kommen aber Sekundärelemente (Akkus) zum Einsatz, deren Kapazitätswerte zwischen 200 mA · h (für Leiterkartenunterbringung) und einigen Amperestunden (externe Anordnung) liegen. Im Zusammenhang mit der Anwendung des CMOS-RAM MH1902 bietet sich die Verwendung von drei Nickel-Cadmium-Akkuzellen an. Hinsichtlich des schaltungstechnischen Aufwands sollten überladungsfeste Zellen eingesetzt werden, da sonst eine entsprechende Ladeschaltung mit auf der Leiterkarte untergebracht werden muß bzw. aufwendige Serviceanforderungen an die Baugruppe bestehen.

Hinsichtlich der Umschaltung der RAM-Versorgungsspannung von der Rechnerversorgungsspannung  $U_{CC}$  auf die Schlafspannung  $U_{DR}$  bestehen zwei prinzipielle Schaltungsmöglichkeiten, die im Bild 6.2.9 dargestellt sind. Die Diodenkopplung (Bild 6.2.9b) erfordert im  $U_{CC}$ -Zweig den Einsatz einer Diode mit geringer Flußspannung (z. B. Schottkydiode, Ge-Diode), um den Toleranzbereich der RAM-Versorgungsspannung nicht zu unterschreiten. Eine Modifikation dieser Prinziplösung ist durch den Einsatz eines pnp-Schalttransistors möglich, der ebenfalls eine geringe Emittterkollektorspannung  $U_{CE}$  bei Übersteuerung aufweist ( $U_{CE} \approx 0,2 \text{ V}$ ). Eine entsprechende Schaltungsanordnung ist im Bild 6.2.10 dargestellt. Abhängig von der Spannungsabschaltung wird ein Signal VG (alternative Möglichkeit zum Bild 6.2.8) gebildet. Zusammenfassend ist im Bild 6.2.11 eine Schaltungsvariante gezeigt, die die Chipauswahl eines CMOS-RAM-Speicherblocks vornimmt. Mit Kodiersteckern bzw. Miniaturvorwahlschaltern kann vom Anwender ein Speicherschreibschutz für den Block festgelegt werden. Diese Maßnahme ist für Softwareentwicklungsaufgaben günstig, bei denen ein ROM-Bereich eines Rechners durch eine CMOS-RAM-Speicherbaugruppe simuliert werden soll.

### 6.2.3.3. Dynamische Schreib/Lese-Speicher

Dynamische RAM-Bauelemente haben gegenüber statischen n-MOS-Elementen vor allem Vorteile hinsichtlich der Verlustleistung (bezogen auf die Speicherkapazität), der Zugriffszeit und der Verfügbarkeit großer Speicherkapazitäten. (Verwendung von Eintransistorspeicherzellen in dynamischen Speichern ermöglicht große Packungsdichte auf dem Chip.) Der dynamische RAM-Speicher U256 hat eine Speicherkapazität von 16 Kbit (voll dekodiert) und ist ein- und ausgangsseitig vollständig TTL-kompatibel. Seine zeitbezogenen Parameter erlauben einen Einsatz im U880-System ohne Verlängerung der Standardspeicherzyklen des Prozessors U880. Andere dynamische Speicherelemente (z. B. U253) weisen keine wichtigen Einsatzvorteile gegenüber verfügbaren statischen Elementen auf; ihre Anwendung im System U880 wird deshalb nicht betrachtet.

Bei der Einbeziehung von dynamischen Speicherelementen in den Speicherbaugruppen des Systems ist die nachfolgende beschriebene Besonderheit der CPU U880 zu berücksichtigen. Das  $\overline{\text{RAS}}$ -Signal für die dynamischen RAM wird üblicherweise durch logische Verknüpfung der höchstwertigen nicht im RAM verarbeiteten Adreßlinien (bei 16-Kbit-RAM:  $A_{14}, A_{15}$ ) und dem Steuersignal  $\overline{\text{MREQ}}$  der CPU gebildet. Es dient im RAM zur Einspeicherung der (niederwertigen) Zeilenadresse. Am Ende eines M1-Befehlslesezyklus kann nun der Fall eintreten, daß nach der steigenden Flanke des Taktzustands T3 bereits ein Adreßwechsel stattfindet, obwohl das Signal  $\overline{\text{MREQ}}$  noch aktiv ist. Im Bild 6.2.12 sind die entsprechenden Zeitverhältnisse dargestellt. Bei einem entsprechenden Inhalt des CPU-Refreshregisters R kann also ein erneutes kurzes  $\overline{\text{RAS}}$ -Signal erzeugt werden. Diese zweite fallende Flanke mit dem nachfolgenden sehr kurzen, aktiven Pegel von  $\overline{\text{RAS}}$  bewirkt im dynamischen RAM das Auslösen eines unvollständigen Lesezyklus. Da die

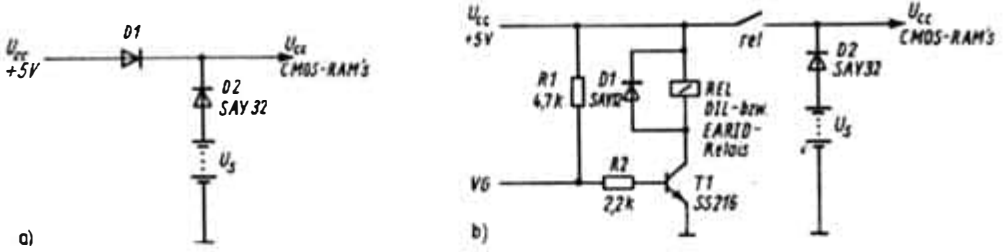


Bild 6.2.9. Prinzipielle Varianten der Betriebsspannungsumschaltung an CMOS-RAM bei Versorgungsspannungsausfall

- a) Umschaltung mit Dioden
- b) Einsatz eines Relais

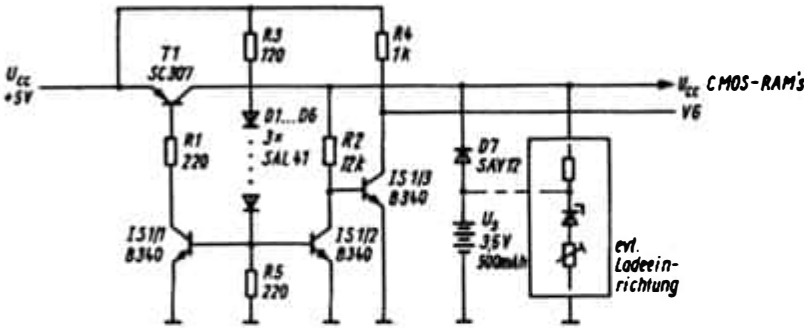


Bild 6.2.10. Schaltungsanordnung zur Betriebsspannungsumschaltung mit Transistoren

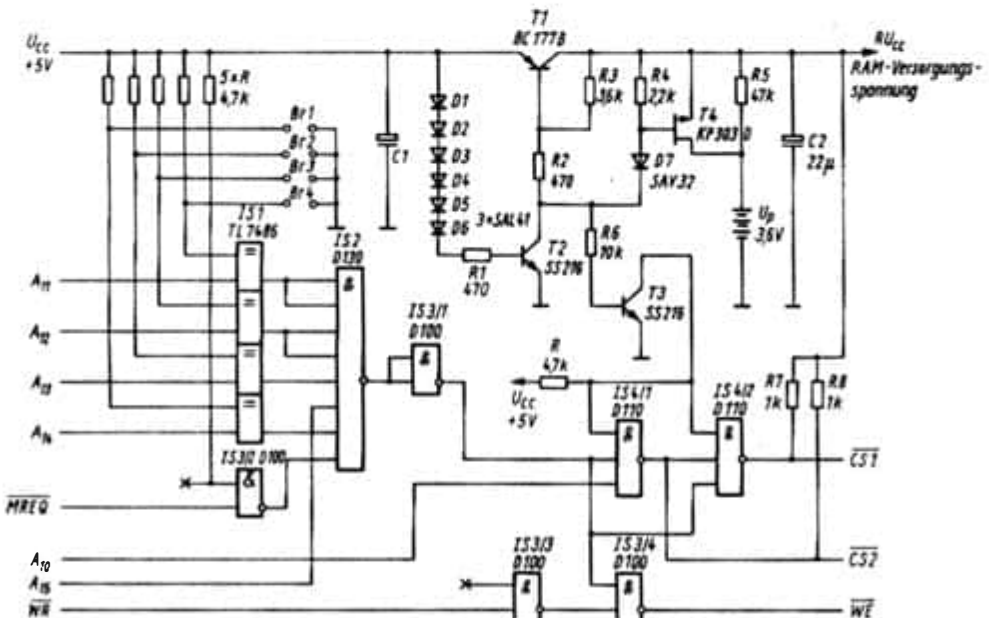


Bild 6.2.11. Chipauswahllogik für CMOS-RAM-Speicherbaugruppen

- $CS1$ ,  $CS2$  Chipfreigabesignale für zwei 1-Kbyte-CMOS-RAM-Blöcke
- $WE$  Schreibfreigabesignal

dynamischen RAM-Zellen prinzipiell zerstörend gelesen werden (mit nachfolgendem Wiedereinschreiben in vollständigen Lesezyklen), kann das Löschen der gesamten willkürlich angewählten Speicherzeile erfolgen. Ein unvollständiger Refreshlesezyklus des RAM kann ebenfalls zu Beginn des CPU-Refreshzyklus ausgelöst werden, wenn  $\overline{MREQ}$  nach Aktivierung des  $\overline{RFSH}$ -Signals noch aktiv ist (z. B. aufgrund hoher TTL-Verzögerungszeiten des Steuersignals  $\overline{MREQ}$ ). Um in Systemen mit dynamischen Speichern derartige Konflikte zu vermeiden, müssen die höchstwertigen Adreblinien und evtl. auch das  $\overline{RFSH}$ -Signal mittels  $\overline{MREQ}$  zwischengespeichert oder eine Verzögerung dieser Signale erreicht werden. Eine entsprechende Schaltungsanordnung zeigt Bild 6.2.13. Zu beachten ist hierbei, daß das Speicherblockauswahlsignal  $\overline{SEL}$  bzw. das  $\overline{RFSH}$ -Signal mit der steigenden Flanke von  $\overline{MREQ}$  abgespeichert und daß der Trigger IS1 (Bild 6.2.13) bei inaktiver Auswahl gesetzt wird. Das ist erforderlich, damit bei einem Lesezyklus, der nicht auf den gezeigten Speicherblock zugreift, kein Störimpuls durch den sonst von einem vorhergehenden Speicher- oder Refreshzugriff evtl. aktiven Triggerausgang  $\overline{Q}$  auf  $\overline{RAS}$  erzeugt wird.

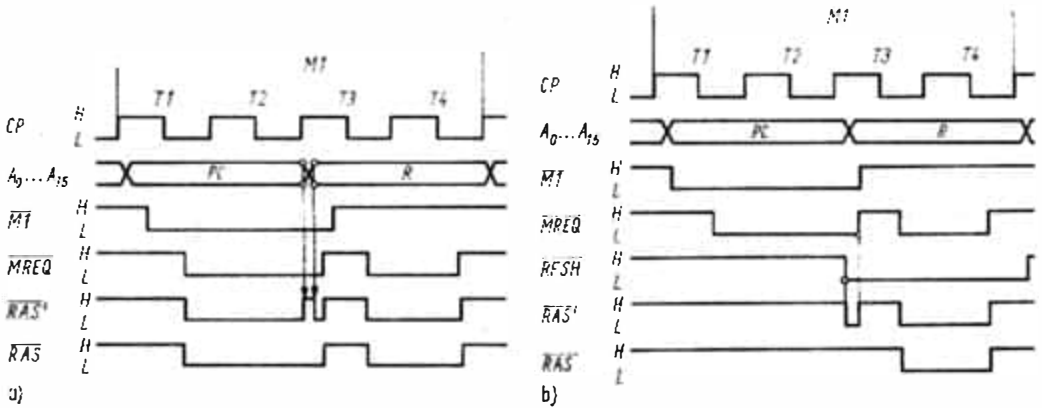


Bild 6.2.12. Zeitverhalten dynamischer Speicherbaugruppen im M1-Befehlslesezyklus der CPU

- a) Konflikte durch frühzeitigen Adreßwechsel
- b) Konflikte durch vorzeitiges Auslösen des Refreshzyklus
- M1 Maschinenzklus der CPU
- CP Systemtakt (T1, T2, T3, T4 Taktzustände der CPU)
- $A_0 \dots A_{15}$  Belegung des CPU-Adreßbusses (PC Programmzählerstand; R Inhalt des Refreshregisters auf  $A_0 \dots A_6$ )
- $\overline{M1}$ ,  $\overline{MREQ}$ ,  $\overline{RFSH}$  Steuersignalausgänge der CPU
- $\overline{RAS'}$  fehlerhaftes Zeilenanwahlsignal der dynamischen RAM
- $\overline{RAS}$  korrektes Zeilenanwahlsignal

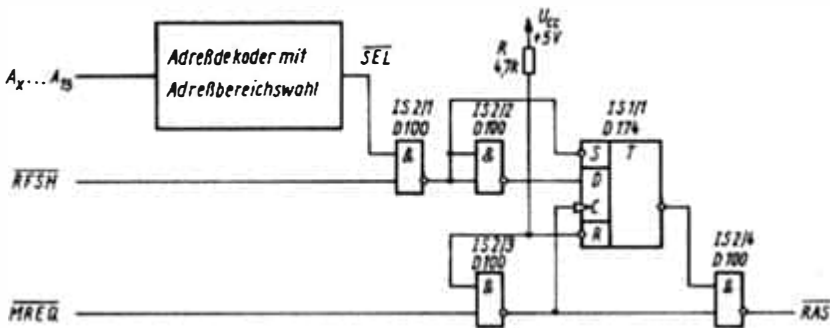


Bild 6.2.13. Schaltung zur Behebung von  $\overline{RAS}$ -Konflikten in dynamischen Speicherbaugruppen

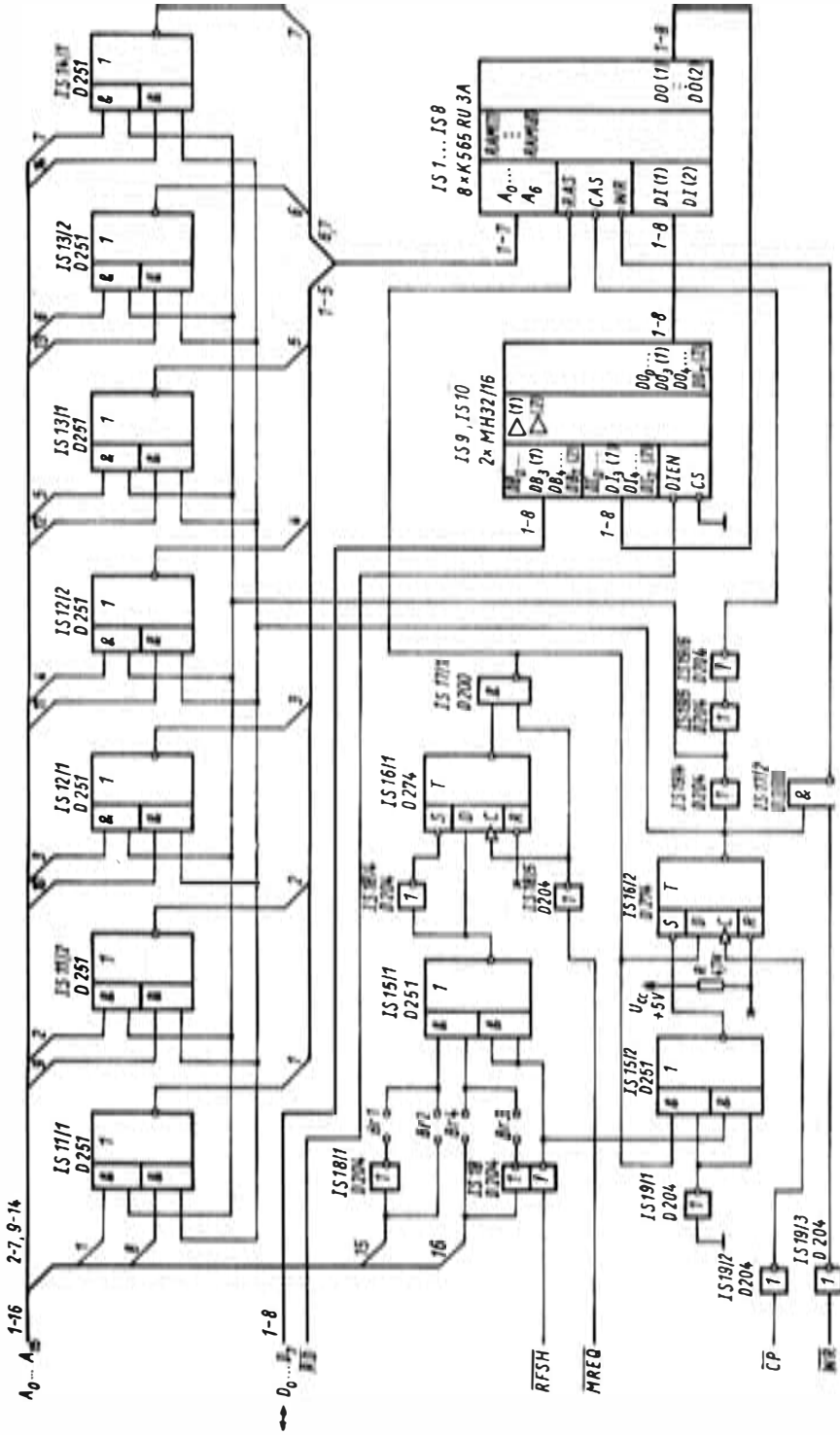


Bild 6.2.14. 16-Kbyte-Operativspeicherbaugruppe mit K565RU3A bzw. U256

Zusammenfassend ist im Bild 6.2.14 die Schaltung einer Speicherbaugruppe dargestellt, die eine Kapazität von 16 Kbyte aufweist. Die gezeigte Speicherbaugruppe besteht im wesentlichen aus den Funktionseinheiten Adreßmultiplexer, Adreßbereichsdekoder, Ansteuerlogik, Datenbuspuffer und dynamischer RAM-Block. Der Adreßmultiplexer dient zur Umschaltung der 14 notwendigen CPU-Adreßlinien auf die Adreßlinien der RAM. In den RAM werden mit den aktiven Flanken der  $\overline{\text{RAS}}$ - und  $\overline{\text{CAS}}$ -Signale nacheinander die Zeilen- und Spaltenadresse eingespeichert. Beim Aufbau des Adreßmultiplexers muß beachtet werden, daß die niederwertigen CPU-Adressen ( $A_0$  bis  $A_6$ ) als Zeilenadressen (eingespeichert mit  $\overline{\text{RAS}}$ ) verwendet werden, da in den Refreshzyklen des Prozessors auf diesen Linien der Inhalt des CPU-R-Registers ausgegeben wird. Eine Invertierung und eine eventuelle Vertauschung der Adreßwertigkeiten beim RAM-Adreßbus haben keine Bedeutung, da in allen Fällen der Adreßbereich eindeutig zugeordnet ist. Die anderen Funktionseinheiten weisen keine Besonderheiten auf; das Zeitverhalten der RAM-Elemente wird unter Worst-case-Bedingungen erfüllt.

#### 6.2.4. Rechnerperipherie

Die peripheren Baugruppen haben die grundsätzliche Aufgabe, die Kommunikation des Mikrorechners zur Umwelt zu ermöglichen. Sie bestimmen deshalb ebenfalls die Leistungsfähigkeit eines Mikrorechnerkonzeptes. Die unterschiedlichen Anforderungen an den Informationsaustausch Rechner-Umwelt bestimmen die jeweilige Konfiguration einer Peripheriebaugruppe. Forderungen an die Peripheriebaugruppe bestehen deshalb hinsichtlich der Realisierung von Standardschnittstellen zur Anpassung von an schlußmäßig genormten Geräten (z. B. Seriendrucker, Bildschirmbedieneinheiten, Lochbandleser und -stanzer) und von Schnittstellen, die an typische Einsatzanforderungen des Mikrorechners angepaßt sind (analoge Ein-/Ausgänge zur prozeßnahen Rechnerkopplung; digitale Ein-/Ausgänge mit Leistungsstufen, Potentialtrennung; Zähler/Zeitgeber-Kanäle). An diesen Faktoren läßt sich zusammenfassen, daß die Konfiguration und Anpaßfähigkeit der peripheren Baugruppen eines Mikrorechnersystems dessen Hauptanwendungsgebiete wesentlich mitbestimmen.

Standardschnittstellen können entsprechend den Anwendungsanforderungen durch parallele (großer Datendurchsatz, hohe Anforderungen an Übertragungsgeschwindigkeit) und serielle (geringe Anzahl von Übertragungsleitungen, große Entfernungen, Modemanwendung) Ein-/Ausgabe-Kanäle realisiert werden. Parallele Standardschnittstellen weisen aufgrund der byteweisen Übertragung einen hohen Datendurchsatz auf und erfüllen damit hohe Anforderungen an die Übertragungsgeschwindigkeit. Die Datenübertragung kann durch einen Handshakebetrieb gesichert werden (die PIO realisiert einen bidirektionalen Handshakedatenaustausch in der Betriebsart 2). Aufgrund der relativ großen Anzahl von Übertragungsleitungen und des notwendigen Treiber- und Empfänger-aufwands können aber nur geringe Entfernungen (typ. 3 m) überbrückt werden.

Im Bild 6.2.15 ist eine parallele Standard-Ein-/Ausgabe-Baugruppe dargestellt. Diese Peripheriebaugruppe realisiert je einen Eingabe- und Ausgabekanal der SIF-1000-Anschlußsteuerung nach TGL 26456. Die Eingangs- und Ausgangspegel sind als KME10-Pegel (TTL) definiert. Die SIF-1000-Anschlußsteuerung beinhaltet Übertragungssteuersignale (RUF, Ausgang; END, Eingang), Kommandosignale (KOM 1 bis KOM 3; Ausgänge), Statussignale (STA 1 ... STA 3, Eingänge) und Datensignale (DAT 1 ... DAT 8). Die Signale eines SIF-1000-Kanals können somit vorteilhaft durch je eine IS U855 erzeugt bzw. überwacht werden. Hierbei sind dem PIO-Kanal A die Daten-

signale zugeordnet, die je nach SIF-1000-Kanaltyp als Ein- oder Ausgänge beschaltet sind. Vom PIO-Kanal B erfolgt entsprechend die Überwachung der SIF-1000-Steuersignale. Hierzu muß das PIO-Port in der Bitmode betrieben werden, um die notwendige Ein-/Ausgangs-Konfiguration zu erreichen. Gleichzeitig bietet diese Maßnahme den Vorteil, daß die SIF-1000-Steuersignale durch Interruptfunktionen überwacht werden können. Die SIF1000-Peripheriebaugruppe enthält zur systemseitigen Anpassung der PIO des weiteren einen Adreßdekoder mit Adreßbereichsvorwahl, einen Datentreiber mit Datenrichtungsumschaltlogik und eine Interruptumgehungslogik.

Serielle Standardschnittstellen haben im Vergleich zu parallelen Schnittstellen den Vorteil, mit sehr wenig Leitungen (i. allg. mit zwei) eine Datenübertragung zu gewährleisten. Dadurch wird es möglich, den Hardwareaufwand zur Realisierung von Leitungsendern und -empfängern beachtlich zu steigern. Große Entfernungen können somit bei der Datenübertragung überbrückt werden. Zur Leitungstreibung sind Stromschalter in Stromschleifen mit  $I = 20$  mA, Spannungsschalter bei V24- (RS232-) Schnittstellen oder Modulator/Demodulator-Schnittstellen üblich. Mit seriellen Standardschnittstellen sind Datenübertragungsraten im Bereich  $DR = 50$  baud ... 192 kbaud typisch erreichbar. Ein wichtiger Gesichtspunkt im Zusammenhang mit seriellen Schnittstellen ist die Möglichkeit, die serialisierten Daten aufwandgünstig auf Massenspeichermedien abzuspeichern. Die Anwendungsbreite serieller Schnittstellen reicht hierbei vom Einsatz einfacher NF-Kassettenmagnetbandgeräte als billige Speichereinheit bis zur Ansteuerung von Folienspeichern (floppy disk), Festplattenspeichern (hard disk) und kommerziellen Magnetbandgeräten.

Im Bild 6.2.16 ist eine serielle Ein-/Ausgabe-Baugruppe dargestellt, die mit einer IS U856 (SIO) bestückt ist. Die Anpassung an die Übertragungsleitung wird wahlweise durch eine Stromschnittstelle (20 mA) oder eine V24-Schnittstelle realisiert. Die Stromschnittstelle erfordert die externe Einspeicherung eines Konstantstroms in die Übertragungsleitung. Die Stromstärke von  $I = 20$  mA ist international für serielle Datenübertragungen üblich und gewährleistet die optimale Ausnutzung verfügbarer optoelektronischer Bauelemente und eine sichere Datenübertragung über große Entfernungen (bis etwa 10 km). Die alternative Nutzung der RS232-Spannungsschnittstelle wird durch die Erzeugung der  $\pm 12$ -V-Versorgungsspannungen in der Baugruppe ermöglicht bzw. vereinfacht. Die Peripheriebaugruppe weist des weiteren einen programmierbaren Zähler/Zeitgeber auf, der durch eine IS U857 (CTC) realisiert wird. Der CTC wird benutzt, um die Empfänger- und Sendetaktversorgung der SIO vorzunehmen. Zum einen wird dadurch der Hardwareaufwand der Baugruppe verringert, und dem Anwender stehen die freien CTC-Kanäle zur Verfügung; zum anderen ergibt sich die Möglichkeit, die Datenübertragungsraten der SIO durch eine entsprechende Programmierung des CTC den Einsatzanforderungen der Baugruppe anzupassen. Die Baugruppe bietet somit weitreichende Programmiermöglichkeiten hinsichtlich der Realisierung der Übertragungsbetriebsart (synchron/asynchron), ihrer Eigenschaften (z. B. Datenformat, Paritäts- und CRC-Überprüfung) und Geschwindigkeit. Sie ist somit programmtechnisch an alle üblichen Standardschnittstellen anpaßbar. Um als Schnittstelle zu datenabspeichernden Geräten eingesetzt zu werden, kann nach Zwischenschaltung eines relativ einfachen Interfaces die Ansteuerung eines NF-Kassettenbandgeräts erfolgen. Bei der Schaltung einer Folienspeicheransteuerbaugruppe müßte zusätzlich eine Steuereinheit (z. B. mit einer PIO) eingesetzt werden, um Kontroll- und Steuersignale des Laufwerks (z. B. Schrittmotoransteuerung, Sektorimpulsauswertung u. dgl.) zu überwachen. Der Hardwareaufwand einer derartigen Ansteuerbaugruppe ist etwa 50% höher als in der dargestellten seriellen Ein-/Ausgabe-Baugruppe. Die weiteren Funktionseinheiten der Baugruppe – Adreßdekoder,

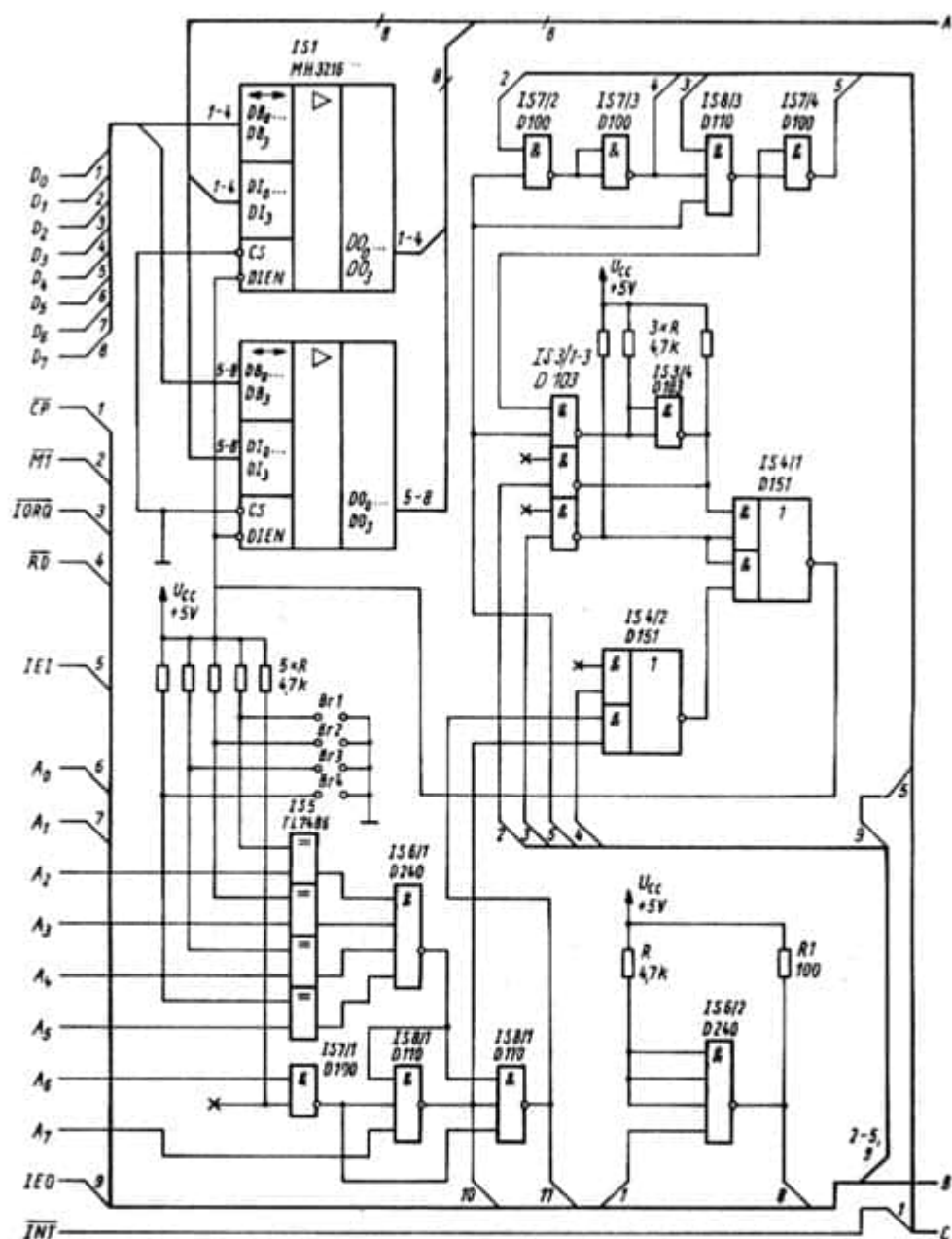
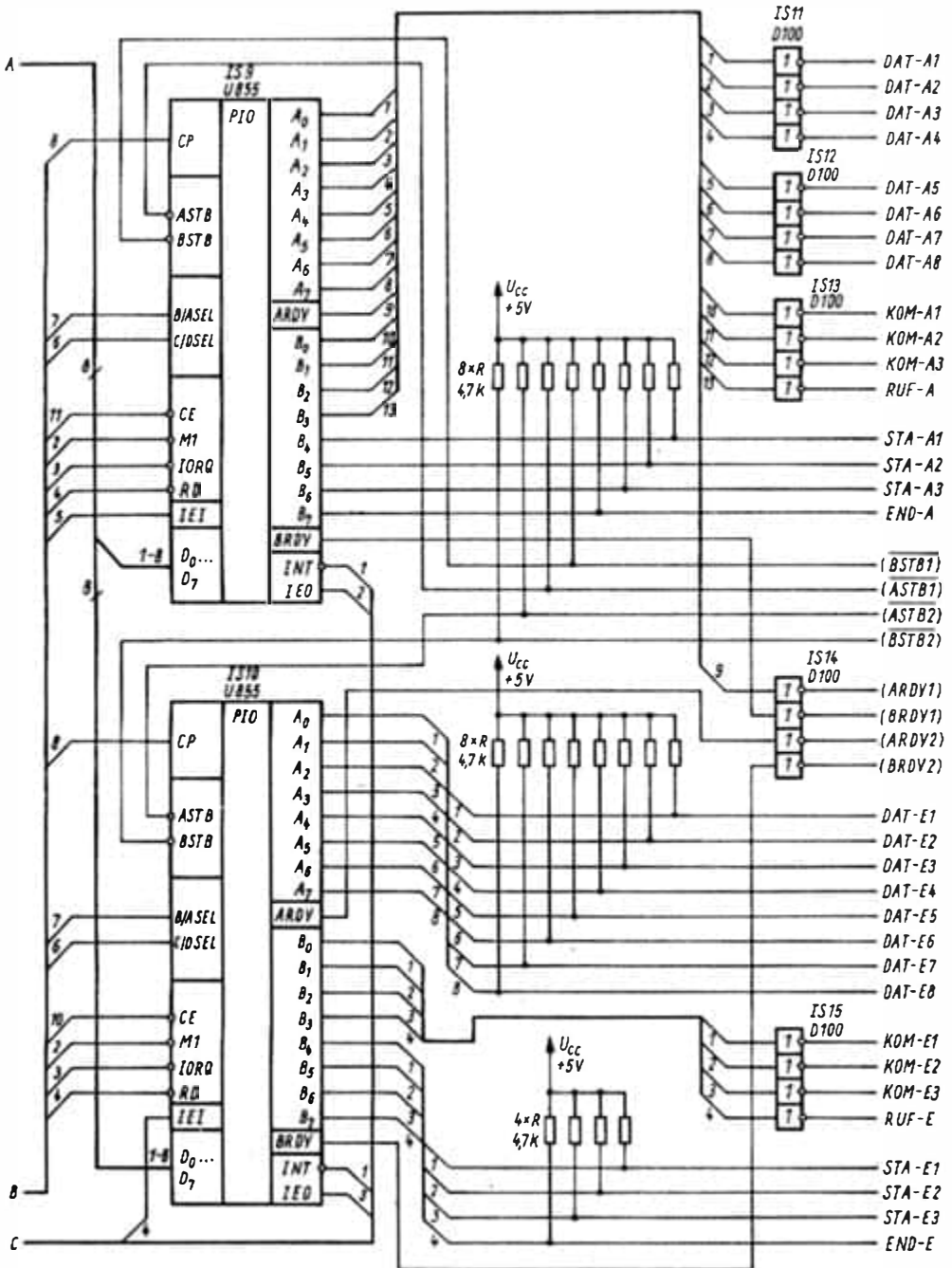
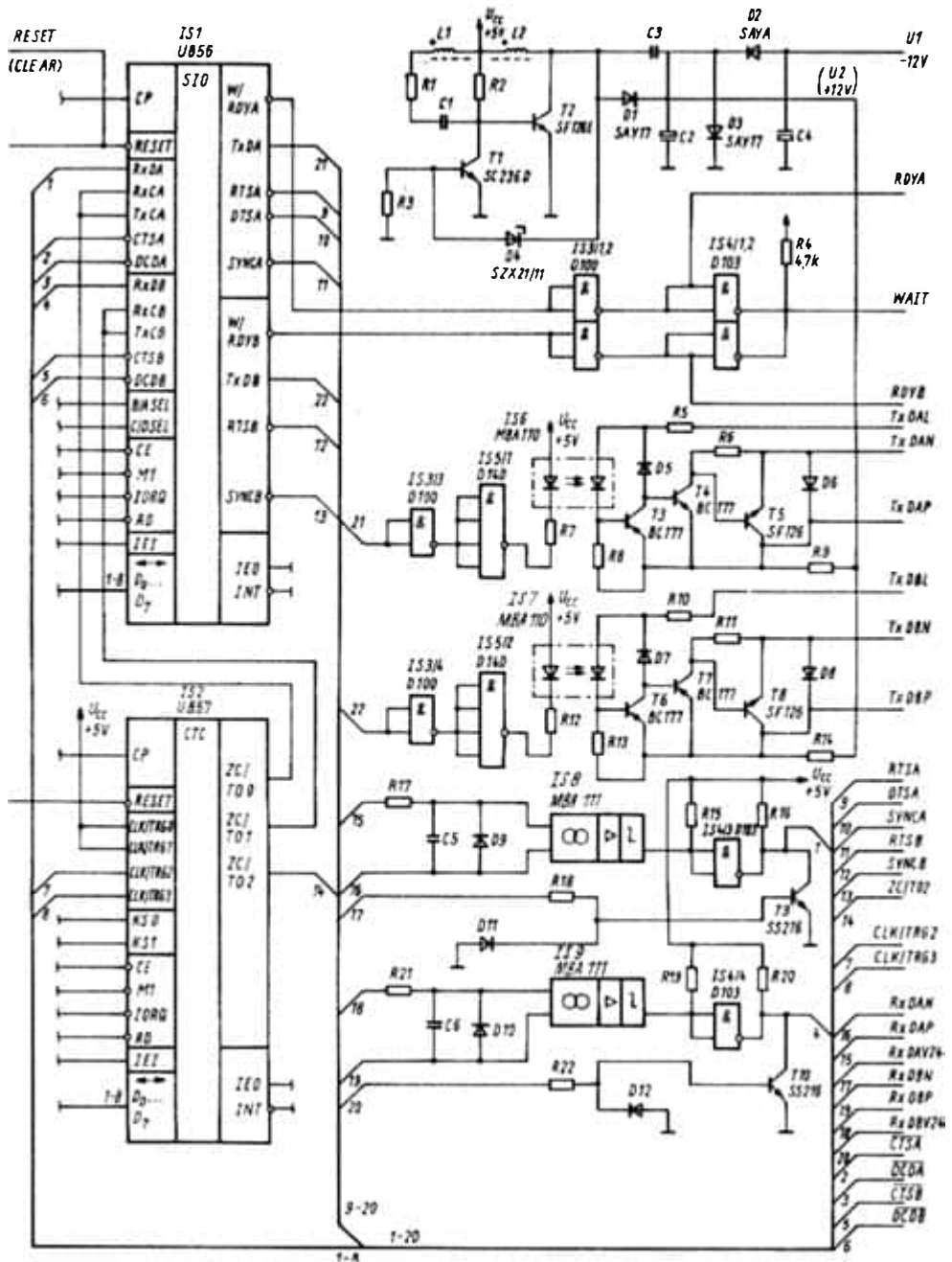


Bild 6.2.15. SIF1000-Standard-Ein-/Ausgabe-Baugruppe





**Bild 6.2.16. Serielle Standard-Ein-/Ausgabe-Baugruppe**

Die systemseitige Ansteuerung der SIO und des CTC entspricht der Anschaltung der PIO im Bild 6.2.15

(Die Steuersignaleingänge **RESET** müssen zusätzlich mit dem systemweiten Rücksetzsignal verknüpft werden)

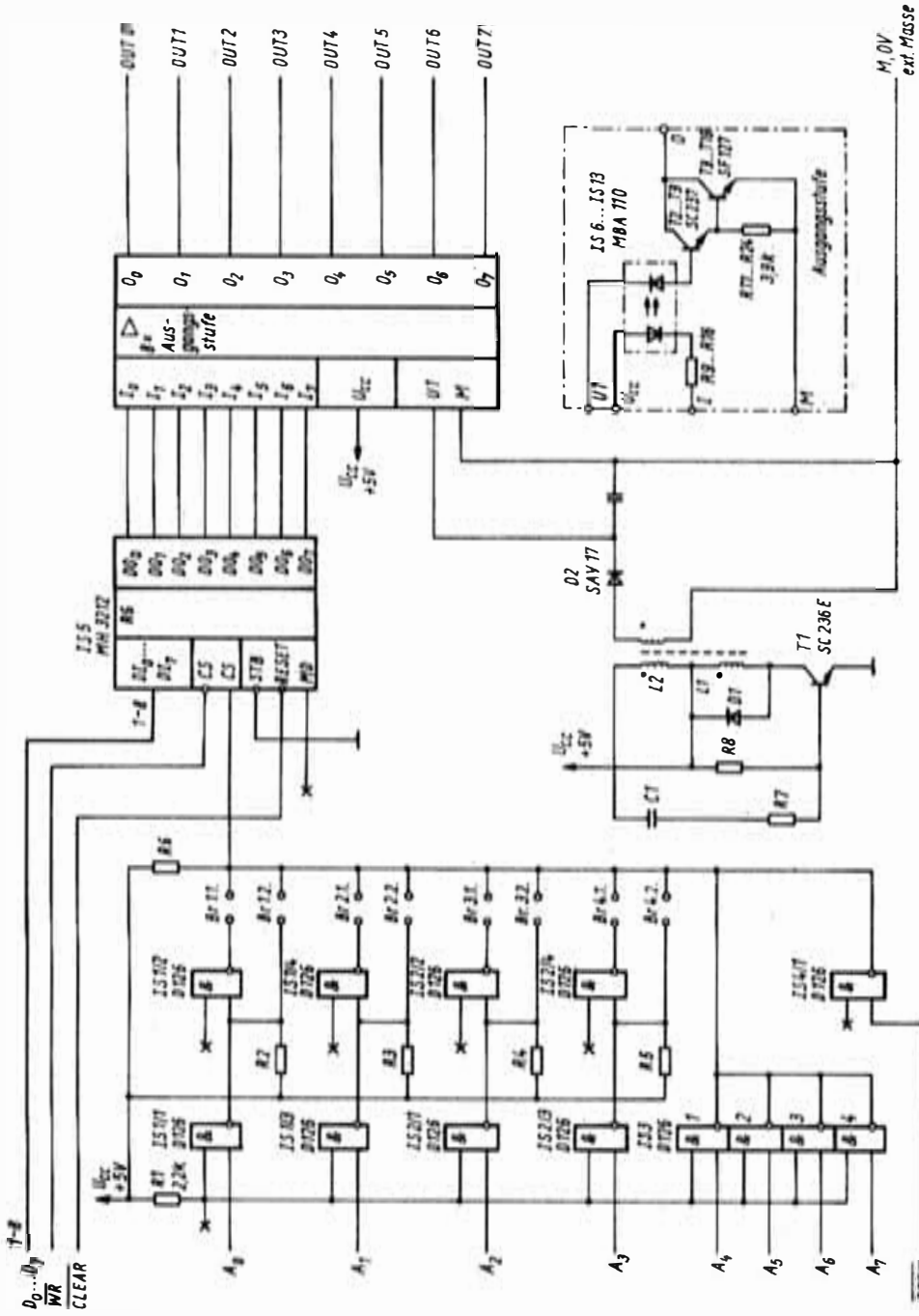


Bild 6.2.17. Parallele Ausgabebaugruppe mit ausgangsseitigen (anwenderseitigen) Schaltstufen und Potentialtrennung vom Rechner

Bereichsvorwahl, Datenbustreiber mit Richtungssteuerlogik und die Interruptumgehungslogik – weisen keine Besonderheiten auf.

Beim Einsatz von Mikrorechnerkonzepten im Anwendungsbereich der Prozeßsteuerungen steht die möglichst aufwandsoptimale Kopplung des Rechners zur Peripherie im Vordergrund. Neben der Verwendung vorhandener Standardschnittstellen durch Modifizierung ihrer Eigenschaften (z. B. Einsatz der SIF-1000-Baugruppe mit anderer programmtechnischer Ansteuerung) kommt die Konzeption von zugeschnittenen, prozeßnahen Schnittstellen in Betracht.

Im Bild 6.2.17 ist eine parallele 8-bit-Ausgabebaugruppe dargestellt, die ausgangseitig eine Potentialtrennung gegenüber dem Rechner aufweist und Schaltstufen besitzt. Eine derartige relativ einfache Baugruppe kann für die Ansteuerung von Schaltschützen bzw. Relais in Industrieanlagen eingesetzt werden. Die Schaltstufen, realisiert mit Transistoren SF127E, können bis 200 mA übernehmen und genügen hinsichtlich der Spannungsfestigkeit den entsprechenden Forderungen (24 V) in Anlagen der BMSR-Technik. Zur Datenspeicherung wird ein Register MH3212 benutzt, da hierbei im Vergleich zum Einsatz der PIO U855 der Schaltungsaufwand sinkt. Das Register MH3212 ist in der Lage, die Optokoppler MB110 direkt anzusteuern. Eine Interruptfähigkeit der Baugruppe ist wegen der Spezifik der Einsatzmöglichkeiten nicht notwendig. Die parallele Ausgabebaugruppe beinhaltet weiterhin die Funktionseinheit Adreßdekoder mit Adreßbereichsvorwahl. Der Adreßdekoder ist alternativ zum Einsatz von Bausteinen 7486 (K155LP5, TL7486, UCY7486) mit Schaltkreisen D126 realisiert. Hierbei ist zu beachten, daß bei der IS D126 ausgangseitig ein geringerer H-Reststrom gegenüber der IS D103 gewährleistet wird, der den Einsatz in der vorwählbaren Wired-or-Verknüpfung erlaubt. Die sonst in Peripheriebaugruppen üblichen Funktionseinheiten Datenbustreiber und Interruptumgehungslogik entfallen.

Bei der Gestaltung von digitalen Eingabeschnittstellen in industriellen Anlagen werden an prozeßnahe Konzepte vor allem Forderungen wie Potentialtrennung, Störschutzbeschaltung der Eingangslinien und Einhaltung von Mindestwerten der Kontaktströme und -spannungen bei der Abfrage gestellt. Derartige Eingabegruppen können dann direkt zur Überwachung von Relaiskontakten, Endlagenschaltern u. dgl. in den Anlagen eingesetzt werden. Im Bild 6.2.18 ist eine digitale Eingabebaugruppe dargestellt, die eine derartige Abfrage unter Prozeßbedingungen vornimmt. Sie beinhaltet systemseitig die Funktionseinheiten Datenbustreiber mit Richtungssteuerung, Adreßdekoder mit Bereichsvorwahl, Interruptumgehungslogik und die parallele Ein-/Ausgabe-Einheit U855 (PIO). Anwenderseitig sind Optokoppler zur Potentialtrennung, Störschutzbeschaltung der Eingänge und Strombegrenzungswiderstände  $R$  eingesetzt. Die Störchutzbeschaltung ist hierbei so dimensioniert, daß typische Belastungen der Leitung keine Zerstörungen in der Baugruppe bewirken. Solche typischen Belastungen werden üblicherweise durch die Entladung eines auf eine Spannung von  $U_C = 250 \text{ V}$  aufgeladenen Kondensators mit  $C = 1 \mu\text{F}$  über einen Widerstand mit  $R_L = 100 \Omega$  auf die Leitung simuliert. Die Strombegrenzungswiderstände  $R$  sind so dimensioniert, daß bei einer externen Spannungsquelle mit  $U_B = 24 \text{ V}$  der Strom durch die Relaiskontakte (und somit auch durch die Optokoppler) auf  $I = 100 \text{ mA}$  begrenzt wird. Diese Stromstärke stellt ein Optimum zwischen Schaltungsaufwand, Kontaktsicherheit der Relaiskontakte (Spannungsabfall am Übergangswiderstand der Kontakte, z. B. durch Oxydation oder Verschmutzung, wird vermindert) und Strombelastung der Optokoppler dar. Programmtechnisch können die PIO-Kanäle in der Mode 3 (Bitbetrieb) betrieben werden, um durch bestimmte Kontaktbelegungen Interruptfunktionen auszulösen (Alarmbetrieb), oder die Baugruppe kann im Polling abgefragt werden.

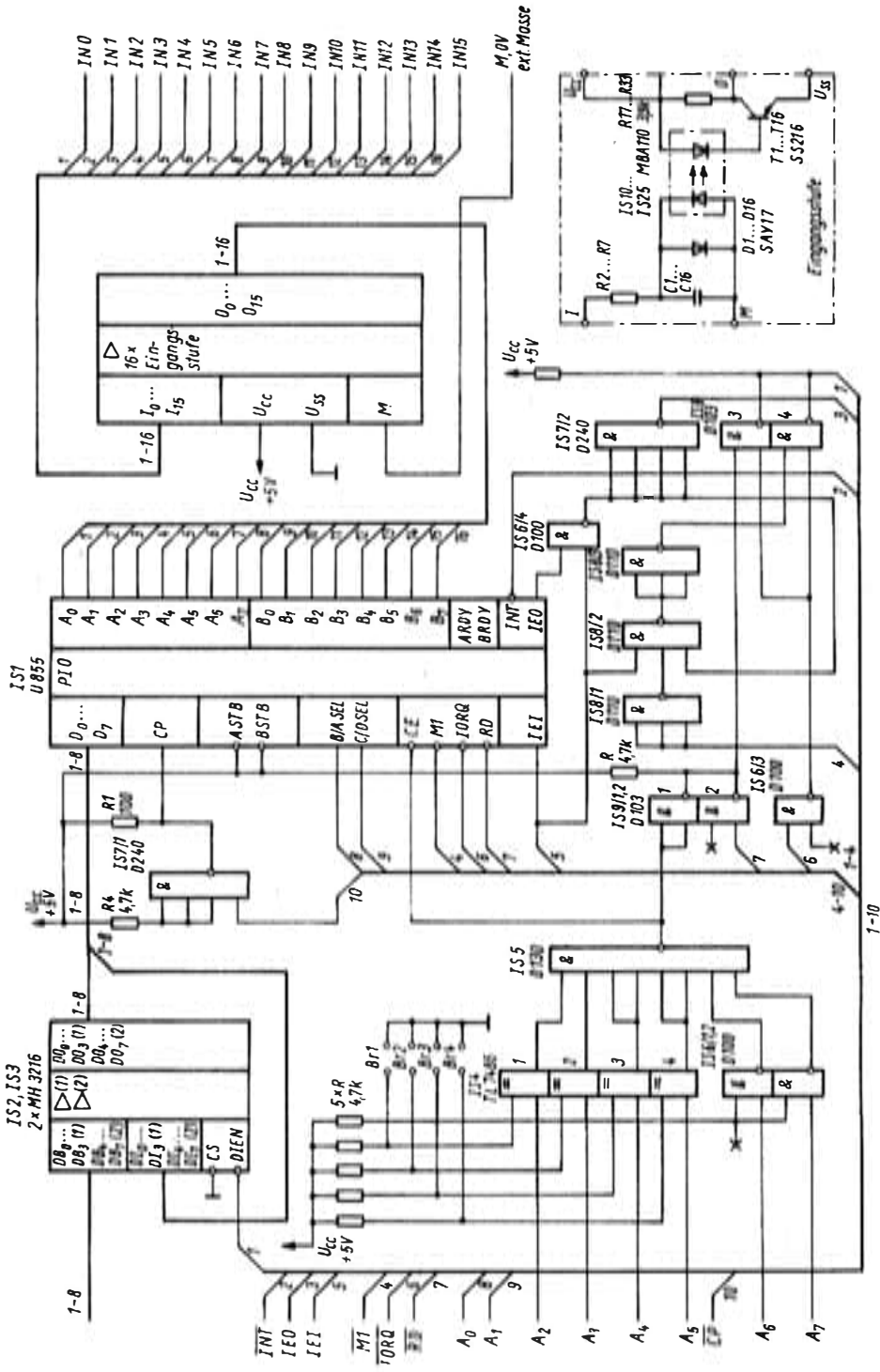
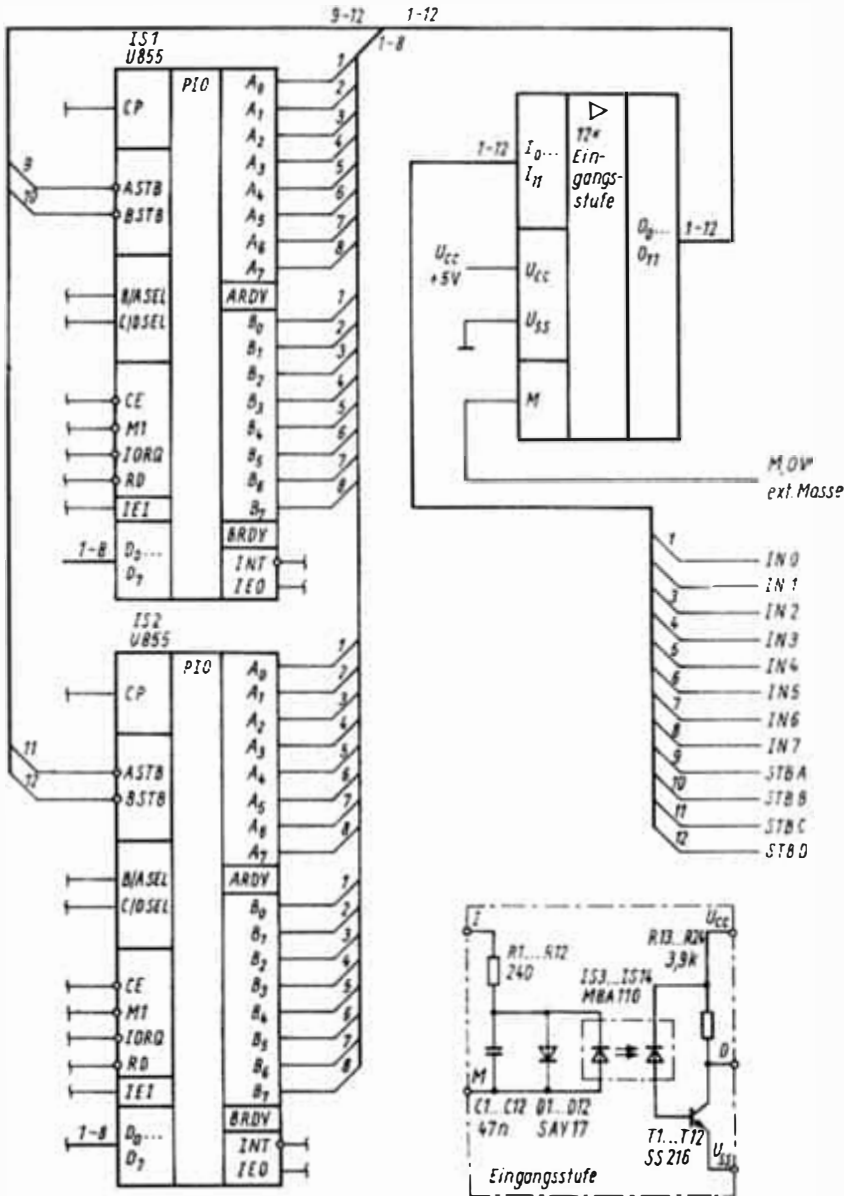


Bild 6.2.18. Parallele Eingebaugruppe mit eingangsseitiger (anwenderseitiger) Störstutzbeschaltung und Potentialtrennung vom Rechner

Nachteilig ist in der dargestellten Variante der prozeßnahen digitalen Eingabebaugruppe, daß bei geschlossenen Kontakten dauernd der Abfragestrom durch die Kontakte und Optokoppler fließt. Dadurch werden einerseits hohe Anforderungen hinsichtlich Strombelastbarkeit an die externe 24-V-Spannungsquelle gestellt; andererseits erfolgt eine hohe Umsetzung von Verlustleistung in den anwenderseitigen Funktionseinheiten der Eingabebaugruppe und damit eine Herabsetzung der Gesamtzuverlässigkeit. Als Alternativlösung bietet es sich deshalb an, die Kontaktmatrix zyklisch abzufragen und die



**Bild 6.2.19. Parallele Eingabebaugruppe zur zyklischen und prozeßnahen Abfrage von Relaiskontakten**  
 Die systemseitige Ansteuerung der beiden PIO entspricht vollständig der im Bild 6.2.15 gezeigten. Die anwenderseitige Anschaltung der Peripherie kann entsprechend der im Abschn. 7.2.4.2., S. 270, dargestellten Variante erfolgen

gewonnene Dateninformation (Kontaktbelegung) mit Hilfe der STB-Eingänge der IS U855 abzuspeichern. Hierzu muß programmtechnisch die PIO in der Eingabemodus betrieben werden (STB sind aktiv). Interruptanmeldungen können dann nur noch zyklisch von den Übernahmeimpulsen abgeleitet werden. Im Bild 6.2.19 ist die Prinzipschaltung einer solchen getaktet betriebenen Eingabebaugruppe gezeigt. Der Hardwareaufwand in der eigentlichen Mikrorechnerbaugruppe vermindert sich bei dieser Eingabevariante, da die PIO-Eingabekanäle parallel über eine 8 bit breite Peripherieschnittstelle (Optokoppler, Eingangsschutzbeschaltung, Strombegrenzung) versorgt werden können. Andererseits erhöht sich aber der Netzteilaufwand, da dort eine Baugruppe eingesetzt werden muß, die die getakteten 24-V-Versorgungsspannungen (Belastbarkeit je 100 mA) mit den dazugehörigen Übernahmeimpulsen (STB-Eingänge der PIO) bereitstellt.

Weitere anwenderspezifische Schnittstellen beziehen sich auf analoge Ein- und Ausgabefunktionen, DMA-Steuereinheiten, Tastatur- und Anzeigebeneinheiten, Bildschirmansteuereinheiten (z. B. mit Video- oder HF-Ausgang, auch für Farbmonitore) und Baugruppen zur systemnahen Mikrorechnerleistung und -inbetriebnahme. Die Realisierung derartiger Baugruppen hängt nicht zuletzt von der Verfügbarkeit leistungsfähiger hochintegrierter Bauelemente ab. Ihre Anwendung ist deshalb z. Z. bestimmten Mikrorechnersystemen und Anwendungsgebieten vorbehalten.

### 6.3. Anwendervarianten

Die Auswahlkriterien für die Konfiguration eines Mikrorechners sind i. allg. stark abhängig von dem geplanten Anwendungsgebiet. An dieser Stelle sollen deshalb einige wichtige Einsatzgebiete von Mikrorechnersystemen mit den hierbei notwendigen Eigenschaften aufgezeigt und Realisierungsvarianten mit dem Prozessorsystem U880 vorgestellt werden.

Tafel 6.3.1. Gerätekomponenten K 1520

Bezeichnung	Abkürzung	Chiffre
Zentrale Recheneinheit komplett	ZRE	K 2521
Zentrale Recheneinheit ohne Taktgenerator	ZRE	K 2522
Zentrale Recheneinheit ohne Zähler/Zeitgeber	ZRE	K 2523
Zentrale Recheneinheit ohne Taktgenerator und ohne Zähler/Zeitgeber	ZRE	K 2524
Zentrale Recheneinheit ohne Mehrrechnerkopplung mit 8-K-EPROM-Speicher	ZRE	K 2525
Operativspeicher; statisch, 4-K-RAM	OPS	K 3520
Operativ-/Festwertspeicher; statisch, 2-K-RAM; 6-K-EPROM-Speicher	OFS	K 3620
Festwertspeicher; 16-K-EPROM-Speicher	PFS	K 3820
Operativspeicher; statisch, 4-K-RAM, (CMOS)	OPS	K 3521
Operativspeicher; dynamisch, 16-K-RAM	OPS	K 3525
Operativ-/Festwertspeicher; statisch, 2-K-RAM (CMOS); 6-K-EPROM-Speicher	OFS	K 3621
Busverstärker	BVE	K 4120
Anschlußsteuerung SIF-1000	ADA	K 6022
Anschlußsteuerung für Bedieneinheit	ABD	K 7022
Bedieneinheit	BDE	K 7622
Anschlußsteuerung Folienspeicher	AFS	K 5121
Anschlußsteuerung V24	ASV	K 8021
Anschlußsteuerung BAB1 (Bildschirm)	ABS	K 7023
Anschlußsteuerung Tastatur und Drucker	ADT	K 7026
Mikrorechnerentwicklungssystem	MRES20	A 5601

Mit dem Mikrorechnersystem K 1520 (VEB Robotron-Elektronik Zella-Mehlis) ist ein Sortiment von aufeinander abgestimmten gerätetechnischen Baugruppen und Systemunterlagen für Anwender verfügbar. Das System K 1520 ist aufgrund seiner umfangreichen Ausstattung besonders für den Einsatz in der Informationsverarbeitungstechnik und in der Steuerungstechnik vorgesehen. Die Funktionseinheiten des K 1520 bestehen aus separaten Steckeinheiten und können durch einen zentralen Rechnerbus gekoppelt werden. Das verwendete Leiterkartenformat der Steckeinheiten beträgt 170 mm × 210 mm (einheitliches Gefäßsystem). Die Systemkonfiguration des K 1520 erlaubt Anwendervarianten als Einplatinenrechner, OEM-Baugruppenrechner und hierarchische Mehrrechnersysteme. In Tafel 6.3.1 sind die wichtigsten Gerätekomponenten des Mikrorechnersystems zusammengestellt. Die anwenderseitige Inbetriebnahme und Wartung des K 1520 wird durch die Bedieneinheit K 7622 unterstützt. Über verschiedene Anschlußsteuereinheiten kann eine Vielzahl von peripheren Geräten (z. B. Lochbandleser, -stanzer, Bediendrucker, Serien- und Mosaikdrucker, Kassettenmagnetbandgerät, Bildschirmterminal, Folienspeicher) angeschlossen werden. Weitere spezifische Baugruppen können durch den Anwender entwickelt und eingesetzt werden. Nach Herstellerangaben sollen des weiteren zahlreiche Systemunterlagen (Crossassembler, Simulations- und Testsystem, Ein-/Ausgabe-System, universell verwendbare Standardprogramme) verfügbar sein [5].

Tafel 6.3.2. Systemkomponenten FPS2

Bezeichnung	Chiffre
Prozessorbaugruppe mit statischem 1-K-RAM- und 2-K-EPROM-Speicher	FPS2.CPU2
Speicherbaugruppe; statisch, 1-K-RAM; 8-K-EPROM-Speicher	FPS2.RR2
Speicherbaugruppe; statisch, 2-K-RAM (CMOS)	FPS2.RAM3
Speicherbaugruppe; statisch, 8-K-RAM	FPS2.RAM4
Speicherbaugruppe; dynamisch, 32-K-RAM	FPS2.RAM5
Ausgabebaugruppe	FPS2.OI1
Eingabebaugruppe	FPS2.II1
Ansteuerbaugruppe für Eingabebaugruppe	FPS2.IMP
Serielle Standardschnittstelle (V24)	FPS2.SI1
Parallele Standardschnittstelle (SIF-1000)	FPS2.SIF1
Programmierbaugruppe für U551/U552	FPS2.PR1,2
Programmierbaugruppe für U555	FPS2.PR3
Analogausgabebaugruppe	FPS2.DA3
Analogeingabebaugruppe	FPS2.AD3
Analogeingabebaugruppe	FPS2.AD4
Rückverdrahtung	FPS2.RV2
Stromversorgungsbaugruppe	FPS2.NT5

Ein weiteres Anwendungsbeispiel des Prozessorsystems U880 stellt das numerische Steuerungssystem CNC 600 (VEB Numerik „Karl Marx“, Karl-Marx-Stadt) dar, das hauptsächlich für den komplexen Einsatz an numerisch gesteuerten Werkzeugmaschinen und Industrierobotern vorgesehen ist. Das Steuerungssystem ist aufgrund der Anforderungen als Mehrrechnersystem konzipiert. Es erfüllt Aufgaben im Bereich der Anwenderprogrammerstellung (z. B. Berechnung von geometrischen Formen), der Anlagenbedienung (z. B. Korrektur des Steuerprogramms über Ein-/Ausgabe-Geräte), der Kommunikation zur Anlage (Standardschnittstellen zu Kommunikationsgeräten, prozeßspezifische Schnittstellen), der Verbesserung der Bearbeitungsqualität an Werkzeugmaschinen (z. B. durch iterative und adaptive Korrekturen) sowie der Inbetriebnahme und Diagnose [6].

Für die Lösung von mittleren Aufgaben der Meß-, Steuer- und Regelungstechnik ist die frei programmierbare Steuerung FPS2 (VEB Funkwerk Erfurt, Applikation Bauelemente) ausgelegt (s. Abschn. 8.). Aufgrund ihres modularen Aufbaus, des gewählten Leiterkartenformats der Steckeinheiten (170 mm × 95 mm, EGS) und der prozeßnahen Realisierung von Ein-/Ausgabe-Schnittstellen (analoge Ein-/Ausgänge, digitale Ein-/Ausgänge mit Potentialtrennung, parallele und serielle Standardschnittstellen) ist die FPS2 aufwandoptimal an eine Vielzahl von typischen Einsatzanforderungen anpaßbar. In Tafel 6.3.2 sind die wichtigsten Komponenten der FPS2 zusammengestellt.

Ein wichtiges Anwendungsgebiet für den Einsatz von Mikrorechnern ist die Qualifizierung und Ausbildung ingenieurtechnischer Kader. Hierfür können abgerüstete Mikrorechner (z. B. K 1520), Mikrorechnerentwicklungssysteme (z. B. A5601, VEB Kombinat Robotron) oder sog. Kits (Mikrorechnerbausätze) bzw. Lernsysteme verwendet werden. An der oberen Einsatzgrenze dieser Anwendungsfälle ist das Lernsystem für frei programmierbare Steuerungen (VEB Funkwerk Erfurt, Applikation Bauelemente) einzuordnen (s. Abschn. 7.). Die Grundausbaustufe dieses Lernsystems dient im wesentlichen für Qualifizierungsaufgaben (Hoch- und Fachschulen, Institute u. dgl.). Jedoch kann das System aufgrund seiner Kompatibilität zur FPS2 zu einem kleineren Prototypenrechner erweitert werden. Der Einsatz vorhandener buskompatibler PROM-Programmierschübe (für U551, U552, U555) und die Verwendung eines leistungsfähigen Betriebssystems (Monitorprogramm) stellen weitere Qualitätserhöhungen dar und erlauben die Realisierung kleinerer Entwicklungsaufgaben (Hardwaretestung, Softwareentwicklung und -testung, PROM-Programmierung).

Weitere Anwendervarianten des Systems U880 sind bestimmten Aufgabenstellungen (z. B. in der Konsumgüterelektronik) angepaßte Konfigurationen (z. B. Minimalsystem; s. Abschn. 6.1.). Sie stellen aufgrund des Leistungsvermögens des Prozessors U880 kostengünstige Realisierungsvarianten dar (z. B. beim Aufbau von Schachcomputern) oder sind hinsichtlich der Verfügbarkeit Alternativlösungen zu Einchipmikrorechnern (z. B. in Haushaltsgeräten, Kfz).

# 7. Anwendung des Systems in einer frei programmierbaren Steuerung

## 7.1. Einsatzbereiche und Konfiguration

Nachdem in den vorangegangenen Abschnitten sowohl die Wirkungsweise der Schaltkreisfamilie U880 als auch ihre Einbeziehung in anwendergerechte Baugruppen dargestellt wurden, soll nachfolgend ein realisiertes Anwendungsbeispiel des Prozessorsystems vorgestellt werden.

Seitens des Anwenders besteht i. allg. unabhängig vom Anwendungsgebiet die Forderung, eine bestehende Aufgabenstellung mit einer möglichst optimal angepaßten Systemkonfiguration zu lösen. Beim Einsatz einer Mikroprozessorkonfiguration ergeben sich aber prinzipielle Vorteile (Entwicklungszeitverkürzung, Standardisierung, Vermeidung von Parallelentwicklungen), wenn vorhandene Leiterkartensysteme an die Anwenderforderungen angepaßt werden. Hierbei kann sich aber durchaus der Neuentwurf zusätzlicher Systemkomponenten, die zur aufgabengerechten Mikrorechnerkopplung dienen, durch den Anwender erforderlich machen.

Das z. Z. verfügbare, mit dem Prozessor U880 realisierte Mikrorechnersystem K 1520 (VEB Kombinat Robotron) überstreicht aufgrund seiner konzipierten Leistungsfähigkeit einen sehr breiten Anwendungsbereich. Dieser Bereich umfaßt vielfältige Einsatzgebiete vor allem in der Rechentechnik und Datenverarbeitung sowie der Informationstechnik und Automatisierungstechnik. Wegen der großen Universalität und Einsatzbreite dieses Mikrorechnersystems sind aber die Anwender auf die weitgehend standardisierten Ein-/Ausgabe-Schnittstellen festgelegt. Prozeßnahe Schnittstellen, wie sie vor allem in der Steuerungs- und Regelungstechnik auftreten, erfordern zusätzlichen Adaptieraufwand vom Anwender.

Das nachfolgend beschriebene Leiterkartensystem FPS2 (VEB Funkwerk Erfurt, Applikation Bauelemente) ist im Gegensatz zum K 1520 als frei programmierbare Steuerung konfiguriert. Seine Anwendungsgebiete befinden sich vor allem in der Automatisierungstechnik, und es realisiert die typischen mittleren Anforderungen dieser Bereiche. Mit dieser Eingrenzung der Einsatzgebiete ist konzeptionsmäßig eine günstige Anpassung an die typischen Anforderungen dieser Anwender verbunden. Das wird besonders durch das Vorhandensein prozeßnaher analoger und digitaler Schnittstellen neben den üblichen Standardschnittstellen ausgedrückt. Damit ergibt sich anwenderseitig eine aufwandsoptimale Anpassung des Leiterkartensystems FPS2 an die verschiedensten Aufgabenstellungen dieses Industriezweigs. Weitere Kriterien der problembezogenen Konfiguration werden durch die Wahl des Leiterkartenformats, die Aufteilung und Konzeption aufgabengerechter und effektiver Speicherbaugruppen, das Vorhandensein von Hilfskomponenten und die Leistungsfähigkeit der minimalen Ausbaustufe aufgestellt. Bei der FPS2 wurde für die Leiterkarten das Einfachkassettenformat des einheitlichen Gefäßsystems (EGS) zugrunde gelegt. Dieses Leiterkartenformat (95 mm × 170 mm) gestattet insbesondere eine kostengünstige Systemkonfigurierung, da die Baugruppen für die vorgesehenen Anwendungsfälle typische Leistungsanforderungen erfüllt. Alle informationsverarbeitenden Komponenten des Leiterkartensystems haben einen 58poligen direkten

Steckverbinder, der zur Übertragung der Systeminformationen (Adressen, Daten, Steuerungssignale) sowie der Versorgungsspannungen (+5, +12, -12, -5 V, GND) dient. Er wird nachfolgend als *Systemsteckverbinder* bezeichnet. Die peripheren Baugruppen des Steuerungssystems FPS2 weisen zusätzlich eine 58polige Buchsenleiste an der Rückseite der Steckeinheiten auf, die zur Übermittlung der anwenderbezogenen Informationen (analoge und digitale Ein-/Ausgabe-Funktionen, Zähler/Zeitgeber-Kanäle u. dgl.) benutzt werden. Dieser Steckverbinder wird *Rückseitensteckverbinder* genannt. Die Belegung des Systemsteckverbinders ist anschlusmäßig einheitlich ausgeführt, so daß eine gedruckte Rückverdrahtung (FPS2.RV2) eingesetzt werden kann. Weitere Hilfskomponenten des Leiterkartensystems sind Netzteilbaugruppen. Zu Test- und Inbetriebnahmezwecken kann das im Abschn. 8. vorgestellte Lernsystem eingesetzt werden. Die minimale Ausbaustufe der FPS2 ist durch den Einsatz der Prozessorbaugruppe, die einen relativ umfangreichen Festwert- und Schreib/Lese-Speicherbereich aufweist, in Verbindung mit einer Peripheriebaugruppe gegeben. Mit ihr können bereits relativ anspruchsvolle Aufgabenstellungen realisiert werden.

Die nachfolgende Baugruppenbeschreibung der FPS2 soll unterstreichen, daß für geräteproduzierende und rationalisierende Anwender die Notwendigkeit besteht, sich neben den Kenntnissen der Prozesseigenschaften (z. B. Interruptverhalten) und der Programmiervarianten des U880 (Befehlssatz, Standardsoftware) zusätzlich die Baugruppenkonfiguration des vorgesehenen Leiterkartensystems anzueignen.

## 7.2. Baugruppenbeschreibung

### 7.2.1. Prozessorbaugruppe

Die Prozessorbaugruppe (FPS2.CPU2) stellt das Herzstück des Steuerungssystems FPS2 dar. Sie bestimmt wesentlich die Leistungsparameter der Steuerung, da einerseits hier die zeitliche Ansteuerung des gesamten Systems erfolgt und andererseits die Struktur dieser Baugruppe die Konfigurierbarkeit der Steuerung festlegt. Die Baugruppe FPS2.CPU2 ist konstruktiv als Steckeinheit mit den im Abschn. 7.1. genannten Eigenschaften ausgeführt. Neben den eigentlichen Systemsteuerelementen (CPU, Bustreiber u. dgl.) enthält sie zusätzlich Funktionseinheiten der Speicherbaugruppe. Hierdurch sind für den Anwender bei Verwendung dieser Steckeinheit bereits ein Festwertspeicherbereich von 2 Kbyte (bzw. 4 Kbyte) und ein Schreib/Lese-Speicher mit einer Kapazität von 1 Kbyte verfügbar. Für spezielle Anwendungsfälle stellt diese Steckeinheit deshalb eine Minimalkonfiguration der FPS2 dar, die (in Verbindung mit Zusatzlogik) bereits relativ hohen Anforderungen genügt.

Im Bild 7.2.1 ist das Blockschaltbild der Prozessorbaugruppe dargestellt. Es enthält die wesentlichen Funktionseinheiten der Baugruppe. Im Mittelpunkt steht hierbei die zentrale Verarbeitungseinheit – die CPU U880. Zur taktmäßigen Ansteuerung des Prozessors dient die Funktionseinheit Taktversorgung. Sie besitzt einen Quarzgenerator, von dem die Systemtaktfrequenz  $f_c = 2,4576 \text{ MHz}$  abgeleitet wird. Ein MOS-Takttreiber übernimmt die Ansteuerung der CPU. Außerdem wird das verstärkte und invertierte Systemtaktsignal ( $\overline{CP}$ ) zur Ansteuerung weiterer Baugruppen auf den Systemsteckverbinder geführt. Die Taktversorgungseinheit hat zusätzlich die Möglichkeit, daß die Taktansteuerung über einen wahlweise einzuspeisenden externen Takt erfolgt. Hierzu sind entsprechend vorhandene Drahtbrücken umzulegen. Die Anwendung dieser Möglichkeit ist vor allem dann denkbar, wenn die Inbetriebnahme oder Prüfung der Prozessorbaugruppe

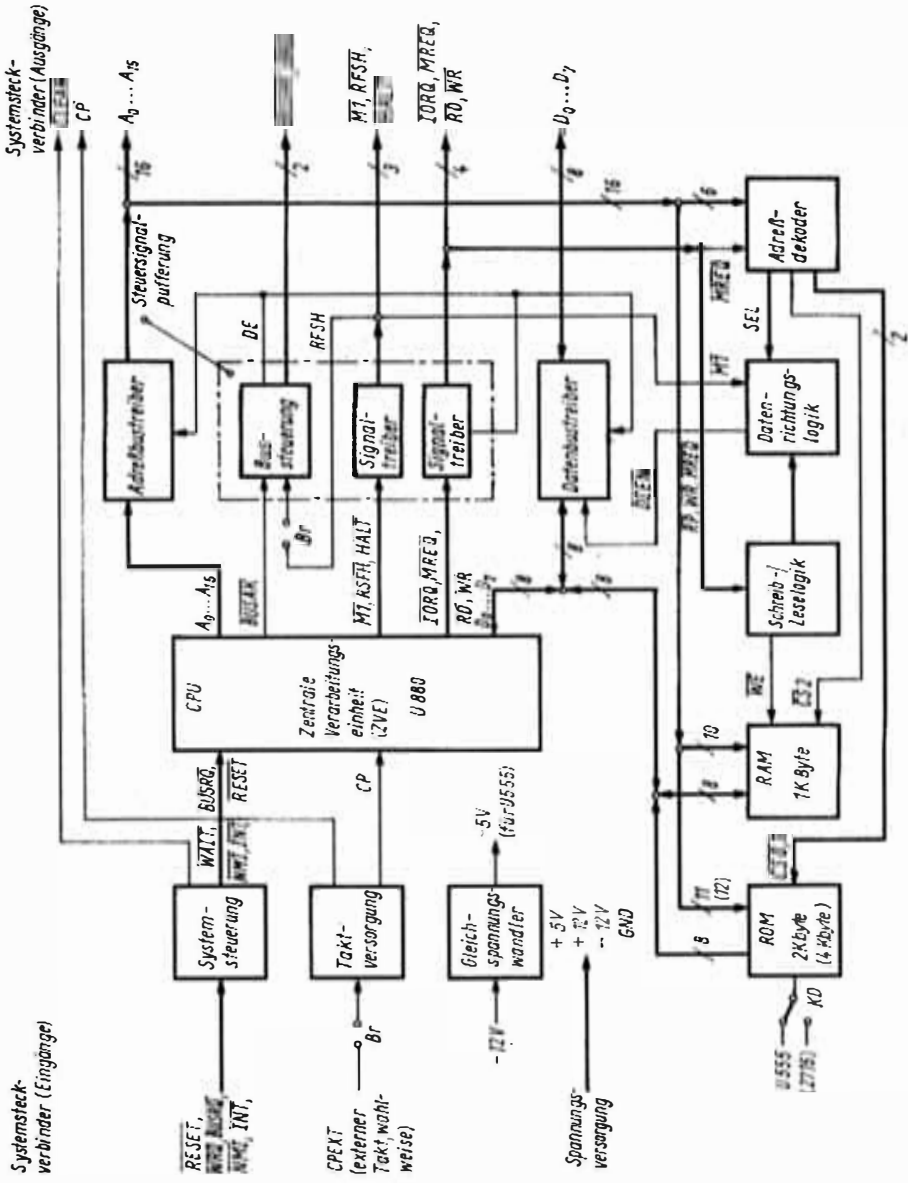


Bild 7.2.1. Blockschaltbild der Prozessorgruppe FPS2.CPU2

mit einem Emulatorrechner (ICE, in-circuit-emulator) vorgenommen wird, der eine taktmäßige Synchronisation erfordert.

Die Funktionseinheit Systemsteuerung dient zur signalmäßigen Ansteuerung des Prozessors. Sie nimmt die Verteilung der Steuersignale  $\overline{WRQ}$ ,  $\overline{BUSRQ}$ ,  $\overline{INT}$ ,  $\overline{NMI}$  und  $\overline{RESET}$  vor und erzeugt beim Zuschalten der 5-V-Stromversorgung einen Power-on-reset-Impuls. Das CPU- $\overline{RESET}$  wird zusätzlich verstärkt und als  $\overline{CLEAR}$ -Signal zu den anderen Baugruppen weitergeleitet. Es kann dort zur Rücksetzung der Peripherie benutzt werden.

Die Funktionseinheiten Adreßbustreiber und Datenbustreiber nehmen die Treibung der zugehörigen Buslinien vor. Der Datenbustreiber wirkt hierbei bidirektional. Die Datenrichtungsumschaltung muß in Abhängigkeit von den CPU-Steuersignalen  $\overline{RD}$  und  $\overline{WR}$  sowie abhängig von der Ansteuerung der zur Steckeinheit gehörenden Speichereinheit erfolgen. Das entsprechende Umschaltsignal  $\overline{DIEN}$  wird in der Funktionseinheit Datenrichtungslogik erzeugt.

Die ausgangsseitige Pufferung des Steuerbusses erfolgt in der Funktionseinheit Steuer-signalpufferung. Diese Einheit enthält neben den eigentlichen Signaltreibern noch eine Bussteuereinheit, in der das Ausgangsfreigabesignal OE (output enable) und die Busquittierungssignale  $\overline{BUSAK}$  und  $\overline{BAO}$  erzeugt werden. Mit dem Freigabesignal OE können die Adreßbustreiber, die Datenbustreiber und die Signaltreiber für die Steuersignale  $\overline{MREQ}$ ,  $\overline{IORQ}$ ,  $\overline{RD}$  und  $\overline{WR}$  im Fall eines DMA-Betriebs in den floatenden Zustand gebracht werden. Eine entsprechende DMA-Baugruppe (in der Peripherie) kann dann diese Buslinien übernehmen. Eine derartige Busquittierung erfolgt bei aktivem  $\overline{BUSAK}$ -Signal der CPU. Wahlweise kann über eine vorgesehene Drahtbrücke aber auch zusätzlich bei jedem aktiven Zustand des  $\overline{RFSH}$ -Steuersignals des Prozessors U880 eine Busquittierung eingeschoben werden. Der peripheren DMA-Baugruppe wird somit die DMA-Betriebsart „refresh cycle stealing“ ermöglicht. Voraussetzung für die Busquittierung mit aktivem  $\overline{RFSH}$  ist jedoch immer, daß keine dynamischen Speicherbaugruppen im Steuerungssystem eingesetzt werden (z. B. FPS2.RAM5).

Die weiteren Funktionseinheiten der Prozessorbaugruppe FPS2.CPU2 bilden eine Speichereinheit. Der Festwertspeicher weist hierbei eine Speicherkapazität von 2 Kbyte auf, wenn Speicherbauelemente vom Typ U555 eingesetzt werden. Über Kodierstecker kann der Einsatz von Speicherbauelementen des Typs 2716 vorbereitet werden. Die ROM-Kapazität erhöht sich dann auf 4 Kbyte. Die Speicherkapazität der Schreib/Lese-Speichereinheit beträgt 1 Kbyte. Die Ansteuerung der beiden Speichereinheiten wird von den Funktionseinheiten Adreßdekoder und Schreib/Lese-Logik vorgenommen. In diesen Einheiten werden gleichzeitig Informationen abgeleitet, die die Datenrichtungsschaltung der bidirektionalen Datenbustreibereinheit beeinflussen. Es muß hierbei u. a. gewährleistet werden, daß bei Zugriff auf den baugruppeninternen Speicherbereich die peripheren Baugruppen den inneren Datenbus abhören können, weil evtl. RETI-Befehle dekodiert werden müssen.

Damit bei Nichtbenutzung der baugruppeninternen Speichereinheiten keine Buskonflikte auftreten, können die zugehörigen Adreßdekoderschaltkreise nicht bestückt werden. Die Datenrichtungslogik wertet in diesem Fall nur noch die CPU-Steuersignale  $\overline{RD}$  und  $\overline{WR}$  aus.

## 7.2.2. Speicherbaugruppen

### 7.2.2.1. Kombinierte Speicherbaugruppe

Die Speicherbaugruppe FPS2.RR2 hat sowohl eine Festwertspeichereinheit als auch einen Schreib/Lese-Speicherbereich. Sie ist als Steckeinheit mit den EGS-Maßen 170 mm × 95 mm ausgelegt und entspricht den im Abschn. 7.1. dargestellten Systemanforderungen. Bei vollständiger Bestückung beinhaltet die Steckeinheit eine ROM-Kapazität von 8 Kbyte und eine RAM-Kapazität von 1 Kbyte. Die Baugruppe ist so ausgelegt, daß bei Nichtbedarf der RAM-Einheit die gesamte RAM-Bestückung einschließlich RAM-Dekoder und -Ansteuerung entfallen können.

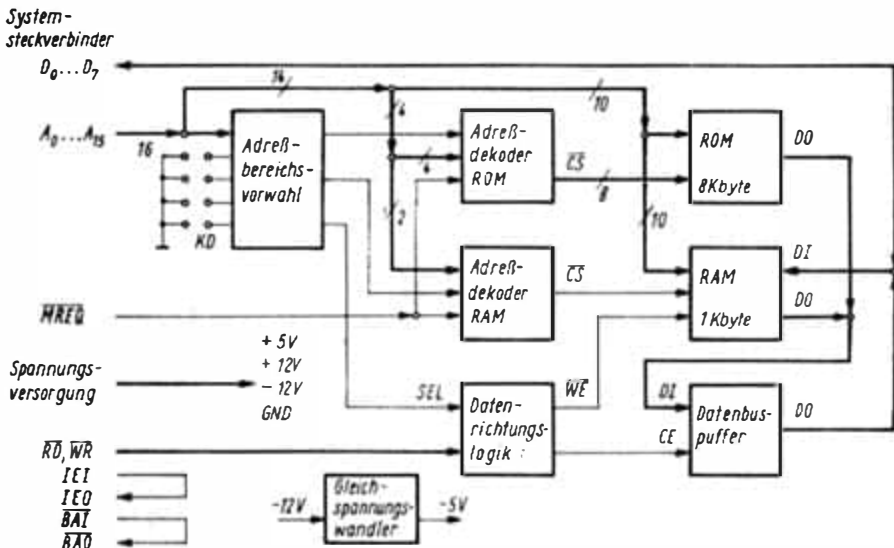


Bild 7.2.2. Blockschaltbild der kombinierten Speicherbaugruppe FPS2.RR2

Im Bild 7.2.2 ist das Blockschaltbild der kombinierten RAM/ROM-Speicherbaugruppe dargestellt. Es enthält die wesentlichen Funktionseinheiten der Baugruppe. Die Einheit Adreßbereichsvorwahl nimmt im Zusammenhang mit den beiden Adreßdekodern der Speicherblöcke eine programmierbare Vorwahl (mittels Kodierstecker) der Speicherbereiche vor. Für die Festwertspeichereinheit ist hierbei der Adressierungsraum 0 bis 32 Kbyte (0 ... 7FFFH) und für die Schreib/Lese-Speichereinheit des 61. bis 64. Kbyte (F000H ... FFFFH) vorgesehen. Im Leiterkartensystem sind deshalb maximal vier derartige Steckeinheiten einsetzbar.

Die RAM-Speichereinheit wird durch acht U202-Bauelemente realisiert. In der ROM-Einheit sind die EPROM-Bauelemente U555 oder die maskenprogrammierten ROM-Bauelemente U505 (jeweils max. acht Stück) einsetzbar. Für die Spannungsversorgung der EPROM U555 werden in der Baugruppe neben der TTL-Versorgungsspannung noch  $U_{DD} = 12\text{ V}$  und  $U_{BB} = -5\text{ V}$  bereitgestellt.

Zur Treibung des Systemdatenbusses dient die Funktionseinheit Datenbuspuffer. Diese Einheit wirkt nicht bidirektional, da die „mal 1“-organisierten RAM-Elemente die Datenlinien kapazitiv relativ gering belasten. Neben der Bustreiberfunktion hat diese Funktionseinheit gleichzeitig die Aufgabe, die Ein- und Ausgangslinien der RAM zu entkoppeln. Die Datenrichtungslogik liefert das zur Ansteuerung der Bustreiber notwendige Richtungssignal.

Signal- und ansteuermäßig ist die Baugruppe FPS2.RR2 den Systemforderungen des Prozessors U880 und der CPU-Baugruppe FPS2.CPU2 angepaßt. Es werden keine Synchronisationssignale (WAIT-Forderungen) benötigt.

### 7.2.2.2. CMOS-Speicherbaugruppe

Die Speicherbaugruppe FPS2.RAM3 beinhaltet CMOS-Schreib/Lese-Speicherelemente, deren Spannungsversorgung aus Ni-Cd-Akkus bei Versorgungsspannungsausfall (Netzausfall) aufrechterhalten wird. Sie ist ebenfalls als dem FPS2-System zugehörige Steckeinheit realisiert. Bei vollständiger Bestückung hat die Baugruppe eine RAM-Kapazität von 2 Kbyte.

Bild 7.2.3 zeigt das Blockschaltbild der CMOS-Speicherbaugruppe. In ihm sind die Funktionseinheiten der Steckeinheit prinzipiell dargestellt.

Die Funktionseinheit Adreßbereichsvorwahl erlaubt die Lokalisation des 2-Kbyte-Speicherbereichs durch fünf Kodierstecker im gesamten Adressierungsraum der CPU. Die Steckeinheit ist deshalb u. a. auch für Programmentwicklungszwecke als Programmspeicher verwendbar. Hierbei ergibt sich die Möglichkeit, einen Speicherschreibschutz mittels Kodierstecker vorzusehen.

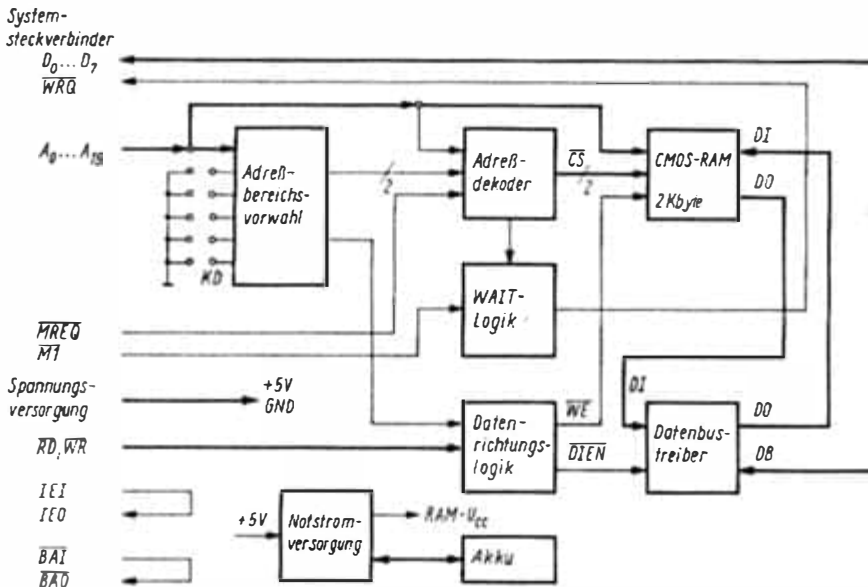


Bild 7.2.3. Blockschaltbild der statischen CMOS-RAM-Baugruppe FPS2.RAM 3

Die eigentliche CMOS-RAM-Speichereinheit ist mit 16 RAM-Bauelementen bestückt. Diese Bauelemente entsprechen in ihrer Organisation prinzipiell dem U202. Die RAM-Versorgungsspannung wird bei Netzausfall über die Funktionseinheit Notstromversorgung aus gasdichten Ni-Cd-Akkus aufrechterhalten. Diese Akkus haben eine Kapazität von 200 mAh und gewährleisten im aufgeladenen Zustand eine Speicherzeit der Steckeinheit von etwa 1000 Betriebsstunden. Sie werden bei vorhandener Versorgungsspannung ständig nachgeladen bzw. gepuffert.

Die Ansteuerung der Speichereinheit erfolgt durch die Funktionseinheiten Adreß-dekoder und Datenrichtungslogik. Über die letztgenannte Einheit wird gleichzeitig die

Richtungssteuerung der Datenbustreiber vorgenommen. Die Datenbustreibereinheit ist bei der Baugruppe FPS2.RAM3 bidirektional ausgelegt.

Da die eingesetzten RAM-Bauelemente gegenüber dem U202 eine etwas größere Zugriffszeit aufweisen ( $t_{ACC} \leq 460$  ns), werden bei M1-Speicherlesezugriffen auf die CMOS-RAM-Baugruppe automatisch je ein WAIT-Zustand eingefügt. Er dient zur Synchronisation des Zeitverhaltens und wird von der Funktionseinheit WAIT-Logik erzeugt.

Bei Einsatz der Speicherbaugruppe als Programmspeicher ist das durch Einfügen dieser WAIT-Zustände veränderte Echtzeitverhalten zu beachten.

### 7.2.2.3. Statische RAM-Speicherbaugruppe

Die Baugruppe FPS2.RAM4 stellt eine RAM-Speichereinheit dar, die mit statisch wirkenden Bauelementen bestückt ist, Sie zeichnet sich durch eine große Speicherkapazität von 8 Kbyte und einen niedrigen Leistungsverbrauch aus. Die Speicherbaugruppe ist deshalb als universelle RAM-Baugruppe im Leiterkartensystem FPS2 einsetzbar. Konstruktiv und konzeptionell ist sie als FPS2-kompatible Steckereinheit (170 mm × 95 mm) ausgelegt.

Im Bild 7.2.4 ist die Zusammenschaltung der wesentlichen Funktionseinheiten der Baugruppe im Blockschaltbild dargestellt.

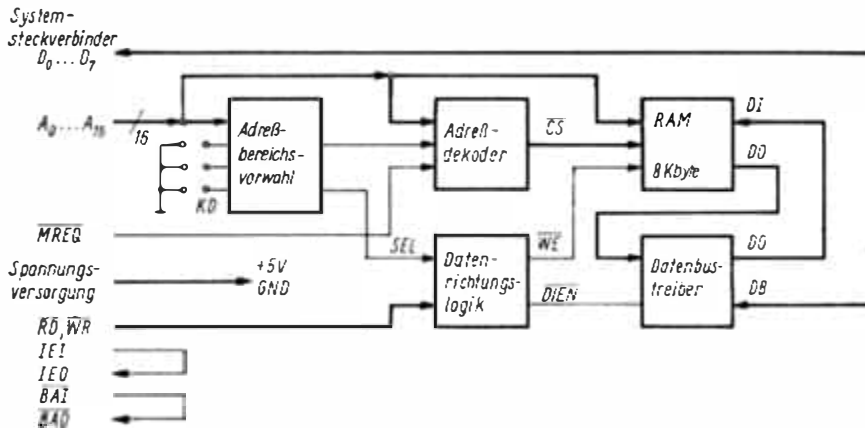


Bild 7.2.4. Blockschaltbild der statischen Operativspeicherbaugruppe FPS2.RAM 4

Die Funktionseinheit Adreßbereichsvorwahl nimmt durch Kodierstecker die Lokalisation des 8-Kbyte-RAM-Speichers im Adressierungsraum des Prozessors U880 vor. Es können entsprechend acht Bereiche vorprogrammiert werden. Der Adreßdekodier nimmt im Zusammenhang mit der Datenrichtungslogik die eigentliche Ansteuerung des RAM-Speicherblocks vor. Die Funktionseinheit Datenbustreiber ist wiederum bidirektional ausgelegt. Die Richtungsumschaltung wird abhängig von den CPU-Steuersignalen und der Kartenselektion von der Datenrichtungslogik vorgenommen.

Die geringe Zugriffszeit der eingesetzten RAM-Bauelemente gestattet die direkte zeitliche Ansteuerung der Steckereinheit durch den Prozessor U880 und die CPU-Steckereinheit. Es sind keine Synchronisationsmaßnahmen notwendig.

### 7.2.2.4. Dynamische RAM-Speicherbaugruppe

Die dynamische RAM-Baugruppe FPS2.RAM5 hat bei Vollbestückung eine Speicherkapazität von 32 Kbyte. In einer Steuerung sind deshalb max. zwei derartige Baugruppen einsetzbar. Die dynamische RAM-Speicherbaugruppe ist als FPS2-kompatible Steckerein-

heit ausgeführt. Sie ist mit 16 RAM-Bauelementen des Typs U256 bestückt, die in zwei Speicherblöcken zu je 16 Kbyte angeordnet sind.

Im Bild 7.2.5 ist das Blockschaltbild der Steckeinheit dargestellt. In ihm sind die wichtigsten Funktionseinheiten in ihrem prinzipiellen Zusammenspiel gezeigt. Mit der Einheit Adreßbereichsvorwahl kann, vorprogrammiert durch Kodierstecker, jeder der beiden 16-Kbyte-Speicherblöcke beliebig im CPU-Adressierungsraum lokalisiert werden.

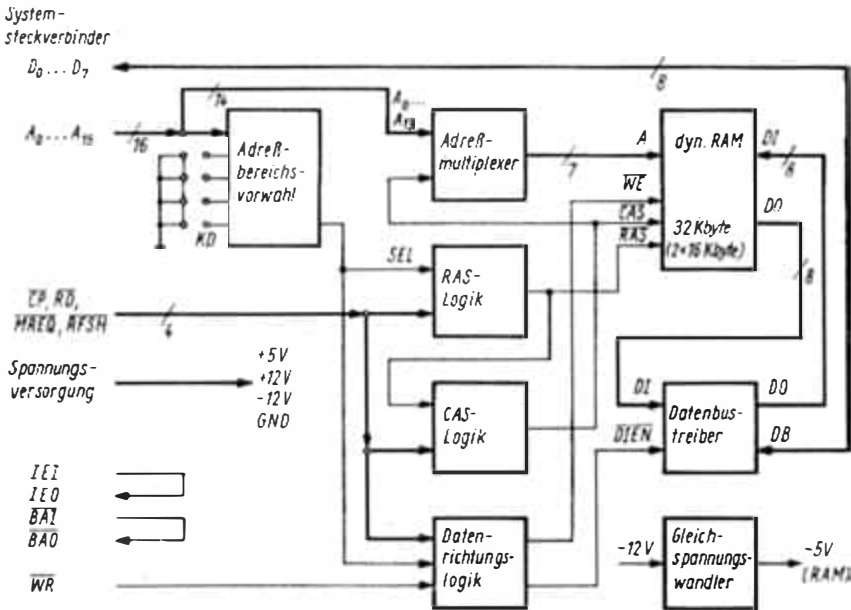


Bild 7.2.5. Blockschaltbild der dynamischen Operativspeicherbaugruppe FPS2.RAM 5

Die eigentliche Ansteuerung der dynamischen Speichereinheiten ( $2 \times 16$  Kbyte) geschieht über die Funktionseinheiten Adreßmultiplexer, RAS-Logik, CAS-Logik und die Datenrichtungslogik. Im Adreßmultiplexer wird der zur adreßmäßigen Versorgung der 16-Kbyte-Blöcke notwendige 14-bit-Adreßbus (CPU-Adressen  $A_0 \dots A_{13}$ ) auf die sieben Adreßlinien der RAM-Bauelemente umgeschaltet. Diese Umschaltung ist abhängig vom CAS-Signal. Die RAS-Logik erzeugt taktsynchron bei Speicherzugriffen und in den Refreshzyklen das Zeilenansteuersignal  $\overline{RAS}$  (row address strobe) für die dynamischen RAM. Im RAM erfolgt hierauf das Einlesen des niederwertigeren Adreßteils, die Auswahl der RAM-Zeile und das Auslesen dieser Zeile in ein 128-bit-Zwischenregister. Alle auch nachfolgenden RAM-Operationen werden durch einen RAM-internen Taktgenerator zeitlich gesteuert. Die CAS-Logik leitet bei Speicherzugriffen bzw. Schreiboperationen das  $\overline{CAS}$ -Signal (column address strobe) ab, mit dem der höherwertige Teil des Adreßbusses übernommen wird. Im RAM erfolgt hiermit die Ansteuerung eines 1-aus-128-Dekoders, der die adressierte Speicherzelle in dem 128-bit-Zwischenregister anspricht. Nach vollzogener Lese- oder Schreiboperation wird das Zwischenregister automatisch in die zuvor ausgewählte RAM-Zeile rückgespeichert (Refreshvorgang; wird auch bei aktivem CPU-Steuersignal  $\overline{RFSH}$  durch die RAS-Logik der Steckeinheit ausgeführt).

Der Datenbus-treiber ist bei der Baugruppe FPS2.RAM5 bidirektional ausgelegt. Er wird richtungsmäßig ebenfalls von der Funktionseinheit Datenrichtungslogik angesteuert.

Beim Einsatz der dynamischen RAM-Speicherbaugruppe sind aufgrund der geringen

RAM-Zugriffszeiten keine Synchronisationsmaßnahmen erforderlich. Bei entsprechend selektierten RAM ( $t_{ACC} \leq 200$  ns) ist die Anwendung sogar in Systemen mit einer Taktfrequenz von 4 MHz denkbar. Im Zusammenhang mit dem Einsatz der Steckeinheit in U880-Systemen ist zu beachten, daß während der Refreshzyklen der Adreßbus aktiv bleibt und nicht für DMA-Zwecke (refresh cycle stealing) verwendet wird.

## 7.2.3. Standard-Ein-/Ausgabe-Baugruppen

### 7.2.3.1. Parallele Standardschnittstelle

Die Baugruppe FPS2.SIF1 realisiert zwei SIF-1000-Anschlußsteuerungen nach TGL 26 456. Eine Anschlußsteuerung ist als Ausgabekanal (Kanaltyp A) ausgelegt, die andere ist ein Eingabekanal (Kanaltyp E). Zur Aufbereitung der Daten- und Steuerungssignale dienen zwei parallele Ein-/Ausgabe-Einheiten U855. Über sie werden die Übertragungssteuer-signale (RUF, END), die Kommandosignale (KOM-1 ... KOM-3), Datensignale (DAT-1 bis DAT-8) und die Statussignale (STA-1 ... STA-3) beider Anschlußsteuerungen übertragen. Sämtliche Signale werden lastmäßig durch Treiberstufen bzw. Eingangsbeschaltungen von den PIO getrennt. Durch entsprechende Bestückungsvarianten können die Ausgänge dieser Treiberstufen wahlweise für KME 3- oder KME 10-Pegel ausgelegt werden. Die Eingänge sind nur mit KME 10-Pegel (TTL) ansteuerbar. Die SIF-1000-Baugruppe ist auf einer FPS2-zugehörigen Steckeinheit mit den Abmessungen 180 mm  $\times$  95 mm untergebracht. Sie hat entsprechend den im Abschn. 7.1. getroffenen Systemvereinbarungen einen System- und einen Rückseitensteckverbinder.

Tafel 7.2.1. Abläufe bei einer SIF-1000-Eingabesteuerung

Signal	SZ0	SZ1	SZ2	SZ3	SZ0
RUF-E					
END-E					
KOM-E					
DAT-E					
STA-E					
Aktivität					
Anschlußsteuerung	Anforderung einer Eingabe, Bereitstellung des Steuerbefehls KOM		Übernahme von Daten und Status	Abschalten des Steuerbefehls	
Peripheriegerät		Befehlsausführung, Datenausgabe	Bereitmeldung durch END	Endemeldung für Eingabezyklus	

SZ Steuerzustand

Jede der beiden SIF-1000-Anschlußsteuerungen der Baugruppe wird durch die Portlinien einer PIO U855 realisiert. Da hierbei gemischte Ein- und Ausgabelinien benötigt werden, müssen diese PIO in der Bitmode betrieben werden. Dadurch stehen dem Anwender die zusätzlichen Interruptfunktionen der IS U855 in dieser Betriebsart zur Ver-

fügung. Die Zuordnung der logischen Pegel zu den jeweiligen RUF-, END-, DAT-, KOM- und STA-Signalen ist gerätespezifisch. Ebenso sind die konkreten Befehls- und Statuskodes (Belegung von KOM und STA) und die zeitlichen Ansteuerbedingungen vom Peripheriegerät (z. B. Lochbandleser) abhängig. Die Anschlußsteuerung muß dementsprechend durch ein gerätespezifisches Softwarepaket an das Peripheriegerät angepaßt werden.

Tafel 7.2.2. Abläufe bei einer SIF-1000-Ausgabesteuerung

Signal	SZ0	SZ1	SZ2	SZ3	SZ0
RUF-A					
END-A					
KOM-A					
DAT-A					
STA-A					
Aktivität					
Anschlußsteuerung	Ankündigung einer Ausgabe, Bereitstellung des Steuerbefehls KOM		Übernahme des Status	Statusquittierung, Abschaltung des Befehlskodes	
Peripheriegerät		Befehlsausführung, Datenübernahme, Statusausgabe	Fertigmeldung durch END	Endemeldung für Ausgabezyklus, Abschalten von STA	

Der prinzipielle zeitliche Ablauf der Steuerzustände einer SIF-1000-Eingabesteuerung ist in Tafel 7.2.1 dargestellt. Gleichzeitig werden die Aktivitäten der Anschlußsteuerung und des Peripheriegeräts erläutert. Die Tafel 7.2.2 zeigt die entsprechenden Vorgänge am Kanaltyp A (Ausgabesteuerung).

Im Bild 7.2.6 ist das Blockschaltbild der SIF-1000-Baugruppe dargestellt. Es zeigt das Zusammenwirken der wichtigsten Funktionseinheiten. Die Adreßbereichsvorwahl dient im Zusammenhang mit den Adreßdekodern dazu, die beiden parallelen Ein-/Ausgabe-Einheiten im 256-bit-Portadressierungsraum des Systems zu lokalisieren. Die konkreten Daten- und Steueradressen der PIO können hierbei mit Kodiersteckern vorprogrammiert werden.

Die Funktionseinheit Taktreiber enthält einen Inverter, dem eine aktive Taktformerschaltung (aktives pull-up) nachgeschaltet ist. Sie nimmt die Taktsteuerung der PIO vor.

Die Interruptlogik nimmt die interruptmäßige Einbeziehung der Steckeinheit ins Steuerungssystem vor. Hierzu gehören einerseits die kartenbezogene Signalerzeugung für das Interruptanforderungssignal  $\overline{INT}$ , abgeleitet von den Forderungen der PIO, und die Einfügung der Ein-/Ausgabe-Einheiten in die Interruptprioritätskette einschließlich der entsprechenden Umgehungslogik; andererseits erfolgt eine Einflußnahme auf die Datenrichtungssteuerung (Funktionseinheit Datenrichtungslogik) dahingehend, daß bei Interruptquittierung eines der beiden auf der Steckeinheit befindlichen PIO eine Richtungs-schaltung in Richtung Prozessorbaugruppe vorgenommen wird. Damit können vektorierte Interrupts durch die Baugruppe FPS2.SIF1 bearbeitet werden.

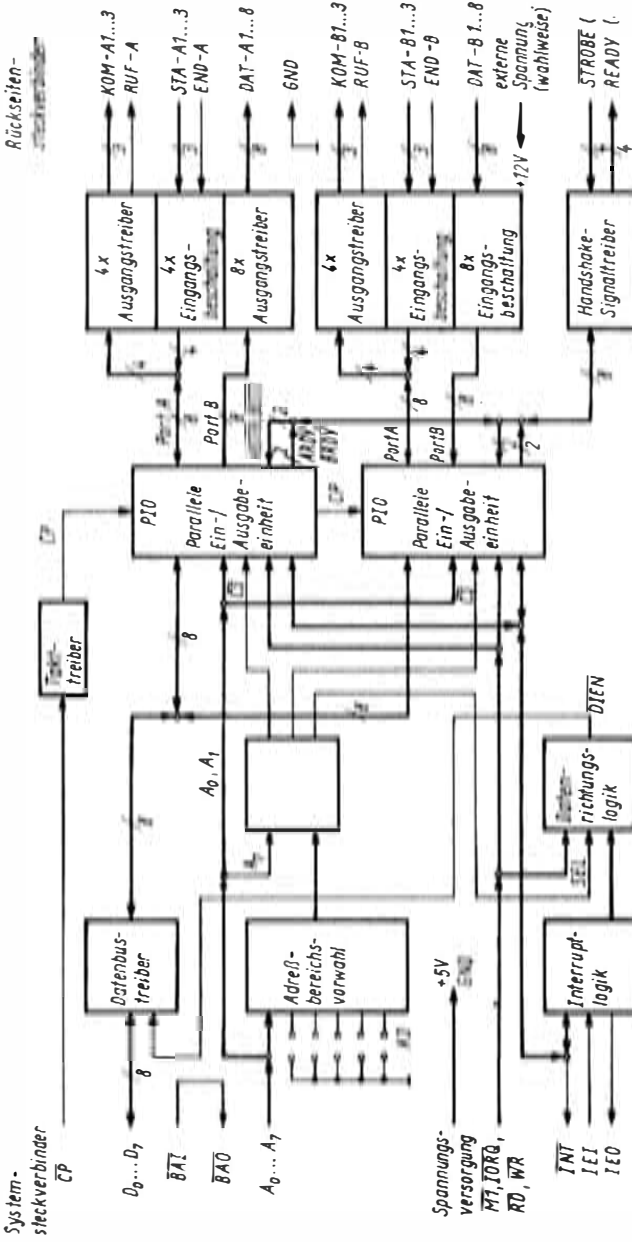


Bild 7.2.6. Blockschaltbild des SIF-1000-Standardinterface FPS2-SIF 1

Die Datenrichtungslogik muß zum einen bei einer entsprechenden Signalkonfiguration der CPU-Steuersignale  $\overline{RD}$ ,  $\overline{WR}$  und des Kartenauswahlsignals (SEL) die Richtung des Datenbustreibers auf „Schreiben“ schalten. Zum anderen muß aber immer gewährleistet bleiben, daß die PIO den Datenbus bei Befehlsholezyklen ( $\overline{MI}$ ) empfangen können. Die dort evtl. auftretenden RETI-Befehlskodes müssen von den peripheren Schaltkreisen dekodiert werden.

Die Datenbustreibereinheit der Baugruppe wirkt bidirektional und ist mit Low-power-Schottky-Elementen bestückt.

Die Baugruppe FPS2.SIF1 kann neben der Realisierung von SIF-1000-Anschlußsteuerungen auch für die Gestaltung anderer, nichtstandardisierter Parallelschnittstellen verwendet werden. Für diese Anwendungen sind zusätzlich die Handshakesignallinien der PIO ( $\overline{STROBE}$  und  $\overline{READY}$ ) nach außen geführt.

### 7.2.3.2. Serielle Standardschnittstelle

Die Baugruppe FPS2.SII stellt ein serielles Interface dar. Sie beinhaltet zwei Kanäle zur seriellen Dateneingabe, zwei serielle Datenausgabeports und zwei frei verfügbare CTC-Kanäle. Diese Kanäle werden durch eine serielle Datenein-/ausgabe-Einheit (SIO) U856 und eine Zähler/Zeitgeber-Einheit (CTC) U857 realisiert.

Das serielle Interface entspricht konstruktiv und signalmäßig der Systemforderung der FPS2. Es ist auf einer Steckeinheit mit den Maßen 180 mm × 95 mm untergebracht und hat einen 58poligen System- und Rückseitensteckverbinder.

Die serielle Datenein- und -ausgabe geschieht im Asynchronbetrieb. Es sind Übertragungsraten von 50 bit/s bis 9,6 kbit/s programmierbar. Die hierzu notwendige Takterzeugung für die Datenkanäle erfolgt über zwei Zähler/Zeitgeber-Kanäle. Die seriellen Datenlinien und die beiden Zählgänge der Baugruppe sind galvanisch von der Rechnermasse getrennt. Sie können jeweils über eine 20-mA-Stromschnittstelle oder eine V24-Schnittstelle (RS 232) betrieben werden. Beim Betrieb über die Stromschnittstellen muß extern eine 20-mA-Stromquelle angeordnet werden, die den Bedingungen der eingesetzten Schalterelemente entspricht. Die Verwendung der RS 232-Schnittstellen wird durch die Erzeugung von einer +12-V- und -12-V-Spannungsversorgung in der Baugruppe mit Hilfe von Gleichspannungswandlern erleichtert. Die Modemsteuersignale der SIO sind ungepuffert auf den Rückseitensteckverbinder geführt.

Im Bild 7.2.7 ist das Blockschaltbild der seriellen Ein-/Ausgabe-Baugruppe dargestellt. Es zeigt das Zusammenwirken der wesentlichen Funktionseinheiten.

Durch die Funktionseinheit Adreßbereichsvorwahl kann über fünf Kodiersteckerstellungen eine Kartenadresse im Adressierungsraum des Prozessors gebildet werden. Der eigentliche Adreßdekoder weist damit den beiden peripheren Einheiten die Adressierung (Kanaladressierung, Umschaltung Datenbetrieb/Steuerbetrieb) zu.

Der Taktreiber erzeugt die zur Ansteuerung der Peripherielemente notwendigen MOS-Pegel des Systemtakts. Er wirkt invertierend und ist mit einem aktiven pull-up ausgestattet.

Der bidirektionale Datenbustreiber übernimmt die Treibung und Verteilung der Systemdatenbusinformation. Seine Richtungsumschaltung wird von der Funktionseinheit Datenrichtungslogik vorgenommen. Der Grundzustand ist hierbei prinzipiell die Datenrichtung zu den peripheren Elementen, damit die Befehlskodes überwacht werden können (interne RETI-Logik der peripheren Systemelemente). Bei erfolgter Kartenselektierung (Signal SEL der Adreßbereichsvorwahl) und gleichzeitiger Leseinformation der CPU (über  $\overline{RD}$ ,  $\overline{WR}$ ) sowie bei Quittierung einer Interruptforderung aus der Baugruppe

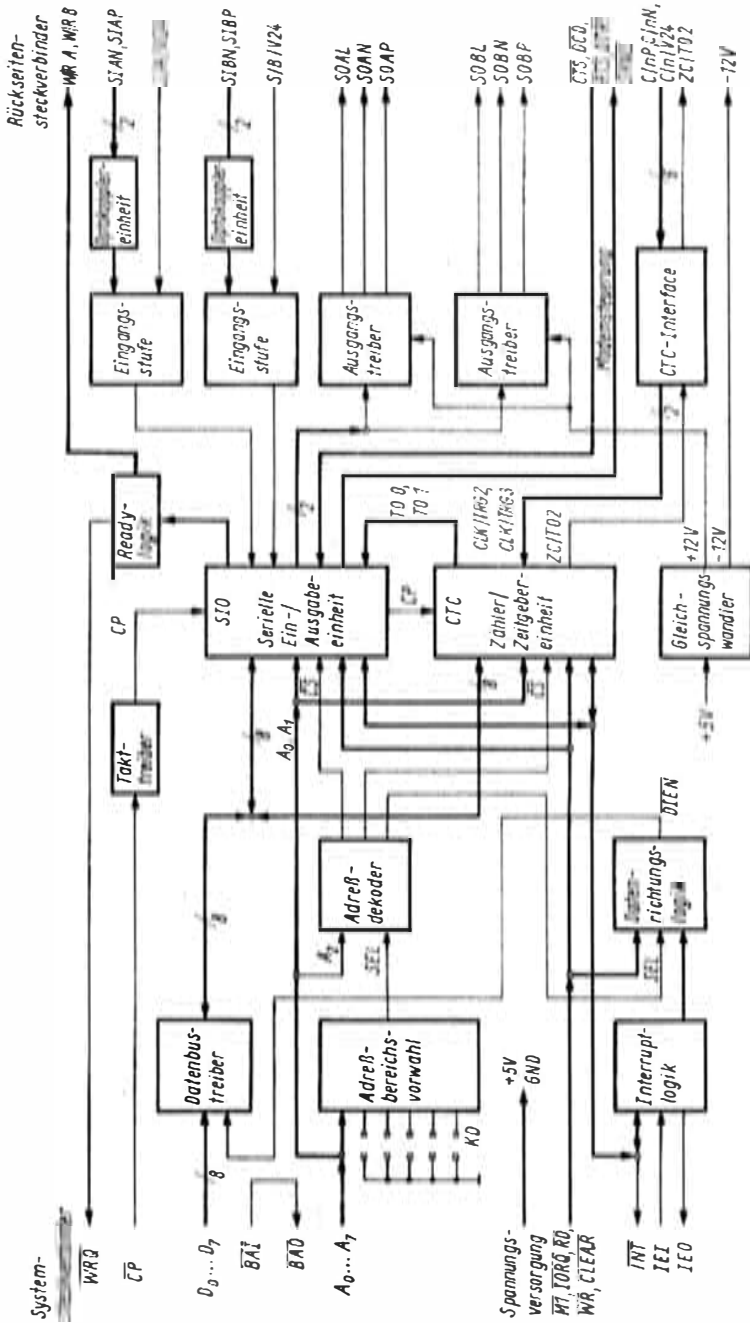


Bild 7.2.7. Blockschaltbild des seriellen Standardinterface FPS2 SI 1

(Aussenden des Interruptvektors) wird die Datenrichtungsumschaltung zum Prozessor vorgenommen.

Die Interruptlogik nimmt neben der Zusammenfassung der  $\overline{\text{INT}}$ -Anmeldelinie die Kaskadierung der Systemelemente SIO und CTC mit Hilfe der Prioritätskette vor. Sie enthält ebenfalls eine Umgehungslogik. Die in der Baugruppe enthaltenen peripheren Elemente können somit vektorisierte Interrupts entsprechend ihrer Spezifikation (siehe Abschnitte 3.3. und 3.4.) auslösen.

Die serielle Ein-/Ausgabe-Einheit kann außerdem schnelle Datenblocktransfers und Übertragungen im direkten Zusammenhang mit DMA-Einheiten ausführen. Hierzu dienen die mit der Readylogik verteilten WAIT/READY-Pins der SIO. Ein als WAIT-Pin programmierter Anschluß aktiviert den WAIT-Zustand des Prozessors (Anmeldelinie  $\overline{\text{WRQ}}$ ), wenn die CPU bei schnellen Blocktransfers versucht, von der SIO Daten zu lesen, obwohl die Empfangsregister leer sind, oder wenn die CPU versucht, Daten zu schreiben, obwohl die Senderegister voll sind. Die Programmierung als READY-Pin ermöglicht einen direkten DMA-Zugriff (Datenaustausch zwischen SIO und Speicherbaugruppe, unter Umgehung der CPU; gesteuert von separater DMA-Baugruppe) im Zusammenhang mit einem entsprechenden Freigabeeingang der DMA-Baugruppe. Durch die Readylogik werden einerseits die WAIT-Signale der beiden SIO-Kanäle AND-verknüpft auf die Anmeldelinie  $\overline{\text{WRQ}}$  geschaltet. Andererseits ermöglicht sie die kanalgetrennte Weitergabe der READY-A- bzw. READY-B-Signale bei DMA-Betrieb (Rückseitensteckverbinder).

Die eigentliche Festlegung der seriellen Betriebsarten, der Übertragungsraten, des Datenformats usw. muß programmäßig durch ein entsprechendes Softwarepaket vorgenommen werden. Gleiches gilt prinzipiell auch für die freien CTC-Kanäle. Die Baugruppe erlaubt somit eine an die bestehenden Forderungen anpaßbare Realisierung der seriellen Standardschnittstellen.

## 7.2.4. Prozeßnahe Ein-/Ausgabe-Baugruppen

### 7.2.4.1. Ausgabeinterface

Die Baugruppe FPS2.O11 stellt eine digitale parallele Ausgabebaugruppe (output interface) mit zwei 8-bit-Ports dar. Die Portausgänge der Baugruppe weisen eine Potentialtrennung zur Rechnermasse auf und sind für die Treibung von Strömen bis  $I_S = 200 \text{ mA}$  bei Schaltspannungen bis  $U_S = 24 \text{ V}$  ausgelegt. Sie eignen sich somit besonders für die Ansteuerung von NSF-Relais in industriellen Steuerungsanlagen.

Die Baugruppe ist auf einer der FPS2-Konfiguration entsprechenden Steckeinheit untergebracht. Diese Steckeinheit hat die Abmessungen  $180 \text{ mm} \times 95 \text{ mm}$  und besitzt einen System- und einen Rückseitensteckverbinder.

Bild 7.2.8 zeigt das Blockschaltbild der Baugruppe. Bei der Realisierung der Baugruppe wurde auf den Einsatz einer parallelen Ein-/Ausgabe-Einheit (PIO) verzichtet. Die verwendeten 8-bit-Register vom Typ DS8212 (Tesla) weisen in zweifacher Hinsicht Vorteile auf. Einerseits kann auf Datenbustreiber verzichtet werden, da die Spezifik der Baugruppe (parallele Ausgabe) keine programmierbare Interface benötigt. Die IS DS8212 haben entsprechend nur Dateneingänge, die je eine Low-power-Schottky-TTL-Last aufweisen. Andererseits sind die Ausgänge der DS8216 in der Lage, die nachgeschalteten Optokoppler direkt zu treiben. Ansonsten notwendige Treiberstufen (bei Einsatz der PIO) entfallen somit.

Die mit Optokopplern realisierten zur Rechnermasse potentialfreien Ausgangstreiber werden aus einem als Flußwandler wirkenden Transverter, der eine Versorgungsspannung von  $U_V = 5\text{ V}$  bei  $I_V = 2\text{ mA}$  bereitstellt, betrieben. Die Portlinien werden durch Open-collector-Schalter (SF128E), die gegen die vom Transverter kommende externe Masse schalten, gebildet.

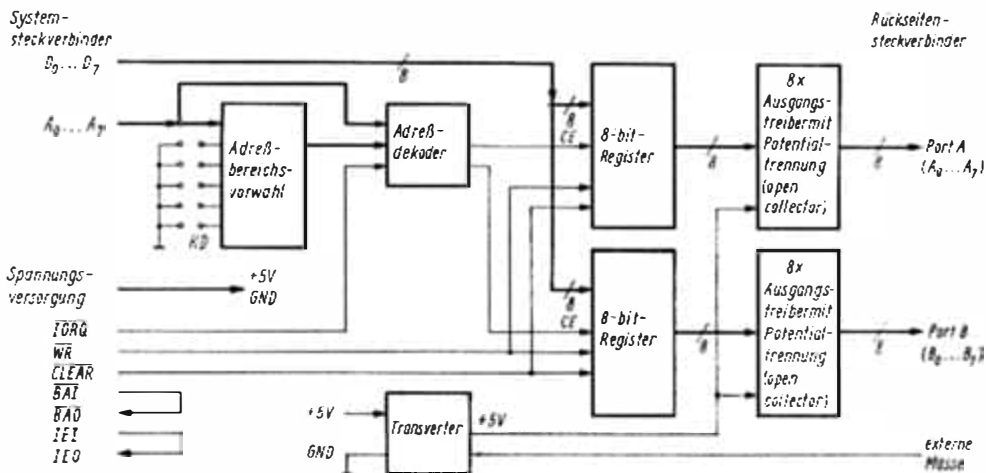


Bild 7.2.8. Blockschaltbild des digitalen parallelen Ausgabeinterface FPS2.01

Die beiden anderen Funktionseinheiten der Baugruppe dienen zur systemseitigen Ansteuerung des Interface. Über Kodierstecker kann mit Hilfe der Funktionseinheit Adreßbereichsvorwahl eine Kartenadresse in einem 64-bit-Teiladressierungsraum des Prozessors gebildet werden. Die beiden höchstwertigen zur Portauswahl benutzten Adreßlinien  $A_6$  und  $A_7$  sind hierbei nicht vorprogrammierbar. Im Adreßdekoder erfolgt, abhängig vom CPU-Steuersignal  $\overline{IORQ}$  und dem Kartenauswahlsignal der Adreßvorwahl, die Auswahl eines der beiden Ports. Das CPU-Steuersignal  $\overline{WR}$  wird direkt über die Freigabelogik der Register DS8212 mit diesem Ansteuersignal verknüpft.

Der Einsatz der Baugruppe FPS2.01 im Leiterplattensystem FPS2 ist aufgrund der Anwendungsspezifik problemlos. Es wird zwar kein gesondertes Ansteuerprogrammpaket benötigt, es können aber auch keine Interruptforderungen von der Baugruppe erzeugt werden.

### 7.2.4.2. Eingabeinterface

Die Baugruppe FPS2.II stellt ein digitales paralleles Eingabeinterface (input interface) dar. Sie dient in Verbindung mit einer zum Netzteil gehörenden Ansteuerbaugruppe FPS2.IMP zur zyklischen Abfrage von Relaiskontakten in industriellen Steuerungen. Aufgrund der hierbei vorliegenden Einsatzspezifik ist die Kontaktabfrage potentialfrei zur Rechnermasse. Der Abfragezyklus beträgt 128 ms. Um eine sichere Kontaktgabe der in industriellen Anlagen eingesetzten Kontakte (z.B. von NSF-Relais) zu gewährleisten, erfolgt die Kontaktabfrage mit Kontaktströmen von etwa 100 mA bei Schaltspannungen von etwa 24 V.

Die Eingabebaugruppe ist zum FPS2-System kompatibel. Sie ist auf einer Steckeinheit mit den Abmessungen 180 mm x 95 mm untergebracht. Im Bild 7.2.9 ist das Blockschalt-

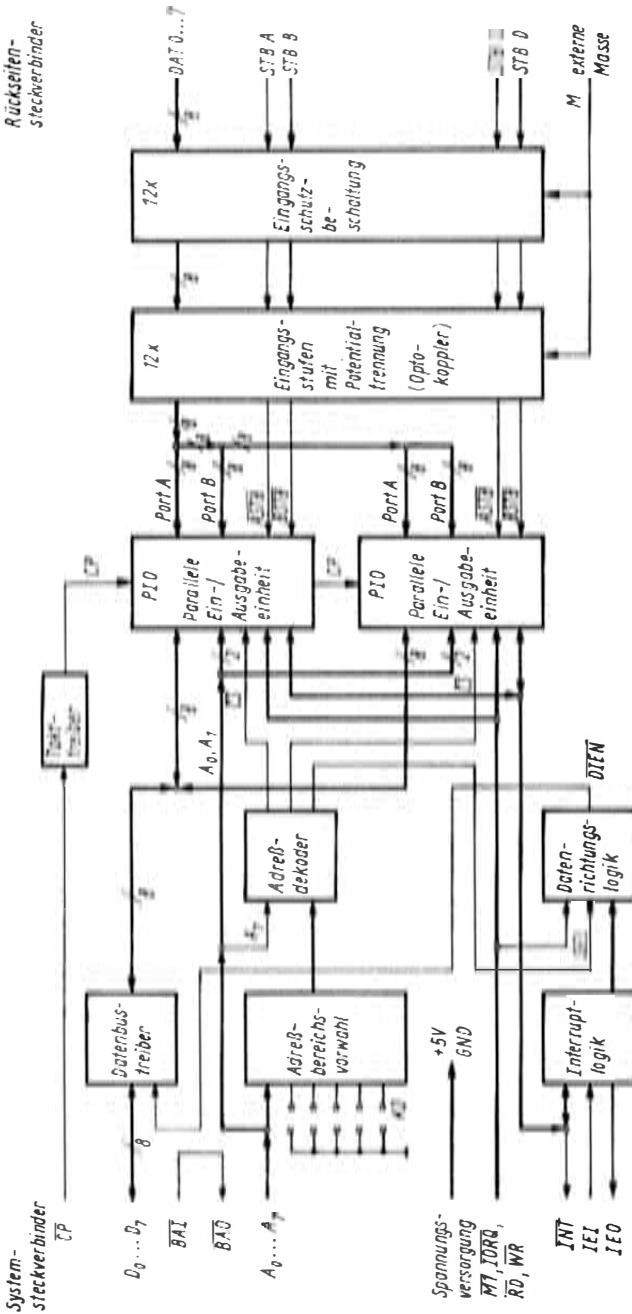


Bild 2.7.9. Blockschalbild des digitalen Eingabeinterface FPS2.II

bild der Baugruppe dargestellt. In ihm ist das Zusammenspiel der wichtigsten Funktionseinheiten verdeutlicht.

Als periphere Elemente wurden in der Eingabebaugruppe zwei parallele Ein-/Ausgabe-Einheiten U855 vorgesehen. Sie müssen jeweils in der Eingabemodus (Mode 1) betrieben werden, da alle vier Kanäle parallelgeschaltet sind und die Portübernahme über die STB-Linien erfolgt. Durch diese Maßnahme ergab sich einerseits vor allem eine Vereinfachung der Funktionseinheiten Eingangsschutzbeschaltung und Eingangsstufen, da nur acht Portlinien (DAT0... DAT7) gepuffert werden müssen; andererseits ergibt sich hieraus die Kapazität der Eingabebaugruppe. Sie beträgt 32 bit, das bedeutet, daß 32 Kontakte durch das Interface abgefragt werden können.

Die Eingangsschutzbeschaltung ist dimensioniert, damit auf den 12 gepufferten Linien (aus der Anlage) typische Belastungen der industriellen Steuerungstechnik derart unterdrückt werden, so daß keine Zerstörungen in der Baugruppe auftreten können. Solche typischen Belastungen auf Leitungen werden durch die Entladung eines auf 250 V aufgeladenen Kondensators, der eine Kapazität von  $C = 1 \mu\text{F}$  aufweist, über einen 100- $\Omega$ -Widerstand simuliert bzw. definiert. Des weiteren sind die Portdatenlinien der Eingabebaugruppe bei Beschaltung kurzschlußfest.

Die weiteren Funktionseinheiten der Eingabebaugruppe nehmen die systemseitige Ansteuerung der peripheren Systemelemente vor. Sie entsprechen in ihrer Funktion prinzipiell den entsprechenden Einheiten der SIF1000-Baugruppe (s. Abschn. 7.2.3.1.).

Zum Betrieb der Eingabebaugruppe sind zum einen Ansteuerimpulse (STB-Linien der PIO) und zum anderen Kontaktfrageimpulse mit entsprechender Strombelastbarkeit notwendig. Diese Impulse werden durch die Baugruppe FPS2.IMP bereitgestellt. Die Baugruppe ist wegen dieser Aufgabenstellung prinzipiell dem Netzteil der Steuerung zugehörig, wobei jeweils zu einer Eingabebaugruppe FPS2.III eine Ansteuerbaugruppe FPS2.IMP vorgesehen werden muß.

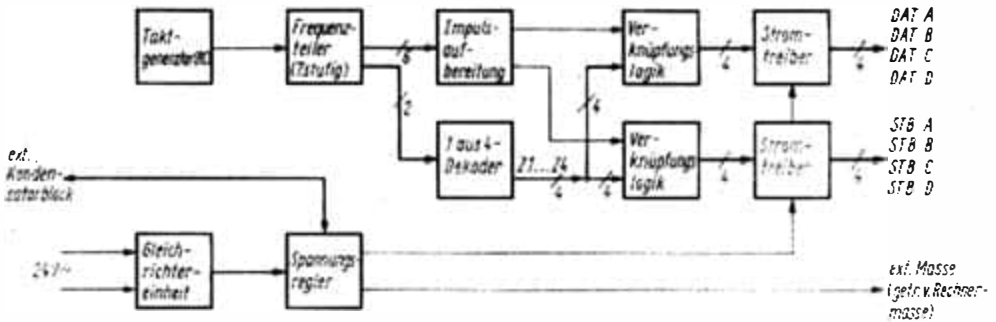


Bild 7.2.10. Blocksaltbild der Impulssteuerbaugruppe FPS2.IMP

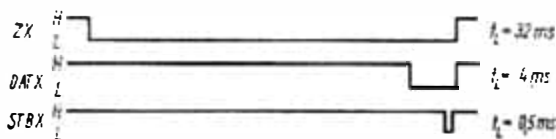
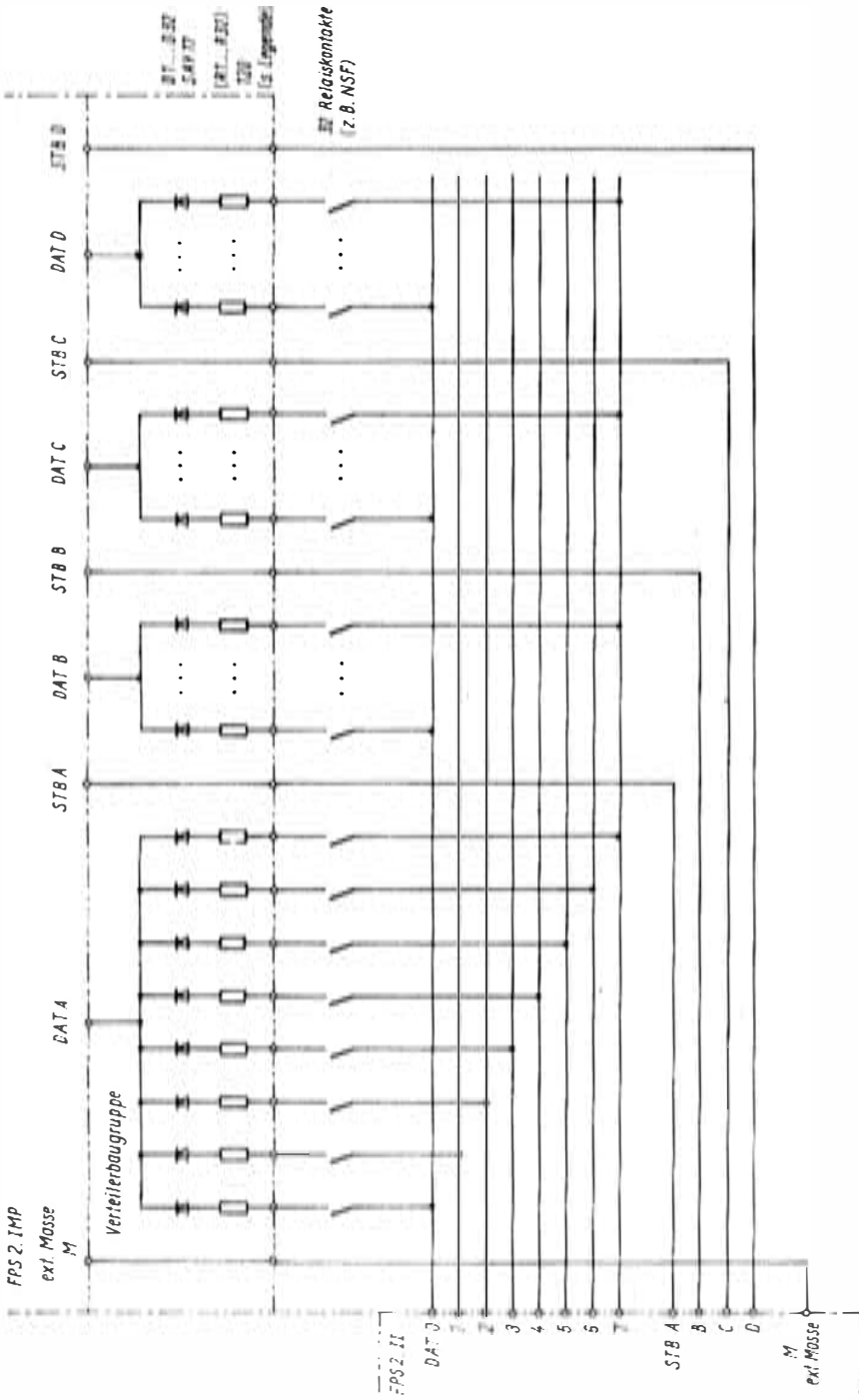


Bild 7.2.11. Zeitverhalten der Impulssteuerbaugruppe während der Abfrage einer Achtergruppe von Kontakten

- ZX Signal an einem der vier Dekoderausgänge (s. Bild 7.2.10, 1-aus-4-Dekoder)
- DAT X Datenabfrageimpuls am zugehörigen DAT-Ausgang der IMP-Baugruppe
- STB X Übernahmeimpuls am zugehörigen STB-Ausgang der Baugruppe



**Bild 7.2.12. Zusammenschaltung der Baugruppen FPS 2.II und FPS 2.IMP mit den Kontakten in der Peripherie**

Die Widerstände R1 ... R32 der Verteilerbaugruppe können vorgesehen werden, um die Verdrahtung der Kontakte in der Anlage untereinander kurzschlußfest zu gestalten. Ihre Werte müssen auf die Dimensionierung der Eingangsstromschaltung der Baugruppe FPS 2.111 abgestimmt sein

Die Ansteuerbaugruppe ist ebenfalls als Steckeinheit konzipiert. Sie hat die Abmessungen 170 mm × 95 mm und besitzt einen 58poligen Steckverbinder nach TGL 29331. Dieser Steckverbinder ist nicht buskompatibel (wegen Zugehörigkeit der Baugruppe zum Netzteil) und dient zur Verteilung der Ansteuersignale und zur Stromversorgung der Baugruppe.

Im Bild 7.2.10 ist das Blockschaltbild der Ansteuerbaugruppe dargestellt. Es enthält die wichtigsten Funktionseinheiten. Dem RC-Taktgenerator ist eine Frequenzteilerkette nachgeschaltet, die eine Impulsaufbereitungsstufe und einen 1-aus-4-Dekoder ansteuert. Es werden somit vier gleiche, aufeinanderfolgende Abfragezyklen erzeugt, die jeweils die Ansteuerung für eine Achtergruppe Kontakte beinhaltet. Bild 7.2.11 zeigt den hierbei auftretenden zeitlichen Ablauf. Hieraus ergibt sich die Zykluszeit für einen vollständigen (geschlossenen) Abfragevorgang. Sie beträgt somit 128 ms.

Zusammenfassend ist im Bild 7.2.12 die Zusammenschaltung der Eingabebaugruppe und der Ansteuerbaugruppe mit den Relaiskontakten dargestellt.

Aufgrund der Spezifik der vorliegenden zyklischen Kontaktabfrage ergeben sich programmtechnisch prinzipiell zwei Möglichkeiten der Informationsübernahme. Zum einen kann das Lesen der PIO-Kanäle durch Polling erfolgen, da neue Informationen prinzipiell nur alle 128 ms zu erwarten sind; zum anderen können von den  $\overline{STB}$ -Übernahmeimpulsen auch Interruptmeldungen abgeleitet werden, die somit alle 32 ms die Aktualisierung eines der vier Kanäle anzeigen.

### 7.2.4.3. Digital/Analog-Wandler

Die Baugruppe FPS2.DA1 stellt einen Digital/Analog-Wandler mit zwei Analogausgängen dar. Die beiden Wandlerkanäle weisen die nachfolgenden technischen Daten auf:

- Ausgangsspannungsbereich 0 ... 2,5 V
- Auflösung 8 bit
- mittlere Umsetzzeit 500 ms
- Genauigkeit  $\pm 0,5\% \pm 1$  LSB.

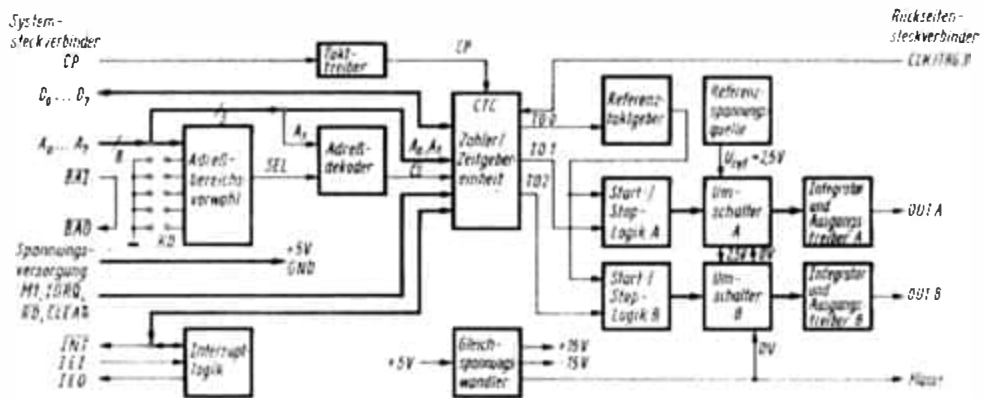


Bild 7.2.13. Blockschaltbild der D/A-Wandlerbaugruppe FPS2.DA1

Die Baugruppe ist als FPS2-zugehörige Steckeinheit realisiert. Ihre Abmessungen betragen 180 mm × 95 mm.

Im Bild 7.2.13 ist das Blockschaltbild der Wandlerbaugruppe mit den wesentlichen Funktionseinheiten dargestellt. Der D/A-Umsetzvorgang erfolgt bei der FPS2.DA1-Bau-

gruppe nach einem integrierenden Verfahren. Hierbei wird von einem Referenzzeitgeber ein dem Digitalwert 256 entsprechendes Zeitintervall erzeugt. Dieser Referenzzeitgeber wird von einem in der Zählermode betriebenen CTC-Kanal (Eingangsfrequenz entspricht der halben Systemtaktfrequenz) gebildet. Von den Nulldurchgängen dieses Referenzzählers werden nun zwei weitere, ähnlich betriebene CTC-Kanäle gestartet. Deren Nulldurchgänge wirken über Trigger auf zwei Umschalter, die jeweils durch den Referenzzähler wieder rückgesetzt werden. Die in diese beiden Zählkanäle eingegebenen Zeitkonstanten bestimmen somit das Tastverhältnis der Umschalter. Entsprechend der Wortbreite der Zeitkonstantenregister ergibt sich je Kanal eine Auflösung von 8 bit. Die den Umschaltern folgenden Integratorstufen setzen die von den auf die Referenzspannung wirkenden Umschaltern erzeugten Tastverhältnisse in entsprechende Analogspannungen um. Das gewählte Umsetzverfahren hat vor allem den Vorteil, daß relativ wenige genaue und stabile (insbesondere nur im Differenztemperaturkoeffizient  $\Delta TK$  stabile) Elemente benötigt werden.

Zur Spannungsversorgung der in der Baugruppe eingesetzten Operationsverstärker wird ein als Sperrwandler betriebener Transverter eingesetzt.

Die weiteren Funktionseinheiten der Baugruppe dienen zur systemseitigen Ansteuerung des Zähler/Zeitgebers U857. Der Adreßdekoder lokalisiert hierbei in Verbindung mit der Adreßbereichsvorwahl die Kanaladressen des CTC in einem 128-bit-Teiladressierungsbereich der CPU (der Adreßanschluß  $A_7$  ist nicht über Kodierstecker veränderbar).

Auf den Einsatz von Datenbuspuffern wurde bei der Baugruppe verzichtet, da der Datenverkehr prinzipiell nur in Richtung CTC erfolgt (Betriebsartenfestlegung, Einschreiben der Zeitkonstanten). Außerdem ist die Erzeugung von Interruptforderungen beim beschriebenen D/A-Wandlervorgang nicht sinnvoll. Das Aussenden von Interruptvektoren seitens der Baugruppe ist somit ebenfalls überflüssig.

In der Baugruppe wurde aber dennoch eine Interruptlogik (Interruptfreigabekaskadierung mit Umgehungsschaltung) vorgesehen, um in kleinen Steuerungssystemen (Last auf Datenbuslinien kleiner als 1,8 mA) den nichtgenutzten CTC-Kanal für Interruptaufgaben (z. B. Vektorerzeugung) nutzen zu können.

Der Einsatz des Zähler/Zeitgebers U857 in der D/A-Wandlerbaugruppe hat insbesondere den Vorteil, daß nach der relativ unkomplizierten Initialisierung der CTC-Kanäle der gesamte Umsetzvorgang ohne Programmunterstützung abläuft.

#### 7.2.4.4. Analog-Digital-Wandler

Das Vorhandensein analoger Eingangsschnittstellen stellt i.allg. ein Leistungskriterium für frei programmierbare Steuerungen dar. Im Leiterkartensystem FPS2 sind hierfür zwei Baugruppen vorgesehen, deren Eigenschaften nachfolgend aufgeführt werden.

Die Baugruppe FPS2.AD3 stellt einen Analog/Digital-Wandler für einen Analogeingang dar. Sie realisiert die nachfolgend aufgeführten technischen Daten:

- Eingangsspannungsbereich 0 ... 2,5 V
- Auflösung 10 bit
- mittlere Umsetzzeit 40 ms
- Genauigkeit  $\pm 0,5\%$   $\pm 1$  LSB.

Die Baugruppe ist auf einer FPS2-kompatiblen Steckeinheit mit den Maßen 180 mm  $\times$  95 mm untergebracht.

Die Wandlerbaugruppe FPS2.AD3 enthält eine monolithische A/D-Wandlereinheit,

die hinsichtlich der Ansteuerung der digitalen Signale über eine parallele Ein-/Ausgabereinheit U855 versorgt wird. Die anderen Funktionseinheiten weisen keine schaltungstechnischen Besonderheiten auf und entsprechen funktionell denen anderer FPS2-Baugruppen.

Die programmtechnische Einbindung der Baugruppe erfordert zu Beginn eines Wandlervorgangs eine softwaremäßige Initialisierung der Wandlereinheit, die über die PIO erfolgt. Nach Beendigung des Wandlervorgangs (etwa 40 ms) kann der Digitalwert auch durch Auslösen eines vektorierten Interrupts seitens der FPS2.AD3-Baugruppe verarbeitet werden. Der Prozessor wird durch den eigentlichen A/D-Wandlervorgang zeitlich nicht belastet.

Die Baugruppe FPS2.AD4 stellt einen A/D-Wandler mit vier Analogeingängen dar. Mit ihr werden die folgenden typischen technischen Daten erreicht:

- Eingangsspannungsbereich 0 ... 5,12 V
- typische Auflösung 10 bit
- mittlere Umsetzzeit 1,6 ms (bei 10-bit-Auflösung)
- Genauigkeit  $\pm 0,5\% \pm 1$  LSB
- vier Analogeingänge über Eingangsmultiplexer.

Die Baugruppe ist wiederum als FPS2-Steckeinheit (180 mm × 95 mm) ausgeführt.

Zur Verteilung der Eingangslinien ist ein 1-aus-4-Analogeingangsmultiplexer eingesetzt. Die hierüber erzeugte analoge Eingangsgröße wirkt auf einen mit diskreten Elementen realisierten A/D-Umsetzer, der nach dem Charge-balancing-Verfahren arbeitet. Die zeitliche Ansteuerung bei diesem Integrationsverfahren erfolgt über einen Zähler/Zeitgeber U857. Die zur systemseitigen Ansteuerung eingesetzten Funktionseinheiten entsprechen denen anderer FPS2-Baugruppen.

Die programmmäßige Ansteuerung des A/D-Wandlers FPS2.AD4 erfordert wiederum ein Programmpaket, das nur entsprechende Initialisierungen des Wandlervorgangs vornimmt. Die zeitliche Belastung des Prozessors bleibt hierbei gering.

### 7.2.5. Ergänzungsbaugruppen

Unter Ergänzungsbaugruppen für das Leiterkartensystem sollen Baugruppen verstanden werden, die keine engeren Rechnerfunktionen ausüben. Vielmehr dienen sie zur Komplettierung, Signalverteilung, Stromversorgung oder unterstützen die Inbetriebnahme, Bedienung und Testung des Steuerungssystems.

Im Leiterkartensystem FPS2 stellt die Baugruppe FPS2.RV2 eine gedruckte Rückverdrahtung dar, die zur Aufnahme von max. 18 systemzugehörigen Steckeinheiten genutzt werden kann. Hierbei ist der erste Steckplatz für die Prozessorbaugruppe reserviert. In den anderen Steckplätzen können nacheinander beliebige Systemkomponenten der FPS2 angeordnet werden. Diese Anordnung bestimmt bei den peripheren Baugruppen die Interrupt- und DMA-Priorität. Neben der Signalverteilung erfolgt auf der gedruckten Rückverdrahtung ebenfalls die Zufuhr der Versorgungsspannungen. Hierbei können die beiden vordersten Steckeinheiten über getrennte Stromversorgungseinheiten betrieben werden (Möglichkeit der Notstromversorgung). Konstruktiv ist die Baugruppe FPS2.RV2 den Gegebenheiten des einheitlichen Gefäßsystems (EGS) angepaßt.

Die Baugruppe FPS2.NT5 stellt eine Stromversorgungsbaugruppe dar, die strommäßig auf die Belange einer typischen Steuerungskonfiguration der FPS2 zugeschnitten ist. Sie

stellt folgende Versorgungsspannungen mit den aufgeführten max. Lastströmen zur Verfügung:

$$\begin{array}{ll} U_{CC} = 5 \text{ V} \pm 2\% & I_{CC} = 10 \text{ A} \\ U_{DD} = 12 \text{ V} \pm 2\% & I_{DD} = 4 \text{ A} \\ U_{GG} = -12 \text{ V} \pm 2\% & I_{GG} = 2 \text{ A} \\ U_{\sim} = 26 \text{ V} \pm 10\% & I_{\sim} = 500 \text{ mA.} \end{array}$$

Die stabilisierten Gleichspannungsquellen weisen hierbei sowohl eine Strombegrenzung mit Fold-back-Verhalten im Überlast- bzw. Kurzschlußfall als auch eine Überspannungssicherung (crow bar) bei falscher Quellbeschaltung auf. Die in der Baugruppe erzeugte, zu den anderen Spannungen potentialfreie Wechselfspannung kann zur Versorgung einer Ansteuerbaugruppe FPS2.IMP (s. Abschn. 7.2.3.2.) verwendet werden.

Im Steuerungssystem FPS2 sind weitere spezifische Ergänzungselemente vorhanden. Hierzu gehören eine Universalkarte (FPS2.UK mit Lochrasterstrukturen), eine Steckerkarte (FPS2.SK) sowie weitere, z. T. baugruppenspezifische Test- und Inbetriebnahmelemente.

Für Kommunikations- und Inbetriebnahmezwecke kann darüber hinaus das im Abschnitt 8. näher beschriebene Lernsystem für frei programmierbare Steuerungen eingesetzt werden.

### 7.3. Programmierarbeit

Im Zusammenhang mit dem Einsatz eines Steuerungssystems steht neben der hardwareseitigen Konfigurierung und Baugruppeninbetriebnahme die Erstellung des Steuerungsprogramms im Vordergrund.

Hierbei sind die notwendigen Einflußgrößen des zu regelnden oder zu steuernden Prozesses zu bestimmen. Hieraus müssen die entsprechenden Systemschnittstellen hardwaremäßig abgeleitet werden. Softwareseitig ergeben sich hieraus die zeitlichen und systembedingten Möglichkeiten der Ansteuerung dieser Schnittstellen. Außerdem muß aus der Aufgabenstellung mit Hilfe der Problemanalyse ein geeigneter Algorithmus für die Steuerung der Ein- und Ausgabefunktionen dieser Schnittstellen gefunden werden.

Prinzipiell erfolgt nach Bearbeitung dieser Schritte ein erneuter Eintritt in die Systemkonfigurierung. Dabei wird insbesondere die Abschätzung des Speicherbedarfs für Programm- und Datenspeicher vorgenommen und, daraus abgeleitet, die Entscheidung über den Einsatz entsprechender Baugruppen. Des weiteren sind zeitliche Abschätzungen des Prozesses hinsichtlich Echtzeitbetrieb der Steuerung vorzunehmen. Die hierbei gewonnenen Erkenntnisse haben Einfluß auf die Wahl der Programmiersprache (Assembler oder höhere Sprache) sowie auf die Struktur des Programms (Makroprogrammierung, Unterprogrammtechnik, Geradeausprogrammierung).

Danach wird i. allg. ein Programmablaufplan (PAP) erstellt, in dem der Ablauf der wichtigsten Operationen dargestellt ist. Bei Anwendung von Unterprogrammtechniken sind auch für diese Teilprogramme PAP zu erstellen. Gleichfalls ist es vorteilhaft, für die Unterprogramme wichtige Ein- und Ausgangsgrößen, Registerbelegungen, Eingangsadressen, benutzte Stackebenen bei weiteren Programmverzweigungen usw. übersichtlich (z. B. tabellarisch) festzuhalten.

Nach Auswahl der eigentlichen Programmiersprache, die abhängig vom Problem sowie

von den gerätetechnischen Möglichkeiten des Anwenders sein kann, geschieht das eigentliche Erstellen des Quellprogramms (source program). Bei Verwendung der Assembler-sprache wird das Umsetzen dieses Quellprogramms mit Hilfe eines Assemblerprogramms in das Maschinenformat (objekt programm) vorgenommen. Ein in einer höheren Programmiersprache erstelltes Programm erfordert zur Umsetzung in das Maschinenformat ein entsprechendes Compilerprogramm. Diese Erarbeitung der Objektprogramme kann auf einem Entwicklungssystem, das mit dem Prozessor U880 arbeitet, erfolgen. Ein solches Mikrorechnerentwicklungssystem ist das A5601 vom VEB Kombinat Robotron. Andererseits besteht aber auch die Möglichkeit, Objektprogramme für den U880 auf anderen Datenverarbeitungsanlagen zu erstellen. Hierzu ist dann aber eine entsprechende Crosssoftware (Crossassembler bzw. Crosscompiler) erforderlich.

Der nächste Schritt der Programmbearbeitung sind die Testung und Fehlerbeseitigung. Hierzu sind wiederum ein Entwicklungssystem [das den Anwenderprogrammmlauf mit Hilfe eines Steuerprogramms (debug) ausführt (z. B. im Schrittbetrieb oder blockweise)] bzw. entsprechende Simulationsprogramme auf anderen Datenverarbeitungsanlagen notwendig. Nach Erkennen von Fehlern wird das Quellprogramm verändert, eine erneute Übersetzung vorgenommen und wieder in den Programmtest eingetreten. Dieser Zyklus erfolgt so lange, bis ein von syntaktischen und logischen Fehlern freies Maschinenprogramm vorliegt.

Das Auffinden von logischen (aufgabenbezogenen) Programmfehlern kann bei Prototypanwendungen von Steuerungen bzw. Mikrorechnern durch eine Schaltkreissimulation (ICE, in-circuit-emulator) vereinfacht werden. Hierbei wird das Mikrorechnerentwicklungssystem über ein Kabel mit einem Simulationsstecker, der anstelle der CPU eingesetzt wird, mit dem Anwendersystem verbunden. Bei der Programmabarbeitung und -testung können hierdurch Baugruppen des Anwendersystems in den Ablauf einbezogen werden. Bei der Programmtestung können somit auch Reaktionen in der Rechnerperipherie getestet oder simuliert werden.

## 8. Anwendung des Prozessors U880 in einem Lernsystem

### 8.1. Aufgabenstellung und Konfiguration

Das Erreichen neuer wissenschaftlich-technischer Lösungen der Automatisierungstechnik, der Informationsverarbeitungstechnik bis hin zur Konsumgütertechnik verlangt die schnelle und umfassende Vorbereitung dieser Industriezweige auf die Technik der Mikroprozessoren. Wesentliche Voraussetzungen für den Einsatz dieser Generation von Bauelementen sind neben der Kenntnis des zu rationalisierenden oder zu automatisierenden Prozesses die Kenntnis der Leistungsfähigkeit der mikroelektronischen Schaltkreise sowie der Programmiersprache und Programmierertechnik.

Neben den theoretischen Betrachtungen und Untersuchungen (entsprechende Grundlagen enthalten die vorstehenden Abschnitte) zur Mikroprozessortechnik wird es für den Anwender i. allg. notwendig sein, einen praktischen, geräte- bzw. prozessorbezogenen Lernprozeß zu vollziehen. Das im folgenden beschriebene Lernsystem für frei programmierbare Steuerungen (VEB Funkwerk Erfurt, Applikation Bauelemente) bietet die Möglichkeit, als Grundlage eines derartigen Prozesses angewendet zu werden. Mit der vorgestellten gerätetechnischen Basis wird der Anwender in die Lage versetzt, sich mit dem Befehlssatz (Maschinenebene), der Arbeitsweise und der Struktur des Mikroprozessorsystems U880 vertraut zu machen. Sie ermöglicht das Erlernen der Programmiersprache und das Testen der vom Anwender selbst erstellten Software.



*Bild 8.1.1*  
*Ansicht des Lernsystems*  
*für frei programmierbare Steuerungen*  
Werkfoto: VEB Funkwerk Erfurt

Das Lernsystem ist als eigenständiges, kompaktes Gerät konzipiert, d. h., es werden prinzipiell keine weiteren Einheiten beim Einsatz benötigt. Es umfaßt in seiner Grundausbaustufe neben den im Frontbereich angeordneten Eingabe- und Anzeigeeinrichtungen, die zur Bedienung des Geräts dienen, bereits Standardschnittstellen und allgemein verwendbare Schnittstellen. Hierdurch wird ermöglicht, daß einerseits ein Standard-Ein-/Ausgabe-Gerät (z.B. Bildschirmbedieneinheit, Lochbandleser/-stanzer, Fernschreiber) oder ein NF-Kassettenbandgerät als billiger Datenspeicher zusätzlich angeschlossen werden kann; andererseits stehen für Anwendungs- bzw. Testzwecke die Interfacelinien

einer PIO U855 und Zähler/Zeitgeber-Kanäle eines CTC U857 an der Geräterückseite zur Verfügung. Bild 8.1.1 zeigt die Ansicht des Lernsystems.

Als Zusatzbaugruppen sind PROM-Programmiereinrichtungen für die Typen U551, U552 und U555 vorgesehen. Außerdem sind in den freien Steckerplätzen des Lernsystems aufgrund der weitgehenden Buskompatibilität Baugruppen der frei programmierbaren Steuerung FPS2 (s. Abschn. 7.) einsetzbar. Die unter diesen Gesichtspunkten erreichte Leistungsfähigkeit des Lernsystems für frei programmierbare Steuerungen erfüllt und übersteigt die Anforderungen von reinen Qualifizierungsaufgaben. Weitere Einsatzmöglichkeiten ergeben sich im Zusammenhang mit der Anwendung des Steuerungssystems FPS2. Das Lernsystem kann hierbei für Prototypanwendungen, als Test- und Inbetriebnahmegerät für die FPS2-Baugruppen oder als Unterrechner (z.B. für Ein-/Ausgabe-Funktionen) in der FPS2 (in hierarchischen Rechnersystemen) genutzt werden. Aufgrund der Eigenständigkeit des Lernsystems können aber auch kleinere Entwicklungsaufgaben (Programmierarbeit und -testung, PROM-Programmierung) ausgeführt werden.

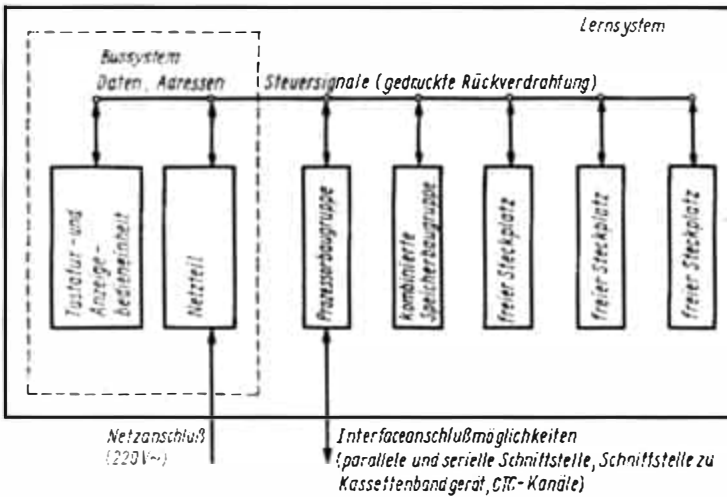


Bild 8.1.2. Struktureller Aufbau des Lernsystems

Im Bild 8.1.2 ist die Grundstruktur des Lernsystems für frei programmierbare Steuerungen dargestellt. Das Gerät enthält die Funktionsgruppen Prozessorbaugruppe, Speicherbaugruppe, Stromversorgungsbaugruppe (alle als Einschübe konzipiert) sowie eine Eingabe- und Anzeigebaugruppe (fest installiert). Als Leiterkartenformat der Einschubbaugruppen wurde das Einfachkartenformat des einheitlichen Gefäßsystems (170 mm × 95 mm) verwendet, um zur Steuerung FPS2-kompatibel zu sein und eine bessere Anpassung an die Aufgabenstellung des Lernsystems zu erreichen. Die Systembusverteilung erfolgt im Gerät über eine gedruckte Rückverdrahtung bzw. über Flachbandkabel (Verbindungen zur Eingabe- und Anzeigebaugruppe). Die Funktionsweise der Baugruppen wird im folgenden dargestellt.

## 8.2. Baugruppenbeschreibung

### 8.2.1. Prozessorbaugruppe

Die CPU-Baugruppe bildet das Herzstück des Lernsystems. Sie dient zur Erzeugung bzw. Auswertung sämtlicher Systeminformationen. Außerdem befinden sich auf dem Leiterkarteneinschub parallele und serielle Anschlußstellen sowie ein Zähler/Zeitgeber-Baustein. Insgesamt umfaßt die Baugruppe die Funktionseinheiten zentrale Verarbeitungseinheit (CPU U880), Systembustreibereinheit, Takterzeugungseinheit, Ansteuerungseinheit, Adreßdekoder, parallele Ein-/Ausgabe-Einheit, serielle Ein-/Ausgabe-Einheit und eine Einheit für Zähler/Zeitgeber-Funktionen.

Die Steckeinheit hat das Leiterkartenformat 180 mm × 95 mm (EGS) und besitzt steckerseitig einen 58poligen direkten Steckverbinder (TGL 29331), der zur Übertragung der Systembusinformationen dient und belegungskompatibel zur gedruckten Rückverdrahtung ist. Auf der Rückseite ist ebenfalls ein 58poliger Steckverbinder vorgesehen (Buchsenleiste, wahlweise direkt oder indirekt), der die anwenderseitig nutzbaren Signallinien nach außen legt.

Die Taktversorgungseinheit stellt den zur Synchronisation des Rechners und zum Betrieb der Systemelemente notwendigen Systemtakt zur Verfügung. Dieser Takt wird durch einen Quarzoszillator ( $f_Q = 9830,4 \text{ kHz}$ ) mit nachgeschalteten Triggern erzeugt und hat eine Frequenz von  $f_C = 2,4576 \text{ MHz}$ . Er kann somit ebenfalls zur Versorgung von seriellen Standardschnittstellen verwendet werden.

Die Systembustreibereinheit dient zur Verstärkung der CPU-Adreßbussignale sowie der CPU-Steuersignale  $\overline{\text{MREQ}}$ ,  $\overline{\text{IORQ}}$ ,  $\overline{\text{RD}}$ ,  $\overline{\text{WR}}$ ,  $\overline{\text{RFSH}}$ ,  $\overline{\text{HALT}}$ ,  $\overline{\text{MI}}$ ,  $\overline{\text{BUSAK}}$ . Sie ist mit Low-power-Schottky-Bauelementen und D10-Gattern bestückt und erhöht den TTL-Lastfaktor der genannten Linien. Dadurch wird die systemseitige Treibung der anderen, mit Bauelementen der D10- und D20-Reihe bestückten Baugruppen des Lernsystems ermöglicht. Die bidirektionalen Datenbuslinien des Prozessors wurden wegen der Einsatzspezifik der Baugruppe (ausschließlich Lernsystem bzw. kleine Rechnersysteme) nicht getrieben. Sie sind deshalb mit einer Standard-TTL-Last belastbar. Das ist aber ausreichend, um eine begrenzte Anzahl von MOS-Eingängen (bis max. 200 pF Gesamtkapazität auf den Linien) und LS-TTL-Eingängen (bis max. vier) zu treiben und genügt den Einsatzanforderungen des Lernsystems (Ansteuerung von max. vier weiteren Baugruppen). Die von der Funktionseinheit verstärkten Signale weisen baugruppenseitig ebenfalls wieder Tristateausgänge auf (bis auf  $\overline{\text{MI}}$  und  $\overline{\text{BUSAK}}$ ) und werden bei einer Busanforderung in den hochohmigen Zustand geschaltet (z. B. bei DMA-Betrieb).

Die Ansteuereinheit übernimmt die systemseitige Versorgung der zentralen Verarbeitungseinheit U880. Hierzu gehört die Erzeugung eines aktiven  $\overline{\text{RESET}}$ -Impulses bei Zuschaltung der Versorgungsspannung, um einen Programmstart vom Speicherplatz 0000H zu erreichen. Der ebenfalls zu dieser Funktionseinheit gehörende WAIT-Generator erzeugt entweder auf externe Anforderung WAIT-Zustände (Linie  $\overline{\text{WRQ}}$ ) oder synchronisiert auf Forderung der Speicherbaugruppe deren Lesezyklen mit langsamen Speicherbauelementen U551 durch Einschleichen von zwei WAIT-Zuständen in die betreffenden Speicherzugriffe des Prozessors (Linie  $\overline{\text{MWRQ}}$ ).

Der Adreßdekoder dient zur Auswahl der in der CPU-Baugruppe lokalisierten peripheren Einheiten. Hierfür werden die Freigabesignale der PIO, des CTC und des USART-Bausteins sowie weitere Ansteuersignale für das serielle Interface erzeugt.

Mit der parallelen Ein-/Ausgabe-Einheit erfolgt die Realisierung einer für den Anwen-

der frei verwendbaren parallelen Schnittstelle. Die hierfür vorgesehenen Interfacelinien der PIO U855 sind über den Rückseitensteckverbinder der CPU-Leiterkarte nach außen geführt und somit von der Rückseite des zusammengebauten Lernsystems erreichbar.

Die in der Baugruppe vorhandene serielle Schnittstelle (serielle Ein-/Ausgabe-Einheit) wird durch einen USART-Baustein 8251 (universal synchronous/asynchronous receiver/transmitter) gebildet und ermöglicht den Datenverkehr zwischen dem Prozessor und einem externen Gerät. Wahlweise kann über den Rückseitensteckverbinder die serielle Schnittstelle über ein NF-Kassettenbandgeräteinterface, eine Stromschleifenschnittstelle oder eine Spannungsschnittstelle ähnlich dem RS232-Standard benutzt werden. Das NF-Kassetteninterface wird durch eine einfache Amplitudenmodulations/-demodulations-Stufe gebildet. Es realisiert die direkte Anschlußmöglichkeit eines handelsüblichen Kassettenbandgeräts und die damit verbundene Nutzung billiger Datenträger. Die Stromschleifenschnittstelle wird durch einen Optokoppler mit nachfolgender Impedanzwandlerstufe und einen pnp-Transistor mit Ansteuerlogik gebildet. Sie dient vor allem zur Ankopplung eines Fernschreibers (TTY-teletype) und wird über eine externe Stromquelle mit  $I_K = 20 \text{ mA}$  betrieben. Eine Spannungsschnittstelle wird durch die wahlweise Nutzung der Interfacetreiberelemente ebenfalls am Rückseitensteckverbinder bereitgestellt. Sie arbeitet ähnlich dem Standardinterface RS 232, ausgenommen die Werte der H- und L-Pegel. Des weiteren sind die Kontrollsignale des USART-Elements an den Rückseitensteckverbinder geführt. Sie signalisieren den jeweiligen Betriebszustand des USART und können in Verbindung mit den PIO-Linien des Rückseitensteckverbinders zur Erzeugung von Interruptanforderungen genutzt werden.

Die Zähler/Zeitgeber-Funktionseinheit wird durch einen CTC U857 gebildet. Der Kanal 0 des CTC dient in der Prozessorbaugruppe zur Erzeugung des Empfänger- und Sendetacts der seriellen Ein-/Ausgabe-Einheit. Er muß deshalb softwaremäßig in die Betriebsart Zeitgeber gesetzt werden und erlaubt die programmabhängige Wahl der Datenübertragungsrate ( $DÜ = 110 \text{ baud}$  für NF-Kassettenanwendung und TTY) an der seriellen Schnittstelle. Der Kanal des CTC wird ebenfalls in der Zeitgebermode betrieben und liefert ein Taktsignal (Interruptauslösung) zur Ansteuerung der Anzeigebaugruppe des Lernsystems. Die Kanalein- und -ausgänge der CTC-Kanäle 1, 2 und 3 sind auf den Rückseitensteckverbinder der Leiterkarte geführt. Damit stehen dem Anwender weitere Rechnerfunktionen frei zur Verfügung.

Die Interruptfreigabelinien (IEI, IEO) der PIO und des CTC sind ohne Verwendung einer Umgehungslogik kaskadiert, so daß der CTC interruptmäßig höher priorisiert ist. Die Freigabelinien sind am Systembussteckverbinder herausgeführt und können dort entsprechend dem vorgesehenen Einsatzfall modifiziert werden.

Eine Teildokumentation zur Prozessorbaugruppe ist im Anhang A.4. vorhanden (Logikplan, Stückliste, Belegungsplan des Rückseitensteckverbinders).

### 8.2.2. Speicherbaugruppe

Die Speicherbaugruppe umfaßt sowohl einen Festwertspeicherbereich (ROM) als auch einen Schreib/Lese-Speicher (RAM). Sie dient somit zur Aufnahme des Monitorbetriebsprogramms des Lernsystems (im ROM), stellt den Stackbereich (Kellerspeicher) des Rechners (im RAM) und weist einen für den Anwender frei verfügbaren RAM-Bereich auf. Der Festwertspeicher der Baugruppe hat eine Kapazität von 2 Kbyte. Er kann mit PROM-Bauelementen U551 (EPROM: U552) bestückt werden. Der Schreib/Lese-Speicher hat eine Kapazität von 1 Kbyte und wird mit RAM U202 realisiert. Die Lokalisation dieser Speicherbereiche ist im Bild 8.2.1 dargestellt.

Die Speicherbaugruppe besteht aus den Funktionseinheiten Adreßdekoder, Festwertspeichereinheit, ROM-Datenbustreiber, WAIT-Anforderung, Schreib/Lese-Speichereinheit und dem bidirektionalen Datenbustreiber. Sie ist auf einer Steckeinheit (170 mm × 95 mm) untergebracht und hat einen direkten 58poligen Steckverbinder zur Übertragung der Systembusinformationen (belegungskompatibel zur gedruckten Rückverdrahtung). Auf eine Adreßbereichsvorwahl wurde wegen der Einsatzspezifik Lernsystem und der geringen Speicherkapazität der PROM-Elemente verzichtet. Speichererweiterungen können somit nur durch den Einsatz der universellen Speicherbaugruppen der FPS2-Steuerung oder durch hardwaremäßige Modifizierung der Lernsystem-Speicherbaugruppe vorgenommen werden.

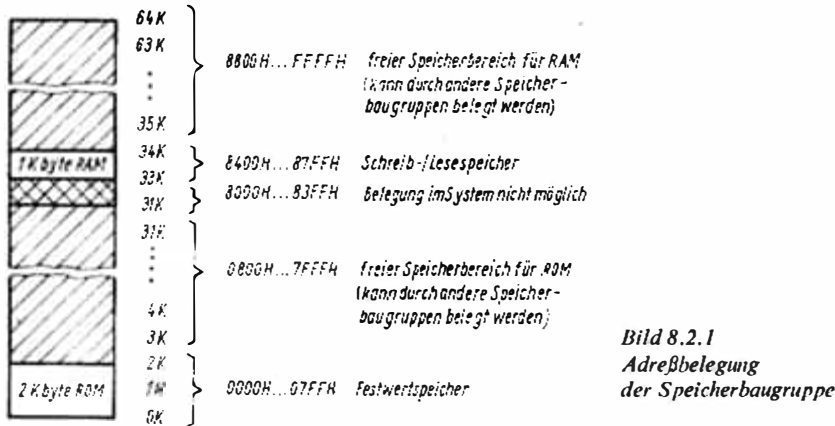


Bild 8.2.1  
Adreßbelegung  
der Speicherbaugruppe

Der Adreßdekoder der Speicherbaugruppe erzeugt die Chipauswahlsignale der Speicherbauelemente. Er wird durch einen 1-aus-8-Dekoder MH3205 und Elementen der D10-Serie gebildet.

Die Festwertspeichereinheit enthält acht Steckplätze für die in p-Kanal-MOS-Technologie gefertigten Elemente U551/U552. Eine Abrüstung der Gesamtkapazität von 2 Kbyte dieser Funktionseinheit ist in Schritten von 256 byte möglich. Die Ausgangslinien der Festwertspeichereinheit werden über eine ROM-Datenbustreibereinheit gepuffert, um Buskonflikte mit den n-Kanal-Bauelementen (RAM) wegen der auftretenden negativen Ausgangsspannungen zu vermeiden.

Aufgrund der hohen Zugriffszeiten der PROM-Bauelemente ( $t_{ACC} \leq 1000$  ns) müssen in die zugehörigen Speicherleseoperationen zwei WAIT-Zustände eingefügt werden. Die Anforderung hierfür erfolgt in Abhängigkeit von der Chipauswahl der Festwertspeichereinheit über die Linie  $\overline{MWRQ}$ . Die eigentliche WAIT-Erzeugung wird auf diese Anforderung in der Prozessorbaugruppe vorgenommen.

Die Schreib/Lese-Speichereinheit wird durch acht RAM-Bauelemente U202 gebildet. Da diese Elemente voll dekodiert (1024 bit × 1-organisiert) sind, ist eine Abrüstung dieser Funktionseinheit nicht gegeben. Das Zeitverhalten der RAM U202 sowie die im System auftretenden TTL-Verzögerungszeiten gestatten die Anwendung der CPU-Standard-speicherzugriffe. Es brauchen somit keine WAIT-Zustände angefordert werden.

Die Speicherbaugruppe enthält außerdem eine bidirektionale Bustreibereinheit, die einerseits in der Baugruppe die Datenlinien der Speicherblöcke zusammenfaßt und eine Trennung der Ein- und Ausgangslinien an der Schreib/Lese-Speichereinheit vornimmt, andererseits wird die kapazitive Belastung des Systemdatenbusses abgebaut. Die Freigabe

des Datenbuspuffers erfolgt bei vorhandener Chipauswahl eines Speicherelements. Die Datenrichtungsschaltung wird durch das CPU-Steuersignal  $\overline{RD}$  vorgenommen.

Im Anhang A.5. ist ebenfalls eine Teildokumentation (Logikplan, Stückliste) zur Speicherbaugruppe vorhanden.

### 8.2.3. Tastatur- und Anzeigebedienbaugruppe

Dem Anwender des Lernsystems stehen zur Kommunikation mit dem Gerät ein Tastenfeld von 32 nichtrastenden Tasten, eine sechsstellige Siebensegmentanzeige sowie zwei Einzel-LED zur Verfügung. Diese Elemente sind auf drei fest im Gerät montierten Leiterkarten untergebracht. Außer der eigentlichen Eingabe- und Anzeigebedienung enthält die Baugruppe eine Funktionseinheit zur Erzeugung eines nichtmaskierbaren Interrupts (NMI). Der prinzipielle Aufbau der Tastatur- und Anzeigebedienbaugruppe ist aus Bild 8.2.2 ersichtlich.

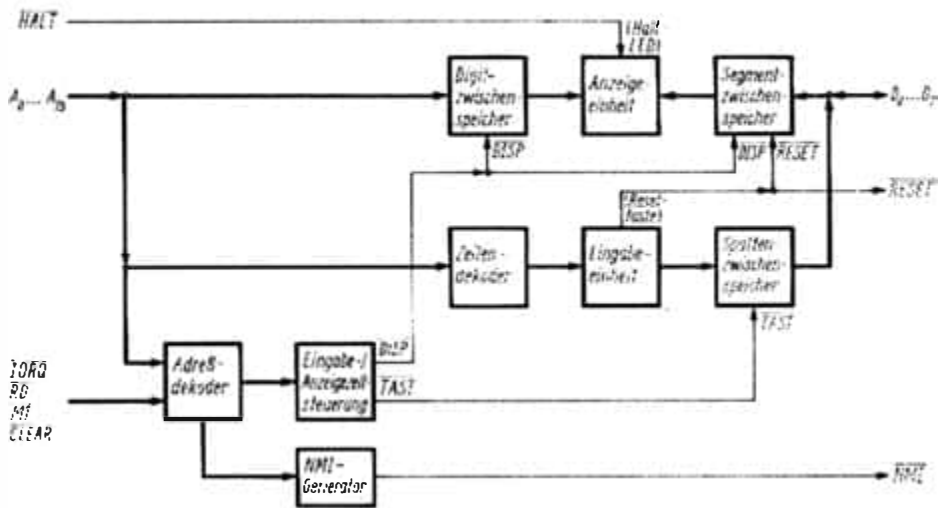


Bild 8.2.2. Blockschnittbild der Tastatur- und Anzeigebedienbaugruppe

Die manuelle Eingabe von Befehlen und Informationen erfolgt über ein aus 32 Einzel-tasten zusammengesetztes Tastenfeld. Die Abfrage von 31 Tasten wird programmäßig vorgenommen, während die eine verbleibende Taste (RESET-Taste) nur im Zusammenhang mit entsprechender Hardware zur Wirkung kommt. Die vom Monitorprogramm des Lernsystems kontrollierten Tasten sind zu einem Feld von vier Zeilen und acht Spalten angeordnet. Die einzelnen Tastenelemente arbeiten als Schließer und haben einen mittleren Übergangswiderstand von  $R_0 = 150 \dots 200 \Omega$ . Das Wirkungsprinzip der Tastaturabfrage besteht darin, daß programmgemäß eine zyklische Übernahme der Zeileninformation erfolgt. Der Abfragezyklus wird etwa alle 20 ms ausgelöst. Die softwaremäßige Übernahme der Tastenwertigkeiten (Zeileninformationen) wird durch den indirekt adressierten Eingabebefehl IN A ausgeführt, um den hierbei durch das CPU-Register B belegten höherwertigen Adreßbus ( $A_{15} \dots A_8$ ) für den Dekodiervorgang der Zeilenauswahl mitbenutzen zu können.

Die Anzeige von Adreß- und Dateninformationen geschieht beim Lernsystem über ein sechsstelliges Siebensegmentdisplay. Über zwei einzelne Leuchtdioden wird die Fehler-

(ERROR-) und Haltzustands- (HALT-) Information zur Anzeige gebracht. Die Ansteuerung der Anzeigeelemente erfolgt im time-sharing. Hierbei wird programmgestützt jede Stelle (Digit) für jeweils etwa 0,5 ms aufgetastet. Ausgenommen von diesem Ablauf ist die HALT-Anzeige, die statisch durch das zugehörige CPU-Steuersignal betrieben wird. Die Stromversorgung für die Anzeigeelemente wird als unregelmäßige Spannung (etwa 7,2 V) der Netzteilbaugruppe entnommen. Die softwaremäßige Ausgabe der Segmentinformation und der Digitwertigkeit erfolgt ebenfalls über einen indirekt adressierten Portbefehl (OUTA), bei dem wiederum die Belegung des höherwertigen Adreßbusses mit dem B-Register der CPU ausgenutzt wird, um die Digitauswahl vorzunehmen. Der Aufruf des Anzeigebedienprogramms wird durch den CTC-Kanal 1 der CPU-Baugruppe erzeugt. Dieser Kanal wird in der Zeitgebermode betrieben und fordert etwa nach jeweils 0,5 ms einen Interrupt an. Deshalb kann durch die Sperrung dieses Kanalinterrupts, bei Sicherstellung der Dunkelastung der Segmente, die Lernsystemanzeige abgeschaltet werden. Eine unvorbereitete Sperrung dieser Kanalanforderung oder der Prozessorinterruptfreigabe kann zur Zerstörung eines Anzeigeelements führen (ständige Aktivierung eines Elements mit hohem Strom).

In der Baugruppe ist des weiteren eine NMI-Erzeugungseinheit angeordnet. Diese Funktionseinheit dient zur Unterstützung der Einzelschrittbearbeitung von Anwenderprogrammen. Die Betätigung der STEP-Taste hat einen Sprung aus dem Monitorbetriebsprogramm des Lernsystems in das Anwenderprogramm zur Folge. Nach der Abarbeitung eines Anwenderprogrammbefehls muß hardwaremäßig ein Rücksprung in das Betriebsprogramm ausgelöst werden. Softwaremäßig wird die Auslösung eines Einzelbefehls durch eine Ausgabeoperation zur NMI-Funktionseinheit vorangekündigt. Dieser OUT-Befehl aktiviert eine  $\overline{MI}$ -Zählschaltung, die nach dem Erkennen des dritten nachfolgenden  $\overline{MI}$ -Signals ( $\overline{MI}$ -Maschinenzyklus) eine NMI-Anforderung auslöst. In der zugehörigen Interruptserviceroutine erfolgt dann die Auswertung des Anwenderprogrammbefehls und die Rückkehr in das Monitorbetriebsprogramm.

Im Anhang A.6. ist eine entsprechende Teildokumentation zur Tastatur- und Anzeigebedienbaugruppe vorhanden (Logikplan, Stücklisten).

#### 8.2.4. Rückverdrahtung

Die im Lernsystem eingesetzten steckbaren Leiterkarten (CPU-Karte, Speicherkarte) sind über eine gedruckte Rückverdrahtungsleiterkarte miteinander verbunden. Mit Ausnahme von vier Anschlüssen ist diese Rückverdrahtung so ausgeführt, daß Steckverbinderpins gleicher Bedeutung jeweils parallel miteinander verknüpft sind. Die Anschlüsse IEI, IEO,  $\overline{BAI}$ ,  $\overline{BAO}$  sind Bestandteile zweier Kaskadierungsschaltungen: der Interruptkaskadierung und der Busbestätigungskaskadierung. Diese Linien sind so ausgeführt, daß der Ausgang (IEO bzw.  $\overline{BAO}$ ) eines Steckplatzes mit dem Eingang (IEI bzw.  $\overline{BAI}$ ) des folgenden Platzes verbunden ist. In den Steckeinheiten muß deshalb immer eine mittelbare oder unmittelbare Verknüpfung der Kaskadierungsein- und -ausgänge vorliegen. Prinzipiell sind alle Steckplätze bis auf den CPU-Steckplatz funktionell gleichwertig. Unterschiede ergeben sich nur hinsichtlich der Priorität in den Kaskadierungslinien (höchstwertiger Steckplatz ist links).

Über die Rückverdrahtungskarte erfolgt ebenfalls der Anschluß der Regelnetzteilkarte und über einen 58poligen direkten Steckverbinder die Zuführung der Informationslinien der Tastatur- und Anzeigebaugruppe sowie der unregelmäßigen Gleichspannungen zum Regelnetzteil.

### 8.2.5. Stromversorgungsbaugruppe

Die Baugruppe besteht aus zwei Baueinheiten, die konstruktiv als Baugruppeneinschübe (EGS) konzipiert sind. Die eine Baueinheit dient zur Bereitstellung der unregelmäßigen Gleichspannungen. Sie enthält die Funktionseinheiten Netzeingang (Gerätebuchse, Sicherungen, Netztafter), Netzverdrosselung, Netztransformator und die Ladekondensatoren für die beiden unregelmäßigen Gleichspannungen. Die zweite Netzteilbaueinheit enthält die Regelnetzteilkarte mit einer zugehörigen Kühlkörperanordnung. Sie dient zur Aufbereitung der geregelten Versorgungsspannungen

$$U_{CC} = 5 \text{ V} \pm 5\%$$

$$U_{DD} = -9 \text{ V} \pm 5\%$$

die zum Betrieb der Leiterkarteneinschübe benötigt werden. Die Spannungsausgänge des Regelnetzteils sind mit einer Überspannungssicherung versehen (Crow-bar-Sicherung) und weisen rückläufiges Verhalten im Kurzschlußfall (Fold-back-Charakteristik) auf. Die Versorgungsspannung  $U_{CC}$  ist bis  $I_{CC} = 5 \text{ A}$ , die Spannung  $U_{DD}$  bis  $I_{DD} = 1 \text{ A}$  belastbar. Zur Erreichung dieser Charakteristik wurden integrierte Spannungsregler MAA723 eingesetzt. Die Bedieneinheiten der Stromversorgungsbaugruppe (Netzbuchse, Netztafter, Sicherungen) sind von der Rückseite des Lernsystems erreichbar.

Eine Teildokumentation zur Stromversorgungsbaugruppe ist im Anhang A.7. enthalten (Stromlaufpläne, Stücklisten).

## 8.3. Beschreibung der Betriebssoftware

### 8.3.1. Systemvereinbarungen

Für den Betrieb benötigt die Systemhardware des Lernsystems ein Steuerprogramm. Dieses wird als Monitor- oder Monitorbetriebsprogramm bezeichnet und organisiert die Ein-/Ausgabe-Funktionen des Geräts sowie eine gezielte Datenbehandlung im Schreib-/Lese-Speicher des Lernsystems. Des weiteren ermöglicht es den schrittweisen oder automatischen Anwenderprogrammablauf und bietet Möglichkeiten der Testung des Anwenderprogramms. Prinzipiell gestattet ein Monitorbetriebsprogramm eine Abarbeitung in zwei unterschiedlichen Betriebsarten. Die erste Betriebsart (monitor mode) realisiert die Bearbeitung des Betriebsprogramms mit den zugehörigen Funktionen der Kommunikation und Systemsteuerung. In der zweiten, der Anwenderbetriebsart (user mode) erfolgt die Umschaltung der CPU auf das Anwenderprogramm. Hiermit wird ein Anwenderprogrammablauf erreicht. Der Übergang zwischen den beiden Betriebsarten kann über Eingabefunktionen des Geräts (Tasten) oder programmgesteuert geschehen.

Das im Lernsystem für frei programmierbare Steuerungen eingesetzte Monitorbetriebsprogramm belegt in seiner Grundaustufe einen Festwertspeicherbereich von 1,5 Kbyte. Es kann somit in sechs PROM U551 der Speicherbaugruppe untergebracht werden. Da bei ROM-Speicherzugriffen automatisch jeweils zwei WAIT-Zustände eingefügt werden und da die verwendete Taktversorgung des Prozessors nicht mit der maximalen Systemtaktfrequenz erfolgt, verlängern sich die Befehlsabarbeitungszeiten der CPU definiert. Die Programmliste zur Grundaustufe des Monitorbetriebsprogramms LS880 ist im Anhang A.3. enthalten. Die in den folgenden Abschnitten genannten Vereinbarungen beziehen sich auf dieses Listing.

Das Monitorprogramm belegt außerdem 61 Speicherzellen (Bytes) im Schreib/Lese-Speicher (RAM) des Lernsystems für Hilfsfunktionen (Hilfszellen). Diese Zellen dürfen vom Anwender nicht anderweitig verwendet werden. Im RAM-Bereich der Speicherbaugruppe ist ebenfalls der Stack (Kellerspeicher) lokalisiert. Die Stackbelastung kann in der Monitorbetriebsart maximal 21 byte umfassen. Für den Anwender bleibt somit in der RAM-Speichereinheit ein Schreib/Lese-Speicherbereich von 942 byte frei verfügbar. Die entsprechende Adreßzuordnung dieser Speicherbereiche ist in Tafel 8.3.1 dargestellt.

Tafel 8.3.1. Adreßzuordnung im RAM-Speicher

Adresse	Kapazität/byte	Verwendung
8400H ... 87ADH	942	freie Nutzung durch Anwender und Stackerweiterung
87AEH ... 87C0H	19	Monitorstackbereich
87C1H 87C2H	2	Beginn des Monitorstack
87C3H ... 87DBH	25	Speicher für Monitor- und Anwender-CPU
87DCH ... 87F7H	28	Adreß- und Markenspeicher für Monitor
87F8H	1	Speicher für Anzeigestatus
87F9H	1	Speicher für Errorbit
87FAH 87FBH	2	Speicher für Daten-LED 0 und Daten-LED 1
87FCH ... 87FFH	4	Speicher für Adreß-LED 0 bis Adreß-LED 3

Mit dem Einschalten der Versorgungsspannungen des Lernsystems (power-on reset) und bei Betätigung der Taste RESET wird im System ein Rücksetzvorgang ausgelöst. Hierdurch werden die Systemelemente CPU und CTC in den Grundzustand gebracht. Die RESET-Taste hat deshalb die höchste Tastenpriorität. Nach dem Rücksetzvorgang beginnt der Prozessor U880 mit der Programmbearbeitung in der Monitorbetriebsart ab Speicherplatzadresse 0000H. Hierbei erfolgt zunächst eine Systeminitialisierung. Sie beinhaltet das Löschen der LED-Anzeigetableaus (0000; 00) und der ERROR-Leuchtdiode sowie das Programmieren der Betriebsarten für die in der Prozessorbaugruppe angeordneten Elemente CTC und USART. Nach der Initialisierungsroutine kommt der Eintritt in die Tastaturabfrageschleife. Der Prozessor erwartet dann eine gültige Tastenbetätigung (Kommando) und organisiert bei deren Erkennung die entsprechende Programmverzweigung.

Der Eintritt in das Monitorbetriebsprogramm kann aber auch aus einem laufenden Anwenderprogramm geschehen. Hierbei ergibt sich neben dem Rückstart zur Programmadresse 0000H (entspricht funktionsmäßig etwa einem hardwaremäßigen Rücksetzvorgang) zum einen die Möglichkeit, einen Sprung in die Tastaturabfrageschleife auszuführen. Zugehörige Marken im Programmlisting sind ATAST, TAST, LEDATA und LEDADR (s. Anhang A.3.). In diesem Fall wird keine Anfangsinitialisierung durchgeführt, und der aktuelle Stackpointerstand (SP) und einige Hilfszellen werden beim Programmeintritt übernommen. Zum anderen kann aber auch ein Sprung in ein Monitorrückkehrprogramm (Marke: MONI) erfolgen. Dabei werden die aktuellen CPU-Werte zunächst in Hilfszellen (virtuelle Anwender-CPU) gerettet. Danach folgt ein Sprung in die Tastaturabfrageschleife. Der Stackpointer wird zuvor auf den Anfang des Monitorstackbereichs gesetzt.

Die Befehlseingabe des Lernsystems wird über zwei Tastenfelder vorgenommen. Das Tastenfeld für die Funktionswahl enthält 15 Tasten und die RESET-Taste. Die Taste RESET wird softwaremäßig nicht erfaßt und hat, wie bereits dargestellt, die zentrale Bedeutung Rücksetzen und Monitorstart. Die anderen 15 Tasten dieses Feldes werden vom Betriebsprogramm in der Monitorbetriebsart erfaßt und bewirken die entsprechenden

Programmverzweigungen. Ihre Bedeutung und Wertigkeit sind in Tafel 8.3.2 zusammengestellt. Da mit den Funktionstasten jeweils Eingabeketten eröffnet werden, ist eine Zweitbelegung der Tasten prinzipiell nicht möglich. Ausnahme hiervon bilden die Tasten „M“ (Wertigkeit 1BH) und „‘“ (Wertigkeit 1EH), da sie eine Spezifizierung von Funktionen vornehmen (z. B. DISP M...; SET A'...). Beide Tasten können somit mit einer Zweitbelegung versehen werden, die beispielsweise Ein-/Ausgabe-Funktionen über die vorhandenen Schnittstellen organisieren. Diese Möglichkeiten werden jedoch durch die Grundaustausstufe des Monitorprogramms nicht ausgenutzt. Das zweite Tastenfeld des Lernsystems enthält ebenfalls 16 Tasten. Es dient zur Dateneingabe und Adressierung. In Tafel 8.3.3 sind Bedeutung und Wertigkeit dieser Tasten zusammengefaßt. Prinzipiell können den Tasten dieses Feldes mehrere Bedeutungen zugewiesen werden, da die Abfrage innerhalb einer durch eine Funktionstaste eröffneten Eingabekette geschieht. Vom Betriebsprogramm wird jeder Taste dieses Feldes als Erstbelegung eine Hexadezimalzahl zugewiesen. Diese Zahl entspricht der jeweiligen Tastenwertigkeit. Die Zweitbelegung

Tafel 8.3.2. Bedeutung der Tasten des Funktionsfelds

Bezeichnung	Wertigkeit	Erste Bedeutung	Zweite Bedeutung
EX	16 (10H)	Funktion EX	—
STORN	17 (11H)	Funktion STORN	—
START	18 (12H)	Funktion START	—
STEP	19 (13H)	Funktion STEP	—
IDM	20 (14H)	Funktion IDM	—
DDM	21 (15H)	Funktion DDM	—
DISP	22 (16H)	Funktion DISP	—
SET	23 (17H)	Funktion SET	—
STORE	24 (18H)	Funktion STORE	—
LOAD	25 (19H)	Funktion LOAD	—
INP	26 (1AH)	Funktion INP	—
M/TPO	27 (1BH)	MEMORY	Funktion TPO
BRK	28 (1CH)	Funktion BRK	—
FILL	29 (1DH)	Funktion FILL	—
'/TPI	30 (1EH)	Alternativregisteradressierung	Funktion TPI

Tafel 8.3.3. Bedeutung der Tasten des Dateneingabefelds

Bezeichnung	Wertigkeit	Erste Bedeutung	Zweite Bedeutung
0/PRG	0 (00H)	Ziffer 0H	Funktion PRG
1/CMP	1 (01H)	Ziffer 1H	Funktion CMP
2/TRF	2 (02H)	Ziffer 2H	Funktion TRF
3/I	3 (03H)	Ziffer 3H	Register I
4/PC	4 (04H)	Ziffer 4H	Register PC
5/SP	5 (05H)	Ziffer 5H	Register SP
6/IY	6 (06H)	Ziffer 6H	Register IY
7/IX	7 (07H)	Ziffer 7H	Register IX
8/H	8 (08H)	Ziffer 8H	Register H
9/L	9 (09H)	Ziffer 9H	Register L
A	10 (0AH)	Ziffer 0AH	Register A
B	11 (0BH)	Ziffer 0BH	Register B
C	12 (0CH)	Ziffer 0CH	Register C
D	13 (0DH)	Ziffer 0DH	Register D
E	14 (0EH)	Ziffer 0EH	Register E
F	15 (0FH)	Ziffer 0FH	Register F

dient zur Registeradressierung (Wertigkeit 03H ... 0FH) bzw. stellt Funktionen dar (Wertigkeit 00H ... 02H). Die in diesem Tastenfeld enthaltenen Funktionstasten sind für die Ansteuerung von PROM-Programmereinheiten vorgesehen. Die vorliegende Ausbaustufe des Monitorbetriebsprogramms beinhaltet jedoch nicht die dazu notwendigen Steuerrouтины. Die Zuordnung zwischen Tastenwertigkeit und der jeweiligen Bedeutung wird bei allen Tasten des Lernsystems durch das Monitorprogramm organisiert. Deshalb können in der Anwenderbetriebsart (user mode) allen softwaremäßig bearbeiteten Tasten auch neue Belegungen zugewiesen werden, die völlig unabhängig von den Monitorbedeutungen sein dürfen.

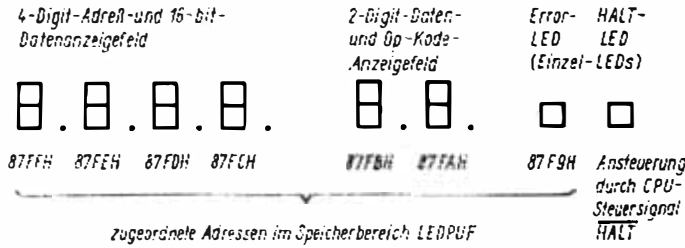


Bild 8.3.1 Speicheradresse Zuordnung der Anzeigeelemente

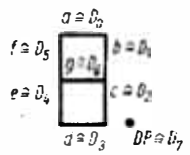


Bild 8.3.2 Datenbitzuordnung zwischen Anzeigesegmenten und Bitwertigkeit der LEDPUF-Speicher

Die Anzeige des Lernsystems wird durch eine sechsstellige Siebensegmentanzeige und zwei Einzelleuchtdioden realisiert. Sie ist in ein vierstelliges Anzeigefeld zur Darstellung von 16-bit-Daten und -Adressen und ein zweistelliges Feld zur Anzeige von 8-bit-Daten und Befehlskoden aufgeteilt. Ein softwaremäßig angesteuertes Einzel-LED dient zur Fehlersignalisation (ERROR). Das HALT-LED wird hardwaremäßig durch das zugehörige CPU-Steuer-signal angesteuert. Im Bild 8.3.1 ist die Zuordnung der Anzeigeelemente zu den Speicherplatzadressen des Monitorprogramms dargestellt (Marke LEDPUF im Programmlisting). Die Ansteuerung der Anzeige ist zeitmultiplex. Dazu ist es erforderlich, daß etwa alle 0,5 ms ein neues Digitsteuersignal und ein neuer Siebensegmentkode an die Anzeigeverstärker gelangt. Die Erzeugung des Zeitrasters sowie der Digit- und Segmentsteuersignale wird softwaremäßig durchgeführt. Das Zeitraster wird durch den CTC-Kanal 1 der Prozessorbaugruppe mit Zeitgeberinterrupt erzeugt. Ein Interrupt dieses Kanals bewirkt den Aufruf einer Bearbeitungsroutine (Marke DISP), in der die Signale für das in der Reihenfolge nächste LED-Tableau erzeugt werden. Die anzuzeigenden Daten sind hierbei im Siebensegmentkode in den LEDPUF-Speicherzellen abgelegt. Die Inhalte dieser Zellen werden durch die Anzeigeroutine nicht verändert. In der Anwenderbetriebsart können in diese an der Marke LEDPUF beginnenden Speicherplätze auch beliebige andere Informationen abgespeichert und somit zur Anzeige gebracht werden. Die Belegung der Datenbits mit den Segmentinformationen zeigt Bild 8.3.2. Die Umkodierung vom Hexadezimalkode auf diese Datenbelegung wird programmäßig durch die an der Marke SEGM beginnende Kodiertabelle unterstützt. Ein Abschalten der Lernsystemanzeige in der Anwenderbetriebsart ist durch das Sperren des CTC-Kanalinterrupts mit nachfolgender Löschung der Anzeige (Marke BLANK) möglich. Diese Operation wird durch das im Monitorprogramm enthaltene Unterprogramm

DISOFF ausgeführt. Das Wiedereinschalten der Anzeige wird durch die Initialisierung des CTC-Kanals 1 erreicht. Diese Unterprogrammoperation ist unter dem Namen DISON enthalten. In der Anwenderbetriebsart ist ein Echtzeitbetrieb nur bei abgeschaltetem Display, also bei gesperrtem Zeitgeberinterrupt, möglich. Dagegen kann bei eingeschalteter Anzeige in der Anwenderbetriebsart nur die Interruptmode IM2 verwendet werden. Bei Verwendung von HALT-Zuständen im Anwenderprogramm muß die mögliche Einwirkung des Anzeiginterrupts beachtet werden [Programmieren einer HALT-Schleife: HALT, JR OFEH].

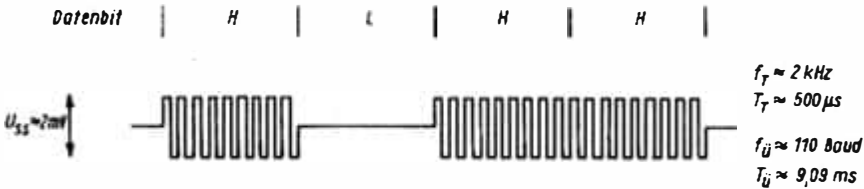


Bild 8.3.3. Zeitverhalten des Kassetteninterface

Als weitere Systemfunktion organisiert das Monitorprogramm den Datenverkehr über die serielle Schnittstelle (USART). Die Monitorfunktionen STORE und LOAD (Tastatur) ermöglichen hierbei den Anschluß eines NF-Kassettenbandgeräts über das vorgesehene Interface der Prozessorbaugruppe. Die Ausgabe der Datenbits aus diesem Kassetteninterface wird durch Amplitudenmodulation (AM, getastete Modulation) einer NF-Trägerschwingung mit etwa  $f_{NF} \approx 2\text{ kHz}$  und  $U_{SS} \approx 2\text{ mV}$  vorgenommen. Damit ergeben sich die im Bild 8.3.3 dargestellten Ausgangssignale. Die Übertragungsgeschwindigkeit für Senden und Empfangen ist auf 110 baud (110 bit/s) festgelegt. Die serielle Ausgabe von Daten (8 bit) über den USART erfolgt im Blockformat in direkter binärer Kodierung. Zusätzlich werden vom USART in jedem Datenblock noch ein Startbit, ein Paritätsbit und zwei Stopbits gesendet. Die entsprechende Signalkonfiguration zeigt Bild 8.3.4. Im Sendebetrieb erfolgt der Start der Datenübertragung durch ein sog. Startbit (L-Pegel). Danach schließen sich die Informationsbits (in diesem Fall acht Datenbits) und das Paritätsprüfbit an. Die asynchrone Übertragung wird durch zwei Stopbits (H-Pegel) beendet. Ohne Sendebetrieb wird aus dem Kassettenbandinterface eine konstante unmodulierte Trägerschwingung (logisch 1) ausgegeben.

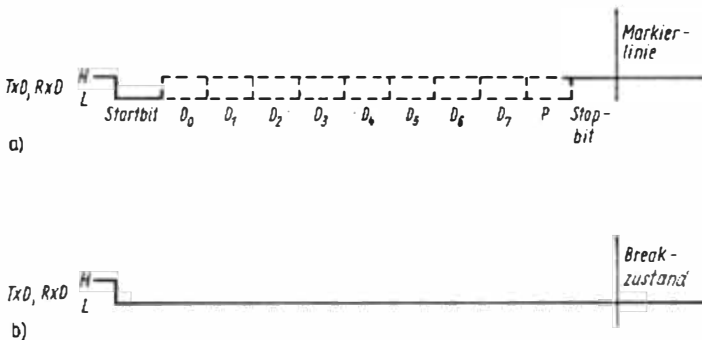


Bild 8.3.4. Signalkonfiguration bei asynchroner serieller Datenübertragung

- a) Datensendebetrieb; b) Senden von Breakzeichen
- TxD, RxD serielle Datenlinie (Ausgang oder Eingang)
- $D_0 \dots D_7$  Datenbits
- P Paritätsbit

### 8.3.2. Aufbau und Arbeitsweise des Monitorprogramms

Das Monitorbetriebsprogramm des Lernsystems enthält u. a. die wesentlichen Programmteile Systeminitialisierung, Tastaturabfrage, Eingabekettenbearbeitung, Funktionsausführung und Anzeigesteuerung. In der Monitorbetriebsart ist der gesamte CPU-Status des Anwenderprogramms in einer im RAM-Speicher lokalisierten virtuellen Anwender-CPU abgelegt. Sämtliche Monitorfunktionen, die in direktem Bezug mit CPU-Registern stehen, arbeiten mit diesen Hilfszellen der virtuellen CPU.

Eine Systeminitialisierung wird immer nach dem Programmstart des Betriebsprogramms ab Adresse 0000H ausgeführt. Durch sie werden alle notwendigen Hilfsspeicherelemente, die virtuelle Anwender-CPU und die peripheren Elemente CTC und USART unabhängig von ihrem momentanen Zustand in einen Grundzustand gebracht. Danach erfolgt der Eintritt in die Tastaturabfrage.

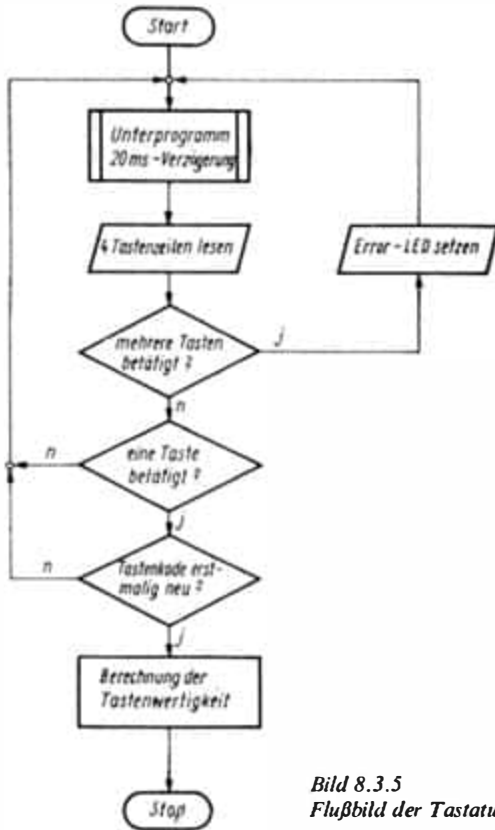


Bild 8.3.5  
Flußbild der Tastaturabfrageroutine

Dieser Programmteil enthält die Programmelemente zur Steuerung der Tastenhardware, eine Prellunterdrückung und den Signifikanztest. Die Tastenabfrage beginnt mit einem Zeitverzögerungsprogramm (etwa 20 ms), das im Zusammenhang mit dem Signifikanztest Eingabefehler (Tastenprellen) ausschalten soll. Es schließt sich dann die Ansteuerung der Tastenhardware und das Lesen der Eingabeports an. Im Signifikanztest geschieht die Kontrolle der Inhalte der vier Tastenzeilen auf korrekten und erstmalig neuen Tastendruck. Die Abarbeitung dieser Programmschleife wird bis zu einem erkannten Tastendruck fortgesetzt. Erfolgt die Erkennung von mehreren gleichzeitig gedrück-

ten Tasten, so wird eine Fehlermeldung über die ERROR-Leuchtdiode ausgegeben und eine neue Tastenabfrage durchgeführt (bis die STORN-Taste erkannt wird). Nach Erkennen einer korrekt gedrückten Taste wird getestet, ob die Taste signifikanterstmalig neu ist. Nur dann sind die Berechnung und Weiterverarbeitung des Tastenkodes möglich. Im Bild 8.3.5 ist das Flußbild der Tastaturabfrageroutine dargestellt.

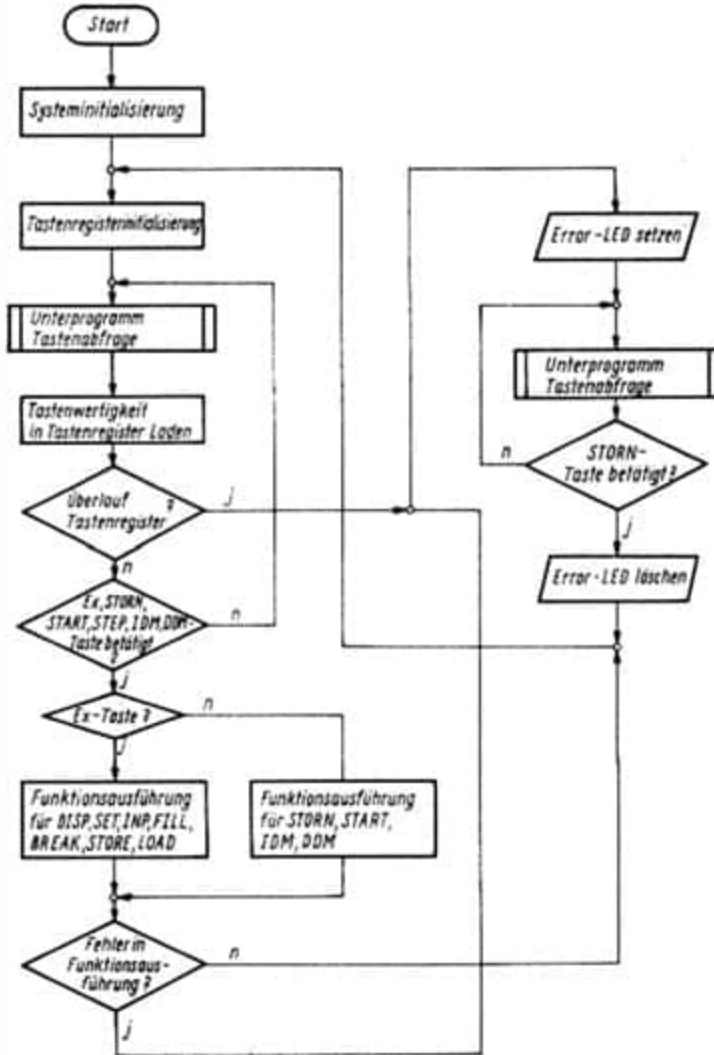


Bild 8.3.6  
Flußbild der Eingaben-  
behandlung des Lernsystems

Die Tastenwertigkeiten der als korrekt betätigt erkannten Tasten werden in der Reihenfolge ihrer Eingaben in ein Tastenregister abgelegt. Dieses Register ist im RAM-Bereich lokalisiert (Hilfzellen) und beginnt bei der Marke TR. Es umfaßt in Abhängigkeit von der geforderten Monitorfunktion bis zu zwölf Tastenwerte (Bytes). Diese Tastenwertigkeiten bilden eine Eingabekette. Bei einem Überlauf wird eine Fehlermeldung über die ERROR-Leuchtdiode und ein erneuter Eintritt in die Tastaturabfrage vorgenommen. Die gesamte Eingabekette muß nach der Fehlerquittierung (STORN-Funktion) erneut eingegeben werden. Eine korrekte Eingabekette enthält einen Funk-

tionskode und die zur Ausführung notwendigen Parameter. Die Reihenfolge der Tastenkode ist hierbei abhängig von der jeweils angewählten Monitorfunktion. Den Abschluß jeder Eingabekette bildet die Ausführungsfunktion (Ex-Taste, execute). Sie bewirkt den entsprechenden Unterprogrammaufruf im Monitorbetriebsprogramm und die damit verbundene Auslösung der eingegebenen Funktion. Während der Abarbeitung der Funktionsroutinen werden die Anzeigedaten aktualisiert. Das entsprechende Flußbild für die Eingabenbehandlung des Lernsystems zeigt zusammenfassend Bild 8.3.6.

### 8.3.3. Kommandobeschreibung

Das Monitorbetriebsprogramm LS880 arbeitet auf der Basis von Eingabeketten, deren Abschluß eine Taste mit auslösender Wirkung bildet. Die Funktionstasten STORN, START, STEP, IDM, DDM weisen eine derartige auslösende Wirkung auf, da die hierdurch aufgerufenen Monitorfunktionen nicht durch Parameter spezifiziert werden müssen. Alle anderen Eingabeketten werden durch den Aufruf der EX-Taste zur Abarbeitung gebracht. Bei der Eingabe von Daten- und Adreßparametern werden vom Monitorprogramm automatisch Führungsnollen erzeugt, falls das Eingabeformat nicht eingehalten wird (underflow). Erfordert eine Monitorfunktion mehrere Parameter, so wird jede Parametereingabe ebenfalls durch die EX-Funktion quittiert. Im folgenden sollen die Funktionen der Monitorkommandos näher erläutert werden.

#### **RESET**; Reset-Funktion (Rücksetzen)

Diese Taste wird hardwaremäßig bearbeitet und bewirkt den Programmstart in der Monitorbetriebsart. Eine Systeminitialisierung wird durchgeführt.

#### **Beispiel**

RESET Anzeige: 0000 00

#### **IDM**; Increment PC, display memory

Mit dieser Tastenfunktion wird der Programmzählerstand PC der virtuellen Anwender-CPU um 1 erhöht und zur Anzeige gebracht. Gleichzeitig erfolgt die Ausgabe des Inhalts der durch diesen neuen PC-Stand adressierten Speicherzelle auf dem Datenanzeigetableau.

#### **Beispiel**

Nach einem Rücksetzvorgang hat der Programmzähler der virtuellen Anwender-CPU den Wert PC = 0000. Die Ausführung der IDM-Funktion hat somit folgende Wirkung:

IDM Anzeige: 0001 C2

#### **DDM**; Decrement PC, display memory

Diese Funktion wirkt ähnlich der IDM-Funktion, ausgenommen, daß der Programmzählerstand PC der Anwender-CPU um 1 erniedrigt wird. Über die Anzeige erfolgt die Ausgabe des neuen PC-Standes und des Inhalts der dadurch adressierten Speicherzelle.

#### **Beispiel**

Die Ausführung der DDM-Funktion nach der IDM-Operation des vorstehenden Beispiels hat die nachstehende Wirkung:

DDM Anzeige: 0000 31

**EX; Execute (Funktionsausführung)**

Diese Funktion schließt prinzipiell die Eingabeketten von Funktionen ab, die eine Parameterspezifizierung aufweisen. Sie bewirkt somit die Funktionsausführung durch das Betriebsprogramm. Bei Memoryfunktionen, die mehrere Eingabeparameter aufweisen oder bei denen die Notwendigkeit zur externen Zeitverzögerung bei der Ausführung besteht, wird die EX-Taste zusätzlich zur Eingabequittierung benutzt.

**STORN; Storno (Eingabestornierung)**

Die Stornofunktion macht alle Eingaben der zuletzt gewählten Monitorfunktion unabhängig von ihrer Vollständigkeit unwirksam und löscht gleichzeitig den ERROR-Anzeigespeicher. Wurde bei einer Eingabekettenbearbeitung oder bei einer Funktionsausführung ein Fehler erkannt (ERROR-LED leuchtet), so werden alle Tasten mit Ausnahme der Tasten STORN und RESET für weitere Eingaben unwirksam. Eine Fortsetzung der Bearbeitung in der Monitorbetriebsart wird nur durch die Quittierung des Fehlers mit der STORN-Taste oder durch eine neue Systeminitialisierung (RESET-Taste) erreicht.

**DISP; Display (Register- und Speicherinhaltsanzeige)**

Mit dieser Funktion kann der Inhalt eines beliebigen Registers der virtuellen Anwender-CPU oder der durch den Programmzählerstand PC der Anwender-CPU adressierten Speicherzelle zur Anzeige gebracht werden. Bei 16-bit-Registern erfolgt die Anzeige über das vierstellige Adreßanzeigetableau. Andernfalls wird die Ausgabe über das zweistellige Datenanzeigefeld ausgeführt. Die Adressierung der Datenquelle geschieht über die Tasten A, F, B, C, D, E, H, L, PC, SP, IY, ', M. Mit der EX-Taste wird die Funktionsausführung bewirkt.

**Beispiele**

Der Programmzähler der virtuellen CPU habe den Stand PC = 300H, das Alternativregister B' = 5CH. Der Aufruf der DISP-Funktion hat dann die folgende Wirkung:

DISP	PC	EX	Anzeige: 0300	00
DISP	B	' EX	Anzeige: 0300	5C
DISP	M	EX	Anzeige: 0300	87

**SET; Setfunktion (Register und Speicherplätze setzen)**

Diese Funktion erlaubt, ein beliebiges Register der virtuellen Anwender-CPU oder die durch den aktuellen Programmzählerstand der CPU adressierte Speicherzelle (RAM-Zelle) auf einen neuen Wert zu setzen. Der neue Wert wird entsprechend dem geforderten Eingabeformat entweder auf dem vierstelligen oder zweistelligen Anzeigefeld ausgegeben. Zur Adressierung der Datensenke dienen wie bei der DISP-Funktion die Tasten A, F, B, C, D, E, H, L, IX, IY, SP, PC, ', M. Danach erwartet das Monitorbetriebsprogramm die Eingabe des neuen Datenwerts. Entsprechend der Spezifizierung der Datensenke dürfen maximal zwei bzw. vier Hexadezimalzahlen eingegeben werden. Die Befehlsausführung wird über die EX-Taste angefordert.

**Beispiel**

Es soll die RAM-Speicherzelle 8400H mit dem Wert 76H geladen werden; danach soll der Stackpointer der virtuellen Anwender-CPU auf den Wert 86FFH gesetzt werden:

SET	PC	8400	EX	Anzeige:	8400	00
SET	M	76	EX	Anzeige:	8400	76
SET	SP	86FF	EX	Anzeige:	86FF	76

**INP; Input (Dateneingabe)**

Die Inputmonitorfunktion gestattet die fortlaufende Eingabe (bytestweise) von Daten in den Schreib/Lese-Speicher des Lernsystems. Nach der Eröffnung der Eingabekette erwartet das Monitorprogramm ein 8-bit-Datenwort und die Ausführung der EX-Funktion zur Eingabequittierung. Der Aufruf der EX-Funktion bewirkt das Ablegen des Datenworts in die durch den aktuellen Anwender-PC-Stand adressierte Speicherzelle (RAM-Speicherbereich ist Voraussetzung). Danach erfolgt die automatische Inkrementierung des Programmzählerstands der virtuellen CPU. Mit diesem PC-Wert kann nun eine weitere Dateneingabe bewirkt werden, die ebenfalls wieder mit der EX-Taste abgeschlossen werden muß. Während jeder dieser Dateneingaben wird die Anzeige des Programmzählerstands über das vierstellige Anzeigefeld und die Ausgabe des zugehörigen Datenwerts über die zweistellige Anzeige vorgenommen. Die Eingabemodus kann durch ein weiteres Betätigen der EX-Taste wieder verlassen werden.

**Beispiele**

Es soll die Programmfolge LD A, 7FH; HALT; JMP 8400H; (Maschinencode: 3EH 7FH, 76H, C3H 00H 84H) in den RAM-Speicher, beginnend ab Adresse 8400H, geladen werden.

SET	PC	8400	EX	Anzeige:	8400	00
INP	3E	EX		Anzeige:	8401	3E
	7F	EX		Anzeige:	8402	7F
	76	EX		Anzeige:	8403	76
	C3	EX		Anzeige:	8404	C3
	0	EX		Anzeige:	8405	00
	84	EX	EX	Anzeige:	8406	84

**FILL; Fillfunktion**

Diese Funktion des Monitorprogramms dient zum automatischen Füllen eines vorzugebenden Speicherbereichs (RAM-Speicherzellen) mit einem konstanten Wert. Diese Monitorfunktion kann verwendet werden, um beispielsweise die korrekte Arbeitsweise des RAM-Speichers zu überprüfen. Andere Anwendungen ergeben sich im Zusammenhang mit der Anwenderprogrammtestung. Hierbei können vom Anwenderprogramm nicht belegte Speicherbereiche mit dem Breakpointcode (Restart 18H: Maschinencode DF) gefüllt werden. Ein fehlerhafter Eintritt des Anwenderprogramms in den gesicherten Speicherbereich würde eine Rückkehr in die Monitorbetriebsart bewirken. Die Eingabe-

kettenbehandlung des FILL-Kommandos erfordert die Spezifizierung der 16-bit-Anfangsadresse, der 16-bit-Endadresse und des einzuschreibenden Datenwerts. Diese Parameter werden jeweils mit Hilfe der EX-Funktion quittiert. Die Funktionsausführung erfolgt nach nochmaligem Betätigen der EX-Taste. Die Parametereingaben werden vom Monitorprogramm durch Anzeige auf dem vierstelligen Tableau quittiert. Nach der korrekten Funktionsausführung wird auf der 16-bit-Anzeige die Bereichsendadresse und auf der 8-bit-Anzeige der Datenwert ausgegeben.

### Beispiel

Der Speicherbereich zwischen 8406H und 85FFH soll mit dem Wert DFH gefüllt werden:

FILL	8406	EX	Anzeige: 8406	00
	85FF	EX	Anzeige: 85FF	00
	DF	EX	Anzeige: <b>85DF</b>	00
	EX		Anzeige: 85FF	DF

### BRK; Breakpoint (Haltepunkt)

Die Unterbrechungsfunktion gestattet das Zurückkehren in die Monitorbetriebsart an einer vorwählbaren Adresse des Anwenderprogramms. Voraussetzung hierfür ist, daß sich das Anwenderprogramm im RAM-Speicher des Lernsystems befindet. Bei der Ausführung der Breakpointfunktion werden das erste Byte des Befehls, bei dem der Anwenderprogrammlauf unterbrochen werden soll, durch den Rückstartbefehl RST 18H ersetzt sowie der Originalbefehlskode und dessen Speicherplatzadresse in Hilfszellen gerettet. Die Ausführung der BRK-Funktion wird durch die Betätigung der EX-Taste ausgelöst. Eine vollzogene Adreßspezifizierung wird auf der 16-bit-Anzeige ausgegeben. Der softwaremäßig eingesetzte RST-Befehl bewirkt in der Anwenderbetriebsart eine Programmunterbrechung an der vorgewählten Adresse. Es wird der Übergang in die Monitorbetriebsart ausgeführt. Dort erfolgt die Abspeicherung der aktuellen CPU-Werte in die Hilfszellen der virtuellen Anwender-CPU. Auf der vierstelligen Anzeige erscheint die Ausgabe der Unterbrechungsadresse. Das Monitorbetriebsprogramm tritt danach in die Tastaturabfrage ein und ermöglicht somit Datenmanipulationen. Bei einem anschließenden Aufruf der START- oder STEP-Funktion geschieht die automatische Löschung des Haltepunkts und die Rückspeicherung des Originalbefehlskodes. Eine Löschung einer gesetzten Unterbrechung kann aber auch durch eine leere BRK-Anweisung vorgenommen werden. Sie ist in jedem Fall die Voraussetzung für die Eingabe einer neuen BRK-Funktion. Zu beachten ist, daß der Befehl, auf den die Breakpointadresse zeigt, noch nicht ausgeführt wird. Er kommt erst durch die erneute Betätigung der START- oder STEP-Taste zur Abarbeitung. Weiterhin darf bei einer noch nicht gelöschten BRK-Anweisung die RESET-Taste nicht betätigt werden, da anderenfalls der Unterbrechungsbefehl (RST 18H) im Anwenderprogramm nicht mehr automatisch durch den Originalbefehlskode (der bei einer Initialisierung verloren geht) ersetzt werden kann.

### Beispiel

Zunächst soll eine evtl. noch anhängige BRK-Funktion gelöscht werden; danach soll ein Breakpoint auf die Speicheradresse 8402H gesetzt werden:

BRK	EX			
BRK	8402	EX	Anzeige: 8402	00

**START**; Startfunktion (Anwenderbetriebsart)

Mit dieser Monitorfunktion kann ein Übergang in die Anwenderbetriebsart organisiert werden. Die Anwenderprogrammbearbeitung wird somit ermöglicht. Nach der Auslösung der Funktionstaste START werden alle Register des Prozessors U880 (ausgenommen das Refreshregister R) mit den Werten der virtuellen Anwender-CPU geladen. Die Anwenderprogrammbearbeitung beginnt somit auf der durch den Anwenderprogrammzählerstand (PC der virtuellen CPU) spezifizierten Speicheradresse. Zur Vorbereitung der START-Operation müssen somit üblicherweise der PC und SP der virtuellen CPU durch das SET-Kommando spezifiziert werden. Ein Laden des Stackpointers ist erforderlich, da der Anzeigeeinterrupt sofort eine Programmverschachtelung bewirken kann. (Die zugehörige ISR benötigt einen Stack zur Programmrückkehr.) Die Rückkehr aus dem Anwenderprogramm in das Monitorprogramm kann durch Anwendung der BRK-Funktion, durch einen entsprechenden programmierten Sprung des Anwenderprogramms oder durch Auslösen eines Rücksetzvorgangs (RESET-Taste) erfolgen. Die STEP-Funktion beeinflusst die Anzeige nicht.

**Beispiel**

Das im Beispiel der INP-Funktion eingegebene kurze Programm soll im Anwendermode gestartet werden:

```

SET SP 8500 EX           Anzeige: 8500 00
SET PC 8400 EX           Anzeige: 8400 00
START
                           Anzeige: HALT-LED 8400 00

```

**STEP**; Stepfunktion (Schrittbetrieb)

Diese Funktion des Monitorprogramms stellt die zweite Möglichkeit des Übergangs in die Anwenderbetriebsart dar. Entsprechend gelten die gleichen Voraussetzungen wie bei der START-Funktion. Der Unterschied besteht darin, daß eine automatische Rückkehr in die Monitorbetriebsart nach Abarbeitung eines Anwenderbefehls erfolgt. Diese Rückkehraktion wird hardwaremäßig durch Ausnutzung der NMI-Linie des Prozessors ausgeführt. Softwaremäßig erfolgt eine Rettung der CPU-Registerinhalte in die Hilfszellen der virtuellen Anwender-CPU. Über die vierstellige 16-bit-Anzeigeeinheit wird die Adresse des bearbeiteten Befehls ausgegeben. Das Monitorprogramm erwartet daraufhin erneute Tasteneingaben (Tastaturroutine), die zu Register- bzw. Speicherplatzmanipulationen oder zur erneuten Einzelschrittauslösung (STEP-Taste) dienen können.

**Beispiel**

Es soll der erste Befehl des im INP-Beispiel eingegebenen Programms bearbeitet werden:

```

SET SP 8500 EX           Anzeige: 8500 00
SET PC 8400 EX           Anzeige: 8400 00
STEP                      Anzeige: 8400 00
DISP A EX                 Anzeige: 8400 7F

```

**STORE**; Storefunktion (Speichern auf Band)

Das Monitorprogramm kann mit der STORE-Funktion einen Datenblock über die serielle Schnittstelle (USART) aussenden. Die Übertragungsrate beträgt hierbei 110 baud. Als Interface können wahlweise das Kassettentonbandinterface, die Stromschleifenschnittstelle oder die Spannungsschnittstelle (RS 232) verwendet werden. Die Programmbearbeitung unterstützt jedoch vorrangig die Anwendung eines NF-Kassettentonbandgeräts zur Datenspeicherung. Nach Aufruf der STORE-Funktion muß der zu übertragende Datenblock durch die Eingabe der 16-bit-Anfangsadresse und der 16-bit-Endadresse spezifiziert werden. Die entsprechenden Adreßstasteneingaben werden jeweils mit der EX-Funktion quittiert. Eine Eingabekontrolle geschieht durch die Ausgabe der beiden Adreßwerte über das vierstellige Anzeigetableau. Nach Betätigen der Quittierung für die Endadresse (EX-Taste) muß das Kassettentonbandgerät in die Aufnahmestellung gebracht werden und eine etwa 5 s dauernde Aufnahme des zu diesem Zeitpunkt ausgesendeten unmodulierten Trägers erfolgen. Diese Aktion dient zur Erleichterung der Rückladung der abgespeicherten Informationen mit Hilfe der LOAD-Funktion in den RAM-Speicher des Lernsystems. Die erneute Betätigung der EX-Taste löst daraufhin den eigentlichen Speichervorgang aus. Programmäßig wird diese Funktionsausführung quittiert, indem auf dem 16-bit-Tableau die Anfangsadresse des Datenblocks angezeigt und auf dem rechten Feld der Datenanzeige das Symbol S (für Store bzw. Senden) ausgegeben wird. Am Ende der Datenblockübertragung wird vom USART für etwa 25 s ein Breakzeichen (L-Pegel, Trägerschwingung unterdrückt) gesendet. Der entsprechende Teil des Tonbands kann zu Identifikationszwecken benutzt werden. Ist die gesamte Übertragung beendet, so erscheint auf dem vierstelligen Anzeigefeld die Adresse des letzten übertragenen Speicherplatzes, und das S-Symbol in der Datenanzeige (zweistelliges Tableau) rückt um eine Stelle nach links. Der Aufnahmeprozess am Kassettentonbandgerät kann beendet werden.

**Beispiel**

Es soll das Beispielprogramm (INP-Befehl) auf Tonband gespeichert werden.

STORE 8400 EX	Anzeige: 8400 00
8405 EX	Anzeige: 8405 00
Beginn der Kassettentonbandaufnahme, nach etwa 5 s:	
EX	Anzeige: 8400 _S
Nach Ende der Übertragung:	Anzeige: 8405 S_
Ende der Aufnahme.	

**LOAD**; Loadfunktion (Rückladen vom Band)

Diese Funktion dient zum Abspeichern seriell empfangener Daten im RAM-Speicher des Lernsystems. Die verwendete Übertragungsrate beträgt wie bei der STORE-Funktion 110 baud. Mit dieser Funktion können die durch die STORE-Funktion abgespeicherten Daten wieder in das Lernsystem rückgespeichert werden. Die Funktion LOAD des Monitorprogramms ist besonders auf die Handhabung eines Kassettentonbandgeräts zugeschnitten. Als Parameter muß lediglich die Anfangsadresse im RAM-Speicher spezifiziert werden. Die Eingabequittierung geschieht durch Betätigen der EX-Taste. Auf der Anzeige (vierstelliges Feld) erscheint die eingegebene Anfangsadresse. Daraufhin ist das Kassettentonbandgerät in Wiedergabestellung zu bringen. Während des etwa 5 s dauernden

unmodulierten Teils der wiedergegebenen Datenaufnahme muß die LOAD-Funktion durch Betätigen der EX-Taste ausgelöst werden. Diese Aktion wird durch Ausgabe des Symbols L auf dem rechten Datenanzeigefeld vom Monitorprogramm bestätigt. Nach beendetem Empfang erscheint auf dem vierstelligen Tableau die Adresse des Speicherplatzes, der das zuletzt empfangene Datenwort enthält; das L-Symbol rückt in der Datenanzeige um eine Stelle nach links. Ein fehlerloser Empfang liegt vor, wenn das ERROR-LED nicht leuchtet und die angezeigte Adresse der erwarteten Endadresse des übertragenen Speicherbereichs entspricht. Ein Leuchten des ERROR-LED zeigt einen erkannten Übertragungsfehler (z. B. Geschwindigkeitsabweichung des Bandgeräts, Qualitätsfehler des Bandes – „drop out“) an.

### Beispiel

Das durch die STORE-Funktion gespeicherte Programm soll auf Adresse 8400H rückgespeichert werden:

LOAD 8400 EX	Anzeige: 8400 00
Wiedergabestellung des Geräts, bei unmoduliertem Trägersignal	
EX	Anzeige: 8400 _L
Ende der Übertragung:	Anzeige: 8405 L_

### 8.3.4. Unterprogrammtechnik

Für die Bedienung der Tastatur und der Anzeige des Lernsystems enthält das Monitorbetriebsprogramm eine Reihe von Unterprogrammen, die ebenfalls vom Nutzer in der Anwenderbetriebsart verwendet werden können. Sie ermöglichen somit eine effektive Anwenderprogrammierung. Im folgenden werden einige dieser allgemein verwendbaren Routinen zusammenfassend dargestellt. Die verwendeten Marken korrespondieren zum Programmlisting im Anhang A.3 und weisen auf die Unterprogrammstartadresse. Die Werte der Marken, Symbole und verwendeten Speicherplatzinhalte können der Referenztafel (cross reference) des Programmlistings entnommen werden. Der Aufruf der Unterprogramme muß im Anwenderprogramm über einen CALL-Befehl organisiert werden.

#### TABFR; Tastaturabfrageroutine

Dieses Unterprogramm enthält alle notwendigen Funktionen für die Ansteuerung der Tastenhardware. Es führt die logische Aufbereitung der Tastensignale einschließlich der Prellunterdrückung und Signifikanzkontrolle aus. Bei Aufruf des Programms werden die Lernsystemtasten so lange abgefragt, bis die korrekte Betätigung einer Taste erkannt wird. Bei mehreren gleichzeitig betätigten Tasten wird die Fehlersignalisation (ERROR-Leuchtdiode) ausgegeben. Die Tastenabfrage wird fortgesetzt. Nach der Rückkehr aus dem Programm befindet sich im Register A des Prozessors die Tastenwertigkeit der betätigten Taste.

- Eintritt: keine Vorbereitung
- Austritt: Tastenwertigkeit in A
- benutzte Register: AF, BC, DE, HL, IX, IY
- benutzte Speicher: TIN, T1A, FEHLER
- benutzte Stackebenen: 3.

**FFSET**; Fehlersignalisationsroutine

Dieses Unterprogramm setzt das Fehlerbit (Bit  $D_7$ ) des Speicherplatzes FEHLER (87F9H). Diese Aktion bewirkt die Aktivierung der ERROR-Leuchtdiode des Lernsystems.

- Eintritt: keine Vorbereitung
- Austritt:  $D_7$  von FEHLER gesetzt ( $D_7 = 1$ )
- benutzte Register: HL.

**FFCLR**; Löschen der Fehlersignalisation

Das Unterprogramm setzt das Fehlerbit des Speicherplatzes FEHLER zurück. Somit wird die ERROR-Leuchtdiode abgeschaltet.

- Eintritt: keine Vorbereitung
- Austritt:  $D_7 = 0$  von FEHLER
- benutzte Register: HL.

**SEGUM**; Siebensegmentkodieroutine

Das Unterprogramm SEGUM wandelt das niederwertigere Halbbyte (Nibble) des Akkumulators in die entsprechende Siebensegmentbelegung um. Als Zeichenvorrat werden die Hexadezimalzahlen 0H ... FH verwendet. Die Siebensegmentkodierung wird in das A-Register eingetragen.

- Eintritt: Hexadezimalzahl in A auf  $A_0 \dots A_3$
- Austritt: Siebensegmentcode in A ( $A_0 \dots A_7$ )
- benutzte Register: AF, DE.

**DISDAT**; Datenanzeigeroutine

Dieses Programm ermöglicht die Anzeige des Inhalts von Register A des Prozessors auf dem zweistelligen Datenanzeigefeld des Lernsystems.

- Eintritt: 8-bit-Wort in A
- Austritt: Siebensegmentcode des H- und L-Nibble des A-Registers in 87FAH und 87FBH
- benutzte Register: AF, BC, DE, IX
- benutzte Speicher: LEDPUF, LEDPUF + 1
- benutzte Stackebenen: 1.

**DISADR**; Adreßanzeigeroutine

Das DISADR-Programm ermöglicht die Ausgabe des Inhalts des Registerpaars HL auf dem vierstelligen Adreßanzeigefeld des Lernsystems. Der Inhalt von H wird auf den beiden linken, der Inhalt von L auf den beiden rechten LED-Tableaus ausgegeben.

- Eintritt: 16-bit-Wort in HL
- Austritt: Siebensegmentcode in 87FCH bis 87FFH
- benutzte Register: AF, BC, DE, IX
- benutzte Speicher: LEDPUF + 2 bis LEDPUF + 5
- benutzte Stackebenen: 1.

**BLANK**; Routine zur Anzeigesperrung

Dieses Unterprogramm sperrt die Segment- und Digittreiber der Anzeigebaugruppe auf Dauer bis zum nächsten Aufruf des Unterprogramms DISON oder bis zur Betätigung der

RESET-Taste (Systeminitialisierung). Die Folge des Programmaufrufs ist, daß sämtliche LED-Anzeigen verlöschen. Diese Aktion ist wiederum Voraussetzung für das Sperren der Anzeigeinterruptbedienung.

- Eintritt: keine Vorbereitung
- Austritt: Hardwaresegmenttreiberspeicher rückgesetzt
- benutzte Register: AF, BC.

#### **DISOFF; Sperrung des Anzeigeinterrupts**

Das Programm DISOFF sperrt die Interruptfreigabe des CTC-Kanals 1 der Prozessorbaugruppe. Die Betriebsart dieses Kanals wird ebenfalls rückgesetzt. Daraufhin erfolgt der automatische Aufruf des Unterprogramms BLANK, um Schäden an der Anzeige zu vermeiden. Folge des Programmaufrufs von DISOFF ist, daß die LED-Anzeige abgeschaltet wird und daß eine anwendergerechte Interruptprogrammierung durchgeführt werden kann.

- Eintritt: keine Vorbereitung
- Austritt: Anzeigeinterrupt und Hardwaresegmenttreiberspeicher rückgesetzt
- benutzte Register: AF, BC.

#### **DISON; Freigabe des Anzeigeinterrupts**

Dieses Unterprogramm initialisiert den CTC-Kanal 1 der Prozessorbaugruppe neu und gibt somit die Anzeigefunktion wieder frei.

- Eintritt: keine Vorbereitung
- Austritt: CTC-Kanal 1 neuinitialisiert für Anzeigefunktion
- benutzte Register: A.

### **8.3.5. Anwendungshinweise**

Eine Erweiterung des Monitorprogramms ist durch die Einbeziehung weiterer Funktionen möglich. Hierbei können die in der vorliegenden Grundausstufe des Monitorprogramms noch nicht definierten Funktionstasten TPI, TPO, PRO, CMP und TRF verwendet werden. Bei einer entsprechenden Spezifizierung können aber auch anderen Tasten weitere Funktionen zugewiesen werden. Es ergeben sich hierdurch Möglichkeiten, die Leistungsfähigkeit des Lernsystems aufzuwerten. Bei der Verwendung von Funktionstasten für Erweiterungszwecke (z. B. PRG, CMP und TRF zum Aufbau einer PROM-Programmerroutine) ist das unterste PROM (0000 ... 00FFH) an den Speicherstellen zu verändern, an denen der Tastenwertvergleich für die entsprechenden Funktionstasten durchgeführt wird. Anstelle der eingesetzten Sprünge auf das Unterprogramm STRNWT sind Sprünge auf die neuen Zieladressen einzutragen. Diese Zieladressen können in den beiden freien PROM-Plätzen der Speicherbaugruppe des Lernsystems liegen.

In der Monitorbetriebsart ist der Prozessor U880 auf die Interruptbetriebsart IM2 programmiert. Die Interrupttabelle beginnt an der Marke INTAB. Sie beinhaltet jedoch nur die Unterprogrammstartadresse für den vektorierten Anzeigeinterrupt. Da INTAB auf dem Speicherplatz 05FAH beginnt, hat das I-Register des Prozessors nach der Systeminitialisierung den Wert  $I = 05H$  (höherwertiger Teil des Interruptpointers). Sollen nun Anwenderprogramme in der Interruptbetriebsart IM2 bearbeitet werden, so muß die Interrupttabelle in den für den Anwender verfügbaren und zugängigen RAM-Bereich ausgelagert werden. Diese Aktion wird durch Setzen des I-Registers auf den höherwertigen

gen Anteil (H-Byte) des RAM-Speicherbereichs, der die neue Tabelle enthält, erreicht (z. B. I = 86H). Damit nun die Anzeigefunktion in der Anwenderbetriebsart erhalten bleibt (notwendig z. B. bei der Programmtestung), muß entweder die Interrupttabelle des CTC-Kanals 1 auf die zugehörige Speicheradresse geladen werden (z. B. beginnend ab 86FA), oder es muß eine Neuinitialisierung des CTC derart vorgenommen werden, daß das Vektorregister des CTC auf die neue Interrupttabelle im RAM zeigt (insbesondere für Kanal 1). In beiden Fällen hat der Anwender danach die Möglichkeit, die Interrupttabellen für die übrigen CTC-Kanäle und die beiden vorhandenen PIO-Kanäle entsprechend seinen Forderungen festzulegen. Die für die Auslagerung der Interrupttabellen notwendigen Operationen können sowohl durch das Anwenderprogramm als auch durch Manipulationen in der Monitorbetriebsart (Tastatureingabe) ausgeführt werden. Eine Variante der Auslagerung soll nachstehend als Beispiel dargestellt werden:

#### Systeminitialisierung

RESET	Anzeige: 0000 00
Setzen des Programmzählers:	
SET PC 86FA EX	Anzeige: 86FA 00
Eingabemodus (Interrupttabelle CTC-Kanal 1):	
INP C1 EX	Anzeige: 86FB C1
05 EX EX	Anzeige: 86FC 05
Setzen des I-Registers der virtuellen CPU	
SET I 86 EX	Anzeige: 86FC 86

Die gleiche Aktion kann im Anwenderprogramm durch die Befehlsfolge LD HL,86FAH; LD M,0C1H; INC HL; LD M,05H; LD A,H; LD I,A; erreicht werden. Bei einem Abbruch des Anwenderprogrammablaufs mit Hilfe der RESET-Taste erfolgt eine Startinitialisierung. Hierbei werden das I-Register und das CTC-Vektorregister wieder auf die monitorinterne Interrupttabelle INTAB programmiert. In allen anderen Fällen der Anwenderprogrammunterbrechung (BRK-Funktion oder Rücksprung in die Monitortastaturabfrage) bleiben die Registerinhalte (I-Register und Vektorregister des CTC) erhalten.

Entsprechend der dargestellten Interruptanzeigefunktion kann das Lernsystem in den anderen Interruptbetriebsarten (IM0, IM1) nur dann betrieben werden, wenn die Anzeige durch das Unterprogramm DISOFF abgeschaltet wird. Eine Anwenderprogrammtestung wird unter diesen Voraussetzungen erschwert, da im Anwenderprogramm die Rückkehr in den Monitor (mit vorhergehendem Aufruf von DISON) organisiert werden muß.

## 8.4. Erweiterungsmöglichkeiten

### 8.4.1. PROM-Programmierbaugruppen

#### 8.4.1.1. Allgemeines

Der Einsatz von PROM-Programmierbaugruppen erhöht insgesamt die Komplexität des Geräts. Durch die damit erweiterten Anwendungsmöglichkeiten erhöht sich die Leistungsfähigkeit des Lernsystems, und weitere Einsatzgebiete können erschlossen werden (z. B. einfache Entwicklungsaufgaben, PROM-Programmierung vor Ort).

Die z. Z. verfügbaren PROM-Typen unterscheiden sich im wesentlichen durch die un-

terschiedliche Speicherkapazität, die Leistungsfähigkeit und vor allem durch die angewendeten Herstellungstechnologien. Daraus ergeben sich für die PROM-Programmierung unterschiedliche Anforderungen sowohl an den Programmierablauf (Zeitverhalten) als auch an die geforderten Spannungsverläufe an den Eingangspins. Der Programmieraufwand ist hierbei bei dem moderneren EPROM U555 wesentlich geringer als bei dem noch in p-Kanal-MOS-Technologie gefertigten PROM/EPROM U551/U552. Dieser Aufwand wird vor allem durch den Hardwareaufwand der zugehörigen Programmierbaugruppen widerspiegelt. Während die Programmierbaugruppe für den U551/U552 aus zwei Karteneinschüben (konstruktiv zu einem Baugruppeneinschub, der drei Lernsystemsteckplätze beansprucht, zusammengefaßt) und einer externen Netzteileneinheit besteht, kommt die Programmiereinrichtung für den U555 mit einer Steckeinheit aus.

Der Einsatz der PROM-Programmiereinheiten erfordert im Lernsystem eine entsprechende Ansteuerungssoftware. Mit diesem Teilprogramm muß einerseits die impuls- und signalmäßige Ansteuerung der Programmierbaugruppen erreicht und andererseits die Datenkommunikation zwischen Lernsystem und zu programmierenden PROM abgesichert werden. Aus diesen Gründen bietet sich die direkte Anbindung dieser Teilprogramme an das Monitorbetriebsprogramm des Lernsystems an. Hierbei können die bereits reservierten Tastenfunktionen PRG, TRF, CMP verwendet werden. Mit Hilfe dieser Tasten müssen die Funktionen *Programmieren* (PRG: Datentransport aus dem Lernsystem in das zu programmierende PROM; notwendige Parameter sind die Startadresse im Lernsystem-RAM-Speicher und der PROM-Typ beim Einsatz mehrerer Programmierbaugruppen bzw. -routinen), *Übertragen* (TRF, transfer: Datentransport aus dem PROM in den RAM-Speicher des Lernsystems; notwendige Parameter sind wiederum die Startadresse im RAM-Speicher und der PROM-Typ bei Vorhandensein mehrerer Ansteuerroutinen) und *Vergleichen* (CMP, compare: Vergleich des PROM-Inhalts mit einem Datenblock im RAM-Speicher des Lernsystems; Parameter sind wiederum Startadresse und PROM-Typ) ausgeführt werden.

Des weiteren ist zu beachten, daß beim Einsatz der Programmierbaugruppe für den U555 ebenfalls eine Erweiterung der Speicherbaugruppe des Lernsystems sinnvoll ist. Einerseits weist ein zu programmierendes EPROM U555 bereits eine Speicherkapazität von 1 Kbyte auf, während die für den Anwender frei verfügbare RAM-Kapazität in der Grundkonfiguration des Lernsystems nur 942 byte beträgt. Andererseits hat der Anwender in der bestehenden Ausbaustufe des Lernsystems nicht die Möglichkeit, die programmierten EPROMS U555 einzusetzen und evtl. Programmtestungen auszuführen. Aus diesen Gesichtspunkten ergibt sich, daß bei Anwendung der Festwertspeicher U555 die Nachrüstung des Lernsystems mit einer Baugruppe FPS2.RR2 (vor allem im Zusammenhang mit der PROM-Programmierung) sinnvoll ist. Der Einsatz dieser Baugruppe im Lernsystem setzt jedoch die Erweiterung bzw. Nachrüstung der entsprechenden Stromversorgungsbaugruppe voraus.

#### 8.4.1.2. Programmierbaugruppe für U551/U552

Die PROM-Programmierbaugruppe U551/U552 ist als Baugruppeneinschub konzipiert. Hierbei sind zwei Leiterkarten und ein Kühlkörper mechanisch verbunden. Sie bilden eine kompakte Steckeinheit. Beim Einsatz im Lernsystem werden drei zusammenhängende Steckplätze belegt. Die Verbindung zur gedruckten Rückverdrahtung des Lernsystems wird über einen 58poligen direkten Steckverbinder der einen Leiterkarteneinheit des Programmierbaugruppeneinschubs realisiert. An der Rückseite der Programmierbaugruppe ist eine 24polige Schwenkhebelbefassung eingesetzt, die zur Aufnahme des zu programmierenden

PROM vom Typ U551/U552 dient. Über diese Fassung können ebenfalls ROM-Bau-  
elemente vom Typ U501 ausgelesen werden. Als weitere Bedienelemente sind an der Ein-  
schubrückseite noch eine Leuchtdiode, ein Taster und eine Anschlußbuchse angebracht.  
Über die Anschlußbuchse erfolgt die Zuführung von zwei erdfreien Wechselspannungen,  
die in der Baugruppe zur Spannungsversorgung benötigt werden. Die Taste dient zur Zu-  
schaltung der Programmierspannung an das PROM-Element. Sie stellt somit einen zu-  
sätzlichen Schutz vor Zerstörung des Speicherelements durch Überspannung dar. Über  
die Leuchtdiode wird das Vorhandensein der Programmierspannung angezeigt.

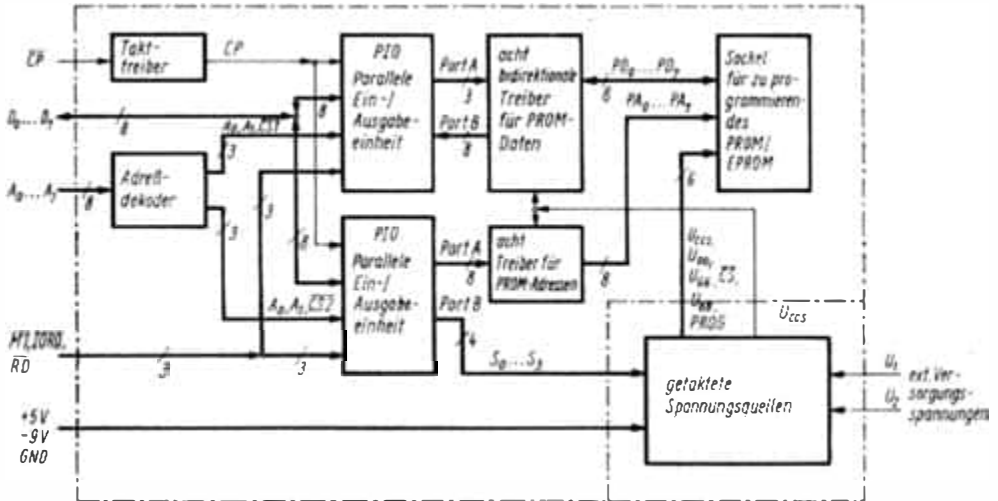


Bild 8.4.1. Blockschaltbild der PROM-Programmierbaugruppe für die Typen U551/U552

Im Bild 8.4.1 ist das Blockschaltbild der PROM-Programmierbaugruppe darge-  
stellt. Die Baugruppe enthält die Funktionsgruppen Impulsaufbereitung (Steckeinheit  
FPS2.PR2), eine externe Spannungsversorgung (Wechselspannungen  $U_1$  und  $U_2$ ) und die  
Ansteuereinheit (Steckeinheit FPS2.PR1). Die Funktionsgruppe zur Impulsaufbereitung  
dient zur Erzeugung der zum Programmiervorgang der PROM-Elemente U551/U552 be-  
nötigten Spannungs-Zeit-Verläufe an den Spannungs- und Steuereingängen des zu pro-  
grammierenden Schaltkreises. Die Zeitsteuerung wird hierbei abhängig von den pro-  
grammäßig erzeugten Statussignalen  $S_0 \dots S_3$  der Ansteuereinheit vorgenommen. Die  
Funktionsgruppe enthält eine Spannungsauflageeinheit, eine Gleichspannungs-  
regelstrecke und verschiedene Schalt- und Umschalteneinheiten, die abhängig von den soft-  
waremäßig erzeugten Ansteuersignalen die geforderten Impulsformen erzeugen. Die  
Spannungsaufbereitungseinheit dieser Funktionsgruppe nimmt eine Spannungsverdopp-  
lung der vom externen Netzteil bereitgestellten Wechselspannung  $U_1$  vor. Damit wird  
erreicht, daß außerhalb der Programmierbaugruppe keine unzulässig hohen Berührungs-  
spannungen anliegen. Der schaltungstechnische und konstruktive Aufwand zur Errei-  
chung der notwendigen Schutzgüte wird dadurch vermindert. Das eingesetzte Relais  
bewirkt in diesem Zusammenhang, daß die Zuführung der externen Wechselspannungen  
nur bei Vorhandensein der  $-9\text{-V}$ -Betriebsspannung erfolgt, also nur, wenn sich der Ein-  
schub im Lernsystem befindet. Die in dieser Funktionseinheit aufbereitete unregelmäßige  
Gleichspannung ( $U_G \approx 80\text{ V}$ ) wird in der nachfolgenden Regelstrecke auf etwa  $48\text{ V}$   
ausgeregelt. Hierzu ist ein integrierter Spannungsregler MAA 723 eingesetzt. An diese  
Funktionseinheit schließt sich eine Spannungsumschalteneinheit an, die abhängig vom

Statussignal  $S_1$  am Ausgang eine Umschaltung von +5 auf +48 V realisiert. Die Ausgangsspannung der Umschaltseinheit entspricht bei der entsprechenden Ansteuerung der Programmierspannung  $U_{CCS}$ . Diese Spannung wird durch eine Crow-bare-Sicherung (Thyristor Th1) vor Überspannungsimpulsen geschützt. Abhängig von dem Zustand der Programmierspannung  $U_{CCS}$  werden mit den anderen Transistorschaltstufen die Spannungsverläufe an den PROM-Pins  $U_{DD}$ ,  $U_{BB}$ ,  $U_{GG}$ ,  $U_{CS}$  und PROG erzeugt. Die elektrische Verbindung mit der Ansteuersteckeinheit erfolgt über Flachbandkabel.

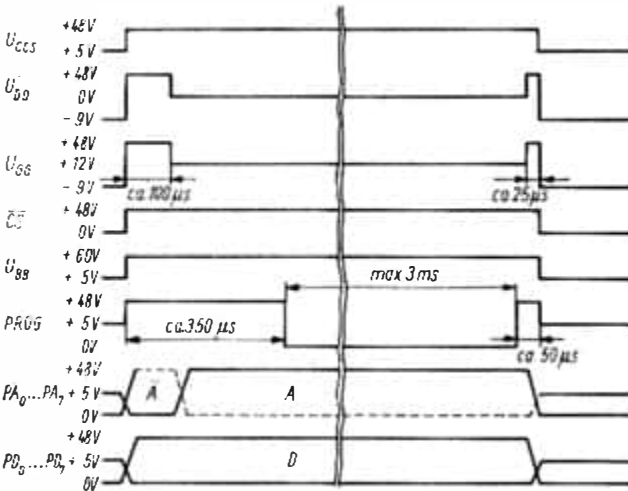


Bild 8.4.2. Zeitverläufe an der Programmierbaugruppe nach Bild 8.4.1.

- $U_{CCS}$  Spannungsverlauf am Pin  $U_{CC}$  des PROM während des Programmierzyklus
- $U_{GG}$ ,  $U_{DD}$ ,  $U_{BB}$ , PROG,  $\overline{CS}$  Spannungsverläufe an den zugehörigen PROM-Pins
- $PA_0 \dots PA_7$  Spannungsverläufe an den Adreßlinien des zu programmierenden PROM (A invertierte Speicherzellenadresse; A nichtinvertierte, direkte Zellenadresse)
- $PD_0 \dots PD_7$  Spannungsverläufe an den Datenlinien des PROM (D einzuschreibende Daten für adressierte Speicherzelle)

Die Funktionsgruppe Ansteuereinheit organisiert über zwei parallele Ein-/Ausgabe-Einheiten U855 (PIO) einerseits den Dialog zwischen Lernsystem und Programmier-PROM (Daten- und Adreßlinien) und andererseits die Bereitstellung der Ansteuersignale für die Transistorschaltstufen. Sie enthält einen 58poligen direkten Steckverbinder, über den der Informationsaustausch zum Lernsystem stattfindet. Auf der Rückseite der Steck-einheit ist die 24polige Schwenkhebelfassung angeordnet, die zur Aufnahme des PROM dient. Die Funktionsgruppe enthält systemseitig neben den PIO noch einen Adreßdeko-der, der eine feste Zuordnung der Baugruppenadresse vornimmt. Die Ansteuerung der PROM-Datenlinien erfolgt durch die beiden Kanalbuse der ersten PIO (Kanal A Daten-eingabe, Lesen des PROM; Kanal B Datenausgabe, Programmieren). Hierbei sind bidirektionale Treiberstufen zwischengeschaltet, die im Programmierzustand eine Pegel-umsetzung des H-Pegels vornehmen. Der Kanal B der zweiten PIO stellt die acht Adreß-informationen für das PROM bereit. Hier sind ebenfalls Pegelwandlerstufen zwischenge-schaltet. Da die eingesetzten Treiber- und Wandlerstufen z. T. invertierend wirken, sind entsprechende Maßnahmen bei der Erstellung der Programmerroutine zu beachten. Über Kanal A der zweiten PIO werden die vier Ansteuersignale zur Impulsaufbereitung er-zeugt. Im Bild 8.4.2 sind zusammenfassend die zur Programmierung notwendigen Zeit-verläufe an den PROM-Pins dargestellt.

### 8.4.1.3. Programmierbaugruppe für U555

Die PROM-Programmierbaugruppe U555 ist als Steckeinheit ausgeführt. Sie enthält einen 58poligen direkten Steckverbinder, über den der Informationsaustausch zwischen Lernsystem und Programmierbaugruppe stattfindet. An der Rückseite der Steckeinheit befindet sich eine 24polige Schwenkhebelfassung, die zur Aufnahme des zu programmierenden PROM dient. Des Weiteren sind dort ein Taster zum Zuschalten der Programmierspannung  $U_{PR} = 27\text{ V}$  und eine Leuchtdiode zum Anzeigen des Programmiervorgangs angebracht. Alle Bedienungselemente der Programmierbaugruppe U555 sind im gesteckten Zustand der Steckeinheit von der Rückseite des Lernsystems erreichbar.

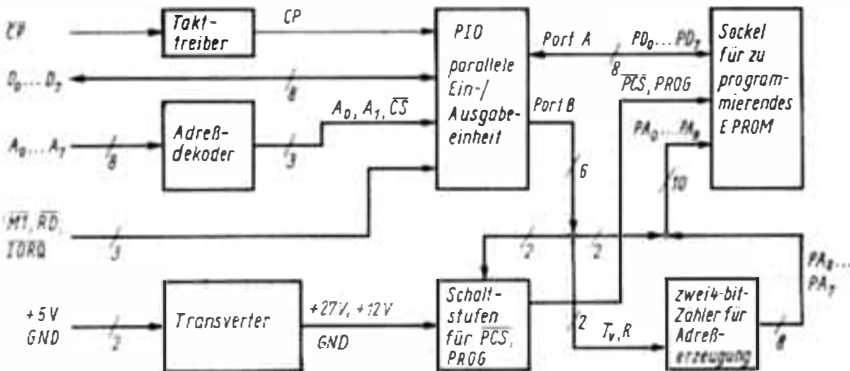


Bild 8.4.3. Blockschaltbild der PROM-Programmierbaugruppe für den Typ U555

Im Bild 8.4.3 ist das Blockschaltbild der Baugruppe dargestellt. Systemseitig sind in der Programmierbaugruppe die Funktionseinheiten parallele Ein-/Ausgabe-Einheit U855 (PIO) und ein Adreßdeko­der eingesetzt. Der Kanal A dient zur Ein- und Ausgabe der EPROM-Dateninformationen. Aufgrund der vollständigen TTL-Kompatibilität der Datenlinien des Festwertspeicherbauelements U555 sowohl beim Lesezugriff als auch im Programmierzustand ist der Einsatz von Buspuffern nicht notwendig. Über zwei Ausgangslinien des PIO-Kanals B erfolgt die impulsmäßige Ansteuerung (Signale PROG und  $\overline{CS}$ ) des zu programmierenden EPROM U555 im Programmierbetrieb. Zwei weitere Ausgangslinien dieses Kanals dienen zur Ansteuerung der Adreßansteuerungslogik des EPROM. Der Adreßdeko­der der Baugruppe nimmt wiederum eine feste Zuordnung der Baugruppenadresse vor. Wegen der Einsatzspezifik der Programmierbaugruppe (vor-rangig Lernsystem) wurde auf bidirektionale Datenbustreiber systemseitig verzichtet.

In der PROM-Programmierbaugruppe sind außerdem die Funktionseinheiten Adreßerzeugungsl­ogik, Spannungsversorgungseinheit und Impulsaufbereitung enthalten. Die Adreßerzeugungsl­ogik bereitet die Adreßinformationen für den EPROM U555 auf. Dazu enthält sie zwei integrierte Zähler D193, deren takt- und ansteuermäßige (RESET-Signal) Versorgung von der PIO übernommen wird. Die beiden höherwertigen Adreßlinien des EPROM werden direkt von der PIO angesteuert. Diese Variante der Adreßerzeugung ist hardware- und softwaremäßig günstig, da einerseits der Schaltungsaufwand gering ist (im Vergleich zum Einsatz einer weiteren PIO) und andererseits die Speicherzellen des U555 zyklisch nacheinander programmiert werden müssen. Die Spannungsversorgungseinheit enthält einen Transverter, der die zum Programmiervorgang benötigte Gleichspannung von  $U_{PR} = 27\text{ V}$  bereitstellt. Diese Programmierspannung wird in der nachfolgenden Impulsaufbereitungseinheit zur Erzeugung der für die Programmierung gefor-

dernten Spannungs-Zeit-Verläufe an den EPROM-Pins PROG und  $\overline{CS}$  benötigt. Die entsprechende zeitliche Ansteuerung erfolgt durch die PIO. Im Bild 8.4.4 ist das Zeitverhalten in der Baugruppe im Programmierzustand enthalten.

Beim Einsatz der PROM-Programmierbaugruppe U555 im Lernsystem ist zu beachten, daß die +12-V-Versorgungsspannung im Lernsystem nachgerüstet werden muß. Die Lernsystemrückverdrahtung enthält entsprechende Vorbereitungen.

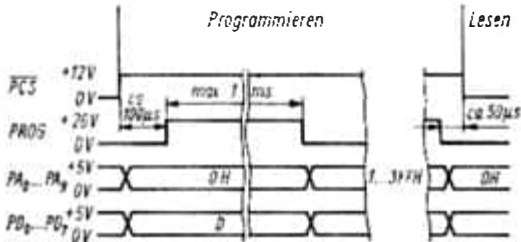


Bild 8.4.4. Zeitverläufe an der Baugruppe nach Bild 8.4.3

$\overline{PCS}$	Spannungsverlauf am EPROM-Pin $\overline{CS}$ während des Programmiervorgangs
PROG	Spannungsverlauf am zugehörigen EPROM-Pin
$PA_0 \dots PA_9$	Spannungsverlauf an den Adreßlinien des zu programmierenden EPROM (0 ... 3FFH die Speicherzellen müssen nacheinander zyklisch programmiert werden)
$PD_0 \dots PD_7$	Spannungsverlauf an den Datenlinien des EPROM (D nichtinvertierte, einzuschreibende Datenwörter)

## 8.4.2. FPS2-Baugruppen

Der Einsatz von FPS2-Baugruppen im Lernsystem kann zum einen zur Erweiterung der Speicherkapazitäten von RAM- und ROM-Einheiten und zum anderen zur Nutzbarmachung verschiedener Ein-/Ausgabe-Schnittstellen dienen. Aufgrund der weitgehenden signalmäßigen Buskompatibilität zwischen Lernsystem und dem FPS2-Leiterkartensatz können FPS2-Speicherbaugruppen direkt in die freien Steckplätze eingesetzt werden. Die CMOS-Speicherbaugruppe FPS2.RAM3 bietet für die Lernsystemanwendung wegen des wahlweisen RAM-Schreibschutzes und der Vorkehrung gegen Spannungsausfälle besonders Vorteile für die Programmentwicklung (als Programmspeicher verwendbar). Die Anwendung der Baugruppen FPS2.RR2 und FPS2.RAM5 erfordert die Bereitstellung einer +12-V-Versorgungsspannung. Eine entsprechende Spannungsversorgungseinheit muß dann extern angeordnet bzw. in die Netzteilbaugruppe des Lernsystems integriert werden. Der Einsatz der relativ leistungsfähigen FPS2-Speicherbaugruppen kann z. B. im Lernsystem die Grundlage für die Anwendung höherer Programmiersprachen bieten (PROM- oder RAM-residente Software). Er kann aber auch die Voraussetzung für die eigenständige Nutzung der Lernsystemkonfiguration im Zusammenhang mit der Verwendung weiterer FPS2-Baugruppen bieten.

Derartige eigenständige Lösungen mit dem Lernsystem können frei programmierbare Einheiten sein, die die Vorteile der Gerätekonfiguration (Tastatur, Anzeige, selbständiges Gerät, eigene Stromversorgung) mit den Leistungsparametern der FPS2-Ein-/Ausgabe-Einheiten verbinden. Anwendungsfälle sind Prototyplösungen u. dgl.

Beim Einsatz der Ein-/Ausgabe-Baugruppen des FPS2-Leiterkartensystems in den Steckplätzen des Lernsystems sind wiederum evtl. Modifikationen der Spannungsversorgung zu beachten. Desgleichen können entsprechende Veränderungen in den  $\overline{BAI}$ - $\overline{BAO}$ -Signalverkettungen der Lernsystemrückverdrahtungsbaugruppe notwendig werden. Ansonsten besteht eine signal- und busmäßige Verträglichkeit der Steckeinheiten.

### 8.4.3. Anwendung höherer Programmiersprachen

Die Leistungsfähigkeit des Lernsystems wird wesentlich durch die vorhandene Betriebssoftware mitbestimmt. Diese Komponente der Leistungsfähigkeit kann einerseits durch Verbesserung des vorhandenen Monitorbetriebsprogramms und andererseits durch Einsatz weiterer Standardprogramm Pakete und Anwendung höherer Programmiersprachen beeinflusst werden. Die Erweiterung des vorhandenen Monitorprogramms ist vor allem hinsichtlich des Einsatzes von PROM-Programmerroutinen, der Verbesserung der Programmunterstützung bei der Anwenderprogrammtestung und der Einbeziehung weiterer Tastenfunktionen sinnvoll.

Beim Einsatz weiterer umfangreicher Betriebssoftware muß abgeschätzt werden, ob als Speichermedium die RAM- oder die ROM-Einheiten der Lernsystemspeicherbaugruppen verwendet werden sollen. Die Anwendung PROM-residenter Software bietet Vorteile bei der Handhabung (schneller Programmzugriff, kein Laden erforderlich). Sie belegt aber große Speicherbereiche und macht den Einsatz mehrerer PROM-Speicherbaugruppen (FPS2.RR2) im Lernsystem erforderlich (zumindest, wenn mehrere Programmpakete gleichzeitig benutzt werden sollen). Alternativ können Programmpakete auch im Schreib/Lese-Speicher untergebracht werden. Als Speicherbaugruppe bietet sich hierfür vor allem die dynamische RAM-Steckeinheit FPS2.RAM5 an, die 32 Kbyte Speicherkapazität aufweist. Vor der Inbetriebnahme solcher Programme müssen diese zunächst in den RAM-Bereich geladen werden. Im Fall des Lernsystems bietet sich hierfür insbesondere das Kassettenbandinterface an, vorausgesetzt, daß Tonbandkassetten als feste Speichermedien verwendet werden. Der eigentliche Programmstart kann entweder über Tastenfunktionen des Monitorbetriebsprogramms (insbesondere bei ROM-residenter Software) oder durch einen Anwenderprogrammstart (vor allem bei RAM-residenten Programmen) erfolgen.

Standardprogrammpakete, die die Leistungsfähigkeit und Handhabbarkeit des Lernsystems verbessern, sind vorrangig Übersetzungsprogramme (Assembler) und Textverarbeitungsprogramme (Editor). Assemblierungsprogramme dienen zur Umsetzung von Quellprogrammen (geschrieben im U880-Mnemonic- bzw. Quellcode) in Maschinenprogramme (Objektprogramme im U880-Maschinencode). Zusätzlich kann vom Assembler ein Programmlisting erzeugt werden (s. auch Anhang A.3), das zur Dokumentation und Programmtestung benötigt wird. Entsprechend der Ausbaustufe des Assemblers kann eine zeilenweise, blockbezogene oder insgesamt verschiebliche Übersetzung erfolgen. Diese Ausbaustufen sind dadurch gekennzeichnet, welche Möglichkeiten der Assembler hinsichtlich der Berechnung von Sprungadressen (relative und absolute Sprünge), der Einbeziehung symbolischer Vereinbarungen (Marken, logische Ausdrücke) und der Verwendung von Makrobefehlen (mehrmalige Verarbeitung ähnlicher Teilprogramme mit Spezifizierung von variablen Größen) aufweist. Bei einer verschieblichen Übersetzung durch den Assembler (gekennzeichnet durch relative, auf die Programmstartadresse 0000H bezogene Sprungadressen) ist ein zusätzliches Bindeprogramm (Linker) erforderlich, das den verschieblichen Objektprogrammen eine feste Anfangsadresse zuweist.

Die Editierungsprogramme dienen zur Unterstützung der Quellprogrammerarbeitung. Sie nehmen eine Zeilenorganisation vor, lassen die Einbeziehung von Kommentartexten zu und bieten Möglichkeiten zur Veränderung des bestehenden Programmtextes.

Entsprechend den vorliegenden Ausbaustufen und Möglichkeiten derartiger Standardprogramme ist es günstig, die Kommunikationsmöglichkeiten des Lernsystems zu verbessern. Hierzu kann über die vorhandene serielle Schnittstelle entweder ein Fernschreiber (TTY-Schnittstelle) oder eine Bildschirmbedieneinheit (RS232-Schnittstelle) angeschlos-

sen werden. Die hierzu notwendigen Ansteuerroutinen können dann entweder im Betriebsprogramm oder jeweils in den Programmpaketen eingefügt werden.

Weitere softwaremäßige Möglichkeiten zur Erhöhung der Leistungsfähigkeit des Lernsystems bestehen in der Anwendung höherer Programmiersprachen (vorrangig BASIC, FORTRAN, COBOL, PASCAL). Ihre Einsatzmöglichkeiten sind hauptsächlich vom Anwendungszweck abhängig. Varianten hierzu sind Berechnungsaufgaben in Verbindung mit der Lernsystemkonfiguration (BASIC), Programmerarbeitung für frei programmierbare Steuerungen und Mikrorechner (FORTRAN, Varianten von PL 1).

Prinzipiell kann bei derartigen Softwarepaketen zwischen Interpreterprogrammen (Interpreter) und Übersetzungsprogrammen (Compiler) unterschieden werden. Die Interpreter nehmen die Umsetzung der jeweiligen Anweisungen (z. B. BASIC-Befehle bei BASIC-Interpreter) in die Maschinensprache des Prozessors U880 vor. Die eigentliche Programmerstellung erfolgt aber in der höheren Programmiersprache (in diesem Fall in BASIC). Das entstandene Programm bleibt im entsprechenden Befehlsformat erhalten. Ein Anwenderprogrammmlauf ist nur mit Hilfe des Interpreterprogramms möglich. Bei Compilerprogrammen erfolgt die Programmerstellung wiederum in der höheren Sprache. Das entstehende Anwenderprogramm liegt aber im U880-Maschinenformat vor. Bei einem Anwenderprogrammmlauf ist somit nur das Anwenderprogramm erforderlich.

Hinsichtlich der Softwareerstellung bietet die Anwendung höherer Programmiersprachen bezüglich der problemorientierten Umsetzung von Algorithmen Vorteile. Zum einen sind die Programmiertechniken bei diesen Sprachen bereits bekannt, und eine Einarbeitung in den U880-Befehlssatz kann entfallen; zum anderen sind weniger Befehlschritte zur Beschreibung von Anwenderproblemen notwendig, und die Programmerstellung wird beschleunigt. Insgesamt werden somit die Kosten auf der Seite der Programmherstellung und die problembezogene Umsetzungszeit verringert. Diese Gesichtspunkte sind vor allem bei der Anwendung von Steuerungen und Rechnern in Anlagen und Maschinen bei geringen Stückzahlen zu beachten.

Jedoch erhöht sich der Speicherbedarf derartig erstellter Programme gegenüber den im Assemblerformat erarbeiteten und optimierten Programmen i. allg. erheblich. Der typische Speicherbedarf eines BASIC-Interpreterprogramms beträgt z. B. je nach Ausbaustufe zwischen 4 und 16 Kbyte. Die Anwendung dieser Programmiertechniken ist somit von der Verfügbarkeit großer Speicherbauelemente abhängig.

## 9. Begriffserklärung

Dieser Abschnitt soll eine zusammenfassende Erläuterung der wichtigsten Fachbegriffe der Mikroprozessortechnik geben. Die Auswahl der Stichwörter und Abkürzungen erfolgte nach dem Gesichtspunkt der Gebräuchlichkeit der Fachwörter im deutschen Sprachraum. Hierbei wird bei Begriffen aus dem Englischen grundsätzlich eine deutsche Übersetzung in Klammern angegeben. Bei deutschen oder eingedeutschten Termini wird der englische Begriff in Klammern angegeben, soweit dies sinnvoll erschien.

**access** (Zugriff) – Möglichkeit des Ansprechens und Lesens bei Speicherzellen

**access time** (Zugriffszeit) – Zeitintervall bei Speichern zwischen dem Anlegen der Adresse und der Verfügbarkeit der Daten

**Adresse** (address) – Kode zur Kennzeichnung einer Datenquelle oder -senke, z. B. einer Speicherzelle oder eines peripheren Elements

**Adreßbus** (address bus) – Sammelleitung für das Übertragen von Adressen, i. allg. zwischen CPU als Sender und Speicher sowie Peripherieelementen als Empfänger

**ADU** (Analog/Digital-Umsetzer; ADC analog to digital converter) – A/D-Wandler, dient zur Umsetzung analoger Eingangsgrößen in einen digitalen Kode

**Akkumulator** (accumulator) – Register, das in enger Verbindung mit dem Rechenwerk (s. ALU) eines Prozessors steht und die Möglichkeit zur Ergebnisausgabe bei den meisten arithmetischen und logischen Befehlen besitzt

**ALGOL** (ALGOritmic Language) – höhere Programmiersprache für mathematische und technisch-wissenschaftliche Probleme, wird jedoch i. allg. nicht bei Programmerstellung für Mikrorechner angewendet

**Algorithmus** (algorithm) – Rechenvorschrift, die ein Problem in endlich vielen Schritten löst bzw. darstellt

**alphanumerisch** – Begriff zur Kennzeichnung eines Zeichenvorrats mit Ziffern, Buchstaben und Sonderzeichen

**ALU** (Arithmetic Logic Unit; ALE Arithmetik-Logik-Einheit) – Rechenwerk eines Prozessors, in dem arithmetische, logische und Verschiebeoperationen ausgeführt werden

**Analog/Digital-Wandler** – siehe ADU

**ASCII** (American Standard Code for Information Interchange, amerikanischer Standardkode für den Informationsaustausch) – international gebräuchlicher 7-bit-Kode, der vielfach zur rechnerinternen Darstellung von alphanumerischen Zeichen und zur Datenübertragung verwendet wird; er beinhaltet 96 darstellbare Zeichen (Ziffern, Kleinbuchstaben, Großbuchstaben, Sonderzeichen) und 32 Steueranweisungen, z. B. Wagenrücklauf, Zeichenvorschub für die Datenübertragung

**Assembler** (assembler) – 1. Übersetzungsprogramm, das in Assemblersprache geschriebene Quellprogramme in die Maschinensprache übersetzt; Syntaxfehler können hierbei gekennzeichnet werden; 2. siehe Assemblersprache

**Assemblersprache** (assembly language) – enthält prinzipiell den gleichen Befehlsvorrat wie die Maschinensprache eines Rechners; für die Befehle werden anstelle der Binärwörter der Maschinensprache gut einprägsame Abkürzungen (sog. Mnemoniks) verwendet; über diesen Befehlsvorrat hinaus können symbolische Vereinbarungen und Pseudobefehle (z. B. logische Entscheidungen, Makrobefehle), die vom zugehörigen Übersetzungsprogramm (Assembler) interpretiert werden, eingefügt werden

**asynchron** (asynchronous, taktunabhängig) – 1. ohne Takt arbeitend; 2. mit unterschiedlichem Grundtakt arbeitende Einheiten, Informationsaustausch über Pufferspeicher; 3. bei asynchroner Datenübertragung liegen die Binärzeichen einer Übertragungsfolge in einem festen Zeitraster; der Anfangszeitpunkt der Übertragung ist jedoch willkürlich

**BASIC** (Beginners All purpose Symbolic Instruction Code) – einfach zu erlernende höhere Programmiersprache, wird i. allg. bei Mikrorechnern über Interpreterprogramme genutzt

**baud** (Baud) – Einheit der Schrittgeschwindigkeit  $1 \text{ baud} = 1/s$ , bei serieller Datenübertragung gilt i. allg.:  $1 \text{ baud} = 1 \text{ bit/s}$

**baud rate** (Baudrate) – Schrittgeschwindigkeit

**BCD** (Binary Coded Decimal, binär bzw. dual verschlüsselte Dezimalziffer) – abweichend von der normalerweise im Rechner (ALU) verwendeten dualen Zahlendarstellung wird zur Vereinfachung arithmetischer Operationen jeder Ziffer einer Dezimalzahl eine 4-bit-Dualzahl zugeordnet

**Befehl** Rechneranweisung

**Befehlskode** siehe op-code

**Befehlszähler** siehe program counter

**Betriebssystem** siehe Monitor

**bidirectional** (bidirektional) – zeitmultiplexer Zweirichtungsbetrieb für Signale

**bit** (binary digit; Bit, bit kleinste binäre Informationseinheit) – 1. Bit: Binärzeichen, Ziffer im Dualsystem (0 oder 1 bzw. falsch oder wahr); 2. bit: Einheit der Binärenentscheidung

**borrow** (borgen, Geborgtes) – Übertrag bei binärer Subtraktion

**breakpoint** (Haltepunkt) – Adresse, bei der bei Abarbeitung des dort abgespeicherten Befehls der Anwenderprogrammablauf unterbrochen wird, um dem Benutzer Test- und Eingabemöglichkeiten zu gestatten (eine solche Haltepunktorganisation erfolgt insbesondere in Debugprogrammen)

**buffer** (Puffer) – Treiberstufe zur Leistungsanpassung

**Bus** (bus) – Leitungsbündel zum Informationsaustausch, an das mehrere Sender (Informationsquellen) und Empfänger (-senken) angeschlossen werden können

**byte** (Byte, byte Bezeichnung einer zusammengehörenden Gruppe von, meist, 8 bit; Oktade) – 1. Byte: kleinste adressierbare Speicherstelle; 2. byte: Einheit des Informationsgehalts

**carry** Übertrag

**chip** (Halbleiterkristall) – wird übertragend auch für einen Baustein verwendet, dessen Schaltung auf einem Halbleiterkristall untergebracht ist

**CMOS** (Complementary Metal-Oxide Semiconductor, komplementäre MOS-Technologie) – benutzt in einer Schaltung p- und n-Kanal-Transistoren, gekennzeichnet durch sehr geringen Leistungsverbrauch, hohe Störsicherheit, mittelschnell

**COBOL** (COmmon Business Oriented Language, kaufmännisch orientierte Sprache) – höhere Programmiersprache, auch für Anwendung in entsprechend konfigurierten Mikrorechnern

**code** (Kode) – Verschlüsselungsvorschrift

**compiler** (Compiler) – Übersetzungsprogramm, das Quellprogramme, die in einer höheren Programmiersprache geschrieben sind, in die Maschinensprache übersetzt; hierbei können Formalfehler erkannt werden

**condition code** (Bedingungskode) – bezeichnet diejenigen Bits eines Registers, die Auskunft über das Auftreten bestimmter Bedingungen bei der zuletzt ausgeführten Operation der ALU geben

**CPU** (Central Processing Unit, ZVE zentrale Verarbeitungseinheit) – Rechen- und Steuerwerk eines Rechnersystems

**CRC** (Cyclic Redundancy Check, zyklische Redundanzkontrolle) – Verfahren zum Ermitteln von Fehlern bei der Datenübertragung und Speicherung

**CRT** (Cathode Ray Tube – Katodenstrahlröhre)

**CRT-display** Datensichtgerät mit Katodenstrahlröhre

**CTC** (Counter Timer Circuit) – Zähler/Zeitgeber, peripheres Systemelement U857 des Prozessorsystems U880

**daisy chain** (kettenförmige Verbindungsstruktur) – dient zum Informationsaustausch zwischen mehreren Funktionseinheiten, wobei i. allg. die Information in der Kette weitergereicht wird; durch die Reihenfolge der Elemente in der Kette kann eine Priorität festgelegt werden

**Datenbus** (data bus) – Sammelleitung für die Datenübertragung in Rechnern

**DAU** (Digital/Analog-Umsetzer, DAC digital to analog converter) – D/A-Wandler, erzeugt aus einem digitalen Kode eine zugehörige analoge Ausgangsgröße

**debugging** („Entwanzung“) – Fehlersuche und Fehlerbeseitigung in Hard- und Software, wird i. allg. durch Debugprogramme unterstützt

**dekrementieren** (decrement) – vermindern, i. allg. Erniedrigen eines gespeicherten Zahlenwerts um 1

**disable** – verhindern, sperren

**Diskette** (floppy disk) – preiswertes Speichermedium in Form einer flexiblen, rotierenden Magnetfolienscheibe mit wahlfreiem Zugriff, gekennzeichnet durch

- Speicherkapazität 0,65 ... 8 Mbit
- Zugriffszeit 0,06 ... 1,7 s
- Übertragungsrate 33 ... 1200 kbit/s

**DMA** (Direct Memory Access, direkter Speicherzugriff) – schneller Datenzugriff auf die Speicherbaugruppe oder peripheren Baugruppen eines Rechners durch eine DMA-Einheit unter Umgehung der CPU

**dynamischer Speicher** (dynamic memory) – Halbleiterspeicher, bei dem zyklisch der Informationsinhalt aufgefrischt werden muß, aufgrund der dadurch erreichten geringen Zeilenabmessungen hohe Speicherkapazitäten, geringe Zugriffszeiten

**EAROM** (Electrically Alterable ROM, elektrisch veränderbarer Festwertspeicher) – Inhalt der Speicherzellen kann vom Anwender blockweise gelöscht und neu eingeschrieben werden (z.Z. jedoch geringe Speicherkapazitäten, hohe Lösch- und Einschreibzeiten)

**editor** (Editor) – Textverarbeitungsprogramm, das die Erstellung und Veränderung von Quellprogrammen unterstützt

**EMR** – Einchipmikrorechner, eine auf einem Halbleiterkristall untergebrachte vollständige Rechnerstruktur

**Emulation** (emulation) – eine Simulation, die soweit eingeschränkt ist, daß nur das äußere Verhalten eines Systems nachgebildet wird

**enable** – freigeben, aktivieren

**Entwicklungssystem** (development system) – Sammelbegriff für Hard- und Softwarekomponenten, die ein rationelles hard- und softwaremäßiges Testen eines zu entwickelnden Rechners sowie die zugehörige Programmentwicklung unterstützen

**EPROM** (Erasable Programmable ROM, löschbarer und elektrisch programmierbarer Festwertspeicher) – der Speicherinhalt kann durch UV-Strahlung insgesamt gelöscht werden, danach kann eine erneute elektrische Programmierung erfolgen

**FIFO** (First In First Out) – Speicherprinzip bei seriell organisierten Speichern; das zuerst in den Speicher Übernommene wird zuerst wieder ausgegeben

**file** (Datei) – zweckbezogene Zusammenfassung von Daten

**flag** (Flagge) – Kennzeichnungsbit, das von der Hard- oder Software gesetzt werden kann, um bestimmte Zustände oder Ereignisse festzuhalten bzw. anzuzeigen

**floating** (schwebend) – hochohmiger Zustand der Ausgangsstufen eines Bussenders

**FORTTRAN** (FORmular TRANslation language, Anweisungen übertragende Sprache) – höhere Programmiersprache für technisch-wissenschaftliche Probleme

**gate** (Gatter) – Torschaltung, elementare Verknüpfungsschaltung digitaler Signale

**glitches** (Spannungsspitzen) – entstehen z.B. bei schnellen DAU durch unsymmetrische Schaltzeiten

**H** (High, hoch), binärer Zustand mit höherem Potential; entspricht bei positiver Logik der logischen 1

**handshaking** (Quittierungsbetrieb) – Datenaustausch, bei dem eine Information erst nach einer Quittierung übernommen wird; ermöglicht Datenaustausch zwischen Geräten mit unterschiedlichen Reaktionsgeschwindigkeiten

**Hardware** (hardware) – Sammelbegriff für alle Bauteile und Geräte eines Rechnersystems

**HDLC** (High level Data Link Control procedure, Datenübertragungssteuerung auf hohem Niveau) – bitorientiertes Steuerungsverfahren zur Datenübertragung

**hexadezimal** (hexadecimal) – Zahlendarstellung zur Basis 16 (Hexadezimalsystem); vier Dualziffern werden jeweils zu einer Hexadezimalziffer zusammengefaßt

**ICE** In Circuit Emulation (Schaltkreisemulation über die Anschlußbeschtaltung) – Prinzip der Hardwaretestung eines Rechners, bei der die Funktionen des Prozessors über eine Kabelverbindung von einem Entwicklungssystem simuliert werden

**ICE bus (ICE-Bus)** – genormtes Bussystem (ursprünglich für den Einsatz in der Meß- und Regelungstechnik konzipiert) mit acht Datenlinien, drei Quittierungsleitungen und fünf Steuerleitungen

**Indexregister (index register)** – CPU-Register, die i. allg. zur Adreßmodifikation bei Operationen verwendet werden

**inkrementieren (increment)** – erhöhen, i. allg. Vergrößern eines gespeicherten Zahlenwerts um 1

**interaktiver Verkehr (interactive mode)** – Dialogart, bei der für den Benutzer unmittelbare Eingriffsmöglichkeiten in den Programmablauf bestehen

**interface (Schnittstelle, Interface)** – Nahtstelle zwischen zwei Systemteilen, die durch Art und Bedeutung der übertragenen Signale, evtl. aber auch durch die mechanische Ausführung beschrieben wird

**interpreter (Interpreter, Interpretierprogramm)** – ein Übersetzungsprogramm, das die Befehle eines in einer höheren Programmiersprache erarbeiteten Quellprogramms nacheinander in die Maschinensprache übersetzt und dabei *sofort* ausführt; es wird somit kein Objektprogramm erzeugt; im Zusammenhang mit Mikrorechnern sind **BASIC**-Interpreter besonders verbreitet

**interrupt (Interrupt, Unterbrechung)** – dient zur Einschachtelung eines Unterprogramms (ISR) in den Programmablauf auf hardwaremäßige Anforderung

**ISR Interrupt Service Routine (Interruptserviceroutine, Interruptbearbeitungsprogramm)**

**k (Kilo)** – Einheitenvorsatz k entspricht  $10^3$

**K (Kilo)** – Einheitenvorsatz K entspricht  $2^{10} = 1024$ , z. B. 2 Kbit = 2048 bit

**keyboard (Tastefeld)** – Eingabetastatur

**kit** – Bausatz

**L Low (niedrig)** – binärer Logikzustand mit dem niedrigeren Potential; entspricht bei positiver Logik der logischen 0

**label (Marke)** – Kennzeichnung, Bezeichnung für symbolische Adreßangabe in Quellprogrammen

**latch (einrasten)** – Auffangregister bzw. Auffangflipflop, bei dem sich der Ausgang mit dem Eingang ändert, solange ein Taktsignal aktiv ist

**LED Light Emitting Diode (Leuchtdiode)** – Licht aussendende Halbleiterdiode

**LIFO Last In First Out** – Speicherprinzip bei seriell organisierten Speichern; das zuletzt in den Speicher Übernommene wird zuerst wieder ausgegeben, Entnahmepprinzip bei Kellerspeichern (stack)

**linker (Linker, Binder)** – Programm, das mehrere verschieblich übersetzte Programmteile zusammensetzt und dem so entstandenen Maschinenprogramm feste Adreßwerte zuordnet

**LSB Least Significant Bit (niedrigstwertiges Bit)** – Bit einer Dualzahl mit dem niedrigsten Stellenwert

**LSI Large Scale Integration (hoher Integrationsgrad)** – kennzeichnet bei monolithischen Halbleiterschaltungen das Unterbringen komplexer Schaltungen (mehr als 100 Gatterfunktionen) auf einem Chip

**M (Mega)** – Einheitenvorsatz, 1. M entspricht  $10^6$ ; 2. in der Digitaltechnik entspricht M i. allg.  $2^{20} = 1048576$

**Maschinenprogramm** (machine program) – rechnerbezogenes (Anwender-) Programm, das aus den Befehlsbitmustern der Maschinensprache des jeweiligen Prozessors zusammengesetzt ist

**Maske** (mask) – Bitmuster, das zum Ausblenden oder Komplettieren bestimmter Bitgruppen bestimmt ist

**Massenspeicher** (mass memory) – Speicher, die eine sehr große Speicherkapazität aufweisen, z. B. Magnetplattenspeicher

**Mikroprogrammierbarkeit** (microprogrammability) – Eigenschaft eines Rechners, auch seine Mikrobefehle dem Benutzer zugänglich zu machen; dadurch kann ein Anwender einen eigenen Befehlsvorrat festlegen

**mnemonic code** (mnemonischer Kode) – Befehlskode, dessen alphanumerische Kürzel Hinweise auf Eigenschaften enthalten und somit gleichzeitig als Gedächtnisstütze genutzt werden können

**modem** modulator/demodulator (Modem) – Kombination von Modulator und Demodulator für die Datenübertragung

**monitor** (Betriebssystem) – Programmpaket, das den Programmablauf in einem Rechner steuert, koordiniert und überwacht; es übernimmt dabei die Zuordnung sämtlicher Geräte und Baugruppen des Rechners

**monolithisch** (monolithic) – auf einem Halbleiterkristall untergebracht

**MSB** Most Significant Bit (höchstwertiges Bit) – Bit einer Dualzahl mit dem höchsten Stellenwert

**MSI** Medium Scale Integration (mittlerer Integrationsgrad) – kennzeichnet das Unterbringen von Schaltungen mittlerer Komplexität (bis etwa 100 Gatterfunktionen) auf einem Halbleiterchip

**MTC** Magnetic Tape Cassette (Magnetbandkassette) – preiswertes Speichermedium für sequentiellen Zugriff (zwei bitserielle Spuren auf 3,81 mm breitem Magnetband)

**nesting** (Schachtelung, Nesting) – einer Unterprogrammverschachtelung entsteht, wenn in einem Unterprogramm ein weiteres Unterprogramm aufgerufen wird; die Verwaltung der Rückkehradressen wird i. allg. von einem Kellerspeicher (stack) vorgenommen

**NMOS** N-channel MOS (N-Kanal-MOS) – MOS-Halbleitertechnologie, die sich durch relativ geringen Kristallflächen- und Leistungsbedarf sowie geringe typische Schaltzeiten auszeichnet

**nonvolatile** (nichtflüchtig) – i. allg. im Zusammenhang mit Speichern verwendet, die bei Abschalten der Rechnerversorgungsspannung ihren Inhalt nicht verlieren

**object program** (Objektprogramm) – (Anwender-) Programm in Maschinensprache; wird i. allg. von Assembler oder Compiler erzeugt

**octal** (oktal) – Zahlendarstellung zur Basis 8 (Oktalsystem)

**Off-line-Verarbeitung** (off-line-processing) – Datenverarbeitungsschritte, die nicht unter Regie eines zentralen Verarbeitungsrechners ablaufen (z. B. Datenerfassung mit Lochband)

**On-line-Verarbeitung** (on-line-processing) – alle Schritte der Datenverarbeitung werden unmittelbar vom Verarbeitungsrechner ausgeführt oder gesteuert

**op-code** operation code (Befehlskode) – i. allg. binärer oder hexadezimaler Kode zur Darstellung von Maschinenbefehlen

**OS** Operation System (Betriebssystem) – siehe Monitor

**PAP** Programmablaufplan (flow chart) – beschreibt (meist symbolisch) den Ablauf von Operationen in einem Datenverarbeitungssystem

**Paritätsbit** (parity bit) – Prüfbit, das einer Information (z. B. einem Byte) angehängt wird, um eine einfache Kontrollmöglichkeit bei Speicherung oder Übertragung dieser Operation zu erhalten; das Paritätsbit ergänzt je nach Definition die Anzahl der logischen „Einsen“ dieser Information zu einer geraden (even parity) oder ungeraden Summe (odd parity)

**PC** Programm Counter (Befehlszähler, Adreßzähler) – Register des Prozessors, das die Adresse des nächsten zu bearbeitenden Befehls enthält

**peripheres Element** (peripheral unit) – zum Rechner gehörendes Element, das eine Interfacesteuerung ausführt

**peripheres Gerät** (peripheral device) – rechnerexterne Ein-/Ausgabe- und Speichergeräte

**PIO** Parallel Input/Output unit (parallele Ein-/Ausgabe-Einheit) – zum System U880 gehörendes Systemelement (U855), das parallele Ein-/Ausgabe-Funktionen ausführt

**pin** (Pin) – Anschlußstift (einer integrierten Schaltung)

**PL/1** Programming Language One (Programmiersprache Eins) – höhere Programmiersprache, die sowohl Eigenschaften von technisch-wissenschaftlichen als auch von kaufmännisch orientierten Programmiersprachen aufweist; im Zusammenhang mit Mikrorechnern werden häufig Untermengen von PL/1 angewendet, z. B. auch zur befehlsmäßigen Ergänzung von Assemblerprogrammen (PLZ, PLM)

**PMOS** P-channel MOS (P-Kanal-MOS) – historisch gesehen älteste MOS-Halbleitertechnologie, die gegenüber NMOS langsamer ist und einen größeren Flächen- und Leistungsbedarf hat

**polling** (Polling) – Programmiertechnik, die durch ein zyklisches Abfragen der Eingabestationen gekennzeichnet ist

**port** (Port) – Kanal; von der Wirkung gesehen eigenständige Ein-/Ausgabe-Schnittstelle, z. B. 8-bit-Port bei parallelem Interface

**Priorität** (priority) – Rangfolge, Elemente mit höherer Priorität werden in Konfliktfällen vorrangig behandelt

**Programm** (program, amerik: programme) – Folge von Befehlen zur Lösung einer Aufgabenstellung

**Programmiersprache** (programming language) – System von Sprachelementen und Regeln zur Beschreibung von Verarbeitungsabläufen:

1. maschinenorientierte Programmiersprachen sind i. allg. rechnerabhängig (z. B. Assembler); 2. problemorientierte Programmiersprachen sind i. allg. anwendungsspezifisch (z. B. ALGOL und COBOL)

**PROM** Programmable ROM (programmierbarer Festwertspeicher) – Festwertspeicher, der ähnlich wie ein EPROM vom Anwender elektrisch programmiert werden kann, aber aufgrund der Gehäusespezifika (fehlendes Quarzglasfenster) nicht löschtbar ist

**protocol** – Übermittlungsvorschrift

**Puffer** (buffer) – Treiberstufe zur Leistungsanpassung

**punch** (Puncher, Stanzer) – Ausgabegerät für Lochband oder Lochkarten

**Quellprogramm** (source program) – in einer Programmiersprache geschriebenes Anwenderprogramm

**RAM** Random Access Memory (Speicher mit wahlfreiem Zugriff) – i. allg. Bezeichnung für Halbleiter-Schreib-/Lese-Speicher

**record** (Satz, Spur) – Zusammenfassung sachlich zusammengehörender Begriffe oder Daten; stellt i. allg. die niedrigste Ebene einer Datenbank dar; mehrere Sätze bilden eine Datei

**refresh** (Refresh, Auffrischen) – das Vermeiden von Informationsverlusten bei dynamischen Prozessen durch Auffrischen des Mediums

**Register** (register) – schneller Zwischenspeicher mit parallelem Zugriff, z. B. in der CPU

**relocatable** (relokativ, verschieblich) – nicht an eine bestimmte feste Adresse gebunden

**ROM** Read Only Memory (Nur-Lese-Speicher) – nichtflüchtiger Festwertspeicher mit wahlfreiem Lesezugriff; i. allg. maskenprogrammierter Festwertspeicher, dessen Inhalt während des Herstellungsprozesses durch eine Metallisierungsmaske festgelegt wird; ist deshalb deutlich billiger als vergleichbare PROM bzw. EPROM, aber aufgrund der notwendigen technologischen Vorbereitungen an Mindeststückzahlen gebunden

**scratch pad** (Notizblock) – Bezeichnung für Speicherbereiche, die für die Ablage von Zwischenergebnissen reserviert sind

**SDLC** Serial Data Link Control procedure (serielle Datenübertragungssteuerung) – ein dem HDLC ähnliches, bitorientiertes Steuerungsverfahren zur Datenübermittlung

**SIO** Serial Input Output Unit (serielle Ein-/Ausgabe-Einheit) – zum Prozessorsystem U880 gehörendes Systemelement (U856) zur seriellen Datenein-/ausgabe

**software** (Software, Weichware) – Sammelbegriff für alle Arten von Programmen und Dokumentationen

**SSI** Small Scale Integration (geringer Integrationsgrad) – kennzeichnet bei Halbleitertechnologien das Unterbringen von einfachen Schaltungen (bis etwa 10 Gatterfunktionen) auf einem Chip

**stack** (Stapelspeicher, Kellerspeicher) – Speicher mit geringer Kapazität, der nach dem LIFO-Prinzip organisiert ist; i. allg. zur Abspeicherung von Rückkehradressen bei Unterprogrammverschachtelungen

**Steuerbus** (control bus) – Sammelleitung für die Übertragung von Steuerinformationen zwischen den Baugruppen eines Rechners

**synchron** (synchronous) taktgesteuert, getaktet – 1. mit festem Grundtakt arbeitend; 2. mit gleichem Takt arbeitende, starr gekoppelte Einheiten; 3. bei synchroner Datenübertragung liegen alle Binärzeichen in einem festen Zeitraster; der Übertragungsbeginn erfolgt synchron zum Sendetakt

**Terminal** (terminal) – Datenstation, i. allg. mit Eingabemöglichkeit über alphanumerische Tastatur und Ausgabe über Datensichtgerät

**three state output** (auch tri-state output; Dreizustandsausgang) – Ausgang, der neben den logischen Zuständen L und H einen hochohmigen Ausgangszustand aufweist; die Anwendung erfolgt z. B. wenn mehrere Treiber (Datenquellen) an einen Bus angeschlossen werden müssen

**TINY BASIC** – höhere Programmiersprache auf niedrigem Niveau; Untermenge von BASIC; Anwendung vor allem bei Mikrorechnerlernsystemen; leicht erlernbar

**TTY** Tele TYpe (-writer), (Fernschreiber) – in der Rechentechnik häufig als Ein-/Ausgabe-Gerät (Terminal) genutzt

**UART** Universal Asynchronous Receiver/Transmitter (universeller asynchroner Empfänger/Sender) – i. allg. programmierbares Peripherieelement zur asynchronen seriellen Datenübertragung

**USART** Universal Synchronous /Asynchronous Receiver/Transmitter (universeller synchroner/asynchroner Empfänger/Sender) – programmierbares Peripherieelement zur synchronen und asynchronen Datenübertragung (ähnlich SIO)

**utility program** (Dienstprogramm) – Sammelbezeichnung für Hilfsprogramme (z. B. Speichertestprogramme, Ansteuerprogramme für periphere Geräte)

**vectored interrupt** (vektoriertes Interrupt, gerichteter Interrupt) – das unterbrechende Element liefert eine Kennung (i. allg. Adresse, Vektoradresse), die zum Aufruf eines zugehörigen Unterprogramms verwendet wird

**VLSI** Very Large Scale Integration (Größtintegration) – kennzeichnet bei monolithischen Halbleiterspeichern das Unterbringen von sehr komplexen Schaltungen (mehr als 10000 Gatterfunktionen) auf einem Chip

**volatile** (flüchtig) – wird i. allg. im Zusammenhang mit Speichern verwendet, die bei Ausfall der Spannungsversorgung ihren Inhalt verlieren

**Wort** (word) – i. allg. eine Folge von binären Zeichen, die bei der Verarbeitung und Speicherung in Rechnern als Einheit behandelt werden



ADR	OBJ-KODE	ZEILE	QUELL-KODE	ADR	OBJ-KODE	ZEILE	QUELL-KODE	BEFEHLSLISTE	SEITE 4
00BF	7B	117	OUT 29	00FD	B5	175	ORA H		
00C0	7C	118	JNP NN	00FE	B6	176	ORA L		
00C3	7D	119	OUT 30	00FF	B7	177	ORA M		
00C4	7E	120	CALL NN	0100	B8	178	CHP A		
00C7	7F	121	OUT 31	0101	B9	179	CHP B		
00C8	80	122	AND A	0102	BA	180	CHP C		
00C9	81	123	ADD B	0103	BB	181	CHP D		
00CA	82	124	ADD C	0104	BC	182	CHP E		
00CB	83	125	ADD D	0105	BD	183	CHP H		
00CC	84	126	ADD E	0106	BE	184	CHP L		
00CD	85	127	ADD H	0107	BF	185	CMP M		
00CE	86	128	ADD L	0108	C0	186	NOP		
00CF	87	129	ADD M	0109	C1	187	MOV A,B		
00D0	88	130	ADC A	010A	C2	188	MOV A,C		
00D1	89	131	ADC B	010B	C3	189	MOV A,D		
00D2	8A	132	ADC C	010C	C4	190	MOV A,E		
00D3	8B	133	ADC D	010E	C6	192	MOV A,L		
00D4	8C	134	ADC E	0110	C7	193	MOV A,N		
00D5	8D	135	ADC H	0110	C8	194	MOV B,A		
00D6	8E	136	ADC L	0111	C9	195	MOV B,B		
00D7	8F	137	ADC M	0112	CA	196	MOV B,C		
00D8	90	138	SUB A	0113	CB	197	MOV B,D		
00D9	91	139	SUB B	0114	CC	198	MOV B,E		
00DA	92	140	SUB C	0115	CD	199	MOV B,L		
00DB	93	141	SUB D	0116	CE	200	MOV B,M		
00DC	94	142	SUB E	0117	CF	201	MOV B,M		
00DD	95	143	SUB H	0118	D0	202	MOV C,A		
00DE	96	144	SUB L	0119	D1	203	MOV C,B		
00DF	97	145	SUB M	011A	D2	204	MOV C,C		
00E0	98	146	SBB A	011B	D3	205	MOV C,D		
00E1	99	147	SBB B	011C	D4	206	MOV C,E		
00E2	9A	148	SBB C	011D	D5	207	MOV C,H		
00E3	9B	149	SBB D	011E	D6	208	MOV C,L		
00E4	9C	150	SBB E	011F	D7	209	MOV C,M		
00E5	9D	151	SBB H	0120	D8	210	MOV D,A		
00E6	9E	152	SBB L	0121	D9	211	MOV D,B		
00E7	9F	153	SBB M	0122	DA	212	MOV D,C		
00E8	AO	154	ANA A	0123	DB	213	MOV D,D		
00E9	A1	155	ANA B	0124	DC	214	MOV D,E		
00EA	A2	156	ANA C	0125	DD	215	MOV D,H		
00EB	A3	157	ANA D	0126	DE	216	MOV D,L		
00EC	A4	158	ANA E	0127	DF	217	MOV D,M		
00ED	A5	159	ANA H	0128	E0	218	MOV E,A		
00EE	A6	160	ANA L	0129	E1	219	MOV E,B		
00EF	A7	161	ANA M	012A	E2	220	MOV E,C		
00F0	A8	162	XRA A	012B	E3	221	MOV E,D		
00F1	A9	163	XRA B	012C	E4	222	MOV E,E		
00F2	AA	164	XRA C	012D	E5	223	MOV E,H		
00F3	AB	165	XRA D	012E	E6	224	MOV E,L		
00F4	AC	166	XRA E	012F	E7	225	MOV E,M		
00F5	AD	167	XRA H	0130	E8	226	MOV H,A		
00F6	AE	168	XRA L	0131	E9	227	MOV H,B		
00F7	AF	169	XRA M	0132	EA	228	MOV H,C		
00F8	B0	170	ORA A	0133	EB	229	MOV H,D		
00F9	B1	171	ORA B	0134	EC	230	MOV H,E		
00FA	B2	172	ORA C	0135	ED	231	MOV H,H		
00FB	B3	173	ORA D	0136	EE	232	MOV H,L		
00FC	B4	174	ORA E						

; VERGL.: FLAGGS\_(A-REG)

; REGISTERLADEBEFEHLE

; ADDITION: A+CY-REG

; SUBTR.: A-REG

; SUBTR.: A-CY-REG

; LOG.AND: A AND REG

; LOG.XOR: A XOR REG

; LOG.OR: A OR REG

ADR	OBJ-KODE	ZEILE	QUELL-KODE	BEFEHLSLISTE	SEITE_5
0137	EF	233	MOV H,H		ASR U808
0138	F0	234	MOV L,A		
0139	F1	235	MOV L,B		
013A	F2	236	MOV L,C		
013B	F3	237	MOV L,D		
013C	F4	238	MOV L,E		
013D	F5	239	MOV L,H		
013E	F6	240	MOV L,L		
013F	F7	241	MOV L,H		
0140	F8	242	MOV H,A		
0141	F9	243	MOV H,B		
0142	FA	244	MOV H,C		
0143	FB	245	MOV H,D		
0144	FC	246	MOV H,E		
0145	FD	247	MOV H,L		
0146	FE	248	HLT		
0147	FF	249	RET	:HALT	
0148	3F	250	RET	:REDUNDANTER OBJ-KODE	
0149		251	H# DEFS 2		
		252	H EQU 11H		
		253	END		

## A.2. Programmunterlagen zum U880

SEITE 2  
ASH U880SEITE 1  
ASH U880SEITE 2  
ASH U880SEITE 1  
ASH U880SEITE 2  
ASH U880SEITE 1  
ASH U880SEITE 2  
ASH U880

ADR	OBJ-KODE	M	ZEILE	QUELL-KODE	BEFEHLSLISTE	ADR	OBJ-KODE	M	ZEILE	QUELL-KODE	BEFEHLSLISTE
0000	00		1	NOP		0055	3A8605		59	LD A,(N	
0001	018605		2	LD BC,NN		0058			60	DEC SP	
0004	02		3	LD (BC),A		0059			61	INC A	
0005	03		4	INC BC		0062			62	DEC A	
0006	04		5	INC B		005A			63	LD A,H	
0007	05		6	DEC B		005B			64	CCF	
0008	0633		7	LD B,N		005D			65	LD B,B	
000A	07		8	RLCA		005E			66	LD B,C	
000B	08		9	EXAF		005F			67	LD B,D	
000C	09		10	ADD HL,BC		0060			68	LD B,E	
000D	0A		11	LD A,(BC)		0061			69	LD B,H	
000E	0B		12	DEC BC		0062			70	LD B,L	
000F	0C		13	INC C		0063			71	LD B,M	
0010	0D		14	DEC C		0064			72	LD B,A	
0011	0E33		15	LD C,N		0065			73	LD C,B	
0013	0F		16	RRCA		0066			74	LD C,C	
0014	102E		17	DJNZ DIS		0067			75	LD C,D	
0016	118605		18	LD DE,NN		0068			76	LD C,E	
0019	12		19	LD (DE),A		0069			77	LD C,H	
001A	13		20	INC DE		006A			78	LD C,L	
001B	14		21	INC D		006B			79	LD C,M	
001C	15		22	DEC D		006C			80	LD C,A	
001D	1633		23	LD D,N		006D			81	LD D,A	
001E	17		24	RLA		006E			82	LD D,C	
0020	1822		25	JR DIS		006F			83	LD D,D	
0022	19		26	ADD HL,DE		0070			84	LD D,E	
0023	1A		27	LD A,(DE)		0071			85	LD D,H	
0024	1B		28	DEC DE		0072			86	LD D,L	
0025	1C		29	INC E		0073			87	LD D,H	
0026	1D		30	DEC E		0075			88	LD D,A	
0027	1E33		31	LD E,N		0076			89	LD E,B	
0029	1F		32	RRR		0077			90	LD E,C	
002A	2018		33	JR NZ DIS		0078			91	LD E,D	
002C	218605		34	LD HL,NN		0079			92	LD E,E	
002F	228605		35	LD (NN),HL		007A			93	LD E,H	
0032	23		36	INC HL		007B			94	LD E,L	
0033	24		37	INC H		007C			95	LD E,M	
0034	25		38	DEC H		007D			96	LD E,A	
0035	2633		39	LD H,N		007E			97	LD H,B	
0037	27		40	DA		007F			98	LD H,C	
0038	280A		41	JR Z DIS		0080			99	LD H,D	
003A	29		42	ADD HL,HL		0081			100	LD H,E	
003B	2A8605		43	LD HL,(NN)		0082			101	LD H,L	
003E	2B		44	DEC HL		0083			102	LD H,M	
003F	2C		45	INC L		0084			103	LD H,W	
0040	2C		46	DEC L		0085			104	LD H,A	
0041	2E33		47	LD L,H		0086			105	LD L,B	
0043	2F		48	CPL		0087			106	LD L,C	
0044	30FE		49	JR NC DIS		0088			107	LD L,D	
0046	318605		50	LD SP,NN		0089			108	LD L,E	
0049	328605		51	LD (NN),A		008A			109	LD L,H	
004C	33		52	INC SP		008B			110	LD L,L	
004D	34		53	INC M		008C			111	LD L,H	
004E	35		54	DEC M		008D			112	LD L,A	
004F	3633		55	LD M,N		008E			113	LD M,B	
0051	37		56	SCF		008F			114	LD M,C	
0052	38F0		57	JR C DIS		0090			115	LD H,D	
0054	39		58	ADD HL,SP		0091			116	LD H,E	

ADR	OBJ-KODE	M	ZEILE	QUELL-KI
0092	74		117	LD M,H
0093	75		118	LD M,L
0094	76		119	HALT
0095	77		120	LD M,A
0096	78		121	LD A,B
0097	79		122	LD A,C
0098	7A		123	LD A,D
0099	7B		124	LD A,E
009A	7C		125	LD A,H
009B	7D		126	LD A,L
009C	7E		127	LD A,M
009D	7F		128	LD A,A
009E	80		129	ADD B
009F	81		130	ADD C
00A0	82		131	ADD D
00A1	83		132	ADD E
00A2	84		133	ADD H
00A3	85		134	ADD L
00A4	86		135	ADD M
00A5	87		136	ADD A
00A6	88		137	ADC B
00A7	89		138	ADC C
00A8	8A		139	ADC D
00A9	8B		140	ADC H
00AA	8C		141	ADC L
00AB	8D		142	ADC M
00AC	8E		143	ADC A
00AD	8F		144	ADC A
00AE	90		145	SUB B
00AF	91		146	SUB C
00B0	92		147	SUB D
00B1	93		148	SUB E
00B2	94		149	SUB H
00B3	95		150	SUB L
00B4	96		151	SUB M
00B5	97		152	SUB A
00B6	98		153	SBC B
00B7	99		154	SBC C
00B8	9A		155	SBC D
00B9	9B		156	SBC E
00BA	9C		157	SBC H
00BB	9D		158	SBC L
00BC	9E		159	SBC M
00BD	9F		160	SBC A
00BE	A0		161	AND B
00BF	A1		162	AND C
00C0	A2		163	AND D
00C1	A3		164	AND E
00C2	A4		165	AND H
00C3	A5		166	AND L
00C4	A6		167	AND M
00C5	A7		168	AND A
00C6	A8		169	XOR B
00C7	A9		170	XOR C
00C8	AA		171	XOR D
00C9	AB		172	XOR E
00CA	AC		173	XOR H
00CB	AD		174	XOR L

OBJ-KODE	M	ZEILE	QUELL-KODE
AE		175	XOR M
AF		176	XOR A
80		177	OR B
81		178	OR C
82		179	OR D
83		180	OR E
84		181	OR H
85		182	OR L
86		183	OR M
87		184	OR A
88		185	OR B
89		186	OR C
8A		187	OR D
8B		188	OR E
8C		189	OR H
8D		190	OR L
8E		191	OR M
8F		192	OR A
90		193	OR B
91		194	OR C
92		195	OR D
93		196	OR E
94		197	OR H
95		198	OR L
96		199	OR M
97		200	RST 0
98		201	RZ
99		202	BET
9A		203	JPZ M
9B		204	JPZ NM
9C		205	CAZ M
9D		206	CALL NM
9E		207	ADC M
9F		208	RST 8
00		209	RNC
01		210	POP DE
02		211	JPNC MN
03		212	OUT N
04		213	CAHC MN
05		214	PUSH DE
06		215	SUB N
07		216	RST 10H
08		217	PC
09		218	EXX
0A		219	JPC MN
0B		220	IN M
0C		221	CAC MN
0D		222	SBC M
0E		223	RST 18H
0F		224	RPO
10		225	POP HL
11		226	LPPO MN
12		227	EX(SP),HL
13		228	CAPO RH
14		229	PUSH HL
15		230	AND N
16		231	RST 20H
17		232	RPE
18		233	JMP M

ADR	OBJ-KODE	M	ZEILE	QUELL-KODE	REFEHLISLISTE
0194	CB26		291	SLA M	
0196	CB27		292	SLA A	
0198	CB28		293	SLA B	
019A	CB29		294	SRA C	
019C	CB2A		295	SRA D	
019E	CB2B		296	SRA E	
01A0	CB2C		297	SRA H	
01A2	CB2D		298	SRA L	
01A4	CB2E		299	SRA M	
01A6	CB2F		300	SRA A	
01A8	CB38		301	SRL B	
01AA	CB39		302	SRL C	
01AC	CB3A		303	SRL D	
01AE	CB3B		304	SRL E	
01B0	CB3C		305	SRL H	
01B2	CB3D		306	SRL L	
01B4	CB3E		307	SRL M	
01B6	CB3F		308	SRL A	
01B8	CB40		309	BIT 0, B	
01BA	CB41		310	BIT 0, C	
01BC	CB42		311	BIT 0, D	
01BE	CB43		312	BIT 0, E	
01C0	CB44		313	BIT 0, H	
01C2	CB45		314	BIT 0, L	
01C4	CB45		315	BIT 0, M	
01C6	CB47		316	BIT 0, A	
01C8	CB48		317	BIT 1, B	
01CA	CB49		318	BIT 1, C	
01CC	CB4A		319	BIT 1, D	
01CE	CB4B		320	BIT 1, E	
01D0	CB4C		321	BIT 1, H	
01D2	CB4D		322	BIT 1, L	
01D4	CB4E		323	BIT 1, A	
01D6	CB4F		324	BIT 1, A	
01D8	CB50		325	BIT 2, B	
01DA	CB51		326	BIT 2, C	
01DC	CB52		327	BIT 2, D	
01DE	CB53		328	BIT 2, E	
01E0	CB54		329	BIT 2, H	
01E2	CB55		330	BIT 2, L	
01E4	CB56		331	BIT 2, M	
01E6	CB57		332	BIT 2, A	
01E8	CB58		333	BIT 3, B	
01EA	CB59		334	BIT 3, C	
01EC	CB5A		335	BIT 3, D	
01EE	CB5B		336	BIT 3, E	
01F0	CB5C		337	BIT 3, H	
01F2	CB5D		338	BIT 3, L	
01F4	CB5E		339	BIT 3, A	
01F6	CB5F		340	BIT 3, A	
01F8	CB60		341	BIT 4, B	
01FA	CB61		342	BIT 4, C	
01FC	CB62		343	BIT 4, D	
01FE	CB63		344	BIT 4, E	
0200	CB64		345	BIT 4, H	
0202	CB65		346	BIT 4, L	
0204	CB66		347	BIT 4, M	
0206	CB67		348	BIT 4, A	

ADR	OBJ-KODE	M	ZEILE	QUELL-KODE	REFEHLISLISTE
0125	EA8605		233	JPEE MN	
0128	EA8605		234	EX DE, HL	
0129	EA8605		235	CAPE MN	
012C	EA8605		236	XOR N	
012E	EF33		237	RST 28H	
012F	FO		238	RP	
0130	F1		239	POP AF	
0131	F28605		240	JPP NH	
0134	F3		241	DI	
0135	F48605		242	CAP NN	
0136	F5		243	PUSH AF	
0139	F633		244	OR N	
013B	F7		245	RST 30H	
013C	F8		246	PM	
013D	F9		247	LD SP, HL	
013E	FAB8605		248	JPH NH	
0141	FB		249	FI	
0142	FC8605		250	CAM NN	
0145	FE33		251	CMP M	
0147	FE33		252	RST 38H	
0148	CB00		253	RLC B	
014A	CB01		254	RLC C	
014C	CB02		255	RLC D	
014E	CB03		256	RLC E	
0150	CB04		257	RLC H	
0152	CB05		258	RLC L	
0154	CB06		259	RLC M	
0156	CB07		260	RLC A	
0158	CB08		261	RLC B	
015A	CB09		262	RLC C	
015C	CB0A		263	RLC D	
015E	CB0B		264	RLC E	
0160	CB0C		265	RLC H	
0162	CB0D		266	RLC L	
0164	CB0E		267	RLC M	
0166	CB0F		268	RLC A	
0168	CB10		269	RL B	
016A	CB11		270	RL C	
016C	CB12		271	RL D	
016E	CB13		272	RL E	
0170	CB14		273	RL H	
0172	CB15		274	RL L	
0174	CB16		275	RL M	
0176	CB17		276	RL A	
0178	CB18		277	RL B	
017A	CB19		278	RL C	
017C	CB1A		279	RL D	
017E	CB1B		280	RL E	
0180	CB1C		281	RL H	
0182	CB1D		282	RL L	
0184	CB1E		283	RL M	
0186	CB1F		284	RL A	
0188	CB20		285	SLA B	
018A	CB21		286	SLA C	
018C	CB22		287	SLA D	
018E	CB23		288	SLA E	
0190	CB24		289	SLA H	
0192	CB25		290	SLA L	

ADR	OBJ-KODE	M	ZEILE	QUELL-KODE
027C	CBA2		407	RES 4,D
027E	CBA3		408	RES 4,E
0280	CBA4		409	RES 4,H
0282	CBA5		410	RES 4,L
0284	CBA6		411	RES 4,M
0286	CBA7		412	RES 4,A
0288	CBA8		413	RES 5,B
028A	CBA9		414	RES 5,C
028C	CBA1		415	RES 5,D
028E	CBA2		416	RES 5,E
0290	CBA3		417	RES 5,H
0292	CBA4		418	RES 5,L
0294	CBA5		419	RES 5,A
0296	CBA6		420	RES 6,B
0298	CBA7		421	RES 6,C
029A	CBA8		422	RES 6,D
029C	CBA9		423	RES 6,E
029E	CBA1		424	RES 6,H
02A0	CBA2		425	RES 6,L
02A2	CBA3		426	RES 6,M
02A4	CBA4		427	RES 6,A
02A6	CBA5		428	RES 6,B
02A8	CBA6		429	RES 7,B
02AA	CBA7		430	RES 7,C
02AC	CBA8		431	RES 7,D
02AE	CBA9		432	RES 7,E
02B0	CBA1		433	RES 7,H
02B2	CBA2		434	RES 7,L
02B4	CBA3		435	RES 7,M
02B6	CBA4		436	RES 7,A
02B8	CBA5		437	RES 7,B
02BA	CBA6		438	RES 7,C
02BC	CBA7		439	RES 7,D
02BE	CBA8		440	RES 7,E
02C0	CBA9		441	RES 7,H
02C2	CBA1		442	RES 7,L
02C4	CBA2		443	RES 7,M
02C6	CBA3		444	RES 7,A
02C8	CBA4		445	RES 7,B
02CA	CBA5		446	RES 7,C
02CC	CBA6		447	RES 7,D
02CE	CBA7		448	RES 7,E
02D0	CBA8		449	RES 7,H
02D2	CBA9		450	RES 7,L
02D4	CBA1		451	RES 7,M
02D6	CBA2		452	RES 7,A
02D8	CBA3		453	RES 7,B
02DA	CBA4		454	RES 7,C
02DC	CBA5		455	RES 7,D
02DE	CBA6		456	RES 7,E
02E0	CBA7		457	RES 7,H
02E2	CBA8		458	RES 7,L
02E4	CBA9		459	RES 7,M
02E6	CBA1		460	RES 7,A
02E8	CBA2		461	RES 7,B
02EA	CBA3		462	RES 7,C
02EC	CBA4		463	RES 7,D
02EE	CBA5		464	RES 7,E

ADR	OBJ-KODE	M	ZEILE	QUELL-KODE
0208	CBA68		349	BIT 5,B
020A	CBA69		350	BIT 5,C
020C	CBA70		351	BIT 5,D
020E	CBA71		352	BIT 5,E
0210	CBA72		353	BIT 5,H
0212	CBA73		354	BIT 5,L
0214	CBA74		355	BIT 5,M
0216	CBA75		356	BIT 5,A
0218	CBA76		357	BIT 5,B
021A	CBA77		358	BIT 5,C
021C	CBA78		359	BIT 5,D
021E	CBA79		360	BIT 5,E
0220	CBA80		361	BIT 5,H
0222	CBA81		362	BIT 5,L
0224	CBA82		363	BIT 5,M
0226	CBA83		364	BIT 5,A
0228	CBA84		365	BIT 5,B
022A	CBA85		366	BIT 5,C
022C	CBA86		367	BIT 5,D
022E	CBA87		368	BIT 5,E
0230	CBA88		369	BIT 5,H
0232	CBA89		370	BIT 5,L
0234	CBA90		371	BIT 5,M
0236	CBA91		372	BIT 5,A
0238	CBA92		373	RES 0,B
023A	CBA93		374	RES 0,C
023C	CBA94		375	RES 0,D
023E	CBA95		376	RES 0,E
0240	CBA96		377	RES 0,H
0242	CBA97		378	RES 0,L
0244	CBA98		379	RES 0,M
0246	CBA99		380	RES 0,A
0248	CBA100		381	RES 0,B
024A	CBA101		382	RES 0,C
024C	CBA102		383	RES 0,D
024E	CBA103		384	RES 0,E
0250	CBA104		385	RES 0,H
0252	CBA105		386	RES 0,L
0254	CBA106		387	RES 0,M
0256	CBA107		388	RES 0,A
0258	CBA108		389	RES 0,B
025A	CBA109		390	RES 0,C
025C	CBA110		391	RES 0,D
025E	CBA111		392	RES 0,E
0260	CBA112		393	RES 0,H
0262	CBA113		394	RES 0,L
0264	CBA114		395	RES 0,M
0266	CBA115		396	RES 0,A
0268	CBA116		397	RES 0,B
026A	CBA117		398	RES 0,C
026C	CBA118		399	RES 0,D
026E	CBA119		400	RES 0,E
0270	CBA120		401	RES 0,H
0272	CBA121		402	RES 0,L
0274	CBA122		403	RES 0,M
0276	CBA123		404	RES 0,A
0278	CBA124		405	RES 0,B
027A	CBA125		406	RES 0,C

ADR	OBJ-KODE	M ZEILE	H ZEILE	QUELL-KODE	ADR	OBJ-KODE	M ZEILE	H ZEILE	QUELL-KODE
02F0	C8DC	465	SET 3,H		0378	D07A11	523	LD (IX+IND),H	
02F2	C8DD	466	SET 3,L		037B	D07511	524	LD (IX+IND),L	
02F4	C8DE	467	SET 3,H		037E	D07211	525	LD (IX+IND),A	
02F6	C8DF	468	SET 3,A		0381	D07E11	526	LD *,(IX+IND)	
02F8	C8E0	469	SET 4,B		0384	D08E11	527	ADD (IX+IND)	
02FA	C8E1	470	SET 4,C		0387	D08E11	528	ADC (IX+IND)	
02FC	C8E2	471	SET 4,D		038A	DD9611	529	SUB (IX+IND)	
02FE	C8E3	472	SET 4,E		038D	DD9E11	530	SEC (IX+IND)	
0300	C8E4	473	SET 4,H		0390	DDA611	531	AND (IX+IND)	
0302	C8E5	474	SET 4,L		0393	DDAE11	532	XOR (IX+IND)	
0304	C8E6	475	SET 4,M		0396	DDB611	533	OR (IX+IND)	
0306	C8E7	476	SET 4,A		0399	DDBE11	534	CHP (IX+IND)	
0308	C8E8	477	SET 5,B		039C	DDE1	535	POP IX	
030A	C8E9	478	SET 5,C		039E	DDE3	536	EX (SP),IX	
030C	C8EA	479	SET 5,D		03A0	DDE5	537	PUSH IX	
030E	C8EB	480	SET 5,E		03A2	DDE9	538	JMP (IX)	
0310	C8EC	481	SET 5,H		03A4	DDF9	539	LD SP,IX	
0312	C8ED	482	SET 5,L		03A6	DDCB1106	540	RLC (IX+IND)	
0314	C8EE	483	SET 5,M		03AA	DDCB110E	541	RRC (IX+IND)	
0316	C8EF	484	SET 5,A		03AE	DDCB1116	542	RL (IX+IND)	
0318	C8F0	485	SET 6,D		03B2	DDCB111E	543	RR (IX+IND)	
031A	C8F1	486	SET 6,C		03B6	DDCB1126	544	SJA (IX+IND)	
031C	C8F2	487	SET 6,D		03BA	DDCB112E	545	SRA (IX+IND)	
031E	C8F3	488	SET 6,E		03BE	DDCB113E	546	SRL (IX+IND)	
0320	C8F4	489	SET 6,H		03C2	DDCB114E	547	BIT 0,(IX+IND)	
0322	C8F5	490	SET 6,L		03C6	DDCB115E	548	BIT 1,(IX+IND)	
0324	C8F6	491	SET 6,M		03CA	DDCB115E	549	BIT 2,(IX+IND)	
0326	C8F7	492	SET 6,A		03CE	DDCB115E	550	BIT 3,(IX+IND)	
0328	C8F8	493	SET 7,C		03D2	DDCB1166	551	BIT 4,(IX+IND)	
032A	C8F9	494	SET 7,B		03D6	DDCB116E	552	BIT 5,(IX+IND)	
032C	C8FA	495	SET 7,D		03DA	DDCB1176	553	BIT 6,(IX+IND)	
032E	C8FB	496	SET 7,E		03DE	DDCB117E	554	BIT 7,(IX+IND)	
0330	C8FC	497	SET 7,H		03E2	DDCB1186	555	RES 0,(IX+IND)	
0332	C8FD	498	SET 7,L		03E6	DDCB118E	556	RES 1,(IX+IND)	
0334	C8FE	499	SET 7,M		03EA	DDCB1196	557	RES 2,(IX+IND)	
0336	C8FF	500	SET 7,A		03EE	DDCB119E	558	RES 3,(IX+IND)	
0338	D009	501	ADD IX,BC		03F2	DDCB11A6	559	RES 4,(IX+IND)	
033A	D019	502	ADD IX,DE		03F6	DDCB11AE	560	RES 5,(IX+IND)	
033C	D021B605	503	LD IX,HN		03FA	DDCB11B6	561	RES 6,(IX+IND)	
0340	D0228605	504	LD (NN),IX		03FE	DDCB11C6	562	RES 7,(IX+IND)	
0344	D023	505	INC IX		0402	DDCB11CE	563	SET 0,(IX+IND)	
0346	D029	506	ADD IX,IX		0404	DDCB11DE	564	SET 1,(IX+IND)	
0348	D02A8605	507	LD IX,(FH)		040E	DDCB11DE	565	SET 2,(IX+IND)	
034C	D02B	508	DEC IX		0412	DDCB11E6	566	SET 3,(IX+IND)	
034E	DD3411	509	INC (IX+IND)		0416	DDCB11E6	567	SET 4,(IX+IND)	
0351	DD3511	510	DEC (IX+IND)		041A	DDCB11E6	568	SET 5,(IX+IND)	
0354	DD361133	511	LD (IX+IND),N		041E	DDCB11FE	569	SET 6,(IX+IND)	
0358	DD39	512	ADD IX,SP		0422	DDCB11FE	570	SET 7,(IX+IND)	
035A	DD4611	513	LD B,(IX+IND)		ED40	DD40	571	AH B	
035D	DD4E11	514	LD C,(IX+IND)		ED41	DD41	572	BIT B	
0360	DD5611	515	LD D,(IX+IND)		ED42	DD42	573	SBC HL,BC	
0363	DD5E11	516	LD E,(IX+IND)		ED48	ED48	574	LD (NN),BC	
0366	DD6611	517	LD H,(IX+IND)		ED438605	ED438605	574	LD (NN),BC	
0369	DD6E11	518	LD L,(IX+IND)		ED44	ED44	575	NEG	
036C	DD7011	519	LD (IX+IND),B		ED45	ED45	576	RETN	
036F	DD7111	520	LD (IX+IND),C		ED46	ED46	577	JMO	
0372	DD7211	521	LD (IX+IND),D		ED47	ED47	578	LD I,A	
0375	DD7311	522	LD (IX+IND),E		ED48	ED48	579	JR C	
					ED49	ED49	580	OUT C	

BEFEHLSLISTE

ADR	OBJ-KODE	M	ZEILE	QUELL-KODE	BEFEHLSLISTE
04CB	FD6E11		642	LD L,(IY+IND)	
04CE	FD7011		643	LD (IY+IND),B	
04D1	FD7111		644	LD (IY+IND),C	
04D4	FD7211		645	LD (IY+IND),D	
04D7	FD7311		646	LD (IY+IND),E	
04DA	FD7411		647	LD (IY+IND),H	
04DD	FD7511		648	LD (IY+IND),L	
04E0	FD7711		649	LD (IY+IND),A	
04E3	FD7E11		650	LD A,(IY+IND)	
04E6	FD8611		651	ADD (IY+IND)	
04E9	FD8E11		652	ADC (IY+IND)	
04F1	FD9611		653	SUB (IY+IND)	
04F4	FD9E11		654	SBC (IY+IND)	
04F7	FDA611		655	AND (IY+IND)	
04FA	FDAA11		656	XOR (IY+IND)	
04FB	FDAB11		657	OR (IY+IND)	
04FC	FDAC11		658	OR (IY+IND)	
04FE	FDE1		659	POP IY	
0500	FDE3		660	EX (SP),IY	
0502	FDE5		661	PUSH IY	
0504	FDE9		662	JMP (IY)	
0506	FD9		663	LD SP,IY	
0508	FDCB1106		664	RLC (IY+IND)	
050C	FDCB110E		665	RRC (IY+IND)	
0510	FDCB1116		666	RL (IY+IND)	
0514	FDCB111E		667	RR (IY+IND)	
0518	FDCB1126		668	SLA (IY+IND)	
051C	FDCB112E		669	SRA (IY+IND)	
0520	FDCB113E		670	SRL (IY+IND)	
0524	FDCB1146		671	BIT 0,(IY+IND)	
0528	FDCB114E		672	BIT 1,(IY+IND)	
052C	FDCB1156		673	BIT 2,(IY+IND)	
0530	FDCB115E		674	BIT 3,(IY+IND)	
0534	FDCB1166		675	BIT 4,(IY+IND)	
0538	FDCB116E		676	BIT 5,(IY+IND)	
053C	FDCB1176		677	BIT 6,(IY+IND)	
0540	FDCB117E		678	BIT 7,(IY+IND)	
0544	FDCB1186		679	RES 0,(IY+IND)	
0548	FDCB118E		680	RES 1,(IY+IND)	
054C	FDCB1196		681	RES 2,(IY+IND)	
0550	FDCB119E		682	RES 3,(IY+IND)	
0554	FDCB11A6		683	RES 4,(IY+IND)	
0558	FDCB11AE		684	RES 5,(IY+IND)	
055C	FDCB11B6		685	RES 6,(IY+IND)	
0560	FDCB11BE		686	RES 7,(IY+IND)	
0564	FDCB11C6		687	SET 0,(IY+IND)	
0568	FDCB11CE		688	SET 1,(IY+IND)	
056C	FDCB11D6		689	SET 2,(IY+IND)	
0570	FDCB11DE		690	SET 3,(IY+IND)	
0574	FDCB11E6		691	SET 4,(IY+IND)	
0578	FDCB11EE		692	SET 5,(IY+IND)	
057C	FDCB11F6		693	SET 6,(IY+IND)	
0580	FDCB11FE		694	SET 7,(IY+IND)	
0584	ED70		695	INP	FLAGS WERDEN GELADEN
0586			696	NN	DEFS 2
			697	IND	EQU
			698	H	EQU
			699	N	EQU
			700	DIS	EQU
			701	END	

BEFEHLSLISTE

ADR	OBJ-KODE	M	ZEILE	QUELL-KODE	BEFEHLSLISTE
0438	ED4A		581	ADC HL,BC	
ED4B8605				LD BC,(RH)	
ED4D	RETI		583	RETI	
0440	ED50		584	IN D	
0442	ED51		585	OUT D	
0444	ED52		586	SBC HL,DE	
0446	ED5318605		587	LD (NN),DE	
044A	ED56		588	IN I	
044C	ED57		589	LD A,I	
044E	ED58		590	IN E	
0450	ED59		591	OUT E	
0452	ED5A		592	ADC HL,DE	
0454	ED5B18605		593	LD DE,(NN)	
0458	ED5E		594	IN H	
045A	ED60		595	IN H	
045C	ED61		596	OUT H	
045E	ED62		597	SBC HL,HL	
0460	ED66		598	RFD	
0462	ED68		599	IN L	
0464	ED69		600	OUT L	
0466	ED6A		601	ACC HL,HL	
0468	ED6F		602	RLD	
046A	ED72		603	SBC HL,SP	
046C	ED738605		604	LD (NH),SP	
0470	ED78		605	IN A	
0472	ED79		606	OUT A	
0474	ED7A		607	ADC HL,SP	
0476	ED7B8605		608	LD SP,(RH)	
047A	EDA0		609	LDI	
047C	EDA1		610	CPI	
047E	EDA2		611	IN I	
0480	EDA3		612	OUTI	
0482	EDA8		613	LDD	
0484	EDA9		614	CPD	
0486	EDA4		615	IND	
0488	EDA8		616	OUTD	
048A	EDB0		617	DIR	
048C	EDB1		618	CPRI	
048E	EDB2		619	INR	
0490	EDB3		620	INR	
0492	EDB8		621	LDDR	
0494	EDB9		622	CPDR	
0496	EDBA		623	INDR	
0498	EDBB		624	OTDR	
049A	FD09		625	ADD IY,BC	
049C	FD19		626	ADD IY,DE	
049E	FD218605		627	LD IY,NN	
04A2	FD228605		628	LD (RH),IY	
04A6	FD23		629	INC IY	
04A8	FD29		630	ADD IY,IY	
04AA	FD2A8605		631	LD IY,(NN)	
04AE	FD2B		632	DEC IY	
04B0	FD3411		633	INC (IY+IND)	
04B3	FD3511		634	DEC (IY+IND)	
04B6	FD361133		635	LD (IY+IND),H	
04BA	FD39		636	ADD IY,SP	
04BC	FD4611		637	LD B,(IY+IND)	
04BF	FD4E11		638	LD C,(IY+IND)	
04C2	FD5611		639	LD D,(IY+IND)	
04C5	FD5E11		640	LD E,(IY+IND)	
04C8	FD6611		641	LD H,(IY+IND)	

### A.3. Monitorbetriebsprogramm des Lernsystems

U880D - LERNSYSTEM (V.2)	LS880	SEITE 1	U880D - LERNSYSTEM (V.2)	LS880	SEITE 2
ADR	OBJ-KODE	M	ZEILE	QUELL-KODE	ASM
0000	31C287		1	;	59
0003	0637		2	;	60
0005	21C387	R	3	;	61
0008	3600		4	LD SP,87C2H ; BEGINN DES MONITOR	62
000A	23		5	LD B,55 ; ANFANG DES STACK	63
000B	10FB		6	LD HL,55 ; 55 HILFSZELLEN LOESCHEN	64
000D	0606		7	LD HL,SPC ;	65
000F	21FA87	R	8	LD HL,H,O ;	66
0012	363F		9	INC HL ;	67
0014	23		10	DJNZ LL	68
0015	1804		11	LD B,6	69
0018	C39D05	R	12	LD HL,LEDPUF	70
001B	10F5		13	LD HL,ANZEIGE AUF NULL	71
001D	3EDE		14	LD M,03FH	72
001F	D37D		15	INC HL	73
0021	212C06	R	16	JR FORT1	74
0024	7C		17	ORG 18H ;	75
0025	ED47		18	ORG 18H ;	76
0027	7D		19	JMP MONI ; RESTART BEI SOFT.-BREAK	77
0028	D3BC		20	FORT1:	78
002A	3E97		21	DJNZ LLD	79
002C	D3BD		22	LD A,0DEH ; USART BETRIEBSART-KODE	80
002E	3E4D		23	OUT USRTC0 ;	81
0030	D3BD		24	LD HL,INTAB	82
0032	3E17		25	LD A,H	83
0034	D3BC		26	LD I,A ; OBERE HAELFTE I N I	84
0036	1802		27	LD A,L ;	85
0038	18C6		28	LD A,L ;	86
003A	E95E		29	OUT CTC0 ; UNTERE HAELFTE	87
003C	FB		30	OUT CTC0 ;	88
003D	21E587		31	LD A,97H ; IN I-REGISTER CTC	89
0040	22F167		32	OUT CTC1 ; STEUERWORT KANAL 1	90
0043	CB4900	R	33	LD A,77 ; ZEITKONST. DISPLAY	91
0046	C3F500	R	34	OUT CTC1 ;	92
0049	CDCC00	R	35	LD A,17H ; CTC BETRIEBSART-KODE	93
004C	21D0D7		36	OUT CTC0 ;	94
004E	11E167		37	JR FORT2	95
0052	010400		38	ORG 38H ;	96
0055	EDB0		39	JR RESET	97
			40	FORT2:	98
			41	LD I2 ;	99
			42	EI ; DISPLAY FREIGEBEN	100
			43	ATAS1 ; EINGABEREKETTE NEU	101
			44	LD HL,TR	102
			45	LD (TR2),HL	103
			46	TAST ; EINGABEREKETTE WEITER	104
			47	CALL TABFR ; TASTENABFRAGE	105
			48	JMP SAVE ; TASTE ERKANNT	106
			49	JMP SAVE ;	107
			50	TABFR:	108
			51	UP ; TASTENABFRAGE,	109
			52	CALL HSEK ; VERT NACH A	110
			53	CALL HSEK ; VERZOECERUNG	111
			54	LD HL,T1H ; FUER PRELLUNTERDRUECKUNG	112
			55	LD DE,T1A ;ADR. TASTENREG.-NEU	113
			56	LD BC,4 ;ADR. TASTENREG.-ALT	114
			57	LD R,4 ;	115
			58	LDIR ;(TR-NEU) NACH (TR-ALT)	116
			59	;	
			60	;	
			61	;	
			62	LD IX,T1H	
			63	LD BC,88DH	
			64	LD BC,88DH ;R:=ZEILENSELEKTOR	
			65	LD BC,88DH ;C:=TASTENADR.	
			66	LD BC,88DH	
			67	LD BC,88DH	
			68	LD BC,88DH	
			69	LD BC,88DH	
			70	LD BC,88DH ;RESTART BEI NMI (STEP)	
			71	PUSH AF ;USER-AF RETTEN	
			72	LD (SL),HL ;USER-HL RETTEN	
			73	POP HL ;	
			74	LD (SF),HL ;	
			75	POP HL ;	
			76	LD (SPC),HL ;USER-PC RETTEN	
			77	LD (SSP),SP ;	
			78	LD HL,MONI ;	
			79	PUSH HL ;SPRUNG NACH MORIN	
			80	RETH ;	
			81	RETH ;	
			82	FORT3:	
			83	JNC IX ;	
			84	INC B ;	
			85	JRNC TA1 ;	
			86	LD IX,T1H ;	
			87	LD B,3 ;	
			88	LADD ;	
			89	LADD ;	
			90	ADD (IX) ;	
			91	JNC IX ;	
			92	DJNZ LADD ;	
			93	JRNC FFU ;	
			94	JRZ TABFR ;	
			95	JRZ TABFR ;	
			96	LD H,1 ;	
			97	LD B,8 ;	
			98	LCP ;	
			99	CHP H ;	
			100	JRZ TOK ;	
			101	SLA H ;	
			102	DJNZ LCP ;	
			103	FU:CALL FFSET ;FEHLERSCHWALZISATION	
			104	FU:CALL FFSET ;UHD NEUE TASTE	
			105	JR TABFR ;	
			106	STNWFJ ; WARTEN AUF STORNO-TASTE	
			107	CALL FFSET ;	
			108	CALL TABFR ;TASTENABFRAGE STORNO	
			109	CRP 17 ;STORNO-TASTE ;	
			110	JPZ STORN ;WENN JA STORNIEREN	
			111	JR STORN+3 ;WENN NEIN NOCH EINMAL	
			112	FFSET:	
			113	UP-SETZEN DER ERROR-LED	
			114	LD HL,F987	
			115	SET 7,H	
			116	RET	

U880D - LERNSYSTEM (V.2)	LS880	SEITE	SEITE
ADR	OBJ-KODE M ZEILE QUELL-KODE	ASM U880	ASM U880
00B6	DD21E187 R	117	175
00B7	DD21E188 R	118	176
00B8	FD21DD87 R	119	177
00B9	FE01	120	178
00C0	0604	121	179
00C1	0604	122	180
00C2	FD7E00	123	181
00C3	DDAE00	124	182
00C4	0604	125	183
00C5	0604	126	184
00C6	0604	127	185
00C7	0604	128	186
00C8	0604	129	187
00C9	0604	130	188
00D0	0604	131	189
00D1	0604	132	190
00D2	0604	133	191
00D3	0604	134	192
00D4	0604	135	193
00D5	0604	136	194
00D6	0604	137	195
00D7	0604	138	196
00D8	0604	139	197
00D9	0604	140	198
00E0	0604	141	199
00E1	0604	142	200
00E2	0604	143	201
00E3	0604	144	202
00E4	0604	145	203
00E5	0604	146	204
00E6	0604	147	205
00E7	0604	148	206
00E8	0604	149	207
00E9	0600	150	208
00EB	CB1F	151	209
00ED	3803	152	210
00EF	04	153	211
00F0	18F9	154	212
00F1	0604	155	213
00F2	79	156	214
00F3	80	157	215
00F4	C9	158	216
00F5	0604	159	217
00F6	0604	160	218
00F7	0604	161	219
00F8	0604	162	220
00F9	0604	163	221
00FA	22F187	164	222
00FB	11F287	165	223
00FC	0604	166	224
00FD	0604	167	225
00FE	0604	168	226
00FF	0604	169	227
0100	A7	170	228
0101	ED52	171	229
0102	289E	172	230
0103	0604	173	231
0104	0604	174	232
0105	FE10	175	233
0106	2846	176	234
0107	FE11	177	235
0108	CA6702	178	236
0109	0604	179	237
0110	FE12	180	238
0111	CA6802	181	239
0112	FE13	182	240
0113	CA6F02	183	241
0114	FE14	184	242
0115	2807	185	243
0116	FE15	186	244
0117	0604	187	245
0118	2829	188	246
0119	0604	189	247
0120	C3A300	190	248
0121	0604	191	249
0122	2AC387	192	250
0123	23	193	251
0124	0604	194	252
0125	22C387	195	253
0126	7E	196	254
0127	0604	197	255
0128	0604	198	256
0129	0604	199	257
0130	32FA87	200	258
0131	7E	201	259
0132	0604	202	260
0133	0604	203	261
0134	CD4001	204	262
0135	CD6802	205	263
0136	32FB87	206	264
0137	0604	207	265
0138	32FB87	208	266
0139	C38F02	209	267
0140	0604	210	268
0141	0604	211	269
0142	0604	212	270
0143	0604	213	271
0144	0604	214	272
0145	0604	215	273
0146	0604	216	274
0147	0604	217	275
0148	0604	218	276
0149	2AC387	219	277
0150	2B	220	278
0151	18D8	221	279
0152	0604	222	280
0153	0604	223	281
0154	0604	224	282
0155	0604	225	283
0156	0604	226	284
0157	0604	227	285
0158	0604	228	286
0159	0604	229	287
0160	0604	230	288
0161	0604	231	289
0162	0604	232	290
0163	0604	233	291
0164	0604	234	292
0165	0604	235	293
0166	0604	236	294
0167	0604	237	295
0168	0604	238	296
0169	0604	239	297
0170	0604	240	298
0171	0604	241	299
0172	0604	242	300
0173	0604	243	301
0174	0604	244	302
0175	0604	245	303
0176	0604	246	304
0177	0604	247	305
0178	0604	248	306
0179	0604	249	307
0180	0604	250	308
0181	0604	251	309
0182	0604	252	310
0183	0604	253	311
0184	0604	254	312
0185	0604	255	313
0186	0604	256	314
0187	0604	257	315
0188	0604	258	316
0189	0604	259	317
0190	0604	260	318
0191	0604	261	319
0192	0604	262	320
0193	0604	263	321
0194	0604	264	322
0195	0604	265	323
0196	0604	266	324
0197	0604	267	325
0198	0604	268	326
0199	0604	269	327
0200	0604	270	328
0201	0604	271	329
0202	0604	272	330
0203	0604	273	331
0204	0604	274	332
0205	0604	275	333
0206	0604	276	334
0207	0604	277	335
0208	0604	278	336
0209	0604	279	337
0210	0604	280	338
0211	0604	281	339
0212	0604	282	340
0213	0604	283	341
0214	0604	284	342
0215	0604	285	343
0216	0604	286	344
0217	0604	287	345
0218	0604	288	346
0219	0604	289	347
0220	0604	290	348
0221	0604	291	349
0222	0604	292	350
0223	0604	293	351
0224	0604	294	352
0225	0604	295	353
0226	0604	296	354
0227	0604	297	355
0228	0604	298	356
0229	0604	299	357
0230	0604	300	358
0231	0604	301	359
0232	0604	302	360
0233	0604	303	361
0234	0604	304	362
0235	0604	305	363
0236	0604	306	364
0237	0604	307	365
0238	0604	308	366
0239	0604	309	367
0240	0604	310	368
0241	0604	311	369
0242	0604	312	370
0243	0604	313	371
0244	0604	314	372
0245	0604	315	373
0246	0604	316	374
0247	0604	317	375
0248	0604	318	376
0249	0604	319	377
0250	0604	320	378
0251	0604	321	379
0252	0604	322	380
0253	0604	323	381
0254	0604	324	382
0255	0604	325	383
0256	0604	326	384
0257	0604	327	385
0258	0604	328	386
0259	0604	329	387
0260	0604	330	388
0261	0604	331	389
0262	0604	332	390
0263	0604	333	391
0264	0604	334	392
0265	0604	335	393
0266	0604	336	394
0267	0604	337	395
0268	0604	338	396
0269	0604	339	397
0270	0604	340	398
0271	0604	341	399
0272	0604	342	400
0273	0604	343	401
0274	0604	344	402
0275	0604	345	403
0276	0604	346	404
0277	0604	347	405
0278	0604	348	406
0279	0604	349	407
0280	0604	350	408
0281	0604	351	409
0282	0604	352	410
0283	0604	353	411
0284	0604	354	412
0285	0604	355	413
0286	0604	356	414
0287	0604	357	415
0288	0604	358	416
0289	0604	359	417
0290	0604	360	418
0291	0604	361	419
0292	0604	362	420
0293	0604	363	421
0294	0604	364	422
0295	0604	365	423
0296	0604	366	424
0297	0604	367	425
0298	0604	368	426
0299	0604	369	427
0300	0604	370	428
0301	0604	371	429
0302	0604	372	430
0303	0604	373	431
0304	0604	374	432
0305	0604	375	433
0306	0604	376	434
0307	0604	377	435
0308	0604	378	436
0309	0604	379	437
0310	0604	380	438
0311	0604	381	439
0312	0604	382	440
0313	0604	383	441
0314	0604	384	442
0315	0604	385	443
0316	0604	386	444

U880D - LERNSYSTEM (V.2)		LS880		SEITE 5		
ADR	OBJ-KODE M ZEILE	QUELL-KODE	ASH	U880	ASK U880	
016E	CACB03	R	233	JPZ STORE ;	WENN STORE-TASTE	
0171	FE19		234	CMP 25		
0173	CA7504	R	235	JPZ LOAD	;	WENN LOAD-TASTE
0176	FE1D		236	CMP 29		
0178	CAE204	R	237	JPZ FILL	;	WENN FILL-TASTE
017B	FE1E		238	CMP 30		
017D	CAA300	R	239	JPZ STRNVT ;	WENN TPI-TASTE	
0180	FE1B		240	CMP 27		
0182	CAA300	R	241	JPZ STRNVT ;	WENN TPO-TASTE	
0185	FE00		242	CMP 0		
0187	CAA300	R	243	JPZ STRNVT ;	WENN PRO-TASTE	
018A	FE01		244	CMP 1		
018C	CAA300	R	245	JPZ STRNVT ;	WENN CHP-TASTE	
018F	FE02		246	CMP 2		
0191	CAA300	R	247	JPZ STRNVT ;	WENN TRF-TASTE	
0194	CA3300	R	248	JMP STRNVT ;	WENN KEINE FUNKTIONSTASTE	
			249			
			250			
0197	010000		251	DISSET ;	DISP. UND SET GEMEINSAM	
019A	DD21E687	R	252	LD BC,0		
019E	DD7E00		253	LD IX,TR+1 ;	AUF 2. TASTE IN TR	
01A1	FE1B		254	LD A,(IX)		
01A3	CAD401	R	255	CMP 27 ;	WENN TASTE ?	
01A6	D604		257	SUB 4		
01A8	DFAA01	R	258	JPC C03	;	WENN I-REG.-TASTE
01AB	DD7E00		259	LD A,(IX)		
01AE	D608		260	SUB 8		
01B0	3857		261	REG8 ;		
01B2	DD7E00		262	LD A,(IX)		
01B5	D610		263	SUB 16		
01B7	D2A300	R	264	JPHC STRNVT ;	WENN TASTENWERT >15	
01BA	DD5E00		265	LD E,(IX)	;	8-BIT-REGISTER
01BD	1600		266	LD D,0		
01BF	DD23		267	INC IX		
01C1	DD7E00		268	LD A,(IX)	;	3. TASTE IN TR
01C4	FE1E		269	CMP 10	;	1. TASTE ?
01C6	2802		270	JZ2 AUXS	;	WENN 1.-REGISTER
01C8	1811		271	JR SET2		
			272			
			273	AUXS ;		
01CA	DD23		274	INC IX	;	4. TASTE IN TR
01CC	010800		275	LD BC,8	;	ANTEIL FUER 1.-REGISTER
01CF	DD7E00		276	LD A,(IX)		
01D2	1807		277	JR SET2		
			278			
			279	MEMSET ;		
01D4	2AC387	R	280	LD HL,(SPC)		
01D7	DD23		281	INC IX ;	AUSGLEICH	
01D9	180E		282	JR SET3		
			283			
			284	SET2 ;		
01DB	21DD05	R	285	LD HL,UNCOD-8	;	ADR.-ADR. DER UNKODETAFEL
01DE	09		286	ADD HL,BC	;	ANTEIL FUER 1.-REG.
01E0	19		288	ADD HL,DE	;	ANTEIL FUER 8-BIT-REG.
01E1	D5		289	PUSH DE		
			290	LD D,0		
			291	LD E,M		
			292	LD HL,SPC		
			293	ADD HL,DE		
			294	POP DE		
			295	SET3 ;		
			296	LD A,(TR)	;	1. TASTE AUS TR HOLEN
			297	CMP 22 ;	1. TASTE IST DISP.-TASTE ?	
			298	JRZ DIS2	;	WENN DISP.-TASTE
			299			
			300	Gg ;		
			301	EX DE,HL		
			302	CALL DAT2		
			303	EX DE,HL		
			304	LD M,E		
			305	DIS2 ;		
			306	LD A,H		
			307	JMP LEDATA		
			308			
			309	C03 ;		
			310	LD A,(IX)		
			311	CMP 3		
			312	JPUZ STRNVT ;	FEHLER	
			313	LD HL,SI		
			314	INC IX ;	AUSGLEICH	
			315	JR SET3		
			316			
			317	C47 ;	16-BIT REGISTER	
			318	LD HL,SPC		
			319	LD A,(IX)		
			320	SUB 4		
			321	ADD A	;	VERDOPPELN
			322	LD E,A		
			323	LD D,0		
			324	ADD HL,DE	;	ADR.-DES L-BYTES
			325	LD A,(TR)	;	1. TASTE IN TR
			326	CMP 22	;	DISP.-TASTE ?
			327	JRZ DIS4 ;	WENN NUR DISP.-TASTE	
			328			
			329	EX DE,HL		
			330	INC IX		
			331	CALL DAT4		
			332	EX DE,HL		
			333	LD M,E		
			334	INC HL		
			335	LD M,D		
			336	DEC HL		
			337	DISM ;		
			338	LD E,M		
			339	INC HL		
			340	LD D,M		
			341	EX DE,HL		
			342	JR LEDADR		
			343			
			344	DAT4 ;	UP: 4 HEX-TASTEN NACH HL	
			345	;	++ FUHRUNGSMULLEN-AUTOMAIK	
			346	LD B,5 ;	5. DUCHG MUSS EX-TASTE SEIN	
			347	LD I,IJHPI		
			348			
			349			
			350			
			351			
			352			
			353			
			354			
			355			
			356			
			357			
			358			
			359			
			360			
			361			
			362			
			363			
			364			
			365			
			366			
			367			
			368			
			369			
			370			
			371			
			372			
			373			
			374			
			375			
			376			
			377			
			378			
			379			
			380			
			381			
			382			
			383			
			384			
			385			
			386			
			387			
			388			
			389			
			390			
			391			
			392			
			393			
			394			
			395			
			396			
			397			
			398			
			399			
			400			
			401			
			402			
			403			
			404			
			405			
			406			
			407			
			408			
			409			
			410			
			411			
			412			
			413			
			414			
			415			
			416			
			417			
			418			
			419			
			420			
			421			
			422			
			423			
			424			
			425			
			426			
			427			
			428			
			429			
			430			
			431			
			432			
			433			
			434			
			435			
			436			
			437			
			438			
			439			
			440			
			441			
			442			
			443			
			444			
			445			
			446			
			447			
			448			
			449			
			450			
			451			
			452			
			453			
			454			
			455			
			456			
			457			
			458			
			459			
			460			
			461			
			462			
			463			
			464			
			465			
			466			
			467			
			468			
			469			
			470			
			471			
			472			
			473			
			474			
			475			
			476			
			477			
			478			
			479			
			480			
			481			
			482			
			483			
			484			
			485			
			486			
			487			
			488			
			489			
			490			
			491			
			492			
			493			
			494			
			495			
			496			
			497			
			498			
			499			
			500			
			501			
			502			
			503			
			504			
			505			
			506			
			507			
			508			
			509			
			510			
			511			
			512			
			513			
			514			
			515			
			516			
			517			
			518			
			519			
			520			
			521			
			522			
			523			
			524			
			525			
			526			
			527			
			528			
			529			
			530			
			531			
			532			
			533			
			534			
			535			
			536			
			537			
			538			
			539			
			540			
			541			
			542			
			543			
			544			
			545			
			546			
			547			
			548			
			549			
			550			
			551			
			552			
			553			
			554			
			555			
			556			
			557			
			558			
			559			
			560			
			561			

ADR	OBJ-KODE	M	ZELLE	QUELL-KODE	ADR	OBJ-KODE	M	ZELLE	QUELL-KODE	ADR	OBJ-KODE	M	ZELLE	QUELL-KODE
0234	210000				0287	18AB		407	JR D24810	0287	18AB		407	JR D24810
0235	DD7E00							406					406	
0236	FE10							409					409	
0237	C8				0289	CD9F02	R	411	LEDATA: ;ANZEIGE INHALT A	0289	CD9F02	R	411	LEDATA: ;ANZEIGE INHALT A
0238	D610				028C	C33D00	R	412	JMP DISDAT	028C	C33D00	R	412	JMP ATAST
0239	302C							413					413	
0240	DD7E00				028F	CD9502	R	414	LEDADR: ;ANZEIGE INHALT HL	028F	CD9502	R	414	CALL DISADR
0241	FE10				0292	C33D00	R	416	JMP ATAST	0292	C33D00	R	416	JMP ATAST
0242	C8							417					417	
0243	C5							418					418	
0244	0604				0295	DD21FC87	R	420	DISADR: ;UP: ANZEIGE INHALT HL	0295	DD21FC87	R	420	LD IX,LEDPUF+2
0245	C825				0299	4D		421	LD C,L	0299	4D		421	LD C,L
0246	C8				029A	0602		422	LD B,2	029A	0602		422	LD B,2
0247	C5							423	LZIF:				423	LZIF:
0248	C825				029C	79		424	LD A,C	029C	79		424	LD A,C
0249	C2NC				029D	E60F		425	AND OFH	029D	E60F		425	AND OFH
0250	C814				029F	CDCE02	R	426	CALL SEGUH	029F	CDCE02	R	426	CALL SEGUH
0251	1810				02A2	DD7700		427	LD (IX),A	02A2	DD7700		427	LD (IX),A
0252	C814				02A5	79		428	LD A,C	02A5	79		428	LD A,C
0253	C813				02A6	CD4001	R	429	CALL SPL4	02A6	CD4001	R	429	CALL SPL4
0254	C812				02A9	CD6802	R	430	CALL SEGUH	02A9	CD6802	R	430	CALL SEGUH
0255	1808				02AC	DD23		431	INC IX	02AC	DD23		431	INC IX
0256	C814				02AE	DD7700		432	LD (IX),A	02AE	DD7700		432	LD (IX),A
0257	C813				02B1	DD23		433	LZIF	02B1	DD23		433	LZIF
0258	C812				02B3	4C		434	LD C,H	02B3	4C		434	LD C,H
0259	C811				02B6	C9		435	DJNZ LZIF ; ENDE UP ANZEIGE	02B6	C9		435	DJNZ LZIF ; ENDE UP ANZEIGE
0260	10E6							437					437	
0261	B5				02B7	CD202	R	438	STORM: ;STORMIEREN DER EINGABEK.	02B7	CD202	R	438	CALL FFCLR;LOESCHEN ERROR-LED
0262	6F				02BA	21F787	R	440	LD HL,EXZ ;EX-ZAEHLER-ADR.	02BA	21F787	R	440	LD HL,EXZ ;EX-ZAEHLER-ADR.
0263	79				02BD	3600		441	LD H,0 ;ZAEHLER LOESCHEN	02BD	3600		441	LD H,0 ;ZAEHLER LOESCHEN
0264	C1				02BF	C33D00	R	442	JMP ATAST ;NEUE EINGABEKETTE	02BF	C33D00	R	442	JMP ATAST ;NEUE EINGABEKETTE
0265	79							443					443	
0266	C1							444					444	
0267	4F							445					445	
0268	DD23				02C2	21F987	R	446	LD HL,FEHLER	02C2	21F987	R	446	LD HL,FEHLER
0269	10D4				02C5	CB8E		447	RES 7,H	02C5	CB8E		447	RES 7,H
0270	D1				02C7	C9		448	RET	02C7	C9		448	RET
0271	C3A300	R						449					449	
0272	D1							450	SEGUH: ;UP: 7-SEGN.-UHWANDLUNG				450	SEGUH: ;UP: 7-SEGN.-UHWANDLUNG
0273	18BB				02C8	11D505	R	451	LD DE,SEGN	02C8	11D505	R	451	LD DE,SEGN
0274	0603				02CB	83		452	ADD E	02CB	83		452	ADD E
0275	FD214E02	R			02CC	5F		453	LD E,A	02CC	5F		453	LD E,A
0276	18BB				02CD	1A		454	LD A,(DE)	02CD	1A		454	LD A,(DE)
0277	0609				02CE	C9		455	RET	02CE	C9		455	RET
0278	FD215202	R						456					456	
0279	18B3				02CF	DD21FA87	R	458	DISDAT: ;UP: ANZEIGE INHALT A	02CF	DD21FA87	R	458	LD IX,LEDPUF
0280	060B				02D3	4F		459	LD C,A	02D3	4F		459	LD C,A
0281	FD215A02	R			02D4	0601		460	LD B,1	02D4	0601		460	LD B,1
0282	0608				02D6	18C4		461	JR LZIF	02D6	18C4		461	JR LZIF
0283	FD215A02	R						462					462	
								463					463	
								464	START: ;START FUER USER-PROGRAMM				464	START: ;START FUER USER-PROGRAMM





U880D - LERNSYSTEM (V.2)	LS880	SEITE 13	U880D - LERNSYSTEM (V.2)	LS880	SEITE 14
ADR	OBJ-KODE	H ZELLE	ADR	OBJ-KODE	H ZELLE
0462	C5	697	04C3	755	
0463	0649	698	04C5	756	
0465	C5	699	04C6	757	
0466	06FF	700	04C7	758	
0468	C5	701	04C9	759	
0469	0680	702	04CA	760	
046B	10FE	703	04CB	761	
046D	C1	704	04CC	762	
046E	10F8	705	04CE	763	
0470	C1	706	04CF	764	
0471	10F2	707	04D1	765	
0473	C1	708	04D3	766	
0474	C9	709	04D5	767	
0475	08F05	710	04D7	768	
0478	FE02	711	04D9	769	
047A	CA9104	712	04DB	770	
047D	DD21E687	713	04DE	771	
0481	CD2E02	714	04DF	772	
0484	CD9502	715		773	
0487	EB	716		774	
0488	21E687	717		775	
048B	CD7105	718		776	
048E	C34300	719		777	
0491	21F787	720		778	
0494	3600	721		779	
0496	DD21E687	722		780	
049A	CD2E02	723		781	
049D	3516	724		782	
049F	D37D	725		783	
04A1	DB7C	726		784	
04A3	35B8	727		785	
04A5	D3BC	728		786	
04A7	3538	729		787	
04A9	32FA87	730		788	
04AC	97	731		789	
04AD	32FB87	732		790	
04B0	DB7D	733		791	
04B2	CB4F	734		792	
04B4	28FA	735		793	
04B6	CB6F	736		794	
04B8	2014	737		795	
04BA	CB5F	738		796	
04BC	200B	739		797	
04BE	CB67	740		798	
04CD	CC2904	741		799	
		742		800	
		743		801	
		744		802	
		745		803	
		746		804	
		747		805	
		748		806	
		749		807	
		750		808	
		751		809	
		752		810	
		753		811	
		754		812	
		755		813	
		756		814	
		757		815	
		758		816	
		759		817	
		760		818	
		761		819	
		762		820	
		763		821	
		764		822	
		765		823	
		766		824	
		767		825	
		768		826	
		769		827	
		770		828	
		771		829	
		772		830	
		773		831	
		774		832	
		775		833	
		776		834	
		777		835	
		778		836	
		779		837	
		780		838	
		781		839	
		782		840	
		783		841	
		784		842	
		785		843	
		786		844	
		787		845	
		788		846	
		789		847	
		790		848	
		791		849	
		792		850	
		793		851	
		794		852	
		795		853	
		796		854	
		797		855	
		798		856	
		799		857	
		800		858	
		801		859	
		802		860	
		803		861	
		804		862	
		805		863	
		806		864	
		807		865	
		808		866	
		809		867	
		810		868	
		811		869	
		812		870	
		813		871	
		814		872	
		815		873	
		816		874	
		817		875	
		818		876	
		819		877	
		820		878	
		821		879	
		822		880	
		823		881	
		824		882	
		825		883	
		826		884	
		827		885	
		828		886	
		829		887	
		830		888	
		831		889	
		832		890	
		833		891	
		834		892	
		835		893	
		836		894	
		837		895	
		838		896	
		839		897	
		840		898	
		841		899	
		842		900	
		843		901	
		844		902	
		845		903	
		846		904	
		847		905	
		848		906	
		849		907	
		850		908	
		851		909	
		852		910	
		853		911	
		854		912	
		855		913	
		856		914	
		857		915	
		858		916	
		859		917	
		860		918	
		861		919	
		862		920	
		863		921	
		864		922	
		865		923	
		866		924	
		867		925	
		868		926	
		869		927	
		870		928	
		871		929	
		872		930	
		873		931	
		874		932	
		875		933	
		876		934	
		877		935	
		878		936	
		879		937	
		880		938	
		881		939	
		882		940	
		883		941	
		884		942	
		885		943	
		886		944	
		887		945	
		888		946	
		889		947	
		890		948	
		891		949	
		892		950	
		893		951	
		894		952	
		895		953	
		896		954	
		897		955	
		898		956	
		899		957	
		900		958	
		901		959	
		902		960	
		903		961	
		904		962	
		905		963	
		906		964	
		907		965	
		908		966	
		909		967	
		910		968	
		911		969	
		912		970	
		913		971	
		914		972	
		915		973	
		916		974	
		917		975	
		918		976	
		919		977	
		920		978	
		921		979	
		922		980	
		923		981	
		924		982	
		925		983	
		926		984	
		927		985	
		928		986	
		929		987	
		930		988	
		931		989	
		932		990	
		933		991	
		934		992	
		935		993	
		936		994	
		937		995	
		938		996	
		939		997	
		940		998	
		941		999	
		942		1000	

U880D - LERNSYSTEM (V.2)		LS880		SEITE 16	
ADR	OBJ-KODE M ZEILE QUELL-KODE	ADR	OBJ-KODE M ZEILE QUELL-KODE	ASM U880	ASM U880
052D	E6F0	813	AND OFOH	871	AND OFOH
052E	CDN001	R 814	CALL SRL4	R 872	CALL SRL4
0532	77	815	LD M,A	873	LD M,A ;LSB 4 BIT HIGH NACH TR
0533	23	816	INC HL	874	INC HL
0534	B	817	LD A,E	875	LD A,E
0535	E60F	818	AND OFH	876	AND OFH
0537	77	819	LD M,A	877	LD M,A ;LSB 4 BIT LOW NACH TR
0538	22F187	R 820	INC HL	878	INC HL
0539	C33300	R 821	LD (TRZ),HL	879	LD (TRZ),HL
053C		R 822	JMP TAST	880	RET
053F	21F787	R 823	FILL3: LD HL,EXZ	881	INCEXZ: ;ZAEHLUNG DER EX-TASTEN
0542	3600	R 825	LD H,0 ;LOSCHEN EX-ZAEHLER	882	LD HL,EXZ
		826		883	INC M
		827	FILLH: ;FUELLEN MEMORY MIT KOHST	884	LD A,M
0544	DD21E687	R 828	LD IX,TR+1	885	LD A,M
0548	CD8102	R 829	CALL DAT10	886	RET
054B	45	830	LD B,L ;TO HEX-TASTEN HOLEN	887	
054C	6C	831	LD L,H ;IN B KONSTANTE	888	INEND: ;ENDE INPUT-MODUS SETZEN
054D	63	832	LD H,E ;IN HL ENDADRESSE	889	SUB A
054E	5A	833	LD E,D ;IN DE STARTADRESSE	R 890	LD (HARKIN),A
054F	51	834	LD D,C	R 891	JMP ATAST
0550	B7	835	OR A	892	
0551	ED52	836	SBC HL,DE ;STARTADR. > ENDADR. ?	893	
0553	DAB300	R 837	JPC STRNMT ;WENN JA FEHLER	894	RST18:
0556	EB	838	EX HL,DE ;WENN JA FEHLER	895	RST18H
0557	13	839	INC DE ;IN DE ADDR.-DIFFERENZ	896	
		840	FILLM:	897	
0558	97	841	SUB A	898	MONI: ;RUECKKEHR AUS SOFTW.-BRK.
0559	82	842	OR D	R 899	DI (SL),HL ;ALLE REGISTER RETTEN
055A	53	843	OR E ;ENDADR. ERREICHT ?	R 900	LD (SL),HL ;ALLE REGISTER RETTEN
055B	2805	844	JRZ FILLM2 ;WENN JA ABRUCH	R 901	PUSH AF
055D	70	845	LD M,B	R 902	POP HL
055E	1B	846	DEC DE	R 903	LD (SF),HL ;AF RETTEN
055F	23	847	INC HL	R 904	POP HL ;AF RETTEN
0560	18F6	848	JR FILLM1	R 905	DEC HL ;RUECKSTELLEN WEGEN RST 18
		849	FILLM2:	R 906	LD (SFC),HL ;USER-PC RETTEN
		850	DEC HL	R 907	LD (SSP),SP ;USER-SP RETTEN
0562	2B	851	LD A,M ;ANZEIGE INHALT LETZTER	R 908	HONIN:
0563	7E	852	LD A,M ;SPEICHERPLATZ	R 909	LD SP,SFC+8
0564	47	853	LD B,A	R 910	PUSH IX
0565	E60F	854	AND OFH	R 911	PUSH IY
0567	DC802	R 855	CALL SECURM	R 912	LD SP,SFC+22
056A	32F87	R 856	LD (LEDPUF),A	R 913	PUSH BC
056D	78	857	LD A,B	R 914	PUSH DE
056E	C33401	R 858	JMP DM2	R 915	EXAF
		859	TRSET:	R 916	EXX
		860	LD A,D ;UP: TASTENREG UMSETZEN	R 917	PUSH HL
0571	7A	861	LD A,D	R 918	PUSH BC
0572	E6F0	862	AND OFOH	R 919	PUSH DE
0574	CDN001	R 863	CALL SRL4	R 920	LD SP,(SSP)
0577	77	864	LD M,A ;MSB 4 BIT HIGH NACH TR	R 921	PUSH AF
0578	23	865	INC HL	R 922	POP HL
0579	7A	866	LD A,D	R 923	LD (SFO),HL
057A	E60F	867	AND OFH	R 924	LD A,I
057C	77	868	LD M,A ;MSB 4 BIT LOW NACH TR	R 925	LD (SL),A
057D	23	869	INC HL	R 926	LD HL,(SFC)
057E	7B	870	LD A,E	R 927	EI
				R 928	JMP LEDADR ;ANZEIGE PC

U880D - LERNSYSTEM (V.2)		LS880		SEITE 15	
ADR	OBJ-KODE M ZEILE QUELL-KODE	ADR	OBJ-KODE M ZEILE QUELL-KODE	ASM U880	ASM U880
052D	E6F0	813	AND OFOH	871	AND OFOH
052E	CDN001	R 814	CALL SRL4	R 872	CALL SRL4
0532	77	815	LD M,A	873	LD M,A ;1. HALBBYTE NACH TR
0533	23	816	INC HL	874	INC HL
0534	B	817	LD A,E	875	LD A,E
0535	E60F	818	AND OFH	876	AND OFH
0537	77	819	LD M,A	877	LD M,A ;2. HALBBYTE NACH TR
0538	22F187	R 820	INC HL	878	INC HL
0539	C33300	R 821	LD (TRZ),HL	879	LD (TRZ),HL
053C		R 822	JMP TAST	880	RET
053F	21F787	R 823	FILL3: LD HL,EXZ	881	INCEXZ: ;ZAEHLUNG DER EX-TASTEN
0542	3600	R 825	LD H,0 ;LOSCHEN EX-ZAEHLER	882	LD HL,EXZ
		826		883	INC M
		827	FILLH: ;FUELLEN MEMORY MIT KOHST	884	LD A,M
0544	DD21E687	R 828	LD IX,TR+1	885	LD A,M
0548	CD8102	R 829	CALL DAT10	886	RET
054B	45	830	LD B,L ;TO HEX-TASTEN HOLEN	887	
054C	6C	831	LD L,H ;IN B KONSTANTE	888	INEND: ;ENDE INPUT-MODUS SETZEN
054D	63	832	LD H,E ;IN HL ENDADRESSE	889	SUB A
054E	5A	833	LD E,D ;IN DE STARTADRESSE	R 890	LD (HARKIN),A
054F	51	834	LD D,C	R 891	JMP ATAST
0550	B7	835	OR A	892	
0551	ED52	836	SBC HL,DE ;STARTADR. > ENDADR. ?	893	
0553	DAB300	R 837	JPC STRNMT ;WENN JA FEHLER	894	RST18:
0556	EB	838	EX HL,DE ;WENN JA FEHLER	895	RST18H
0557	13	839	INC DE ;IN DE ADDR.-DIFFERENZ	896	
		840	FILLM:	897	
0558	97	841	SUB A	898	MONI: ;RUECKKEHR AUS SOFTW.-BRK.
0559	82	842	OR D	R 899	DI (SL),HL ;ALLE REGISTER RETTEN
055A	53	843	OR E ;ENDADR. ERREICHT ?	R 900	LD (SL),HL ;ALLE REGISTER RETTEN
055B	2805	844	JRZ FILLM2 ;WENN JA ABRUCH	R 901	PUSH AF
055D	70	845	LD M,B	R 902	POP HL
055E	1B	846	DEC DE	R 903	LD (SF),HL ;AF RETTEN
055F	23	847	INC HL	R 904	POP HL ;AF RETTEN
0560	18F6	848	JR FILLM1	R 905	DEC HL ;RUECKSTELLEN WEGEN RST 18
		849	FILLM2:	R 906	LD (SFC),HL ;USER-PC RETTEN
		850	DEC HL	R 907	LD (SSP),SP ;USER-SP RETTEN
0562	2B	851	LD A,M ;ANZEIGE INHALT LETZTER	R 908	HONIN:
0563	7E	852	LD A,M ;SPEICHERPLATZ	R 909	LD SP,SFC+8
0564	47	853	LD B,A	R 910	PUSH IX
0565	E60F	854	AND OFH	R 911	PUSH IY
0567	DC802	R 855	CALL SECURM	R 912	LD SP,SFC+22
056A	32F87	R 856	LD (LEDPUF),A	R 913	PUSH BC
056D	78	857	LD A,B	R 914	PUSH DE
056E	C33401	R 858	JMP DM2	R 915	EXAF
		859	TRSET:	R 916	EXX
		860	LD A,D ;UP: TASTENREG UMSETZEN	R 917	PUSH HL
0571	7A	861	LD A,D	R 918	PUSH BC
0572	E6F0	862	AND OFOH	R 919	PUSH DE
0574	CDN001	R 863	CALL SRL4	R 920	LD SP,(SSP)
0577	77	864	LD M,A ;MSB 4 BIT HIGH NACH TR	R 921	PUSH AF
0578	23	865	INC HL	R 922	POP HL
0579	7A	866	LD A,D	R 923	LD (SFO),HL
057A	E60F	867	AND OFH	R 924	LD A,I
057C	77	868	LD M,A ;MSB 4 BIT LOW NACH TR	R 925	LD (SL),A
057D	23	869	INC HL	R 926	LD HL,(SFC)
057E	7B	870	LD A,E	R 927	EI
				R 928	JMP LEDADR ;ANZEIGE PC

```

929 ; TABELLE HEX AUF 7-SGHT.
930 SEGH DEF8 03FH ;0
931 DEF8 006H ;1
932 DEF8 05BH ;2
933 DEF8 0AFH ;3
934 DEF8 066H ;4
935 DEF8 06DH ;5
936 DEF8 07DH ;6
937 DEF8 027H ;7
938 DEF8 07FH ;8
939 DEF8 06FH ;9
940 DEF8 077H ;A
941 DEF8 07CH ;B
942 DEF8 039H ;C
943 DEF8 05EH ;D
944 DEF8 079H ;E
945 DEF8 073H ;F
946 DEF8 071H ;F
947 ;
948 UMCOD DEF8 23D ;UMKOD-TABELLE
949 DEF8 22D
950 DEF8 11D
951 DEF8 21D
952 DEF8 20D
953 DEF8 19D
954 DEF8 18D
955 DEF8 10D
956 DEF8 17D
957 DEF8 16D
958 DEF8 9D
959 DEF8 15D
960 DEF8 14D
961 DEF8 13D
962 DEF8 12D
963 DEF8 8D
964 ;
965 DISP: ;DISPLAY-ROUTINE
966 PUSH AF
967 PUSH BC
968 PUSH HL
969 ;
970 CALL BLANK ;C:=DISPAD
971 LD HL,DISSTA ;STATUS HOLEN
972 LD A,N
973 DEC H
974 AND 7
975 JNZ DISP1
976 DEC B
977 DISPO: ;R:=FEH
978 LD HL,FEHLER-1
979 RLC B
980 INC HL
981 DEC A
982 JNZ DISPO
983 LD A,M
984 OUT A
985 DISPI:
986 POP HL
987 POP BC
988 ;
989 ;
990 ;
991 ;
992 ;
993 ;
994 ;
995 ;
996 ;
997 ;
998 ;
999 ;
1000 ;
1001 ;
1002 ;
1003 DISON: ;EINSCHALTEN DISP-INTERRUPT
1004 LD A,91H
1005 OUT CTC1
1006 RET
1007 ;
1008 ORG 062AH
1009 ;INTERRUPTLISTE:
1010 DEF8 0
1011 INTAB DEF8 DISP
1012 ;
1013 USRTO EQU 7DH ;USART-STEUERWORT
1014 USRDA EQU 7CH ;USART-DATEN-KANAL
1015 CTC0 EQU 0BCH ;STEUERADRESSE
1016 NIPORT EQU 0D0H ;INH-ADRESSE
1017 CTC1 EQU 0BDH ;STEUERADRESSE
1018 DISP EQU 0DAH ;ALLGEMEINE
1019 ;STEUERADRESSE FUEER HILFSZELLEN
1020 ;
1021 ;
1022 ;
1023 ORG 87C3H;
1024 SPC DEFS 2
1025 SSP DEFS 2
1026 SIY DEFS 2
1027 SIX DEFS 2
1028 SFO DEFS 1
1029 SAO DEFS 1
1030 SF DEFS 1
1031 SA DEFS 1
1032 SEO DEFS 1
1033 SDO DEFS 1
1034 SGO DEFS 1
1035 SBO DEFS 1
1036 SLO DEFS 1
1037 SHO DEFS 1
1038 SE DEFS 1
1039 SD DEFS 1
1040 SC DEFS 1
1041 SB DEFS 1
1042 SL DEFS 1
1043 SH DEFS 1
1044 SI DEFS 1

```

```

929 ;
930 ;
931 ;
932 ;
933 ;
934 ;
935 ;
936 ;
937 ;
938 ;
939 ;
940 ;
941 ;
942 ;
943 ;
944 ;
945 ;
946 ;
947 ;
948 ;
949 ;
950 ;
951 ;
952 ;
953 ;
954 ;
955 ;
956 ;
957 ;
958 ;
959 ;
960 ;
961 ;
962 ;
963 ;
964 ;
965 ;
966 ;
967 ;
968 ;
969 ;
970 ;
971 ;
972 ;
973 ;
974 ;
975 ;
976 ;
977 ;
978 ;
979 ;
980 ;
981 ;
982 ;
983 ;
984 ;
985 ;
986 ;
987 ;
988 ;
989 ;
990 ;
991 ;
992 ;
993 ;
994 ;
995 ;
996 ;
997 ;
998 ;
999 ;
1000 ;
1001 ;
1002 ;
1003 ;
1004 ;
1005 ;
1006 ;
1007 ;
1008 ;
1009 ;
1010 ;
1011 ;
1012 ;
1013 ;
1014 ;
1015 ;
1016 ;
1017 ;
1018 ;
1019 ;
1020 ;
1021 ;
1022 ;
1023 ;
1024 ;
1025 ;
1026 ;
1027 ;
1028 ;
1029 ;
1030 ;
1031 ;
1032 ;
1033 ;
1034 ;
1035 ;
1036 ;
1037 ;
1038 ;
1039 ;
1040 ;
1041 ;
1042 ;
1043 ;
1044 ;

```

U880D - LERNSYSTEM (V. 2) LS880  
 ADR OBJ-KODE N ZELLE QUELL-KODE  
 1045 ;  
 1046 ;BRBUF DEFS 1 ;BEIHEBER BRV.-KODE  
 1047 ;T1X DEFS 4 ;SPEICHER FUER TX-NEU  
 1048 ;T1A DEFS 4 ;SPEICHER FUER TX-ALT  
 1049 ;T1 DEFS 12 ;TASIERZEIGER TX  
 1050 ;TR2 DEFS 2 ;TASIERZEIGER TX  
 1051 ;MARKS DEFS 1 ;MARKE START-MODUS  
 1052 ;MARKIN DEFS 1 ;MARKE INPUT-MODUS  
 1053 ;BRADR DEFS 2 ;SPEICHER FUER  
 1054 ; ;BREM-ADRESSE  
 1055 ;EXZ DEFS 1 ;EXZ-ZAEBLER  
 1056 ; ;  
 1057 ;ORG 87F0H ;  
 1058 ;DISSTA DEFS 1 ;DISPLAY-STATUS  
 1059 ;FEHLER DEFS 1 ;SPEICHER FUER  
 1060 ; ;FEBREIT (LIT)  
 1061 ;LEDRUF DEFS 6 ;LED-ANZEIGESPEICHER  
 1062 ; ;  
 1063 ; ;

ADR	OBJ-KODE	N	ZELLE	QUELL-KODE	REFERENZ-TAFEL	WEIT	M	DEF.	BEZ.	LS880	SEITE	20
87DC					ABRF1	0432	R	661	665	676		
87DD					ABRF3	0480	R	744	747	759		
87E1					ARET	034A	R	536	533			
87E5					ATAT	003D	R	44	412	416	442	592 606 891
87F1					AUS2	04C6	R	764	749			
87F3					AUXS	01CA	R	274	271			
87F4					BLANK	0617	R	991	969	1001		
87F5					BRADR	87F5	R	1053	500	506	509	578 585 595 601 604
87F7					BRBUF	87DC	R	1046	505	588	600	
87F8					BRCH	0444	R	677	669			
87F8					BREAK	036C	R	571	23			
87F8					BREKL	0393	R	561	577			
87F9					BREI	0346	R	532	530			
87F9					C03	01FA	R	309	258			
87FA					C47	0209	R	317	281			
87FA					CG	0369	R	554	551			
87FA					C0TA	00E9	R	153	130			
87FA					C0TA1	00E2	R	160	157			
87FA					C1C0	00BC	R	1015	29	36	654	685 739 768
87FA					C1C1	00BD	R	1018	32	34	1000	1005
87FA					D24810	0234	R	349	397	402	407	
87FA					DA	0241	R	357	388			
87FA					DAT10	0281	R	404	829			
87FA					DAT2	0271	R	393	302	555	807	
87FA					DAT4	022E	R	344	331	584	616	628 722 734 785 796
87FA					DAT8	0279	R	399	641	645		
87FA					DDH	0149	R	214	187			
87FA					DIS2	01F6	R	305	298			
87FA					DIS4	0228	R	337	327	591	617	629 643 723 786 797
87FA					DISADR	0295	R	419	415			
87FA					DISDAT	02CF	R	457	809			
87FA					DISOFF	061F	R	998	411			
87FA					DISON	0625	R	1003				
87FA					DISP	0363	R	945	1011			
87FA					DISPO	0808	R	977	981			
87FA					DISP1	0611	R	984	974			
87FA					DISPAD	00DA	R	1020	994			
87FA					DISSET	0197	R	251	225	227		
87FA					DISSTA	87F8	R	1058	970			
87FA					DM1	0127	R	194	217			
87FA					DM2	0134	R	201	858			
87FA					ERROR	04C9	R	760	751	754		
87FA					EX	014F	R	219	177			
87FA					EX2	87F7	R	1055	440	635	729	824 883
87FA					FEHLER	87F9	R	1059	114	446	976	
87FA					FFCLR	02C2	R	445	439			
87FA					FFDAT	026D	R	389	355			
87FA					FFSET	0080	R	113	103	107	762	
87FA					FFU	009E	R	103	93			
87FA					FILL	04E2	R	778	237			
87FA					FILL1	04FE	R	792	782			
87FA					FILL2	0517	R	803	793			
87FA					FILL3	053F	R	821	804			
87FA					FILL4	0544	R	827	848			
87FA					FILLM1	0538	R	840	848			
87FA					FILLM2	0562	R	849	844			
87FA					FORT1	001B	R	21	16			

REFERENZ-TAFEL  
SYMBOL WERT M DEF. BEZ.

SH0	87D4 R	1037	
SI	87D2 R	1044	
SIX	87C9 R	1027	313 925
SIX	87C7 R	1056	
SLO	87D9 R	1042	72 900
SLO	87D3 R	1056	
SFC	87C3 R	1024	7 76
			192 195 215 280 292 318
			480 463 486 499 513 557 568 906
SRL4	0140 R	207	
SRP	87C5 R	1025	77 479 482 490 512 517 522 907
STAPE	0435 R	638	
START	0206 R	464	161
STEP	02DF R	472	183
STORE	C3CB R	603	233
STORE1	03C7 R	624	613
STORE2	04C0 R	635	625
STORM	02E7 R	438	110 179
STRAW	00A3 R	106	111 174
			239 241 242 245 247 248
			265 312 391 582 598 650 637
TIA	87E1 R	1042	56 120
TIN	87DD R	1047	86 121
TAI	03EE R	55	62
TAFER	0049 R	51	42 04
TAL	0042 R	47	180
TOST	0082 R	119	106
TOT	0082 R	119	106
TR	87E5 R	1049	45
			171 223 253 296 325 542 549
			523 574 615 620 621 631 940 844
			721 725 733 784 788 795 799 806
			811 828
TRSET	0371 R	660	621 632 726 789 800
TRZ	87F1 R	1050	46 167 170 321 379
UNICOD	05E5 R	943	285
USRTCO	097D	1013	24
USRIDA	097C	1014	673
ZEIK1	0465 R	700	711
ZEIK2	0460 R	703	709
ZEIK3	0468 R	706	707
ZEIKO	0462 R	697	600
ZEIS1	003F R	143	149
ZEIS2	003E R	146	147

REFERENZ-TAFEL  
SYMBOL WERT M DEF. BEZ.

FORT2	003A R	41	37
FORT3	007C R	82	66
GG	01FD R	300	
HRET	0325 R	511	503
IDM	0123 R	191	185
IRCEXZ	058F R	682	611
INEND	0595 R	688	544
ININ	0346 R	552	546
INPUT	034C R	541	222
INTAB	062C R	1011	25
IYJNP1	024E R	367	347
IYJNP2	0252 R	370	401
IYJNP3	025A R	375	406
IYJNPE	0262 R	380	369
LADD	0088 R	88	92
LCP	0097 R	98	102
LEDADR	028F R	414	205
LEDATA	0289 R	410	307
LEDPUF	87FA R	1061	660
			667 690 741 743 770 772 856
LL	0008 R	6	10
LLD	0012 R	13	22
LOAD	0475 R	716	235
LOAD1	0491 R	729	720
LSEP	00C2 R	124	136
LTAPE	0456 R	732	435
LZLF	029C R	423	548
HARKIN	87F4 R	1052	220
HARKS	87F3 R	1051	466
HENSET	01DA R	279	256
MONI	059D R	598	19
MONIH	05AF R	508	78
MSEK	00DC R	140	53
RIPORT	00D0	1017	535
ROCHI	024A R	363	381
REC8	01B2 R	262	
RESET	0000 R	4	40
RETURN	02E3 R	477	468
PRA	00EB R	155	159
RST18	059C R	894	589
SA	87CE R	1031	
SAD	87CC R	1029	
SAVE	00F5 R	165	49
SB	87DB R	1041	
SBO	87D2 R	1035	
SC	87D7 R	1040	
SCO	87D1 R	1034	
SD	87D6 R	1039	
SDO	87D0 R	1033	
SE	87D5 R	1038	
SEO	87CF R	1032	
SEGH	05D5 R	931	451
SECUMW	02C8 R	450	198
SET2	01DB R	284	272
SET3	01E9 R	295	282
SET3	01E9 R	295	282 315
SF	87CD R	1030	74 497 903
SFO	87CB R	1028	493 923
SH	87DA R	1043	

### A.4. Unterlagen zur Prozessorbaugruppe des Lernsystems

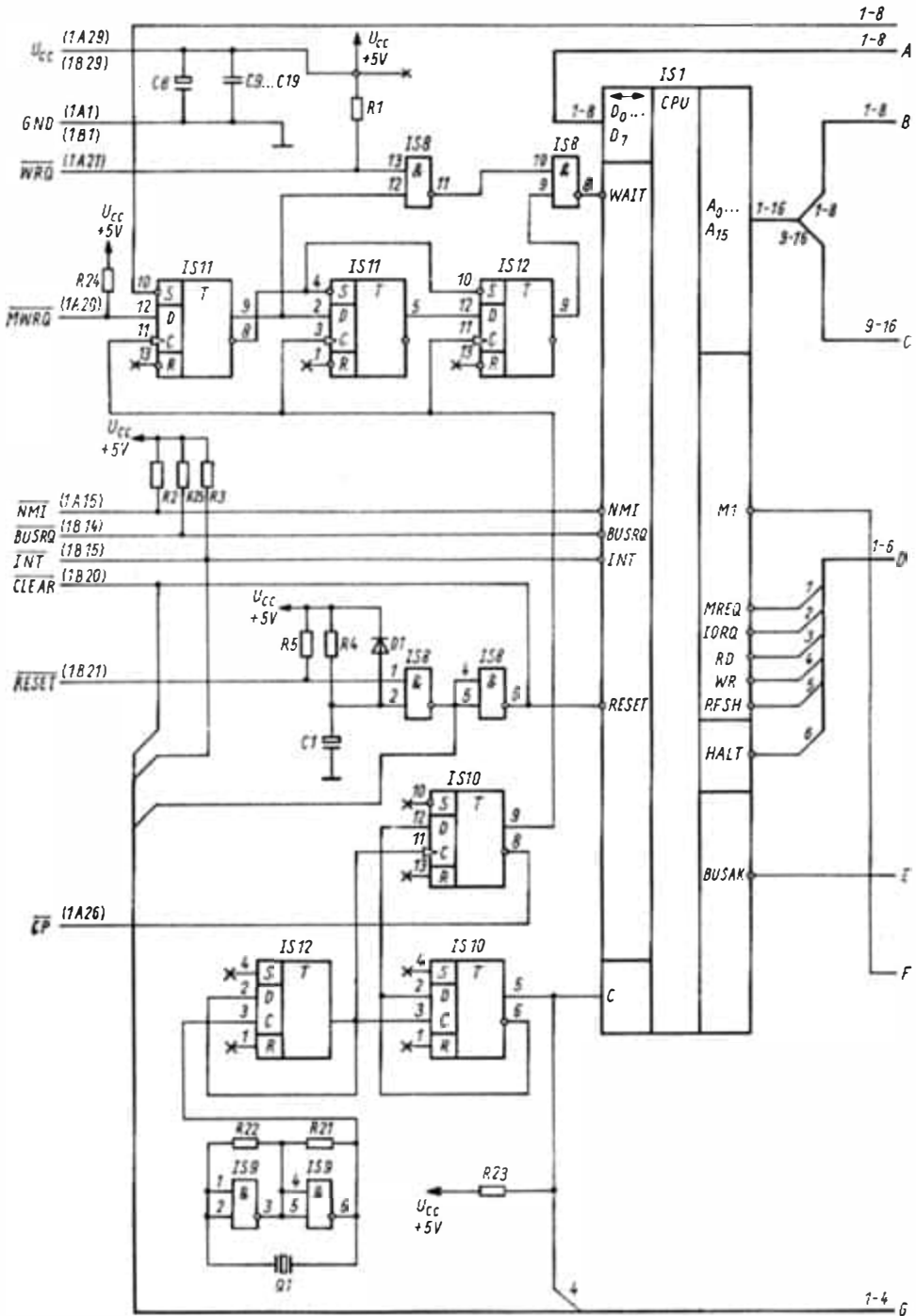
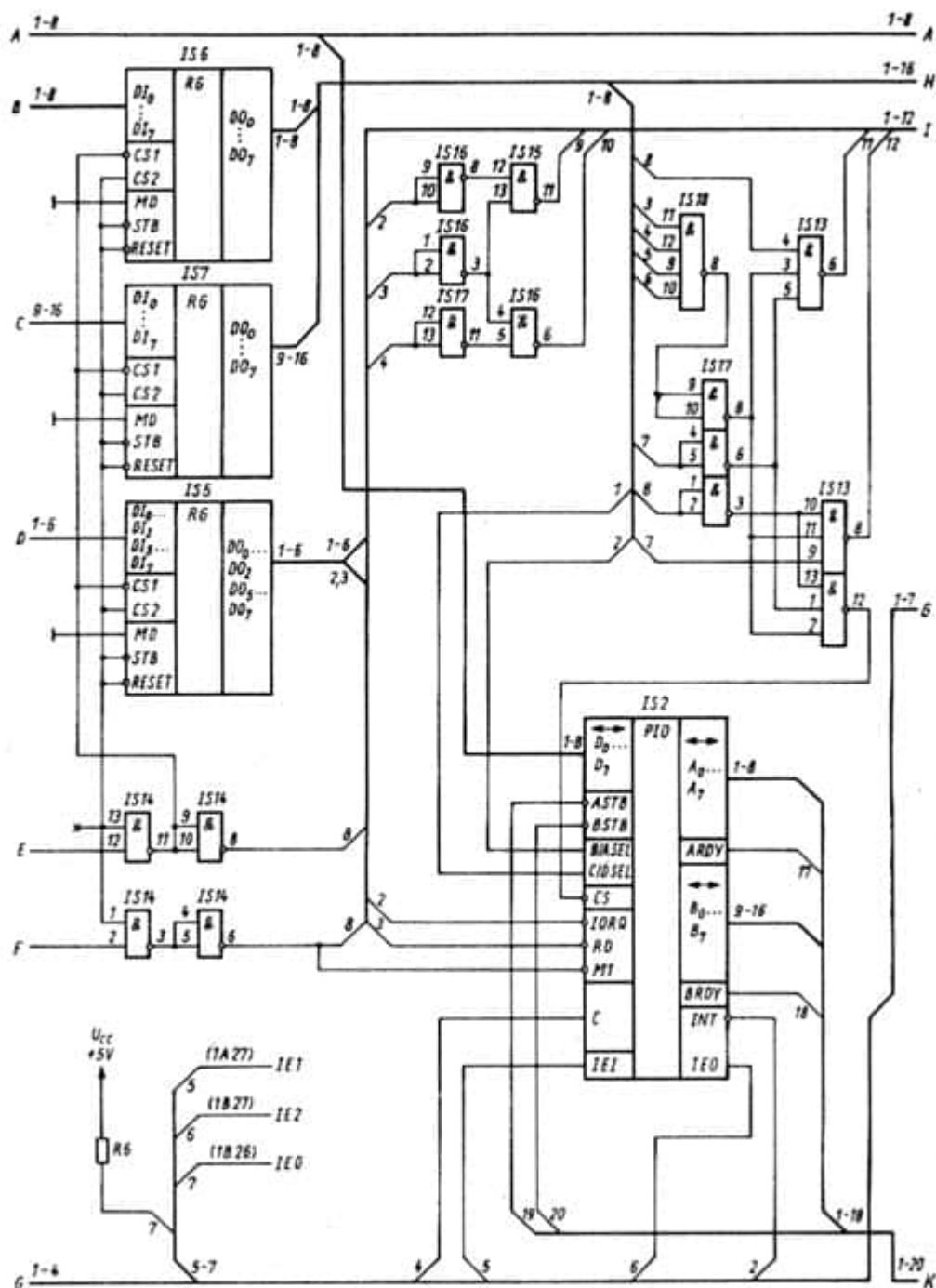
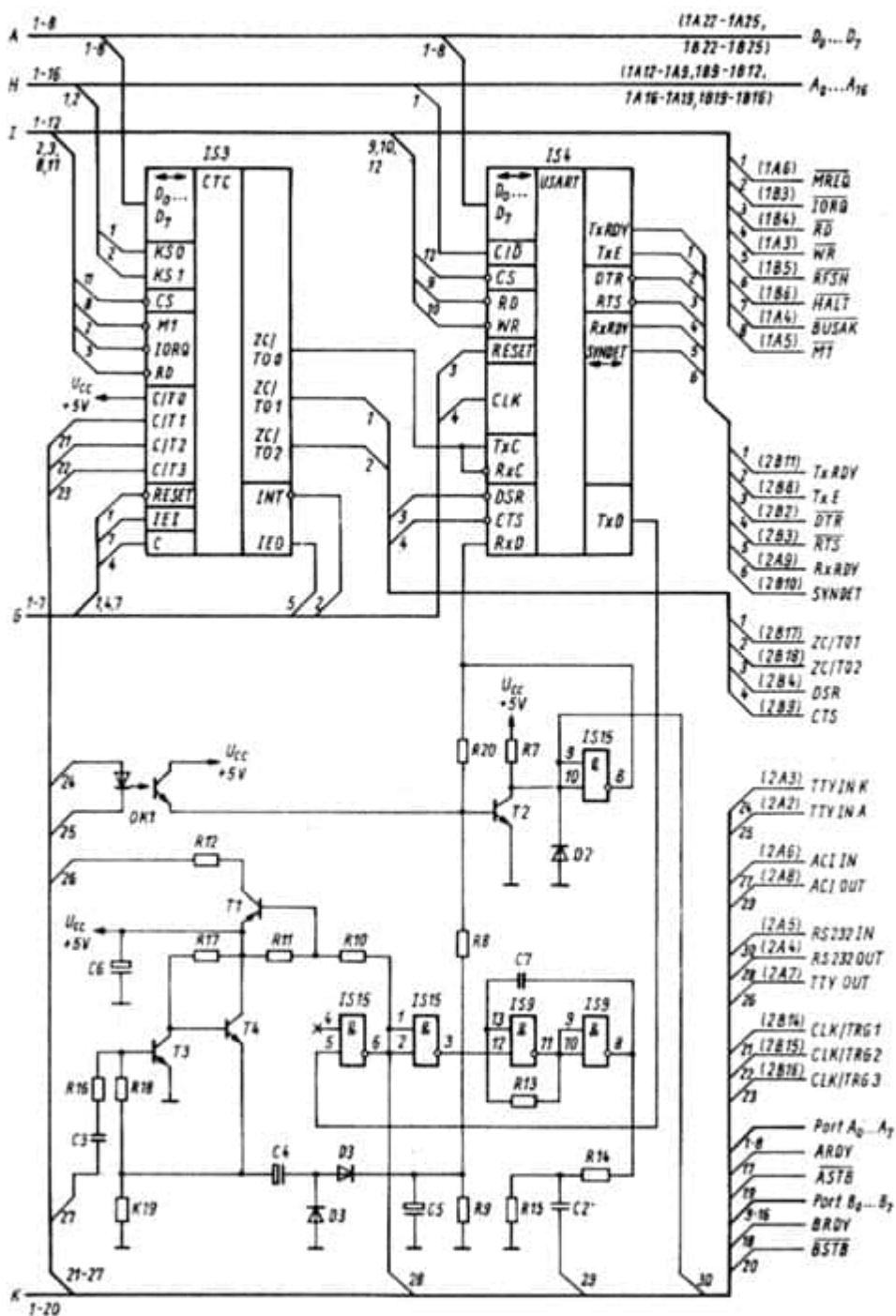


Bild A.4.1. Stromlaufplan FPS2.CPU1





Tafel A.4.1. Stückliste FPS2.CPU1

Lfd. Nr.	Stückzahl	Benennung	Sachnummer	Bemerkungen
1	1	CPU U880		IS1
2	1	PIO U855		IS2
3	1	CTC U857		IS3
4	1	USART 8251		IS4
5	3	Schaltkreis MH 3212	Tesla	IS5 ... IS7
6	5	Schaltkreis D100	TGL 26152	IS8, IS9 IS14 ... IS17
7	2	Schaltkreis D274	TGL 28816	IS10, IS11
8	1	Schaltkreis D174	TGL 29266	IS12
9	1	Schaltkreis D110	TGL 26152	IS13
10	1	Schaltkreis D140	TGL 26152	IS18
11	1	Transistor KT326B		T1
12	3	Transistor SC237D	TGL 29953	T2 ... T4
13	2	Diode SAY40	TGL 200-8422	D1, D2
14	1	Diode SAL41	TGL 27975	D3
15	1	Optokoppler MB101	WFB	OK1
16	2	Tantalkondensator 22 $\mu$ F/6 V	TGL 200-8519	C1, C8
17	1	Kondensator MK T1 0,22 $\mu$ F/+20%/100 V	TGL 31 680/01	C2
18	1	Polyesterkondensator 6,8 nF/160 V	TGL 200-8424	C3
19	2	Tantalkondensator 1 $\mu$ F/10 V	TGL 200-8519	C4, C5
20	1	Tantalkondensator 4,7 $\mu$ F/6 V	TGL 200-8519	C6
21	1	Kondensator MKL3 0,47 $\mu$ F/+20%/63 V	TGL 10793/03	C7
22	11	Kondensator SDVU 47 nF 50/63	TGL 7619.84	C9 ... C19
23	8	Schichtwiderstand 250.207 TK 4,7 k $\Omega$ 5%	TGL 8728	R1 ... R4, R6, R7, R24, R25 R5, R8, R11
24	3	Schichtwiderstand 250.207 TK 10 k $\Omega$ 5%	TGL 8728	R9
25	1	Schichtwiderstand 250.207 TK 2,2 k $\Omega$ 5%	TGL 8728	R10
26	1	Schichtwiderstand 250.207 TK 1 k $\Omega$ 5%	TGL 8728	R12
27	1	Schichtwiderstand 250.207 TK 150 $\Omega$ 5%	TGL 8728	R13
28	1	Schichtwiderstand 250.207 TK 160 $\Omega$ 5%	TGL 8728	R14, R20
29	2	Schichtwiderstand 250.207 TK 220 k $\Omega$ 5%	TGL 8728	R15
30	1	Schichtwiderstand 250.207 TK 68 k $\Omega$ 5%	TGL 8728	R16, R19
31	2	Schichtwiderstand 250.207 TK 470 $\Omega$ 5%	TGL 8728	R17
32	1	Schichtwiderstand 250.207 TK 18 k $\Omega$ 5%	TGL 8728	R18
33	1	Schichtwiderstand 250.207 TK 1 M $\Omega$ 5%	TGL 8728	R21, R22
34	2	Schichtwiderstand 250.207 TK 820 $\Omega$ 5%	TGL 8728	R23
35	1	Schichtwiderstand 250.207 TK 330 $\Omega$	TGL 8728	Q1
36	1	Quarz Q51/0/31/9999,5 kHz	TGL 33 584	

Tafel A.4.2. Rückseitensteckverbinderbelegung FPS2.CPU1

Pin	Lötseite (A)	Bestückungsseite (B)	Pin	Lötseite (A)	Bestückungsseite (B)
1	$U_{CC}$ (+5 V)	$U_{CC}$ (+5 V)	16		CLK/TRG3
2	TTY IN A	$\overline{DTR}$	17		ZC/TO1
3	TTY IN K	$\overline{RTS}$	18		ZC/TO2
4	RS232 OUT	$\overline{DSR}$	19	A7	B7
5	RS232 IN		20	A6	B6
6	ACI IN		21	A5	B5
7	TTY OUT		22	A4	B4
8	ACI OUT	TxE	23	A3	B3
9	RxRDY	$\overline{CTS}$	24	A2	B2
10		SYNDET	25	A1	B1
11		TxRDY	26	A0	B0
12	GND	GND	27	$\overline{ASTB}$	$\overline{BSTB}$
13	GND	GND	28	$U_{CC}$ (+5 V)	$U_{CC}$ (+5 V)
14		CLK/TRG1	29	ARDY	BRDY
15		CLK/TRG2			

## A.5. Unterlagen zur Speicherbaugruppe des Lernsystems

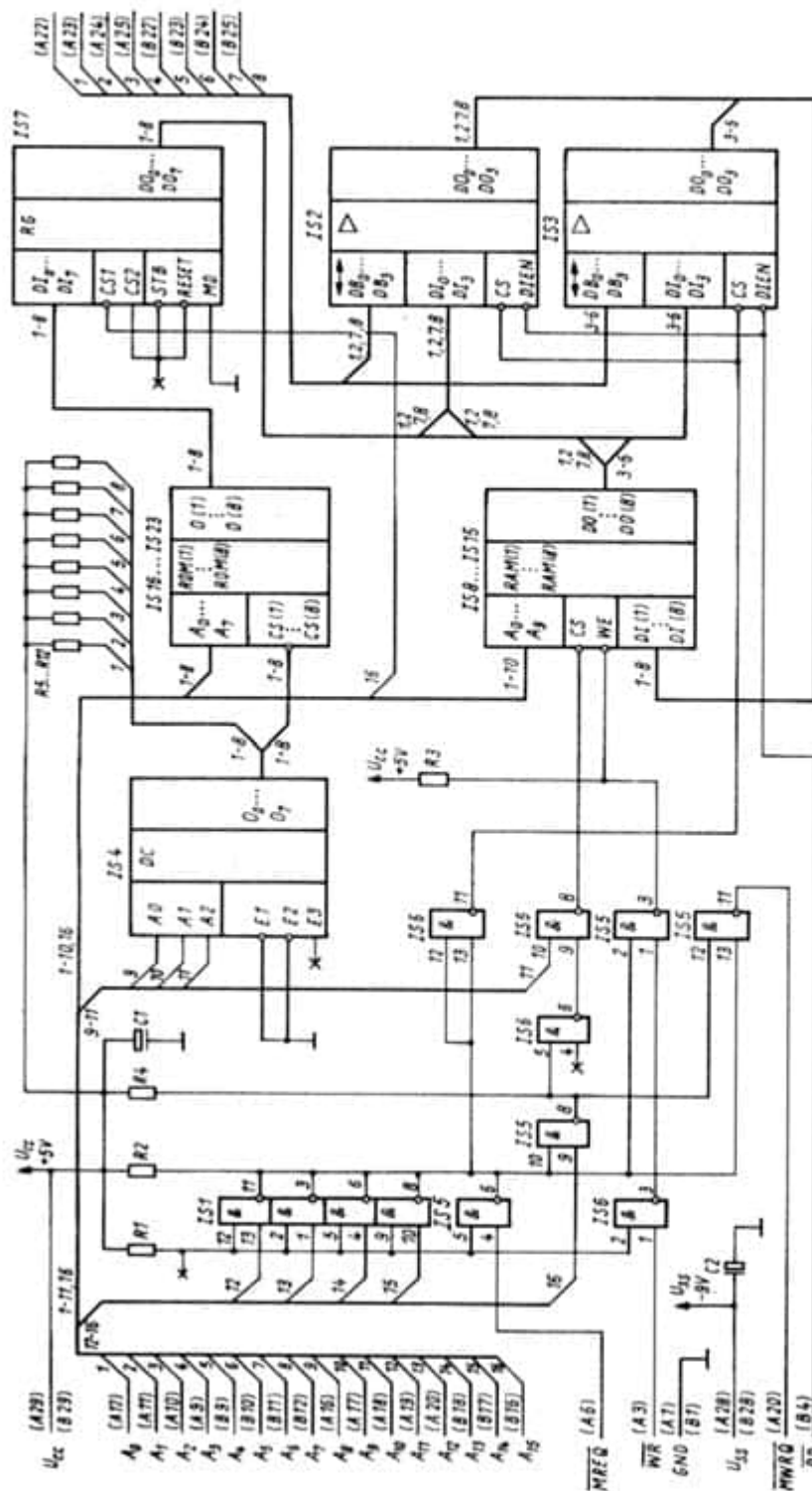


Bild A.5.1. Stromlaufplan FPS2.RR1

*Tafel A.5.1. Stückliste FPS2.RR1*

Lfd. Nr.	Stückzahl	Benennung	Sachnummer	Bemerkungen
1	1	Schaltkreis D103	TGL 27148	IS1
2	2	Schaltkreis MH3216	Tesla	IS2, IS3
3	1	Schaltkreis MH3205	Tesla	IS4
4	1	Schaltkreis D201	TGL 29262	IS5
5	1	Schaltkreis D100	TGL 26152	IS6
6	1	Schaltkreis MH3212	Tesla	IS7
7	8	RAM U202		IS8 ... IS15
8	8	PROM/EPROM/ROM U551/U552/U501		IS16 ... IS23
9	1	Tantalkondensator 22 $\mu$ F/6 V 20%	TGL 200-8519	C1
10	1	Tantalkondensator 25 $\mu$ F/10 V 20%	TGL 200-8519	C2
11	11	Schichtwiderstand 250.207 TK 1 k $\Omega$ 5%	TGL 8728	R1, R2, R4 ... R12
12	1	Schichtwiderstand 250.207 TK 300 $\Omega$ 5%	TGL 8728	R3

# A.6. Unterlagen zur Tastatur/Anzeige-Bedienbaugruppe

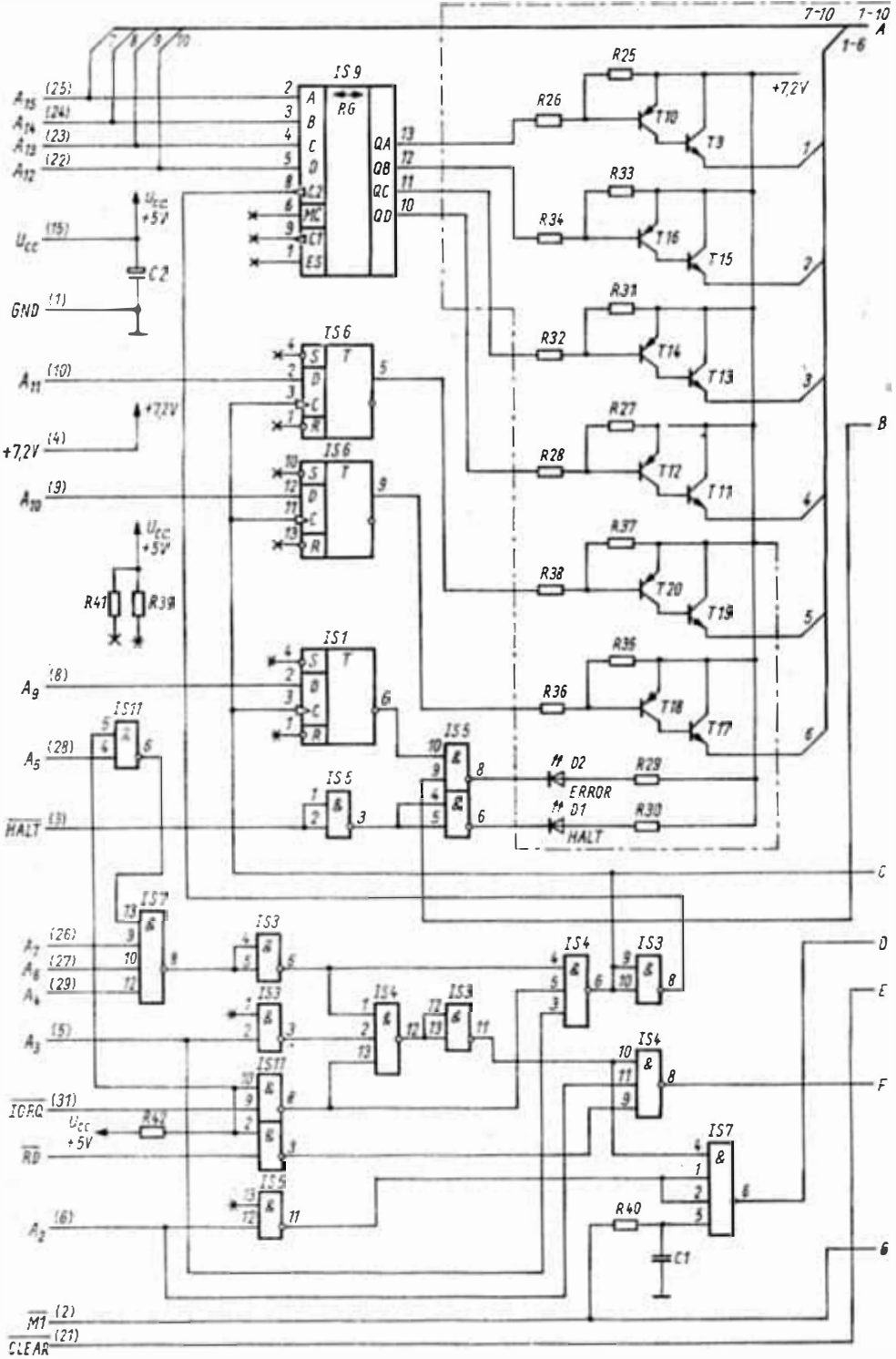
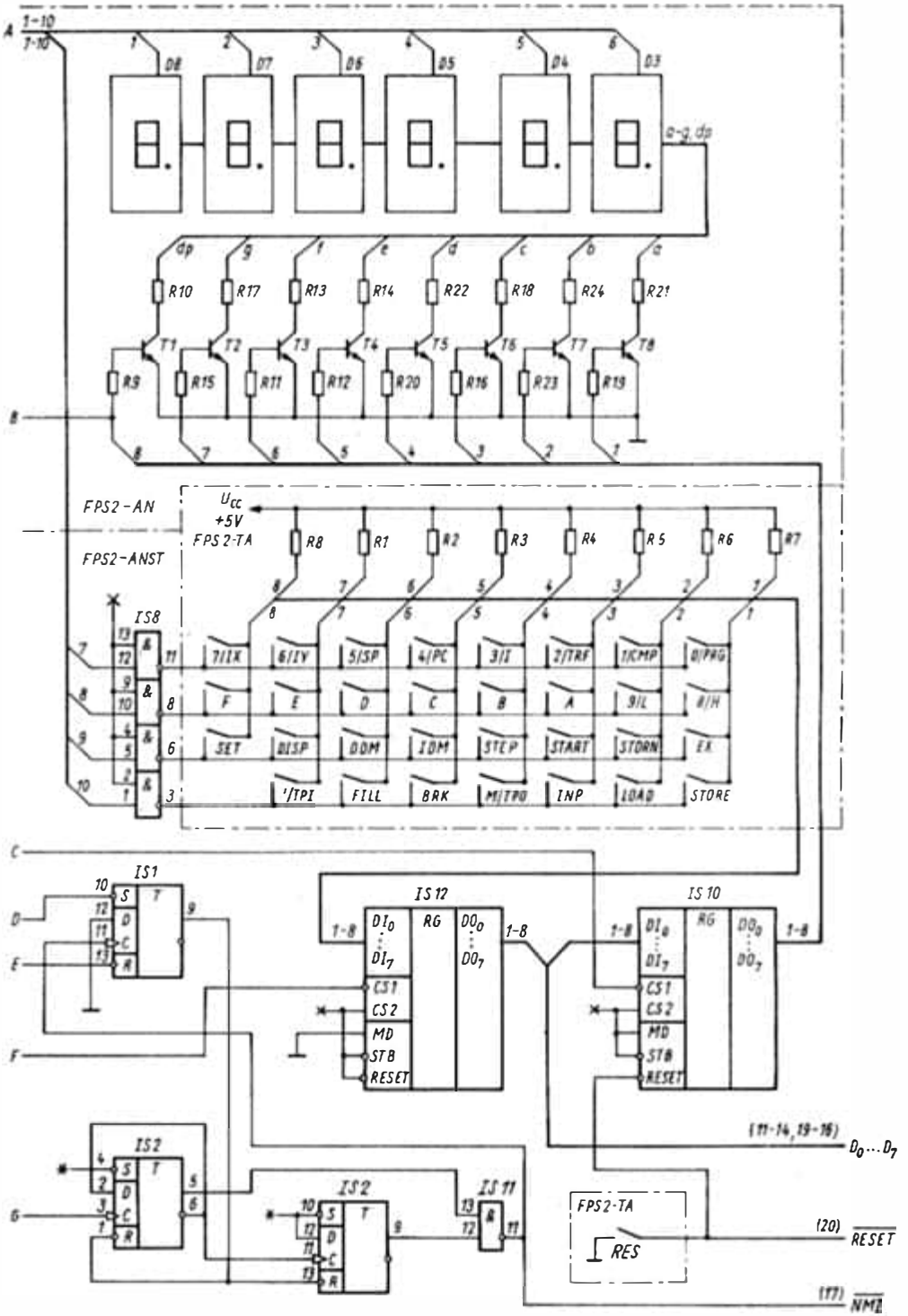


Bild A.6.1. Stromlaufplan FPS2.AN/TA/ANST



Tafel A.6.1. Stückliste FPS2.AN/TA

Lfd. Nr.	Stückzahl	Benennung	Sachnummer	Bemerkungen
1	8	Transistor SC237E	TGL 29953	T1 ... T8
2	6	Transistor SF126E	TGL 11811	T9, T11, T13, T15, T17, T19
3	6	Transistor KT326B		T10, T12, T14, T16, T18, T20
4	2	Lumineszenzdiode VQA12	TGL 31243	D1, D2
5	6	Lumineszenzanzeige VQB 71	TGL 31245	D3 ... D8
6	8	Schichtwiderstand 250.207 TK 6,8 k $\Omega$ 5%	TGL 8728	R1 ... R8
7	8	Schichtwiderstand 250.207 TK 1 k $\Omega$ 5%	TGL 8728	R9, R11, R12, R15, R16, R19, R20, R23
8	1	Schichtwiderstand 250.207 TK 75 $\Omega$ 5%	TGL 8728	R10
9	7	Schichtwiderstand 250.207 TK 43 $\Omega$ 5%	TGL 8728	R13, R14, R17, R18, R21, R22, R24
10	6	Schichtwiderstand 250.207 TK 1,2 k $\Omega$ 5%	TGL 8728	R25, R27, R31, R33, R35, R37
11	6	Schichtwiderstand 250.207 TK 3,3 k $\Omega$ 5%	TGL 8728	R26, R28, R32, R34, R36, R38
12	1	Schichtwiderstand 250.207 TK 390 $\Omega$ 5%	TGL 8728	R29
13	1	Schichtwiderstand 250.207 TK 150 $\Omega$ 5%	TGL 8728	R30

Tafel A.6.2. Stückliste FPS2.AN/ST

Lfd. Nr.	Stückzahl	Benennung	Sachnummer	Bemerkungen
1	3	Schaltkreis D174	TGL 29266	IS1, IS2, IS6
2	3	Schaltkreis D100	TGL 26152	IS3, IS5, IS11
3	1	Schaltkreis D110	TGL 26152	IS4
4	1	Schaltkreis D120	TGL 26152	IS7
5	1	Schaltkreis D103	TGL 27148	IS8
6	1	Schaltkreis D195	TGL 28467	IS9
7	2	Schaltkreis MH3212	Tesla	IS10, IS12
8	1	Kondensator EDVU 470 pF E2000 (V)	TGL 24100	C1
9	1	Tantalkondensator 22 $\mu$ F/6 V	TGL 200-8519	C2
10	3	Schichtwiderstand 250.207 TK 1 k $\Omega$ 5%	TGL 8728	R39, R41, R42
11	1	Schichtwiderstand 250.207 TK 360 $\Omega$ 5%	TGL 8728	R40

### A.7. Unterlagen zum Netzteil des Lernsystems

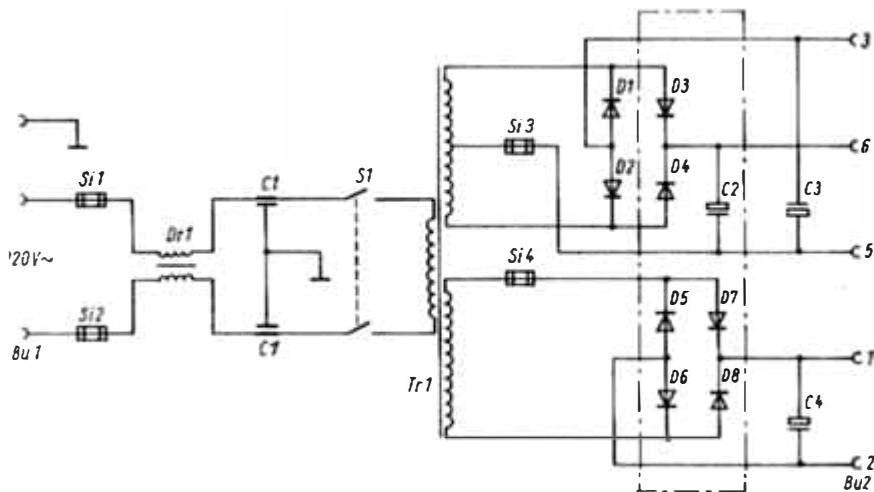


Bild A.7.1. Stromlaufplan FPS2.NT1

Tafel A.7.1. Stückliste FPS2.NT1

Lfd. Nr.	Stückzahl	Benennung	Sachnummer	Bemerkungen
1	2	Dioden SY 170/2		D1, D2
2	6	Dioden SY 320/2		D3 ... D8
3	1	Entstörkondensator D0, 1 (x) + 2 x 2500 (y)	TGL 11 840	C1
4	1	Elektrolytkondensator 1000 µF/25 V	TGL 7198	C2
5	1	Elektrolytkondensator 10000 µF/16 V	TGL 5151	C3
6	1	Elektrolytkondensator 4700 µF/16 V	TGL 5151	C4
7	1	Netztransformator LL60/20	BV	Tr1
8	1	Störschutzdrossel Typ I 2 x 0,25 mH 1,6 A 500 V	TGL 200-8402	Dr1
9	1	Netztaste 1/UNU 1 T 34,9/17, 2, 3, 2	0642.200-501	S1
10	2	G-Schmelzeinsatz T 0,8 A	TGL 0-41571	Si1, Si2
11	1	G-Schmelzeinsatz T 4,0 A	TGL 0-41571	Si3
12	1	G-Schmelzeinsatz T 1,25 A	TGL 0-41571	Si4

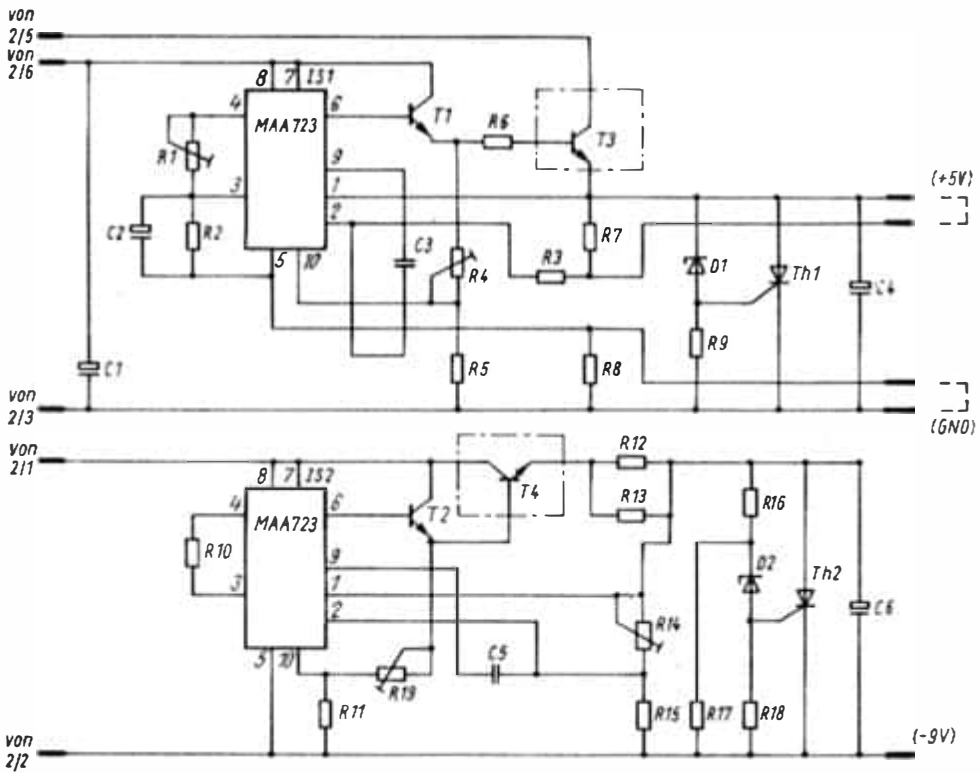


Bild A.7.2. Stromlaufplan FPS2.NT2

Tafel A.7.2. Stückliste FPS2.NT2

Lfd. Nr.	Stückzahl	Benennung	Sachnummer	Bemerkungen
1	2	Schaltkreis MAA723	Tesla	IS1, IS2
2	2	Transistor SF126E	TGL 11811	T1, T2
3	1	Transistor KU607 B 30	Tesla	T3
4	1	Transistor KU605 B 30	Tesla	T4
5	1	Z-Diode SZ600/5,6		D1
6	1	Z-Diode SZX21/5,6		D2
7	1	Thyristor ST111/1		Th1
8	1	Thyristor ST103/1		Th2
9	2	Elektrolytkondensator 220 $\mu$ F/16 V	TGL 26629	C1, C6
10	1	Elektrolytkondensator 1 $\mu$ F/63 V	TGL 26629	C2
11	2	Kondensator EDVU 220 pF N 750	TGL 24100	C3, C5
12	1	Elektrolytkondensator 1000 $\mu$ F/25 V	TGL 7198	C4
13	4	Dickschichtestellregler 513.1313 1 k $\Omega$ 10%	TGL 27423	R1, R4, R14, R19
14	1	Schichtwiderstand 250.207 TK 2,2 k $\Omega$ 5%	TGL 8728	R2
15	2	Schichtwiderstand 250.207 TK 1 k $\Omega$ 5%	TGL 8728	R3, R10
16	1	Schichtwiderstand 250.207 TK 2 k $\Omega$ 5%	TGL 8728	R5
17	1	Schichtwiderstand 25.412 10 $\Omega$ 5%	TGL 8728	R6
18	2	Schichtwiderstand 250.207 TK 3,9 $\Omega$ 5%	TGL 8728	R7, R8
19	1	Schichtwiderstand 25.412 47 $\Omega$ 5%	TGL 8728	R9
20	1	Schichtwiderstand 250.207 TK 4,7 k $\Omega$ 5%	TGL 8728	R11
21	2	Drahtwiderstand 22.616 1 $\Omega$ 5%	TGL 200-8041	R12, R13
22	1	Schichtwiderstand 250.207 TK 1,2 k $\Omega$ 5%	TGL 8728	R15
23	1	Schichtwiderstand 250.207 TK 150 $\Omega$ 5%	TGL 8728	R16
24	2	Schichtwiderstand 250.207 TK 62 $\Omega$ 5%	TGL 8728	R17, R18

# Literaturverzeichnis

- [1] *Jugel, A.*: Mikroprozessorsysteme. 2. Aufl. Berlin: VEB Verlag Technik 1980.
- [2] *Schwarz, W.; Meyer, G.; Eckhard, D.*: Mikrorechner – Wirkungsweise, Programmierung, Applikation. Berlin: VEB Verlag Technik 1980.
- [3] *Höhne, M.*: Der Mikroprozessor U 808 D, Teil 1. radio fernsehen elektronik 26 (1977) 5, S. 145–150
- [4] TESLA ROZNOV: Halbleiterkatalog 1979–1980.
- [5] *Bader, B.*: Mikrorechnersystem K 1520. radio fernsehen elektronik 28 (1979) 10, S. 616–620.
- [6] *Wollenberg, G.*: CNC 600 – Ein numerisches Steuerungssystem auf Mikrorechnerbasis. radio fernsehen elektronik 28 (1979) 11, S. 694–700.
- [7] *Eckhard, D.; Konrad, E.; Leupold, W.*: Hochintegrierte digitale Schaltungen und ihre Anwendung. 3. Aufl. REIHE AUTOMATISIERUNGSTECHNIK, Bd. 189. Berlin: VEB Verlag Technik 1980.
- [8] *Claßen, L.*: Programmierung des Mikroprozessorsystems U 880 – K 1520. REIHE AUTOMATISIERUNGSTECHNIK, Bd. 192. Berlin: VEB Verlag Technik 1981.
- [9] *Weller, W.*: Anwendung der Mikroelektronik in der Prozeßautomatisierung. 2. Aufl. REIHE AUTOMATISIERUNGSTECHNIK, Bd. 187. Berlin: VEB Verlag Technik 1981.
- [10] TGL 35333: Statischer Lese-Schreib-Speicher U 202 D.
- [11] TGL 34815: Unipolarer Festwertspeicherschaltkreis U 505 D.
- [12] TGL 37787: Festwertspeicherschaltkreis U 555 C.
- [13] TGL 26176: Unipolarer Mikroprozessorschaltkreis U 880 D.
- [14] TGL 35837: Unipolarer Parallel-Eingabe-Ausgabe-Schaltkreis U 855 D.
- [15] TGL 37001: Unipolarer Serieller Eingabe-Ausgabe-Schaltkreis U 856 D.
- [16] TGL 37002: Unipolarer Zähler/Zeitgeberschaltkreis U 857 D.

# Sachwörterverzeichnis

- Adressierungsarten 34 ff.
- AD-Wandler 275 f.
- Assembler 14, 277 f., 308 f.
  
- Basisbefehle des U880 33 ff.
  - Adressierungsarten 34 ff.
  - Befehlstypen 33
  - Flagoperationen 46 ff.
  - Mnemonikdarstellung 39 ff.
  - Operandenbeschreibung 37 f.
  - Zeitangaben 67 ff.
- Baudratenerzeugung 138 f., 155
- Bustreiber, bidirektional 218 ff.
  
- Compiler 14, 308 f.
- CMOS-Schreib/Lese-Speicher 211 ff.
  - Einsatz im FPS2 261 f.
  - Einsatz im System U880 236 f.
- CTC 139 ff.
  
- daisy chain 96 ff., 172 ff.
- Datenerhalt bei Netzausfall 239 f.
- Datenregister, 8 bit- 218 ff.
- DA-Wandler
  - Beispiel mit CTC 156 f.
  - FPS2-Baugruppe 274 f.
- Dekoder, 1-aus-8- 217
- Direkter Speicherzugriff 199 ff.
  - Anforderung 30 f., 203
  - Bearbeitung 199 ff.
  - Prioritätsauswahl 202
  - Quittierung 30 f., 203
  - Rückkehr 204
  - transparenter Betrieb 200 f.
  
- Editor 14, 277 f., 308 f.
- Eingangsschutzbeschaltung 250 f., 272
  
- Festwertspeicherbauelemente 206 ff.
  - des Systems U808 16 f.
  - des Systems U880 206 ff.
  - Einsatz im System U880 234 ff.
  - Einsatz in der FPS2 260 f.
  - PROM-Programmierung im Lernsystem 303 ff.
- Flagbeeinflussung 65
- Flagoperationen 46 ff.
- Frei programmierbare Steuerung FPS2 256 f.
  - AD/DA-Wandler 274 ff.
  - Ergänzungsbaugruppen 276 ff.
  - parallele Standardperipherien 264 ff.
  - prozeßnahe Peripherie 269 ff.
  - Prozessorbaugruppe 257 ff.
  - serielle Standardperipherie 267 ff.
  - Speicherbaugruppen 260 ff.
  - Systemkomponenten 254
  
- HALT-Zustand 32 f., 44
- handshaking
  - Ausgabemodus der PIO 89 f.
  - bidirektionaler Betrieb der PIO 93 f.
  - Eingabemodus der PIO 91 f.
  
- Indexregister 22, 36 f., 40
- Interpreter 308 f.
  
- Interrupt 71 ff., 163 ff.
  - Anforderung 31 f., 176 ff.
  - Bearbeitung in Peripherie 96 ff., 110 ff., 151 ff.
  - Freigabe 71 f.
  - maskierbarer 73, 167 ff.
  - nichtmaskierbarer 72 f., 164 ff.
  - Priorität im System 72, 163 f.
  - Prioritätsauswahl 172 ff.
  - Programmbearbeitung 100, 155, 197 ff.
  - Quittierung 31 f., 176 ff.
  - Rückkehr 199 ff.
  - Sperrung 71 f.
  - Struktur der peripheren Elemente 192 ff.
- Interruptgenerator mit CTC 160, 205, 230
- Interruptprioritätenkette 96 ff., 172 ff.
  - Einschwingprobleme 174 ff., 181 ff.
- Interruptvektor 75 f., 178 f.
  - der PIO 84 f., 96
  - der SIO 113, 127, 134
  - des CTC 144 f., 152
- Interruptvektorregister 22 f., 179
  
- K1520-Mikrorechnersystem
  - Gerätekomponenten 253
  - Systemunterlagen 254
  
- Lernsystem 279 ff.
  - Betriebssoftware 286 ff., 328 ff.
  - Konfiguration 279 f.
  - PROM-Programmierbaugruppen 302 ff.
  - Prozessorbaugruppe 281 f., 339 ff.
  - Speicherbaugruppe 282 ff., 343 ff.
  - Tastatur- und Anzeigebaugruppe 284 f., 345 ff.
- Linker 14, 308 f.
- Löschung von EPROM 210 f.
  
- Maschinenprogramm 14, 278, 308 f.
- Maschinenzyklus 26 ff., 64 ff.
  - Befehlskodeszenen 27
  - I/O-Lesen 30
  - I/O-Schreiben 30
  - Speicherlesen 28 f.
  - Speicherschreiben 28 f.
- Mikroprozessorsystem U880
  - Anwendervarianten 253 ff.
  - leistungsbestimmende Parameter 162 f.
  - Minimalsystem 223 ff.
  - Übersicht 18
- Mikroprozessor U808 16 ff.
  - Anschlußbelegung 18
  - Befehlsliste 319
  - Blockschaltbild 17
- Mikroprozessor U880 20 ff.
  - Anschlußbelegung 24 ff.
  - Befehlsliste 322
  - Blockschaltbild 21
  - Registerstruktur 21
  - technische Daten 76 ff.
- Mnemonik des U880-Assemblersprache 39 ff.
- Monitorbetriebsprogramm 286 ff., 328 ff.
- MOS-Taktreiber 231 f.
  
- Netzausfallsignalisation 239

- Objektkode 14  
 Objektprogramm 14, 277f., 308 f.
- PAP 277 f.
- Parallele Ein-/Ausgabe-Einheit U855 78 ff.  
 Anschlußbelegung 81 ff.  
 Anwendung 98 ff., 223 ff., 233, 244ff.  
 Betriebsarten 89 ff.  
 Interruptbearbeitung 96 ff.  
 Programmierung 84 ff.  
 Struktur 79 f.  
 technische Daten 101 ff.
- Periphere Baugruppen 244 ff.  
 Anforderungen 230  
 Einsatz in FPS2 264ff.  
 im Minimalsystem 227  
 prozennahe Schnittstellen 250 ff.  
 serielle Standardschnittstelle 245 ff.  
 SIF-1000-Anschlußsteuerung 244 ff.
- PIO 78 ff.
- Potentialtrennung 250 ff., 267, 269ff.
- Programmierarbeit 13 ff., 277 f., 308 f.  
 bei DMA-Betrieb 204 f.  
 bei Einbindung der PIO 100 ff.  
 bei Einbindung des CTC 155  
 bei Interruptbedienung 197 ff.
- Programmiersprachen, höhere 13, 308 f.
- Programmierung (Zusammenfassung)  
 der PIO 88  
 der SIO 125 ff.  
 des CTC 147
- Programmzähler 22, 27f.
- PROM/EPROM-Programmierung  
 U555 209f.  
 U555 im Lernsystem 306f.  
 U551/U552 im Lernsystem 303 ff.
- Prozessorbaugruppe 231 ff.  
 Anforderungen 227 f.  
 der FPS2 257 ff.  
 des Lernsystems 281 f., 339 ff.  
 im Minimalsystem 223 ff.
- Quellprogramm 14, 277f., 308 f.
- Refresh 25, 27f., 240ff.  
 Refreshregister 23  
 Registerstruktur des U880 21 ff.  
 RS-232-Interface 245 ff.
- Schlafzustand  
 bei RAM 212, 236ff.  
 bei ROM 236
- Schreib/Lese-Speicherbauelemente  
 des Systems U808 18  
 des Systems U880 211 ff.
- Schreib/Lese-Speicher, dynamische 211 ff.  
 Einsatz im System U880 240ff.  
 Einsatz in der FPS2 262ff.
- Schreib/Lese-Speicher, statische 211 ff.  
 Einsatz im System U880 236ff.  
 Einsatz in der FPS2 262
- SDLC 112, 123f.
- Serielle Ein-/Ausgabe-Einheit U856 105 ff.  
 Anschlußbelegung 107 ff.  
 Anwendung 134 ff., 245 ff.  
 Betriebsarten 110 ff.  
 Programmierung 124 ff.  
 Struktur 106 f.  
 technische Daten 135 ff.
- Serielle Standardschnittstelle 245 ff.  
 Einsatz in der FPS2 267 ff.
- SIF-1000-Anschlußsteuerung 244 ff.  
 Einsatz in der FPS2 264ff.
- SIO 105 ff.
- Speicherbaugruppen 234 ff.  
 Anforderungen 228 ff.  
 der FPS2 260 ff.  
 des Lernsystems 282 ff., 343 ff.  
 im Minimalsystem 225 f.
- Stackpointer 22, 66ff.
- Stromversorgung  
 der FPS2 276 f.  
 des Lernsystems 286, 348 f.  
 für prozennahe Peripheriebaugruppen 252 f., 274  
 Signalisation bei Netzausfall 239, 261  
 von Speicherbaugruppen 234 ff.
- Systemprobleme  
 Einsatz dynamischer Speicher 231 f., 240 ff.  
 Interruptkaskadierungslogik 174 ff., 181 ff.
- Systemtakt 24  
 Ableitung einer Echtzeituhr 139  
 Ableitung von Baudraten 138 f., 155  
 Takttreiber 231 f.  
 Taktzustände 26 ff.
- Takttreiber 231 f.
- Tastaturabfrageeinheit 100 ff.
- Tastatur- und Anzeigebedienbaugruppe 284 f., 345 ff.
- Transparenter DMA 200 f.
- V-24-Schnittstelle 245 ff.
- WAIT-Zustand 25 ff.
- Zähler/Zeitgeber U857 139 ff.  
 Anschlußbelegung 142 ff.  
 Anwendung 154 ff., 245 ff., 274 f.  
 Betriebsarten 148 ff.  
 Interruptbearbeitung 151 ff.  
 Programmierung 144 ff.  
 Struktur 140 f.  
 technische Daten 157 ff.
- Zugriffszeit  
 adressenaktiver Speicher 228  
 chipaktiver Speicher 229