mikroelektronik

Integrierte
Schaltkreise

RFT

Hinweise zur Anwendung

Handbuch U8000

Mikroprozessor CPU U8001/U8002 -Technische Beschreibung-

Applikation

U 8000 – Handbuch

Technische Beschreibung

CPU U 8001 / U 8002

veb mikroelektronik ›karl marx‹ erfurt
stammbetrieb

DDR – 5010 Erfurt, Rudolfstraße 47 Telefon 580 Telex 061 306

Inhalt

Kapitel 1

1.	Prozessor U 8000 im Überblick	7
1.1.	Einleitung	7
1.2.	Allgemeine Organisation	7
1.3.	Architektur-Merkmale	8
1.3.1.	Mehrzweck-Registerdatei	8
1.3.2.	Befehlssatz	9
1.3.3.	Datentypen	9
1.3.4.	Adressierungsarten	10
1.3.5.	Mehrfachspeicheradreßräume	10
1.3.6.	Betriebsart System/Normal	10
1.3.7.	Getrennte E/A-Adreßräume	11
1.3.8.	Interrupt-Struktur	11
1.3.9.	Multi-Processing	12
1.3.10.	Großer Adreßraum für den U 8001	12
1.3.11.	Segmentierte Adressierung des U 8001	12
1.3.12.	Speicherverwaltung	13
1.4.	Vorzüge der Architektur	13
1.4.1.	Kodedichte	13
1.4.2.	Compilereffizienz	14
1.4.3.	Betriebssystemunterstützung	14
1.4.4.	Möglichkeit vieler Arten von Datenstrukturen	15
1.4.5.	Zwei CPU-Versionen U 8001 und U 8002	16
1.5.	Befehlserweiterungsfähigkeit	16
1.6.	Zusammenfassung	16

Kapitel 2

2.	Architektur	17
2.1.	Einleitung	17
2.2.	Allgemeines	17
2.3.	Hardware-Interface	20
2.3.1.	Adressen-/Datenleitungen	20
2.3.2.	Segmentnummer (nur U 8001)	20
2.3.3.	Bus-Zeitsteuerung	21
2.3.4.	Status	21
2.3.5.	CPU-Steuerung	21
2.3.6.	Bussteuerung	22
2.3.7.	Interrupts	22
2.3.8.	Segment-Trap-Anforderung (nur U 8001)	22
2.3.9.	Multi-Mikro-Steuerung	22
2.3.10.	Systemeingänge	22
2.4.	Zeitsteuerung	22
2.5.	Adreßräume	23
2.5.1.	Speicheradreßraum	23
2.5.2.	E/A-Adreßraum	24
2.6.	Mehrzweck-Register	24
2.7.	Spezialregister	26
2.7.1.	Programm-Status-Register	27
2.7.2.	Program Status Area Pointer (PSAP)	28
2.7.3.	Refresh-Zähler	28

2.8. Befehlsausführung 29
2.9. Befehle 29
2.9.1. Befehlsformate 30
2.10. Datentypen 31
2.11. Adressierungsarten 31
2.12. Erweiterte Prozessorarchitektur 32
2.13. Ausnahmen 32
2.13.1. Rücksetzen (Reset) 32
2.13.2. Traps 32
2.13.3. Interrupts 33
2.13.4. Trap- und Interruptservice-Prozeduren 33

Kapitel 3

3. Adreßräume 34
3.1. Einleitung 34
3.2. Adreßraumtypen 34
3.3. E/A-Adreßräume 35
3.4. Speicheradreßräume 35
3.4.1. Adressierbare Datenelemente 35
3.4.2. Segmentierte und nichtsegmentierte Adressen 36
3.4.3. Segmentierung und Speicherverwaltung 37

Kapitel 4

4. CPU-Betrieb 39
4.1. Einleitung 39
4.2. Betriebszustände 39
4.2.1. Arbeitszustand (running state) 40
4.2.2. Stop/Refreshzustand 40
4.2.3. Bus-Abschalt-Zustand 40
4.2.4. Wirkung von Reset 40
4.3. Befehlsausführung 41
4.3.1. Betriebsarten des Arbeitszustandes 42
4.3.2. Segmentierte und nichtsegmentierte Betriebsart 42
4.3.3. Betriebsart Normal und System 42
4.4. Erweiterte Befehle 43

Kapitel 5

5. Adressierungsarten 45
5.1. Einleitung 45
5.2. Verwendung der CPU-Register 45
5.3. Beschreibung der Adressierungsarten 47
5.4. Beschreibungen und Beispiele (U 8002 und U 8001 in nichtsegmentierter Betriebsart) 48
5.4.1. Register (R) 48
5.4.2. Direktwert (IM) 49
5.4.3. Indirektes Register (IR) 49
5.4.4. Direkte Adressierung (DA) 50
5.4.5. Indiziert (X) 51
5.4.6. Relative Adresse (RA) 52
5.4.7. Basisadresse (BA) 53
5.4.8. Basis Indiziert (BX) 54

5.5.	Beschreibung und Beispiele (U 8001 in segmentierter Betriebsart)		55
5.5.1.	Register	(R)	56
5.5.2.	Direktwert	(IM)	56
5.5.3.	Indirektes Register	(IR)	57
5.5.4.	Direkte Adressierung	(DA)	58
5.5.5.	Indiziert	(X)	59
5.5.6.	Relative Adresse	(RA)	60
5.5.7.	Basisadresse	(BA)	62
5.5.8.	Basis Indiziert	(BX)	63

Kapitel 6

6.	Ausnahmen	65
6.1.	Einleitung	65
6.2.	Interrupts	65
6.2.1.	Nichtmaskierbarer Interrupt (NMI)	65
6.2.2.	Vektorisierter Interrupt (VI)	65
6.2.3.	Nichtvektorisierter Interrupt (NVI)	66
6.3.	Traps	66
6.3.1.	Trap bei erweitertem Befehl	66
6.3.2.	Trap bei privilegiertem Befehl	66
6.3.3.	Trap bei System-Call	66
6.3.4.	Segment-Trap	66
6.4.	Reset	66
6.5.	Sperren von Interrupts	67
6.6.	Interrupt- und Trapbehandlung	67
6.6.1.	Bestätigungszyklus	67
6.6.2.	Statusretten	68
6.6.3.	Laden des neuen Programmstatus	69
6.6.4.	Ausführung der Serviceroutine	70
6.6.5.	Rückkehr von einem Interrupt oder Trap	70
6.7.	Priorität	71

Kapitel 7

7.	Refresh	72
7.1.	Einleitung	72
7.2.	Refreshzyklen	72
7.3.	Periodisches Refresh	72
7.4.	Refresh im Stop-Zustand	73

Kapitel 8

8.	Externes Interface	74
8.1.	Einleitung	74
8.2.	Busoperationen	74
8.3.	CPU-Pins	75
8.3.1.	Zugriffsteuerungspins	75
8.3.2.	Bussteuerpins	76
8.3.3.	Interrupt/Trap-Pins	76
8.3.4.	Multi-Mikro-Pins	76
8.3.5.	Prozessor-Steuerung	77
8.4.	Zugriffe	78
8.4.1.	Wait	80

8.4.2. Speicherzugriffe 80
8.4.3. E/A-Zugriffe 82
8.4.4. EPU-Transferzugriffe 83
8.4.5. Interrupt/Trap-Acknowledge 84
8.4.6. Interne Operation und Refresh-Zyklen 85
8.5. Wechselwirkung zwischen CPU und EPU 87
8.6. Anforderungen (Requests) 88
8.6.1. Interrupt/Trap-Anforderungen 88
8.6.2. Busanforderung 89
8.6.3. Ressource-Anforderung 90
8.6.4. Stopanforderung 90
8.7. Reset 91

Kapitel 9

9. Hardware-Informationen 93

Kapitel 10

10. Kennwerte 94

Kapitel 1

1. Prozessor U 8000 im Überblick

1.1. Einleitung

Dieses Kapitel enthält eine zusammenfassende Beschreibung der modernen Architektur des Mikroprozessors U 8000 unter besonderer Berücksichtigung derjenigen strukturellen Eigenschaften, durch die sich die CPU U 8000 von ihren Vorgängern unterscheidet. Das Kapitel 2 enthält eine vollständige Darstellung des Aufbaus mit ausführlicher Beschreibung der verschiedenen Eigenschaften des Prozessors.

1.2. Allgemeine Organisation

Der Typ U 8000 wurde für eine Vielzahl von Anwendungsmöglichkeiten entwickelt, die von relativ einfachen bis zu großen und komplizierten Aufgaben reichen können. Die CPU U 8000 wird in zwei Versionen angeboten, U 8001 und U 8002. Beide können mit weiteren Systemschaltkreisen wie Speicherverwaltungs-Einheit, DMA-Steuereinheit, serielle und parallele E/A-Steuereinheiten sowie Erweiterungsprozessoren (EPU's) ergänzt werden. Zusammen mit anderen Bauelementen der U 8000-Familie ermöglicht die moderne Architektur dieses LSI-Mikroprozessors eine Anpassungs- und Leistungsfähigkeit, die für Mini- oder Großcomputer charakteristisch ist.
Die strukturellen Hauptmerkmale der CPU U 8000, welche Durchsatz- und Verarbeitungsleistung erhöhen, sind eine Mehrzweckregisterdatei, System- und Normal-Betriebsart, Mehrfachadreßräume, ein leistungsfähiger Befehlssatz, zahlreiche Adressierungsarten, mehrere Stacks, komfortable Interruptstrukturen, ein großer Vorrat an Datentypen, getrennte E/A-Adreßräume sowie - für den Typ U 8001 - ein großer Adreßraum und segmentierte Speicheradressierung. Jede dieser Eigenschaften wird im folgenden Abschnitt ausführlich behandelt.
Diese Architektur-Eigenschaften ergeben einen leistungsfähigen und vielseitigen Mikroprozessor, der wiederum solche Vorzüge wie Kodedichte, Compilereffizienz, Möglichkeiten für typische Betriebssystemoperationen sowie komplexe Datenstrukturen aufweist, die im Abschnitt 1.3. behandelt werden.
Die CPU wurde so ausgelegt, daß ein leistungsfähiges Speicherverwaltungssystem angewandt werden kann, um die Ausnutzung des Hauptspeichers zu verbessern und Schutzmöglichkeiten für das System vorzusehen. Dies wird im Abschnitt 1.3.12. beschrieben. Die CPU U 8000 als äußerst komplizierter Prozessor besitzt keine interne Speicherverwaltung, dafür aber bestimmte Merkmale, die die Integration einer externen Speicherverwaltungseinheit in eine U 8000-Systemkonfiguration ermöglichen.
Schließlich wurde besonderer Wert auf die Schaffung eines sehr allgemeinen Mechanismus zur Erweiterung des Grundbefehlsvorrates durch Anwendung externer Elemente gelegt, welche als EPU's (Extended Processing Unit, Erweiterungsprozessor) bezeichnet werden.
Im allgemeinen dient eine EPU zur Entlastung der CPU von komplizierten und zeitraubenden Aufgaben. Typische Aufgaben für spezielle EPU's sind Gleitkommaarithmetik, Datenbankverwaltung, Überwachungsfunktionen, Netzwerk-Interfaces und viele andere.
Diese Punkte werden im Abschnitt 1.5. behandelt.

1.3. Architektur-Merkmale

Die Architektur der CPU U 8000 wird charakterisiert durch sechzehn 16-Bit-Mehrzweck-Register, sieben Datentypen von Einzel-Bits bis zu 32-Bit langen Worten und Bytefolgen, acht anwender-wählbaren Adressierungsarten sowie einen leistungsfähigeren Befehlsvorrat als die meisten Minicomputer. In Kombination mit den verschiedenen Datentypen und Adressierungsarten bilden die 110 einzelnen Befehlstypen den umfangreichen Vorrat von 414 Befehlen. Der Vorrat weist einen hohen Regelmäßigkeitsgrad auf; mehr als 90 % der Befehle sind für jede der fünf Hauptadressierungsarten mit Datentypen von 8-Bit-Bytes, 16-Bit-Worten und 32-Bit-Langworten geeignet. Die CPU erzeugt Statussignale, welche die Art der ausgeführten Buszugriffe anzeigen. Diese können zur Realisierung komplizierter Systeme mit Mehrfachadreßräumen dienen, d. h. Speicherbereiche werden spezifischen Zwecken zugeordnet. Die CPU verfügt über die zwei Betriebsarten System und Normal, die angewandt werden können, um Betriebssystemfunktionen von allgemeinen Anwenderprogrammen zu trennen. E/A-Operationen sind von Speicherzugriffen getrennt, wodurch die Leistungs- und Erweiterungsfähigkeit von Systemen auf U 8000-Basis erhöht wird. Eine komfortable Interrupt-Struktur ermöglicht den hocheffektiven Betrieb peripherer E/A-Einheiten. Außerdem gestattet die EPU-Fähigkeit des Prozessors die Verlagerung vieler zeitaufwendiger Aufgaben auf externe Einheiten. Spezielle Eigenschaften des U 8000 erleichtern die Realisierung von Mehrfachprozessorsystemen. Weiterhin verfügt die CPU U 8001 über eine große, segmentierte Adressierungsmöglichkeit, welche die Anwendbarkeit dieses Mikroprozessors für Großsystemanwendungen erheblich verbessert.

1.3.1. Mehrzweck-Registerdatei

Das Herz der CPU U 8000 bildet eine Datei aus sechzehn 16-Bit-Mehrzweck-Registern.
Diese Mehrzweck-Register geben dem U 8000 seine Leistungsfähigkeit und Anpassungsfähigkeit und tragen zu seiner regelmäßigen Befehlsstruktur bei.
Mehrzweck-Register können als Akkumulatoren, Zeiger auf eine Speicherzelle oder Indexregister verwendet werden. Ihr Hauptvorteil ist, daß der besondere Verwendungszweck, dem sie dienen, sich im Laufe eines Programms mit den veränderten Erfordernissen des Programms ebenfalls ändern kann. Somit vermeidet die Mehrzweck-Registerdatei die kritischen Engpässe einer Registerarchitektur mit zweckorientiertem Charakter, bei der der Inhalt bestimmter Register gerettet und wiederhergestellt werden muß, wenn mehr Register eines besonderen Typs erforderlich sind, als sie der Prozessor liefern kann. Die Registerdatei der CPU U 8000 kann auf verschiedene Weise adressiert werden: als sechzehn Byte-Register, (Belegung der oberen Hälfte der Datei), als sechzehn Wort-Register oder - durch Anwendung des Registerpaarmechanismus - als acht Langwort - (32-Bit-Register) bzw. als vier Doppellangwort-Register (64-Bit-Register). Durch diese Registerflexibilität ist es z. B. für einen U 8000-Anwender nicht nötig, ein 32-Bit-Register nur für ein Datenbyte zu vergeben. Im U 8000 können die Register äußerst effektiv eingesetzt werden.

1.3.2. Befehlssatz

Eine der hervorragenden Eigenschaften des U 8000 ist sein leistungsfähiger Befehlssatz. Der Befehlssatz ist ein Maß für die Anpassungsfähigkeit und Vielseitigkeit eines Computers. Die Realisierung einer bestimmten Operation als Hardware-Lösung bedeutet Speichereinsparung und Geschwindigkeitserhöhung. Es ist häufig wichtiger, daß der Operationssatz möglichst vollständig für die einzelnen Datentypen anwendbar ist, als über ausgefallene Spezialoperationen zu verfügen, welche die Leistungsfähigkeit nur unwesentlich beeinflussen können. Die CPU U 8000 bietet einen vollen Satz arithmetischer, logischer, Verzweigungs-, E/A-, Schiebe-, Rotations- sowie Stringbefehle. Außerdem sind spezielle Befehle für Multi-Processing sowie typische Funktionen für höhere Programmiersprachen und Betriebssystemfunktionen vorhanden.
Das allgemeine Prinzip des Befehlsvorrates besteht in Register-Speicheroperationen mit zwei Operanden, welche Register-Register-Operationen als speziellen Untervorrat enthalten. Zur Verbesserung der Kode-Dichte werden einige Speicher-Speicher-Operationen zur Bearbeitung von Strings (Datenfolgen) verwendet. Die am häufigsten vorkommenden Operationen (wie A ⟵ A + B) werden durch das Zwei-Adressen-Format wiedergegeben. Dadurch, daß einer der Operanden in einem schnell zugänglichen Mehrzweck-Register vorhanden ist, wird auch die Verwendung von Zwischenergebnissen, welche während einer Berechnung erzeugt werden, erleichtert.
Die Mehrzahl der Operationen betrifft Byte-, Wort- oder Doppelwort-Operanden, wobei ein hoher Regelmäßigkeitsgrad gewährleistet ist. Ebenfalls im Befehlsvorrat enthalten sind kompakte Einzelwortbefehle für die am häufigsten verwendeten Operationen (beispielsweise relative Sprünge).
Der Befehlssatz enthält einige bemerkenswerte Zusätze zum Standard-Repertoire früherer Mikro-Prozessoren. Die Befehlsgruppe Load (Laden) und Exchange (Austausch) wurde hinsichtlich der Unterstützung von Betriebssystemfunktionen und der Übersetzung bestehender prozessorprogramme erweitert. Die üblichen Arithmetikbefehle können jetzt mit Operanden höherer Genauigkeit arbeiten, während auch Hardware-Multiplikations- und Divisionsbefehle aufgenommen wurden. Die Bitmanipulationsbefehle können für die Festlegung der Bitstelle innerhalb eines Bytes oder Wortes neben einem festen Wert (Bit statisch) auch einen berechneten Wert (Bit dynamisch) verwenden.
Die Rotations- und Schiebebefehle sind erheblich flexibler als in früheren Mikroprozessoren. Die Stringbefehle sind nützlich für die Umwandlung verschiedener Zeichenkodes. Mehrfachprozessor-Konfigurationen werden durch Spezialbefehle unterstützt.

1.3.3. Datentypen

Die Architektur des U 8000 ermöglicht viele Datentypen.
Ein Datentyp gilt als implementiert, wenn seine Hardwaredarstellung existiert, also Befehle direkt mit ihm arbeiten können. Zusätzliche Datentypen können stets mit Hilfe der Basis-Datentypen gebildet werden, jedoch bietet die Realisierung durch die Hardware schnellere und komfortablere Operationen. Der Grunddatentyp ist das Byte, welches gleichzeitig das adressierbare Basiselement darstellt. Die Architektur ermöglicht auch folgende Datentypen: Worte (16 Bit), Langworte (32 Bit), Bytefolgen und Wortfolgen. Außerdem kann mit einem einzelnen Bit innerhalb eines Bytes oder Wortes gearbeitet werden, indem es mit seiner Stellenziffer adressiert wird. BCD-Zahlen-Verarbeitung wird ermöglicht durch Darstellung von zwei 4-Bit-Ziffern in einem Byte. Felder werden durch die indizierte Adressierung unterstützt (siehe Abschn. 1.3.4. und Kapitel 5).

1.3.4. Adressierungsarten

Die Adressierungsart, mit der ein Operand in einem Befehl angegeben wird, bestimmt, wie seine Adresse erzeugt wird. Die CPU U 8000 bietet acht Adressierungsarten. Gemeinsam mit der großen Anzahl von Befehlen und Datentypen verbessern diese die Verarbeitungsleistung der CPU.
Die Adressierungsarten sind Register, Direktwert, Indirekte Register, Direkte Adresse, Indiziert, Relative Adresse, Basisadresse und Basis Indiziert. Weitere Adressierungsarten werden bei speziellen Befehlen angewendet (z.B. automatisches Erhöhen der Adresse). Die ersten fünf der o. a. Arten sind Grundadressierungsarten, welche am häufigsten verwendet werden und für die meisten Befehle gelten, die mehr als eine Adressierungsart haben. (Beim U 8002 sind Basis-Adressierungen und indizierte Adressierung identisch; beim U 8001 besteht bei Verwendung der Speicherverwaltungseinheit die Möglichkeit der Basis Adressierung für alle Befehle).

1.3.5. Mehrfachspeicheradreßräume

Die CPU U 8000 ermöglicht die Anwendung von Mehrfachadreßräumen. Wenn die CPU U 8000 eine Adresse erzeugt, so gibt sie auch Signale aus, die die jeweilige interne Aktivität anzeigen, welche zur Speicheranforderung führte: Befehlsaufruf, Operanden- oder Stackzugriff. Diese Information kann auf zweierlei Art verwendet werden: einmal zur Erhöhung des dem Prozessor zur Verfügung stehenden Speicherraumes (z. B. durch Unterbringung von Programmen in einem Raum und von Daten in einem anderen), oder zum Schutz von Speicherbereichen, indem nur bestimmte Zugriffstypen für diesen Bereich gestattet werden (z. b. indem Befehle nur aus einem Bereich geholt werden dürfen, der zur Aufnahme der Software des Anwenders reserviert ist). Zur Erreichung dieser Schutzmerkmale dient die Speicherverwaltungseinheit, die hierfür die von der CPU gebildeten Statusinformationen auswertet.

1.3.6. Betriebsart System/Normal

Die CPU U 8000 kann in dem System-Modus oder im Normal-Modus betrieben werden. In der Betriebsart 'System' können alle Befehle ausgeführt und auf alle CPU-Register zugegriffen werden. Diese Betriebsart ist zur Anwendung durch Programme vorgesehen, welche Betriebssystemfunktionen ausführen. Im Normal-Modus werden einige Befehle nicht ausgeführt (z. B. E/A-Operationen), außerdem besteht zu den Steuerregistern der CPU kein Zugriff. Im allgemeinen dient diese Betriebsart für Anwenderprogramme. Diese Trennung der CPU-Ressourcen erhöht die Systemzuverlässigkeit, da in der Betriebsart 'Normal' arbeitende Programme keinen Zugriff zu denjenigen Bereichen der CPU haben, die sich mit zeitkritischen Vorgängen oder System-Interface-Aufgaben befassen.
Bei mit Fehlern behafteten Programmen, welche im Normal-Modus laufen, können diese Fehler zum Zwecke der Fehlersuche immer wieder durch einfache erneute Ausführung des Programms mit seinen Originaldaten reproduziert werden. Programme, die Möglichkeiten nutzen, die nur in der Betriebsart 'System' vorhanden sind, können aus Gründen der Zeitabhängigkeit (z. B. aufgrund der Plattenanforderungshäufigkeit oder der Position des Plattenarmes) Fehler aufweisen, die schwerer zu suchen sind, da diese Fehler nicht leicht reproduziert werden können.

Somit besteht die bevorzugte Methode der Programmentwicklung in der Aufgabenteilung, und zwar in einen Teil, der ohne die nur im System-Modus verfügbaren Ressourcen ausgeführt werden kann (was normalerweise den größten Teil der Aufgabe ausmacht) und einen Teil, welcher die Ressourcen der System-Betriebsart erfordert. Das klassische Vorbild dieser Teilung bilden die derzeitigen Minicomputer und Großrechner; das Betriebssystem läuft im System-Modus ab, während die einzelnen Anwender ihre Programme in der Betriebsart 'Normal' schreiben. Zur weiteren Unterstützung der Teilung in die Betriebsarten System/Normal gibt es zwei Ausführungen des Stackpointers: ein Stack für den System- und ein zweiter für den Normal-Modus. Beide erleichtern die beim Auftreten von Interrupts oder Traps erforderliche Task-Umschaltung. Um sicherzustellen, daß der "Normal"-Stack frei ist von Systeminformationen, wird die beim Auftreten von Interrupts oder Traps zu rettende Information stets auf den System-Stack abgelegt, bevor der neue Programmstatus geladen wird.

1.3.7. Getrennte E/A-Adreßräume

Die Architektur des U 8000 unterscheidet zwischen Speicher- und E/A-Räumen und erfordert deshalb spezifische E/A-Befehle. Diese strukturelle Trennung gestattet einen besseren Schutz und besitzt ein größeres Erweiterungspotential. Die Verwendung getrennter E/A-Räume bewahrt auch den begrenzten Datenspeicherraum des U 8002 vor Einschränkungen durch die E/A-Adressierung. Es gibt zwei vollständig getrennte E/A-Adreßräume, und zwar Standard-E/A und Spezial-E/A. Der Hauptvorteil von zwei Räumen besteht darin, daß zwei Arten peripherer Ergänzungsschaltkreise vorsehbar sind, periphere Standard-E/A-Elemente und periphere Spezial-E/A-Elemente wie die Speicherverwaltungseinheit MMU U 8010, welche dann nicht auf Standard-E/A-Befehle, sondern auf Spezial-E/A-Befehle ansprechen. Ein zweiter Vorteil dieser beiden Räume ist, daß sie die Angliederung von peripheren 8-Bit-Einheiten an die acht niederwertigen Bits (Standard E/A) oder die acht höherwertigen Bits (Spezial-E/A) des Adreß-/Datenbusses des Prozessors gestatten.
Die erhöhten Geschwindigkeitsanforderungen zukünftiger Mikroprozessoren werden wahrscheinlich durch spezielles Anpassen an die Zeitbedingungen der Speicher- und E/A-Einheiten und durch Ermöglichung gleichzeitigen E/A- und Speicherzugriffs erreicht. Diese zukünftigen Möglichkeiten erfordern heute schon eine strukturelle Trennung. Adressierung der E/A-Einheiten im Speicheradreßraum (memory-mapped I/O) ist weiterhin möglich, bedeutet aber den Verlust an Schutz- und Erweiterungsmöglichkeiten.

1.3.8. Interrupt-Struktur

Die komplizierte Interruptstruktur des U 8000 gestattet dem Prozessor, während er auf peripherenEreignisse wartet, andere wichtige Aufgaben durchzuführen. Die Vermeidung einer periodischen Abruftechnik (Polling) und von Warteschleifen (die normalerweise verwendet werden, um zu ermitteln, wann ein Gerät zur Datenübertragung bereit ist) erhöht den Durchsatz des Systems. Die CPU besitzt drei Interruptebenen. Ein nicht-maskierbarer Interrupt wird an solche Ereignisse vergeben, die sofortiges Handeln erfordern, um die Integrität des Systems zu bewahren. Außerdem gibt es zwei Arten maskierbarer Interrupts: nicht vektorisierte und vektorisierte Interrupts. Letztere gewährleisten einen automatischen Aufruf von Interruptbearbeitungsroutinen in Abhängigkeit vom Vektor, der der CPU U 8000 durch die periphere Einheit übergeben wird.

Zur Interrupt-Behandlung besitzt der U 8000 ein Prioritätssystem. Vektorisierte Interrupts haben eine höhere Priorität als nicht vektorisierte. Dieses Prioritätsschema gestattet die wirksame Steuerung einer Vielzahl peripherer Geräte in einem U 8000-System. Bei einem Interrupt wird in einem speziellen Stack neben dem aktuellen Programmzustand ein Kode abgelegt, welcher die Ursache für die Programmunterbrechung beschreibt. Dies gestattet das Auftreten rekursiver Aufgabenwechsel (recursive task switching), wobei der Normal-Stack von der Systeminformation unberührt bleibt.
Der für die Interruptserviceroutine zu übernehmende neue Programmstatus wird von einem besonderen Bereich im Speicher, dem Program Status Area (PSA, Programmstatusbereich), geladen. Seine Lage wird durch das CPU-Register Program Status Area Pointer (PSAP) festgelegt. Diese Stack- und PSAP-Organisation wurde auch gewählt, um bei Weiterentwicklungen des Prozessors, mit zur Architektur neu hinzukommenden Interrupts oder Traps, strukturell kompatibel zu bleiben.

1.3.9. Multi-Processing

Die erhöhte Rechenleistung des U 8000 erlaubt die Entwicklung nichtzentraler Verarbeitungssysteme mit vielen preisgünstigen Mikroprozessoren, welche Teilprozesse abarbeiten.
Der U 8000 verfügt über einige grundlegende Mechanismen, welche die gemeinsame Benutzung von Adreßräumen durch verschiedene Mikroprozessoren gestatten. Dies wird durch große segmentierte Adreßräume und die Möglichkeit externer Speicherverwaltung erreicht. Ebenso ist ein Ressourcenanforderungsbus vorhanden, über den in Verbindung mit der Software die Vergabe gemeinsamer kritischer Ressourcen an eine CPU vorgenommen wird.

1.3.10. Großer Adreßraum für den U 8001

Für viele Anwendungszwecke ist ein Grundadreßraum von 64 KByte nicht ausreichend. Ein großer Adreßraum erhöht den Anwendungsbereich eines Systems, weil große, komplexe Programme und Datensätze in einem Speicher bleiben können und nicht geteilt und nacheinander in den kleinen Speicher geholt und wieder ausgelagert werden müssen. Programm- und Datenverwaltung werden durch einen großen Adreßraum sehr vereinfacht. Außerdem verringern große Adreßräume und -speicher die Notwendigkeit der Minimierung der Programmgröße und gestatten die Verwendung höherer Programmiersprachen. Die segmentierte Version des U 8000 erzeugt 23 Bit Adressen für einen Grundadreßraum von 8 Megabyte (8388608 Byte).

1.3.11. Segmentierte Adressierung des U 8001

Die segmentierte Version der CPU U 8000 gliedert ihre 23 Bit Adressen auf in eine 7 Bit Segmentnummer und einen 16 Bit Segmentoffset.
Die Segmentnummer dient als logischer Name eines Segments; sie wird durch die effektive Adressenberechnung (z.B. durch Indizieren) nicht geändert. Dies entspricht der typischen Verwendung eines Speichers durch ein Programm: ein Teil des Speichers wird für Befehle reserviert, ein anderer für Daten. In einem segmentierten Adreßraum könnten die Befehle in einem Segment bleiben (oder mehrere unterschiedliche Module in verschiedenen Segmenten), während jeder Datensatz in einem eigenen Segment abgelegt werden könnte. Ein Vorteil der Segmentierung ist es, daß sie die Adressenberechnung und -verschiebung beschleunigt. So gestattet Segmentierung die Verwendung langsamerer Speicher, als dies bei linearen Adressierverfahren der Fall ist. Außerdem bieten Segmente eine bequeme Art der Speicherteilung, so daß auch jedem Teil besondere Zugriffsattribute zugeordnet werden können (z. B. "Nur-lesbar"). Das beim U 8000 angewendete Verfahren der Segmentierung (gleichzeitiger Zugriff zu einer großen Anzahl von Segmenten) ist nötig, wenn alle Vorteile der

Segmentierung realisiert werden sollen. Zum Beispiel würde einem System mit dem direkten Zugriff zu nur vier Segmenten die erforderliche Flexibilität fehlen, und es wäre durch Adreßraumbegrenzungen beschränkt.

1.3.12. Speicherverwaltung

Speicherverwaltung bedeutet Realisierung von dynamischer Verschiebbarkeit, Schutzfunktionen und Mehrfachnutzung von Speicherbereichen. Sie bietet folgende Vorteile: Die logische Struktur des Speicherraumes ist unabhängig vom tatsächlichen physischen Speicherplatz der Daten. Der Anwender ist vor Fehlern wie beispielsweise der versuchten Datenausführung geschützt. Weiterhin dient sie der Verhinderung des unbefugten Zugriffes zu Speicherressourcen und dem Schutz der Betriebssystemfunktionen vor Fehlern des Anwenders. Die vom Programmierer festgelegte, von den Befehlen verwendete und von der segmentierten CPU U 8000 ausgegebene Adresse wird logische Adresse genannt. Das externe Speicherverwaltungssystem nimmt die logischen Adressen und überträgt sie in die für den Speicherzugriff erforderlichen physischen Adressen. Dieser Adreßübertragungsprozeß wird Verschiebung genannt, wodurch die Anwendersoftware unabhängig vom physischen Speicher wird.
So braucht der Anwender nicht genau festzulegen, wo die Information tatsächlich im Speicher ihren räumlichen Platz findet. Die segmentierte CPU U 8000 unterstützt die Speicherverwaltung sowohl mit einer segmentierten Adressierung als auch mit Programmstatusinformationen. Ein segmentierter Adreßraum erlaubt die unterschiedliche Behandlung einzelner Segmente.
Die von der CPU erzeugte Programmstatusinformation gestattet die Überwachung der Art eines jeden Speicherzugriffes durch eine externe Speicherverwaltungseinheit. So können unerlaubte Zugriffsarten unterdrückt und Speichersegmente gegen unbeabsichtigte oder unerwünschte Anwendungsarten geschützt werden. Beispielsweise könnten Systemtabellen gegen direkten Anwenderzugriff geschützt werden. Diese zusätzliche Schutzmöglichkeit erlangt um so mehr Bedeutung, wenn Mikroprozessoren für große und komplizierte Aufgaben eingesetzt werden.

1.4. Vorzüge der Architektur

Insgesamt ergeben die Merkmale der U 8000-Architektur verschiedene wesentliche Vorzüge: Verbesserung der Kodedichte, Compilereffizienz, Unterstützung von Betriebssystemfunktionen sowie Ermöglichung von Datenstrukturen höherer Programmiersprachen.

1.4.1. Kodedichte

Die Kodedichte beeinflußt sowohl die Prozessorgeschwindigkeit als auch die Speicherausnutzung. Ein kompakter Kode spart Speicherplatz - ein besonders für kleinere Systeme wichtiger Faktor - und verbessert die Prozessorgeschwindigkeit durch Verringerung der Anzahl von Befehlsworten, die geholt und dekodiert werden müssen. In bezug auf Kodedichte bietet der U 8000 verschiedene Vorteile. Die am häufigsten verwendeten Befehle werden in Einzelwort-Formate kodiert. Zur Ausführung einer bestimmten Aufgabe werden so weniger Befehle benötigt. Auch die klare und regelmäßige Architektur reduziert die Anzahl der erforderlichen Befehle weiter.

Teilweise wird die Kodedichte durch Anwendung eines besonderen Kurzformates für bestimmte Befehle erreicht, die - nach statistischer Analyse - am häufigsten von Assemblern verwendet werden. Ebenso wurde ein Kurz-Offset-Mechanismus vorgesehen, um eine segmentierte 2-Wort-Adresse auf ein einzelnes Wort zu reduzieren; dieses Format kann von Assemblern und Compilern verwendet werden. Die größte Verringerung der Programmgröße und eine Geschwindigkeitserhöhung ergeben sich aus der konsistenten und regelmäßigen Struktur der Architektur sowie dem äußerst leistungsfähigen Befehlsvorrat; Faktoren, welche die für eine Aufgabe erforderliche Anzahl von Befehlen wesentlich reduzieren. In bezug auf bisherige Mikroprozessoren ist die Architektur regelmäßiger, weil ihre Register, Adressierungsarten und Datentypen in geordneter Weise kombiniert werden können. Jedes Allzweckregister kann als Akkumulator, Indexregister oder Basisregister festgelegt werden. Mit wenigen Ausnahmen lassen sich alle grundsätzlichen Adressierungsarten wie auch die verschiedenen Datentypen mit allen Befehlen anwenden.
Allzweckregister müssen nicht so oft verändert werden wie Register, die für einen bestimmten Zweck vorgesehen sind. Es verringert sich die Programmgröße,weil häufiges Auslagern der Registerinhalte in den Speicher nicht erforderlich ist.

1.4.2. Compilereffizienz

Für Anwender von Mikroprozessoren bietet der Übergang von der Assemblersprache auf höhere Programmiersprachen mehr Freiheit von Architektureigenheiten und ermöglicht ein leichteres Programmieren. Die Architektur des U 8000 wurde jedoch nicht einer speziellen höheren Programmiersprache angepaßt, vielmehr wurde diese CPU als Universal-Mikroprozessor entwickelt. (Der direkte Zuschnitt eines Prozessors auf Leistungsfähigkeit in einer Sprache führt oft zu mangelnder Leistung bei anderen Sprachen.)
Für den U 8000 wurde die Unterstützung höherer Programmiersprachen durch Realisierung verschiedener Fähigkeiten zur Minimierung typischer Übersetzungs- und Codeerzeugungsprobleme erreicht. Eine dieser Fähigkeiten ist die Regelmäßigkeit der U 8000-Adressierungsarten und Datentypen. Zugriff zu den Parametern und lokalen Variablen des Prozedur-Stacks wird durch die Adressierungsart indiziert (auch mit verkürztem Offset) sowie die Basis- und Basis-Indiziert-Adressierungsarten unterstützt. Außerdem sind die Inkrementierungs- und Dekrementierungsbefehle eine Hilfe für die Adressier-Arithmetik.
Das Testen von Daten, die logische Auswertung, Initialisierung und der Vergleich von Daten werden ermöglicht durch die Befehle Test, Teste Bedingungskode, Lade Direktwert in Speicher und Vergleiche Direktwert mit Speicher. Da Compiler und Assembler häufig Zeichenketten (Strings) behandeln, ergeben die Befehle Übersetze (Translate), Übersetze und Teste (Translate And Test), Blockvergleich (Block Compare) und Vergleiche Datenfolge (Compare String) insgesamt drastische Geschwindigkeitsverbesserungen gegenüber Softwaresimulationen dieser wichtigen Aufgaben.Außerdem kann jedes Register mit Ausnahme von RØ von den Push- und Pop-Befehlen als Stackpointer verwendet werden.

1.4.3. Betriebssystemunterstützung

Die Interrupt- und Aufgabenwechsel- (task - switching) -fähigkeiten sind wichtige Eigenschaften für die Implementierung von Betriebssystemen. Auch die Leistungsfähigkeit der Speicherverwaltung und die Compilereffizienz sind an dieser Stelle recht bedeutend.
Die Interrupt-Struktur besitzt drei Ebenen: nicht-maskierbar, nicht-vektorisiert und vektorisiert. Tritt ein Interrupt auf, so wird der Programmstatus zusammen mit einem Wort, aus dem sich der Grund für diesen Zustandswechsel identifizieren läßt, auf den Stack gerettet, bevor ein neuer Programmstatus aus einem Spezialbereich des Speichers (der PSA) geladen wird.

Der Programmzustand besteht aus dem Flag-Register mit den Steuerbits und dem Befehlszähler. Der Grund für dieses Ereignis ist in einem Vektor kodiert, der vom Systembus gelesen und auf dem Stack abgelegt wird. Bei einem vektorisierten Interrupt bestimmt der Vektor auch eine Sprungtabellenadresse, die zur Interrupt-Bearbeitungsroutine zeigt. Die Betriebsarten 'System' und 'Normal' verbessern die Betriebssystemorganisation. In der Systembetriebsart sind alle Operationen erlaubt, während im Normal-Modus gewisse Systembefehle verboten sind. Der Systemaufruf-Befehl(System Call, SC) gestattet einen gesteuerten Wechsel der Betriebsart. Unterstützt wird dieses durch die Implementierung von Traps. Bei Traps erfolgt die gleiche Art von Programmstatusrettung wie bei Interrupts. In beiden Fällen wird der System-Stack verwendet. Der Normal-Stack bleibt unverändert. Durch den Mehrfachladebefehl (Load Multiple) können die Registerinhalte effektiv im Speicher oder auf den Stack abgelegt werden. Laufende Programme können Programmstatusveränderungen unter direkter Softwaresteuerung mit dem Befehl "Lade Programmstatus" (Load Program Status) verursachen.

Mit dem nicht teilbaren Test- und Setz-Befehl (Test And Set) läßt sich schließlich erreichen, daß asynchrone kooperierende Prozesse untereinander synchronisiert werden können.

1.4.4. Möglichkeit vieler Arten von Datenstrukturen

Eine Datenstruktur ist die logische Organisation einfacher Datenelemente (Byte, Wort usw.), deren Format und Zugriffsvereinbarungen definiert sind. Übliche Datenstrukturen sind Felder, Listen, Stacks und Strings (Datenfolgen).
Da Datenstrukturen in höheren Programmiersprachen häufig angewendet werden, wird die Prozessorleistung wesentlich erhöht, wenn die CPU für deren leistungsfähige Bearbeitung entsprechende Mechanismen besitzt. Die CPU U 8000 bietet solche Mechanismen.
In vielen Anwendungsfällen ist das Feld die am häufigsten anzutreffende Datenstruktur. Felder werden im U 8000 durch die Adressierungsart "Basis Indiziert" sowie durch segmentierte Adressierung ermöglicht. Die "Basis Indiziert"-Adressierungsart gestattet die Verwendung von Zeigern in einem Feld (d. h. Offsets von der Startadresse des Feldes). Die segmentierte Adressierung erlaubt die Zuordnung eines Feldes zu einem Segment, auf das einfach mit der Segmentnummer Bezug genommen werden kann.
In der kommerziellen und allgemeinen Datenverarbeitung kommen Listen häufiger vor als Felder. Listen werden durch die Adressierungsarten "Indirektes Register" und "Basis-Adresse" ermöglicht. Für kompliziertere Listen ist auch die "Basis Indiziert"-Adressierungsart recht nützlich.
Stacks dienen in allen Anwendungsfällen zur Schachtelung (nesting) oder für Routinen blockstrukturierter Sprachen und für die Interrupt-Behandlung. Stacks werden durch die Push- und Pop-Befehle bedient. Mehrfach-Stacks können aufgrund der Allzweck-Register des U 8000 realisiert werden. Außerdem dienen zwei Hardware-Stackpointer zur Zuordnung separater Stacks zu den Betriebsarten "System" und "Normal", wodurch die bereits erwähnte Trennung von System- und Normalbetrieb erleichtert wird.
Byte- und Wort-String-Bearbeitung werden durch die Befehle "Übersetzen" (Translate) und "Übersetzen und Testen" (Translate And Test) unterstützt. Für Strings mit Dezimalzahlen kann der Befehl "Dezimal Korrektur" (Decimal Adjust) bei der Ausführung von Dezimal-Arithmetik auf BCD-Datenfolgen angewendet werden. Die gepackte Darstellung erlaubt zwei BCD-Ziffern pro Byte. Die Digit-Rotationsbefehle manipulieren 4-Bit-Daten.

1.4.5. Zwei CPU-Versionen U 8001 und U 8002

Die CPU U 8000 wird in zwei Versionen angeboten, und zwar als segmentierte CPU U 8001 mit 48 Pins und als nichtsegmentierte CPU U 8002 mit 40 Pins. Der Hauptunterschied zwischen beiden liegt im Adressierungsbereich. Der U 8001 kann 8 MBytes des Speichers direkt adressieren, während der U 8002 64 KBytes direkt adressiert. Der U 8001 verfügt über eine nicht segmentierte Betriebsart, welche es erlaubt, für den U 8002 geschriebene Programme auszuführen. Nicht alle Anwendungsfälle erfordern den großen Adreßraum des U 8001, deshalb wird hierfür der U 8002 empfohlen. Außerdem können viele Mehrfachprozessorsysteme mit einem U 8001 und mehreren U 8002 realisiert werden anstelle der ausschließlichen Verwendung des U 8001.
Da der Assembler den Kode für beide CPU-Versionen erzeugen kann, gibt es keine Kompatibilitätsprobleme bei der Softwareerstellung.

1.5. Befehlserweiterungsfähigkeit

Die U 8000-Architektur verfügt über einen Mechanismus zur Erweiterung des Grundbefehlssatzes durch Anwendung externer Einheiten. Zur Realisierung dieser Möglichkeit wurden spezielle Operationskodes vorgesehen. Trifft die CPU in ihrem Befehlsstrom auf einen Befehl, der keinen solchen Operationskodes enthält, so führt sie die angegebene Adressenberechnung und die Datenbearbeitung durch. Ermittelt die CPU dagegen einen sogenannten erweiterten Befehl, so geht sie davon aus, daß er von einer externen Einheit ausgeführt wird. In diesen erweiterten Befehlen sind Felder reserviert, die durch externe Einheiten (Erweiterungsprozessoren, Extended Processing Unit - EPU) als Operationskode interpretiert werden können. So kann mit Hilfe geeigneter EPU's der Befehlssatz des U 8000 auf spezielle Befehle erweitert werden.
Im allgemeinen dient eine EPU zur Entlastung der CPU von komplizierten und zeitaufwendigen Aufgaben. Typische Aufgaben für spezielle EPU's sind Gleitkommaarithmetik, Datenbasissuche und Kontroll- bzw. Überwachungsvorgänge, Netzwerk-Interfaces, Unterstützung von Grafikoperationen ... . Eine vollständige Aufzählung würde die meisten Gebiete der Rechentechnik berühren.

1.6. Zusammenfassung

Die Architektur des Mikroprozessors U 8000 befindet sich auf einer mit einem Minicomputer vergleichbaren Ebene. Solche Eigenschaften wie große Adreßräume, Mehrfachspeicherräume, segmentierte Adressen sowie Multi-Processing-Möglichkeiten übertreffen die Fähigkeiten eines herkömmlichen Mikroprozessors.
Die Vorzüge dieser Architektur wie Kodedichte, Compilereffizienz sowie Betriebssystemunterstützung erhöhen die Vielseitigkeit und das Anwendungsspektrum der CPU beträchtlich. Die CPU-Merkmale, welche ein externes Speicherverwaltungssystem gestatten, erhöhen auch die Anwendbarkeit der CPU auf dem Gebiet der Großsysteme.

Kapitel 2

2. Architektur

2.1. Einleitung

Dieses Kapitel gibt eine Übersicht zur CPU-U 8000-Architektur mit einer Beschreibung des Grundaufbaus, der Betriebsarten und des Befehlssatzes. Falls erforderlich, werden auch die Unterschiede zwischen den beiden Versionen (U 8002 nichtsegmentiert und U 8001 segmentiert) hervorgehoben. Die meisten der hier behandelten Themen werden in späteren Kapiteln dieses Handbuches noch ausführlicher erläutert.

2.2. Allgemeines

Bild 2.1 zeigt das Blockschaltbild mit den folgenden Hauptelementen der CPU U 8000:

- ein interner 16-Bit-Datenbus zum Transport von Adressen und Daten innerhalb der CPU
- ein Bus-Interface zur Steuerung der Wechselbeziehung zwischen CPU und Außenwelt
- ein Satz von 16 Mehrzweck-Registern zur Aufnahme von Adressen und Daten
- vier Sonderregister zur Steuerung der CPU-Operation
- eine ALU (arithmetisch-logische Einheit) zur Datenmanipulation und zum Erzeugen der Adressen
- eine Befehlsausführungssteuerung, welche die U 8000-Befehle holt und ausführt
- eine Ausnahme-Behandlungssteuerung zur Verarbeitung von Interrupts und Traps
- eine Refresh-Steuerung, welche Speicherauffrischzyklen organisiert

Jedes dieser Elemente wird in den folgenden Abschnitten beschrieben. Sie sind sowohl Bestandteil der CPU U 8001 als auch der Version U 8002. Der Unterschied zwischen beiden besteht hauptsächlich in der Anzahl der erzeugten Adressen. Die CPU U 8002 erzeugt stets eine lineare 16-Bit-Adresse, während vom U 8001 eine segmentierte 23-Bit-Adresse (d. h. eine aus einer 7-Bit-Segmentnummer und einem 16-Bit Offset zusammengesetzte Adresse) erzeugt wird.

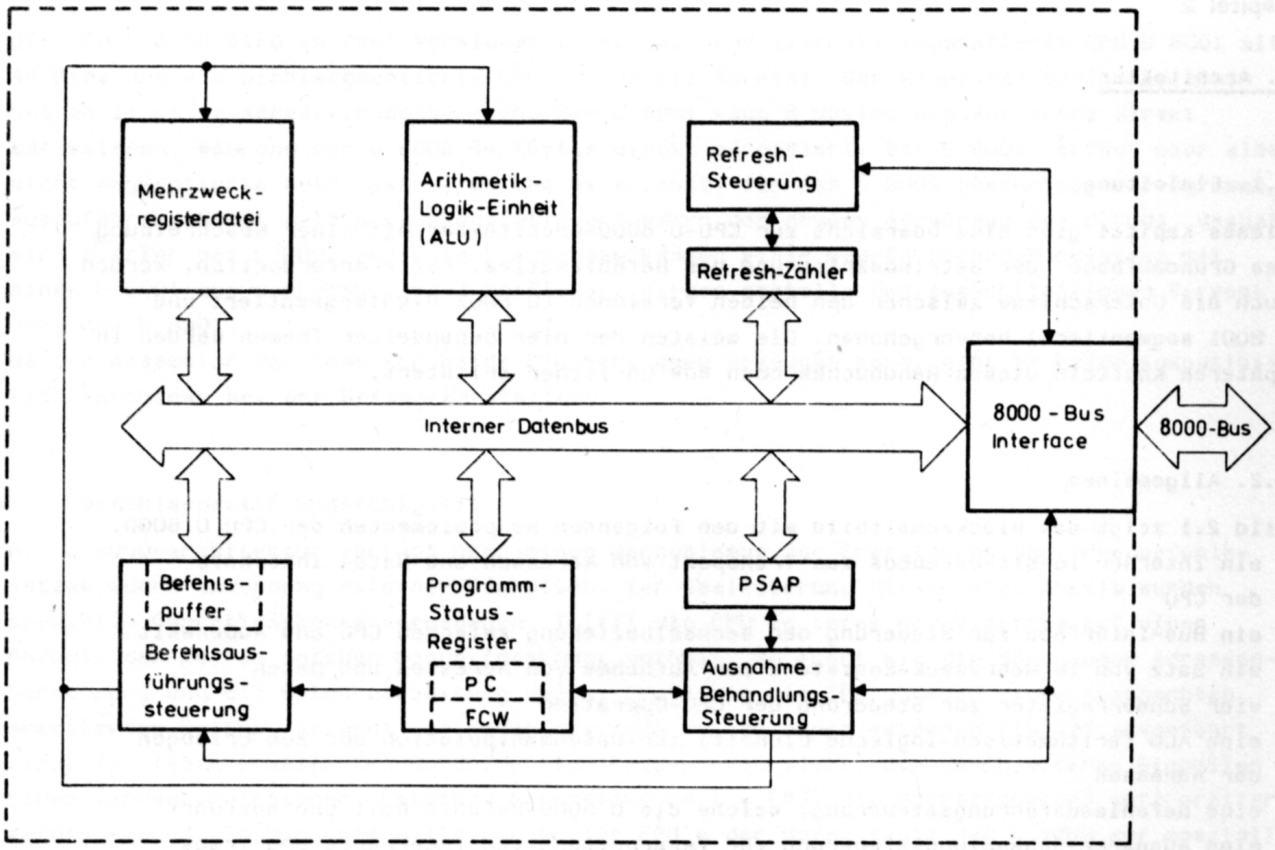


Bild 2.1 CPU U 8000 Funktionsblockschaltbild

Im Bild 2.2 ist eine U 8000-Systemkonfiguration dargestellt. An die CPU U 8000 ist ohne großen Aufwand eine Reihe von Ergänzungsschaltkreisen anpaßbar.

Ein leistungsfähiges Speicherverwaltungssystem kann mit einem U 8001 und einer oder mehreren Speicherverwaltungseinheiten (MMU) U 8010 aufgebaut werden. Zwischen CPU und MMU gibt es keine Interface-Probleme, da beide für den 8000-Bus konzipiert sind.

Weiterhin ist es möglich, die peripheren Einheiten des 8-Bit-Systems UA 880 wie PIO, SIO, CTC und DMA zu verwenden. Das Interface zwischen diesen Peripheriebausteinen und dem 8000-Bus kann mit wenigen Standard-SSI und -MSI-Schaltkreisen realisiert werden. Zur intelligenten Steuerung von Peripheriegeräten bieten sich die Einchipmikrorechner (EMR) U 881 bis U 884 an.

Als Erweiterungsprozessoren (EPU) sind sowohl monolithische Spezialprozessoren als auch Lösungen basierend auf einem Standardmikroprozessor denkbar.

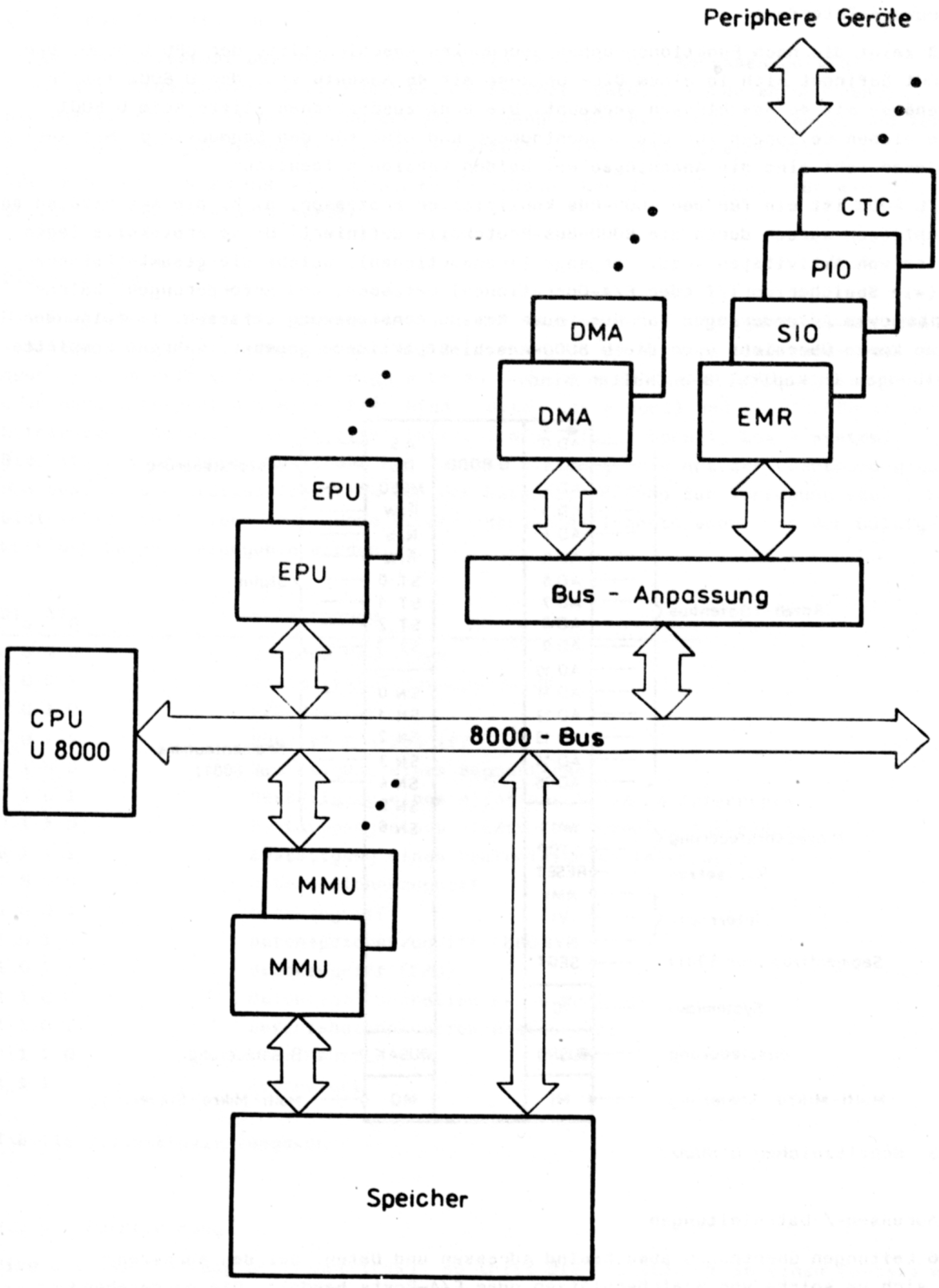


Bild 2.2 Systemkonfiguration

2.3. Hardware-Interface

Bild 2.3 zeigt die nach Funktionsgruppen geordneten Anschlußstifte der CPU U 8000. Der Typ U 8001 befindet sich in einem DIL- Gehäuse mit 48 Anschlüssen, der U 8002 ist in einem Gehäuse mit 40 Anschlüssen verkappt. Die acht zusätzlichen Stifte beim U 8001 sind die sieben Leitungen für die Segmentnummer und eine für den Segmenttrap. Mit Ausnahme dieser acht sind die Anschlüsse bei beiden Versionen identisch.

Die CPU U 8000 ist ein für den 8000-Bus konzipierter Prozessor, d. h. die Aktivitäten an den Anschlüssen werden durch die 8000-Bus-Protokolle definiert. Diese Protokolle legen zwei Arten von Aktivitäten fest: Vorgänge (Transaktionen), welche die gesamte Datenbewegung (wie Speicherzugriff oder E/A-Operationen) umfassen, und Anforderungen, welche Interrupts sowie Anforderungen für Bus- oder Ressourcensteuerung umfassen. Im folgenden wird eine kurze Übersicht über die U 8000-Anschlußfunktionen gegeben, während komplette Beschreibungen in Kapitel 8 enthalten sind.

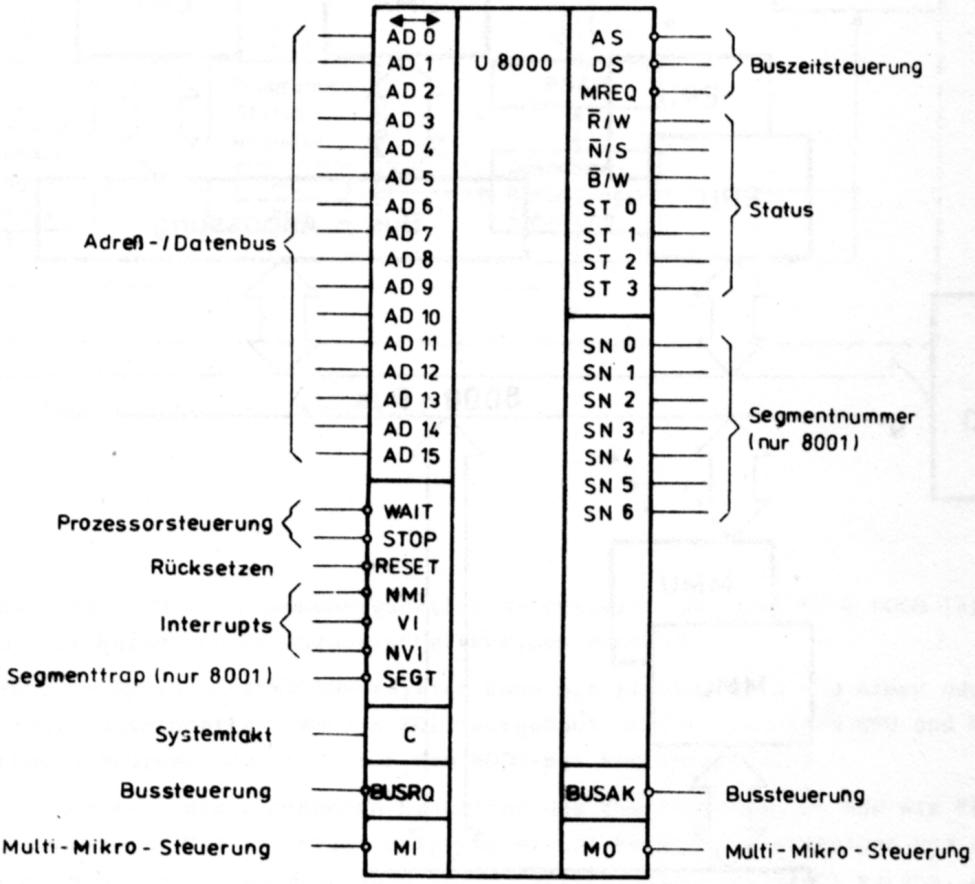


Bild 2.3 Schaltzeichen U 8000

2.3.1. Adressen-/ Datenleitungen

Diese 16 Leitungen übertragen abwechselnd Adressen und Daten. Bei den Adressen kann es sich um solche von Speicherplätzen oder E/A-Ports handeln. Die nachstehend beschriebenen Buszeitsteuerungs-Signalleitungen zeigen an, welche Art von Informationen die Adressen-/Datenleitungen tragen.

2.3.2. Segmentnummer (nur U 8001)

Diese sieben Leitungen kodieren die Nummer des aktuellen Speichersegmentes. Die Segmentsignale werden vor den Adressenoffsetsignalen gültig und unterstützen somit die Adressenübersetzung durch das Speicherverwaltungssystem.

2.3.3. Bus-Zeitsteuerung

Diese Gruppe besteht aus den Leitungen Adress Strobe ($\overline{AS}$, Adressen gültig), Data Strobe ($\overline{DS}$, Daten gültig) und Memory Request ($\overline{MREQ}$, Speicheranforderung). Sie dienen zur zeitlichen Steuerung des Vorganges auf dem Adreß-/Datenbus und zur Signalisierung, ob dieser Multiplexbus Adressen oder Daten bereit hält.

Das $\overline{MREQ}$-Signal kann zur zeitlichen Beeinflussung der Steuersignale für ein Speichersystem verwendet werden.

2.3.4. Status

Die Funktion dieser Leitungen besteht in der Anzeige der Übertragungsart auf dem Bus (ST_0 - ST_3), weiterhin in der Signalisierung,ob es sich um eine Lese- oder Schreiboperation handelt (R/$\overline{W}$, wobei High = Lesen und Low = Schreiben ist); ob ein Byte oder Wort transportiert wird (B/$\overline{W}$, High = Byte, Low = Wort) und ob die CPU in der Betriebsart "Normal" oder "System" läuft. (N/$\overline{S}$, High = Normal, Low = System). Die ST_0-ST_3-Leitungen kodieren, wie Tabelle 2.1 zeigt die Art der Übertragung auf dem Bus. Da die Statusinformation vor der Datenübertragung zur Verfügung steht, können bidirektionale Treiber und andere externe Hardware-Elemente schon vor der Gültigkeit der Daten freigegeben werden.

ST_3-ST_0	Erläuterung
0 0 0 0	Interne Operation
0 0 0 1	Speicher-Refresh-Operation
0 0 1 0	E/A-Zugriff
0 0 1 1	Spezial-E/A-Zugriff
0 1 0 0	Bestätigung eines Segmenttrap
0 1 0 1	Bestätigung eines nicht-maskierbaren Interrupts
0 1 1 0	Bestätigung eines nichtvektorisierten Interrupts
0 1 1 1	Bestätigung eines vektorisierten Interrupts
1 0 0 0	Datenspeicherzugriff
1 0 0 1	Stackzugriff
1 0 1 0	Datenspeicherzugriff (EPU)
1 0 1 1	Stackzugriff (EPU)
1 1 0 0	Befehlsholeoperation n-tes Wort
1 1 0 1	Befehlsholeoperation erstes Wort
1 1 1 0	EPA-Transfer
1 1 1 1	Reserviert

Tabelle 2.1 Statusleitungskodes

2.3.5. CPU-Steuerung

Wird die $\overline{WAIT}$-Leitung aktiviert (Low), so tritt die CPU solange in einen Zustand ein, in dem sie $\overline{WAIT}$-Zyklen ausführt, bis die $\overline{WAIT}$-Leitung wieder inaktiv wird, d. h. sie verlängert ihre Busübertragung um zusätzliche Taktzyklen. Dieser Eingang wird von Speichern oder peripheren Einheiten aktiviert, die langsamer als die CPU arbeiten.

Mit der $\overline{STOP}$-Leitung kann die interne Operation der CPU, nachdem das ersten Befehlswort geholt wurde, angehalten werden. Dieses Signal ist sehr nützlich für die Realisierung des Einzelschrittbetriebes zum Zwecke der Fehlersuche und wird auch verwendet zum zeitweiligen Anhalten der CPU durch eine EPU.

2.3.6. Bussteuerung

Diese Leitungen sind die Voraussetzung für die Übergabe der Steuerung des Systembusses, die normalerweise von der CPU übernommen wird, an andere Einheiten wie z. B. DMA-Steuereinheiten. Die externe Einheit, die die Steuerung des Busses anfordert, aktiviert die $\overline{\text{BUSREQ}}$-Leitung. Nachdem die CPU den Adreß-/Datenbus, die Status-, Steuer- und Buszeitsteuerleitungen in den hochohmigen Zustand gebracht hat, bestätigt sie die Anerkennung der Busanforderung durch das $\overline{\text{BUSACK}}$-Signal. Bei mehreren externen Einheiten, die die Bussteuerung übernehmen können, erlaubt der 8000-Bus die Bildung einer Prioritätskette.

2.3.7. Interrupts

Es sind drei Interrupt-Eingänge vorhanden, und zwar nicht-maskierbare Interrupts ($\overline{\text{NMI}}$), vektorisierte Interrupts ($\overline{\text{VI}}$) und nicht-vektorisierte Interrupts ($\overline{\text{NVI}}$). Diese gestatten es externen Einheiten, die Ausführung des laufenden Programmes der CPU aufzuschieben und mit der Ausführung eines für den jeweiligen Unterbrechungsfall vorgesehenen Programms (interrupt service routine) zu beginnen.

2.3.8. Segment-Trap-Anforderung (nur U 8001)

Dieser CPU-Eingang wird von einem externen Speicherverwaltungssystem zur Mitteilung eines versuchten unerlaubten Speicherzugriffes aktiviert.

2.3.9. Multi-Mikro-Steuerung

Die Multi-Mikro-In ($\overline{\text{MI}}$) und Multi-Mikro-Out ($\overline{\text{MO}}$)-Leitungen dienen in Verbindung mit Befehlen wie MSET und MREQ zur Koordinierung von Systemen mit mehreren CPU's. Sie gestatten die ausschließliche Verwendung eines gemeinsamen Systemelements (Ressource) in einem Multiprozessorsystem durch eine CPU.

2.3.10. Systemeingänge

Zu dieser Gruppe sind der Eingang für den Einphasentakt, der Rücksetzeingang sowie die Anschlüsse für Masse und +5 V-Versorgungsspannung zu rechnen.
Die Rücksetzfunktion wird im Kapitel 6 beschrieben.

2.4. Zeitsteuerung

Bild 2.4 zeigt die drei grundsätzlichen Zeitperioden des U 8000: einen Taktzyklus, einen Buszugriff und einen Maschinenzyklus. Ein Taktzyklus ist eine Periode des CPU-Taktes, der mit einer steigenden Flanke beginnt.
Ein Buszugriff umfaßt einen einzelnen Datentransport auf dem CPU-Bus und dauert mindestens drei Taktzyklen; er beginnt mit der fallenden Flanke von $\overline{\text{AS}}$ und endet mit der steigenden Flanke von $\overline{\text{DS}}$. Ein Maschinenzyklus umfaßt eine CPU-Grundoperation und beginnt stets mit einem Buszugriff. Ein Maschinenzyklus kann über eine nicht begrenzte Anzahl von Taktzyklen verlängert werden. Anhang A enthält eine komplette Beschreibung der U 8000-Zeitabläufe.

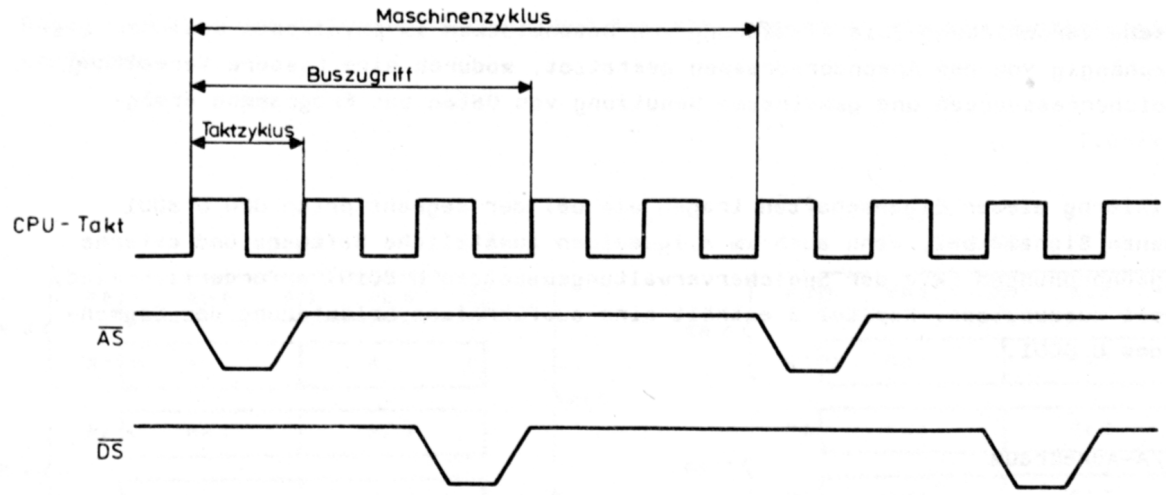


Bild 2.4 Zeitperioden

2.5. Adreßräume

Der U 8000 besitzt zwei verschiedene Hauptadreßräume:

- Speicheradreßraum
  Dieser besteht aus den Adressen aller Speicherplätze im Hauptspeicher des Computersystems.
- E/A-Adreßraum
  Dieser besteht aus den Adressen aller E/A-Ports, mit denen der Zugriff auf periphere Geräte erfolgt.

Weitere Angaben zu Adreßräumen sind in Kapitel 3 enthalten.

2.5.1. Speicheradreßraum

Der Speicheradreßraum läßt sich weiter unterteilen in den Programmspeicheradreßraum, den Datenspeicheradreßraum sowie den Stackadreßraum sowohl für die Normal- als auch für die System-Betriebsart.
Der jeweilige adressierte Raum wird durch die externe Schaltungsanordnung aus dem an den Ausgangsstatusstiften der CPU (ST_0-ST_3) erscheinenden Kode und dem Status des Normal-/Systemsignals (N/$\overline{S}$-Anschluß) bestimmt. Datenspeicherzugriff, Stackzugriff und Programmspeicherzugriff entsprechen jeweils einem unterschiedlichen Statuskode an den ST_0-ST_3-Ausgängen, so daß für jede der zwei Betriebsarten drei Adreßräume unterschieden werden können, womit insgesamt sechs Speicheradreßräume zur Verfügung stehen.

Jeder der sechs Adreßräume besitzt einen Umfang, der so groß ist wie die Adressierfähigkeit des Prozessors. Beim nichtsegmentierten Typ U 8002 kann jeder Adreßraum bis zu 64 KByte besitzen, was eine potentielle Gesamtsystemkapazität von 384 KByte direkt adressierbaren Speichers ergibt. Andererseits besitzt der segmentierte Typ U 8001 aufgrund der segmentierten 23-Bit-Adressen bis zu 48 MBytes direkt adressierbaren Speichers.

Die Segmentierung ist ein Mittel zur Aufteilung des Speichers in Segmente variabler Größe, so daß sich eine Anzahl nützlicher Funktionen ergibt wie z. B. folgende:

- Schutzmechanismen (Der Anwender wird beispielsweise am Zugriff auf Daten, die anderen gehören, gehindert. Ebenso könnte die versuchte Änderung von Nur-Lese-Daten oder das Überlaufen eines Stacks unterbunden werden.)

- Virtueller Speicher (Dem Anwender wird das Schreiben von Funktionsprogrammen unter der Annahme gestattet, daß das System mehr Speichermöglichkeit enthält als tatsächlich vorhanden ist.)

- Dynamische Verschiebung (Die Plazierung von Datenblöcken im physischen Speicher ist unabhängig von den Anwenderadressen gestattet, wodurch eine bessere Verwaltung der Speicherressourcen und gemeinsame Benutzung von Daten und Programmen ermöglicht wird.)

Zur Realisierng dieser Eigenschaften tragen die bei der segmentierten CPU U 8001 vorgesehenen Signale bei, wenn auch im allgemeinen zusätzliche Software und externe Schaltungsanordnungen (wie der Speicherverwaltungsbaustein U 8010) erforderlich sind, um sie voll auszunutzen. Kapitel 3 enthält eine ausführliche Erläuterung der Segmentierung des U 8001.

2.5.2. E/A-Adreßraum

Sowohl im U 8001 als auch beim U 8002 werden E/A-Adressen als 16-Bit-Worte dargestellt. Es sind zwei E/A-Adreßräume vorhanden: Standard-E/A und Spezial-E/A. Beide sind getrennt vom Speicheradreßraum. Der Zugriff zu jedem E/A-Raum geschieht durch einen getrennten Satz von E/A-Befehlen, die nur ausgeführt werden können, wenn sich die CPU in der System-Betriebsart befindet.
Standard-E/A-Befehle übertragen Daten zwischen der CPU und peripheren Einheiten; Spezial-E/A-Befehle übertragen Daten zu oder von externen CPU-Unterstützungsschaltkreisen wie z. B. dem Speicherverwaltungs-Baustein U 8010. Ob der Zugriff zum Standard- oder Spezial-E/A-Raum erfolgt, wird durch die Statusleitungen (ST_0-ST_3) angezeigt.

2.6. Mehrzweck-Register

Die CPU U 8000 enthält 16 Mehrzweck-Register mit einer Breite von je 16 Bit. Jedes Mehrzweck-Register kann für jeden Befehlsoperanden verwendet werden. (Es gibt geringe Ausnahmen, die zu Beginn von Kapitel 5 beschrieben werden.)

In Bild 2.5 ist dieser Mehrzweck-Registersatz dargestellt.
Es werden Datenformate vom Byte bis zum Doppellangwort (64 Bit) erlaubt. Die Wortregister werden in Assemblersprachanweisungen mit R0 bis R15 bezeichnet. Die ersten acht Wortregister lassen sich einzeln teilen und können wie sechzehn Byteregister RH0 bis RL7 betrachtet werden.
Register für größere Operanden lassen sich durch Zusammenfügung von Wortregistern bilden. So können bis zu acht Langwortregister (32 Bit) gebildet werden, die auch als Doppelregister bezeichnet werden und mit RR0 bis RR14 gekennzeichnet werden. Es ist auch möglich, bis zu vier Doppellangwortregister (64 Bit) zu definieren, welche RQ0 bis RQ12 genannt werden. Diese werden von Operationen wie MULT, DIV und EXTS verwendet.

Der segmentierte Typ U 8001 verwendet bei beiden Betriebsarten einen Langwort-Stackpointer (R14'/R15' bzw. R14/R15), während der nicht-segmentierte Typ U 8002 für jede Betriebsart nur ein Wort verwendet (R15' oder R15).
Der Systemstackpointer dient bei einem Interrupt oder einem Trap zum Retten der Statusinformation und zur Unterstützung von Unterprogrammaufrufen im Systemmodus. Der Normalstackpointer dient für Aufrufe von Unterprogrammen im "Normal"-Modus. In der Betriebsart "Normal" wird nur mit dem Normalstackpointer gearbeitet.

Dagegen ist in der Systembetriebsart der Normalstackpointer wie ein Spezialregister verfügbar.

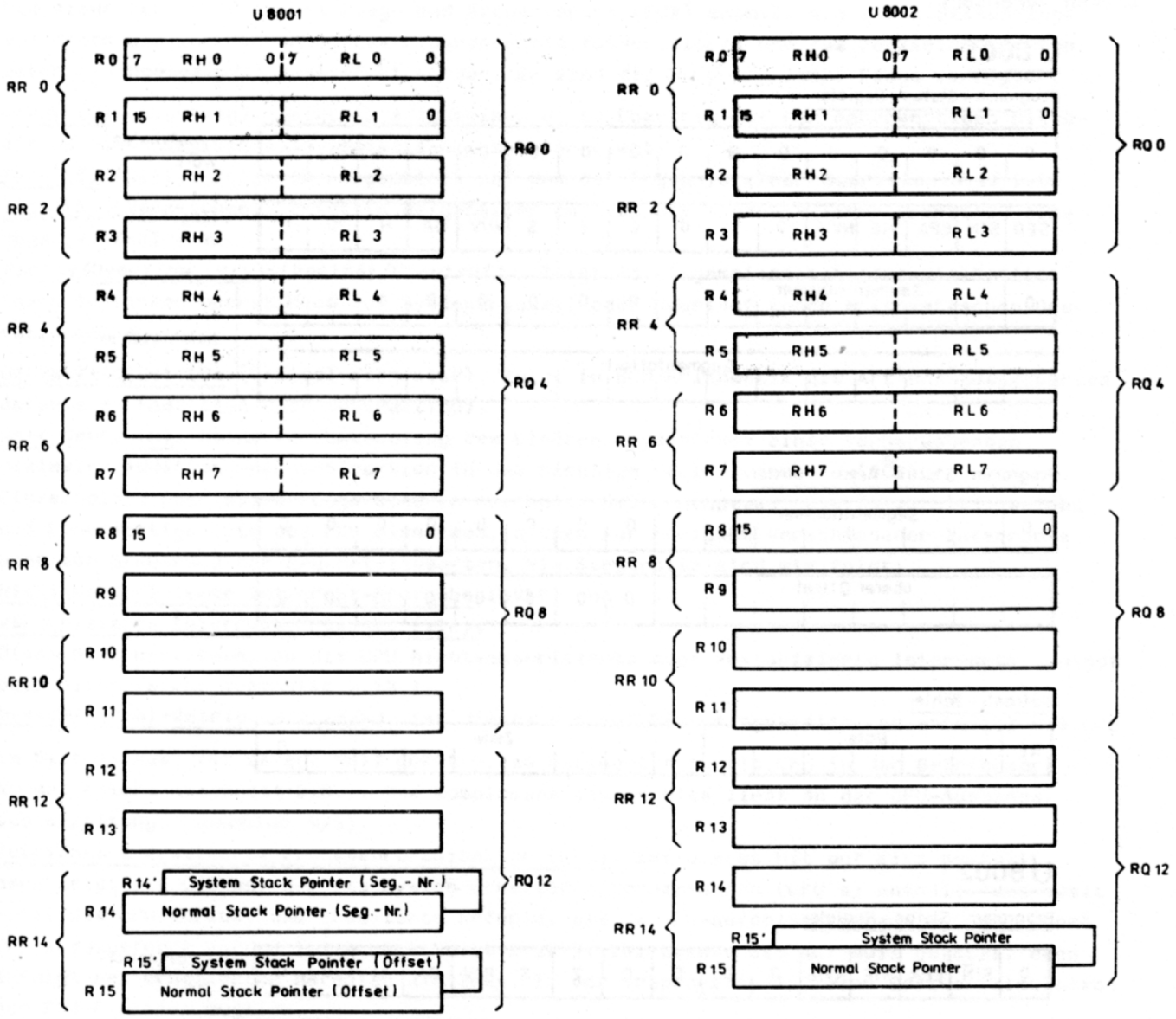


Bild 2.5 a) U 8001-Mehrzweckregisterdatei b) U 8002-Mehrzweckregisterdatei

2.7. Spezialregister

Zusätzlich zu den Mehrzweckregistern sind Spezialregister vorhanden. Sie umfassen die Programm-Status-Register, den Programm Status Area Pointer sowie den Refresh-Zähler. Sie sind für beide CPU-Versionen in Bild 2.6 dargestellt. Jedes Register kann durch im Systemmodus laufende Software manipuliert werden; einige werden automatisch durch bestimmte Operationen verändert:

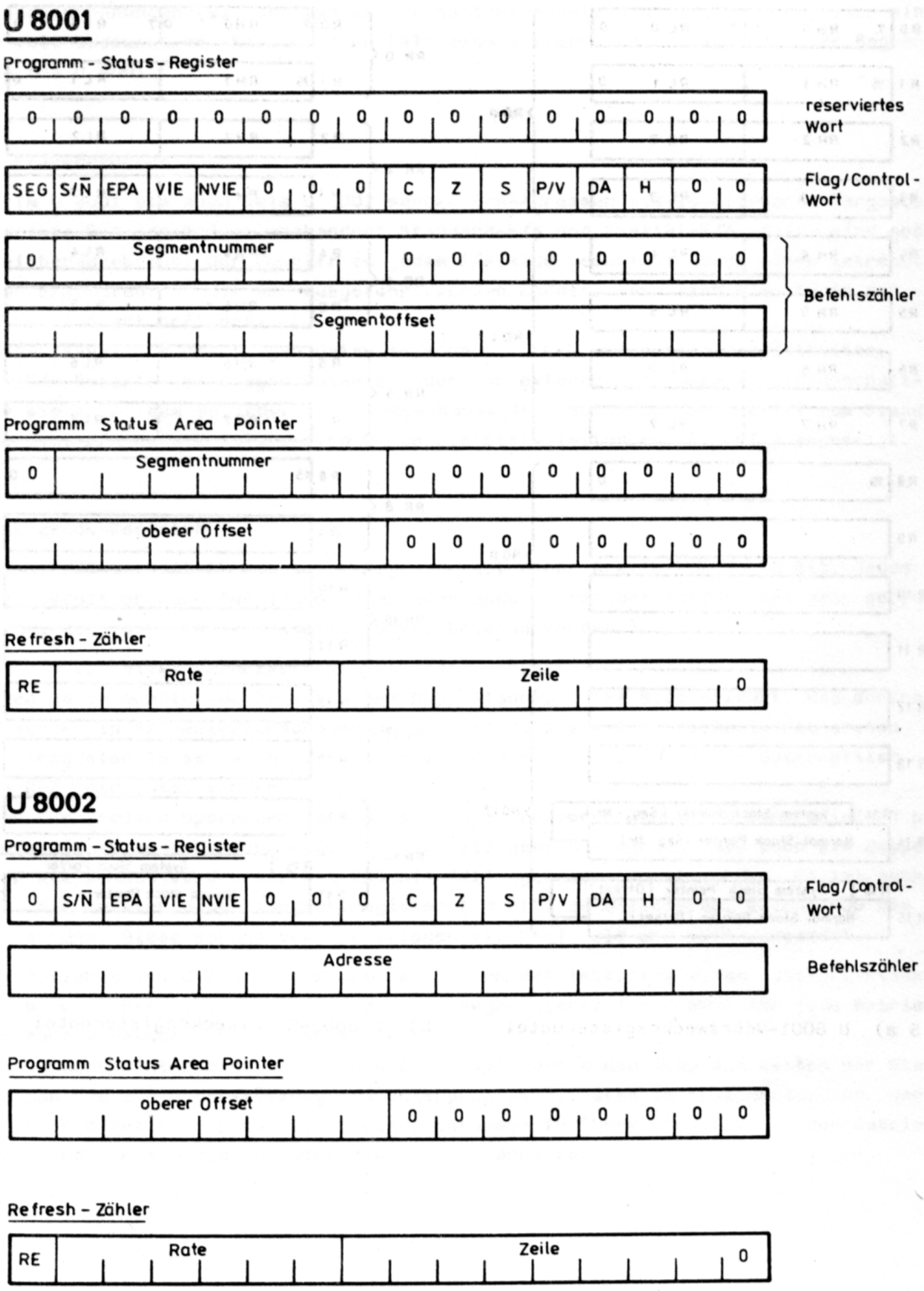


Bild 2.6 CPU-Spezialregister

2.7.1. Programm-Status-Register

Diese Registergruppe besteht aus dem Flag- und Steuerwortregister (Flag Control Word, FCW) sowie dem Befehlszähler (Programm Counter, PC). Sie bestimmen den Bearbeitungsablauf eines Programmes.
In dem nichtsegmentierten U 8002 bestehen die Programmstatusregister aus zwei Worten: einem für das FCW und einem für den PC. In dem segmentierten U 8001 sind es vier Worte: ein reserviertes Wort, ein Wort für FCW und zwei Worte für den segmentierten PC. Das niederwertige Byte des Flag- und Steuerwortes (FCW) enthält die sechs Statusflags, von denen die zur Steuerung des Programmschleifendurchlaufs und zum Verzweigen verwendeten Bedingungskodes abgeleitet werden. Es sind die folgenden sechs Flags verfügbar:

Carry (C) (Übertrag). Zeigt im allgemeinen einen Übertrag aus der höherwertigen Bitposition eines Registers an, das als Akkumulator dient.
Zero (Z) (Null). Zeigt im allgemeinen an, daß das Ergebnis einer Operation Null ist.
Sign (S) (Vorzeichen). Zeigt im allgemeinen an, daß das Ergebnis einer Operation eine negative Zahl ist.
Parity/Overflow (P/V) (Parität/Überlauf). Zeigt im allgemeinen entweder die Parität (nach logischen Operationen auf Byteoperanden) oder Überlauf (nach arithmetischen Operationen) an.

Decimal-Adjust (D) (Dezimal-Korrektur). Zeigt in BCD-Arithmetik die Art des ausgeführten Befehls an (Addition oder Subtraktion).
Half Carry (H). Dient zur Umwandlung des binären Ergebnisses einer vorhergehenden dezimalen Addition oder Subtraktion in das richtige Dezimalergebnis (BCD).
Einzelheiten über diese Flags sind in Abschnitt 6.3. enthalten. Die Steuerbits, welche das höherwertige Byte des FCW einnehmen, dienen zur Freigabe verschiedener Interrupts oder zur Steuerung der CPU-Betriebsarten. Die Steuerbits sind wie folgt:
Nicht-vektorisierte Interrupt-Freigabe (NVIE) und
vektorisierte Interrupt-Freigabe (VIE).
Diese Bits bestimmen, ob die CPU nichtvektorisierte oder vektorisierte Interrupts annimmt oder nicht (siehe Abschnitt 2.13.).
System/Normal-Betriebsart (S/$\overline{N}$). Ist dieses Bit auf Eins eingestellt, so arbeitet die CPU im Systemmodus; ist es auf Null gesetzt, so befindet sich die CPU in der Betriebsart Normal (siehe Abschnitt 2.8.). Das Komplement dieses Bits liegt an der CPU-Ausgangsstatusleitung (Anschluß N/$\overline{S}$).
Betriebsart Erweiterte Prozessorarchitektur (EPA). Ist dieses Bit auf Eins gestellt, dann zeigt die CPU an, daß das System Erweiterungsprozessoren (EPU's) enthält, und somit setzt die CPU voraus, daß erweiterte Befehle, die im CPU-Befehlsstrom auftauchen, durch EPU's ausgeführt werden (siehe Abschnitt 2.12.). Ist dieses Bit auf Null gesetzt, dann erfolgt bei erweiterten Befehlen ein Trap. In der Traproutine ist eine Softwaresimulation der EPU-Funktion möglich.
Segmentierungsart (SEG). Dieses Bit ist nur im U 8001 vorhanden; im nichtsegmentierten U 8002 ist es stets gelöscht. Wird es auf Eins gesetzt, dann arbeitet die CPU im segmentierten Modus, dagegen bei Null im nichtsegmentierten (siehe Abschnitt 2.8.).

2.7.2. Program Status Area Pointer (PSAP)

Der PSAP (Programmstatusbereichszeiger) zeigt auf ein Feld von Programmstatuswerten (FCW und PC) im Hauptspeicher, welches Program Status Area (PSA, Programmstatusbereich) genannt wird. Aus diesem Bereich werden neue Werte für die Programmstatusregister geholt, wenn ein Interrupt oder ein Trap auftritt. Wie in Bild 2.6 dargestellt, umfaßt der PSAP entweder ein Wort (nichtsegmentierter U 8002) oder zwei Worte (segmentierter U 8001); wobei für jede Konfiguration das untere Byte des Zeigers Null sein muß. Einzelheiten zum Programmstatusbereich und seinem Aufbau sind im Kapitel 6 zu finden.

2.7.3. Refresh-Zähler

Die CPU enthält einen programmierbaren Zähler, der zum automatischen Auffrischen des dynamischen Speichers verwendet werden kann. Das Refresh-Zählerregister besteht aus einem 9-Bit-Zeilenzähler, einem 6-Bit-Register, in dem die Häufigkeit des Auffrischens (Refreshrate) festgelegt wird, sowie einem Freigabebit (Bild 2.6). Einzelheiten zum Refresh-Mechanismus sind im Kapitel 7 dargestellt.

2.8. Befehlsausführung

Bei gewöhnlichen Abläufen verwendet die CPU U 8000 die meiste Zeit für das Holen der Befehle aus dem Speicher und zu deren Ausführung. Dieser Vorgang wird als Arbeitszustand oder "Running"-Status der CPU bezeichnet. Außerdem hat die CPU noch zwei andere Zustände, in die sie eintreten kann.

Stop/Refresh-Zustand. Der Stopzustand und der Refresh-Zustand sind in Wirklichkeit identisch, werden aber auf zwei verschiedene Arten eingenommen, und zwar entweder automatisch für eine periodische Speicherauffrischung, oder wenn die $\overline{\text{STOP}}$-Leitung aktiviert wird.
In diesem Zustand wird die Programmausführung zeitweilig aufgeschoben, und die CPU sorgt für Speicher-Refresh unter Verwendung des Refresh-Zählers. Kapitel 7 enthält nähere Einzelheiten hierzu.

Bus-Abschalt-Zustand. Diesen Status nimmt die CPU ein, wenn eine DMA-Einheit oder ein anderer Busanforderer den Bus übernimmt. Die Programmausführung wird aufgeschoben, die CPU schaltet sich selbst vom Bus ab. Weitere Angaben hierzu sind in Kapitel 9 enthalten.
Befindet sich die CPU im "Running"-Status oder Arbeitszustand, so kann sie entweder Interrupts behandeln oder Befehle ausführen. Führt sie Befehle aus, so kann sich der U 8000 in der Betriebsart System oder Normal befinden. Im Systemmodus können privilegierte Befehle (wie z. B. solche, die E/A bewirken) ausgeführt werden, was bei Normal-Betrieb nicht möglich ist.
Diese Zweiteilung gestattet die Schaffung von Betriebssystem-Software, welche CPU-Ressourcen steuert und gegen Anwenderprogrammeinwirkungen schützt.
Außerdem befindet sich die CPU entweder in der segmentierten oder nichtsegmentierten Betriebsart. Bei Segmentierung, die nur beim Typ U 8001 gegeben ist, verwendet das Programm segmentierte 23-Bit-Adressen zum Speicherzugriff; beim unsegmentierten Modus, der bei beiden CPU's möglich ist, verwendet das Programm nichtsegmentierte 16-Bit-Adressen zum Speicherzugriff. Während der Befehlsausführung wird die CPU-Betriebsart durch Bits im FCW (Abschnitt 2.7.1.) gesteuert. Bei der Behandlung von Interrupts befindet sich die CPU stets in der System-Betriebsart und, im Falle des U 8001, in segmentierter Betriebsweise.

2.9 Befehle

Der Befehlssatz des U 8000 enthält über 400 verschiedene Befehle, die durch Kombination der 110 bestimmten Befehlstypen (Operationskodes oder Opcodes) mit den verschiedenen Datentypen und Adressierungsarten gebildet werden. Der vollständige Satz ist in folgende Gruppen unterteilt:

Lade- und Austauschbefehle für Operationen von Register zu Register und von Register zum Speicher einschließlich Stackverwaltung.

Arithmetische Befehle für arithmetische Operationen, einschließlich Multiplizieren und Dividieren, an Daten entweder in Registern oder im Speicher. Ebenso sind Vergleichs-, Inkrementierungs- und Dekrementierungs-Funktionen darin enthalten.

Logische Befehle für Boolesche Operationen an Daten in den Registern oder im Speicher.

Programmsteuerbefehle für Programmverzweigung (bedingt oder unbedingt), Aufrufe (calls) und Rückkehr (returns).

Bitmanipulation zum Setzen, Rücksetzen und Testen einzelner Bits von Bytes oder Worten in den Registern oder im Speicher.

Rotations- und Verschiebebefehle für Bytes oder Worte in Registern.

Blocktransfer- und Stringbehandlungsbefehle zur automatischen Speicher-Speicher-Übertragung von Datenblocks oder Datenfolgen einschließlich Vergleichs- und Übersetzungsfunktionen.

Eingabe/Ausgabebefehle zum Datentransfer zwischen E/A-Ports und dem Speicher oder Registern.

Erweiterte Befehle für Operationen, an denen Erweiterungsprozessoren (EPU's) beteiligt sind.

CPU-Steuerbefehle für den Zugriff auf Spezialregister, zur Steuerung des Betriebszustandes der CPU, zur Synchronisierung der Arbeit von Mehrfachprozessorkonfigurationen, zur Freigabe/Sperrung von Interrupts, zur Betriebsartenwahl und zum Speicher-Refresh. Detailliert ist der Befehlssatz in der "CPU U 8001/U 8002 - Befehlsbeschreibung" erläutert.

2.9.1. Befehlsformate

Bild 2.7 zeigt die Formate der Befehle. Die zwei höchstwertigen Bits im Befehlswort bestimmen, ob das kompakte Befehlsformat (A) oder das allgemeine Befehlsformat (B) verwendet wird. Das kompakte Format kodiert die vier am häufigsten verwendeten Befehle in Einwortbefehle, was einen geringen Befehlsspeicherbedarf und eine Erhöhung der Ausführungsgeschwindigkeit ergibt. Solange die zwei höchstwertigen Bits nicht logisch Eins sind, gilt das allgemeine Format. Beim allgemeinen Format reichen die zwei höchstwertigen Bits in Verbindung mit dem Quellenregisterfeld zur Festlegung einer der fünf Hauptadressierungsarten aus. Quell- und Zielfelder sind vier Bit breit zur Adressierung der 16 Allzweck-Register.

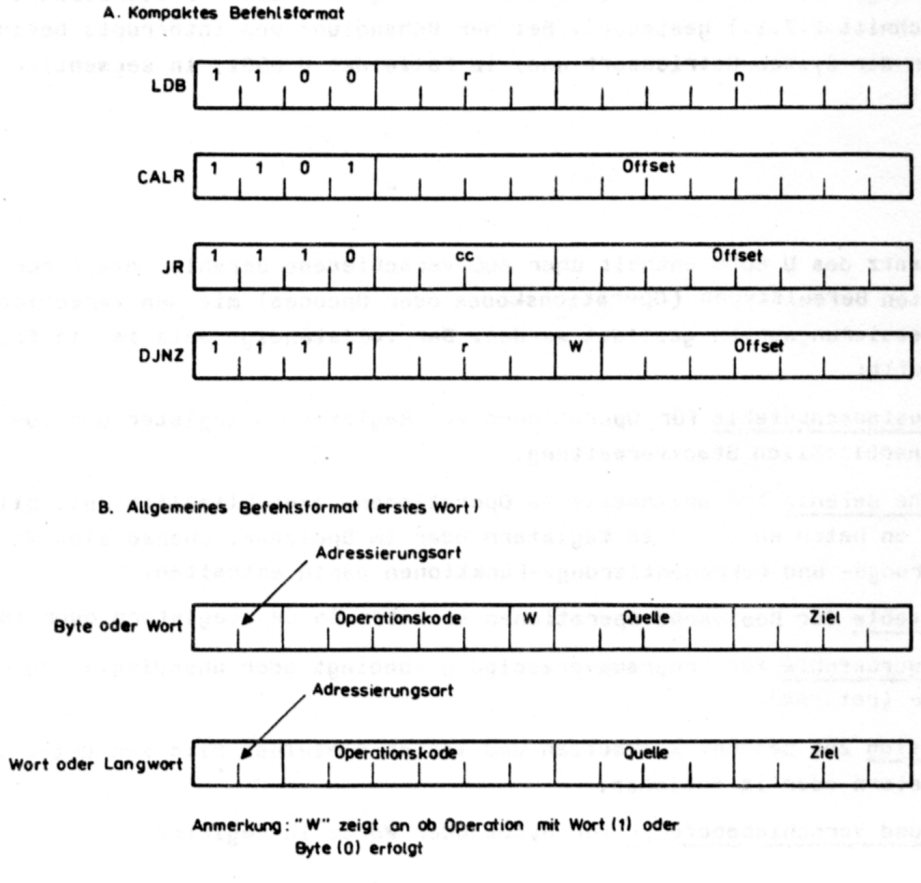


Bild 2.7 Befehlsformate

2.10. Datentypen

Die CPU U 8000 gestattet die Manipulation von acht Datentypen, von denen fünf feste Längen besitzen, während sich die Längen der drei anderen dynamisch ändern können. Jeder Datentyp wird durch eine Anzahl von Befehlen unterstützt, d. h. sie arbeiten direkt mit diesem Datentyp. Folgende Datentypen existieren:

- Bit
- Vorzeichenbehaftetes oder vorzeichenloses binäres ganzzahliges Byte, Wort, Langwort oder Doppellangwort
- Byte oder Wort als logischer Wert
- Wort-Adresse (unsegmentiert) oder Langwortadresse (segmentiert)
- Zweistellige BCD-Zahl ohne Vorzeichen
- Folge von Bytedaten mit dynamischer Länge
- Folge von Wortdaten mit dynamischer Länge
- Stackbereich aus Wortdaten mit dynamischer Länge.

Bits können in Registern oder im Speicher manipuliert werden.
Binäre und BCD-Zahlen und logische Werte können nur in Registern manipuliert werden, wenn auch Operanden direkt vom Speicher geholt werden können. Adressen werden nur in Registern akzeptiert, während Datenfolgen und Stackbereich nur im Speicher manipuliert werden können.

2.11. Adressierungsarten

Die in den U 8000-Befehlen enthaltene Information besteht aus der auszuführenden Operation, der Art und Größe der zu behandelnden Datenelemente sowie der Lokalisierung der Datenelemente. Die Lokalisierung erfolgt mit Hilfe einer der folgenden acht Adressierungsarten:

"Register" Das Datenelement ist in einem der 16 Allzweck-Register untergebracht.

"Direktwert" Das Datenelement ist im Befehl untergebracht.

"Indirektes Register" Das Datenelement befindet sich auf dem Speicherplatz, dessen Adresse in einem Register steht.

"Direkte Adressierung" Das Datenelement befindet sich auf dem Speicherplatz, dessen Adresse im Befehl enthalten ist.

"Indiziert" Das Datenelement befindet sich auf dem Speicherplatz, dessen Adresse sich aus der Summe des Inhaltes eines Registers (16-Bit-Index) und einer Adresse im Befehl ergibt.

"Relative Adressierung" Das Datenelement befindet sich auf dem Speicherplatz, dessen Adresse die Summe aus Inhalt des Befehlszählers (PC) und einem im Befehl enthaltenen 16-Bit-Verschiebungswert ist.

"Basisadressierung" Das Datenelement befindet sich auf dem Speicherplatz, dessen Adresse die Summe aus einer Basisadresse in einem Register und einem im Befehl enthaltenen 16-Bit-Verschiebungswert ist.
"Basis Indiziert" Das Datenelement befindet sich auf der Stelle, deren Adresse die Summe aus einer Basisadresse in einem Register und einem indizierten Wert in einem anderen Register ist.

2.12. Erweiterte Prozessorarchitektur

Ein wichtiges Merkmal der U 8000-Architektur ist die Möglichkeit der erweiterten Prozessorarchitektur (EPA). Hier ist ein Mechanismus vorhanden, mit dem der Grundbefehlssatz der CPU über externe Einheiten, die als EPU's oder Erweiterungsprozessoren bezeichnet werden, erweitert werden kann. Ein spezieller Teil des Befehlssatzes (erweiterte Befehle) dient zur Steuerung. Trifft die CPU in ihrem Befehlsstrom auf einen dieser erweiterten Befehle, so führt sie entweder einen Trap aus und simuliert in der Traproutine den erweiterten Befehl durch Software, oder sie führt den Datentransferteil des Befehls aus und überläßt die Ausführung des Datenmanipulationsteils des Befehls dem Erweiterungsprozessor. Ob die CPU einen Trap ausführt oder Informationen zur EPU überträgt, hängt von der Einstellung des EPA-Bits im Flag- und Steuerwortregister (FCW) ab.

Das der EPA-Möglichkeit zugrundeliegende Prinzip ist die Betrachtung der CPU als Befehlsprozessor; die CPU holt Befehle, holt die dem Befehl zugehörigen Daten, führt die Operationen aus und speichert das Ergebnis. Eine Erweiterung der Anzahl von ausführbaren Operationen beeinflußt nicht den Befehlshol- und Adressenberechnungsteil der CPU-Aktivität. Die erweiterten Befehle nutzen diese Möglichkeit aus, d. h. die CPU holt den Befehl und führt jede erforderliche Adressenberechnung aus. Sie erzeugt auch die Zeitsignale zum Speicherzugriff, falls Daten zwischen dem Speicher und dem Erweiterungsprozessor übertragen werden müssen. Die eigentliche Datenmanipulation jedoch wird durch die EPU ausgeführt. Die erweiterte Prozeßarchitektur (EPA) wird in Kapitel 4 ausführlicher erläutert.

2.13. Ausnahmen

Drei Ereignisse können die normale Ausführung eines U 8000-Programms verändern: Hardware-Interrupts, die auftreten, wenn für eine periphere Einheit ein Service erforderlich ist; synchrone Software-Traps, welche bei einer Fehlerbedingung auftreten, und Systemrücksetzen. Kapitel 6 enthält eine ausführliche Beschreibung der Ausnahmen und wie sie zu behandeln sind.

2.13.1. Rücksetzen (Reset)

Ein Systemrücksetzen hebt alle anderen Betriebsbedingungen auf. Es setzt die CPU in einen bekannten Zustand und bewirkt dann, daß aus einem reservierten Speicherbereich ein neuer Programmstatus geholt wird zur Initialisierung des Flag- und Steuerworts (FCW) und des Befehlszählers (PC).

2.13.2. Traps

Traps sind synchrone Ereignisse, die normalerweise durch spezifische Befehle ausgelöst werden und jedesmal wieder auftreten, wenn der Befehl mit dem gleichen Datensatz und dem gleichen Prozeß oder Zustand ausgeführt wird. Die vier Trap-Arten sind wie folgt:

In der Nicht-EPA-Betriebsart versuchter erweiterter Befehl.
Der laufende Befehl ist ein EPU-Befehl, das System befindet sich jedoch nicht in der EPA-Betriebsart. Dieser Trap gestattet es der Systemsoftware, den Befehl entweder zu simulieren oder das Programm abzubrechen.

Im Normal-Modus versuchter privilegierter Befehl.
Der laufende Befehl ist privilegiert (z. B. E/A), die CPU befindet sich jedoch im Normal-Modus.

Systemaufruf-Befehl (SC, system call).
Dieser Befehl sieht einen gesteuerten Zugriff von Normal-Modus- auf System-Modus-Operation vor.

Segmentierungsverletzung (wird durch externen Schaltkreis geliefert).
Ein externes Speicherverwaltungssystem kann durch einen Segmenttrap der CPU U 8001 signalisieren, daß eine Segmentierungsverletzung aufgetreten ist. Dies wäre z. B. bei Verwendung eines Offsets, der größer als die definierte Länge des Segments ist, der Fall.

2.13.3. Interrupts

Interrupts sind asynchrone Vorgänge, die typischerweise durch periphere Einheiten ausgelöst werden. Die drei Arten von Interrupts, die den drei Interruptleitungen der CPU zugeordnet sind, sind wie folgt:

Nicht-maskierbare Interrupts ($\overline{NMI}$). Diese Interrupts können nicht abgewiesen werden und sind normalerweise für wichtige externe Vorgänge reserviert, die sofortige Bedienung erfordern.

Vektorisierte Interrupts ($\overline{VI}$). Diese Interrupts nutzen acht Bits des durch die unterbrechende Einheit ausgegebenen Vektors zur Auswahl einer besonderen Interruptserviceprozedur, in welche automatisch eingetreten wird. Sie können maskiert werden.

Nicht-vektorisierte Interrupts ($\overline{NVI}$). Diese Interrupts sind maskierbare Interrupts, die alle durch die gleiche Interruptprozedur verarbeitet werden.

2.13.4. Trap- und Interruptservice-Prozeduren

Interrupts und Traps werden durch die CPU U 8000 ähnlich behandelt. Die CPU U 8000 bestätigt auftretende Interrupts bei Freigabe automatisch. Interrupts und Traps werden im System-Modus bearbeitet. Beim segmentierten Typ U 8001 verwendet die CPU die segmentierte Betriebsweise ohne Rücksicht auf ihre Betriebsart zur Zeit des Interrupts oder Traps. Die gerade vor dem Interrupt oder dem Trap wirksame Programmstatusinformation wird auf den Systemstack gebracht. Ein zusätzliches Wort (Identifier), das als spezielle Kennung für den Interrupt oder den Trap dient, wird ebenfalls auf den Systemstack abgelegt, wo es von der Interrupt- oder Trap-Bearbeitungsroutine ausgewertet werden kann. Die Programmstatusregister werden mit der vom Programmstatusbereich des Speichers erhaltenen neuen Statusinformation geladen. Dann wird die Steuerung auf die Serviceprozedur übertragen, deren Adresse sich nun im Befehlszähler befindet. Einzelheiten zur Interrupt- und Trapbehandlung siehe Kapitel 6.

Kapitel 3

3. Adreßräume

3.1. Einleitung

Programme und Daten können im Hauptspeicher des Computersystems oder in peripheren Geräten untergebracht sein. In beiden Fällen muß der Ort der Information durch eine Adresse gegeben sein, bevor zu dieser Information zugegriffen werden kann. Ein Satz dieser Adressen wird als Adreßraum bezeichnet. Die CPU U 8000 unterhält zwei verschiedene Arten von Adressen und damit zwei Kategorien von Adreßräumen:

- Speicheradressen, welche die Speicherplätze im Hauptspeicher angeben.
- E/A-Adressen, welche die Ports angeben, über welche zu peripheren Geräten zugegriffen wird.

Die CPU erzeugt bei vier Operationsarten Adressen:
- Holen von Befehlen (siehe Kapitel 4)
- Holen und Ablegen von Operanden (siehe Kapitel 5)
- Ausnahmebearbeitung (siehe Kapitel 6)
- Refresh (siehe Kapitel 7)

Kapitel 8 enthält eine Beschreibung und Darstellung der bei der Adreßerzeugung auftretenden Zeitabläufe.

3.2. Adreßraumtypen

Innerhalb der beiden allgemeinen Adreßraumtypen (Speicher und E/A) ist es möglich, verschiedene Untergruppen zu unterscheiden.
Bild 3.1 zeigt die Adreßräume, welche sowohl beim Typ U 8001 als auch beim U 8002 zur Verfügung stehen. Der Unterschied zwischen dem U 8001 und U 8002 liegt nicht in der Anzahl und Art der Adreßräume, sondern in der Organisation und Maximalgröße eines jeden Raumes. Beim U 8001 enthält jeder der sechs Speicheradreßräume 8 MByte, die in 128 Segmente unterteilt sind, welches eine gesamte Speicheradressiermöglichkeit von 48 MBytes ergibt. Beim U 8002 ist jeder Speicherraum ein nicht weiter unterteilter Bereich aus 64 KByte-Adressen.
Beim U 8001 wie auch beim U 8002 enthält der E/A-Adreßraum 64 K Port-Adressen.
Wird eine Adresse zum Datenzugriff benutzt, dann können die Adreßräume durch den Zustand der Statusleitungen (ST_0-ST_3) unterschieden werden.

- Befehlsräume (Status = 1100 oder 1101).
  Betriebsart Normal ($N/\overline{S}$=1) oder System ($N/\overline{S}$=0). Diese Räume adressieren typischerweise einen Speicher, der Anwenderprogramme (Normal) oder Systemprogramme (System) enthält.
- Datenräume (Status = 1000 oder 1010), Betriebsart Normal ($N/\overline{S}$=1) oder System ($N/\overline{S}$=0). Diese Räume können zum Adressieren der Daten verwendet werden, mit denen Anwender- oder Systemprogramme arbeiten.
- Stackspeicherräume (Status = 1001 oder 1011), Betriebsart Normal ($N/\overline{S}$=1) oder System ($N/\overline{S}$=0). Diese Räume können zum Adressieren der System- und Normal-Programmstackspeicher verwendet werden.

- Standard E/A-Raum (Status = 0010). Dieser Raum adressiert alle E/A-Ports, die für Peripheriegeräte des U 8000 verwendet werden.
- Spezial-E/A-Raum (Status = 0011). Dieser Raum adressiert Ports von CPU-Ergänzungsschaltkreisen (wie z. B. den Speicherverwaltungsbaustein MMU U 8010).

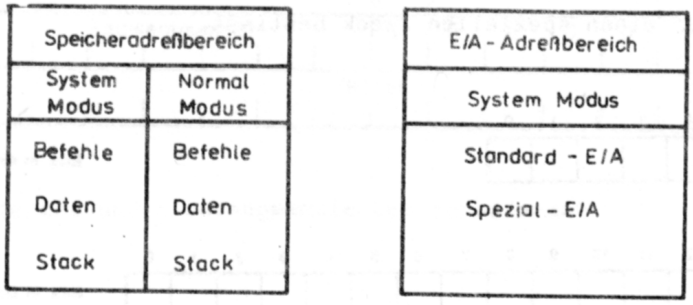


Bild 3.1 Adreßbereiche von U 8001 bzw. U 8002

3.3. E/A-Adreßräume

Alle E/A-Adressen werden durch 16-Bit-Worte dargestellt. Jedes der adressierten Ports hat eine Breite von entweder acht oder 16 Bit. Transfers zu oder von 16-Bit-Ports erfolgen immer mit Wortdaten und bei 8-Bit-Ports stets mit Byte-Daten.

Für beide Adreßräume kann die Adresse eines 16-Bit-Ports gerade oder ungerade sein. Beim Standard-E/A-Raum müssen Byte-Ports eine ungerade Adresse haben, während beim Spezial-E/A-Raum die Byte-Ports eine gerade Adresse besitzen müssen.

3.4. Speicheradreßräume

Jeder Speicheradreßraum im U 8002 oder jedes Segment in jedem Speicheradreßraum beim U 8001 kann als Adressierung einer Datenfolge von 64 KBytes angesehen werden, die fortlaufend in aufsteigender Folge numeriert ist.
In den Speicheradreßräumen des U 8000 ist das Byte (8 Bit) das adressierbare Grundelement. Es gibt jedoch noch drei andere adressierbare Datenelemente:

- Bits, entweder in Bytes oder Worten
- Wort (16 Bit)
- Langworte (32 Bit)

3.4.1. Adressierbare Datenelemente

Die Art des adressierten Datenelementes hängt vom ausgeführten Befehl ab. Wie in der "CPU U 8001/U 8002 - Befehlsbeschreibung" ausführlich erläutert, dienen unterschiedliche Assembler-Mnemoniks zum Adressieren von Bytes, Worten und Langworten. Außerdem können nur bestimmte Befehle Bits adressieren.
Ein Bit kann durch Spezifizierung einer Byte- oder Wortadresse sowie der Nummer des Bits innerhalb des Bytes (0 - 7) oder des Wortes (0 - 15) adressiert werden. Bits werden von rechts nach links numeriert, d. h. vom niederwertigsten zum höchstwertigen. Damit entspricht das Bit n der Position in der herkömmlichen Darstellung positiver Binärzahlen (siehe Bild 3.2). Die Adresse eines Datentyps, der länger ist als ein Byte (Wort oder Langwort), ist die gleiche wie die Adresse des Bytes mit der niedrigsten Speicheradresse innerhalb des Wortes oder Langwortes (Bild 3.2). Dies ist

das ganz links stehende, höchstwertige Byte des Wortes oder Langwortes. Wort- oder Langwortadressen sind stets geradzahlig. Niederwertige Bytes von Worten werden an ungeradzahligen Speicherplätzen abgelegt, höherwertige Bytes an geradzahligen Stellen. Byteadressen können gerad- oder ungeradzahlig sein. Bestimmte Speicherplätze sind für die Behandlung von Reset reserviert. Diese werden ausführlich im Kapitel 6 beschrieben. Mit Ausnahme dieser reservierten Plätze sind keine Speicheradressen für einen speziellen Zweck bestimmt.

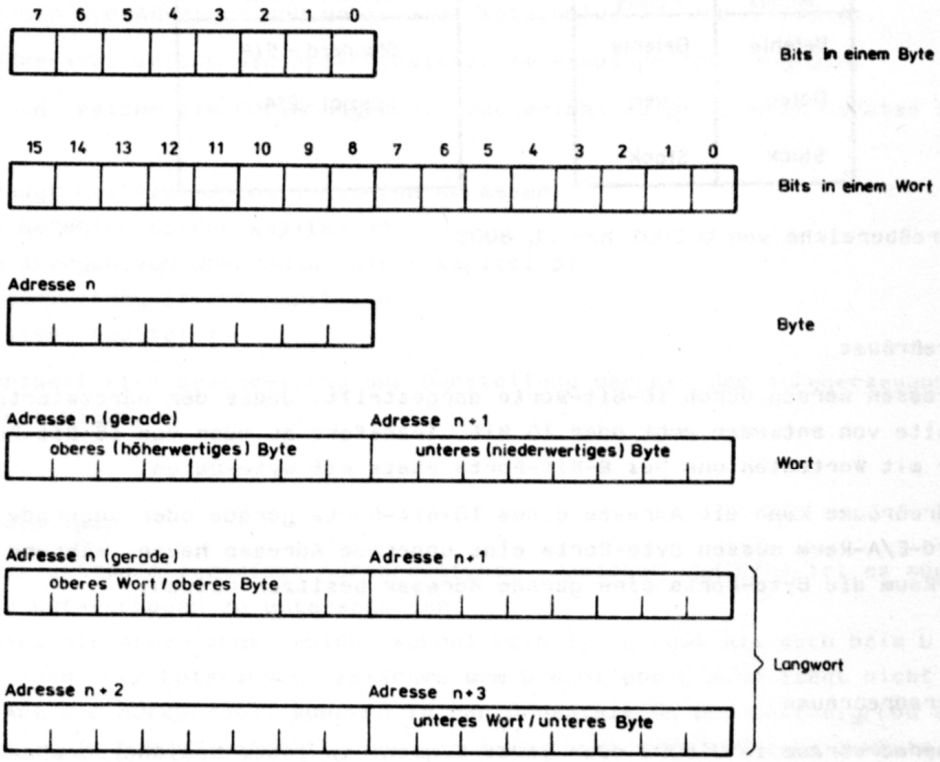


Bild 3.2 Adressierbare Datenelemente

3.4.2. Segmentierte und nichtsegmentierte Adressen

Die zwei Versionen der CPU U 8000 erzeugen zwei Adressenarten mit unterschiedlichen Längen. Der Typ U 8002 erzeugt eine 16-Bit-Adresse, die ein Byte von 64 KBytes spezifiziert. Der Typ U 8001 erzeugt eine segmentierte 23-Bit-Adresse. Eine segmentierte Adresse besteht aus einer 7-Bit-Segmentnummer, welche eines von 128 Segmenten festlegt, sowie einem 16-Bit-Offset, das ein Byte von bis zu 64 KBytes im Segment festlegt. Jedes Segment ist unabhängig von den anderen, somit können Befehle und Mehrfachbyte-Datenelemente die Segmentgrenzen nicht überschreiten. Einige Vorteile der Adressensegmentierung sind im Abschnitt 3.4.3. aufgeführt.

Bild 3.3 zeigt das Format der segmentierten und nichtsegmentierten Adressen. Letztere sind 16 Bit lang und können somit in Wortregistern (Rn) oder im Speicher als Wortelemente gespeichert werden. Die segmentierten 23-Bit-Adressen sind in ein 32-Bit-Langwort eingebettet und können somit in einem Langwortregister (RRn) oder einem Langwortspeicherelement gespeichert werden.

Es ist wichtig zu verstehen, daß der Typ U 8001 zwar im unsegmentierten Modus arbeiten kann (Kapitel 4), er aber stets segmentierte Adressen erzeugt.

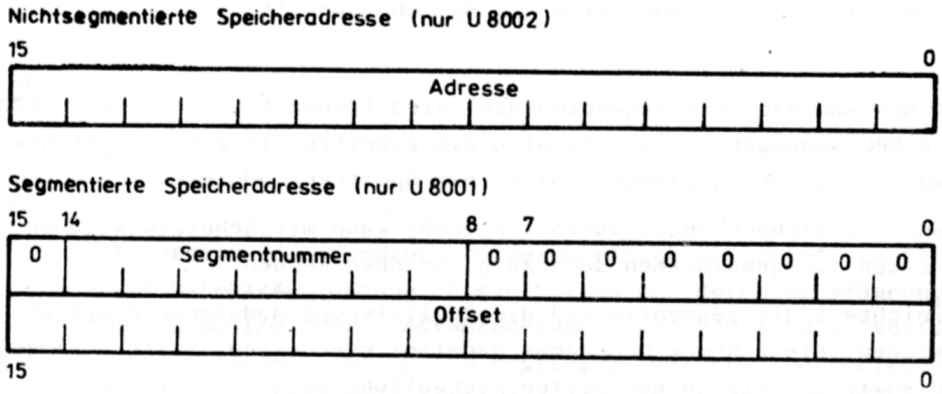


Bild 3.3 Segmentiertes und nichtsegmentiertes Adreßformat

3.4.3. Segmentierung und Speicherverwaltung

Die vom Programmierer manipulierten, von den Befehlen benutzten und durch den U 8001 ausgegebenen Adressen werden "logische Adresse" genannt. Ein externer Speicherverwaltungsbaustein kann logische Adressen in physische (tatsächliche) Speicheradressen übersetzen und bestimmte Kontrollen ausführen, um zu gewährleisten, daß auf die Daten und Programme richtig zugegriffen wird.
Der Speicherverwaltungsbaustein MMU U 8010 führt diese Funktion für die von der CPU U 8001 erzeugten segmentierten Adressen aus.
Ein einzelner Speicherverwaltungsbaustein hält für 64 Segmente je einen Deskriptor bereit. Dieser Deskriptor sagt, wo das Segment im physischen Speicher liegt, wie lang es ist und welche Art von Zugriff zum Segment durchgeführt werden darf. Der U 8010 verwendet diese Deskriptoren zur Übersetzung logischer Segmentnummern und Offsets in physische 24-Bit-Adressen (wie in Bild 3.4 dargestellt). Gleichzeitig erkennt die MMU solche Fehler wie Schreiben in ein Nur-Lese-Segment oder Zugriff zu einem Systemsegment durch ein Programm außerhalb des Betriebssystems.
Die MMU's wurden kombinierbar ausgelegt, so daß mehr als 64 Segmente auf einmal verwaltet werden können. Einige der durch die MMU gewährten Vorteile der Speicherverwaltung sind folgende:

- Flexible und wirksame Zuordnung von physischen Speicherressourcen während der Ausführung von Programmen.
- Möglichkeit mehrerer, unabhängig voneinander ausgeführter Programme, die sich den Zugriff zu gemeinsamen Programmen und Daten teilen können.
- Schutz vor unbefugten oder unbeabsichtigten Zugriff auf Daten oder Programme.
- Feststellung offensichtlich falscher Verwendung des Speichers durch ein ausführendes Programm.
- Trennung der Anwender von Systemfunktionen.

Durch die Segmentierung im U 8001 wird die Speicherverwaltung auf zweierlei Art unterstützt:

- Ein Teil der Adresse (die Segmentnummer) wird frühzeitig in einem Speicherzyklus durch die CPU ausgegeben. Dadurch wird die Zugriffszeit zum Segmentdeskriptor in der MMU nicht zur Basiszugriffszeit des Speichers addiert.
- Eine Standardspeichereinheit variabler Größe kann mit Schutzfunktionen und den Möglichkeiten zur gemeinsamen Benutzung versehen werden.

Außerdem erleichtert die Segmentierung die Realisierung modularer Programme und Daten in Systemen, die mehrere Programme parallel bearbeiten. Weitere Einzelheiten über die MMU sowie Prinzipien der Speicherverwaltung werden in "MMU U 8010 - Technische Beschreibung" erläutert.

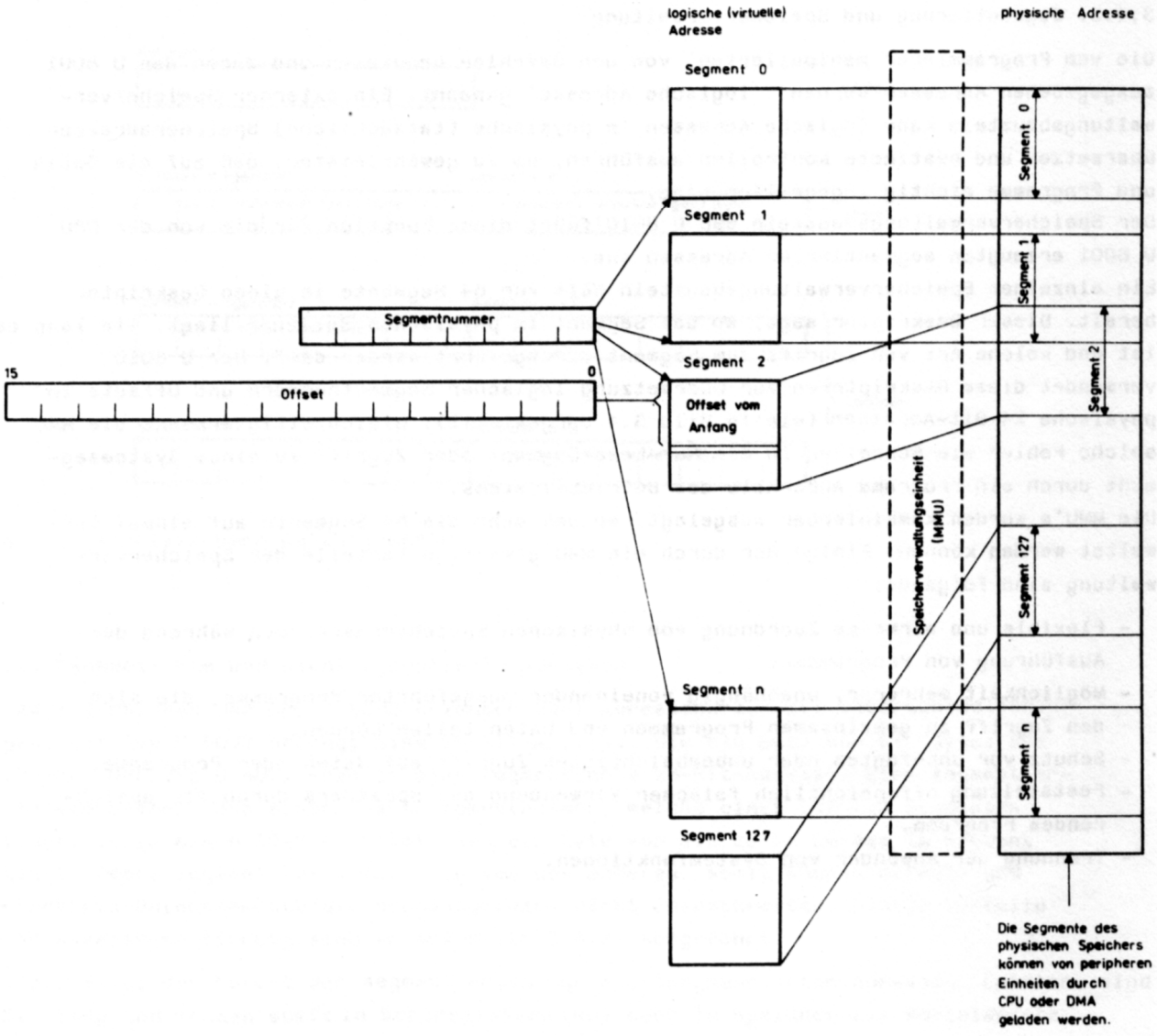


Bild 3.4 Übersetzung der segmentierten Adresse durch die MMU

Kapitel 4

4. CPU-Betrieb

4.1. Einleitung

Das folgende Kapitel enthält die grundlegende Beschreibung der Betriebszustände der CPU U 8000 und der Befehlsausführung. Einzelheiten der Befehlsausführung sind im Kapitel 5 "CPU U 8001/U 8002 - Befehlsbeschreibung" zu finden. Weitere Aspekte sind im Kapitel 6 "Ausnahmen" und im Kapitel 7 "Refresh" beschrieben. Kapitel 8 enthält die CPU-Aktivitäten, wie sie an den äußeren Anschlüssen der CPU auftreten.

4.2. Betriebszustände

Die CPU U 8000 verfügt über drei Betriebszustände: Arbeitszustand, Stop/Refresh-Zustand und Bus-Abschalt-Zustand. Der Arbeitszustand ist der übliche Zustand des Prozessors: Die CPU führt Befehle aus oder behandelt Ausnahmen. Sie tritt in den Stop/Refresh-Zustand ein, wenn die $\overline{\text{STOP}}$-Leitung aktiviert wird, oder wenn der Refreshzähler die Notwendigkeit eines periodischen Auffrischens anzeigt. In diesem Zustand werden Speicherrefresh-Vorgänge erzeugt (siehe Kapitel 7). Die CPU nimmt den Bus-Abschalt-Zustand ein, wenn sie eine Busanforderung bestätigt und die Steuerung des Systembusses abgibt. Bild 4.1 zeigt die drei Zustände und die Bedingungen, welche Zustandsübergänge verursachen.

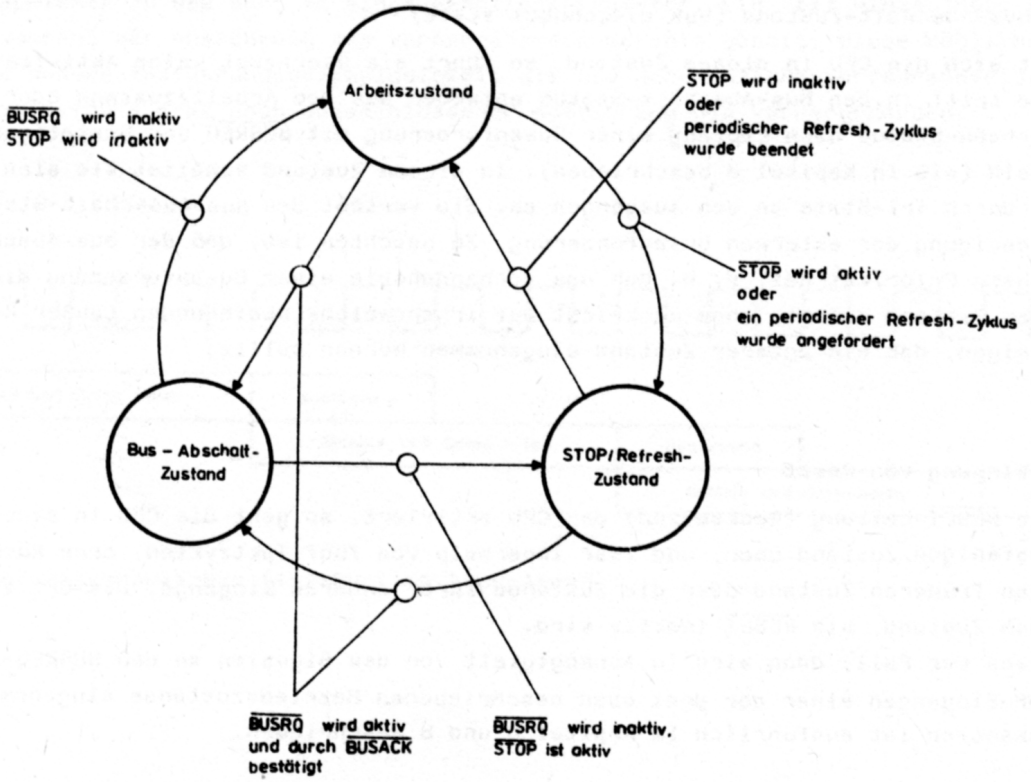


Bild 4.1 Betriebszustände und Übergänge

4.2.1. Arbeitszustand (running state)

Befindet sich die CPU im Arbeitszustand, so führt sie entweder Befehle aus (wie in Abschnitt 4.3. beschrieben), oder sie behandelt Ausnahmen (wie in Kapitel 6 beschrieben). Normalerweise befindet sich die CPU im Arbeitszustand, sie verläßt ihn jedoch beim Auftreten einer der drei folgenden Bedingungen:

- Der Refresh-Mechanismus zeigt an, daß ein periodisches Auffrischen erforderlich ist. In diesem Fall nimmt die CPU zeitweilig den STOP/Refresh-Zustand ein.
- Eine externe Stopanforderung bringt die CPU in den STOP/Refresh-Zustand.
- Eine externe Busanforderung bringt die CPU in den Bus-Abschaltzustand.

4.2.2. Stop/Refreshzustand

Befindet sich die CPU in diesem Zustand, so erzeugt sie ständig Auffrischzyklen (wie in Kapitel 7 beschrieben) und führt keine andere Funktion aus.
Dieser Zustand ermöglicht die Erzeugung von Speicherauffrischungen durch die CPU und gestattet externen Geräten das Aufschieben der CPU-Operation. Dadurch kann Einzelschrittbetrieb des Prozessors erzwungen oder die CPU mit einem Erweiterungsprozessor synchronisiert werden (wie in Abschnitt 4.4. beschrieben).
Die CPU nimmt den Stop/Refresh-Zustand ein, wenn der Refresh-Mechanismus ein Auffrischen verlangt oder die Stop-Leitung aktiviert wird. Sie verläßt den Stop/Refresh-Zustand, wenn keine dieser Bedingungen zutrifft oder wenn die CPU durch eine Busanforderung in den Bus-Abschalt-Zustand übergeht.

4.2.3. Bus-Abschalt-Zustand (bus disconnect state)

Befindet sich die CPU in diesem Zustand, so führt sie überhaupt keine Aktivitäten aus. Sie tritt in den Bus-Abschalt-Status entweder aus dem Arbeitszustand oder dem Stop/Refresh-Status nach Empfang einer Busanforderung mit $\overline{\text{BUSREQ}}$ und Bestätigung mit $\overline{\text{BUSACK}}$ ein (wie in Kapitel 8 beschrieben). In diesem Zustand schaltet sie sich selbst vom Bus durch Tri-State an den Ausgängen ab. Sie verläßt den Bus-Ausschalt-Status nach Beendigung der externen Busanforderung. Zu beachten ist, daß der Bus-Abschalt-Zustand die höchste Priorität hat, d. h. daß das Vorhandensein einer Busanforderung die CPU in diesem Zustand zwingt, ohne Rücksicht auf irgendwelche Bedingungen (außer Reset), die anzeigen, daß ein anderer Zustand eingenommen werden sollte.

4.2.4. Wirkung von Reset

Wird die $\overline{\text{RESET}}$-Leitung (Rücksetzen) der CPU aktiviert, so geht die CPU in einen nichtbetriebsfähigen Zustand über, und zwar innerhalb von fünf Taktzyklen, ohne Rücksicht auf ihren früheren Zustand oder die Zustände ihrer anderen Eingänge. Die CPU verbleibt in diesem Zustand, bis $\overline{\text{RESET}}$ inaktiv wird.
Ist dieses der Fall, dann wird in Abhängigkeit von den Signalen an den BUSREQ-Eingängen und STOP-Eingängen einer der drei oben beschriebenen Betriebszustände eingenommen.
Das Rücksetzen ist ausführlich im Kapitel 6 und 8 beschrieben.

4.3. Befehlsausführung

Die CPU wird durch die Programmstatusregister (Bild 2.6) gesteuert, wenn sie sich im Arbeitszustand befindet und Befehle ausführt. Der Befehlszähler gibt die Adresse an, von welcher der nächste Befehl geholt wird, die Flags steuern die bedingten Verzweigungen (siehe "Befehlsbeschreibung") und die Steuerbits bestimmen die Betriebsart, in welcher die CPU weiterarbeitet (siehe Abschnitt 4.3.) sowie die Interrupts, die maskiert werden sollen (siehe Kapitel 6).
Die Befehlsausführung besteht aus der wiederholten Anwendung von zwei Schritten:

- Das Holen eines oder mehrerer Worte, die einen einzelnen Befehl definieren, aus dem Programmspeicher bei der vom Befehlszähler (PC) angegebenen Adresse.
- Durchführung der von dem Befehl angegebenen Operation sowie Aktualisieren des Befehlszählers und der Flags in den Programmstatusregistern.

Die von einem Befehl ausgeführte Operation und die Art, in welcher die Flags aktualisiert werden, hängt von dem jeweils ausgeführten Befehls ab und werden ausführlich in der "Befehlsbeschreibung" erläutert. Für die meisten Befehle wird der PC-Wert aktualisiert, um auf das dem letzten Wort des Befehls unmittelbar folgende Wort zeigen zu können. Dadurch werden die Befehle nacheinander aus dem Speicher geholt. Ausnahmen hierzu sind Verzweigungs-, Aufruf- und Rückkehrbefehle, durch welche der Befehlszähler (PC) auf einen vom Befehl erzeugten Wert eingestellt wird. Dies bewirkt die Fortsetzung der Programmabarbeitung bei der im PC neu festgelegten Adresse.
Die CPU U 8000 ist in der Lage, das Holen eines Befehles mit der Ausführung des vorherigen Befehles zu überlappen. Diese Möglichkeit, welche als Instruction Look-Ahead (Befehls-Vorausschau) bezeichnet wird, ist in Bild 4.2 dargestellt. Sie zeigt die Ausführung einer Reihe von Speicher-zu-Register-Befehlen, z. B. das Addieren eines Wertes im Speicher zu dem Wert in einem Mehrzweck-Register. Ein Teil eines jeden Befehls wird noch während der Ausführung des vorhergehenden Befehls geholt. Diese Möglichkeit bietet eine höhere Ausführungsgeschwindigkeit als die herkömmliche Verfahrensweise des Holens eines Befehls erst nach abgeschlossener Ausführung des vorhergehenden.

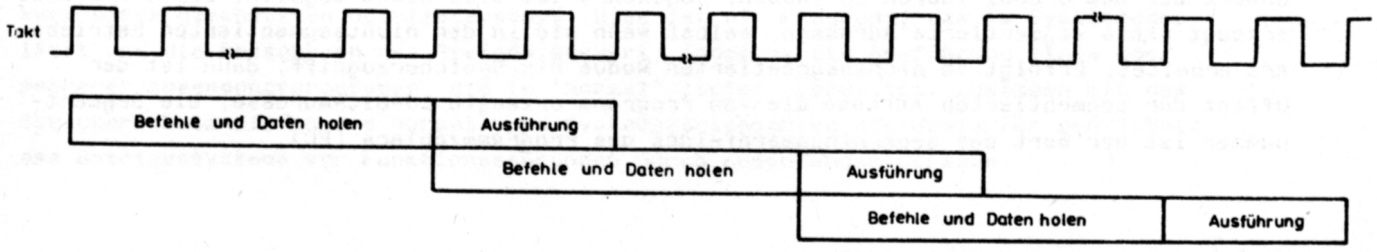


Bild 4.2 Befehlsvorausschau (Instruction Look-Ahead)

Nach Ausführung und in einigen Fällen (siehe "Befehlsbeschreibung" und Kapitel 6) auch während der Ausführung eines Befehls prüft die CPU, ob Traps oder Interrupts anhängig und nicht maskiert sind. Ist dies der Fall, so verschiebt sie zeitweilig die Befehlsausführung und beginnt einen Standard-Ausnahmebehandlungszyklus. Während dieses im Kapitel 6 ausführlich beschriebenen Zyklus wird der Wert der Programmstatusregister gerettet und ein neuer Wert geladen. Dann geht die Befehlsausführung mit einem neuen PC-Wert sowie Flag- und Steuerwortwert weiter. Auf diese Weise wird durch die CPU die Ausführung eines Programms unterbrochen und in das Trap- oder Interruptserviceprogramm eingetreten.

4.3.1. Betriebsarten des Arbeitszustandes

Während die CPU Befehle ausführt, wird ihre Betriebsart durch zwei Steuerbits im FCW gesteuert, und zwar durch das System/Normal-Bit (S/$\overline{N}$) und das Bit für die Segmentierungsart (SEG).

4.3.2. Segmentierte und nichtsegmentierte Betriebsart

Das Segmentierungsbit im FCW der CPU bestimmt die Größe und das Format der Adressen, welche durch Programme direkt manipuliert werden. Bei Segmentierung (SEG = 1) manipulieren die Programme segmentierte 23-Bit-Adressen; bei Nichtsegmentierung (SEG = 0) bearbeiten die Programme nichtsegmentierte 16-Bit-Adressen. Durch die unterschiedliche Adressengröße treten im Adreßteil der Befehle folgende Differenzen auf:

- Indirekte und Basisregister sind 32-Bit-Register in der segmentierten und 16-Bit-Register in der nichtsegmentierten Betriebsart.
- Adressen im Befehl haben stets 16 Bit im nichtsegmentierten Modus. Dagegen bestehen sie aus einer 7-Bit-Segmentnummer und einem 8-Bit- oder 16-Bit-Offset bei segmentiertem Modus.

Die Segmentierung ist nur bei der CPU U 8001 möglich. Bei der U 8002 ist das Segmentbit stets auf Null gestellt und zeigt Nichtsegmentierung an. Da die CPU U 8001 beide Segmentierungsarten erlaubt, ist es möglich, für den U 8002 geschriebene Programme unverändert auf dem U 8001 laufen zu lassen; umgekehrt ist dies nicht möglich. Die CPU U 8001 erzeugt stets segmentierte Adressen, selbst wenn sie in der nichtsegmentierten Betriebsart arbeitet. Erfolgt im nichtsegmentierten Modus ein Speicherzugriff, dann ist der Offset der segmentierten Adresse die vom Programm erzeugte 16-Bit-Adresse; die Segmentnummer ist der Wert des Segmentnummernfeldes des Programmzählers (PC).

4.3.3. Betriebsart Normal und System

Die Betriebsart der CPU (System oder Normal) bestimmt, welche Befehle ausgeführt werden können und welcher Stackpointer verwendet wird. Im System-Modus (S/$\overline{N}$ = 1) können alle Befehle ausgeführt werden.
Bei "Normal" können bestimmte privilegierte Befehle, die empfindliche Teile des Maschinenzustandes ändern (wie E/A-Operationen oder Beeinflussung der Steuerregister), nicht ausgeführt werden. Der zweite Unterschied zwischen der Betriebsart "System" und "Normal" ist der Zugriff zum System- oder Normalstackpointer.

Wie in Bild 2.5 gezeigt, gibt es zwei Exemplare von Stackpointerregistern (Register 15 im U 8002 und Register 14 und 15 im U 8001); und zwar einmal für "Normal" und einmal für "System".

Wird im Normal-Modus durch einen Befehl auf das Stackpointerregister zugegriffen, so wird der Normalstackpointer verwendet. Bei "System" bezieht sich ein Zugriff zum Stackpointerregister auf den Systemstackpointer. Wenn der U 8001 in der nichtsegmentierten Systembetriebsart läuft, so bezieht sich ein Zugriff auf R14 auf das allgemeine Register R14. Zusammengefaßt ist dies in Tabelle 4.1 dargestellt.

Im Normal-Modus ist der Systemstackpointer nicht zugreifbar; in der Systembetriebsart wird zum Normalstackpointer unter Verwendung eines speziellen Ladebefehls (LDCTL) zugegriffen.

Register-Bezugnahme durch Befehl	System-Modus		Normal-Modus	
	segmentiert	nichtsegmentiert	segmentiert	nichtsegmentiert
R14	System R14	Normal R14	Normal R14	Normal R14
R15	System R15	System R15	Normal R15	Normal R15
RR14	System R14	Normal R14	Normal R14	Normal R14
	System R15	System R15	Normal R15	Normal R15

Anmerkung: Der U 8002 läuft stets im nichtsegmentierten Modus

Tab. 4.1 Verwendete Register bei Bezugnahme von R14 und R15

Die CPU schaltet die Betriebsarten bei Änderung der Programmzustands-Steuerbits um. Das geschieht, wenn ein privilegierter Ladesteuerbefehl ausgeführt wird oder wenn eine Ausnahme (Interrupt, Trap oder Reset) auftritt. Es gibt den Spezialbefehl Systemaufruf (SC), dessen Zweck es ist, ein Trap zu erzeugen und somit einen gesteuerten Übergang von Normal- auf Systemmodus zu schaffen.
Die Unterscheidung zwischen der Betriebsart Normal und System gestattet den Entwurf eines geschützten Betriebssystems. Dies ist ein Programm, das im Systemmodus läuft und die Ressourcen des Systems steuert, indem es die Ausführung eines oder mehrerer Anwendungsprogramme, die in "Normal" laufen, verwaltet. Zusammen mit dem Speicherschutz bieten die Normal- und Systembetriebsarten die Basis für den Schutz des Betriebssystems vor Funktionsstörungen durch Anwendungsprogramme.

4.4. Erweiterte Befehle

Die CPU U 8000 gestattet sechs erweiterte Befehle, die gemeinsam durch die CPU und einen externen Erweiterungsprozessor (EPU) ausgeführt werden können. Die Ausführung dieser Befehle wird durch das Steuerbit der erweiterten Prozeßarchitektur (EPA) im FCW gesteuert. Ist das EPA-Bit Null, so zeigt es an, daß kein Erweiterungsprozessor an die CPU angeschlossen ist und verursacht ein Trap der CPU (wie im Kapitel 6 beschrieben), wenn sie auf einen erweiterten Befehl trifft. Dadurch kann die Operation des erweiterten Befehls durch auf der CPU laufende Software simuliert werden.
Ist das EPA-Bit gesetzt, so zeigt es an, daß ein Erweiterungsprozessor an die CPU angeschlossen ist, um die in dem erweiterten Befehl kodierte Operation zu verarbeiten.

Die CPU holt den erweiterten Befehl und führt jede von diesem Befehl geforderte Adressenberechnung aus. Verlangt der Befehl einen Datentransport, dann erzeugt die CPU die Zeitsignale für diesen Transfer. Die CPU holt den nächsten Befehl in ihren Befehlsstrom und beginnt seine Ausführung. Der Erweiterungsprozessor muß die CPU-Aktivität überwachen, an den von der CPU eingeleiteten Datentransfers von erweiterten Befehlen teilnehmen und den erweiterten Befehl ausführen.

Während der Erweiterungsprozessor den Befehl ausführt, kann die CPU weitere Befehle holen und ausführen. Holt die CPU einen anderen erweiterten Befehl, bevor der Erweiterungsprozessor die Ausführung eines früheren Befehls beendet hat, dann kann die $\overline{\text{STOP}}$-Leitung zum Anhalten der CPU verwendet werden, bis der frühere Befehl abgeschlossen ist. Dieser Vorgang wird ausführlich in "CPU U 8001/U 8002 - Befehlsbeschreibung" und im Kapitel 8 beschrieben.

Kapitel 5

5. Adressierungsarten

5.1. Einleitung

Dieses Kapitel beschreibt die acht Adressierungsarten, die von den Befehlen verwendet werden, um auf Daten im Speicher oder in CPU-Registern zuzugreifen. Am Ende des Kapitels werden Beispiele für die nichtsegmentierte und die segmentierte Betriebsart gegeben.
Die meisten Befehle besitzen neben dem Operationskode (opcode) Operanden. Diese Operanden können in CPU-Registern oder Speicherstellen lokalisiert sein. Die Art, in der auf Operanden zugegriffen wird, wird Adressierungsart genannt. Diese Arten sind in Bild 5.1 dargestellt. Nicht alle Befehle können sämtliche Adressierungsarten verwenden; manche Befehle verwenden nur einige, manche Befehle überhaupt keine. Der Begriff Operand steht für die Daten, die bearbeitet werden sollen.

5.2. Verwendung der CPU-Register

Die 16 Mehrzweckregister der CPU können - mit den nachstehend erwähnten Ausnahmen - auf folgende Weise verwendet werden:
- Als Akkumulatoren, wo die zu verarbeitenden Daten im Register abgelegt sind.
- Als Zeiger (Pointer), wo der Wert im Register nicht der Operand selbst, sondern vielmehr die Speicheradresse des Operanden ist. In String- und Stackbefehlen können die Zeiger automatisch schrittweise zum vorhergehenden oder auch zum nachfolgenden Speicherplatz gerichtet werden.
- Als Index- oder Basisregister, wo der Inhalt des Registers und das (die) dem Befehl folgende Wort (e) zur Erzeugung der Adresse des Operanden kombiniert werden. Dies erlaubt einen effektiven Zugriff zu einer Vielzahl von Datenstrukturen.

Für die o. a. Verwendung von Mehrzweckregistern gibt es zwei Ausnahmefälle:
- Das Register RO (bzw. das Doppelregister RRO bei segmentierter Betriebsweise) kann nicht als indirektes Register, als Basisregister oder Indexregister verwendet werden.
- Das Register R15' (bzw. das Doppelregister RR14' im U 8001) wird bei der Interruptbestätigung verwendet und kann deshalb nie als Akkumulator genutzt werden. Auf die Systemmodus-Register R14' und R15' wird auch dann zugegriffen, wenn Befehle in der Systembetriebsart ausgeführt werden, und R14, R15 oder RR14 angesprochen werden.

Zusätzlich zur allgemeinen Verwendung der U 8000-Register sind folgende Register für Sonderzwecke vorgesehen:
- Register R15 (bzw. das Doppelregister RR14 im U 8001 (dient als Stackpointer für Unterprogrammaufrufe und Rückkehr aus Unterprogrammen.
- Das Byteregister RH1 wird bei erweiterten Befehlen verwendet.

Adressierungsart	Operandenadressierung			Operand
	Im Befehl	In einem Register	Im Speicher	
R Register		Registeradresse → Operand		Der Inhalt des Registers
IM Direktwert	Operand			Im Befehl
*IR Indirektes Register		Registeradresse → Adresse → Operand		Der Inhalt des Speicherplatzes, dessen Adresse sich im Register befindet
DA Direkte Adresse			Adresse → Operand	Der Inhalt des Speicherplatzes, dessen Adresse sich im Befehl befindet
*X Indiziert		Registeradresse → Index Basisadresse → ⊙	→ Operand	Der Inhalt des Speicherplatzes, dessen Adresse im Befehl ist, plus Inhalt des Arbeitsregisters
RA Relative Adresse		PC - Wert Verschiebung → ⊙	→ Operand	Der Inhalt des Speicherplatzes, dessen Adresse der Inhalt des Befehlszählers ist, versetzt durch die Verschiebung im Befehl
*BA Basis - Adresse		Registeradresse → Basisadresse Verschiebung → ⊙	→ Operand	Der Inhalt des Speicherplatzes, dessen Adresse die Adresse im Register ist, versetzt durch die Verschiebung im Befehl
*BX Basis Indiziert		Registeradresse → Basisadresse Registeradresse → Index → ⊙	→ Operand	Der Inhalt des Speicherplatzes, dessen Adresse die Adresse in einem Register ist, plus Indexwert in einem anderen Register

* R0 bzw. RR0 darf nicht als Adreß-, Index- oder Basisregister verwendet werden

Bild 5.1 Adressierungsarten

Ein Befehl besteht aus einem Wort oder mehreren aufeinanderfolgenden Worten und ist stets auf einer geraden Byteadresse im Speicher abgelegt. In der Adressierung "Relative Adresse" (RA) wird der Befehlszähler (PC) anstelle eines Mehrzweck-CPU-Registers zur Lieferung der Basisadresse für die Adressenberechnung verwendet.
Der Befehlszähler dient normalerweise als Zeiger auf den nächsten abzuarbeitenden Befehl, d. h. nachdem ein Befehl aus dem Speicher geholt wurde, wird der Befehlszähler erhöht, um auf den nächsten Befehl zu zeigen. Bei der relativen Adressierung jedoch dient der aktualisierte PC als Basisadresse für den Zugriff auf einen Operanden, der versetzt zum Speicherplatz des Befehls lokalisiert ist. Durch relative Adressierung angegebene Operanden stehen im Programmadressenraum, wenn das Speichersystem zwischen Programm-, Daten- oder Stackadressenräumen unterscheidet.
Zu zwei der Adressierungsarten, nämlich direkte Adresse und indizierte Adresse, gehört eine E/A- oder Speicheradresse als Teil des Befehls. E/A-Adressen sind stets 16 Bit lang, ebenso wie nichtsegmentierte Speicheradressen (U 8002), so daß diese Adressen ein Wort im Befehl einnehmen. Die vom U 8001 erzeugten segmentierten Adressen sind 23 Bit lang. Innerhalb eines Befehls kann eine segmentierte Adresse zwei Worte (16-Bit-langes-Offset) oder ein Wort (8-Bit-kurzes-Offset) einnehmen.
Wie Bild 5.2 zeigt, unterscheidet das Bit 7 des Segmentnummernbytes die beiden Formate. Ist dieses Bit gesetzt, so bedeutet dies lange Offsetdarstellung. Ist das Bit gelöscht, so bedeutet dies kurze Offsetdarstellung. Für eine Adresse mit kurzem Offset wird die segmentierte 23-Bit-Adresse auf 16 Bit reduziert, indem man die acht höchstwertigen Bits des Offset, die als Null angenommen werden, ausläßt.

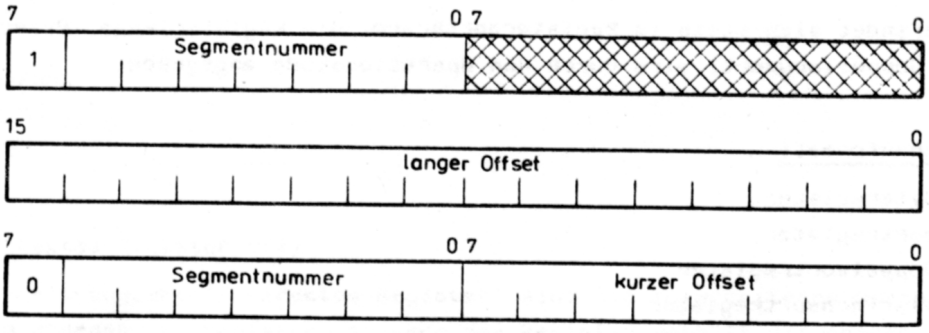


Bild 5.2 Segmentierte Speicheradresse in einem Befehl
(Bemerkung: schraffiertes Feld ist reserviert)

5.3. Beschreibung der Adressierungsarten

Die folgenden Seiten enthalten Beschreibungen der Adressierungsarten des U 8000. Jede Beschreibung
- erläutert, wie die Operandenadresse berechnet wird,
- zeigt an, in welchem Adressenraum (Register, E/A, Spezial E/A, Datenspeicher, Stackspeicher oder Programmspeicher) sich der Operand befindet,
- zeigt das zur Festlegung der Adressierungsart verwendete Assemblersprachenformat und
- verdeutlicht dies durch ein Beispiel.

Die Beschreibungen sind in zwei Abschnitte unterteilt, und zwar einmal für nichtsegmentierte CPU's und zum anderen für segmentierte CPU's.

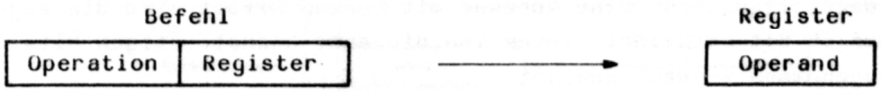
Anwender des U 8002 brauchen sich nur auf den ersten Abschnitt zu konzentrieren; Anwender des U 8001 im unsegmentierten Modus werden ebenfalls auf den ersten Abschnitt verwiesen, während Anwender des U 8001 in segmentierter Betriebsweise auch auf den zweiten Abschnitt Bezug nehmen sollten.

5.4. Beschreibungen und Beispiele (U 8002 und U 8001 in nichtsegmentierter Betriebsart)

Dieser Abschnitt beschreibt die Adressierungsarten sowohl des U 8002 als auch des nichtsegmentierten U 8001.

5.4.1. Register (R)

In der Adressierungsart Register verarbeitet der Befehl Daten, die von einem angegebenen Mehrzweckregister geholt werden. Das Ablegen von Daten in einem Register ermöglicht kürzere Befehle und eine schnellere Ausführung als bei Befehlen, die auf den Speicher zugreifen.



Der Operand ist der Inhalt des Registers.

Der Operand befindet sich stets im Registeradreßraum. Die Registerlänge (Byte, Wort, Langwort oder Doppellangwort) wird durch den Operationskode angegeben.

Assemblersprachenformat:

RHn, RLn	Byteregister
Rn	Wortregister
RRn	Doppelwortregister
RQn	Vierfachwortregister

Beispiel der Adressierungsart R:

```
LD R2, R3      !Lade den Inhalt von!
               !R3 nach R2!
```

Vor Ausführung

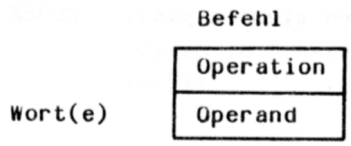
R2	A6B8
R3	9A20

Nach Ausführung

R2	9A20
R3	9A20

5.4.2. Direktwert (IM-immediate)

Diese Adressierungsart ist die einzige, welche keine Register- oder Speicheradresse für den Quelloperanden besitzt. Die vom Befehl verarbeiteten Daten befinden sich im Befehl selbst.



Der Operand ist im Befehl.
Da ein Direktwert-Operand Teil des Befehls ist, befindet er sich stets im Programmspeicheradreßraum. Die Adressierungsart Direktwert wird oft zum Initialisieren von Registern verwendet. Der U 8000 ist für diese Funktion optimiert. Er besitzt verschiedene Direktwert-Befehle im kompakten Format, wodurch die Programmlänge verringert werden kann.

Assemblersprachenformat

Daten

Beispiele der Adressierungsart IM:

```
LDB RH2, # %55          !Lade Hex 55 in RH2!
```

Vor Ausführung

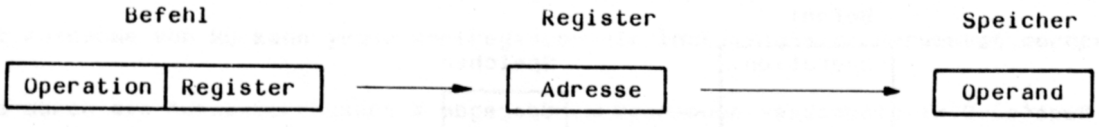
R2 | 6789 |

Nach Ausführung

R2 | 5589 |

5.4.3. Indirektes Register (IR)

In der Adressierungsart "Indirektes Register" sind die verarbeiteten Daten nicht der Wert im angegebenen Register, sondern das Register enthält die Adresse der Daten.



Der Operand ist der Inhalt des Speicherplatzes, dessen Adresse im Register steht.

Zum Ablegen der Adresse dient ein Wortregister. Mit Ausnahme von R0 kann jedes Mehrzweck-Wortregister verwendet werden.
Je nach dem Befehl befindet sich der durch die Adressierungsart "Indirektes Register" angegebene Operand entweder im E/A-Adreßraum (E/A-Befehle), im Spezial-E/A-Adreßraum (Spezial-E/A-Befehle), im Daten- oder Stackadreßraum.

Unter der Voraussetzung, es handelt sich nicht um E/A-Operationen, wird bei Anwendung des Stackpointer (R15) als indirektes Register der Stackbereich angesprochen. Dagegen wird mit den anderen Registern der Datenspeicher adressiert.
Die Adressierungsart "Indirektes Register" kann Platz sparen und die Ausführungszeit verkürzen, wenn auf fortlaufende Speicherzellen zugegriffen werden soll. Diese Art kann auch zur Simulierung komplizierter Adressierungsarten verwendet werden, da die Adressen berechnet werden können, bevor auf die Daten zugegriffen wird.

Assemblersprachenformat

@ Rn

Beispiel der Adressierungsart IR:

```
LD R2, @ R5      !Lade R2 mit den!
                 !Daten, die durch den!
                 !Inhalt von R5 adressiert werden!
```

Vor Ausführung

R2	030F
R3	0005
R4	2000
R5	170C

Speicher

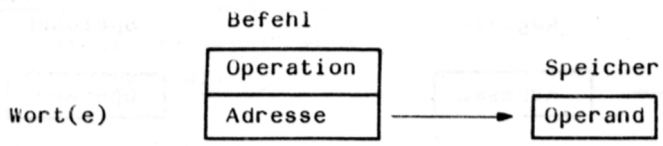
	⋮
170A	A023
170C	0B0E
170E	10D0
	⋮

Nach Ausführung

R2	0B0E
R3	0005
R4	2000
R5	170C

5.4.4. Direkte Adressierung (DA)

In dieser Adressierungsart befinden sich die zu verarbeitenden Daten auf der im Befehl angegebenen Adresse.



Der Operand ist der Inhalt des Speicherplatzes, dessen Adresse sich im Befehl befindet.

Je nach dem Befehl ist der von der direkten Adressierungsart angegebene Operand entweder im E/A-Raum (E/A-Befehle), im Spezial-E/A-Raum (Spezial E/A-Befehle) oder im Datenspeicherraum. Diese Adressierungsart wird auch von Sprungbefehlen (JP) und Unterprogrammaufrufbefehlen (CALL) zur Festlegung der Adresse des nächsten auszuführenden Befehls verwendet. (Tatsächlich dient die Adresse als ein Direktwert, der in den Befehlszähler geladen wird).

Assemblersprachenformat

Adresse **entweder Speicher, E/A oder Spezial-E/A**

Beispiel der Adressierungsart DA

```
LDB RH2, %5E23    !Lade RH2 mit den!
                  !Daten in Adresse!
                  !5E23!
```

Vor Ausführung

R2 | 6789 |

Speicher

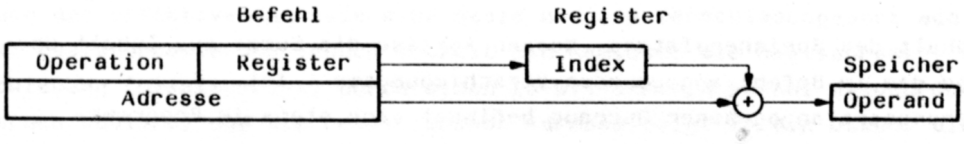
5E22	0106
5E24	0304

Nach Ausführung

R2 | 0689 |

5.4.5. Indiziert (X)

In der indizierten Adressierungsart verarbeitet der Befehl Daten, die sich auf einer indizierten Adresse im Speicher befinden. Die indizierte Adresse wird berechnet durch Addition der im Befehl angegebenen Adresse zu einem in einem Wortregister enthaltenen "Index", welches ebenfalls durch den Befehl festgelegt wird. Indizierte Adressierung gestattet einen beliebigen Zugriff zu Tabellen oder anderen komplexen Datenstrukturen, wo die Adresse der Basis der Tabelle bekannt ist, jedoch der besondere Elementindex durch das Programm berechnet werden muß.



Der Operand ist der Inhalt des Speicherplatzes, dessen Adresse die Summe aus der Adresse im Befehl und dem Inhalt des Registers ist.

Mit Ausnahme von R0 kann jedes Wortregister als Index-Register verwendet werden.

Die durch die Adressierungsart X angegebenen Operanden sind stets im Datenspeicheradreßraum.

Assemblersprachenformat

Adresse (Rn)

Beispiel der Adressierungsart X:

```
LD R4, %231A(R3)        !Lade in R4 den Inhalt des!
                        !Speicherplatzes, dessen!
                        !Adresse 231A + !
                        !dem Wert in R3 ist!
```

Vor Ausführung

R3	01FE
R4	203A

Speicher

2516	F3C2
2518	3D0E
251A	7ADA

Adressenberechnung

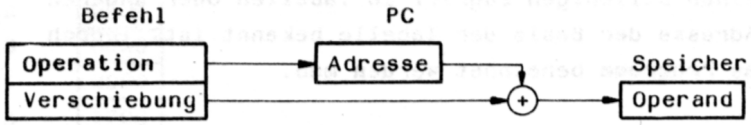
```
    231A
+   01FE
    2518
```

Nach Ausführung

R3	01FE
R4	3D0E

5.4.6. Relative Adresse (RA)

In dieser Adressierungsart sind die zu verarbeitenden Daten relativ zur Adresse des gerade laufenden Befehls lokalisiert. Im Befehl wird eine Verschiebung in Zweierkomplementdarstellung angegeben, die zur Bildung der Zieladresse zum Befehlszähler addiert wird. Der zur Addition verwendete Befehlszählerstand ist die Adresse des dem gerade ausgeführten Befehl folgenden Befehls. (Der Assembler berücksichtigt dies bei der Berechnung der Konstanten, die in den Befehl eingesetzt wird.).



Der Operand ist der Inhalt des Speicherplatzes, dessen Adresse die Summe aus Inhalt des Befehlszählers (PC) und dem im Befehl angegebenen Verschiebungswert ist.
Ein in der RA-Adressierungsart angegebener Operand befindet sich stets im Programmspeicheradreßraum.
Wie die direkte Adressierungsart wird die relative Adressierungsart von bestimmten Programmsteuerbefehlen zur Angabe der Adresse des nächsten auszuführenden Befehls verwendet. (Normalerweise wird der Verschiebungswert mit dem Inhalt des PC addiert und in den PC geladen. Bei den Befehlen DJNZ und CALR dagegen wird der Verschiebungswert vom Inhalt des PC subtrahiert.) Die relative Adressierung gestattet Zugriff sowohl vorwärts als auch rückwärts vom aktuellen Befehlszählerstand aus und wird nur von solchen Befehlen verwendet, welche die normalerweise strenge Grenze zwischen Programm- und Datenspeicher überschreiten können (JR, CALR, LDR).

Assemblersprachenformat

Adresse

Beispiel der Adressierungsart RA:(Zu beachten ist, daß das Symbol "$" für den Wert des aktuellen Befehlszählers verwendet wird.)

```
LDR R2, $+%6                    !Lade in R2 den Inhalt des Speicher-!
                                !platzes, dessen!
                                !Adresse der aktuelle!
                                !Befehlszähler!
                                !+ hex 6 ist!
```

Da der Befehlszähler automatisch erhöht wird, um auf den nächsten Befehl zu zeigen, muß die in dem Befehl auftretende Konstante zur Durchführung der Adressenberechnung in Wirklichkeit +2 sein.

Vor Ausführung

R2	A0F0
PC	0202

Speicher

0202	3102
0204	0002
0206	E801
0208	FFFE

Befehl

Adressenberechnung

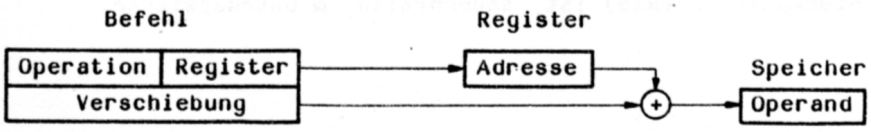
```
   0206
+     2
------
   0208
```

Nach Ausführung

R2	FFFE
PC	0206

5.4.7. Basisadresse (BA)

Diese Adressierungsart ähnelt der indizierten Adressierung insoweit, als zur Erzeugung der effektiven Adresse eine Basis und ein Verschiebungswert addiert werden. Bei der Basisadressierung jedoch enthält ein Register die Basisadresse, während die Verschiebung (Displacement) durch einen 16-Bit-Wert im Befehl ausgedrückt wird. Die zwei werden addiert, und die resultierende Adresse zeigt zu den Daten, die verarbeitet werden sollen. Diese Adressierungsart kann bei den Ladebefehlen verwendet werden. Als Ergänzung zur "Indizierten Adressierung" gestattet die Basisadressierung beliebigen Zugriff zu Tabellen oder anderen Datenstrukturen, wo die Verschiebung eines Elementes innerhalb der Struktur bekannt ist, die Basis der besonderen Struktur jedoch durch das Programm berechnet werden muß.
Mit Ausnahme von R0 kann jedes Wortregister für die Basisadresse verwendet werden. Ein durch die BA-Adressierungsart angegebener Operand befindet sich im Stackbereich, wenn das Basisregister der Stackpointer (R15) ist, andernfalls ist er im Datenspeicherraum.



Der Operand ist der Inhalt des Speicherplatzes, dessen Adresse die Summe aus dem Inhalt des Registers und dem Verschiebungswert im Befehl ist.

Assemblersprachenformat

Rn (# disp)

Beispiel der Adressierungsart BA:

```
LDL R5 (#%18), RR2          !Lade das Langwort!
                            !in RR2 auf den!
                            !Speicherplatz!
                            !dessen Adresse!
                            !der Wert in R5!
                            !+ hex 18 ist!
```

Vor Ausführung

RR2	R2	0A00
	R3	1500
	R4	3100
	R5	20AA

Speicher

	⋮
20C0	0ABE
20C2	F50D
20C4	BADE
20C6	B0D1
	⋮

Adressenberechnung

```
  20AA
+   18
------
  20C2
```

Nach Ausführung

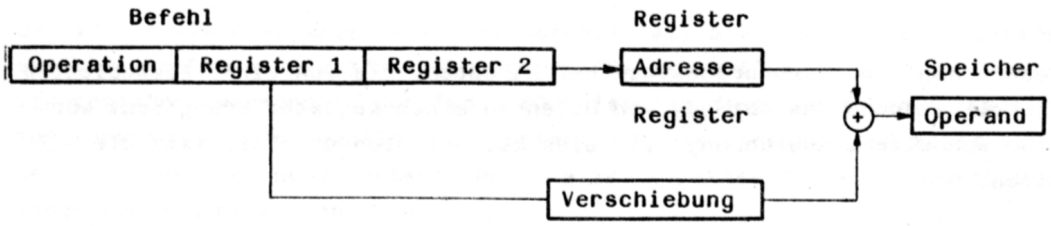
RR2	R2	0A00
	R3	1500
	R4	3100
	R5	20AA

Speicher

	⋮
20C0	0ABE
20C2	0A00
20C4	1500
20C6	B0D1
	⋮

5.4.8. Basis Indiziert (BX)

Diese Adressierungsart ist eine Erweiterung der Basisadressierung und kann von den Ladebefehlen verwendet werden. In diesem Fall wird sowohl die Basisadresse als auch der Index (Verschiebungswert) in Registern gehalten. Diese Adressierungsart gestattet Zugriff zu Speicherplätzen, deren physische Adressen während des Programmlaufs berechnet werden und zum Zeitpunkt der Assemblierung noch nicht bekannt sind.
Mit Ausnahme von R0 kann jedes Wortregister für die Basisadresse oder den Index verwendet werden.
Ein durch die BX-Adressierungsart angegebener Operand befindet sich im Stackbereich, wenn das Basisregister der Stackpointer (R15) ist, andernfalls im Datenspeicher.



Der Operand ist der Inhalt des Speicherplatzes, dessen Adresse die Summe aus dem Inhalt des einen Registers und dem Verschiebungswert im zweiten Register ist.

Assemblersprachenformat

Rn (Rm)

Beispiel von BX:

```
LD R2, R5(R3)          !Lade in R2 den!
                       !Wert dessen Adresse die Summe!
                       !aus dem Wert in R5 und!
                       !dem Wert in R3 ist!
```

Vor Ausführung

R2	1F3A
R3	FFFE
R4	0300
R5	1502

Speicher

	⋮
14FE	0101
1500	BODE
1502	F732
	⋮

Adressenberechnung

```
  1502
+ FFFE
  1500
```

Nach Ausführung

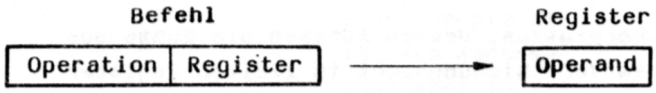
R2	BODE
R3	FFFE
R4	0300
R5	1502

5.5. Beschreibungen und Beispiele (U 8001 segmentiert)

In diesem Abschnitt wird ≪nn≫zur Bezeichnung der Segmentnummer nn verwendet.

5.5.1. Register (R)

In der Adressierungsart Register verarbeitet der Befehl Daten, die von einem festgelegten Mehrzweckregister geholt werden. Das Ablegen von Daten in einem Register ermöglicht kürzere Befehle und eine schnellere Ausführung, als dies mit Befehlen der Fall ist, die auf den Speicher zugreifen.



Der Operand ist der Inhalt des Registers.
Der Operand befindet sich stets im Registeradreßraum. Die Registerlänge (Byte, Wort, Langwort oder Doppellangwort) wird durch den Operationskode angegeben.

Assemblersprachenformate

RHn, RLn	Byteregister
Rn	Wortregister
RRn	Doppelwortregister
RQn	Vierfachwortregister

Beispiel der R-Adressierungsart:

```
LDL RR2, RR4        !Lade den Inhalt!
                    !von RR4 in RR2!
```

Vor Ausführung

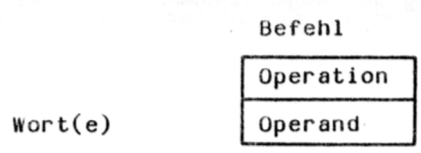
RR2	R2	A6B8
	R3	9A20
RR4	R4	38A6
	R5	745E

Nach Ausführung

RR2	R2	38A6
	R3	745E
RR4	R4	38A6
	R5	745E

5.5.2. Direktwert (IM-immediate)

Die Adressierungsart Direktwert ist die einzige, die keine Register- oder Speicheradresse für den Quelloperanden besitzt. Die vom Befehl verarbeiteten Daten befinden sich im Befehl selbst.



Der Operand ist im Befehl.

Da ein Direktwert-Operand Teil des Befehls ist, befindet er sich stets im Programmspeicheradreßraum. Die Adressierungsart "Direktwert" wird oft zur Initialisierung von Registern verwendet. Der Typ U 8000 ist für diese Funktion optimiert, indem er zur Kürzung der Programmlänge verschiedene kurze Direktwert-Befehle vorsieht. Er besitzt verschiedene Direktwert-Befehle im kompakten Format, wodurch die Programmlänge verringert werden kann.

Assemblersprachenformat

Daten

Beispiel der Adressierungsart IM

```
LDB RH2,    #55              !Lade Hex 55 in RH2!
```

Vor Ausführung

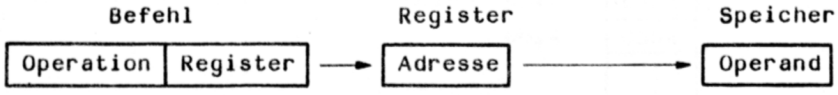
R2 | 6789 |

Nach Ausführung

R2 | 5589 |

5.5.3. Indirektes Register (IR)

In der Adressierungsart "Indirektes Register" sind die verarbeiteten Daten nicht der Wert im angegebenen Register; sondern das Register enthält die Adresse der Daten.



Der Operand ist der Inhalt des Speicherplatzes, dessen Adresse im Register steht. Je nach dem Befehl befindet sich der durch die Adressierungsart "IR" angegebene Operand entweder im E/A-Adreßraum (E/A-Befehle), im Spezial-E/A-Adreßraum (Spezial-E/A-Befehle) oder im Daten- bzw. Stackadreßraum.

Unter der Voraussetzung, es handelt sich nicht um E/A-Operationen, wird bei Anwendung des Stackpointer (RR14) als indirektes Register der Stackbereich angesprochen. Dagegen wird mit den anderen Registern der Datenspeicher adressiert.
Ein 16-Bit-Register dient zum Halten einer E/A- oder Spezial-E/A-Adresse; ein Registerpaar wird zum Halten einer Speicheradresse verwendet. Mit Ausnahme von RO oder RRO kann kann jedes Mehrzweckregister oder Registerpaar verwendet werden.
Die Adressierungsart "Indirektes Register" kann Platz sparen und die Ausführungszeit verkürzen, wenn auf fortlaufende Speicherplätze zugegriffen werden soll. Diese Adressierungsart kann auch zur Simulierung komplizierter Adressierungsarten angewendet werden, da Adressen berechnet werden können, bevor auf die Daten zugegriffen wird.

Assemblersprachenformate

@	Rn	Enthält E/A oder Spezial-E/A-Adresse
@	RRn	Enthält Speicheradresse

Beispiel der Adressierungsart IR (Speicherzugriff)

```
LD R2,  @ RR4                    !Lade in R2 den!
                                 !Wert vom Speicher-!
                                 !platz, der!
                                 !durch den Inhalt von!
                                 !RR4 adressiert wird!
```

Vor Ausführung

RR2	R2	030F
	R3	0005
RR4	R4	2000
	R5	170C

Speicher

	⋮
170A*	A023
170C	0B0E
170E	1003
	⋮

* Segmentnummer 20

Nach Ausführung

RR2	R2	0B0E
	R3	0005
RR4	R4	2000
	R5	170C

Beispiel der Adressierungsart IR (E/A-Zugriff)

```
OUTB @ R1, RL0
```

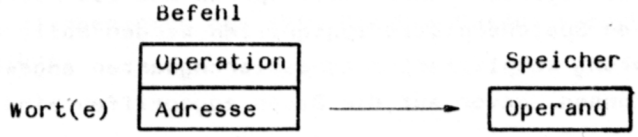
Vor Ausführung

R0	0A23
R1	0011

Bei der Ausführung werden die Daten "23" zu der durch "0011" adressierten E/A-Einheit gesendet.

5.5.4. Direkte Adressierung (DA)

Bei direkter Adressierung befinden sich die zu verarbeitenden Daten auf der im Befehl angegebenen Adresse.



Der Operand ist der Inhalt des Speicherplatzes, dessen Adresse sich im Befehl befindet.
Je nach Befehl befindet sich der durch die direkte Adressierung festgelegte Operand entweder im E/A-Raum (E/A-Befehle), im Spezial-E/A-Raum (Spezial-E/A-Befehle) oder im Datenspeicherraum.
E/A- und Spezial-E/A-Adressen sind ein Wort lang; Speicheradressen können

ein oder zwei Worte lang sein, je nachdem, ob das lange oder kurze Format verwendet wird.
Diese Adressierungsart wird auch von Sprung- und Unterprogrammaufrufbefehlen (Jump und Call) zur Angabe der Adresse des nächsten auszuführenden Befehls verwendet. (Tatsächlich dient die Adresse als Direktwert, der in den Befehlszähler geladen wird.)

Assemblersprachenformat

Adresse — entweder Speicher, E/A oder Spezial-E/A, wobei doppelte Winkelklammern "<<" und ">>" die Segmentnummer einschließen und Vertikallinien "|" und "|" Speicheradressen in der ..urzform einschließen.

Beispiel der Direkten Adressierung:

```
LDB RH2,  |≪15≫ %23|     !Lade RH2 mit dem!
                          !Wert im Speicher!
                          !Segment 15, Ver-
                          !schiebung 23 (hex)!
```

Vor Ausführung

R2 | 6789 |

Speicher

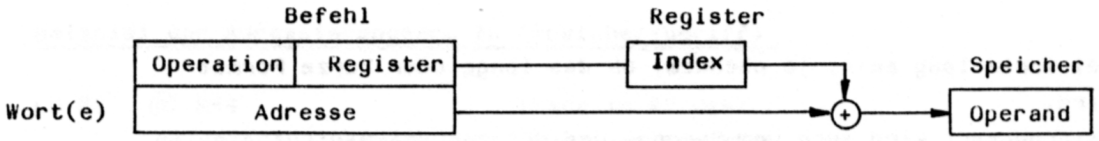
	⋮
≪15≫ 0022	0206
0024	0304
	⋮

Nach Ausführung

R2 | 0689 |

5.5.5. Indiziert (X)

Bei dieser Adressierungsart befinden sich die zu verarbeitenden Daten an einer indizierten Adresse im Speicher. Die indizierte Adresse wird berechnet durch Addition der im Befehl angegebenen Adresse zu einem "Index", der in einem Wortregister enthalten ist, das ebenfalls durch den Befehl festgelegt wird. Der Offset der Operandenadresse wird berechnet, indem der 16-Bit-Indexwert zu dem 8- oder 16-Bit-Offsetteil der Adresse im Befehl addiert wird. Die Segmentnummer der Operandenadresse kommt direkt vom Befehl. (Ein Überlauf wird ignoriert - er setzt weder das Überlauf-Flag noch erhöht er die Segmentnummer.) Die indizierte Adressierung gestattet beliebigen Zugriff zu Tabellen oder anderen komplexen Datenstrukturen, wo die Adresse der Basis der Tabelle bekannt ist, der besondere Elementindex jedoch durch das Programm berechnet werden muß.



Der Operand ist der Inhalt des Speicherplatzes, dessen Adresse die Adresse im Befehl ist, versetzt durch den Inhalt des Registers.
Mit Ausnahme von R0 kann jedes Wortregister als Indexregister verwendet werden.
Die Adresse im Befehl kann ein oder zwei Worte betragen, je nachdem, ob ein langes oder kurzes Offset in der Adresse verwendet wird.
Die durch die Adressierungsart X festgelegten Operanden sind stets im Datenspeicheradreßraum.

Assemblersprachenformat:

Adresse (Rn)

Beispiel der Adressierungsart X:

```
LD R4,    ≪5≫   %231A(R3)    !Lade in R4 den!
                               !Inhalt des!
                               !Speicherplatzes!
                               !dessen Adresse!
                               !Segment 5 ist mit dem Offset  %231A!
                               !verschoben um den Wert!
                               !in R3!
```

Vor Ausführung

R3	01FE
R4	203A

Speicher

≪5≫

	⋮
2516	F3C2
2518	3D0E
251A	7ADA
	⋮

Adressenberechnung

```
≪5≫    %231A
  +    %01FE
-------------
≪5≫    %2518
```

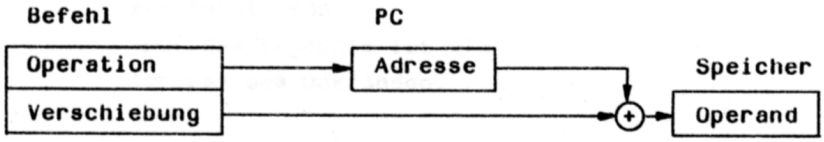
Nach Ausführung

R3	01FE
R4	3D0E

5.5.6. Relative Adresse (RA)

In der relativen Adressierungsart sind die zu verarbeitenden Daten relativ zur Adresse des gerade laufenden Befehls lokalisiert. Im Befehl wird eine Verschiebung in Zweierkomplementdarstellung angegeben, die zur Bildung der Zieladresse zum Offset des Befehlszählers addiert wird. Der zur Addition verwendete Befehlszählerstand ist die Adresse des dem gerade ausgeführten Befehl folgenden Befehls. (Der Assembler berücksichtigt

dies bei der Berechnung der Konstanten, die in den Befehl eingesetzt wird.)



Der Operand ist der Inhalt des Speicherplatzes, dessen Adresse die Summe aus Inhalt des Befehlszählers (PC) und dem im Befehl angegebenen Verschiebungswert ist.

Ein durch die Adressierungsart RA angegebener Operand befindet sich stets im Programmspeicheradreßraum. Es können Adressen im Lang- oder Kurzformat verwendet werden.
Wie bei direkter Adressierung wird auch die relative Adressierung von bestimmten Programmsteuerbefehlen zur Angabe der Adresse des nächsten auszuführenden Befehls verwendet. (Normalerweise wird der Verschiebungswert mit dem Inhalt des PC addiert und in den PC geladen. Bei den Befehlen DJNZ und CALR dagegen wird der Verschiebungswert vom Inhalt des PC subtrahiert.)
Die relative Adressierung ermöglicht Zugriffe sowohl vorwärts als auch rückwärts vom aktuellen Befehlszählerstand aus und wird nur von Sprüngen, Aufrufen und relativen Ladeoperationen (LDR) verwendet. Zu beachten ist, daß die Segmentnummer unverändert bleibt und sich damit die relative Adresse im gleichen Segment wie der Befehl befindet.

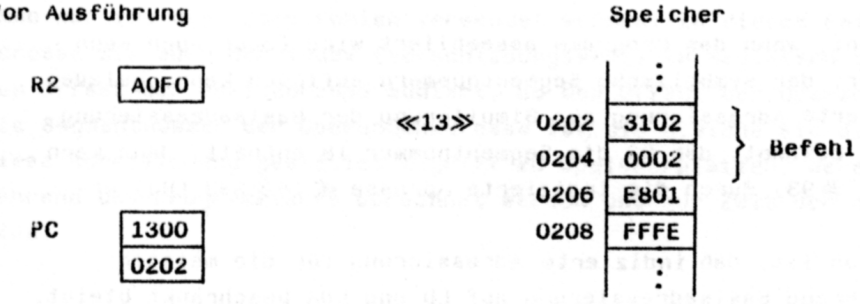
Assemblersprachenformat

Adresse

Beispiel der Adressierungsart RA

```
LDR, $ + 6          !Lade in R2 den In-!
                    !halt des Speicher-!
                    !platzes, dessen!
                    !Adresse der!
                    !aktuelle Befehls-!
                    !zähler +6 ist!
```

Da der Befehlszähler automatisch erhöht wird, um auf den nächsten Befehl zu zeigen, muß die in dem Befehl auftretende Konstante zur Durchführung der Adressenberechnung in Wirklichkeit +2 sein.



Adressenberechnung

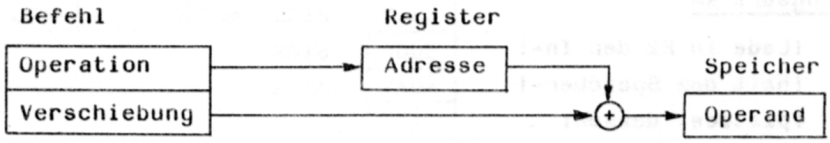
≪13≫	0206
+	2
≪13≫	0208

Nach Ausführung

R2	FFFE
PC	1300
	0206

5.5.7. Basisadresse (BA)

Die Basisadressierung ist der indizierten Adressierung ähnlich, indem zur Erzeugung der effektiven Adresse eine Basis und ein Verschiebungswert addiert werden. Bei der Basisadressierung enthält ein Registerpaar die segmentierte 23-Bit-Basisadresse, während der Verschiebungswert als 16-Bit-Wert im Befehl ausgedrückt wird. Der Verschiebungswert wird zum Offset der Basisadresse hinzuaddiert, und die resultierende Adresse zeigt auf die Daten, die verarbeitet werden sollen. (Die Segmentnummer wird nicht geändert).
Diese Adressierungsart kann nur mit den Lade-Befehlen verwendet werden. Als Ergänzung der indizierten Adressierungsart gestattet die Basisadresse beliebigen Zugriff zu Blöcken oder anderen Datenstrukturen, wo die Verschiebung eines Elementes innerhalb der Struktur bekannt ist, die Basis der besonderen Struktur jedoch durch das Programm berechnet werden muß.



Der Operand ist der Inhalt des Speicherplatzes, dessen Adresse die Summe aus dem Inhalt des Registers und dem Verschiebungswert im Befehl ist.

Mit Ausnahme von RR0 kann jedes Doppelwort-Register für die Basisadresse verwendet werden. Die Basisadressierung gestattet Zugriff zu Speicherplätzen, deren Segmentnummer zur Zeit der Assemblierung nicht bekannt ist.
Ein durch die Basisadressierung festgelegter Operand befindet sich im Stackbereich, wenn das Basisregister der Stackpointer (RR14) ist, andernfalls ist er im Datenspeicherraum.
Ist die Segmentnummer bekannt, wenn das Programm assembliert wird (oder auch wenn das Programm mit einem Lader, der symbolische Segmentnummern auflösen kann, geladen wird), dann kann die indizierte Adressierung zur Simulierung der Basisadressierung verwendet werden. Ist z. B. bekannt, daß R2 die Segmentnummer 18 enthält, dann kann die Basisadressierung RR2 (#93) durch die indizierte Adresse ≪18≫%93 (R3) ersetzt werden.
Der Vorteil dieser Simulation ist, daß indizierte Adressierung für die meisten Operationen möglich ist, während Basisadressierung auf LD und LDA beschränkt bleibt. Außerdem ist die indizierte Adressierung schneller und führt zu einem kompakten Kode.

Assemblersprachenformat:

RRn (#disp) Addiere den Verschiebungswert zum Inhalt von RRn ;das Ergebnis ist die Adresse des Operanden.

Beispiel der Basisadressierung:

```
LDL RR4 (#%18), RR2     !Lade das Langwort!
                        !in RR2 in den!
                        !Speicherplatz!
                        !dessen Adresse die Summe aus!
                        !dem Inhalt von RR4 und dem Verschiebungswert!
                        !%18 ist!
```

Vor Ausführung

RR2	R2	0A00
	R3	1500
RR4	R4	3100
	R5	20AA

≪31≫

20C0	0ABE
20C2	F50D
20C4	BADE
20C6	B0D1

Speicher

Adressenberechnung

≪31≫	20AA
+	18
≪31≫	20C2

Nach Ausführung

RR2	R2	0A00
	R3	1500
RR4	R4	3100
	R5	20AA

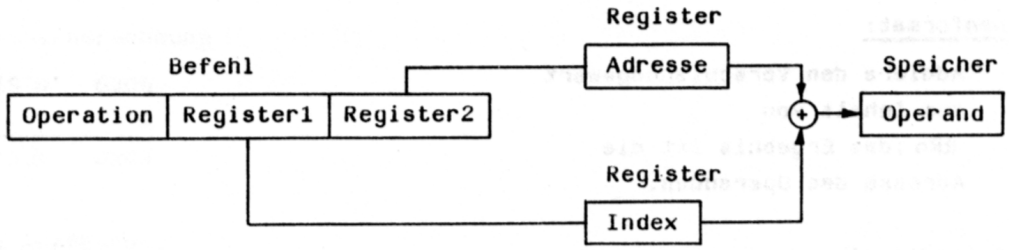
≪31≫

20C0	0ABE
20C2	0A00
20C4	1500
20C6	B0D1

Speicher

5.5.8. Basis Indiziert (BX)

Die Adressierungsart Basis Indiziert ist eine Erweiterung der Basisadressierung und kann nur von den Ladebefehlen verwendet werden. In diesem Fall wird sowohl die Basisadresse als auch der Index (Verschiebungswert) in Registern gehalten. Der Index wird zum Offset der Basisadresse addiert, um den Offset der Operandenadresse zu erzeugen. Die Segmentnummer der Operandenadresse ist die gleiche wie die der Basisadresse. Diese Adressierung gestattet Zugriff zu Speicherplätzen, deren physische Adressen während des Programmlaufs berechnet werden und zur Zeit der Assemblierung nicht bekannt sind.



Der Operand ist der Inhalt des Speicherplatzes, dessen Adresse die Summe aus dem Inhalt eines Registerpaares und dem Verschiebungswert im zweiten Register ist.
Mit Ausnahme von RR0 kann jedes Registerpaar für die Basisadresse verwendet werden. Jedes Wortregister mit Ausnahme von R0 kann für den Index verwendet werden.
Zu beachten ist, daß das Kurzoffsetformat für Basisadressen in Registern unzulässig ist.
Ist das Basisregister der Stackpointer (RR14), so befindet sich ein durch die Adressierungsart BX angegebener Operand im Stackbereich, andernfalls ist er im Datenspeicher.

Assemblersprachenformat:

RRn (Rm)

Beispiel der Adressierungsart BX:

```
LD R2, RR4 (R3)          !Lade in R2 den Wert!
                         !dessen Adresse die Summe aus dem!
                         !Inhalt von RR4 und!
                         !dem Inhalt von R3 ist!
```

Vor Ausführung

RR2	R2	3535
	R3	FFFE
RR4	R4	1300
	R5	1502

		Speicher
≪13≫	14FE	0101
	1500	BODE
	1502	F732

Adressenberechnung

≪13≫	1502
+	FFFE
≪13≫	1500

Nach Ausführung

RR2	R2	BODE
	R3	FFFE
RR4	R4	1300
	R5	1502

		Speicher
≪13≫	14FE	0101
	1500	BODE
	1502	F732

Kapitel 6

6. Ausnahmen

6.1. Einleitung

Die CPU U 8000 ermöglicht drei Arten von Ausnahmen, d. h. Bedingungen, die den normalen Ablauf der Programmausführung ändern:

- Interrupts
- Traps
- Reset

Interrupts sind asynchrone Ereignisse, die typischerweise durch periphere Geräte ausgelöst werden, welche Bedienung anfordern. Sie verursachen, daß der Prozessor seine gegenwärtige Programmausführung zeitweilig unterbricht, um das anfordernde Gerät bedienen zu können.
Traps sind synchrone Ereignisse und stellen die Reaktion der CPU auf bestimmte Ereignisse dar, die während der versuchten Ausführung eines Befehls festgestellt werden. Somit ist in der Herkunft der Hauptunterschied zwischen Traps und Interrupts zu sehen. Eine Trapbedingung ist stets durch Wiederausführung des Programms, welches den Trap verursachte, reproduzierbar, während ein Interrupt im allgemeinen von der aktuell ausgeführten Aufgabe unabhängig ist.
Reset hat gegenüber anderen Bedingungen einschließlich aller Interrupts und Traps höchste Priorität. Reset tritt auf, wenn die $\overline{\text{RESET}}$-Leitung aktiviert wird und verursacht die Initialisierung bestimmter Steuerregister. Die Arbeitsweise der CPU U 8000 bei der Reaktion auf Interrupt, Trap oder Reset ist ähnlich, deshalb werden diese im folgenden Abschnitt gemeinsam behandelt.

6.2. Interrupts

An drei verschiedene Pins der CPU U 8000 können drei Arten von Interrupts aktiviert werden. Die Behandlung der einzelnen Interrupts wird in Abschnitt 6.6. beschrieben.

6.2.1. Nichtmaskierbarer Interrupt ($\overline{\text{NMI}}$)

Diese Art von Interrupt kann durch Software nicht gesperrt (maskiert) werden. Sie ist typischerweise für externe Ereignisse höchster Priorität reserviert, die sofortige Bedienung erfordern.

6.2.2. Vektorisierter Interrupt ($\overline{\text{VI}}$)

Ein Ergebnis eines Interrupts oder Traps ist es, daß ein 16-Bit-Kennwort (Identifier) auf dem Systemstack abgelegt wird (siehe Abschnitt 6.6.2.). Dieses Wort kann dazu verwendet werden, die Quelle des Interrupts oder Traps festzustellen. Bei vektorisierten Interrupts wird dieses Kennwort von der CPU-Hardware als Zeiger zur Auswahl einer besonderen Interrupt-Serviceroutine verwendet. Die Bearbeitung vektorisierter Interrupts ist somit erheblich schneller, als es der Fall wäre, wenn ein allgemeines Interruptbehandlungsprogramm das Kennwort prüft und erst nach dessen Auswertung in die jeweilige Service-Routine eintreten würde. Vektorisierte Interrupts können durch Software gesperrt werden.

6.2.3. Nichtvektorisierter Interrupt (NVI)

Diese Interrupts legen ebenfalls ein Kennwort auf den Systemstack ab. Die CPU verwendet jedoch das Kennwort nicht als Vektor zur Auswahl einer Service-Routine. Alle nichtvektorisierten Interrupts werden durch die gleiche Routine bedient, in der eine Auswertung des Kennworts erfolgen kann. Sie können durch Software gesperrt werden.

6.3. Traps

Die CPU's U 8001 und U 8002 ermöglichen drei Arten von intern erzeugten Traps. Die U 8001 ermöglicht einen vierten Trap, der extern (jedoch synchron) durch die MMU (Speicherverwaltungsbaustein) erzeugt wird. Ein Trap kann nicht gesperrt werden und tritt deshalb immer dann auf, wenn die für ihn definierten Bedingungen vorhanden sind. (Die Trapbehandlung wird im Abschnitt 6.6. beschrieben.)

6.3.1. Trap bei erweitertem Befehl

Dieser Trap tritt auf, wenn die CPU auf einen erweiterten Befehl trifft (siehe "CPU U 8001/U 8002 - Befehlsbeschreibung") und das EPA-Bit im FCW Null ist. Dieser Trap gestattet bei Systemen, die über keine EPU verfügen, die Simulation der EPU-Funktion durch Software oder das Abbrechen des Programms.

6.3.2. Trap bei privilegiertem Befehl

Dieser Trap tritt auf, wenn versucht wird, einen privilegierten Befehl auszuführen, während sich die CPU im Normalmodus befindet ($S/\overline{N}$-Bit im FCW ist Null). Dieser Trap gestattet dem Betriebssystem, Operationen festzustellen und zu verhindern, die die Integrität des Systems zerstören könnten (z. B. E/A-Befehl).

6.3.3. Trap bei System-Call

Dieser Trap tritt auf, wenn ein System-Call-Befehl (SC) ausgeführt wird. Er gestattet einen kontrollierten Übergang vom Normalmodus in den Systemmodus.

6.3.4. Segment-Trap

Ohne Rücksicht auf den Status des Segmentbits im FCW tritt dieser Trap auf, wenn die $\overline{SEGT}$-Leitung des U 8001 aktiviert wird. Dieser Trap wird durch externe Speicherverwaltungs-Hardware, wie z. B. den Speicherverwaltungsbaustein U 8010 erzeugt und ist das Ergebnis der Feststellung einer Speicherzugriffsverletzung wie z. B. eines Offsets, der größer ist als die zugeordnete Segmentlänge, oder einer Schreibwarnung (ein Schreiben in die untersten 256 Bytes eines Stacks).

6.4. Reset

Bei Aktivierung der $\overline{RESET}$-Leitung wird Reset ausgelöst. Es wird in jedem Taktzyklus akzeptiert und initialisiert ausgewählte Steuerregister der CPU auf Werte, die spezifiziert werden können. Reset hat gegenüber anderen Bedingungen einschließlich Interrupts, Traps, Bus- oder Stoprequest (Bus- oder Stopanforderung) Vorrang. Reset ist zur Initialisierung des Systems bei Anlegen der Betriebsspannung zu verwenden. Innerhalb von fünf Taktzyklen nach dem Aktivieren von $\overline{RESET}$ sind AD_0 bis AD_{15} im Tri-State. $\overline{AS}$, $\overline{DS}$, $\overline{MREQ}$, $\overline{BUSACK}$, $\overline{MO}$ und ST_0 bis ST_3 werden auf High und SN_0 bis SN_6 auf Low gebracht.

Der Zustand der R/$\overline{W}$-, B/$\overline{W}$- und N/$\overline{S}$-Leitungen ist undefiniert. Zum korrekten Rücksetzen der CPU muß $\overline{RESET}$ mindestens fünf Taktzyklen lang auf Low gehalten werden. Drei Taktzyklen nach Rückkehr von $\overline{RESET}$ auf High werden zur Initialisierung der Programmstatusregister fortlaufende Speicherlesezyklen im Befehlsspeicheradreßraum im Systemmodus ausgeführt. Beim U 8001 geschieht folgendes im Segment 0. Vom Speicherplatz 0002 wird im ersten Zyklus das Flag- und Steuerwort gelesen, im nächsten vom Speicherplatz 0004 die 7-Bit-Segmentnummer für den Befehlszähler und im darauffolgenden vom Speicherplatz 0006 der 16-Bit-Offset. Der U 8002 holt sich von der Speicherzelle 0002 das Flag- und Steuerwort und von der 0004 den Wert für den PC. Bei beiden Prozessoren erfolgt der Programmstart auf der im PC spezifizierten Adresse mit der ersten Befehlsholeoperation.

6.5. Sperren von Interrupts

Vektorisierte und nichtvektorisierte Interrupts können von einander unabhängig über Software durch Setzen oder Löschen entsprechender Steuerbits im Flag- und Steuerwort (FCW) freigegeben oder gesperrt werden, denn die zwei Steuerbits VIE und NVIE im FCW steuern die maskierbaren Interrupts. Ist VIE = 1, so werden vektorisierte Interrupts freigegeben. Wenn NVIE = 1 ist, werden nichtvektorisierte Interrupts freigegeben. Beide Flags können zusammen oder getrennt gesetzt oder gelöscht werden. Außerdem werden diese Steuerbits beeinflußt, wenn ein neues FCW geladen wird. Dies geschieht auch bei der Akzeptierung eines Interrupts oder Traps. Durch den IRET-Befehl wird der Zustand der Interruptsteuerbits wieder hergestellt, der vor dem Interrupt oder Trap existierte. Wurde eine Interruptart gesperrt, so ignoriert die CPU Interruptanforderungen an dem entsprechenden Eingangspin. Da maskierbare Interruptanforderungen von der CPU nicht gespeichert werden, muß das Anforderungssignal solange geltend gemacht werden, bis die CPU die Anforderung bestätigt.

6.6. Interrupt- und Trapbehandlung

Die Reaktion der CPU auf einen Trap oder eine Interruptanforderung erfolgt in fünf Schritten:

- Bestätigung der externen Anforderung (für Interrupts und Segmenttraps)
- Retten der alten Programmstatusinformation
- Laden eines neuen Programmstatus
- Ausführen der Serviceroutine
- Rückkehr zu der unterbrochenen Aufgabe

6.6.1. Bestätigungszyklus

Ein Bestätigungszyklus ist nur für extern erzeugte Anforderungen erforderlich. Wie auch in Kapitel 8 beschrieben, ist der Hauptzweck eines solchen Zyklus der Empfang eines 16-Bit-Kennwortes (Identifier) von der externen Einheit, das mit dem alten Programmstatus gerettet wird.
Vor dem Bestätigungszyklus tritt die CPU in den Systemmodus (außerdem segmentierte Betriebsart bei U 8001) ein. Die N/$\overline{S}$-Leitung zeigt an, daß ein Übergang zum Systemmodus stattfand. Die CPU verbleibt in diesem Modus, bis sie mit der Ausführung der Ausnahme-Serviceroutine beginnt. Dann wird ihr Modus vom FCW bestimmt.

6.6.2. Statusretten

Die alte Programmstatusinformation wird gerettet, indem sie in folgender Reihenfolge auf den Systemstack abgelegt wird:

- **Befehlszähler (PC), (16 Bit für U 8002; 16-Bit-Offset und 7-Bit-Segmentnummer für U 8001)**
- **Flag- und Steuerwort (FCW)**
- **Interrupt/Trap-Kennwort (Identifier)**

Das Kennwort enthält den Grund oder die Quelle für den Trap oder Interrupt. Für interne Traps ist das Kennwort das erste Wort des Befehls, bei dem der Trap erfolgte. Für den Segmenttrap oder Interrupts ist das Kennwort der Wert auf dem Datenbus, der von der CPU während des Interrupt-Acknowledge- oder Trap-Acknowledge-Zyklus (Annahme-, Bestätigungszyklus) gelesen wird. Das Format des in dem Systemstack geretteten Programmstatus ist in Bild 6.1 dargestellt.

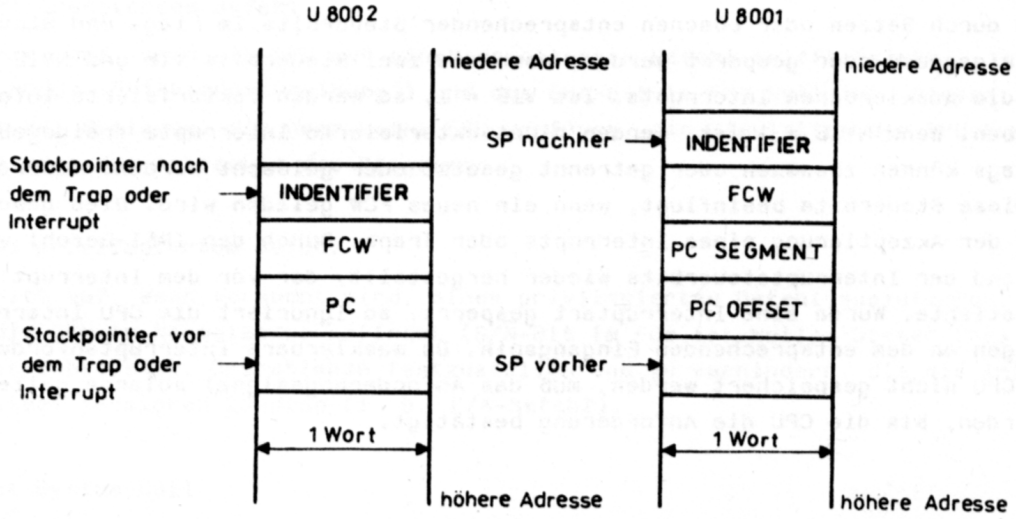


Bild 6.1 Anordnung des geretteten Programmzustandes im Systemstack

Tabelle 6.1 zeigt den PC-Wert, der für jeden Interrupt- oder Traptyp auf den Stack gerettet wird.

Ausnahme:	PC-Wert ist Adresse des:
Trap bei erweitertem Befehl	nächsten Befehls
Trap bei privilegiertem Befehl - privilegierter Einwortbefehl - privilegierter Mehrwortbefehl	 nächsten Befehls zweiten Wortes des privilegierten Befehls
Trap bei System Call	nächsten Befehls
Segment Trap	nächsten Befehls[1,2]
alle Interrupts	nächsten Befehls[2]

[1] unter der Annahme des erfolgreichen Abschlusses der Befehlsholeoperation

[2] wird ein Befehl ausgeführt, der durch Interrupt unterbrochen werden kann (z. B. LDIR), dann ist der gerettete PC-Wert die Adresse des aktuellen Befehls.

6.6.3. Laden des neuen Programmstatus

Nach Retten des aktuellen Programmstatus (PC und FCW) wird der neue Programmstatus automatisch von dem Programmstatusbereich (PSA) im Systemprogrammspeicher geladen. Die besonderen, vom Programmstatusbereich geholten Statusworte sind abhängig von dem Trap- oder Interrupttyp und beim vektorisierten Interrupt von dem Interruptvektor. Bild 6.2 zeigt das Format des Programmstatusbereiches. Mit Ausnahme der vektorisierten Interrupts existiert für jeden Interrupt oder Trap ein eigener Programmstatusblock, welcher automatisch in die Programmstatusregister geladen wird. Zu beachten ist, daß die Größe eines jeden Programmstatusblocks von der U 8000-Version abhängt (zwei Worte bei der nichtsegmentierten und vier Worte bei der segmentierten CPU).
Für alle vektorisierten Interrupts wird das gleiche Flag- und Steuerwort (FCW) vom entsprechenden Programmstatusblock geladen. Der entsprechende Befehlszählerwert (PC) wird aus der nachfolgenden Tabelle ausgewählt, die beim U 8002 bis zu 256 und beim U 8001 bis zu 128 Startadressen für die Interruptserviceroutinen enthalten kann. Die Auswahl geschieht in folgender Weise:

- Die acht niederwertigen Bits des durch das unterbrechende Gerät auf den Datenbus gegebenen Kennwortes werden mit zwei multipliziert.
- Dieser Wert wird zur Adresse des Wortes addiert, das dem FCW für vektorisierte Interrupts folgt.
- Das Ergebnis ist die Adresse, auf der die Startadresse der entsprechenden Interruptserviceroutine abgelegt ist.
- Diese Startadresse wird in den PC geladen.

Beim U 8002 bestimmt damit das Kennwort 0 den ersten PC-Wert, das Kennwort 1 den zweiten PC-Wert usw. bis zum Kennwort 255, das den 256. PC-Wert auswählt. Beim U 8001 wählt das Kennwort 0 den ersten PC-Wert, das Kennwort 2 den zweiten PC-Wert usw. bis zum Kennwort 254, das den 128. PC-Wert auswählt.
Der Programmstatusbereich wird durch das spezielle Steuerregister PSAP (Program Status Area Pointer; Programmstatusbereichszeiger) adressiert. Dieser Zeiger ist beim nicht segmentierten U 8002 ein Wort. Dagegen besteht er beim segmentierten U 8001 aus zwei Worten. Wie in Bild 6.2 dargestellt, enthält der Zeiger eine Segmentnummer (nur U 8001) sowie das höherwertige Byte einer 16-Bit-Offsetadresse. Beim niederwertigen Byte wird angenommen, daß es Nullen enthält. Deshalb muß der Programmstatusbereich bei einer 256-Byte-Adressengrenze beginnen. Der Programmierer hat die Möglichkeit, mit dem LDCTL-Befehl auf den PSAP zuzugreifen.

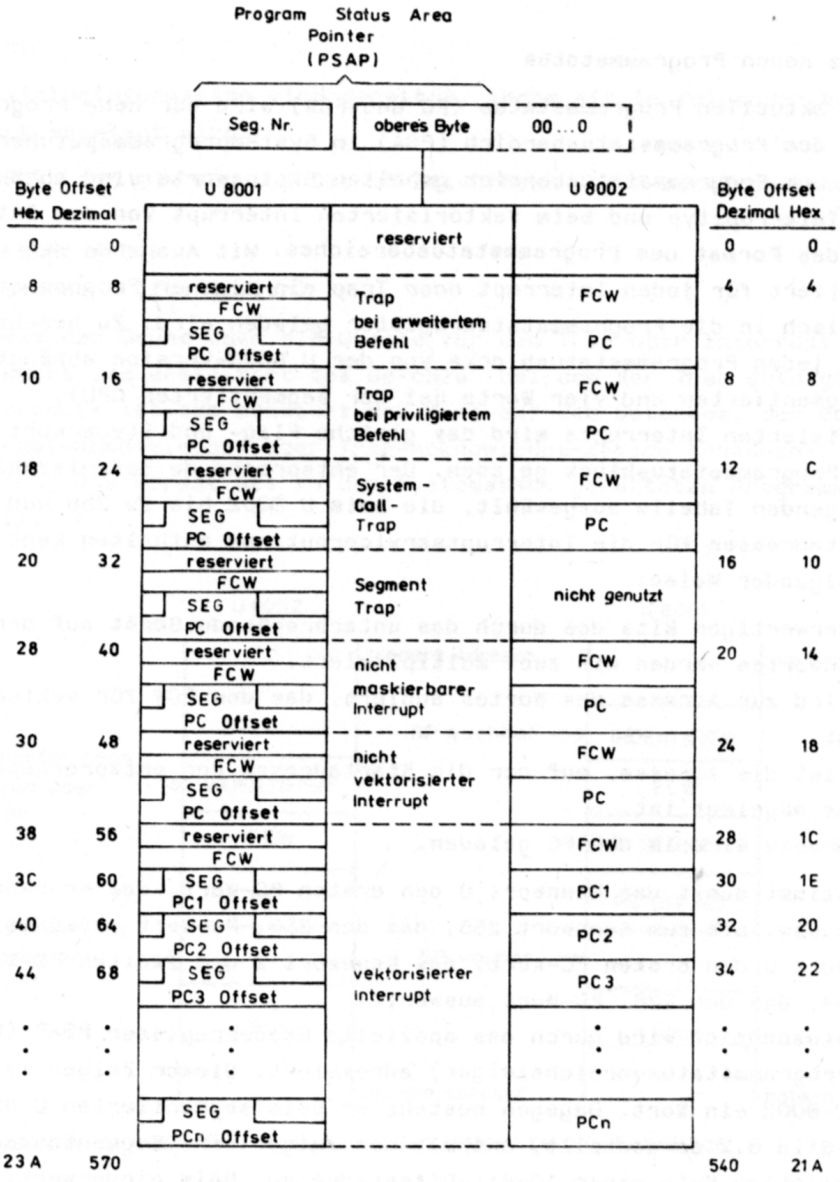


Bild 6.2 Program Status Area

6.6.4. Ausführung der Serviceroutine

Wie vorangehend dargestellt, setzt das Laden des neuen Programmstatus automatisch den Programmzähler auf die Startadresse der Serviceroutine des Interrupts oder Traps. Diese Routine wird jetzt ausgeführt. Da ein neues FCW geladen wurde, können die maskierbaren Interrupts in der Serviceroutine durch entsprechende Wahl des FCW gesperrt werden. Dies erlaubt das Retten wichtiger Informationen, bevor nachfolgende Interrupts akzeptiert werden. Serviceroutinen, die vor ihrer Beendigung Interrupts wieder freigeben, gestatten die Verschachtelung von Interruptroutinen.

6.6.5. Rückkehr von einem Interrupt oder Trap

Zur Beendigung einer Interrupt- oder Trapserviceroutine kann ein IRET-Befehl ausgeführt werden. Damit wird das Programm dort fortgesetzt, wo es durch den Interrupt bzw. Trap abgebrochen wurde. Durch IRET wird der Zustand des FCW und PC, der beim Abbruch existierte, wieder hergestellt. Das während des Bestätigungszyklus auf dem Stack abgelegte Kennwort wird hierbei ignoriert. Dagegen werden die anderen Informationen, die beim Statusretten in den Systemstack abgelegt wurden, in FCW und PC geladen. Danach findet ein Befehlsholezyklus auf der im PC wiederhergestellten Adresse

statt, wobei das regenerierte FCW wirksam wird.
Bei der CPU U 8001 muß der IRET-Befehl im segmentierten Modus ausgeführt werden. Im nichtsegmentierten Modus wäre die Operation nicht definiert.

6.7. Priorität

Da verschiedene Ausnahmen gleichzeitig auftreten können, verwendet die CPU ein Prioritätsschema, um zu entscheiden, welchem Ereignis zuerst entsprochen wird.
Die Priorität der Ausnahmen ist im folgenden in absteigender Ordnung aufgeführt:

1. Reset
2. Interner Trap (bei privilegiertem Befehl, Systemaufruf, erweitertem Befehl)
3. Nichtmaskierbarer Interrupt
4. Segment-Trap (nur U 8001)
5. Vektorisierter Interrupt
6. Nichtvektorisierter Interrupt

Das Prioritätssystem arbeitet wie folgt:

- Reset hat höchste Priorität und wird immer akzeptiert.
- Falls mehrere andere Ausnahmen gleichzeitig auftreten, wird diejenige bestätigt, die die höchste Priorität besitzt und freigegeben ist (Traps und nichtmaskierbare Interrupts sind stets freigegeben). Der alte Status wird gerettet und der neue Status geladen. Der neue Status besteht aus der Startadresse der Serviceroutine (PC) und einem neuen FCW, welches vektorisierte und nichtvektorisierte Interrupts sperren kann.
- Liegen noch weitere freigegebene Ausnahmen an, so wird wiederum die mit der höchsten Priorität bestätigt. Der alte Status wird gerettet und der neue Status geladen. In diesem Fall ist zu beachten, daß der alte Status der PC und das FCW der Serviceroutine der ersten Ausnahme ist.
- Dieser Prozeß wird wiederholt, bis keine weitere freigegebene Ausnahme anliegt. Dann enthält der aktuelle PC und das FCW die Statuswerte für die bestätigte Ausnahme mit niedrigster Priorität.
- Die Ausführung der Serviceroutinen geht jetzt in umgekehrter Prioritätsordnung vor sich, das heißt, die Ausnahme mit niedrigster Priorität wird zuerst bedient.
- Nachdem alle Ausnahmen bedient wurden, wird der Ausgangszustand wieder hergestellt und die Ausführung dort fortgesetzt.

Innerhalb jeder der o. a. Klassen kann es mehrere Quellen geben. Bei den internen Traps kann immer nur einer auftreten und es ist deshalb keine Prioritätsauflösung innerhalb dieser Klasse erforderlich. Die anderen Typen entstehen aus externen Quellen. Wenn sich mehrere Geräte die gleiche Anforderungsleitung teilen, besteht die Möglichkeit, daß mehrere Geräte gleichzeitig Bedienung durch die CPU anfordern.
Entweder müssen mehrere Quellen einer Klasse gleichzeitig bedient werden (wie bei den Speicherverwaltungsbausteinen) oder rivalisierende Anforderungen müssen extern aufgelöst werden, zum Beispiel mit Hilfe des Daisy-chain-Verfahrens (Prioritätenkette) oder eines Interrupt-Prioritäts-Controllers.
Diese Auflösung geschieht während des Interrupt-Acknowledgezyklus.

Kapitel 7

7. Refresh

7.1. Einleitung

Die CPU U 8000 besitzt einen internen Mechanismus zum Auffrischen dynamischer Speicher (Refresh). Dieser Mechanismus kann auf zwei Arten aktiviert werden:

- Wird das Refresh-Enable-Bit (RE) im Refreshzähler der CPU auf eins gesetzt (Bild 7.1), so werden periodisch Speicherrefreshzyklen eingeschoben. Der Abstand zwischen zwei Refresh-Zyklen wird im RATE-Feld des Refreshzählers festgelegt (siehe Abschnitt 7.3. Periodisches Refresh).
- Wird die STOP-Leitung aktiviert, so erzeugt die CPU ständig Speicher-Refresh-Zyklen (siehe Abschnitt 7.4. Refresh im Stop-Zustand).

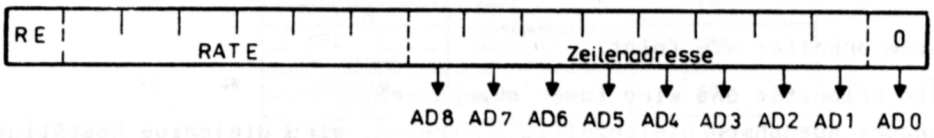


Bild 7.1 Refresh-Steuerregister, periodisches

7.2. Refreshzyklen

Der **Refreshmechanismus** bietet die Möglichkeit zur Erzeugung eines besonderen Zugriffs. Dieser wird Refreshzyklus genannt und ist in Abschnitt 8. beschrieben. Ein Refreshzyklus ist drei Taktzyklen lang und kann unmittelbar an den letzten Taktzyklus eines beliebigen Zugriffs angehängt werden. Während eines Refreshzyklus werden die Statusleitungen auf 0001 gesetzt. Die Adreßleitungen AD_1 - AD_8 führen den Wert des Zeilenzählers. Dieser Wert bestimmt, welche Zeile des Speichers im laufenden Zyklus aufgefrischt werden soll. Da der Speicher wortorganisiert ist, ist AD_0 stets Null. Der Zustand der Adreßleitungen AD_9 - AD_{15} ist nicht definiert. Nach Beendigung des Refreshzyklus wird der Zeilenzähler um zwei erhöht. Er durchläuft einen Bereich von 256 Zeilen.

7.3. Periodisches Refresh

Das Refresh-Enable-Bit (RE) hat nur Bedeutung für das periodische Refresh. Refreshzyklen werden auch bei Aktivierung der $\overline{\text{STOP}}$-Leitung erzeugt. Dabei ist aber der Zustand des RE-Bits beliebig. Ist RE auf Eins **gesetzt**, so ist das periodische Refresh freigegeben, und der Wert des 6-Bit-Rate-Feldes bestimmt die Zeit **zwischen** den aufeinanderfolgenden Refreshzyklen (Refresh-Periode). Ist RATE = 0, so beträgt die Refreshperiode 256 Taktzyklen; ist RATE = n, so beträgt die Refreshperiode 4n Taktzyklen (somit kann bei einem 4-MHz-Takt die Refreshperiode zwischen 1 µs und 64 µs liegen).

Der LDCTL-Befehl dient zum Setzen der Refreshrate, zum Setzen oder Löschen von RE oder zum Initialisieren bzw. Lesen des ROW-Feldes. (Eine ausführliche Beschreibung dieses Befehls ist in "CPU U 8001/U 8002 - Befehlsbeschreibung" zu finden).
Der Refreshzyklus wird so bald als möglich nach Ablauf der Refresh-Periode erzeugt. Dies bedeutet normalerweise nach dem letzten Taktzyklus des laufenden Vorganges. Erhält die CPU einen Trap oder einen Interrupt gleichzeitig mit einer Anforderung zum periodischen Refresh, so wird die Refreshoperation zuerst durchgeführt. Wenn die CPU nicht über den Bus verfügt (d. h., wenn $\overline{\text{BUSACK}}$ aktiv ist und sich damit die CPU im Bus-Abschalt-Zustand befindet), oder wenn die $\overline{\text{WAIT}}$-Leitung aktiviert wird, so kann die CPU keine Refreshzyklen ausgeben. Um auch solche Situationen zu beherrschen, besitzt die CPU U 8000 eine interne Schaltungsanordnung, die aufzeichnet, daß die Refreshperiode abgelaufen ist und Refreshzyklen nicht erzeugt werden konnten. Übernimmt die CPU wieder die Steuerung des Busses, oder wird die $\overline{\text{WAIT}}$-Leitung wieder inaktiv, so holt sie sofort die übersprungenen Refreshzyklen nach. Die interne Schaltungsordnung kann bis zu zwei solcher übersprungener Refreshoperationen aufzeichnen.
Nach einer Reset-Operation wird periodisches Refresh gesperrt (RE wird gelöscht) und die interne Schaltungsordnung, welche die übersprungenen Refreshs zählt, wird auf Null zurückgestellt.

7.4. Refresh im Stop-Zustand

Die CPU besitzt drei interne Betriebszustände: Arbeitszustand, Stop-Zustand und Bus-Abschaltzustand (siehe Abschnitt 2.8.). Sie tritt in den Stop-Zustand ein, wenn die $\overline{\text{STOP}}$-Leitung aktiviert wird.
Befindet sich die CPU in diesem Zustand, so erzeugt sie ständig drei Taktzyklen dauernde Refresh-Zugriffe. Geht $\overline{\text{STOP}}$ wieder auf High, so wird noch ein Refreshzyklus ausgeführt. Anschließend wird mit den restlichen Taktzyklen die Befehlsholeoperation fortgesetzt.

Kapitel 8

8. Externes Interface

8.1. Einleitung

Dieses Kapitel beschreibt die Aktivitäten an den CPU-Anschlüssen, welche sich aus den in Kapitel 2 bis 7 beschriebenen Operationen ergeben. Da die Pins an den Systembus angeschlossen sind (siehe Bild 2.3 in Kapitel 2), wird sich das Wesentliche auf den Bus und die Busoperationen konzentrieren. Die CPU U 8000 realisiert an ihren Anschlüssen das 8000-Bus-Protokoll. In den folgenden Abschnitten wird das Interface zwischen der CPU U 8000 und ihrer Umgebung ausführlich beschrieben.

8.2. Busoperationen

Auf dem Systembus können zwei Operationsarten auftreten: Zugriffe und Anforderungen. Zu jeder Zeit kann eine Einheit (sie wird als Busmaster bezeichnet), den Bus steuern. Dies kann entweder eine CPU oder eine DMA sein. Eine andere Einheit auf dem Bus reagiert auf den vom Busmaster eingeleiteten Zugriff.
Es kann immer nur ein Zugriff auf einmal stattfinden.
Es werden sechs verschiedene Zugriffe unterschieden:

- Speicherzugriff:
  Diese Art dient zur Übertragung von 8 oder 16 Datenbits zu oder von einem Speicherplatz (Abschnitt 8.4.2.).
- E/A-Zugriff
  Diese Art dient zur Übertragung von 8 oder 16 Datenbits zu oder von einem peripheren Gerät oder einem CPU-Ergänzungsschaltkreis, wie z. B. der MMU (Abschnitt 8.4.3.).
- Interrupt/Trap-Bestätigung:
  Diese Art dient der Bestätigung eines Interrupts oder Traps und zur Übernahme eines Kennwortes (Abschnitt 8.4.4.).
- EPU-Transfer:
  Diese Art dient der Übertragung von 16 Datenbits zwischen der CPU und einem Erweiterungsprozessor (EPU) (Abschnitt 8.4.5.).
- Refresh:
  Diese Zugriffe übertragen keine Daten. Sie frischen dynamische Speicher auf (Abschnitt 8.4.6.).
- Interne Operation:
  Es werden keine Daten übertragen. Die CPU führt eine interne Operation aus (Abschnitt 8.4.6.).
  Zugriffe kann nur der Busmaster einleiten. Anforderungen können jedoch auch von anderen Einheiten ausgelöst werden, welche den Bus nicht steuern. Folgende vier Arten von Anforderungen können auftreten:
  - Interruptanforderung:
    Diese Art dient dazu, eine Bedienung durch die CPU anzufordern (Abschnitt 8.6.1.).

- Busanforderung:
  Diese Art fordert die Übergabe der Bussteuerung an eine andere Einheit (Abschnitt 8.6.2.).
- Ressource-Anforderung:
  Diese Art fordert die Übergabe der Steuerung einer besonderen Systemressource (Abschnitt 8.6.3.).
- Stopanforderung:
  Diese Art dient dazu, die Ausführung von CPU-Befehlen zu verzögern (Abschnitt 8.6.4.).

Tritt eine Anforderung auf, so wird sie in folgender Weise durch die CPU beantwortet: Bei einem Interrupt-Request wird ein Interrupt-Acknowledge-Zyklus eingeleitet. Bei Bus-Request nimmt die CPU den Bus-Abschalt-Zustand ein, gibt den Bus frei und aktiviert ein Acknowledge-Signal. Eine Stopanforderung stoppt die Ausführungen der CPU und veranlaßt diese,in den Stop/Refresh-Zustand einzutreten. Die CPU erzeugt eine Ressource-Anforderung, wenn sie den Multi-Mikro-Anforderungsbefehl (MREQ) ausführt.

8.3. CPU-Pins

Gemäß ihrer Funktion können die CPU-Pins in fünf Kategorien eingeteilt werden:

8.3.1. Zugriffssteuerungspins

Diese Signale sind vorgesehen zur zeitlichen Steuerung der Datenübertragung auf dem U 8000-Bus.

AD_0 bis AD_{15}, Adressen/Daten (Ausgang, aktiv High, Tri-State). Diese Multiplex-Daten/Adreßleitungen führen E/A-Adressen, Speicheradressen oder Daten während der Buszugriffe. Beim U 8001 wird nur der Offset-Teil der Speicheradressen auf diesen Leitungen transportiert.

SN_0 bis SN_7, Segmentnummer (nur U 8001, Ausgang, aktiv High, Tri-State). Diese Leitungen enthalten den Segmentnummernteil einer Speicheradresse.

ST_0 bis ST_3, Status (Ausgang, aktiv High, Tri-State).
Diese Leitungen zeigen die Art des Zugriffs an, welche auf dem Bus stattfindet und geben zusätzliche Informationen über den Zugriff (wie z. B. den Adreßraum bei Speichervorgängen).

$\overline{AS}$ Adreß-Strobe (Ausgang, aktiv Low, Tri-State). Die steigende Flanke von $\overline{AS}$ zeigt den Beginn eines Zugriffs an. Zu diesem Zeitpunkt sind die Adreß-, ST_0 ... ST_3, $N/\overline{S}$, $R/\overline{W}$ und $B/\overline{W}$-Signale gültig.

$\overline{DS}$ Daten-Strobe (Ausgang, aktiv Low, Tri-State).
$\overline{DS}$ dient zur Zeitsteuerung des Datentransportes zur oder von der CPU.

$R/\overline{W}$ Read/Write (Lesen/Schreiben) (Ausgang, Low = Write, Tri-State).
Dieses Signal bestimmt die Richtung der Datenübertragung bei Speicher-, E/A- oder EPU-Zugriffen.

$N/\overline{S}$ Normal/System (Ausgang, Low = System, Tri-State).
Diese Leitung führt das Komplement des $S/\overline{N}$-Bits im FCW und signalisiert, ob sich CPU im Normal- oder Systemmodus befindet.

$B/\overline{W}$ Byte/Wort (Ausgang, Low = Wort, Tri-State)
Dieses Signal zeigt an, ob während eines Zugriffes ein Datenbyte oder ein Datenwort übertragen werden soll.

$\overline{WAIT}$ (Warten) (Eingang, aktiv Low).
Ein Low auf dieser Leitung zeigt an, daß die angesprochene Einheit für den Zugriff mehr Zeit benötigt.

$\overline{MREQ}$ Memory Request (Speicheranforderung) (Ausgang, aktiv Low, Tri-State).
Eine fallende Flanke auf dieser Leitung zeigt an, daß der Adreß-/Datenbus eine Speicheradresse führt.

8.3.2. Bussteuerpins

Diese Pins tragen die Signale zur Anforderung und Übergabe der Bussteuerung.

$\overline{\text{BUSREQ}}$ Bus Request (Busanforderung) (Eingang, aktiv Low).
Ein Low zeigt an, daß ein Busanforderer die Steuerung des Busses erhalten hat oder versucht, sie zu erhalten.

$\overline{\text{BUSACK}}$ Bus Acknowledge (Busbestätigung) (Ausgang, aktiv Low).
Ein Low auf dieser Leitung zeigt an, daß die CPU die Steuerung des Busses als Reaktion auf eine Busanforderung abgegeben hat.

8.3.3. Interrupt/Trap-Pins

Diese Pins übertragen Interrupt- und externe Trapanforderungen zur CPU.

$\overline{\text{NMI}}$ Nicht-Maskierbarer Interrupt (Eingang, aktiv Low).
Ein High-zu-Low-Übergang auf der $\overline{\text{NMI}}$-Leitung signalisiert einen nichtmaskierbaren Interrupt.

$\overline{\text{NVI}}$ Nicht-Vektorisierter Interrupt (Eingang, aktiv Low).
Ein Low auf dieser Leitung zeigt einen nichtvektorisierten Interrupt an.

$\overline{\text{VI}}$ Vektorisierter Interrupt (Eingang, aktiv Low).
Ein Low auf dieser Leitung signalisiert einen vektorisierten Interrupt.

$\overline{\text{SEGT}}$ Segment-Trap (nur U 8001, Eingang, aktiv Low).
Ein Low auf dieser Leitung zeigt einen Segmenttrap an.

8.3.4. Multi-Mikro-Pins

Diese Pins stellen das Interface des U 8000 zu den Ressourceanforderungsleitungen des U 8000-Bus dar.

$\overline{\text{MI}}$ Multi-Mikro-In (Eingang, aktiv Low). Dieser Eingang dient zum Abtasten des Status der Ressourceanforderungsleitung.

$\overline{\text{MO}}$ Multi-Mikro-Out (Ausgang, aktiv Low).
Die CPU verwendet diese Leitung für Ressourceanforderungen.

8.3.5. Prozessor-Steuerung

Diese Pins führen Signale, welche die Gesamtoperation der CPU steuern.

$\overline{\text{STOP}}$ (Eingang, aktiv Low). Diese Leitung dient zum Verzögern der CPU-Operation beim Holen des ersten Wortes eines Befehls.

$\overline{\text{RESET}}$ (Eingang, aktiv Low). Ein Low auf dieser Leitung setzt die CPU zurück.

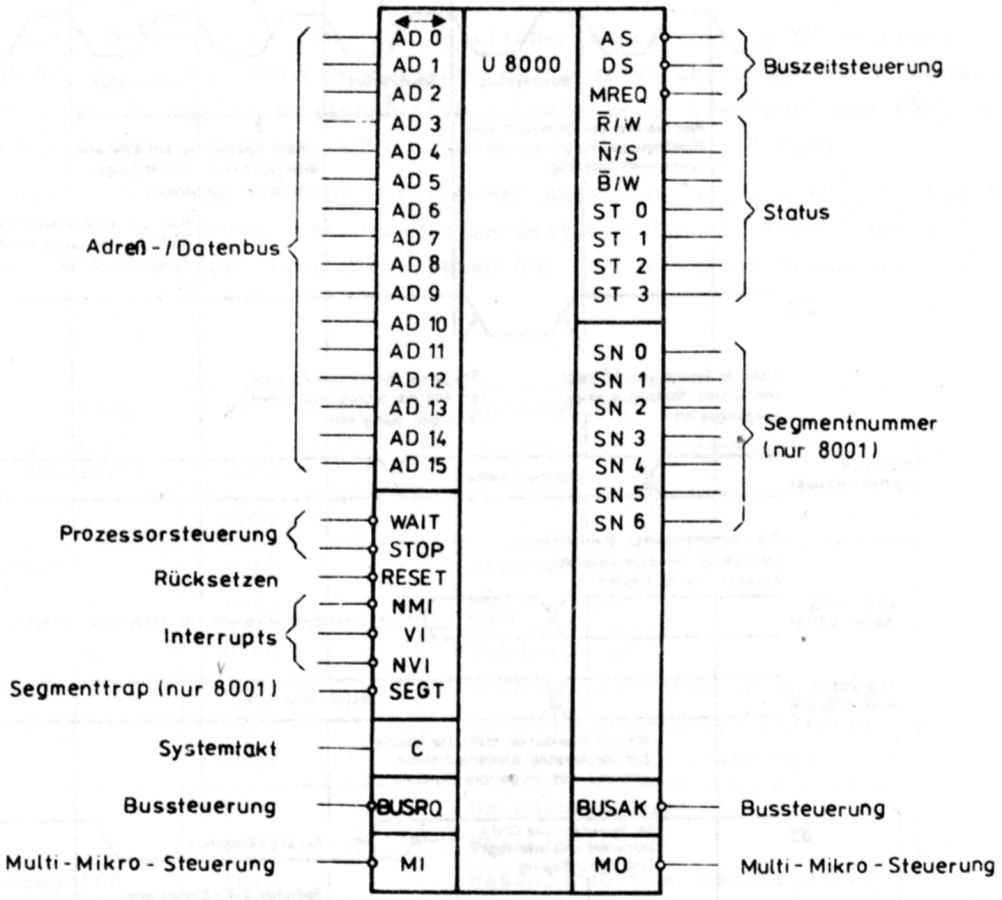


Bild 8.1 Schaltzeichen U 8000

8.4. Zugriffe

Datentransfers zur und von der CPU geschehen mit Hilfe von Zugriffen. Bild 8.2 zeigt hierfür das allgemeine Zeitschema.

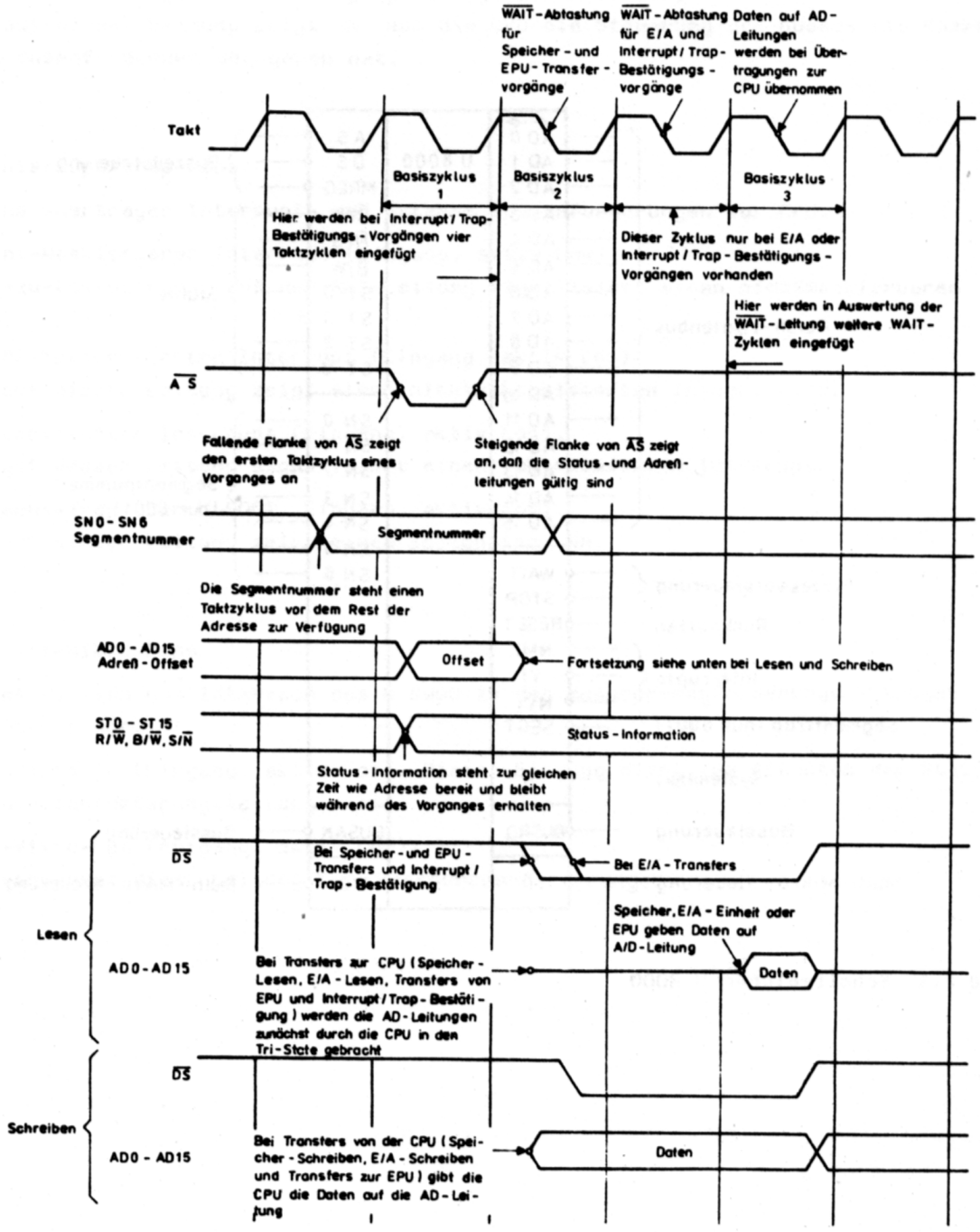


Bild 8.2 Genereller Zeitablauf

Alle Zugriffe beginnen mit der fallenden Flanke von $\overline{AS}$. Die steigende Flanke von $\overline{AS}$ zeigt, daß die Statussignale ST_0 - ST_3 gültig sind.
Diese Leitungen zeigen die Art des eingeleiteten Zugriffs an (siehe Tabelle 8.1.).
Die sechs Zugriffsarten werden in den folgenden Abschnitten beschrieben. Der Statusleitung zugeordnet sind drei andere Leitungen, die zu dieser Zeit gültig sind.
Diese sind Normal/System (N/$\overline{S}$), Read/Write (R/$\overline{W}$) und Byte/Wort (B/$\overline{W}$).

Erfordert der Zugriff eine Adresse, so ist diese ebenfalls bei der ansteigenden Flanke von $\overline{AS}$ gültig. Für Interrupt-Acknowledge, EPU-Transfer oder interne Operation ist keine Adresse erforderlich. Um den Erfordernissen der externen Speicherverwaltungseinheit zu entsprechen, sind beim U 8001 die Segmentnummerleitungen SN_0 ... SN_6 einen Taktzyklus früher gültig. (Weitere Einzelheiten siehe Kapitel 2).

Die CPU nutzt Daten-Strobe ($\overline{DS}$) zur zeitlichen Steuerung des Datentransfers. Zu beachten ist, daß Refresh und interne Operationen keine Daten übertragen und deshalb $\overline{DS}$ auch nicht aktiv wird. Bei Schreiboperationen (R/$\overline{W}$ = Low), zeigt Low-Pegel der $\overline{DS}$-Leitung an, daß gültige Daten des Busmasters auf den Leitungen AD_0 - AD_{15} liegen.

Bei Leseoperationen (R/$\overline{W}$ = High) bringt der Busmaster AD_0 bis AD_{15} in den Tri-State, bevor er $\overline{DS}$ auf Low bringt, so daß die adressierte Einheit ihre Daten auf den Bus geben kann. Der Busmaster übernimmt diese Daten bei der fallenden Taktflanke kurz bevor DS auf High geht.

Art des Zugriffs	ST_3 bis ST_0	Erläuterung
Interne Operation	0000	
Refresh	0001	
E/A-Zugriff	0010	Standard-E/A
	0011	Spezial-E/A
Interrupt/Trap-Acknowledge	0100	Segmenttrap
	0101	Nichtmaskierbarer Interrupt
	0110	Nichtvektorisierter Interrupt
	0111	Vektorisierter Interrupt
Speicherzugriff	1000	Datenadreßraum
	1001	Stackadreßraum
	1010	Datenadreßraum, EPU-Transfer
	1011	Stackadreßraum, EPU-Transfer
	1100	Programmadreßraum, n-tes Wort eines Befehls (IFn)
	1101	Programmadreßraum, erstes Wort eines Befehls (IF1)
EPU-Transfer	1110	
Reserviert	1111	

Tabelle 8.1.: Statuskodes

8.4.1. Wait

Wie Bild 8.3 zeigt, wird $\overline{WAIT}$ bei der fallenden Taktflanke im T_2-Zyklus von der CPU abgetastet. Wenn $\overline{WAIT}$ aktiv ist, wird vor dem Zyklus, in dem die steigende Flanke von $\overline{DS}$ liegt, ein Zyklus eingefügt. In diesem und jedem weiteren eingefügten Zyklus wird $\overline{WAIT}$ erneut bei der fallenden Flanke des Taktes abgefragt und falls Low vorliegt, wird ein weiterer Zyklus eingefügt. Auf diese Weise kann ein Zugriff beliebig verlängert werden. Bedeutung hat dies für langsamere Speicher oder E/A-Einheiten.
Es muß betont werden, daß $\overline{WAIT}$ ein synchroner Eingang ist. Deshalb müssen die dort angelegten Signale den Setz- und Haltezeiten nach Anhang A entsprechen. Damit die CPU richtig arbeitet kann, ist es erforderlich, daß asynchron erzeugte $\overline{WAIT}$-Signale synchronisiert werden.

8.4.2. Speicherzugriffe

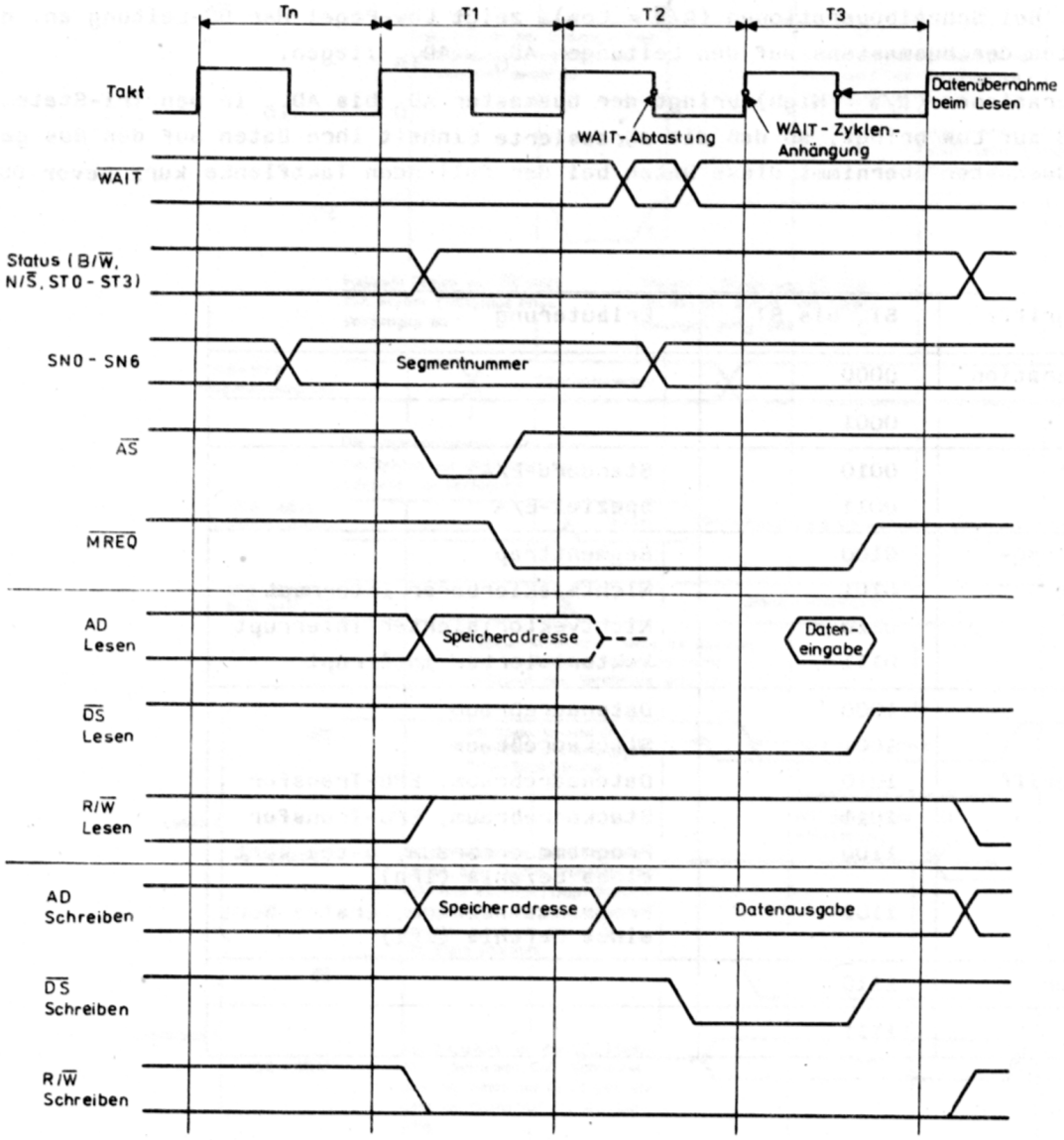


Bild 8.3 Speicherlese- und -schreibvorgang

Bei Speicherzugriffen werden Daten zum Speicher transportiert oder vom Speicher geholt. Sie werden während der Programmausführung erzeugt, um Befehle vom Speicher zu holen (Kapitel 4) und um Speicherdaten zu holen oder abzulegen (Kapitel 5). Sie werden auch erzeugt, um den alten Programmstatus zu retten und den neuen Programmstatus während der Interrupt- und Trapbehandlung und nach Reset (Kapitel 6) zu holen.

Wie Bild 8.3. zeigt, dauert ein Speicherzugriff drei Taktzyklen, wenn er nicht durch $\overline{WAIT}$ verlängert wird.

Neben dem Anzeigen eines Speicherzugriffs geben die Statuspins folgende Informationen:

- Ob der Speicherzugriff zum Daten- (1000, 1010), Stack- (1001, 1011) oder Programm- (1100, 1101) Adreßraum geschieht (Kaptiel 3).
- Ob das erste Wort eines Befehls geholt wird (1101).
- Ob die Daten von einem Erweiterungsprozessor (1010, 1011) geliefert (write) oder entgegen genommen (read) werden sollen. Auch die Statuskodes 1000 und 1001 können zeigen, daß der Erweiterungsprozessor die Daten übernehmen oder bereitstellen soll.

Bei steigender Flanke von $\overline{AS}$ befindet sich beim U 8002 die vollständige und gültige Speicheradresse auf den Leitungen AD_0 bis AD_{15}. Beim U 8001 befindet sich der Offsetteil der segmentierten Adresse auf AD_0 bis AD_{15}, die Segmentnummer befindet sich auf SN_0 bis SN_6. Der Segmentteil SN_0 bis SN_6 ist etwa einen Zyklus vor AD_0 bis AD_{15} gültig.
Die zu oder von ungeraden Speicherplätzen (Adreßbit 0 ist 1) transportierten Bytes werden stets auf den Leitungen AD_0 bis AD_7 übertragen (Bit 0 auf AD_0). Die zu oder von geraden Speicherplätzen (Adreßbit 0 ist 0) transferierten Bytes werden stets auf den Leitungen AD_8 bis AD_{15} übertragen (Bit 0 auf AD_8).

Somit hat der einem U 8000-Prozessor zugeordnete Speicher, wie in Bild 8.4. dargestellt auszusehen.
Für Bytelesen B/$\overline{W}$ High, R/$\overline{W}$ High) verwendet die CPU nur dasjenige Byte, dessen Adresse sie ausgibt. Für Byteschreiben (B/$\overline{W}$ High, R/$\overline{W}$ Low) hat der Speicher nur das Byte zu übernehmen, dessen Adresse durch die CPU ausgegeben wurde. Während des Schreibens eines Bytes in den Speicher wird nämlich von der CPU das gleiche Byte auf beide Hälften des AD-Busses gelegt, wobei die entsprechende Speicherzelle durch A_0 ausgewählt wird. Bei Worttransfers (B/$\overline{W}$ = Low) werden alle 16 Bit von der CPU übernommen (Lesen: R/$\overline{W}$ = High) oder dem Speicher übergeben (Schreiben: R/$\overline{W}$ = Low).

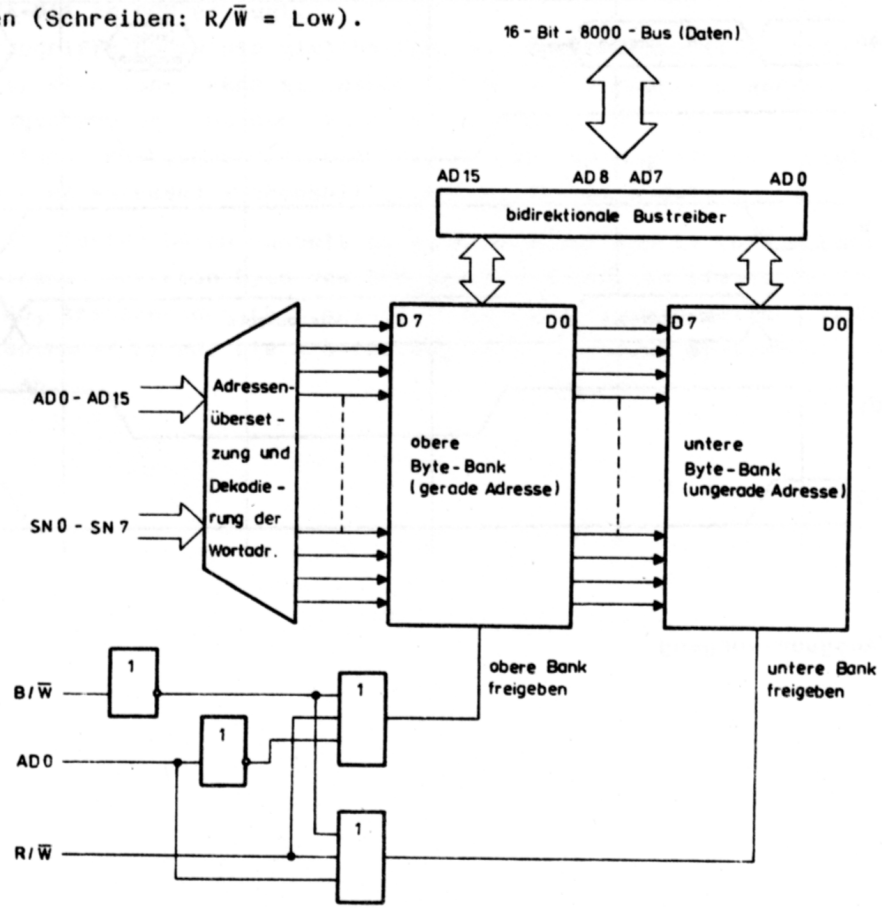


Bild 8.4 Speicherorganisation

Wie in Abschnitt 8.5. ausführlicher beschrieben, erscheinen eine CPU U 8000 und ein Erweiterungsprozessor (EPU) wie eine einzelne CPU, wobei die CPU die Adressen-, Status- und Zeitsteuersignale bereitstellt, während der Erweiterungsprozessor Daten bereitstellt oder entgegen nimmt.

8.4.3. E/A-Zugriffe

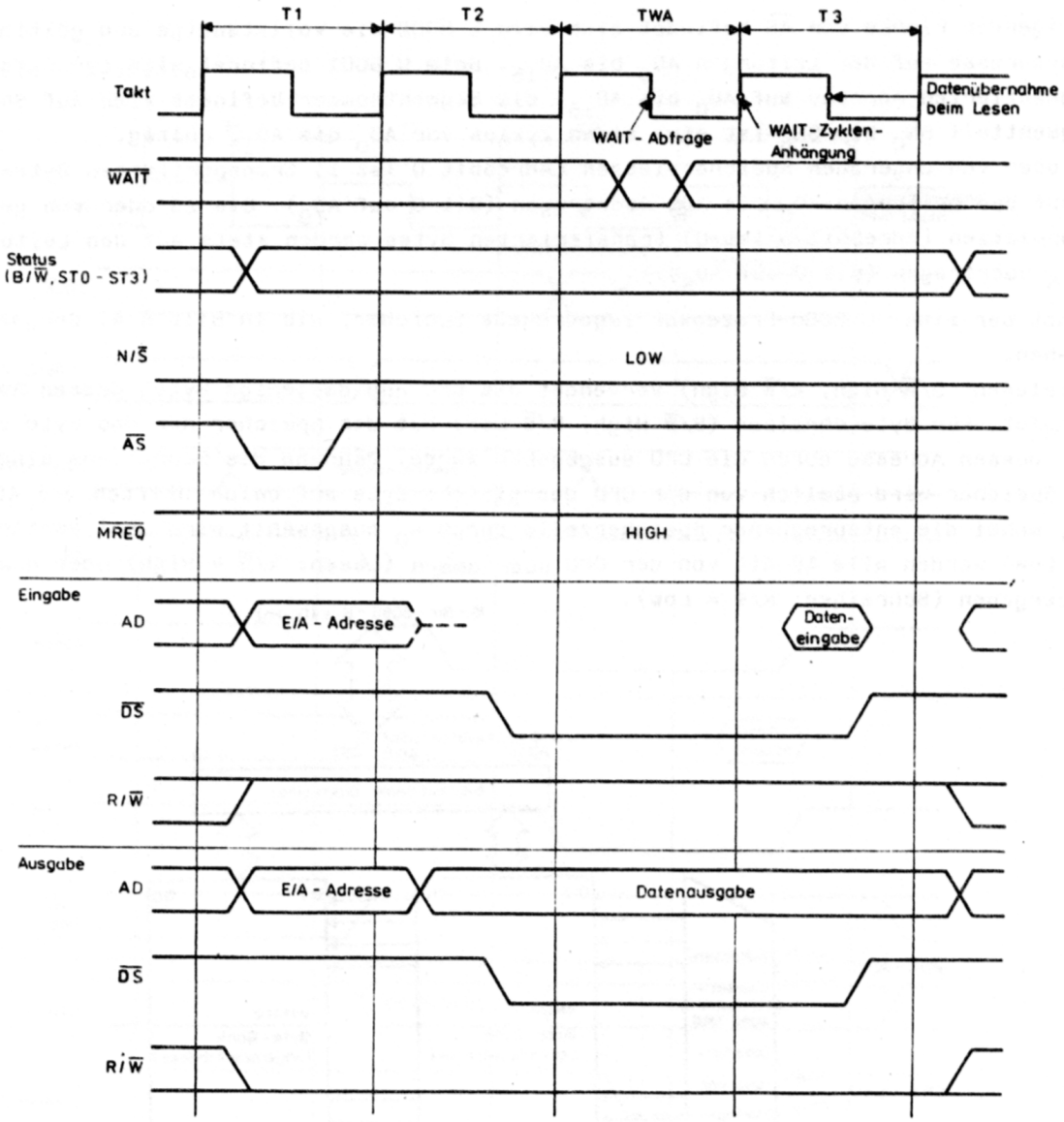


Bild 8.5 Ein/Ausgabe-Vorgang

Bei E/A-Zugriffen werden Daten zwischen peripheren Einheiten oder CPU-Ergänzungsschaltkreisen (z. B. MMU's) und der CPU transportiert. Sie werden während der Ausführung von E/A-Befehlen erzeugt. Wie in Bild 8.5 dargestellt, sind E/A-Zugriffe mindestens vier Taktzyklen lang und können durch Einfügen von $\overline{WAIT}$-Zyklen verlängert werden. Diese zusätzlichen Taktzyklen ermöglichen die Arbeit mit langsamerer Peripherie.

Die Statusleitungen zeigen an, ob der Zugriff zum Standard-E/A- (0010) oder Spezial-E/A- (0011) Adreßraum geschieht. Die N/$\overline{S}$-Leitung ist stets Low und zeigt den Systemmodus an. Bei steigender Flanke von $\overline{AS}$ ist die E/A-Adresse auf den Leitungen AD_0 bis AD_{15} stabil. Die E/A-Adresse ist stets 16 Bit lang. Deshalb sind die Segmentnummerleitungen bei der CPU U 8001 nicht definiert. Für Bytetransfers (B/$\overline{W}$ = High) im Standard-E/A-Raum müssen die Adressen ungerade sein, während die Adressen für Bytetransfers im Spezial-E/A-Raum gerade sein müssen.

Wortdaten (B/$\overline{W}$ = Low) zur oder von der CPU werden auf AD_0 bis AD_{15} übertragen. Bytedaten (B/$\overline{W}$ = High) werden auf AD_0 bis AD_7 für Standard-E/A und auf AD_8 bis AD_{15} für Spezial-E/A übertragen. Dadurch können periphere Einheiten oder CPU-Ergänzungsschaltkreise nur acht der 16 AD-Leitungen zugeordnet werden.

Die Lese/Schreibleitung (R/$\overline{W}$) zeigt die Richtung des Datentransfers an: Peripheres Gerät zur CPU (Lesen: R/$\overline{W}$ = High) oder CPU zum peripheren Gerät (Schreiben: R/$\overline{W}$ = Low).

8.4.4. EPU-Transferzugriffe

Diese Zugriffe transportieren Daten zwischen der CPU und einer EPU und gestatten es somit der CPU, Daten zu oder von einer EPU zu transferieren oder die Statusregister einer EPU zu lesen oder zu beschreiben. Sie werden während der Ausführung eines erweiterten Befehls bei gesetztem EPA-Bit im FCW erzeugt.
EPU-Transferzugriffe haben die gleiche Form wie Speicherzugriffe (Bild 8.3) und sind somit drei Taktzyklen lang, wenn sie nicht durch $\overline{WAIT}$ verlängert werden. Es wird keine Adresse erzeugt, und es gibt nur einen Statuskode (1110).
In einem Mehrfach-EPU-System wird die EPU, die an dem Zugriff beteiligt sein soll, implizit (nicht durch eine Adresse) ausgewählt (siehe Abschnitt 8.5.).

Bei den transferierten Daten handelt es sich um 16-Bit-Worte (B/$\overline{W}$ = Low) mit Ausnahme von Transfers zwischen dem Flag-Byte des FCW und einer EPU. In diesem Fall wird ein Datenbyte auf AD_0 bis AD_7 transferiert (B/$\overline{W}$ =High) . Die Lese/Schreib-Leitung (R/$\overline{W}$) zeigt die Richtung des Datentransfers an. Die N/$\overline{S}$-Leitung zeigt entweder Systemmodus (Low) oder Normalmodus (High) an.

8.4.5. Interrupt/Trap-Acknowledge

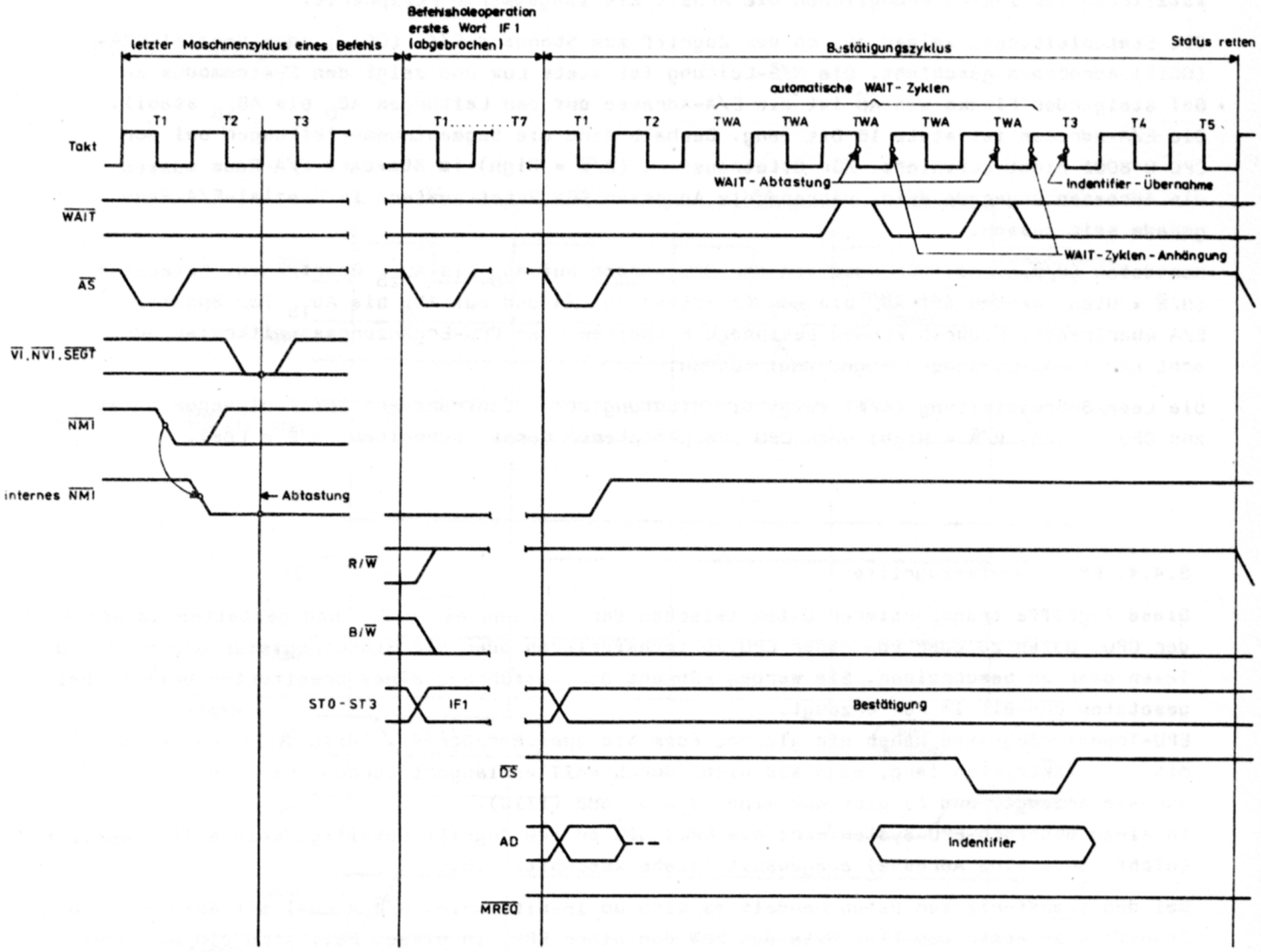


Bild 8.6 Interrupt- und Segmenttrap-Anforderung und deren Bestätigungsvorgang

Diese Vorgänge bestätigen einen Interrupt oder Trap und übernehmen ein 16-Bit-Kennwort (Identifier) von der Einheit, die den Interrupt oder Trap auslöste. Die Vorgänge werden automatisch durch die Hardware erzeugt, wenn ein Interrupt- oder Segmenttrap festgestellt wird.

Diese Vorgänge sind mindestens acht Taktzyklen lang (wie in Bild 8.6 dargestellt) und haben fünf automatische $\overline{\text{WAIT}}$-Zyklen. Die $\overline{\text{WAIT}}$-Zyklen dienen dazu, der Interrupt-Prioritätskette (oder einer Prioritätsverwaltungseinheit) genügend Zeit zu geben, bevor das Kennwort gelesen wird.

Die Statusleitungen signalisieren die Art der bestätigten Ausnahme. Die Möglichkeiten sind Segmenttrap (0100), nichtmaskierbarer Interrupt (0101), nichtvektorisierter Interrupt (0110) und vektorisierter Interrupt (0111), Es wird keine Adresse erzeugt. Die N/$\overline{S}$-Leitung zeigt Systemmodus (Low) an, die R/$\overline{W}$-Leitung zeigt Lesen (High) und die B/$\overline{W}$-Leitung Wort (Low) an.

Die einzige transferierte Datenposition ist das Kennwort, welches stets 16 Bit lang ist und von den Leitungen AD_0 bis AD_{15} bei der fallenden Taktflanke übernommen wird, bevor $\overline{DS}$ auf High geht.
Wie in Bild 8.6 dargestellt, gibt es zwei Stellen, wo $\overline{WAIT}$ abgetastet wird, und somit $\overline{WAIT}$-Zyklen eingeschoben werden können. Die erste dient zur Verzögerung der fallenden Flanke von $\overline{DS}$, so daß die Prioritätskette eine längere Stabilisierungszeit hat. Die zweite Stelle dient dazu, den Punkt zu verzögern, bei dem die Daten übernommen werden.

8.4.6. Interne Operationen und Refresh-Zyklen

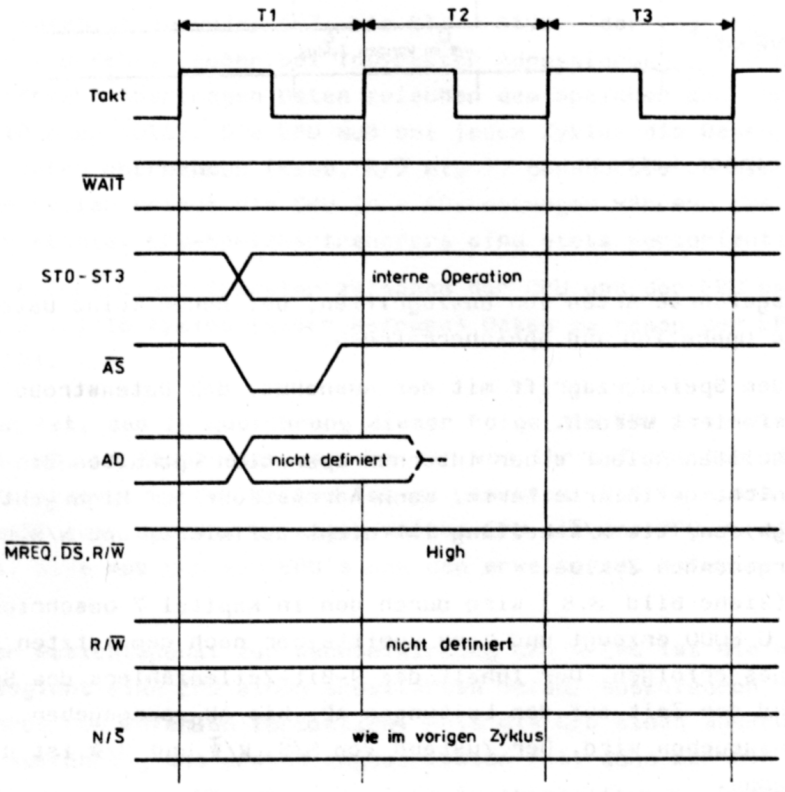


Bild 8.7 Zeitablauf bei interner Operation

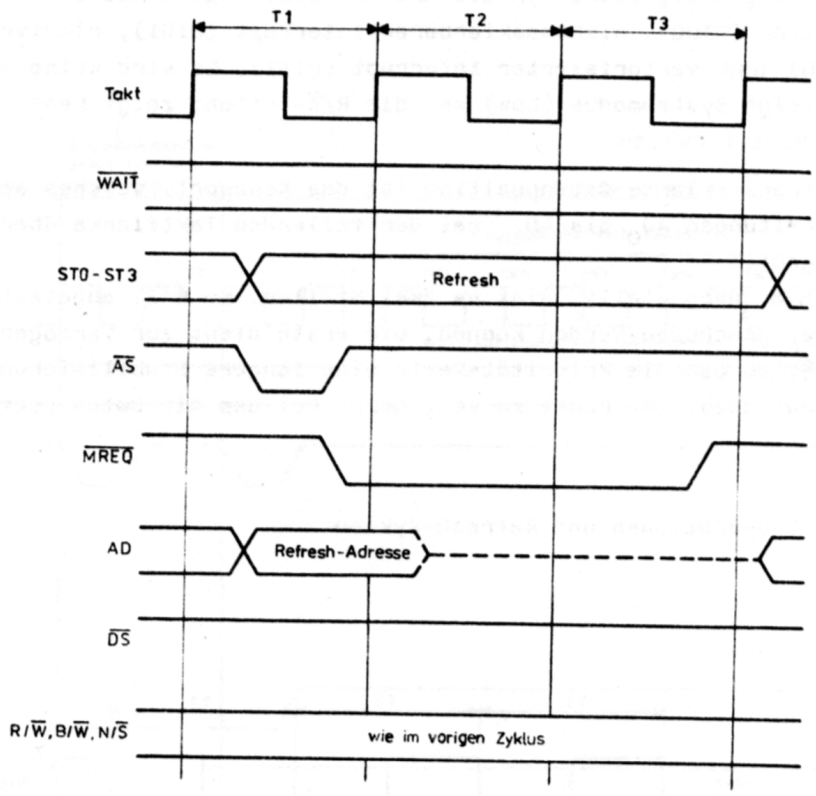


Bild 8.8 Zeitablauf bei Speicher-Refresh

Es gibt zwei von der CPU ausgeführte Arten von Buszugriffen, bei denen keine Daten transferiert werden: interne Operation und Speicherrefresh.

Beide Zugriffe entsprechen dem Speicherzugriff mit der Ausnahme, daß Datenstrobe High bleibt und keine Daten transferiert werden.
Bei dem in Bild 8.7 dargestellten Ablauf einer internen Operation enthalten die Adressen- und Segmentnummerleitungen nicht definierte Werte, wenn Adreßstrobe auf High geht. Die R/$\overline{W}$-Leitung zeigt Lesen (High) an; die B/$\overline{W}$-Leitung ist nicht definiert, und N/$\overline{S}$ bleibt in dem Zustand wie im vorhergehenden Zyklus.
Ein Speicherrefreshvorgang (siehe Bild 8.8) wird durch den in Kapitel 7 beschriebenen Refresh-Mechanismus der CPU U 8000 erzeugt und kann unmittelbar nach dem letzten Taktzyklus eines anderen Vorganges erfolgen. Der Inhalt des 9-Bit-Zeilenzählers des Speicherrefreshregisters wird während der Zeit auf den Leitungen AD_0 bis AD_8 ausgegeben, zu der normalerweise die Adresse ausgegeben wird. Der Zustand von N/$\overline{S}$, R/$\overline{W}$ und B/$\overline{W}$ ist der gleiche wie beim vorhergehenden Vorgang.
Während interner Operationen oder Refreshzyklen wird $\overline{WAIT}$ nicht abgetastet.

8.5. Wechselwirkung zwischen CPU und EPU

Die Arbeit einer CPU U 8000 mit einer oder mehreren EPU's erscheint wie die Arbeit eines einzigen CPU-Elements. Die CPU sorgt dabei für die Adresse sowie für Status- und Zeitsteuersignale, während der Erweiterungsprozessor die Daten übergibt bzw. übernimmt. Die EPU überwacht die durch die CPU ausgegebenen Status- und Zeitsteuersignale, so daß sie weiß, wann sie sich an einem Speicher- oder EPU-Transfervorgang beteiligen muß.
Soll die EPU an einem Speicherzugriff teilnehmen, so bringt die CPU bei aktivem $\overline{DS}$ ihre AD-Leitungen in den Tri-State, so daß die EPU über die Daten verfügen kann.
Um zu wissen, an welchem Zugriff sich die EPU beteiligen soll, muß sie nachstehende Folge von Ereignissen verfolgen:

- Wenn die CPU das erste Wort eines Befehls holt (ST_3 bis ST_0 = 1101), muß die EPU ebenfalls den vom Speicher geholten Befehl übernehmen und interpretieren. Handelt es sich hier um einen erweiterten Befehl, so besitzt er ein Identifizierungsfeld, das (gemeinsam mit dem zweiten Befehl) anzeigt, ob die EPU den Befehl ausführen soll oder nicht.

- Soll der Befehl durch die EPU ausgeführt werden, so wird im nächsten Zyklus (außer Refresh) durch die CPU das zweite Wort des Befehls geholt (ST_3 bis ST_0 = 1100). Die EPU muß auch dieses Wort übernehmen.

- Beinhaltet der Befehl eine Speicherschreib- oder Leseoperation, so erfolgt in den nächsten Befehlsholezyklen (ST_3 bis ST_0 = 1100) der Zugriff auf den Adressenteil des erweiterten Befehls (außer bei indirekter Adressierung). Die nächsten 1 ... 16 Zyklen (außer Refresh) übertragen Daten zwischen dem Speicher und der EPU (ST_3 bis ST_0 = 1000, 1001, 1010 oder 1011). Die EPU muß bei jedem Zyklus die Daten liefern (Write, R/$\overline{W}$ Low) oder die Daten übernehmen (Read, R/$\overline{W}$ High), gerade als ob sie ein Teil der CPU wäre. In beiden Fällen bringt die CPU ihre AD-Leitungen während des Datentransfers ($\overline{DS}$ Low) in den Tri-State. EPU-Speichertransfers sind stets wortorientiert (B/$\overline{W}$ Low).
- Wenn der Befehl einen Transfer zwischen der CPU und der EPU beinhaltet, so übertragen die nächsten 1 ... 16 Zyklen (außer Refresh) Daten zwischen der EPU und der CPU (ST_3 bis ST_0 = 1110).

Zu beachten ist, daß in Ausführung dieser Folge die EPU auch die $\overline{BUSACK}$-Leitung überwachen muß, um festzustellen, daß der von ihr auf dem Bus überwachte Vorgang von der CPU erzeugt wurde. Weiter ist zu beachten, daß in einem Mehrfach-EPU-System keine Auswahl über den Bus erfolgt, welche EPU mit der CPU zu einer bestimmten Zeit kooperiert. Dies muß von den EPU's aus den erweiterten Befehlen ermittelt werden, die sie übernehmen.

Ein letzter Gesichtspunkt zur Wechselwirkung CPU - EPU ist die Nutzung des $\overline{STOP}$-Pins der CPU. Beginnt eine EPU einen erweiterten Befehl auszuführen, dann kann die CPU das Holen und Ausführen von Befehlen fortsetzen. Holt die CPU einen anderen erweiterten Befehl, bevor die Ausführung des ersten abgeschlossen ist, dann muß die EPU das $\overline{STOP}$-Pin der CPU aktivieren und die CPU stoppen, (wie in Abschnitt 8.7. beschrieben) bis die Ausführung des ersten EPU-Befehls abgeschlossen ist.

Neben der Festlegung, ob sich die EPU an der Ausführung eines Befehls der erweiterten Prozessorarchitektur beteiligt oder nicht, muß die EPU aus den ersten beiden Befehlsworten auch folgendes bestimmen:

- Ob ein Speicherzugriff ausgeführt wird oder nicht und wieviele Befehlsworte geholt werden, ehe die Daten übertragen werden.
- Wieviele Datenworte zwischen Speicher und EPU bzw. zwischen EPU und CPU transportiert werden sollen.
- Welche Operation mit den Daten auszuführen ist.

8.6. Anforderungen (Requests)

Es gibt drei Arten von Anforderungssignalen, die der U 8000-Bus überträgt und die die CPU U 8000 akzeptiert.

- Interrupt/Trap-Anforderungen: Diese werden von einer anderen Einheit ausgegeben und von der CPU entgegengenommen und bestätigt.
- Busanforderungen: Diese werden von einem anderen potentiellen Busmaster eingeleitet und von der CPU entgegengenommen und bestätigt.
- Ressource-Anforderungen: Diese werden von Einheiten ausgelöst und entgegengenommen, die das Ressource-Request-Protokoll realisieren können.

Über die Möglichkeiten des U 8000-Busses hinaus akzeptiert die CPU U 8000 ein weiteres Anforderungssignal:

- Stopanforderung: Diese wird von einer anderen Einheit aktiviert und von der CPU akzeptiert.

Wird eine Anforderung ausgelöst, so wird sie ihrem Typ entsprechend beantwortet: Für Interrupt/Trap-Anforderungen wird ein Interrupt/Trap-Bestätigungsvorgang eingeleitet (Abschnitt 8.4.4.); für Busanforderungen wird ein Bestätigungssignal ausgegeben (Abschnitt 8.6.2. und 8.6.3.), für Stopanforderungen tritt die CPU in den Stop/Refreshzustand ein. Mit Ausnahme von Stop bietet der U 8000-Bus für alle Arten eine Prioritätskette zur Entscheidung zwischen gleichzeitigen Anforderungen.

8.6.1. Interrupt/Trap-Anforderungen

Wie in Bild 8.6 dargestellt, werden durch die CPU U 8000 drei Interrupts und ein externer Trap akzeptiert. Die Interrupt-Anforderungsleitung einer Einheit, die in der Lage ist, einen Interrupt zu erzeugen, kann mit einem der drei Interrupteingänge ($\overline{NMI}$, $\overline{NVI}$, $\overline{VI}$) der CPU U 8000 verbunden werden. Mehrere Geräte können an einen Interrupteingang angeschlossen werden.
Sie klären ihre Priorität über die Prioritätsketten. Das Segmenttrappin ($\overline{SEGT}$) wird durch die Speicherverwaltungs-Hardware aktiviert. Zur Bearbeitung der unterschiedlichen Anforderungen verwendet die CPU das gleiche Protokoll.

- Jeder High-Low-Übergang am $\overline{NMI}$-Eingang setzt das interne $\overline{NMI}$-Latch (asynchron flankengetriggert). Zu Beginn des letzten Taktzyklus im letzten Maschinenzyklus eines jeden Befehls werden die Eingänge $\overline{VI}$, $\overline{NVI}$, $\overline{SEGT}$ und der Zustand des internen $\overline{NMI}$-Latches abgefragt.
- Wurde ein Interrupt oder Trap ermittelt, so wird der darauf folgende Befehlsholezyklus (IF1) zwar ausgeführt, dann aber abgebrochen.
- Der nächste Maschinenzyklus ist der Interruptbestätigungszyklus (siehe Abschnitt 8.4.4.). Dessen Ergebnis ist das Einlesen eines Kennwortes (Identifier),von der Interrupteinheit mit der höchsten Priorität, über die AD-Leitungen.
- Dieses Kennwort wird zusammen mit dem alten Programmstatus auf dem Stack abgelegt und der neue Programmstatus wird geladen (siehe Kapitel 6).

8.6.2. Busanforderung

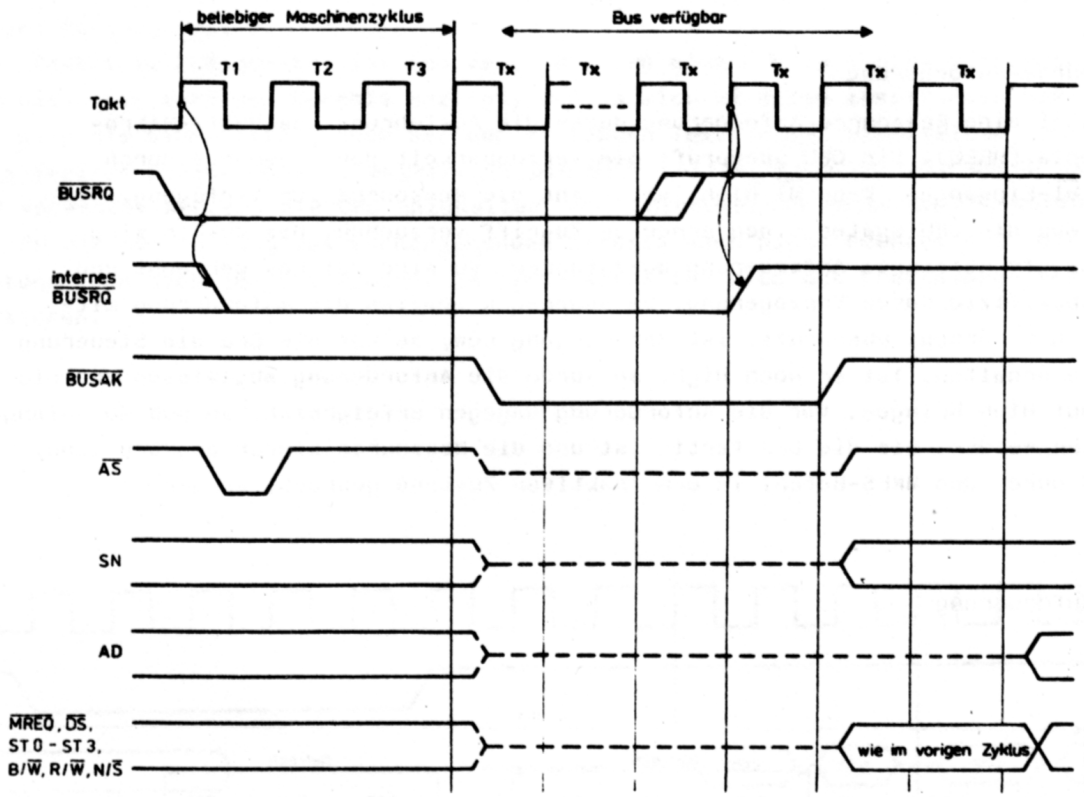


Bild 8.9 Zeitverlauf bei Busrequest und Busacknowledge

Durch Ausgabe einer Busanforderung erhält ein potentieller Bus-Master (z. B. DMA-Baustein) zur Datenübertragung auf dem Bus die Steuerung des Busses (siehe Bild 8.9). Eine Busanforderung wird ausgelöst, indem das $\overline{\text{BUSREQ}}$-Pin auf Low gebracht wird. Die Priorität wird auch hier normalerweise mittels einer Prioritätskette außerhalb der CPU geklärt. Das asynchrone $\overline{\text{BUSREQ}}$-Signal erzeugt ein internes synchrones $\overline{\text{BUSREQ}}$. Wenn das externe $\overline{\text{BUSREQ}}$ zu Beginn eines beliebigen Maschinenzyklus Low ist, sorgt das interne $\overline{\text{BUSREQ}}$ dafür, daß die Busfreigabebestätigungsleitung ($\overline{\text{BUSACK}}$) nach der Beendigung des laufenden Maschinenzyklus aktiviert wird. Die CPU tritt in den Bus-Abschalt-Zustand ein und gibt die Steuerung des Busses ab.

Alle CPU-Ausgänge mit Ausnahme von $\overline{\text{BUSACK}}$ und $\overline{\text{MO}}$ werden in den Tri-State gebracht. Die CPU erhält die Steuerung des Busses zwei Taktzyklen, nachdem $\overline{\text{BUSREQ}}$ auf High zurückgeht. Einheiten, die die Steuerung des Busses wünschen, müssen nachdem $\overline{\text{BUSREQ}}$ auf High gegangen ist, mindestens zwei Taktzyklen warten, bevor es erneut aktiviert wird.

8.6.3. Ressource-Anforderung

Die CPU erzeugt eine Ressource-Anforderung durch die Ausführung des Multi-Mikro-Request-Befehls (MREQ). Die CPU überprüft die Verfügbarkeit der Ressource durch Abfrage des $\overline{\text{MI}}$-Einganges. Wenn $\overline{\text{MI}}$ High ist, steht die Ressource zur Verfügung. Andernfalls muß die CPU später einen erneuten Zugriff versuchen. Das $\overline{\text{MO}}$-Pin wird verwendet, um die Ressource-Anforderung auszuführen. $\overline{\text{MO}}$ wird auf Low gebracht und nach einer spezifizierbaren Verzögerung, um anderen Einheiten die Anforderung mitzuteilen, wird $\overline{\text{MI}}$ erneut überprüft. Ist der Eingang Low, so hat die CPU die Steuerung der Ressource erhalten. Ist er noch High, so wurde die Anforderung abgewiesen und die CPU muß $\overline{\text{MO}}$ auf High bringen. War die Anforderung dagegen erfolgreich, so muß $\overline{\text{MO}}$ solange aktiv gehalten werden, bis die CPU fertig ist und die Ressource wieder abgeben kann. Dabei muß $\overline{\text{MO}}$ durch den MRES-Befehl in den inaktiven Zustand gebracht werden.

8.6.4. Stopanforderung

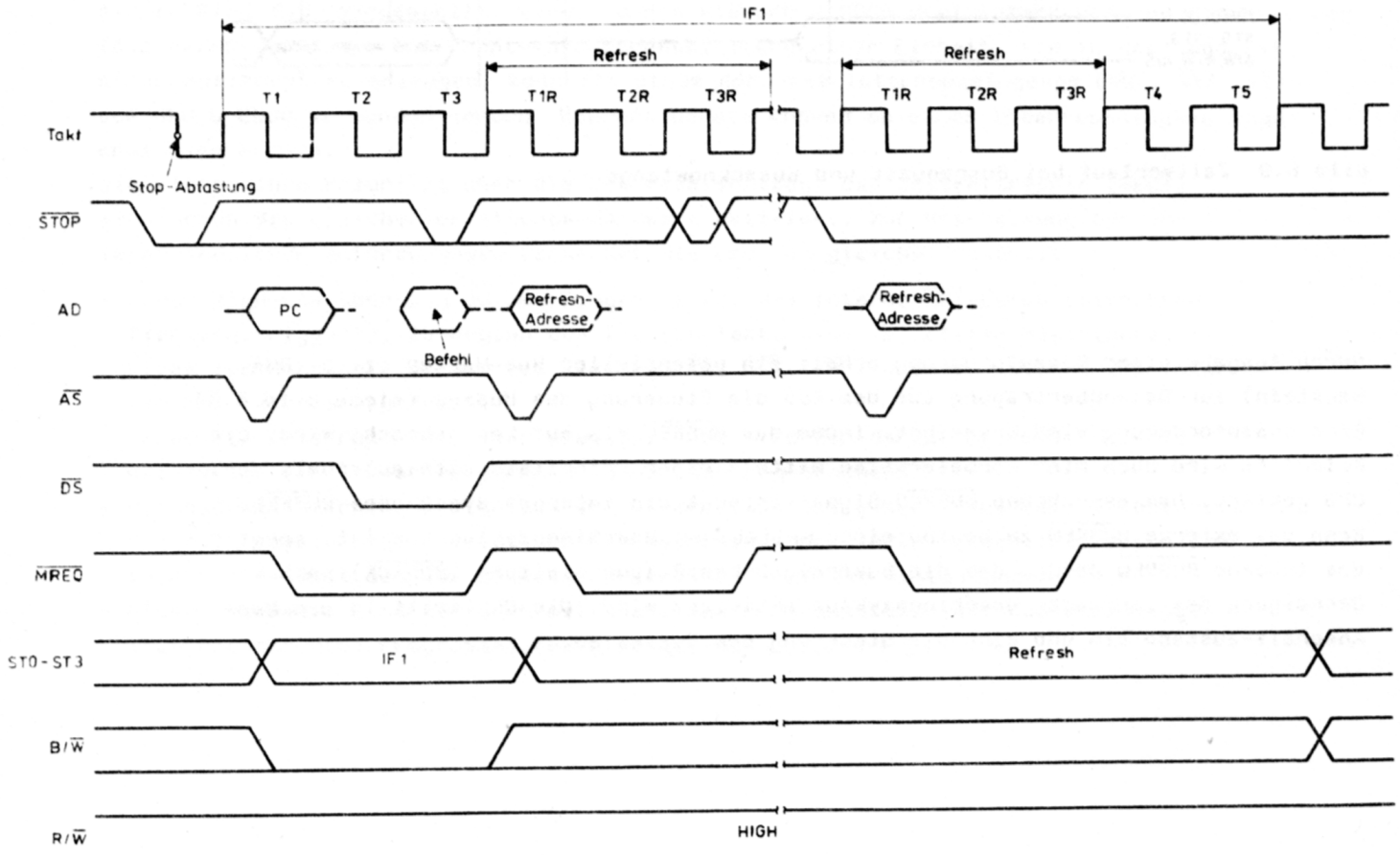


Bild 8.10 Zeitablauf bei STOP

Wie in Bild 8.10 dargestellt, wird das $\overline{\text{STOP}}$-Pin normalerweise bei der dem Befehlsholezyklus (IF1) vorhergehenden fallenden Flanke des Taktes abgefragt. Ist $\overline{\text{STOP}}$ Low, so tritt die CPU in den Stop/Refresh-Zustand ein, wobei die CPU nach dem dritten Takt des Befehlsholezyklus eine Folge von Speicherauffrischzyklen einschiebt. Das Zeilenfeld im Refresh-Zähler wird nach jedem Refreshzyklus um zwei erhöht. Wenn $\overline{\text{STOP}}$ wieder als High ermittelt wird, so wird der nächste Refreshzyklus zu Ende geführt, und der ursprüngliche Befehl wird fortgesetzt.
Wenn das EPA-Bit im FCW gesetzt ist (es zeigt an, daß eine EPU im System vorhanden ist) und das erste Wort des Befehls anzeigt, daß es sich um einen erweiterten Befehl handelt, wird die $\overline{\text{STOP}}$-Leitung auch bei der fallenden Taktflanke, unmittelbar vor dem Holen des zweiten Befehlswortes, abgefragt. Die $\overline{\text{STOP}}$-Leitung kann deshalb durch eine EPU dazu verwendet werden, die CPU anzuhalten, wenn diese einen weiteren EPU-Befehl holt und die EPU die Bearbeitung des vorhergehenden Befehls noch nicht beendet hat. Die $\overline{\text{STOP}}$-Leitung kann auch genutzt werden, um Einzelschrittbetrieb des Prozessors extern zu realisieren.

8.7. Reset

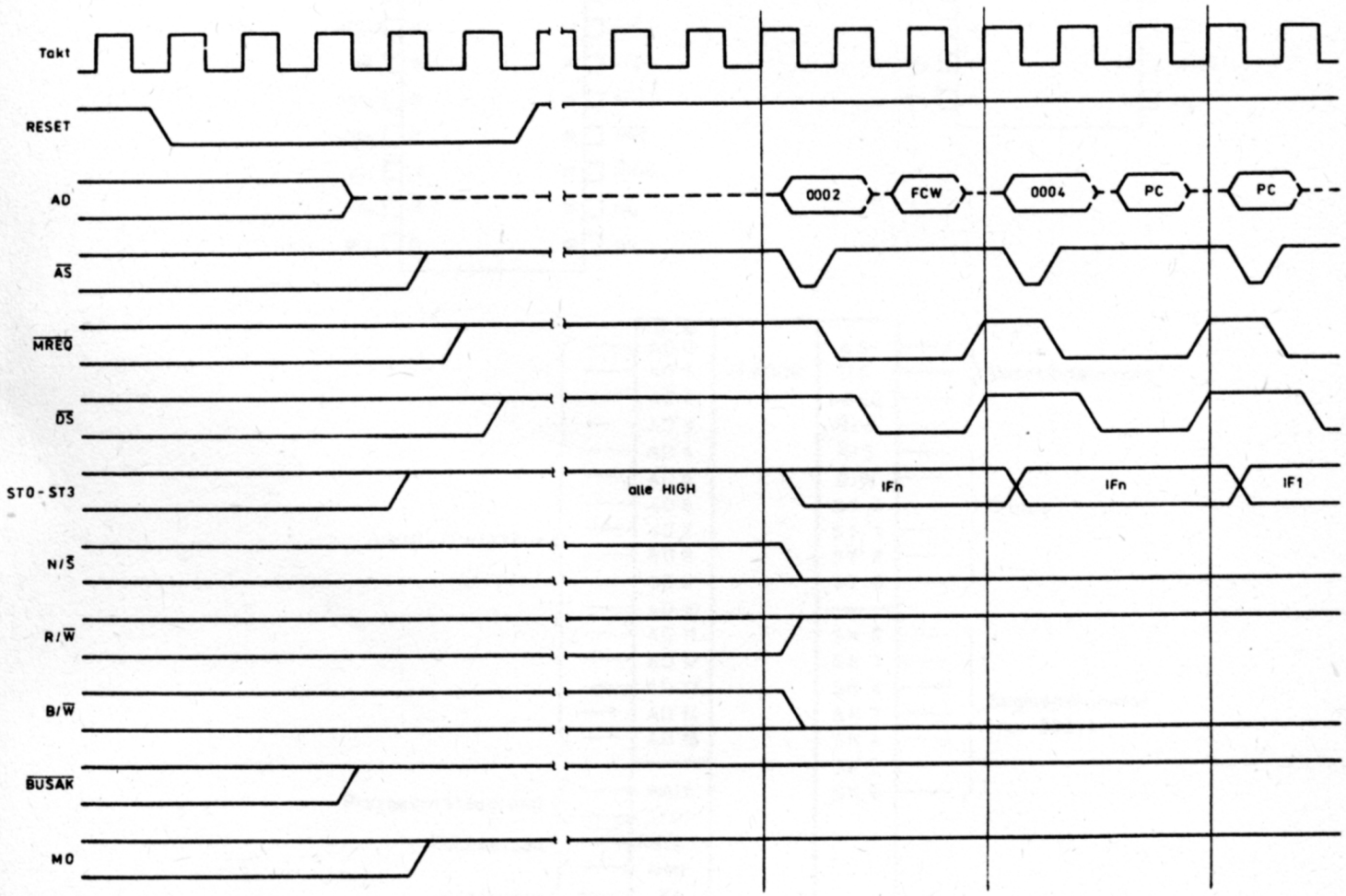
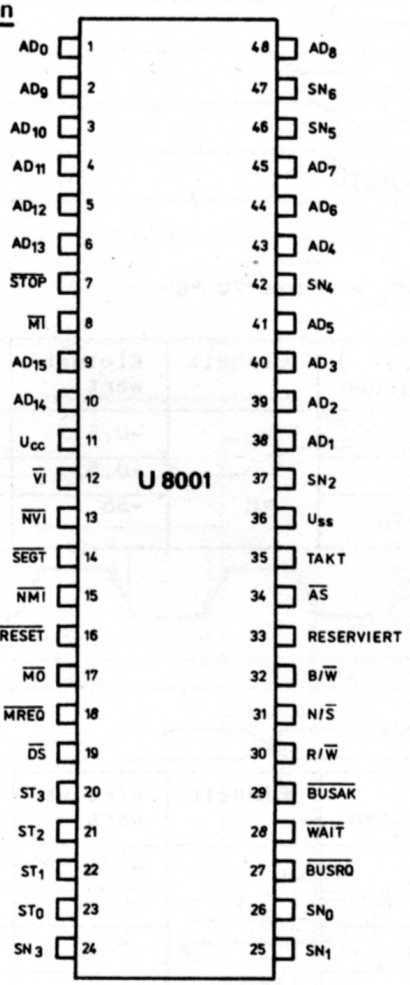
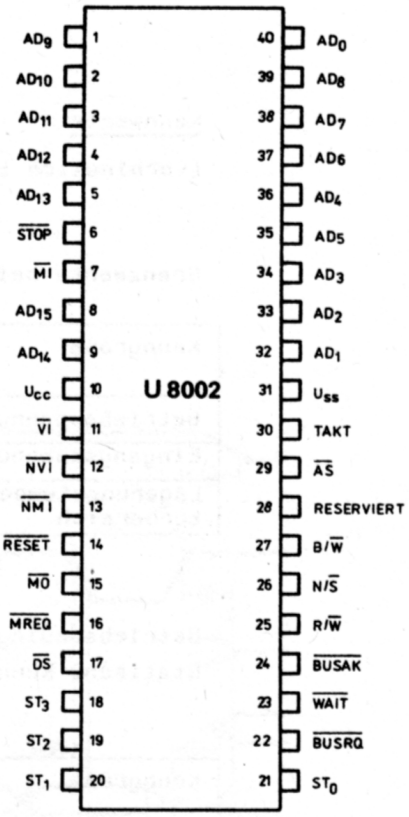


Bild 8.11 Ablauf bei Reset (U 8002)

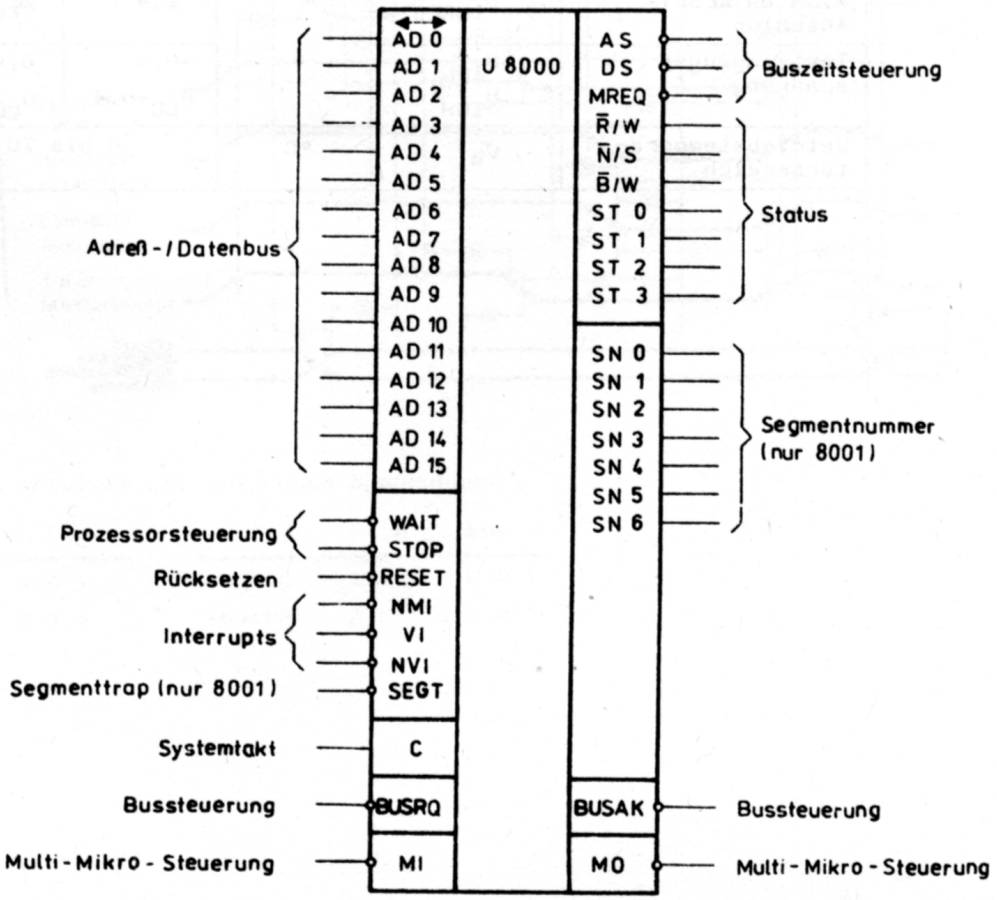
Ein Hardware-Reset bringt den U 8000 in einen definierten Zustand und initialisiert ausgewählte Steuerregister der CPU auf spezifizierbare Werte (siehe Kapitel 6).
Wenn die $\overline{\text{RESET}}$-Leitung auf Low gebracht wird, erfolgt das Rücksetzen am Ende des Taktzyklus.
Ein Systemreset hat gegenüber allen anderen Operationen des Chips einschließlich Interrupt, Traps, Busanforderungen und Stopanforderungen höchste Priorität. Reset ist zur Initialisierung eines Systems beim Anlegen der Betriebsspannung (power-up) zu verwenden. Innerhalb der fünf Taktzyklen, in denen $\overline{\text{RESET}}$ auf Low zu halten ist, gehen AD_0 bis AD_{15} in den Tri-State. $\overline{AS}$, $\overline{DS}$, $\overline{MREQ}$, $\overline{BUSACK}$, $\overline{MO}$, ST_0 bis ST_3 werden auf High gebracht und SN_0 bis SN_6 auf Low.
Die R/$\overline{W}$, B/$\overline{W}$ und N/$\overline{S}$-Leitungen sind nicht definiert. $\overline{\text{RESET}}$ muß mindestens fünf Taktzyklen lang auf Low gehalten werden.
Drei Taktzyklen nachdem RESET auf High zurückgekehrt ist, werden im Systemmodus fortlaufende Speicher-Lese-Zugriffe zur Initialisierung der Programmstatusregister ausgeführt. Diese entsprechen den unter 6.4. beschriebenen Speicherzugriffen.

Hardware-Informationen





Schaltzeichen



Kennwerte

(verbindlich ist die TGL 43019)

Grenzwerte bei U_{SS} = 0 V, ϑ_a = 0 bis 70 °C

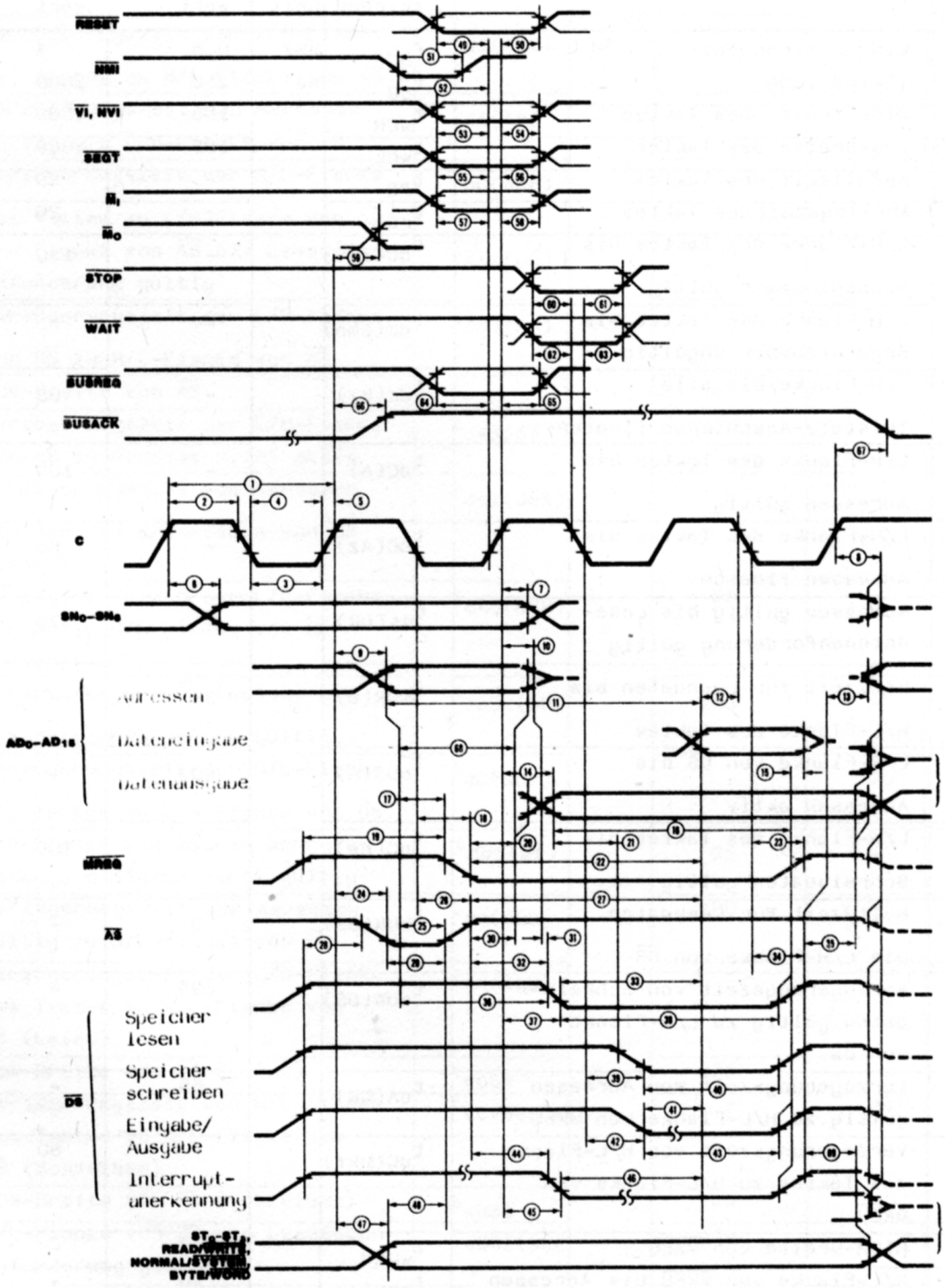
Kenngröße	Kurzzeichen	Einheit	Kleinstwert	Größtwert
Betriebsspannung	U_{CC}	V	-0,5	7
Eingangsspannung	U_I		-0,5	7
Lagerungstemperaturbereich	ϑ_{stg}	°C	-55	125

Betriebsbedingungen

Statische Kennwerte

Kenngröße	Kurzzeichen	Einheit	Kleinstwert	Größtwert
Betriebsspannung	U_{CC}	V	4,75	5,25
Eingangsspannung	U_{IL}		-0,5	0,8
	U_{IH}		2	U_{CC}+0,3
Eingangsspannung HIGH am $\overline{\text{RESET}}$-Anschluß	U_{IHRES}		2,4	U_{CC}+0,3
Takteingangsspannung	U_{ICL}		-0,5	0,45
	U_{ICH}		U_{CC}-0,4	U_{CC}+0,3
Betriebstemperaturbereich	ϑ_a	°C	0 bis 70	

Dynamische Kennwerte



Die Zeitmessungen erfolgen bei folgenden Spannungen:

	High	Low		High	Low
Takt	4 V	0,8 V	Eingang	2 V	0,8 V
Ausgang	2 V	0,8 V	Floaten	Δ L	+ 0,5 V

Nr.	Kenngröße	Kurz-zeichen	Ein-heit	Kleinst-wert	Größt-wert	Bemerk.
	Eingangsfrequenz	f_{OC}	MHz	0,5	4	
1	Taktperiode	t_{cC}	ns	250	2000	
2	HIGH-Breite des Taktes	t_{wCH}		105	2000	
3	LOW-Breite des Taktes	t_{wCL}		105	2000	
4	Abfallzeit des Taktes	t_{fC}		-	20	
5	Anstiegszeitdes Taktes	t_{rC}		-	20	
6	L/H-Flanke des Taktes bis Segmentnummer gültig	$t_{dC(SNv)}$		-	130	
7	L/H-Flanke des Taktes bis Segmentnummer ungültig	$t_{dC(SNn)}$		20	-	
8	L/H-Flanke bis alle Tristate-Anschlüsse floaten	$t_{dC(Bz)}$		-	65	
9	L/H-Flanke des Taktes bis Adressen gültig	$t_{dC(A)}$		-	100	
10	L/H-Flanke des Taktes bis Adressen floaten	$t_{dC(Az)}$		-	65	
11	Adressen gültig bis Lese-datenanforderung gültig	$t_{dA(DR)}$		-	475	x
12	Setzzeit für Lesedaten bis H/L-Flanke des Taktes	$t_{sDR(C)}$		30	-	
13	L/H-Flanke von $\overline{DS}$ bis Adressen aktiv	$t_{dDS(A)}$		80	-	x
14	L/H-Flanke des Taktes bis Schreibdaten gültig	$t_{dC(DW)}$		-	100	
15	Haltezeit für Lesedaten bis L/H-Flanke von $\overline{DS}$	$t_{hDR(DS)}$		0	-	
16	Verzögerungszeit von Schreib-daten gültig zu L/H-Flanke von $\overline{DS}$	$t_{dDW(DS)}$		295	-	x
17	Verzögerungszeit von Adressen gültig zu H/L-Flanke von $\overline{MREQ}$	$t_{dA(MR)}$		55	-	x
18	Verzögerungszeit von H/L-Flanke des Taktes zu H/L-Flanke von $\overline{MREQ}$	$t_{dC(MRf)}$		-	80	
19	HIGH-Breite von $\overline{MREQ}$	t_{wMRH}		210	-	x
20	H/L-Flanke von $\overline{MREQ}$ bis Adressen nicht aktiv	$t_{dMR(A)}$		70	-	x
21	Verzögerungszeit von Schreib-daten gültig zu H/L-Flanke von $\overline{MREQ}$	$t_{dDW(DSW)}$		55	-	x
22	H/L-Flanke von $\overline{MREQ}$ bis Lese-datenanforderung gültig	$t_{dMR(DR)}$		375	-	x
23	Verzögerungszeit der H/L-Flanke des Taktes zu L/H-Flanke von $\overline{MREQ}$	$t_{dC(MRr)}$		-	80	

Nr.	Kenngröße	Kurz-zeichen	Ein-heit	Kleinst-wert	Größt-wert	Bemerk.
24	Verzögerungszeit der L/H-Flanke des Taktes zu H/L-Flanke von $\overline{AS}$	$t_{dC(ASf)}$	ns	-	80	
25	Verzögerungszeit von Adressen gültig bis L/H-Flanke von $\overline{AS}$	$t_{dA(AS)}$		55	-	x
26	Verzögerungszeit der H/L-Flanke des Taktes zu L/H-Flanke von $\overline{AS}$	$t_{dC(ASr)}$		-	90	
27	L/H-Flanke von $\overline{AS}$ bis Lesedaten-anforderung gültig	$t_{dAS(DR)}$		360	-	x
28	Verzögerungszeit der L/H-Flanke von $\overline{DS}$ zu H/L-Flanke von $\overline{AS}$	$t_{dDS(AS)}$		70	-	x
29	LOW-Breite von $\overline{AS}$	t_{wAS}		85	-	x
30	Verzögerungszeit der L/H-Flanke von $\overline{AS}$ zu Adresse nicht aktiv	$t_{dAS(A)}$		70	-	x
31	Verzögerungszeit von Adressen floaten bis H/L-Flanke von $\overline{DS}$ (Lesen)	$t_{dAz(DSR)}$		0	-	
32	Verzögerungszeit der L/H-Flanke von $\overline{AS}$ zu H/L-Flanke von $\overline{DS}$ (Lesen)	$t_{dAS(DSR)}$		80	-	x
33	H/L-Flanke von $\overline{DS}$ (Lesen) bis Lesedatenanforderung gültig	$t_{dDSR(DR)}$		205	-	x
34	Verzögerungszeit der H/L-Flanke des Taktes zu L/H-Flanke von $\overline{DS}$	$t_{dC(DSr)}$		-	70	
35	L/H-Flanke von $\overline{DS}$ bis Schreib-daten und STATUS nicht gültig	$t_{dDS(DW)}$		75	-	x
36	Verzögerungszeit von Adressen gültig zu H/L-Flanke von $\overline{DS}$	$t_{dA(DSR)}$		180	-	x
37	Verzögerungszeit der L/H-Flanke des Taktes zu H/L-Flanke von $\overline{DS}$ (Lesen)	$t_{dC(DSR)}$		-	120	
38	LOW-Breite von $\overline{DS}$ (Lesen)	t_{wDSR}		275	-	x
39	Verzögerungszeit von H/L-Flanke des Taktes zu H/L-Flanke von $\overline{DS}$ (Schreiben)	$t_{dC(DSW)}$		-	95	
40	LOW-Breite von $\overline{DS}$ (Schreiben)	t_{wDSW}		185	-	x
41	H/L-Flanke von $\overline{DS}$ bis Lesedaten-anforderung gültig	$t_{dDSI(DR)}$		330	-	x
42	Verzögerungszeit der H/L-Flanke des Taktes zu H/L-Flanke von $\overline{DS}$ (E/A)	$t_{dC(DSf)}$		-	120	
43	LOW-Breite vnn $\overline{DS}$ (Ein-/Aus-gabe)	t_{wDS}		410	-	x
44	Verzögerungszeit der H/L-Flanke von $\overline{AS}$ zu H/L-Flanke von $\overline{DS}$ Anerkennung	$t_{dAS(DSA)}$		1065	-	x
45	Verzögerungszeit der L/H-Flanke des Taktes zu H/L-Flanke von $\overline{DS}$ (Anerkennung)	$t_{dC(DSA)}$		-	120	

Nr.	Kenngröße	Kurz-zeichen	Ein-heit	Kleinst-wert	Größt-wert	Bemerk.
46	Verzögerungszeit der H/L-Flanke von $\overline{DS}$ (Anerkennung) zu Lesedaten-anforderung	$t_{dDSA(DR)}$	ns	455	-	x
47	Verzögerungszeit der H/L-Flanke des Taktes zu STATUS gültig	$t_{dC(S)}$		-	110	
48	Verzögerungszeit von STATUS gültig zu L/H-Flanke von $\overline{AS}$	$t_{dS(AS)}$		50	-	x
49	Setzzeit für $\overline{RESET}$ bis L/H-Flanke des Taktes	$t_{sR(C)}$		180	-	
50	Haltezeit für $\overline{RESET}$ bis L/H-Flanke des Taktes	$t_{hR(C)}$		0	-	
51	LOW-Breite von $\overline{NMI}$	t_{wNMI}		100	-	
52	Setzzeit für $\overline{NMI}$ bis L/H-Flanke des Taktes	$t_{sNMI(C)}$		140	-	
53	Setzzeit für $\overline{VI}$, $\overline{NVI}$ bis L/H-Flanke des Taktes	$t_{sVI(C)}$		110	-	
54	Haltezeit für $\overline{VI}$, $\overline{NMI}$ bis L/H-Flanke des Taktes	$t_{hVI(C)}$		20	-	
55	Setzzeit für $\overline{SEGT}$ bis L/H-Flanke des Taktes	t_{sSGT}		70	-	
56	Haltezeit für $\overline{SEGT}$ bis L/H-Flanke des Taktes	t_{hSGT}		0	-	
57	Setzzeit für $\overline{MI}$ bis L/H-Flanke des Taktes	$t_{sMI(C)}$		180	-	
58	Haltezeit für $\overline{MI}$ bis L/H-Flanke des Taktes	$t_{hMI(C)}$		0	-	
59	Verzögerungszeit der L/H-Flanke des Taktes bis $\overline{MO}$	$t_{dC(MO)}$		-	120	
60	Setzzeit für $\overline{STOP}$ bis H/L-Flanke des Taktes	$t_{sSTP(C)}$		140	-	
61	Haltezeit für $\overline{STOP}$ bis H/L-Flanke des Taktes	$t_{hSTP(C)}$		0	-	
62	Setzzeit für $\overline{WAIT}$ bis H/L-Flanke des Taktes	$t_{sW(C)}$		50	-	
63	Haltezeit für $\overline{WAIT}$ bis H/L-Flanke des Taktes	$t_{hW(C)}$		10	-	
64	Setzzeit für $\overline{BUSRQ}$ bis L/H-Flanke des Taktes	$t_{sBRQ(C)}$		90	-	
65	Haltezeit für $\overline{BUSRQ}$ bis L/H-Flanke des Taktes	$t_{hBRQ(C)}$		10	-	
66	Verzögerungszeit der L/H-Flanke des Taktes zu L/H-Flanke von $\overline{BUSACK}$	$t_{dC(BAKr)}$		-	100	

Nr.	Kenngröße	Kurz-zeichen	Ein-heit	Kleinst-wert	Größt-wert	Bemerk.
67	Verzögerungszeit der L/H-Flanke des Taktes zu H/L-Flanke von $\overline{BUSACK}$	$t_{dC(BAKf)}$	ns	-	100	
68	gültige Adressenbreite	t_{wA}		150	-	x
69	L/H-Flanke von $\overline{DS}$ bis STATUS nicht gültig	$t_{dDS(S)}$		80	-	x

Berechnung der vom Takt abhängigen Zeiten (in der Tabelle durch x gekennzeichnet)

Nr.	Symbol	Gleichung
11	$t_{dA(DR)}$	$2t_{cC}+t_{wCH}-130$ ns
13	$t_{dDS(A)}$	$t_{wCL}-25$ns
16	$t_{dDW(DS)}$	$t_{cC}+t_{wCH}-60$ns
17	$t_{dA(MR)}$	$t_{wCH}-50$ns
19	t_{wMRH}	$t_{cC}-40$ns
20	$t_{dMR(A)}$	$t_{wCL}-35$ns
21	$t_{dDW(DSW)}$	$t_{wCH}-50$ns
22	$t_{dMR(DR)}$	$2t_{cC}-130$ns
25	$t_{dA(AS)}$	$t_{wCH}-50$ns
27	$t_{dAS(DR)}$	$2t_{cC}-140$ns
28	$t_{dDS(AS)}$	$t_{wCL}-35$ns
29	t_{wAS}	$t_{wCH}-20$ns
30	$t_{dAS(A)}$	$t_{wCL}-35$ns
32	$t_{dAS(DSR)}$	$t_{wCL}-25$ns
33	$t_{dDSR(DR)}$	$t_{cC}+t_{wCH}-150$ns
35	$t_{dDS(DW)}$	$t_{wCL}-30$ns
36	$t_{dA(DSR)}$	$t_{cC}-70$ns
38	t_{wDSR}	$t_{cC}+t_{wCH}-80$ns
40	t_{wDSW}	$t_{cC}-65$ns
41	$t_{dDSI(DR)}$	$2t_{cC}-170$ns
43	t_{wDS}	$2t_{cC}-90$ns
44	$t_{dAS(DSA)}$	$4t_{cC}+t_{wCL}-40$ns
46	$t_{dDSA(DR)}$	$2t_{cC}+t_{wCH}-150$ns
48	$t_{ds(AS)}$	$t_{wCH}-55$ns
68	t_{wA}	$t_{cC}-90$ns
69	$t_{dDS(S)}$	$t_{wCL}-25$ns

Bestellbezeichnung	
Schaltkreis UB 8001C	TGL 43019
Schaltkreis UB 8002D	TGL 43019

Bemerkungen und Verbesserungsvorschläge des Lesers

Werter Leser!

Bei unseren technischen Beschreibungen legen wir Wert auf korrekte, vollständige, verständliche und praxisgerechte Darstellung. Bitte teilen Sie uns ihre Meinung zu der vorliegenden technischen Beschreibung mit. Sie helfen damit, eventuelle Fehler zu beseitigen und die Beschreibung weiter zu verbessern. Dabei interessieren uns besonders umseitig gestellte Fragen.

Ihre Hinweise richten Sie bitte an:

veb mikroelektronik "karl marx" erfurt
Applikation Bauelemente
Rudolfstr. 47
Erfurt
5010

Bemerkungen und Verbesserungsvorschläge des Lesers

Beschreibung:

Titel: Technische Beschreibung CPU U 8001/U 8002
Ausgabe: 2/87

Hinweise des Lesers:

1. Sind Ihnen Druckfehler oder inhaltliche Fehler aufgefallen?

2. Enthält die Beschreibung die von Ihnen erwarteten und benötigten Informationen? Haben Sie Ergänzungsvorschläge?

3. Ist die Beschreibung für Sie verständlich? Wenn Sie Verständnisprobleme hatten, teilen Sie uns bitte das Kapitel und kurz den Sachverhalt mit.

4. Entsprechen Niveau, Umfang und Form dieser Beschreibung Ihren Anforderungen?

Absender:

Betrieb: .....
Anschrift: .....
Leser: .....
Telefon: .....
Datum: .....

RFT

veb mikroelektronik ›karl marx‹ erfurt
stammbetrieb

DDR – 5010 Erfurt, Rudolfstraße 47 Telefon 580 Telex 061306

elektronik
export·import
Volkseigener Außenhandelsbetrieb der
Deutschen Demokratischen Republik
DDR - 1026 Berlin, Alexanderplatz 6
Telex: BLN 114721 elei, Telefon: 2180