mikroelektronik

Integrierte
Schaltkreise

RFT

Hinweise zur Anwendung

Speicherverwaltungseinheit MMU U 8010

Applikation

Technische Beschreibung

Speicherverwaltungseinheit MMU

U8010

veb mikroelektronik ›karl marx‹ erfurt
stammbetrieb

DDR – 5010 Erfurt, Rudolfstraße 47 Telefon 580 Telex 061306

Inhalt

Anhänge

0. Einleitung

In diesem Handbuch wird die Speicherverwaltungseinheit U8010 umfassend beschrieben. Dieses Bauelement ist ein Ergänzungsschaltkreis für die CPU U8001. Der Anwender sei deshalb auch auf folgende technische Handbücher verwiesen:

"CPU U8001/U8002 - Technische Beschreibung"
"CPU U8001/U8002 - Befehlsbeschreibung"

Das technische Handbuch der MMU U8010 wurde in 8 Kapitel und 7 Anhänge unterteilt:

Kapitel 1 beeinhaltet eine allgemeine Beschreibung des Speicherverwaltungkonzepts.

Kapitel 2 enthält einen Überblick über Architektur und Leistungsfähigkeit der U8010.

Kapitel 3 beschreibt vollständig die Register und Flags der U8010 und deren Funktionen.

Kapitel 4 analysiert die in der U8010 verwendeten Adreßübersetzungsmethoden.

Kapitel 5 beschreibt Zugriffsfehler und Schreibwarnungen, sowie die Bedingungen, unter denen die U8010 Trap- oder Suppress-Signale generiert.

Kapitel 6 erläutert die Programmierung der U8010 und beschreibt die Kommandos.

Kapitel 7 beschreibt detaillierter einige Besonderheiten (DMA, Reset, DSCR, Zusammenarbeit mehrerer MMUs).

Kapitel 8 stellt zwei Beispielsysteme mit U8010-MMUs vor.

Anhang A beschreibt die fünf internen Zustände der U8010 und die Bedingungen, die einen Zustandswechsel auslösen.

Anhang B erläutert das Zeitverhalten des Bauelements.

Anhang C gibt eine Anschlußbeschreibung.

Anhang D enthält eine Zusammenstellung der Register und Flags.

Anhang E führt alle Kommandos auf.

Anhang F faßt die wesentlichen elektrischen Parameter zusammen.

1. Allgemeine Beschreibung

1.0. Einführung

Die MMU U8010 (MMU - memory management unit - Speicherverwaltungseinheit) verwaltet die 8-MByte-Adreßräume der CPU U8001. Jeder U8001-Adreßraum besteht aus bis zu 128 Segmenten, deren Größe im Bereich von 256 Bytes bis zu 64 KBytes liegen kann und die in Schritten von 256 Bytes gestaffelt sind. Die MMU weist diesen Segmenten den physischen Platz im Speicher zu und führt verschiedene Speicherschutzfunktionen aus.
Segmentierte Adreßräume, wie vom U8001 realisiert, besitzen entscheidende Vorteile gegenüber den konventionellen linearen Adreßräumen. Die segmentierten Adreßräume gestatten, daß sich die einzelnen Programmoduln und Datenfelder in jeweils eigenen Segmenten befinden. Aus diesem Grund sind diese besonders für moderne modulare Programmiertechniken geeignet. Darüberhinaus erlaubt die Möglichkeit des Segmentschutzes die Festlegung zweckmäßiger Schnittstellen zwischen den Moduln. Dies ist eine wesentliche Forderung komplexer Systeme mit einer großen Anzahl separater Moduln, wobei jedes seine eigene Schutzfunktion fordert.

1.1. Vorteile der Speicherverwaltung

Die Vorteile der Speicherverwaltung sind in zwei Kriterien zu sehen:

- Verschiebbarkeit
- Schutzfunktion

1.1.1. Verschiebbarkeit

Segmente werden verschiebbar (relokativ) genannt, wenn sie sich zu verschiedenen Zeitpunkten in verschiedenen Bereichen des physischen Speichers befinden können. Dabei ist es auch möglich, daß sich Segmente zeitweilig mehrfach im Speicher befinden. Diese dynamische Verschiebbarkeit der Segmente macht die Softwareadressen unabhängig von den physischen Speicheradressen und befreit dabei den Nutzer von der Notwendigkeit zu spezifizieren, wo sich die Informationen in Wirklichkeit im physischen Speicher befinden. Die Unabhängigkeit der vom Nutzer definierten logischen Adressen von den wirklichen physischen Adressen bringt vor allem für Multiprogrammsysteme bedeutende Vorteile. Verschiebbarkeit ist immer dann wünschenswert, wenn ein System aus verschiedenen mehr oder weniger miteinander verknüpften Tasks besteht (z.B. mehrere Nutzer in einem System oder mehrere Tasks innerhalb einer Anwendung). Unter einer Task versteht man hier die Ausführung eines Programms zur Bearbeitung seiner Daten. In der Regel werden nicht alle Tasks gleichzeitig im Speicher gebraucht. Wenn sie dann tatsächlich benötigt werden, ist es vorteilhaft, wenn man sie an beliebigen (gerade freien) Stellen im Speicher plazieren kann. Durch die Zuordnung von Tasks zu entsprechenden Segmenten, kann man mit der dynamischen Segmentverschiebbarkeit auch relokative Tasks erreichen.
Um die Verschiebbarkeit praktisch zu realisieren, wird in der MMU die Technik der Adreßübersetzung angewendet. Darunter versteht man den Prozeß des Zuordnens der, von der CPU gelieferten, logischen Adresse zu einer physischen Adresse im Speicher. Die Adreßübersetzung befreit die Anwendersoftware nicht nur davon, die aktuelle Speicherorganisation im einzelnen festzulegen, sie kann auch die Leistungsfähigkeit und Flexibilität des Systems entscheidend erhöhen. Zum Beispiel kann die schwerfällige Technik des Reservierens fester Speicherräume für Overlays (evtl. nachzuladende Programmsektionen) durch eine sehr schnelle Routine ersetzt werden, die nur die MMU umprogrammieren muß. Bild 1.1 illustriert einige grundlegende Gedanken. Zwei Anwender werden gezeigt,

wobei jeder über mehrere Segmente verfügt. In diesem Beispiel befinden sich alle Segmente gleichzeitig im Speicher. Es ist nicht notwendig, daß die zu einem Nutzer gehörenden Segmente aufeinanderfolgend im Speicher angeordnet sind. Es müssen sich auch nicht alle Segmente gleichzeitig im Speicher befinden. Der Nutzer kann das Betriebssystem anweisen, bestimmte Programme und deren Daten nur dann vom Sekundärspeicher (z.B. Plattenspeicher) in den Primärspeicher (Hauptspeicher) zu laden, wenn sie gebraucht werden. Würde z.B. ein Editor oder Compiler einem eigenen Segment zugeordnet, so könnte dieses solange auf der Platte bleiben, bis ihn der Nutzer aufruft. Damit würde veranlaßt, daß dieses Segment in den Hauptspeicher geladen wird.
Die Trennung der logischen von den physischen Adressen erleichtert auch die Organisation des Zugriffs mehrerer Nutzer auf gemeinsame Programme und Daten, da zwei oder mehrere logische Adressen durch ein und dieselbe physische Adresse repräsentiert werden können. Z.B. wäre es möglich, daß sich zwei Nutzer einen Compiler teilen, ohne daß jeder eine Kopie dieses Programms im Speicher benötigt. Im Bild 1.1 teilen sich Nutzer A und B das Segment, welches den PASCAL-Compiler enthält. Dasselbe physische Segment wird durch unterschiedliche logische Segmentnummern angesprochen.

1.1.2. Schutzfunktion

Da die MMU zwischen der CPU und dem Speicher angeordnet ist, können logische Adressen auf verschiedene Fehlerarten untersucht werden, bevor sie in physische Adressen übersetzt werden. So kann z.B. der Zugriff eines Nutzers auf geschützte Daten eines anderen Nutzers verhindert werden. Jedem Segment können bestimmte Attribute zugeordnet werden, die festlegen, für wen ein Zugriff gestattet ist und welcher Art dieser Zugriff sein darf. Jede Speicheroperation wird überprüft, um sicherzustellen, daß die Task berechtigt ist, auf diese logische Adresse in der gegebenen Form zuzugreifen.
Es gibt unter anderem folgende Attribute:
"Nur lesbar" (Read-Only) - Dieses Attribut kann verwendet werden, um eine Veränderung von Datensegmenten zu verhindern. Es kann auch auf Programmsegmente angewendet werden, die sich nicht selbst verändern. Damit läßt sich die Datensicherheit erhöhen.
"Nur ausführbar" (Execute-Only) - Dieses Attribut kann für Programmsegmente gesetzt werden, um ihr unerlaubtes Lesen und Kopieren zu verhindern. Es ermöglicht den Schutz von Eigentumsrechten an Software.
"Nur Systemmodus" (System-Only) - Das Attribut dient zum Schutz wichtiger Systemfunktionen (z.B. Peripheriesteuerprogramme) vor unerlaubtem Zugriff durch nicht privilegierte Nutzer und damit zur Erhöhung der Systemsicherheit.
Im Bild 1.1 ist die Anwendung der Attribute "Read-Only" und "Execute-Only" demonstriert. Stellt die MMU während eines Speicherzugriffs eine Verletzung dieser Segmentattribute fest, sendet sie eine spezielle Unterbrechungsanforderung an die CPU, welche als Segmenttrap-Anforderung bezeichnet wird (vgl. Kapitel 5.). Die MMU kann dabei auch ein Signal generieren (Suppress), das von der Speichersteuerung verwendet werden kann, um ein Beschreiben des Speichers mit fehlerhaften Daten zu verhindern.

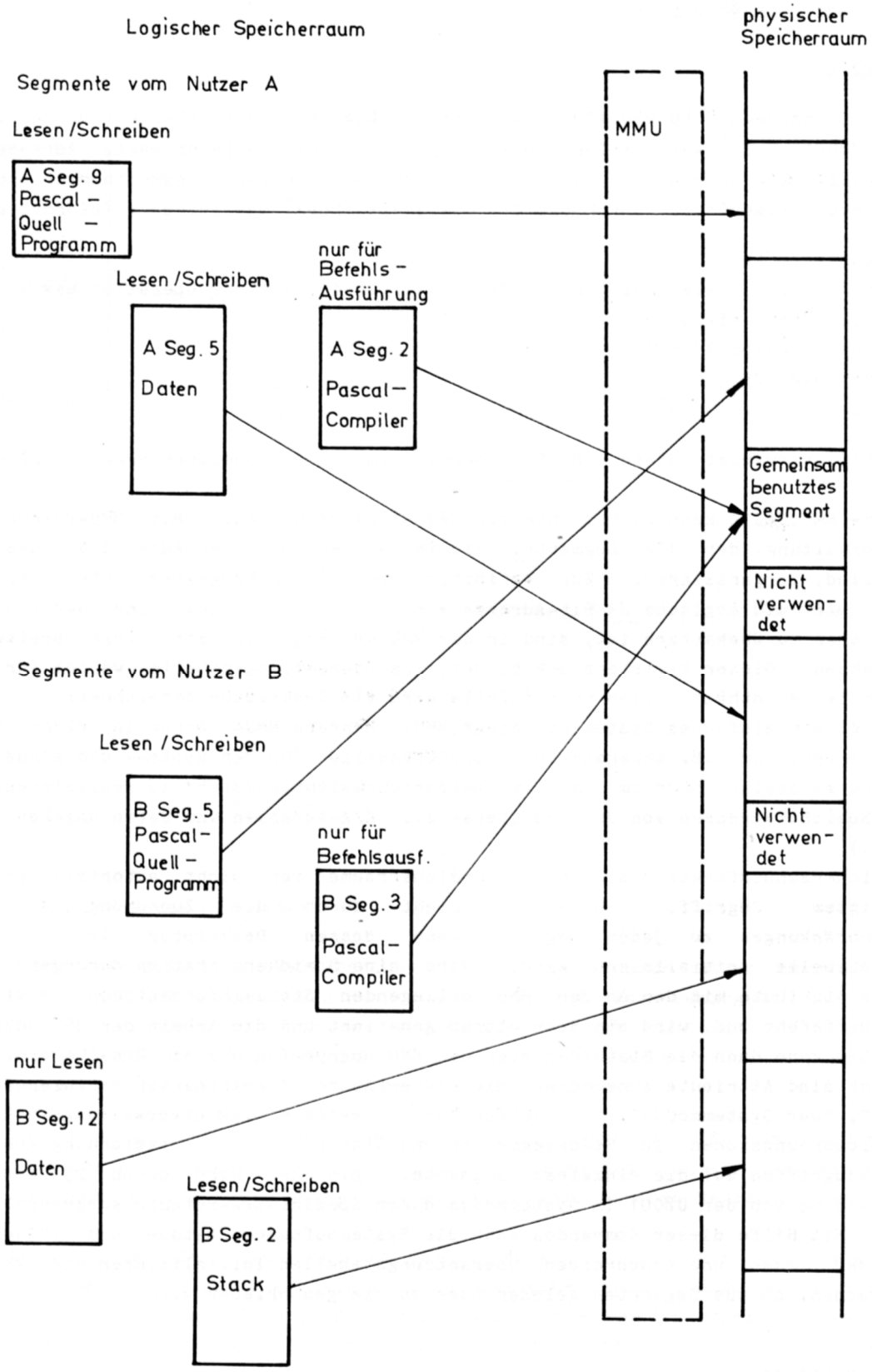


Bild 1.1: Grundgedanken der Speicherverwaltung

2. Überblick über die Architektur

2.0. Einführung

Wie bereits im Handbuch "CPU U8001/U8002-Technische Beschreibung" erläutert wurde, werden durch die U8001 bei Buszugriffen Adressen und Statussignale erzeugt. Adressen für Speicherzugriffe sind segmentiert, d.h. sie bestehen aus der 7-Bit-Segmentnummer und der 16-Bit-Segmentoffsetadresse. Durch die Statussignale erhält man folgende Informationen:

- Typ des Buszugriffs
  (Befehlsholezyklus, Datenspeicherzugriff, Stackzugriff, interne Operation usw.)
- Richtung des Buszugriffs
  (Lese- oder Schreibzugriff),
- Betriebsart der CPU
  (Normal- oder Systemmodus)

Die MMU U8010 verwendet diese Informationen, um die Speicherverwaltungsfunktion auszuführen.
Eine einzelne MMU U8010 kann 64 Segmente der CPU U8001 verwalten. Mit MMU-Paaren kann man die Verwaltung der 128 Segmente, die in den verschiedenen Adreßräumen des U8001 verfügbar sind, organisieren. Zur Definition der Segmentzugriffsrechte und zur Festlegung, wie die logische 23-Bit-Adresse von der CPU in die physische 24-Bit-Adresse für den Speicher zu übersetzen ist, sind in der MMU 64 Register mit einer Breite von 32 Bit vorhanden. Dieser Registerblock bildet eine Übersetzungstabelle, welche für jedes Segment eine Zeile enthält. Eine solche Zeile wird als Deskriptor bezeichnet.
Bild 2.1 zeigt ein einfaches System mit einer MMU. Mehrere MMUs können in einem System verwendet werden, um z.B. separate Übersetzungstabellen für den System- und Normalmodus zur Verfügung zu stellen oder um komplexe Speicherverwaltungssysteme zu realisieren. (Es werden nur Speicheradressen von der MMU übersetzt. E/A-Adressen und Daten umgehen diesen Schaltkreis.)
Die MMU-Speicherschutzfunktion sichert die Speicherräume vor nicht autorisiertem oder unbeabsichtigtem Zugriff. Dies geschieht durch die Zuordnung spezieller Zugriffsbeschränkungen zu jedem Segment, wenn dessen Deskriptor in der MMU-Übersetzungstabelle initialisiert wird. Wird eine Speicheroperation durchgeführt, so werden diese Attribute mit den an der MMU anliegenden Statusinformationen verglichen. Tritt eine Differenz auf, wird ein Segmenttrap generiert und die Arbeit der CPU unterbrochen. Die CPU kann dann die Statusregister der MMU überprüfen und die Ursache ermitteln. Jedem Segment sind Attribute zugeordnet, die die erlaubten Zugriffsarten definieren (z.B. "nur lesbar", "nur Systemmodus", "nur ausführbar"). Weitere Segmentverwaltungsfunktionen sind Schreibwarnungszonen in Stacksegmenten und Statusflags zur Aufzeichnung von Lese- oder Schreibzugriffen auf die einzelnen Segmente. Die MMU wird durch 23 Kommandos gesteuert, welche von der U8001 im Systemmodus durch Spezial-E/A-Befehle ausgesendet werden können. Mit Hilfe dieser Kommandos kann die Systemsoftware Status- und Steuerregister der MMU lesen und beschreiben, Übersetzungstabellen initialisieren und verändern sowie überwachen, ob aus Segmenten gelesen oder in sie geschrieben wurde.

2.1. MMU-Operationen

Die MMU-Eingänge sind mit dem Systembus des U8001-Systems verbunden (vgl. Bild 2.1 und 2.2).

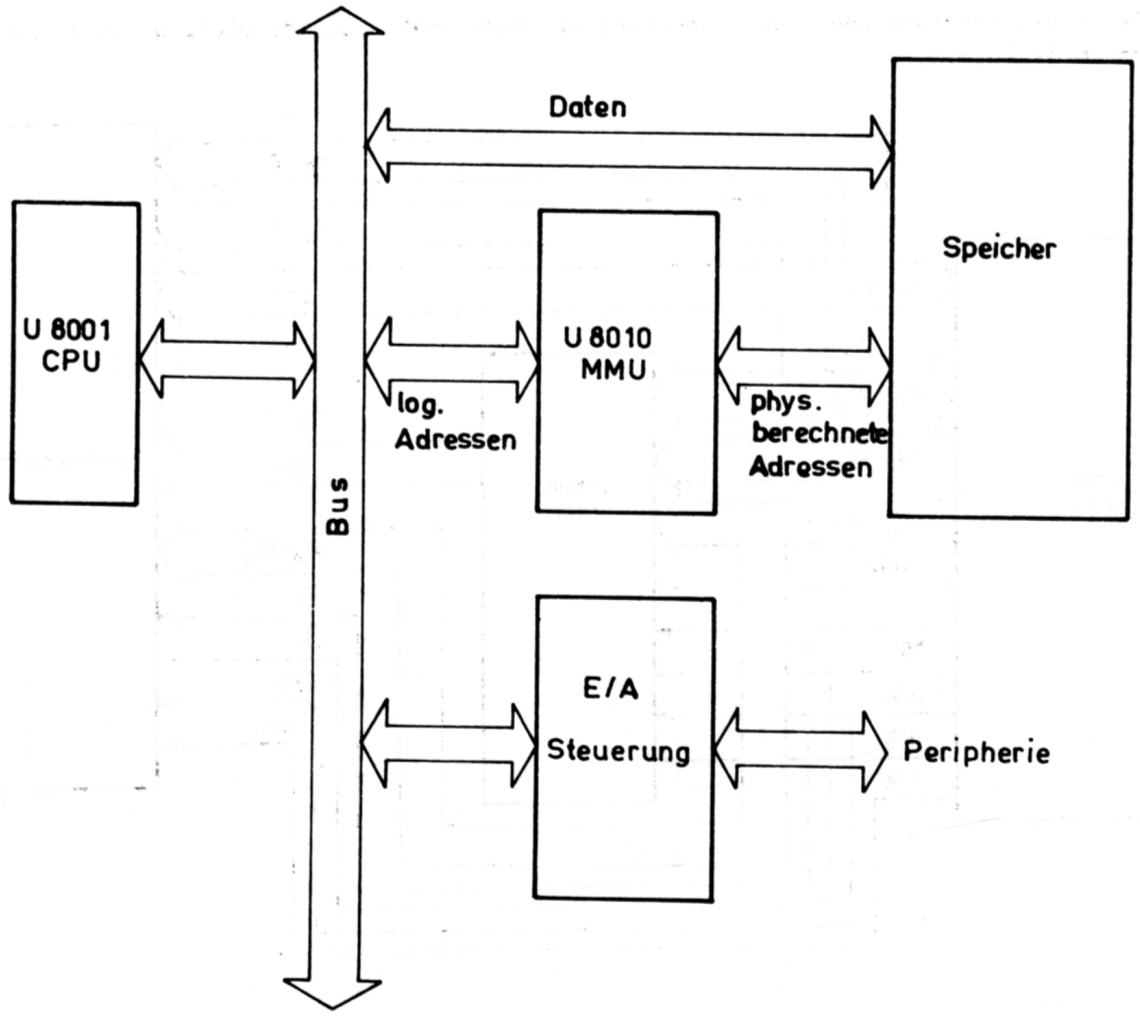


Bild 2.1: U8001-System mit einer MMU

Die MMU überwacht alle Busaktivitäten, welche von der CPU oder einem anderen Busmaster (z.B. DMA) ausgeführt werden. Die Reaktion der MMU auf einen Buszugriff hängt sowohl von dessen Typ ab (signalisiert durch die Statusleitungen), als auch von der Programmierung der MMU (festgelegt durch den Inhalt der internen Register der MMU). Auf Busoperationen eines Busmasters gibt es drei Reaktionsmöglichkeiten der MMU:

1. Sie kann die logische Adresse auf dem Bus in eine physische **Adresse übersetzen** und diese an ihren Ausgängen ($A_0 \ldots A_{23}$) ausgeben. Diese Reaktion wird bei einer Teilmenge der Speicherzugriffe ausgeführt, die vom Inhalt der internen Register der MMU festgelegt wird.
2. Sie kann die Adresse, welche auf dem Bus liegt, als ein für sie bestimmtes **Kommando interpretieren.** Diese Operation wird nur bei einer Spezial-E/A-Operation ausgeführt, wenn der Chip-Auswahl-Eingang ($\overline{CS}$) aktiv ist. Nur so werden Adressen eines Spezial-E/A-Befehls als Kommando interpretiert. Diese Kommandos werden zum Lesen oder zum Verändern des Inhalts der internen MMU-Register verwendet.
3. Sie kann die **Busoperationen ignorieren.** Alle Standard-E/A-Zugriffe, internen CPU-Operationen und Refresh-Zugriffe werden nicht beachtet. Dies geschieht auch bei Speicherzugriffen, die nicht in der von den internen Registern festgelegten Teilmenge enthalten sind sowie bei Spezial-E/A-Operationen, wenn $\overline{CS}$ nicht aktiv ist.

Empfängt die MMU Kommandos oder ignoriert sie Busoperationen, so bleiben ihre Adreßausgänge im Tri-State.

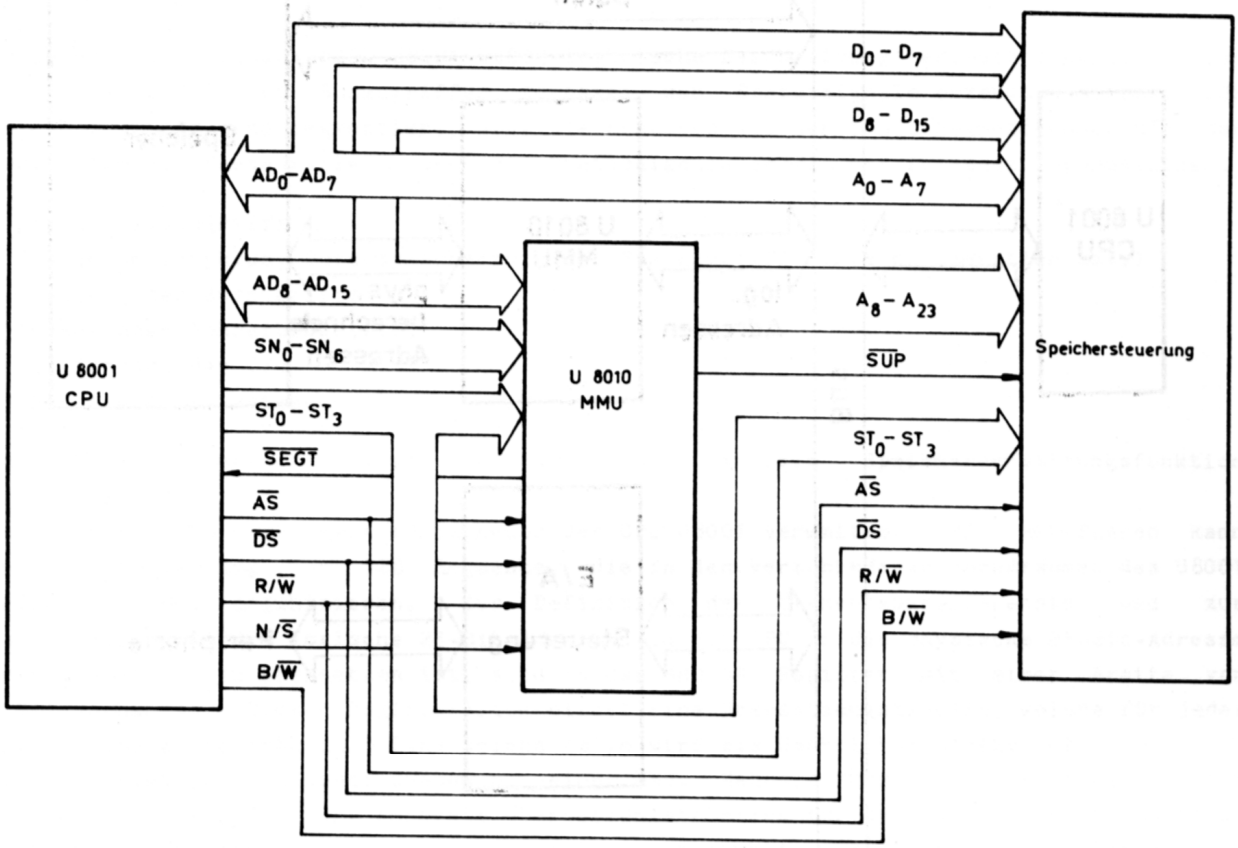


Bild 2.2: MMU U8010 in einem U8000-System

2.1.1. Adreßübersetzung

Bild 2.3 zeigt, welche Sektionen der MMU während der Adreßübersetzung aktiv sind. Wenn eine Busoperation eine Adreßübersetzung veranlaßt, wird der MMU die Adresse auf den Segmentnummer- und den Adreß-/Datenleitungen übergeben. Die MMU gibt die übersetzte, physische Speicheradresse an ihren Adreßleitungen aus und aktiviert bei einer Zugriffsverletzung (Violation oder Schreibwarnung) den Segmenttrap- und/oder Suppress-Ausgang. Im folgenden soll dieser Prozeß genauer beschrieben werden:

- Die Segmentnummer gelangt über die Eingänge $SN_0 \ldots SN_6$ in die MMU. Mit SN_6 wird die MMU als Ganzes selektiert, und mit $SN_0 \ldots SN_5$ wird einer von den 64 in der MMU verwalteten 32-Bit-Segment-Deskriptoren ausgewählt.
- Der Offset der logischen Adresse gelangt über die Eingänge $AD_8 \ldots AD_{15}$ in die MMU und wird zur physischen Basisadresse des entsprechenden Segments addiert. (Die Basisadresse ist Bestandteil des Segment-Deskriptors.) Dadurch wird die eigentliche physische Adresse gebildet. Diese Adresse wird an den Ausgängen $A_8 \ldots A_{23}$ ausgegeben, unabhängig davon, ob eine Zugriffsverletzung festgestellt wurde oder nicht.

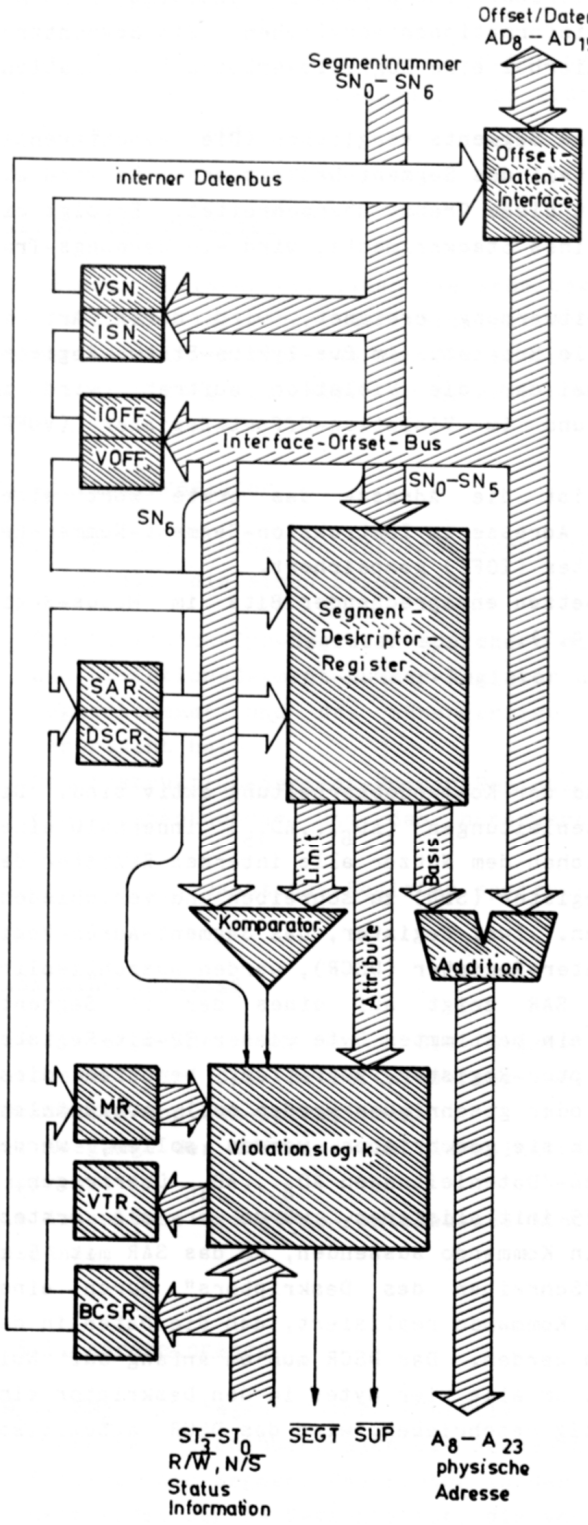


Bild 2.3:
Aktive Sektionen der MMU
bei der Adreßübersetzung

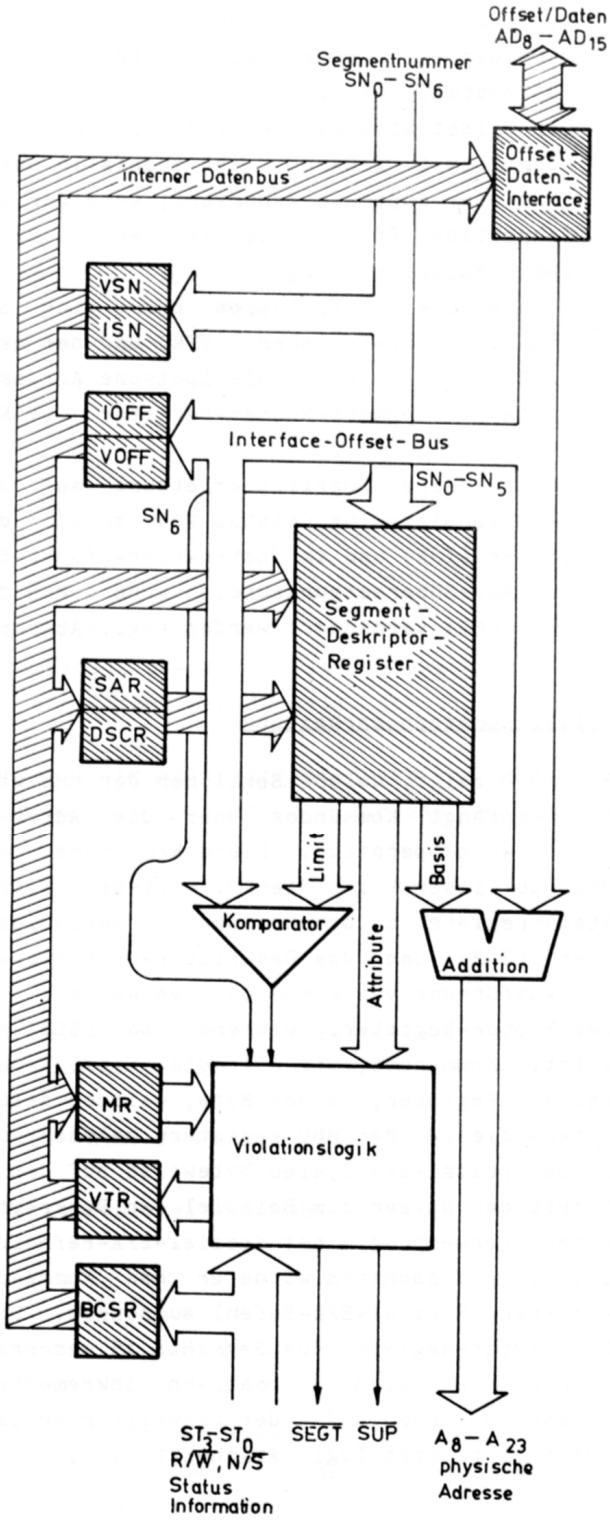


Bild 2.4:
Aktive Sektionen der MMU
bei der Kommandobearbeitung

- Parallel zu der Adreßberechnung werden die Attribute des Segments (ebenfalls Bestandteil des Segment-Deskriptors) mit den Statusinformationen verglichen. Ein Segmenttrap und/oder ein Suppress wird generiert, wenn hierbei eine Zugriffsverletzung (Violation) festgestellt wird.
- Der Offset wird mit der festgelegten Größe des Segments verglichen.(Die Segmentgrenze, auch Limit genannt, ist ebenfalls Bestandteil des Segment-Deskriptors.) Es wird ein Trap oder Suppress generiert, wenn der Offset diese Grenze überschreitet. Erfolgt ein Schreibzugriff in die letzten 256 Bytes eines Stacksegments, wird ein Warnungs-Trap (ohne Suppress) ausgelöst.
- Wenn sich eine Violation oder eine Schreibwarnung ereignet, wird deren Art im Violation-Typ-Register (VTR) und der aktuelle Busstatus im Bus-Zyklus-Status-Register (BCSR) gespeichert. Die logische Adresse, bei der die Violation auftrat, wird im Violation-Segment-Nummer-Register (VSN) und im Violation-Offset-Register (VOFF) abgelegt.
- Tritt keine Zugriffsverletzung auf und ist die Adresse das erste Wort eines Befehlsholezyklus (Status %D), so wird diese Adresse im Instruktion-Segment-Nummer-Register (ISN) und im Instruktion-Offset-Register (IOFF) gespeichert.
- Der Adreßübersetzungsprozeß kann durch das Setzen entsprechender Bits im Modus-Register (MR) verhindert werden (vgl. Abschnitt 4.1).

2.1.2. Kommandobearbeitung

Bild 2.4 zeigt, welche Sektionen der MMU während der Kommandobearbeitung aktiv sind. Die MMU empfängt Kommandos über die Adreß-/Datenleitungen (AD_8...AD_{15}) innerhalb einer Spezial-E/A-Operation. Diese Kommandos ermöglichen dem Nutzer alle internen Register der MMU zu lesen, in die Segment-Deskriptor-Register (SDR) zu schreiben und verschiedene Steuerregister zu beschreiben und zurückzusetzen. Zwei Register, das Segment-Adreß-Register (SAR) und das Deskriptor-Selektion-Counter-Register (DSCR), werden ausschließlich zur Ausführung von Kommandos verwendet. Das SAR zeigt auf eines der 64 Segment-Deskriptor-Register, während das DSCR auf ein bestimmtes Byte dieser 32-Bit-Register zeigt. Kommandos, die auf die Segment-Deskriptor-Register zugreifen, benutzen diese beiden Register, um das Byte, welches gelesen oder geschrieben werden soll, auszuwählen. Daten, die aus den MMU-Registern gelesen oder in sie geschrieben werden sollen, werden durch Spezial-E/A-Zyklen byteweise auf den Adreß-/Datenleitungen AD_8...AD_{15} übertragen. Möchte der Nutzer zum Beispiel den Deskriptor 15 initialisieren, würde er als erstes, unter Verwendung eines Spezial-E/A-Befehls, ein Kommando aussenden, um das SAR mit 15 zu laden. Als nächstes würde er das Kommando "Schreiben des Deskriptors" durch einen weiteren Spezial-E/A-Befehl aussenden. Dieses Kommando realisiert, daß die Daten in das Deskriptor-Register des Segments 15 geschrieben werden. Das DSCR muß am Anfang auf Null stehen. Es wird automatisch inkrementiert, um alle vier Bytes in den Deskriptor eintragen zu können. Ist der Deskriptor vollständig beschrieben, wird das DSCR automatisch auf Null gesetzt (vgl. Abschnitt 7.4).

2.2. Interne Register

Wie vorangehend beschrieben, enthält die MMU interne Register, welche zur Adreßübersetzung, zur Definition der Attribute und zur Steuerung der MMU verwendet werden. Diese Register werden ausführlich im Kapitel 3 beschrieben.

2.2.1. Segment-Deskriptor-Register

Diese Register enthalten alle Informationen für die Adreßübersetzung und den Speicherschutz eines gegebenen Segments. Es gibt 64 dieser Register; eins für jedes von der MMU verwaltete Segment.

2.2.2. Steuerregister

Diese drei Register bestimmen die Arbeitsweise der MMU. Das Modus-Register spezifiziert, bei welcher Teilmenge der Speicherzugriffe die MMU Adreßübersetzungen ausführt. Das Segment-Adreß-Register und das Deskriptor-Selektion-Counter-Register werden bei der Kommandobearbeitung als Zeiger im Deskriptor-Register-Satz verwendet.

2.2.3.Statusregister

Diese Register enthalten Informationen, die verwendet werden können, um die Bedingungen zu ermitteln, die einen Segmenttrap verursacht haben. Es gibt sechs solcher Register:

- Das Violation-Typ-Register speichert die Art des Fehlers.
- Das Violation-Segment-Nummer-Register und das Violation-Offset-Register enthalten die Segmentnummer und das höherwertige Byte des Offset der Adresse, die die Violation verursacht hat.
- Das Instruktion-Segment-Nummer-Register und das Instruktion-Offset-Register enthalten die Segmentnummer und das höherwertige Byte des Offset der Adresse des letzten Befehls.
- Das Bus-Zyklus-Status-Register speichert den Busstatus zum Zeitpunkt des Fehlers.

2.3. Ein-/Ausgänge

Die Eingänge der MMU sind die Segmentnummerleitungen, die Busstatusleitungen und Leitungen für die Buszeitsteuerung sowie spezielle Steuerleitungen für Chip-Auswahl, Rücksetzen und DMA-Operationen.
Die Ausgänge der MMU sind die Adreßleitungen und die Leitungen für Segmenttrap und Suppress.
Die Anschlüsse des Adreß-/Datenbusses sind bidirektional, also sowohl Eingänge als auch Ausgänge.

2.3.1. Adreßeingänge

$SN_0 \ldots SN_6$ - Segmentnummer (H-aktiv)
Diese Leitungen übertragen die Segmentnummer der logischen Adresse. $SN_0 \ldots SN_5$ spezifizieren den Segment-Deskriptor in der MMU und SN_6 bestimmt, ob das entsprechende Segment eines der 64 Segmente ist, die von der MMU verwaltet werden.

$AD_8 \ldots AD_{15}$ - logischer Offset/Daten (H-aktiv, bidirektional, Tri-State)
Diese multiplexten Leitungen werden für die Übertragung sowohl der Kommandos und deren Daten, als auch des höherwertigen Byte des Offset der logischen Adresse verwendet.

2.3.2. Adreßausgänge

$A_8 \ldots A_{23}$ - physische Adresse (H-aktiv, Tri-State)
Diese Adreßleitungen bilden die 16 höherwertigen Bits der physischen Speicheradresse.

2.3.3. Statuseingänge

$ST_0 \ldots ST_3$ - Status (H-aktiv)
Diese Leitungen spezifizieren den Typ des auf dem Bus stattfindenden Zugriffs (vgl. Tabelle 2.1).

Art des Zugriffs	ST_3 ... ST_0 hex	binär	Definition
Interne Operation	%0	0000	
Refresh	%1	0001	
E/A-Zugriff	%2	0010	Standard-E/A
	%3	0011	Spezial-E/A
Interrupt-/Trap-Bestätigung	%4	0100	Segmenttrap
	%5	0101	Nichtmaskierbarer Interrupt
	%6	0110	Nichtvektorisierter Interrupt
	%7	0111	Vektorisierter Interrupt
Speicherzugriff	%8	1000	Datenadreßraum
	%9	1001	Stackadreßraum
	%A	1010	Datenadreßraum (EPU)
	%B	1011	Stackadreßraum (EPU)
	%C	1100	Programmadreßraum, n-tes Wort (IFn) eines Befehls
	%D	1101	Programmadreßraum, erstes Wort (IF1) eines Befehls
EPU-Transfer	%E	1110	
Reserviert	%F	1111	

Tabelle 2.1: Statuskodierung

R/$\overline{W}$ - Lesen/Schreiben (H für Lesen, L für Schreiben)
R/$\overline{W}$ zeigt an, daß die CPU oder eine DMA-Einheit eine Lese- oder Schreiboperation auf den Speicher oder die MMU ausführt.

N/$\overline{S}$ - Normal-/Systemmodus (H für Normalmodus, L für Systemmodus)
N/$\overline{S}$ zeigt an, ob die CPU oder eine DMA-Einheit im Normal- oder Systemmodus arbeitet. Dieser Eingang kann auch verwendet werden, um zwischen MMUs zu unterscheiden, die verschiedenen Phasen der Befehlsbearbeitung zugeordnet sind (z.B Trennung zwischen Programm-, Daten- und Stackzugriffen).

2.3.4. Buszeitsteuerungseingänge

$\overline{AS}$ - Adreß-Strobe (L-aktiv)
Die ansteigende Flanke von $\overline{AS}$ zeigt an, daß $AD_0 \ldots AD_{15}$, $ST_0 \ldots ST_3$, $R/\overline{W}$ und $N/\overline{S}$ gültig sind.

$\overline{DS}$ - Daten-Strobe (L-aktiv)
Dieses Signal steuert den zeitlichen Ablauf der Datenübertragung zwischen der MMU und der CPU.

C - Systemtakt
C ist ein 5V-Einphasentakt und dient als Zeitbasis sowohl für die CPU als auch für die MMU.

2.3.5. Steuereingänge

$\overline{CS}$ - Chip-Auswahl (L-aktiv)
Über diese Leitung wird die MMU für ein Steuerkommando ausgewählt. Das Signal wird im Kommandomodus verwendet. Während der Adreßübersetzung wird dieser Eingang nicht beachtet.

DMASYNC - DMA/Segment-Nummer-Synchronisation-Strobe (H-aktiv)
L-Pegel dieses Signals zeigt die Durchführung eines DMA-Zugriffs an und ein H-Pegel signalisiert die Gültigkeit der Segmentnummer. Bei der Durchführung von CPU-Zyklen muß diese Leitung immer H-Zustand haben.

$\overline{RESET}$ - Reset (L-aktiv)
L-Zustand dieses Signals bewirkt das Rücksetzen der MMU.

2.3.6. Verletzungssignalausgänge

$\overline{SEGT}$ - Segmenttrap-Anforderung (L-aktiv, Open-Drain)
Wenn die MMU eine Violation oder Schreibwarnung erkannt hat, unterbricht sie die Arbeit der U8001 durch ein L-Signal auf dieser Leitung.

$\overline{SUP}$ - Suppress (L-aktiv, Open-Drain)
Wenn eine Violation (keine Schreibwarnung) erkannt wurde, wird dieses Signal während des aktuellen Buszyklus ausgesendet. Es kann durch die Speichersteuerung verwendet werden, um den Speicher vor fehlerhaften Speicherzugriffen zu verriegeln.

2.3.7. Versorgungsspannung

U_{CC} - +5V-Betriebsspannung

U_{SS} - Masse

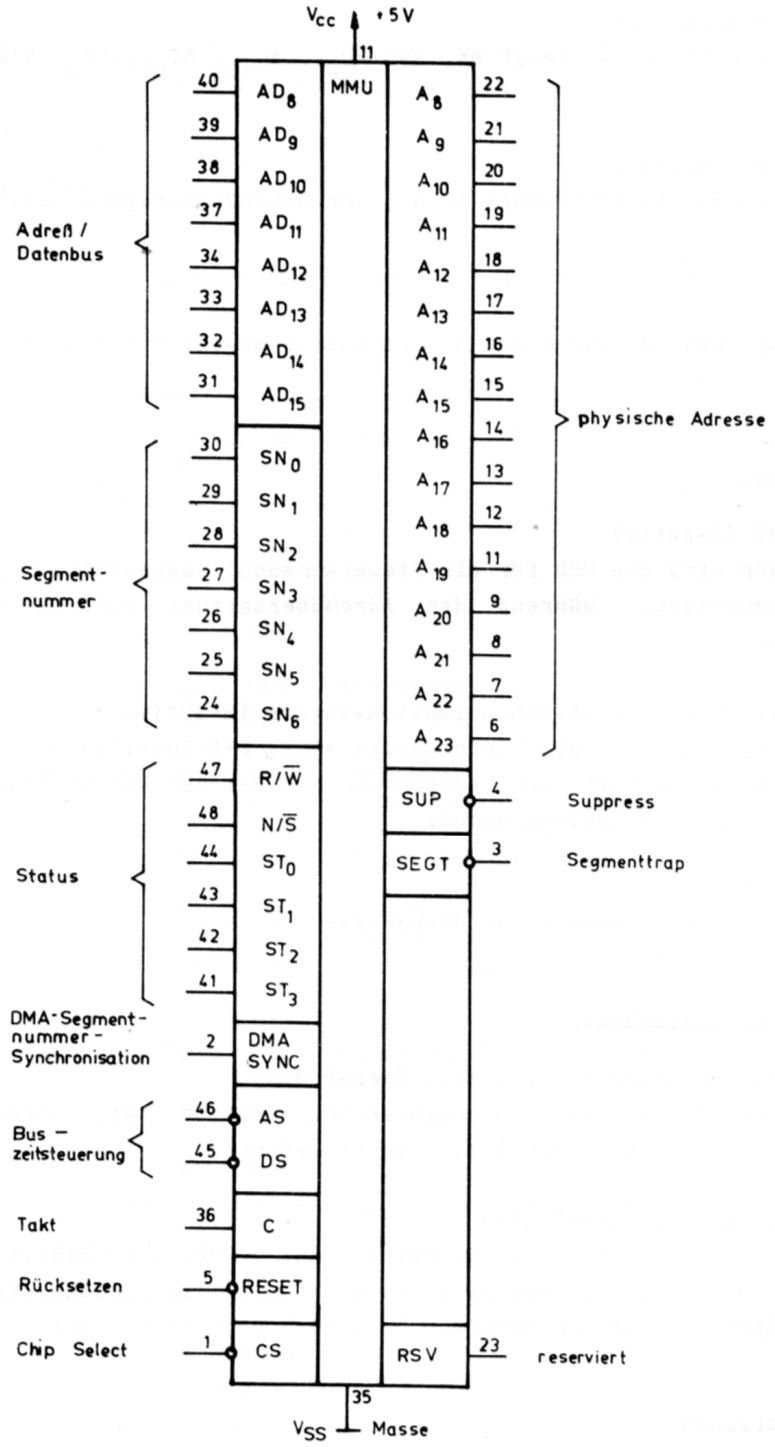


Bild 2.5: Ein-/Ausgänge der MMU U8010

3. Register und Flags

3.0. Einführung

Die MMU U8010 enthält interne Register unterschiedlicher Größe. In einigen dieser Register wurden einzelne Bits ausgewählt und mit separaten Namen versehen. Diese speziellen Bits werden Flags genannt. Man kann den Registersatz in drei Gruppen einteilen:

- Segment-Deskriptor-Register
- Steuerregister
- Statusregister

Es gibt 64 Segment-Deskriptor-Register; eins für jedes durch die MMU verwaltete Segment. Dies sind 32-Bit-Register, wobei jedes Definitionen für die Adreßübersetzung und den Zugriffsschutz eines Segments enthält.
Des weiteren gibt es drei Steuerregister. Eins von ihnen enthält verschiedene Flags, die die MMU an- und abschalten und bestimmen, in welcher Weise die MMU die verschiedenen von der CPU generierten Signale zu interpretieren hat. Die anderen zwei Register werden verwendet, um die Segment-Deskriptoren der MMU zu programmieren.
Außerdem gibt es sechs Statusregister. Diese dienen dazu, alle der MMU verfügbaren Informationen zu speichern, wenn diese eine Speicherzugriffsverletzung oder eine Schreibwarnungsbedingung feststellt. Es werden aufgezeichnet:

- die Ursache der Verletzung oder Warnung
- 15 Bits der logischen Adresse, die die Verletzung oder die Warnung verursacht hat
- 15 Bits der logischen Adresse des letzten Befehls
- der Busstatus zum Zeitpunkt der Verletzung oder der Warnung

3.1. Segment-Deskriptor-Register (SDR)

Jedes der 64 Segment-Deskriptor-Register enthält drei Felder (Bild 3.1):

- ein 16-Bit-Basisadreßfeld
- ein 8-Bit-Limitfeld
- ein 8-Bit-Attributfeld

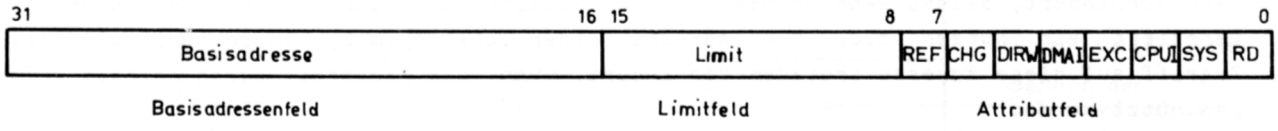


Bild 3.1: Segment-Deskriptor-Register

3.1.1. Basisadreßfeld

Dieses Feld spezifiziert von einem Segment die physische Anfangsadresse im Speicher. Da eine physische Adresse aus 24 Bits besteht, das Basisadreßfeld aber nur 16 Bits enthält, wird die Basisadresse so interpretiert, als wäre ihr niederwertiges Byte Null (Bild 3.2). Das Basisadreßfeld wird bei der Initialisierung oder Modifizierung byteweise geladen.

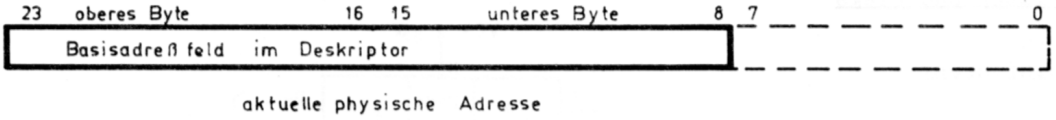


Bild 3.2: Aufbau der physischen Basisadresse

3.1.2. Limitfeld

Dieses Feld spezifiziert die Segmentgröße. Es enthält einen Wert (N), der die Anzahl der 256-Byte-Blöcke angibt, welche dem Segment zugeordnet werden (in Abhängigkeit vom DIRW-Flag entweder N+1 Blöcke oder 256-N Blöcke; vgl. Abschnitt 3.1.3). Das höherwertige Byte eines jeden logischen Adreßoffset wird mit diesem Limit verglichen. Wenn der Offset außerhalb der Segmentgrenze liegt, tritt eine Segmentlängenverletzung auf.

3.1.3. Attributfeld

Dieses Feld enthält acht Flags. Fünf Flags schützen das Segment vor bestimmten Zugriffsarten, ein Flag zeigt eine spezielle Orientierung des Segments an und zwei Flags zeichnen die Art erfolgter Zugriffe auf das Segment auf. Die folgende Beschreibung zeigt die Verwendung eines jeden Flag:

RD - Read-Only (nur lesbar)
Wenn dieses Flag gesetzt ist, kann auf das Segment nur durch eine Speicherleseoperation zugegriffen werden (z.B. im Befehlsholezyklus oder beim Datenlesen). Schreibzugriffe werden verhindert. Dieses Flag ist nützlich, um Daten oder Programme vor Überschreiben zu schützen. Befehle in nur lesbaren Segmenten können ausgeführt werden. Wenn z.B. ein Nutzer auch anderen Nutzern den Zugriff auf kritische Daten erlauben will, kann er mit diesem Attribut seine Daten vor Überschreiben schützen. Dies geschieht durch Setzen des RD-Flag, wenn die Daten von der Platte in den Speicher geladen werden.

SYS - System-Only (nur Systemmodus)
Wenn dieses Bit gesetzt ist, kann auf das Segment nur im Systemmodus zugegriffen werden. Die Benutzung im Normalmodus wird verhindert. Dieses Flag ist nützlich in einem System, in dem eine MMU sowohl vom Betriebssystem, als auch vom Nutzer logische Adressen empfängt. Ist dieses Flag gesetzt, so werden Zugriffe des Nutzers grundsätzlich verhindert, selbst wenn er die korrekte Segmentnummer generieren kann. Kennt und generiert ein Nutzer z.B. die Adresse einer Tabelle im Systemspeicher, wird sein Zugriff auf diese Adresse trotzdem verhindert, wenn das Segment durch dieses Flag geschützt wurde.

CPUI - CPU-Inhibit (CPU-Zugriffe verboten)
Setzt man dieses Flag, wird das Segment sofort (also auch für den gerade laufenden Prozeß) gesperrt und ist damit auch vor weiteren Speicherzugriffen der CPU geschützt. Das Segment ist dann nur für die DMA-Einheit verfügbar. Dieses Flag ist sinnvoll zur Verhinderung von Zugriffen eines Programms auf Segmente, deren Inhalte sich im Sekundärspeicher (z.B. Platte) befinden und noch nicht in den Hauptspeicher geladen wurden. Erfolgt ein CPU-Zugriff auf solch ein Segment, wird ein Trap ausgelöst, und in der darauffolgenden Trap-Behandlung kann das Laden des Segments von der Platte in

den Hauptspeicher veranlaßt werden.

EXC - Execute-Only (nur ausführbar)
Dieses Flag schränkt die Benutzung des Segments auf Befehlsholezyklen ein. Das bedeutet, auf dieses Segment sind nur Zugriffe mit dem Status %C und %D gestattet. Das Segment ist vor Speicherlese- und -schreibzyklen geschützt. Dies hat zur Folge, daß der Inhalt dieses Segments nicht kontrolliert oder modifiziert werden kann. Das EXC-Flag wird verwendet, um das Kopieren geschützter Programme zu verhindern. Ist dieses Flag für ein Segment gesetzt, welches Befehlskode enthält, können die Befehle zwar ausgeführt, der Inhalt aber nicht gelesen oder kopiert werden.

DMAI - DMA-Inhibit (DMA-Zugriffe verboten)
Dieses Flag verbietet DMA-Zugriffe auf das Segment. Zugriffe sind nur der CPU gestattet. Dieses Flag ist nützlich, um die Modifizierung eines Segments, welches von einer befehlsausführenden Task verwendet wird, durch eine DMA-Einheit zu verhindern.

DIRW - Direction and Warning (Richtung und Warnung)
Wenn dieses Flag gesetzt ist, werden die Speicherplätze des Segments in absteigender Ordnung, ausgehend von der logischen Adresse 64K, abwärts bis zur Grenze (Limit) organisiert. Sonst sind sie in ansteigender Ordnung organisiert, ausgehend von der Basisadresse (logische Adresse 0) aufwärts bis zur Grenze (Limit). Daraus ergibt sich:

- Wenn das DIRW-Flag rückgesetzt ist, enthält das Segment (N+1) Blöcke zu je 256 Bytes.
- Ist das DIRW-Flag gesetzt (Stacksegment), enthält das Segment (256-N) Blöcke zu je 256 Bytes.

Beim Schreiben in den niederwertigsten Block eines Segments mit DIRW = 1, generiert die MMU einen Segmenttrap, der vor der akuten Gefahr des Überschreitens der Segmentgrenze warnt (Schreibwarnung). Dabei wird Suppress aber nicht aktiviert. Bild 3.3 erläutert den Unterschied zwischen Segmenten mit gesetztem bzw. rückgesetztem DIRW-Flag.

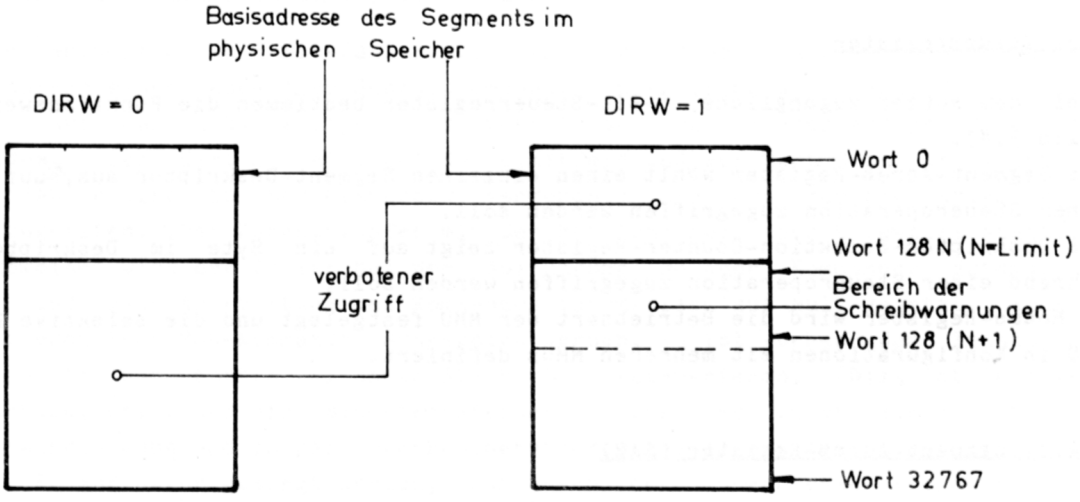


Bild 3.3: Segment mit DIRW = 0 und DIRW = 1

Das DIRW-Flag wird verwendet, wenn Segmente als Stack benutzt werden, denn die CPU U8001 besitzt spezielle Befehle zur Behandlung von Stacks, die in absteigender Ordnung organisiert sind (vgl. Abschnitt 5.4). Eine Schreibwarnung für den Stack bedeutet

also, daß die Gefahr besteht, den zugeteilten Stackspeicherraum zu überschreiten und folglich mehr physischer Speicherplatz erforderlich ist. So kann ein Segment z.B. als dynamischer Stack für blockstrukturierte Programmiersprachen (z.B. PASCAL) organisiert werden. Ihm wird nur soviel Speicher zugewiesen, wie es die Befehlsausführung gerade erfordert. Ohne diese Möglichkeit wäre man gezwungen, für den Stack ständig so viel Speicher freizuhalten, wie das Programm im ungünstigsten Fall einmal benötigen könnte. Dies ist natürlich längst nicht so effektiv wie die Organisation eines dynamischen Stack.

CHG - Changed (verändert)

Ist dieses Flag gesetzt, wurde der Inhalt des entsprechenden Segments durch die CPU verändert (beschrieben). Dieses Bit wird automatisch gesetzt, wenn ein Schreibzugriff auf das Segment erfolgt und dieser keine Violation verursacht. Dieses Flag wird benötigt, wenn Speicherbereiche für andere Aufgaben geräumt werden müssen. Dazu wird ausgewertet, ob Segmente verändert wurden. Segmente, die nicht verändert wurden, brauchen nicht auf die Platte zurückgeschafft werden, denn die auf der Platte existierende Version stimmt dann noch mit dem aktuellen Zustand überein. Wird zum Beispiel die Ausführung einer Task in einem Multi-Task-System unterbrochen, und ist diese Task zeitweilig aus dem Hauptspeicher auszulagern, um Platz für eine andere Task zu schaffen, so brauchen nur die Segmente auf die Platte geschrieben werden, die verändert wurden.

REF - Referenced (benutzt)

Wenn dieses Flag gesetzt ist, wurde auf das Segment durch die CPU zugegriffen (gelesen oder geschrieben). Dieses Bit wird während des Zugriffs auf das Segment automatisch gesetzt, wenn sich keine Violation ereignet. Es wird verwendet, um darauf hinzuweisen, daß das Segment aktiv verwendet wird. Dies ist von Bedeutung, wenn Segmente im Hauptspeicher ausgetauscht werden müssen, um Platz für andere Tasks zu schaffen. Zum Beispiel können selten benutzte Teile des Betriebssystems, die sich im Hauptspeicher befinden, ermittelt und ausgelagert werden, um Platz für Nutzer zu schaffen, die einen größeren Speicherbedarf haben.

3.2. Steuerregister

Drei, dem Nutzer zugängliche, 8-Bit-Steuerregister bestimmen die Funktionsweise der MMU (Bild 3.4).
Das Segment-Adreß-Register wählt einen einzelnen Segment-Deskriptor aus, auf den während einer Steueroperation zugegriffen werden soll.
Das Deskriptor-Selektion-Counter-Register zeigt auf ein Byte im Deskriptor, auf das während einer Steueroperation zugegriffen werden soll.
Im Modus-Register wird die Betriebsart der MMU festgelegt und die selektive Freigabe der MMU in Konfigurationen mit mehreren MMUs definiert.

3.2.1. Segment-Adreß-Register (SAR)

Dieses Register zeigt auf einen der 64 Deskriptoren. MMU-Steuerkommandos, die auf Segment-Deskriptoren zugreifen, verwenden diesen Zeiger, um eins von diesen auszuwählen. Das Register besitzt die Fähigkeit sich nach einem Zugriff automatisch zu inkrementieren, so daß auf mehrere aufeinanderfolgende Deskriptoren unter Verwendung von Block-E/A-Befehlen der CPU U8001 zugegriffen werden kann.
Sollen z.B. die Deskriptoren 0 bis 4 modifiziert werden, wird das SAR mit 0 initialisiert

und damit auf den Deskriptor 0 gerichtet. Bei der Durchführung des Block-E/A-Befehls wird es automatisch erhöht, so daß es nacheinander auch auf die Deskriptoren 1,2,3 und 4 zeigt.
Die Segment-Deskriptor-Nummer ist ein 6-Bit-Feld im SAR, welches die Adresse des Deskriptors innerhalb der MMU definiert. Das Feld beinhaltet die sechs niederwertigen Bits der logischen Segment-Deskriptor-Nummer. Verwaltet die MMU die Segmente 64...127, so muß z.B. beim Zugriff auf das Segment 64 dieses Feld Null sein.

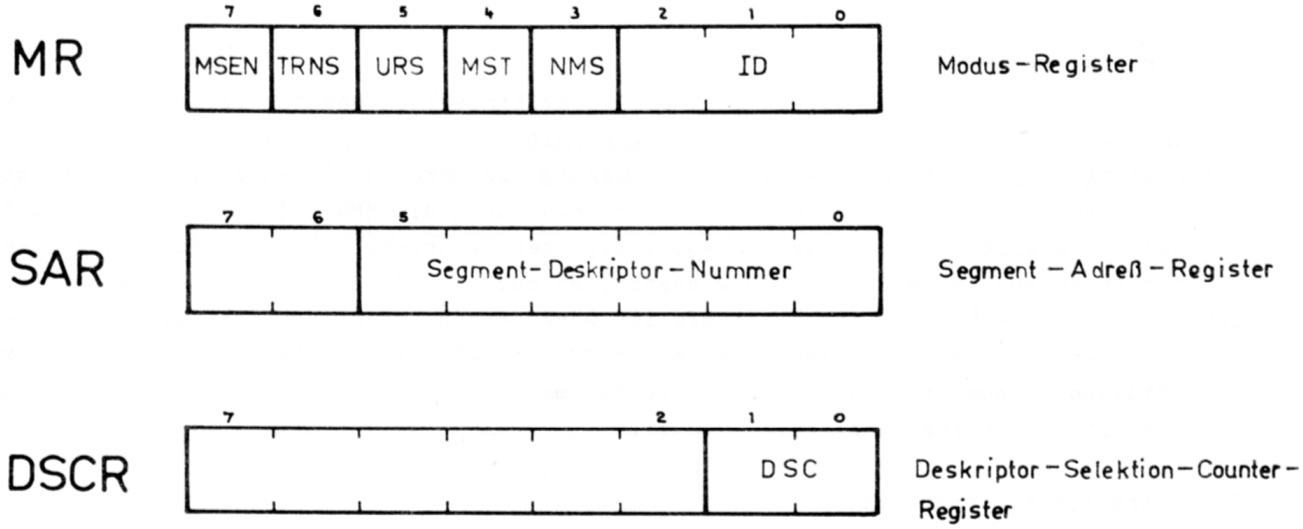


Bild 3.4: Steuerregister

3.2.2. Deskriptor-Selektion-Counter-Register (DSCR)

Dieses Register enthält einen 2-Bit-Zähler (Counter), der anzeigt, auf welches Byte des Deskriptors während eines Lese- oder Schreibkommandos zugegriffen wird. Mit Wert 0 zeigt dieser Zähler auf das höherwertige, mit Wert 1 auf das niederwertige Byte des Basisadreßfeldes. Der Wert 2 weist auf das Limitfeld und der Wert 3 auf das Attributfeld. Dieser Zähler wird in MMU-Kommandos verwendet, die auf mehrere Bytes eines Deskriptors zugreifen. (Der Datentransfer mit der MMU kann immer nur byteweise stattfinden.) Im allgemeinen wird der Zähler bei der Ausführung solcher Kommandos automatisch inkrementiert und nach Abarbeitung des Kommandos automatisch auf Null gesetzt. Wenn in Interruptserviceroutinen MMU-Kommandos ausgegeben werden, so ist vorher der Inhalt des Zählers zu retten und später bei deren Abschluß zu regenerieren. (Dies ist in Anwendungen zu beachten, bei denen ein Blocktransferbefehl zur MMU durch einen Interrupt unterbrochen werden kann und in der Interruptserviceroutine selbst eine Modifikation des DSCR erfolgt. Es muß dafür gesorgt werden, daß nach der Interruptserviceroutine der Blocktransferbefehl an der richtigen Position im Deskriptor fortgesetzt werden kann; vgl. Abschnitt 7.3.)

3.2.3. Modus-Register (MR)

Dieses Register enthält ein 3-Bit-Identifikationsfeld (ID), welches die Unterscheidung von acht aktiven MMUs in einer Mehrfach-MMU-Konfiguration erlaubt. Wenn ein Segmenttrap durch die CPU bestätigt wird (Status %4), verwendet die MMU dieses Feld, um eine A/D-Leitung auszuwählen, wobei jeder aktiven MMU eine andere Leitung zuzuweisen ist. Die den Segmenttrap anfordernde MMU sendet dann auf der ihr zugewiesenen A/D-Leitung H-Pegel aus. Die anderen aktiven MMUs senden auf ihren Leitungen L-Pegel. Leitungen, denen keine aktive MMU zugeordnet ist, verbleiben im Tri-State. Ein Befehl kann Violationen in mehreren MMUs verursachen, so daß sich die Trap-Behandlungssoftware mit mehreren MMUs befassen muß.
Des weiteren sind im Modus-Register fünf Flags vorhanden:

MSEN - Master Enable (Hauptfreigabe)
Dieses Flag aktiviert oder deaktiviert die Adreßübersetzung und die Segmentschutzfunktionen der MMU. Ist das Flag gesetzt, kann die MMU diese Funktionen ausführen. Ist das Flag nicht gesetzt, bleiben die Adreßausgänge der MMU im Tri-State, so daß bei rückgesetztem MSEN-Flag keine Speicheranforderung über die MMU erfolgen kann. Es werden dann auch alle anderen Flags im Modus-Register ignoriert. Ist in Konfigurationen mit nur einer MMU das MSEN-Flag rückgesetzt, so muß die CPU Zugriff auf einen speziellen Speicher haben, da es ihr nicht möglich wäre, einen Befehl vom Speicher zu holen, der von der MMU verwaltet wird. Neben der Möglichkeit dieses Flag durch Kommandos zu beeinflussen, kann es während der Ausführung des Hardware-Reset gesetzt oder rückgesetzt werden (vgl. Abschnitt 7.2.1).

TRNS - Translate (Übersetzen)
Dieses Flag zeigt an, ob die MMU die logischen Adressen des Programms in physische Speicheradressen übersetzen muß oder ob sie die logischen Adressen unverändert und ohne Prüfung der Zugriffsrechte zum Speicher hindurchläßt. Im Modus "nicht Übersetzen" (TRNS = 0) entspricht das höherwertige Byte der ausgegebenen Adresse der 7-Bit-Segmentnummer, wobei A_{23} L ist. Wenn MSEN und TRNS gesetzt sind, führt die MMU die Adreßübersetzung und die Überprüfung der Attribute aus (falls URS, MST und NMS dies erlauben). Wenn das TRNS rückgesetzt ist, und MSEN ist gesetzt, werden URS, MST und NMS ignoriert und die Adressen gehen unverändert durch die MMU hindurch.

URS - Upper Range Select (Auswahl im oberen Bereich)
Dieses Flag zeigt an, ob die MMU den unteren (0...63) oder oberen (64...127) Segmentnummernbereich verwaltet. Die MMU übersetzt die Adresse, wenn das höchstwertige Bit der Segmentnummer dem URS-Flag entspricht. Andernfalls bleiben die Adreßausgänge im Tri-State. Sie enthält die Deskriptoren der Segmente 0...63, wenn das Flag rückgesetzt ist. Dagegen enthält sie die Deskriptoren der Segmente 64...127, wenn das Flag gesetzt ist.

MST - Multiple Segment Table (mehrere Segmentübersetzungstabellen)
Dieses Flag zeigt an, ob mehrere Segmentübersetzungstabellen in der Hardwarekonfiguration benutzt werden. Wenn dieses Flag gesetzt ist, gibt es mehr als eine Tabelle und der N/$\overline{S}$-Eingang wird verwendet, um zu entscheiden, ob die MMU die entsprechende Tabelle enthält. Wenn es z.B. zwei Tabellen im Speicherverwaltungssystem gibt (eine für das Betriebssystem und eine für die Nutzer), ist das NMS-Flag in den MMUs gesetzt, die die Segment-Deskriptoren der Nutzer enthalten. In den System-MMUs ist es dagegen rückgesetzt. Bei allen MMUs in einem solchen System ist das MST-Flag gesetzt um anzuzeigen, daß mehrere Übersetzungstabellen im System existieren.

NMS - Normal Modus Select (Auswahl bei Normalmodus)
Dieses Flag ist nur bei gesetztem MST-Flag von Bedeutung. Es zeigt an, ob die MMU bei H- oder L-Zustand der N/$\overline{S}$-Leitung übersetzen soll. Wenn das MST-Flag gesetzt ist, muß der Zustand des N/$\overline{S}$-Eingangs dem des NMS-Flag entsprechen, damit die MMU die Segmentadressen übersetzt. Besteht keine Übereinstimmung, so bleiben die MMU-Ausgänge im Tri-State.

3.3. Statusregister

Sechs 8-Bit-Register enthalten Informationen, die zur Ermittlung der Ursache eines Segmenttrap dienen (Bild 3.5).

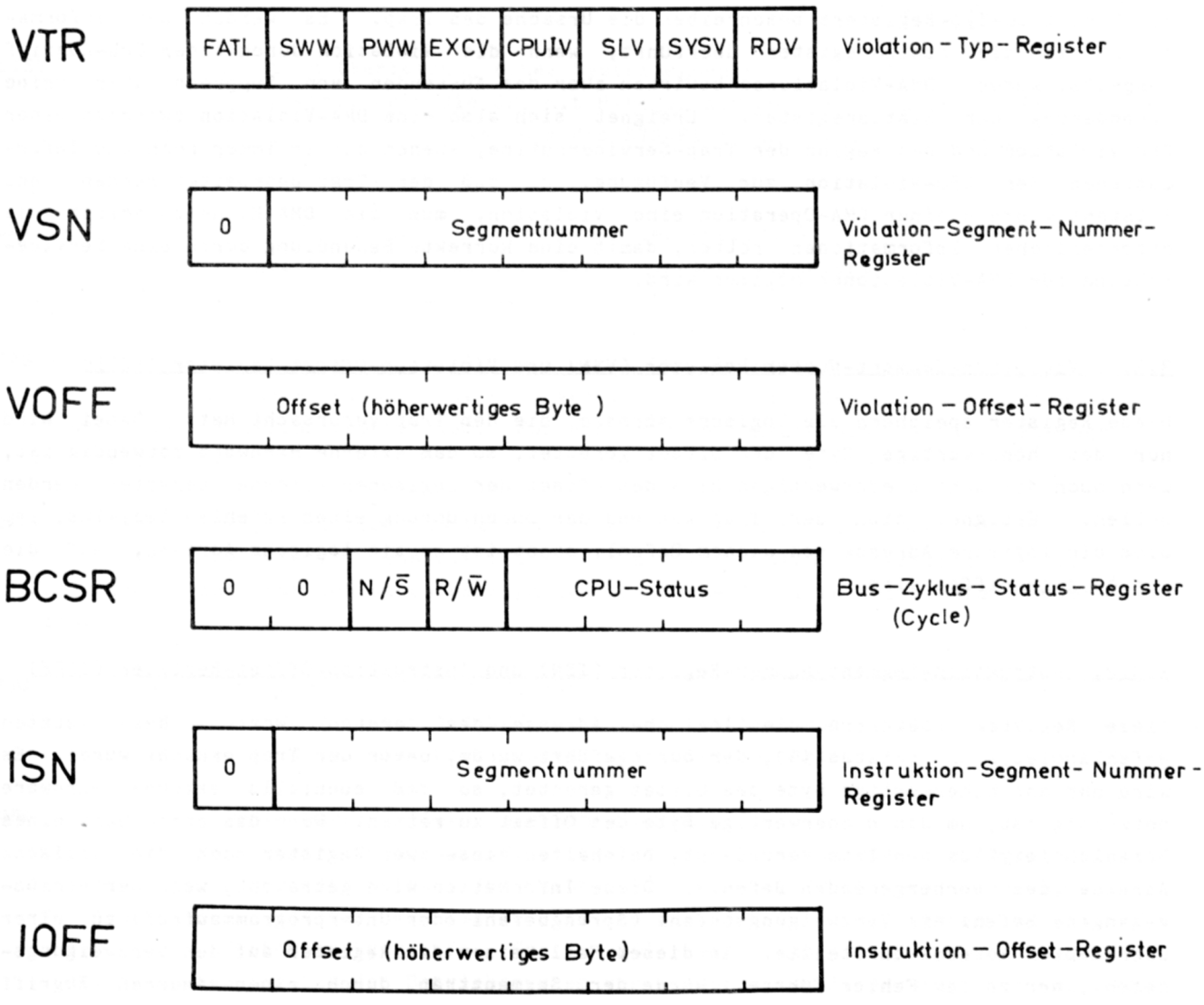


Bild 3.5: Statusregister

Das Violation-Segment-Nummer-Register und das Violation-Offset-Register enthalten die Segmentnummer und das höherwertige Byte des Offset der logischen Segmentadresse, die den Segmenttrap verursachte.

Das Bus-Zyklus-Status-Register speichert zum Zeitpunkt, in dem die Trap-Bedingung erkannt wurde, den aktuellen Busstatus.
Das Instruktion-Segment-Nummer-Register und das Instruktion-Offset-Register enthalten die Segmentnummer bzw. das höherwertige Byte des Offset der logischen Adresse des ersten Wortes vom letzten, vor dem Segmenttrap ausgeführten, Befehlsholezyklus.
Das Violation-Typ-Register beschreibt die Bedingungen, die die Segmenttrap-Anforderung generierten.
Die MMU erzeugt eine Segmenttrap-Anforderung in zwei Fällen:

- Sie erkennt eine Verletzung der Zugriffsbedingungen. (Es wird z.B. versucht, in ein nur lesbares Segment zu schreiben.)
- Sie erkennt eine Schreibwarnungsbedingung. (Es wird in die letzten 256 Bytes eines Segments geschrieben, dessen DIRW-Flag gesetzt ist.)

Wenn eine Violation oder eine Schreibwarnung erkannt wird, generiert die MMU eine Segmenttrap-Anforderung und setzt automatisch die entsprechenden Flags. Die acht Flags des Violation-Typ-Registers beschreiben die Ursache des Trap. Es werden nur Informationen in die Statusregister überführt, wenn die Violation durch einen CPU-Zugriff ausgelöst wurde. DMA-Violationen bewirken zwar das Aussenden von Suppress aber keine Veränderung der Statusregister. Ereignet sich also eine DMA-Violation zwischen einer CPU-Violation und dem Beginn der Trap-Serviceroutine, stehen dieser immer noch die Informationen der CPU-Violation zur Verfügung, so daß der Trap bearbeitet werden kann. Entsteht während einer DMA-Operation eine Violation, muß die DMA-Einheit selbst die erforderlichen Informationen retten, damit eine korrekte Behandlung durch eine Serviceroutine für DMA-Violationen möglich wird.

3.3.1. Violation-Segment-Nummer-Register (VSN) und Violation-Offset-Register (VOFF)

Diese Register speichern die logische Adresse, die den Trap verursacht hat. Dabei wird nur das höherwertige Byte des Offset gerettet, so daß externe Hardware notwendig ist, wenn auch die acht niederwertigen Bits des Offset der logischen Adresse gerettet werden sollen. Ereignet sich der Trap während der Durchführung eines Befehlsholezyklus, ist dies die logische Adresse von diesem Befehl, sonst ist es die logische Adresse, auf die gerade zugegriffen wird.

3.3.2. Instruktion-Segment-Nummer-Register (ISN) und Instruktion-Offset-Register (IOFF)

Diese Register speichern die logische Adresse des ersten Wortes des letzten Befehlsholezyklus (Status %D), der durchgeführt wurde, bevor der Trap erkannt wurde. Es wird nur das höherwertige Byte des Offset gerettet, so daß ebenfalls externe Hardware notwendig ist, um das niederwertige Byte des Offset zu retten. Wenn das erste Wort eines Befehlsholezyklus den Trap verursacht, beinhalten diese zwei Register noch die logische Adresse des vorhergehenden Befehls. Diese Information wird gebraucht, wenn der vorausgegangene Befehl ein Verzweigungsbefehl (Sprungbefehl oder Unterprogrammaufruf) zu einer unerlaubten Adresse darstellte. In diesem Fall weisen die Register auf den Verzweigungsbefehl, der zu dem Fehler führte. Würde der Segmenttrap durch einen anderen Zugriff verursacht, beinhalten die Register die logische Adresse des ersten Wortes des Befehls, der den verbotenen Zugriff auslöste.

3.3.3. Bus-Zyklus-Status-Register (BCSR)

Dieses Register enthält Informationen über den Busstatus im Augenblick des Erkennens der Trap-Bedingung. Dazu gehört der Typ der Operation ($ST_0 \ldots ST_3$). Außerdem zeigen zwei Flags, ob gelesen oder geschrieben (R/$\overline{W}$) und ob im Normalmodus oder im Systemmodus (N/$\overline{S}$) gearbeitet wurde.

3.3.4. Violation-Typ-Register (VTR)

Dieses Register enthält acht Flags, welche die Ursache des Segmenttrap beschreiben. Fünf Flags werden durch Verletzung von Zugriffsbedingungen beeinflußt, zwei sind Warnungsflags und eins zeigt an, ob es sich um eine Violation oder Schreibwarnung handelt.

RDV - Read-Only Violation (Verletzung eines nur lesbaren Segments)
Dieses Flag wird gesetzt, wenn die CPU mit L-Pegel auf der R/$\overline{W}$-Leitung versucht auf ein Segment zuzugreifen, für welches das RD-Flag gesetzt ist (nur lesbares Segment).

SYSV - System Violation (Verletzung eines Systemsegments)
Dieses Flag wird gesetzt, wenn die CPU mit H-Pegel auf der N/$\overline{S}$-Leitung versucht auf ein Segment zuzugreifen, für welches das SYS-Flag gesetzt ist (nur Zugriffe im Systemmodus erlaubt).

SLV - Segment Length Violation (Verletzung der Segmentlänge)
Dieses Flag wird gesetzt, wenn die CPU einen Offset generiert, der außerhalb des erlaubten Bereichs des Segments liegt.

CPUIV - CPU-Inhibit Violation (Verletzung eines CPUI-Segments)
Dieses Flag wird gesetzt, wenn die CPU versucht auf ein Segment zuzugreifen, für welches das CPUI-Flag gesetzt ist. Für ein solches Segment sind nämlich CPU-Zugriffe verboten.

EXCV - Execute-Only Violation (Verletzung eines nur ausführbaren Segments)
Dieses Flag wird gesetzt, wenn die CPU Zugriffe auf ein Segment versucht, dessen EXC-Flag gesetzt ist (nur ausführbares Segment) und diese Zugriffe keine Befehlsholezyklen sind.

PWW - Primary Write Warning (Primäre Schreibwarnung)
Dieses Flag wird gesetzt, wenn die CPU auf die letzten 256 Bytes eines Segments schreibt, für welches das DIRW-Flag gesetzt ist (Stacksegment).

SWW - Secondary Write Warning (Sekundäre Schreibwarnung)
Dieses Flag wird gesetzt, wenn die CPU im Systemmodus Daten in die Schreibwarnungszone des Systemstacks schreibt, (vgl. Abschnitt 5.4) und EXCV, CPUIV, SLV, SYSV, RDV, oder PWW gesetzt sind. Ist das SWW-Flag gesetzt, so werden bei nachfolgenden Schreibwarnungen durch Systemstackzugriffe (Status %9) keine Segmenttrap-Anforderungen mehr ausgelöst. Dies schützt das Betriebssystem vor wiederholten Interrupts für ein und dieselbe Warnung, solange es sich mit der Beseitigung der Ursache für diese Warnung beschäftigt.

FATL - Fatal Condition (Fatale Bedingung)

Dieses Flag wird gesetzt, wenn ein anderes Flag des VTR bereits gesetzt ist und eine weitere Violation erkannt wird oder eine Schreibwarnungsbedingung im Normalmodus hinzukommt oder wenn sich eine sekundäre Schreibwarnungsbedingung im Systemmodus ergibt, ohne daß dabei auf den Systemstack zugegriffen wird. (Im letzten Fall zeigen die Statusleitungen keinen Stackzugriff an.) Dieses Flag wird nicht bei Zugriffen auf den Systemstack gesetzt. Würde dabei eine Schreibwarnung auftreten, würde eine sekundäre Schreibwarnung signalisiert (Setzen des SWW-Flag). Das FATL-Flag zeigt an, daß sich Speicherzugriffsfehler während einer Trap-Serviceroutine ereignet haben. Ist das Flag gesetzt, so generieren nachfolgende Violationen keine Trap-Anforderungen mehr. Jedoch werden bei nachfolgenden CPU-Violationen Suppress-Signale generiert, bis das FATL-Flag zurückgesetzt wird.

4. Adreßübersetzung

4.0. Einführung

Bei der Bearbeitung einer logischen Adresse durch die MMU laufen zwei Prozesse parallel ab:

- Logische Adressen (Segmentnummer und Offset) werden in physische Adressen übersetzt.
- Die Statusinformationen der Speicherzugriffe werden mit den in den Segment-Deskriptor-Registern definierten Attributen verglichen. Eine Segmenttrap-Anforderung und/oder Suppress werden generiert, wenn eine Verletzung der Zugriffsbedingungen festgestellt wird.

Dieses Kapitel beschreibt den ersten Prozeß, die Adreßübersetzung. Der zweite Prozeß wird im Kapitel 5 diskutiert.

4.1. Bedingungen für die Adreßübersetzung

Adreßübersetzung bedeutet, daß die MMU mit Hilfe ihrer Segment-Deskriptoren aus den logischen Adressen an ihren SN- und AD-Eingängen physische Adressen an ihren Adreßausgängen $A_0 \ldots A_{23}$ herstellt. Neben dem Zustand, in dem die MMU Adreßübersetzungen ausführt, gibt es den Zustand, daß logische Adressen die MMU unverändert passieren oder den Zustand, daß sich die Ausgänge $A_0 \ldots A_{23}$ im Tri-State befinden.
Unter folgenden Bedingungen (geordnet nach ihrer Priorität) übersetzt die MMU, die an ihren SN-und AD-Eingängen anliegenden Adressen:

1. Die Adresse wird bei einem Speicherzugriff generiert (Status %8, %9, %A, %B, %C, %D). Anderenfalls sind die Ausgänge $A_0 \ldots A_{23}$ im Tri-State.
2. Das MSEN-Flag im Modus-Register ist gesetzt. Anderenfalls bleiben die Ausgänge $A_0 \ldots A_{23}$ im Tri-State und alle weiteren Flags im MR werden ignoriert.
3. Das TRNS-Flag im Modus-Register ist gesetzt. Anderenfalls wird die logische Adresse unverändert nach $A_0 \ldots A_{22}$ überführt und auf A_{23} wird L-Pegel ausgegeben. Dann werden auch bei gesetztem MSEN die restlichen Flags im MR ignoriert.
4. Das URS-Flag im MR hat denselben Zustand wie das Bit SN_6 der Segmentnummer. Anderenfalls bleiben die Ausgänge $A_0 \ldots A_{23}$ im Tri-State.
5. Das MST-Flag im MR ist entweder rückgesetzt oder das MST-Flag ist gesetzt und das NMS-Flag im MR hat denselben Zustand wie der $N/\overline{S}$-Eingang (d.h. wenn $N/\overline{S}$ = H ist muß NMS = 1 sein; wenn $N/\overline{S}$ = L ist muß NMS = 0 sein). Anderenfalls bleiben die Ausgänge $A_0 \ldots A_{23}$ im Tri-State.

4.2. Prozeß der Adreßübersetzung

Wenn die MMU eine Adresse übersetzt, verwendet sie die Bits $SN_0 \ldots SN_5$ der von der CPU U8001 generierten Segmentnummer als Zeiger, um eines der 64 Segment-Deskriptor-Register (SDR) auszuwählen. Durch das Bit SN_6 wird die MMU als Ganzes ausgewählt (vgl. Abschnitt 4.1.). Sollen alle 128 Segmente, die durch die CPU spezifiziert werden können, verwaltet werden, sind mindestens zwei MMUs erforderlich: Eine behandelt die Segmente 0...63 (URS = 0) und die andere die Segmente 64...128 (URS = 1).
Vom ausgewählten SDR wird der Inhalt des Basisadreßfeldes zum höherwertigen Byte des logischen Offset addiert und das Ergebnis vor das niederwertige Byte des logischen Offset

gestellt. Als Resultat entsteht eine 24-Bit-physische Adresse. Das niederwertige Byte des Offset wird durch die MMU nicht bearbeitet (Bild 4.1).

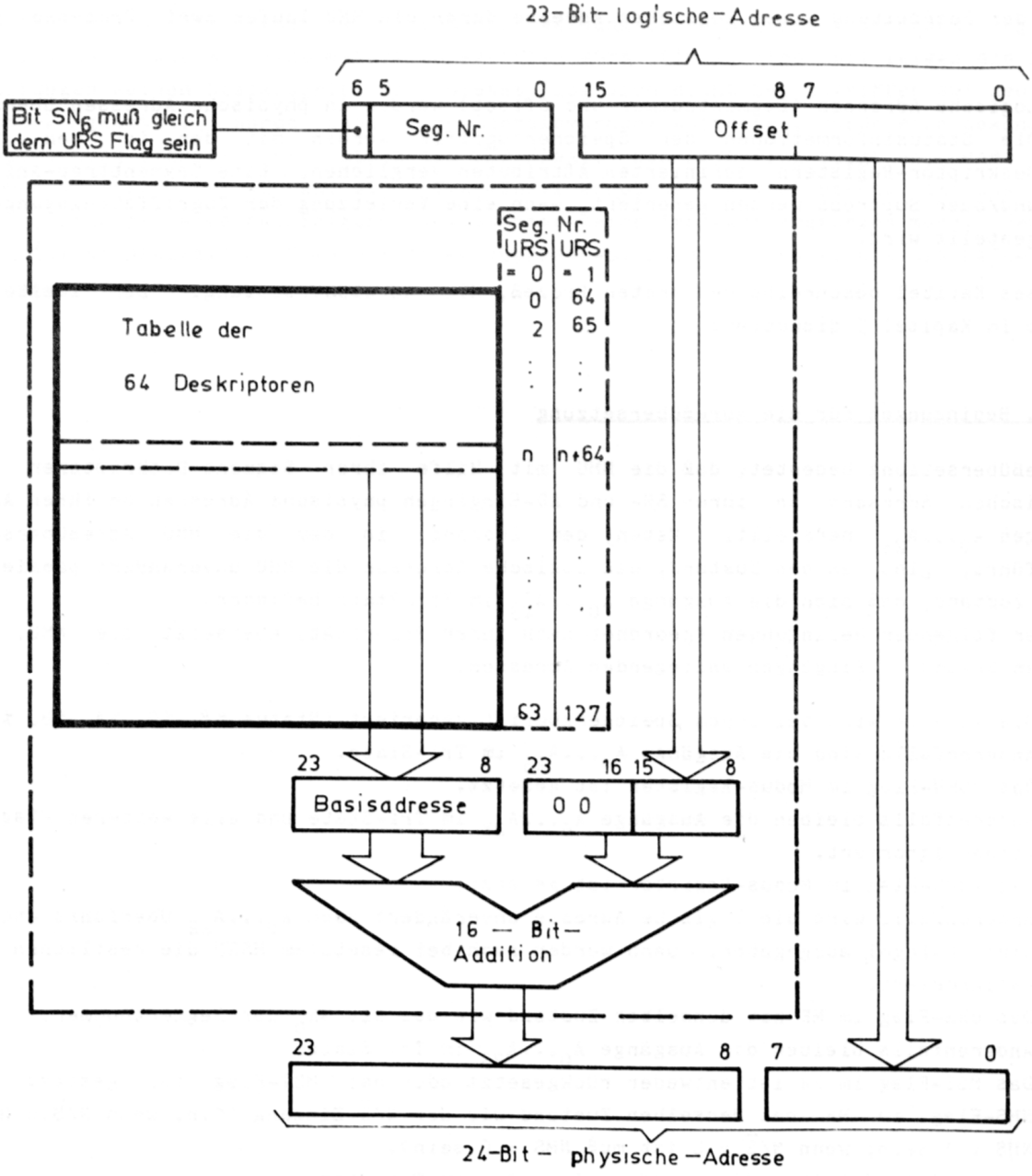


Bild 4.1: Adreßübersetzung

Der Vorgang der Adreßübersetzung in der MMU entspricht der Addition des logischen Offset zur physischen Anfangsadresse des Segments. Da der physische Speicher in 256-Byte-Blöcke aufgeteilt wird, sind die acht niederwertigen Bits der physischen Anfangsadresse Null und brauchen nicht im SDR gespeichert zu werden. Das Basisadreßfeld enthält in Wirklichkeit die 16 höherwertigen Bits der 24-Bit-physischen Anfangsadresse (Bild 4.2). Betrachtet sei als Beispiel, wie die logische Adresse <<5>> %1528 einem physischen Speicherplatz zugeordnet wird, wenn das Segment 5 auf dem Speicherplatz %231100 beginnt. (Gemäß der U8000-Assembler-Syntax werden Hexadezimalzahlen durch Voranstellen von "%" gekennzeichnet und Segmentnummern in zwei spitze Klammern eingeschlossen.)
Die Segmentnummer 5 wählt das Segment-Deskriptor-Register 5 in der MMU mit URS = 0 aus

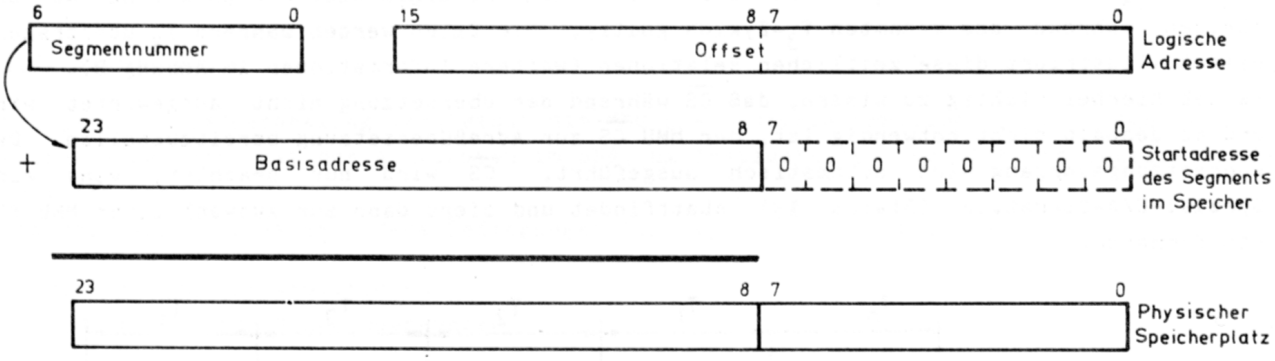


Bild 4.2: Erzeugung der physischen Adresse aus der logischen Adresse

(da SN_6 = 0). Das Basisadreßfeld des Registers enthält den Wert %2311, d.h. das Segment beginnt auf der Adresse %231100. Das höherwertige Byte des logischen Offset wird zum Inhalt des Basisadreßfeldes der MMU addiert (%15 wird zu %2311 addiert). Damit werden die 16 höherwertigen Bits des physischen Speicherplatzes bestimmt. Das niederwertige Byte des physischen Platzes ist gleich dem niederwertigen Byte des logischen Offset (Bild 4.3). Dieser Vorgang entspricht rechnerisch der Addition des logischen Offset (%1528) zur vollständigen Segmentanfangsadresse (%231100). Beide Algorithmen ergeben dieselbe physische Adresse %232628. Allerdings erfordert der erste Algorithmus, der in der MMU praktisch realisiert wird, nur eine 16-Bit-Addition. Dagegen würde der zweite eine 24-Bit-Addition, acht zusätzliche Eingänge (niederwertiges Byte des logischen Offset) und acht zusätzliche Ausgänge (niederwertiges Byte der physischen Adresse) erfordern.
Wenn die Summe aus dem Offset und der Basisadresse %FFFFFF überschreitet, gelten als physische Adresse die 24 niederwertigen Bits der Summe (Bit 25 wird ignoriert). Dies bedeutet, der Adresse %FFFFFF folgt unmittelbar die Adresse %000000. Diese Eigenschaft ("Umrunden des Speichers") kann nützlich sein, wenn der Nutzer Stacks in den untersten 64 KBytes des physischen Speichers anordnen will (vgl. Abschnitt 5.4.).

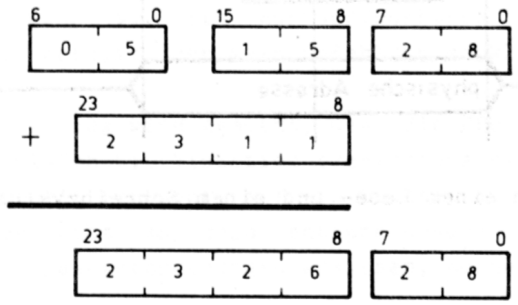


Bild 4.3: Darstellung der Adreßübersetzung

4.3. Lese-/Schreibzyklus der Adreßübersetzung

Bei einem Speicherlese-/Speicherschreibzyklus, wird die 7-Bit-Segmentnummer an $SN_0 \ldots SN_6$ eine Taktperiode früher als die Offsetadresse übernommen. $SN_0 \ldots SN_6$ müssen in der zweiten Hälfte des Taktzyklus gültig sein, damit die Segmentnummer für den Übersetzungsprozeß rechtzeitig verfügbar ist. Die höherwertigen acht Bits des Offset werden an

$AD_8 \ldots AD_{15}$ zu Beginn von T_1 übernommen. Die 16 höherwertigen Bits der physischen Adresse bleiben während des gesamten T_3-Zyklus gültig. Die Daten werden während T_3 übertragen. Bild 4 illustriert diese zeitlichen Relationen (weitere Informationen im Anhang B).
Es ist hierbei wichtig zu wissen, daß $\overline{CS}$ während der Übersetzung nicht ausgewertet wird und es deshalb nicht notwendig ist, der MMU $\overline{CS}$ zur Adreßübersetzung bereitzustellen. Der Übersetzungsvorgang wird automatisch ausgeführt. $\overline{CS}$ wird nur beachtet, wenn eine Spezial-E/A-Operation (Status %3) stattfindet und dient dann zur Auswahl einer MMU für ein Kommando.

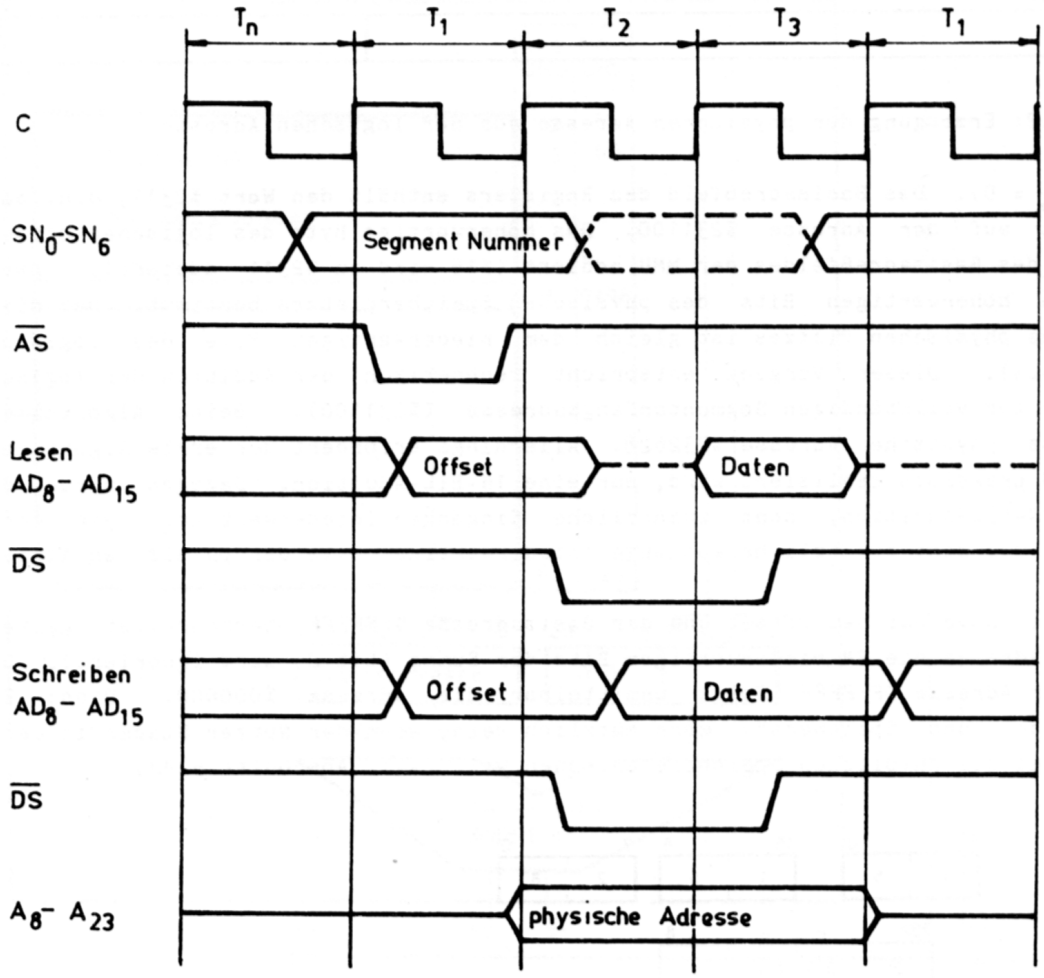


Bild 4.4: Zeitablauf der Adreßübersetzung in einem Lese- und einem Schreibzyklus

5. Violationen und Schreibwarnungen

5.0. Einführung

Bei jeder Adreßübersetzung vergleicht die MMU die Statusinformationen, die den Zugriff charakterisieren, mit den Attributen im SDR, um festzustellen, ob sich ein Zugriffsfehler ereignet hat.
Die MMU kann zwei verschiedene Bedingungen für Zugriffsfehler feststellen:

- Bedingungen für Violationen
- Bedingungen für Schreibwarnungen

5.1. Bedingungen für Violationen

Es gibt sechs verschiedene Bedingungen bei denen Violationen erfolgen. Fünf davon setzen Flags im VTR.

- Eine Read-Only-Violation (Verletzung eines nur lesbaren Segments) ereignet sich bei dem Versuch in ein nur lesbares Segment (Segment, dessen RD-Flag gesetzt ist) zu schreiben.
  Das RDV-Flag wird gesetzt, wenn die CPU hier einen Schreibzugriff generiert.
- Eine System-Violation (Verletzung eines Systemsegments) ereignet sich bei dem Versuch, auf ein Systemsegment (Segment, dessen SYS-Flag gesetzt ist) im Normalmodus zuzugreifen.
  Das SYSV-Flag wird gesetzt, wenn die CPU diesen Zugriff generiert.
- Eine Segment-Length-Violation (Verletzung der Segmentlänge) ereignet sich, wenn ein Offset den für das Segment zugewiesenen Bereich überschreitet.
  Das SLV-Flag wird gesetzt, wenn die CPU solch einen Zugriff generiert.
- Eine CPU-Inhibit-Violation (Verletzung eines CPUI-Segments) ereignet sich, wenn die CPU versucht auf ein Segment zuzugreifen, dessen CPUI-Flag gesetzt ist.
  Das CPUIV-Flag wird gesetzt, wenn die CPU diesen Zugriff generiert.
- Eine Execute-Only-Violation (Verletzung eines nur ausführbaren Segments) ereignet sich, wenn auf ein nur ausführbares Segment (Segment, dessen EXC-Flag gesetzt) ein Zugriff erfolgt, der kein Befehlsholezyklus ist, dessen Status also nicht %C oder %D ist.
  Das EXCV-Flag wird gesetzt, wenn die CPU diesen Zugriff generiert.
- Eine DMA-Inhibit-Violation (Verletzung eines DMAI-Segments) ereignet sich, wenn eine DMA-Einheit versucht, auf ein Segment zuzugreifen, dessen DMAI-Flag gesetzt ist.
  Hierfür gibt es kein entsprechendes Flag im VTR, weil DMA-Violationen keine Segmenttrap-Anforderungen auslösen, sondern nur Suppress-Signale aktivieren.

Ist irgendein Flag im VTR gesetzt, und es ereignet sich eine weitere Violation, wird das FATL-Flag gesetzt. Dies ist nur dann der Fall, wenn nach der ersten Violation in einem folgenden Befehl eine weitere Violation auftritt. Ereignen sich dagegen in einem Befehl mehrere Violationen, können mehrere Violationsflags gesetzt werden, aber das FATL-Flag wird nicht gesetzt.

5.2. Bedingungen für Schreibwarnungen

Eine Schreibwarnungsbedingung liegt vor, wenn in die unteren 256 Bytes eines Segments mit gesetztem DIRW-Flag geschrieben wird. Eine Schreibwarnung meldet dem Nutzer, daß ein Stacksegment bald überläuft (vgl. Abschnitt 5.4.). In Abhängigkeit von den Umständen, die zur Schreibwarnung führen, können drei verschiedene Flags im VTR gesetzt werden:

- Wenn kein anderes Flag im VTR gesetzt ist, wird beim Schreiben in die Schreibwarnungszone eines Segments mit gesetztem DIRW das primäre Schreibwarnungsflag gesetzt (PWW).
- Sind durch einen vorher ausgeführten Befehl primäre Flags im VTR (PWW, RDV, SYSV, CPUIV, EXCV oder SLV - vgl. Abschnitt 3.3.4.) gesetzt, und ein nachfolgender Systemstackzugriff (Status %9, N/$\overline{S}$ = L) verursacht eine Schreibwarnung, dann wird das sekundäre Schreibwarnungsflag (SWW) gesetzt.
- Unter der Voraussetzung, daß irgendein Flag im VTR durch einen vorhergehenden Befehl gesetzt wurde, erfolgt in zwei Fällen das Setzen des FATL-Flags:
  * Es ereignet sich eine Schreibwarnung durch eine im Systemmodus stattfindende Schreiboperation, die keine Systemstackoperation ist (nicht Status %9).
  * Es ereignet sich eine Schreibwarnung im Normalmodus.

  (Dieses Flag wird auch gesetzt, wenn sich in einem nachfolgenden Befehl eine Violation ereignet, und im VTR bereits ein anderes Flag gesetzt ist.) FATL und SWW werden nur gesetzt, wenn mehrere verbotene Zugriffe (Violationen oder Schreibwarnungen) in verschiedenen Befehlen erfolgen. Ereignen sich dagegen mehrere Violationen oder Schreibwarnungen bei der Ausführung ein und desselben Befehls, werden SWW und FATL nicht gesetzt. Nach dem Befehl, der eine oder mehrere Violationen verursacht hat, arbeitet die U8001 im Systemmodus (auf Grund der Segmenttrap-Anforderung) und nachfolgende Violationen oder Schreibwarnungen würden das FATL- oder SWW-Flag setzen.

5.3. Segmenttrap und Suppress

Die MMU kann auf Fehlerbedingungen mit zwei verschiedenen Signalen antworten: Segmenttrap-Anforderung ($\overline{SEGT}$) und Suppress ($\overline{SUP}$).

- Der Segmenttrap ist eine spezielle Art von Interrupt für die CPU. Dieser aktiviert die Trap-Serviceroutine. Dies ist ein Programm, daß den Trap bearbeitet und die Violation oder Schreibwarnung behandelt.
- Suppress ist ein Signal, das verwendet werden kann, um den Speicher bei fehlerhaften Schreiboperationen zu verriegeln. Es kann auch zur Alarmierung des Systems benutzt werden, um die Informationen zu retten, die einen Fehler verursacht haben.

Die MMU sendet verschiedene Kombinationen von Suppress- und Segmenttrap-Signalen aus, abhängig davon, ob der Fehler von einem CPU- oder DMA-Zugriff verursacht wurde und ob die Fehlerbedingung eine Violation oder eine Schreibwarnung war (Tabelle 5.1).

Fehlerbedingung	**CPU-Zugriff**	**DMA-Zugriff**
Violation	Trap und Suppress	nur Suppress
Schreibwarnung	nur Trap	kein Signal

Tabelle 5.1: Bedingungen für Trap- und Suppress-Signale

Es gelten folgende Grundsätze:

Das Suppress-Signal wird niemals bei Schreibwarnungen ausgesendet. (Es ist nicht notwendig das Schreiben auf den Speicher zu verhindern, nur weil ein Stack sich seinem Ende nähert.)
DMA-Fehler verursachen niemals einen Trap. (Segmenttrap-Anforderungen unterbrechen die CPU, aber nicht die DMA-Einheit.) Das bedeutet, DMA-Schreibwarnungen werden nicht signalisiert. (Angenommen solch ein Fall könnte sich überhaupt ereignen, denn DMA-Einheiten erzeugen normalerweise keine Stackzugriffe mit dem Status %9).

In einem System mit mehreren MMUs kann ein einziger Befehl Violationen in mehreren MMUs verursachen, so daß $\overline{SEGT}$ und/oder $\overline{SUP}$ von mehreren MMUs ausgesendet werden könnten. Die Segmenttrap-Anforderungen und die Suppress-Signale werden beide im Taktzyklus T_2 ausgesendet (Bild 5.1).

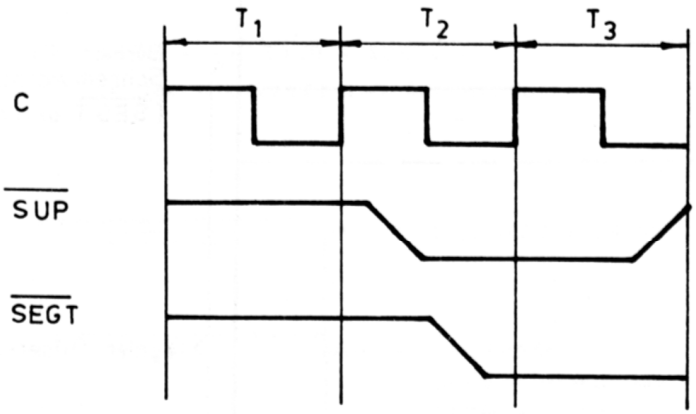


Bild 5.1: Segmenttrap- und Suppress-Signale

Das Segmenttrap-Signal bleibt L, bis die Segmenttrap-Anerkennung der CPU empfangen wird. Ereignet sich eine Violation, wird das Suppress-Signal für diesen Zugriff und für alle folgenden Zugriffe des Befehls ausgesendet. Bei dazwischengeschobenen DMA-Operationen erfolgt kein Suppress, vorausgesetzt sie verursachen nicht selbst eine Violation. Violationen bei der Durchführung von DMA-Operationen bewirken das Aussenden des Suppress-Signals nur während der Durchführung dieses Zugriffs. Dabei wird keine Trap-Anforderung generiert. CPU-Operationen werden durch DMA-Violationen nicht beeinflußt.

5.4. Stacksegmente und Schreibwarnungen

Normalerweise geht der erlaubte Bereich des Offset innerhalb eines Segments von

0 ... 256*N + 255 Bytes,

wobei $0 \leq N \leq 255$ und $N = n - 1$ ist. (N ist der Wert im Limitfeld des Segment-Deskriptor-Registers und n gibt an, wieviele 256-Byte-Blöcke dem Segment zugeordnet sind.)

Jedoch kann ein Segment durch Setzen des DIRW-Flag so spezifiziert werden, daß der erlaubte Offset im Bereich

256*N ... 65535 Bytes liegt,

wobei $0 \leq N \leq 255$ und $N = 256 - n$ ist.

Der letzte Segmenttyp wird bei Stacks angewendet. Die U8001-Stackbearbeitungsbefehle organisieren nämlich Stacks rückwärts (von höheren zu niederen Speicherplätzen). Wenn die Belegung eines Stack seine Grenze erreicht, kann an das eigentliche Ende des Segments

zusätzlicher Speicher angegliedert werden.
Zur Unterstützung der Stackverwaltung erkennt die MMU, wenn in die untersten 256 Bytes eines Stacksegments geschrieben wird und generiert dann eine Trap-Anforderung. Jedoch wird kein Suppress-Signal generiert, so daß die Eintragung in den Stack ausgeführt werden kann. Diese Schreibwarnung kann verwendet werden, um anzuzeigen, daß diesem Stacksegment mehr Speicher zuzuweisen ist.
Bild 5.2 zeigt die Lokalisation der Worte im Stack (logischer Bereich).

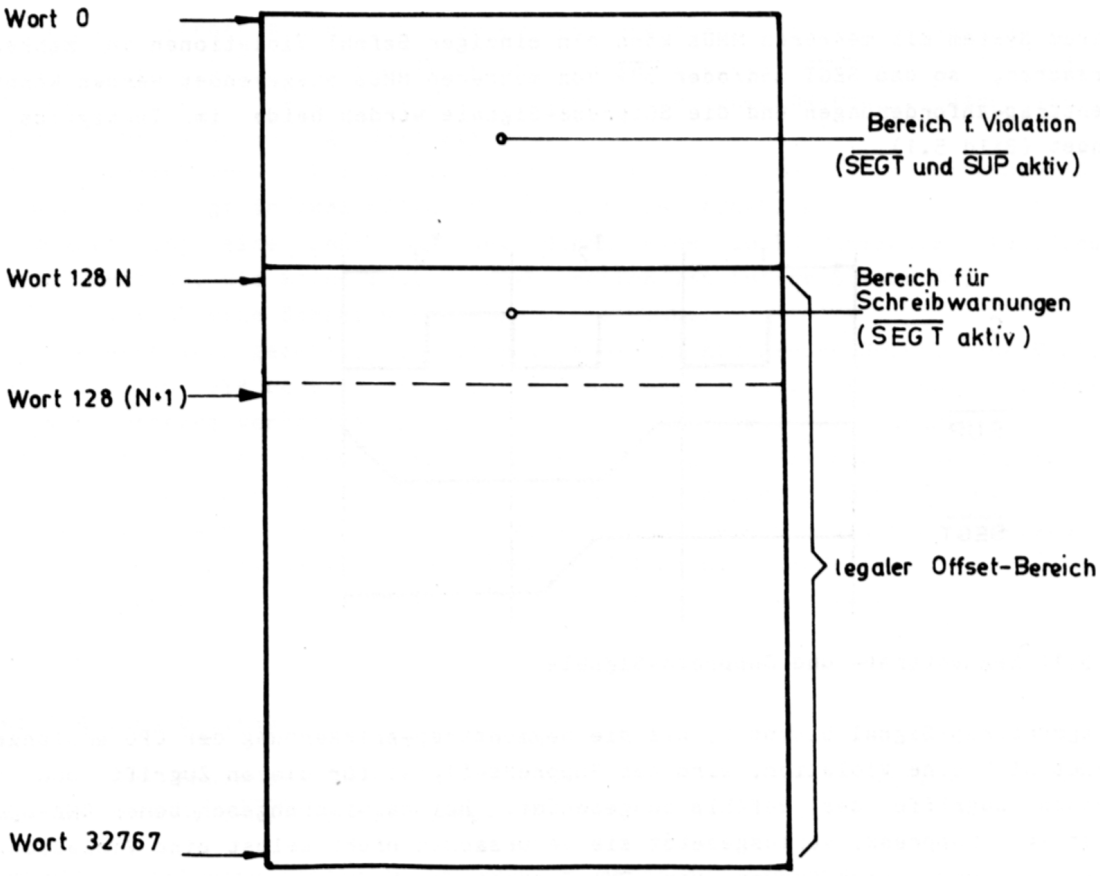


Bild 5.2: Stacksegmente und Schreibwarnungen

Bild 5.3 zeigt einen Stack, der n 256-Byte-Blöcke enthält.
Zur Definition eines Stack wird der Parameter TOP eingeführt. Er hat einen Wertebereich von

$$1 \leqq TOP \leqq \%10000 \quad \text{bzw.} \quad 1 \leqq TOP \leqq 65536$$

und kennzeichnet, im wievielten 256-Byte-Block des physischen Speichers ein Stacksegment beginnt (die höchste Adresse dieses Segments lokalisiert ist).

Die höchste Wortadresse im Stack ist TOP * %100 - 2.

Die niedrigste Wortadresse im Stack ist (TOP - n) * %100.

Es sind folgende Definitionen zu treffen, wenn ein Stacksegment gebildet werden soll (vgl. Bild 5.3):

Basisfeld = %FF00 + TOP

Limitfeld = %100 - n

Beispiel:

Ein Stacksegment soll definiert werden, dessen höchste Adresse im 488. 256-Byte-Block lokalisiert ist und eine Länge von 9 KByte hat.

Daraus ergeben sich:

TOP = 488 = %1E8

n = 4 * 9 = 36 = %24

höchste Wortadresse im Stack:

TOP * %100 - 2 = %1E8 * %100 - 2 = %1E7FE

niedrigste Wortadresse im Stack:

(TOP - n) * %100 = (%1E8 - %24) * %100 = %1C400

Basisfeld:

%FF00 + TOP = %FF00 + %1E8 = %100E8

⟶ In das Basisfeld ist %00E8 einzutragen.

Limitfeld:

N = %100 - n = %100 - %24 = %DC

⟶ In das Limitfeld ist %DC einzutragen.

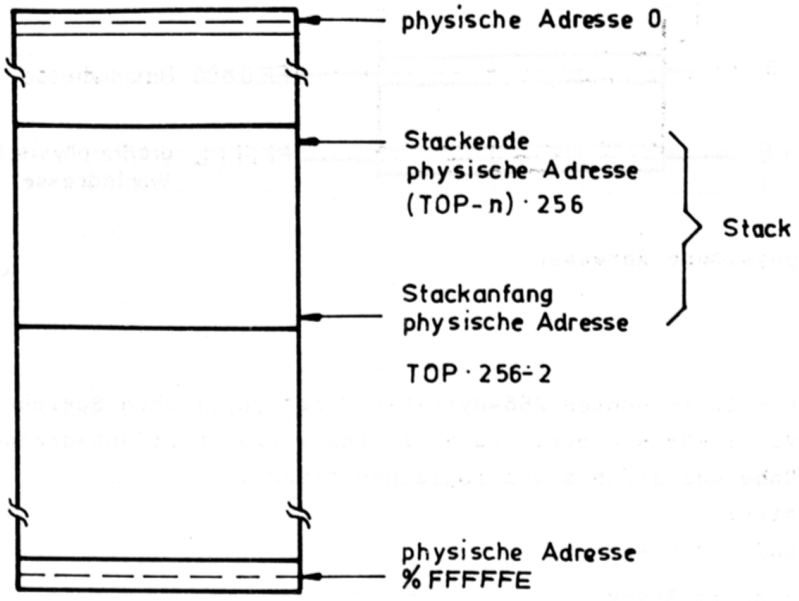


Bild 5.3: Physische Adressen des Stacksegments

Ein spezieller Fall erfordert detailiertere Ausführungen:
Ein Stack soll in den untersten 64 KBytes des physischen Speichers platziert werden. Der Fall ist ungewöhnlich, weil es anscheinend erfordert, die Basisadresse des Stacksegments kleiner als die Adresse 0 zu wählen. In Wirklichkeit wird die Eigenschaft des "Umrundens des Speichers" angewandt (vgl. Abschnitt 4.2.) und die Basisadresse liegt am Ende des physischen Adreßraumes. Bild 5.4 zeigt die logischen und physischen Adressen, die im folgenden Beispiel verwendet werden.

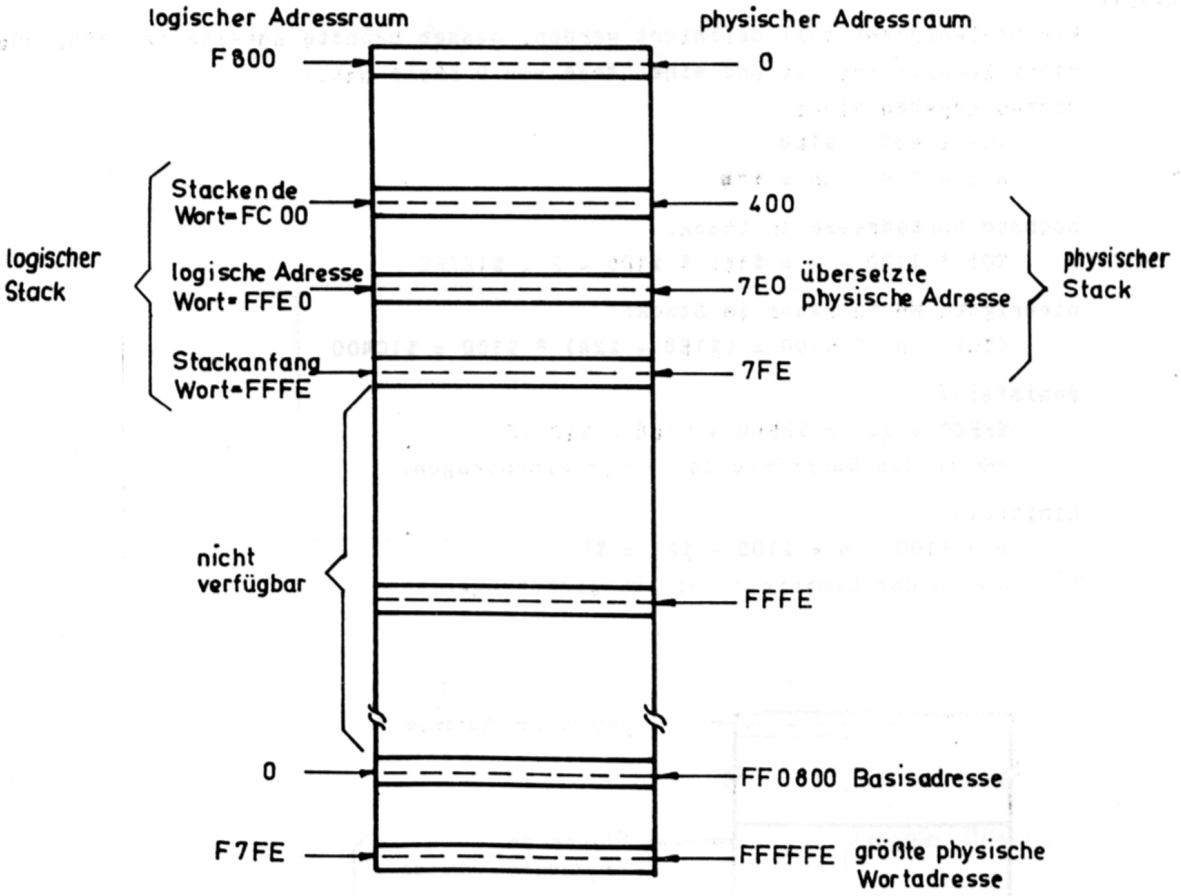


Bild 5.4: Logische und physische Adressen

Beispiel:

Ein Stacksegment soll im achten 256-Byte-Block des physischen Speichers beginnen und eine Länge von 1 KByte haben. Es soll eine logische Offsetadresse %FFE0 vorliegen (in der Nähe des Anfangs des logischen Stack).

Daraus ergeben sich:

TOP = 8 und n = 4

höchste Wortadresse im Stack:

TOP * %100 - 2 = 8 * %100 - 2 = %7FE

niedrigste Wortadresse im Stack:

(TOP - n) * %100 = (8 - %4) * %100 = %400

Basisfeld:

%FF00 + TOP = %FF00 + %8 = %FF08

⟶ In das Basisfeld ist %FF08 einzutragen.

Limitfeld:

%100 - n = %100 - %4 = %FC

⟶ In das Limitfeld ist %FC einzutragen.

physische Adresse:

Basisadresse + Offset = %FF0800 + %FFE0 = %10007E0

⟶ Es liegt die physische Adresse %7E0 vor.

Das DIRW-Flag im Attributfeld des Segment-Deskriptors zeigt an, daß das Segment von der MMU als ein Stacksegment zu behandeln ist. Wenn das DIRW-Flag gesetzt ist, ist der Bereich des erlaubten Offset (256*N...65535) Bytes. Der Versuch, in den Bereich 256*N...256*N + 255 zu schreiben, generiert eine Segmenttrap-Anforderung, aber kein Suppress und zeigt dadurch eine Schreibwarnung an.
Systemstacks (d.h. Stacks, die RR14' als Stackpointer verwenden) erhalten eine spezielle Behandlung. Wenn eines der primären Flags im VTR (vgl. Anhang A) gesetzt ist und sich eine Schreibwarnung infolge des Rettens in den Systemstack ereignet, dann wird das SWW-Flag gesetzt und ein Segmenttrap (aber kein Suppress) generiert. Durch nachfolgendes Retten in den Systemstack, welches wieder eine Schreibwarnungsbedingung verursacht, werden keine Segmenttraps mehr erzeugt. Damit wird in der Trap-Serviceroutine der Fehler verhindert, daß durch die Trap-Bearbeitung selbst immer weitere Trap-Anforderungen entstehen.

5.5. Segmenttrap-Anerkennungszyklus

Segmenttrap-Anforderungen an die CPU U8001 werden wie Interrupts behandelt. (Segmenttraps haben eine höhere Priorität als andere Interrupts, ausgenommen nichtmaskierbare Interrupts, aber eine niedrigere Priorität als interne Traps.) Bild 5.5 zeigt den Segmenttrap-Anerkennungszyklus.

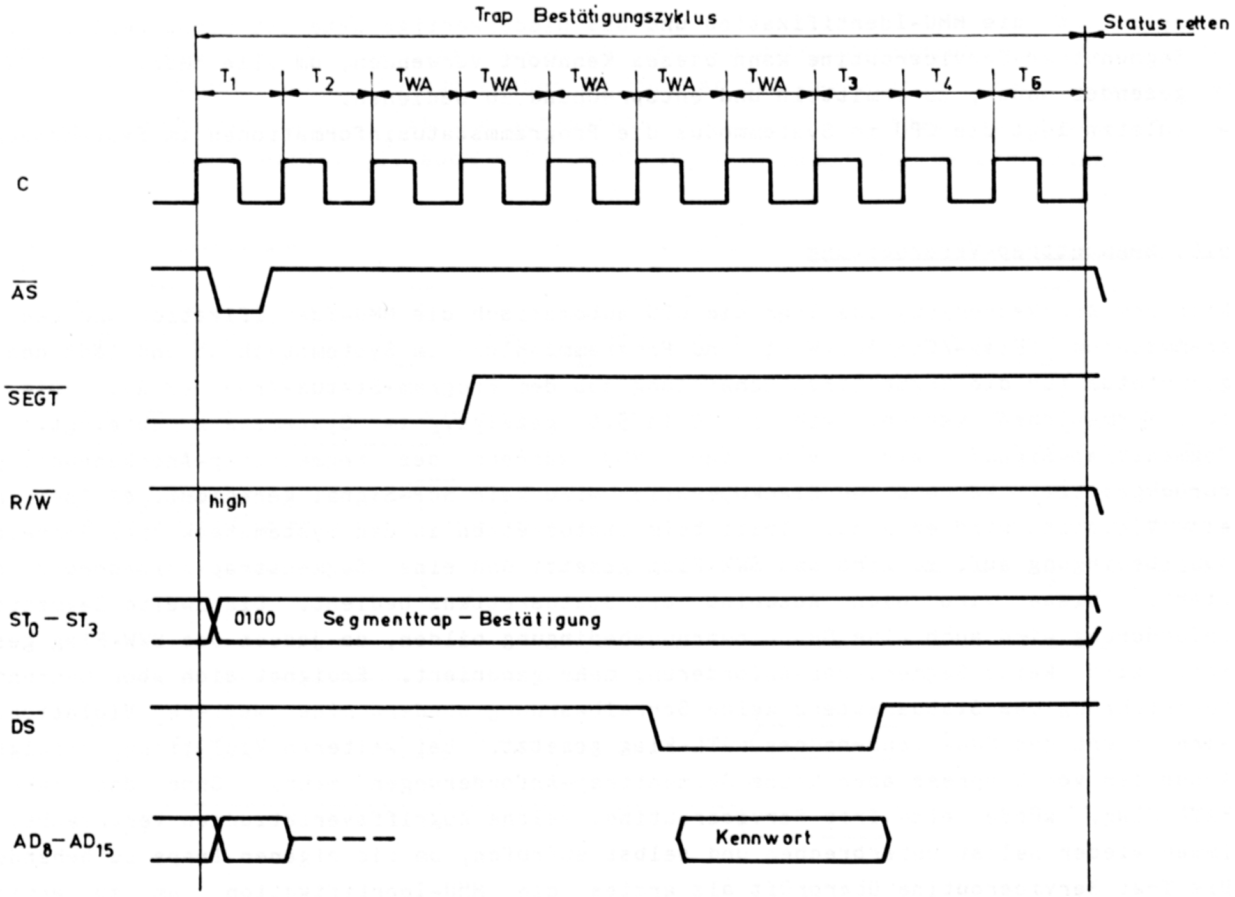


Bild 5.5: Segmenttrap-Anerkennungszyklus

Wenn ein Segmenttrap durch die CPU anerkannt wird, laufen folgende Phasen ab:

- Der folgende Befehlsholezyklus wird begonnen, aber dann abgebrochen. Der Programmzähler wird nicht aktualisiert, aber der Systemstackpointer wird dekrementiert (Vorbereitung zum Retten des Programmstatus). Die MMU ignoriert diesen Zyklus. (Eine Violation hierbei generiert Suppress, aber keinen Segmenttrap. Vgl. Anhang A.3: "Unechter" Befehlsholezyklus)
- Die CPU generiert den Segmenttrap-Anerkennungszyklus (vgl. "CPU U8001/U8002 - Technische Beschreibung"). Der Kode für den Segmenttrap-Anerkennungszyklus (Status %4) wird zu Beginn des Zyklus auf den Statusleitungen ausgegeben. Der Zyklus enthält fünf automatische Wait-Zyklen. Empfängt die MMU die Anerkennung, so bringt sie $\overline{\text{SEGT}}$ auf H.
- Bei Durchführung des Anerkennungszyklus, verwenden alle MMUs mit gesetztem MSEN die Adreß-/Datenleitungen, um ihren Zustand anzuzeigen. Eine MMU, die eine Segmenttrap-Anforderung generiert hat, gibt auf der AD-Leitung, die durch die Nummer in ihrem ID-Feld definiert wird, H-Pegel aus. Eine MMU die keine Segmenttrap-Anforderung generiert hat, gibt auf der ihr zugeordneten AD-Leitung L aus. AD-Leitungen, denen keine freigegebene MMU zugeordnet ist, verbleiben im Tri-State. Eine MMU verwendet die AD-Leitung AD_{8+i}, wenn ihr ID-Feld i enthält.
- Nach dem letzten Wait-Zyklus des Anerkennungszyklus, liest die CPU das Kennwort vom AD-Bus ($AD_0 \ldots AD_{15}$) und speichert es vorübergehend, damit es nachfolgend in den Stack gerettet werden kann. Dieses Wort kennzeichnet die Quelle des Trap. Das höherwertige Byte ist die MMU-Identifikation und das niederwertige Byte ist nicht definiert. Die Segmenttrap-Serviceroutine kann dieses Kennwort verwenden, um alle MMUs, die $\overline{\text{SEGT}}$ ausgesendet haben, zu ermitteln und entsprechend zu bedienen.
- Zuletzt legt die CPU im Systemmodus die Programmstatusinformationen im Systemstack ab.

5.6. Segmenttrap-Verarbeitung

Nach dem Anerkennungszyklus legt die CPU automatisch die MMU-Identifikation und den Programmstatus (Flag-/Control-Wort und Programmzähler) im Systemstack ab und lädt den Progammstatus für die Segmenttrap-Behandlung aus dem Program-Status-Area (PSA). Die Statusinformationen werden, wie in Bild 5.6 gezeigt, im Systemstack abgelegt. Das Segmenttrap-Signal wird von der MMU während des Segmenttrap-Anerkennungszyklus zurückgesetzt. Während der Stackoperation wird kein $\overline{\text{SUP}}$-Signal generiert, es sei denn, eine Violation wird erkannt. Tritt beim Statusretten in den Systemstack eine Schreibwarnungsbedingung auf, so wird das SWW-Flag gesetzt und eine Segmenttrap-Anforderung generiert. Diese wird nach Abschluß des Statusrettens bedient. Die zweite Segmenttrap-Anforderung wird auch eine Schreibwarnunsbedingung bilden, da jedoch das SWW-Flag gesetzt ist, wird keine Segmenttrap-Anforderung mehr generiert. Ereignet sich aber während der Durchführung des Statusrettens keine Schreibwarnung sondern eine weitere Violation, so wird nicht das SWW- sondern das FATL-Flag gesetzt. Bei weiteren Violationen erfolgt das Aussenden von Suppress aber keine Segmenttrap-Anforderungen mehr. Ohne das SWW- und FATL-Flag, würde eine Trap-Serviceroutine, welche Zugriffsverletzungen verursacht, sich immer wieder selbst unterbrechen und selbst aufrufen, um die eigenen Traps zu behandeln. Die Trap-Serviceroutine überprüft als erstes die MMU-Identifikation, um zu ermitteln welche MMUs Traps ausgesendet haben. Dann schickt sie Kommandos zu den MMUs, um das VTR zu lesen (vgl. Kapitel 6). Als nächstes wird das FATL-Flag im VTR überprüft, um festzustellen, ob sich ein fataler Systemfehler ereignet hat. Ist dies nicht der Fall, wird das SWW-Flag überprüft, um festzustellen, ob mehr Speicher für den Systemstack erforderlich ist. Anschließend wird der eigentliche Trap behandelt und das VTR zurückgesetzt (vgl. Abschnitt 8.1.2).

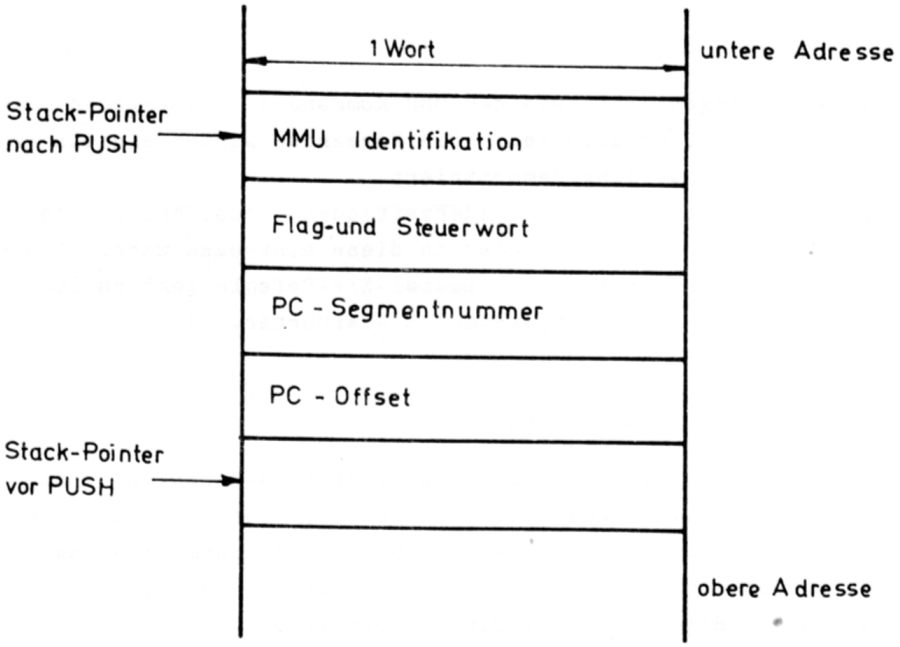


Bild 5.6: Systemstack bei einem Segmenttrap

Bild 5.7 zeigt den vollständigen Zeitablauf vom letzten Maschinenzyklus des Befehls, der den Trap verursacht hat, bis zum ersten Zyklus, ab dem der Status gerettet wird. Der abgebrochene Befehlsholezyklus ist der sogenannte "unechte" Befehlsholezyklus, der im Anhang A.3. erläutert wird (vgl. auch "CPU U8001/U8002 - Technische Beschreibung").

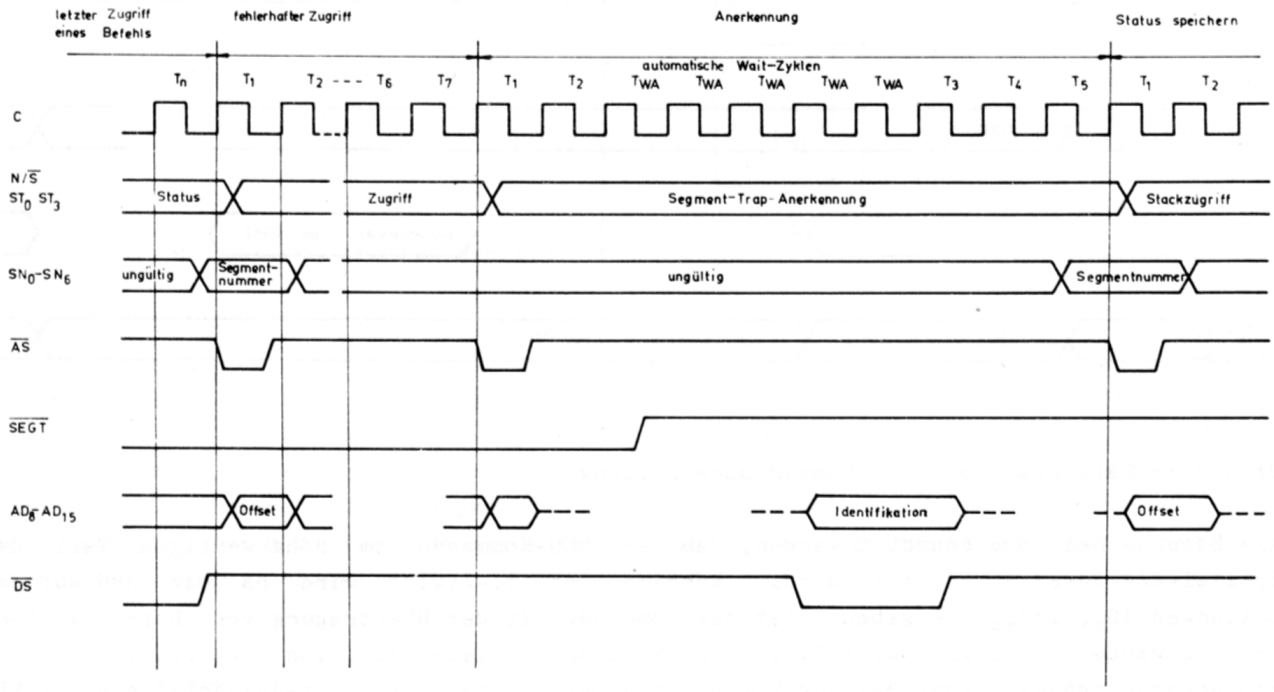


Bild 5.7: Zeitlicher Ablauf eines Segmenttrap

6. Kommandobearbeitung

6.0. Einführung

Im folgenden Kapitel wird dargestellt, wie der MMU Kommandos zu übergeben sind und wie die Phasen der Kommandobearbeitung ablaufen. Die Kommandos werden einzeln erläutert, und an einem Beispiel wird deren Anwendung demonstriert.
Grundsätzlich gilt, daß das Betriebssystem jederzeit durch Ausführung von Spezial-E/A-Befehlen Informationen aus der MMU lesen oder in diese eintragen kann. Das MMU-Kommando wird dabei in der E/A-Adresse kodiert. Die Spezial-E/A-Befehle gehören zu den privilegierten Befehlen, d.h. sie sind nur im Systemmodus ausführbar.

6.1. Zeitablauf bei der Kommandobearbeitung

Die MMU akzeptiert ein Kommando, wenn sie bei aktiviertem $\overline{CS}$-Anschluß einen Spezial-E/A-Zugriff auf den Statusleitungen ermittelt (Status %3). Der Zeitablauf bei der Kommandobearbeitung entspricht dem der Adreßübersetzung mit der Ausnahme, daß zwischen T_2 und T_3 ein Wait-Zyklus eingeschoben wird. Diesen Wait-Zyklus organisiert die CPU U8000 bei E/A-Zugriffen automatisch. Bild 6.1 zeigt diese zeitlichen Abläufe. Nähere Informationen sind im Anhang B "Zeitabläufe" zu finden.

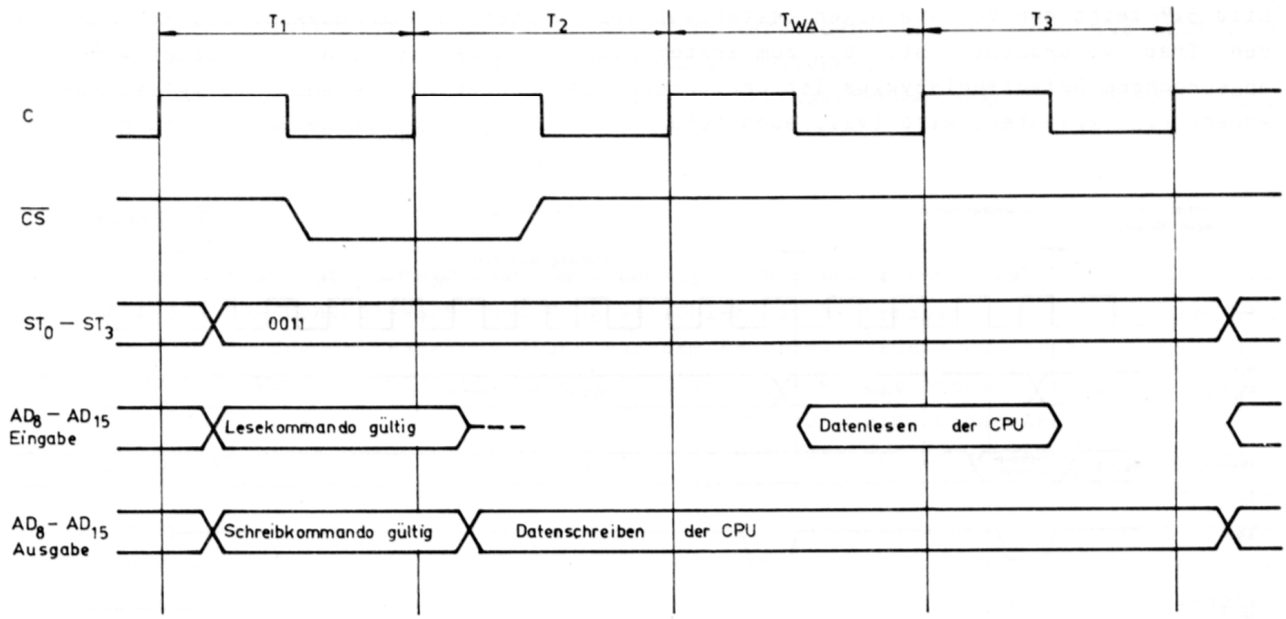


Bild 6.1: Zeitablauf bei der Kommandobearbeitung

Als Besonderheit muß beachtet werden, daß das MMU-Kommando im höherwertigen Teil der Spezial-E/A-Adresse kodiert wird. Während des T_1-Zyklus wird es der MMU auf den Leitungen $AD_8 \ldots AD_{15}$ übergeben. Ist das Kommando mit der Übertragung von Daten verbunden, so werden diese im Zyklus T_3 in die MMU eingeschrieben bzw. von dort gelesen.
Ein entscheidendes Signal bei der Kommandobearbeitung ist das Chip-Auswahl-Signal (Chip Select, $\overline{CS}$). Es wird nur im Kommandomodus, nicht im Übersetzungsmodus, aktiviert. Mit diesem Signal wird die entsprechende MMU in einem System mit mehreren MMUs ausgewählt. Es muß aber auch in einem System mit nur einer MMU aktiviert werden, wenn diese ein Kommando verarbeiten soll. Das $\overline{CS}$-Signal wird aus dem niederwertigen Byte der Spezial-E/A-Adresse ($AD_0 \ldots AD_7$) dekodiert.

Für U8000-Systeme wurde vereinbart, daß Bytetransfers mit Einheiten im Spezial-E/A-Raum auf dem höherwertigen Teil des Adreß-/Datenbusses zu erfolgen haben und daß diese mit geraden Adressen auszuwählen sind. AD_0 muß deshalb bei der Bildung des $\overline{CS}$-Signals L sein. Das Lesen eines Byte mit ungerader Adresse im Spezial-E/A-Raum wäre ein nicht definierter Zugriff. Bild 6.2 zeigt den prinzipiellen Anschluß mehrerer MMUs an den Adreß-/Datenbus der CPU.

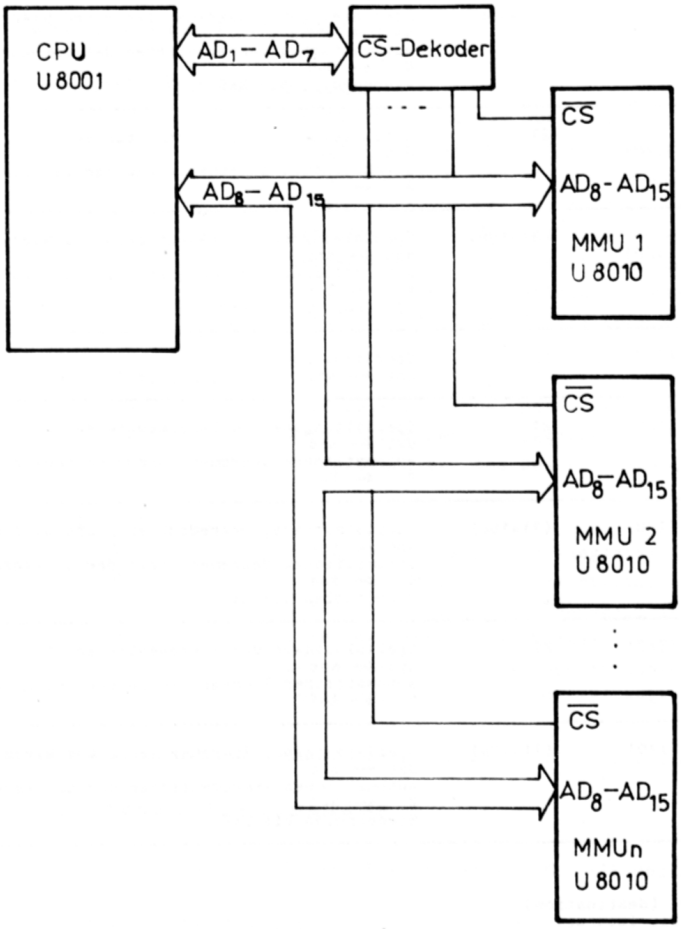


Bild 6.2: Prinzipieller Anschluß mehrerer MMUs an den AD-Bus

Wie die Auswahl einer MMU erfolgt, hängt vom jeweiligen System ab. Die einfachste Methode wäre, AD_1 auf den $\overline{CS}$-Eingang der MMU1 zu legen, AD_2 auf den der MMU2 usw. bis AD_7 auf den Eingang der MMU7. Bei einem Spezial-E/A-Zugriff mit L-Pegel auf einer dieser Leitungen würden dann die entsprechenden MMUs selektiert. Dies setzt allerdings voraus, daß nicht mehr als sieben MMUs zu steuern sind. Bei den nachfolgenden Beispielen wird angenommen, daß diese Methode angewendet wird. Für Systeme mit mehr als sieben MMUs muß deren $\overline{CS}$-Signal durch zusätzliche Hardware aus dem niederwertigen Teil des Adreß-/Datenbusses dekodiert werden.

6.2. MMU-Kommandos

In der Tabelle 6.1 sind alle Spezial-E/A-Befehle mit deren Wirkung, den möglichen Adressierungsarten und der Zahl der benötigten Taktzyklen zusammengestellt. Das Bild 6.3 zeigt Syntax und Bitschema der Spezial-E/A-Befehle. Das MMU-Kommandowort ist bei den Eingabebefehlen der Quelloperand, bei den Ausgabebefehlen der Zieloperand. Es enthält

Mnemonik	Operanden	Adressierungsarten	Taktzyklen	Wirkung
SIN SINB	dst,src	dst: R src: DA (I/O)	12	Spezialeingabe dst ← src
SIND SINDB	dst,src,r	dst: IR src: IR (I/O) r: R	21	Spezialeingabe und Dekrementieren dst ← src automatisches Dekrementieren der dst-Adresse R ← R-1
SINDR SINDRB	dst,src,r	dst: IR src: IR (I/O) r: R	(11+10n)	Spezialeingabe, Dekrementieren und Wiederholen dst ← src automatisches Dekrementieren der dst-Adresse R ← R-1 Wiederholen bis R=0
SINI SINIB	dst,src,r	dst: IR src: IR (I/O) r: R	21	Spezialeingabe und Inkrementieren dst ← src automatisches Inkrementieren der dst-Adresse R ← R-1
SINIR SINIRB	dst,src,r	dst: IR src: IR (I/O) r: R	(11+10n)	Spezialeingabe, Inkrementieren und Wiederholen dst ← src automatisches Inkrementieren der dst-Adresse R ← R-1 Wiederholen bis R=0
SOUT SOUTB	dst,src	dst: DA (I/O) src: R	12	Spezialausgabe dst ← src
SOUTD SOUTDB	dst,src,r	dst: IR (I/O) src: IR r: R	21	Spezialausgabe und Dekrementieren dst ← src automatisches Dekrementieren der src-Adresse R ← R-1
SOTDR SOTDRB	dst,src,r	dst: IR (I/O) src: IR r: R	(11+10n)	Spezialausgabe, Dekrementieren und Wiederholen dst ← src automatisches Dekrementieren der src-Adresse R ← R-1 Wiederholen bis R=0
SOUTI SOUTIB	dst,src,r	dst: IR (I/O) src: IR r: R	21	Spezialausgabe und Inkrementieren dst ← src automatisches Inkrementieren der src-Adresse R ← R-1
SOTIR SOTIRB	dst,src,r	dst: IR (I/O) src: IR r: R	(11+10n)	Spezialausgabe, Inkrementieren und Wiederholen dst ← src automatisches Inkrementieren der src-Adresse R ← R-1 Wiederholen bis R=0

Erklärungen:

dst	Ziel (destination)
src	Quelle (source)
r	Register
R	Allzweckregister
IR	indirekte Adressierung
IR (I/O)	indirekte Adressierung einer E/A-Einheit (MMU-Kommandowort)
DA (I/O)	direkte Adressierung einer E/A-Einheit (MMU-Kommandowort)
n	Anzahl der transportierten Datenelemente

Tabelle 6.1: Spezial-E/A-Befehle

sowohl den Befehlskode zur Festlegung der auszuführenden Operation als auch den Auswahlkode zur Selektierung der MMU. Wenn die oben genannte einfache Dekodiermethode angewendet wird, können durch entsprechende Auswahlkodes auch mehrere MMUs gleichzeitig angesprochen werden. Im Bild 6.4 ist das Bitschema der MMU-Kommandoworte dargestellt.

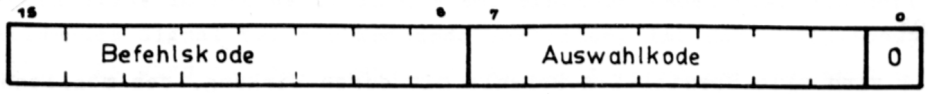


Bild 6.4: Bitschema der MMU-Kommandoworte

Die Tabelle 6.2 zeigt alle MMU-Kommandos, deren Befehlskode und Wirkung.

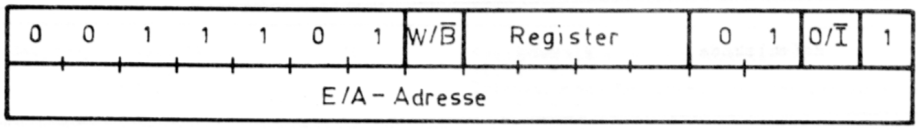
Befehlskode	Wirkung
Lese-/Schreibkommandos	
Segment-Deskriptor-Register	
08	Lesen/Schreiben des Basisfeldes im Deskriptor
09	Lesen/Schreiben des Limitfeldes im Deskriptor
0A	Lesen/Schreiben des Attributfeldes im Deskriptor
0B	Lesen/Schreiben des Deskriptors (alle Felder)
0C	Lesen/Schreiben des Basisfeldes und Erhöhen des SAR
0D	Lesen/Schreiben des Limitfeldes und Erhöhen des SAR
0E	Lesen/Schreiben des Attributfeldes und Erhöhen des SAR
0F	Lesen/Schreiben des Deskriptors (alle Felder) und Erhöhen des SAR
Steuerregister	
00	Lesen/Schreiben des Modus-Registers (MR)
01	Lesen/Schreiben des Segment-Adreß-Registers (SAR)
20	Lesen/Schreiben des Deskriptor-Selektor-Counter-Registers (DSCR)
Statusregister	
02	Lesen des Violation-Typ-Registers (VTR)
03	Lesen des Violation-Segmentnummer-Registers (VSN)
04	Lesen des Violation-Offset-Registers, höherwertiges Byte (VOFF)
05	Lesen des Bus-Zyklus-Status-Registers (BCSR)
06	Lesen des Instruktion-Segmentnummer-Registers (ISN)
07	Lesen des Instruktion-Offset-Registers, höherwertiges Byte (IOFF)
Setz-/Rücksetzkommandos	
10	Reset (Rücksetzen von MR, VTR und DSCR)
Violation-Status-Register	
11	Rücksetzen Violation-Typ-Register (VTR)
13	Rücksetzen SWW-Flag im VTR
14	Rücksetzen FATL-Flag im VTR
Segment-Deskriptor-Register	
15	Setzen aller CPU-Inhibit-Flags
16	Setzen aller DMA-Inhibit-Flags
Reservierte Kommandos	
12	nicht benutzen
17-1F	nicht benutzen
21-FF	nicht benutzen

Tabelle 6.2: MMU-Kommandos

Die MMU-Kommandos bestehen aus der Gruppe der Setz-/Rücksetzkommandos und der Gruppe der Lese-/Schreibkommandos.

Assembler-Syntax: **SIN<B>** Register,E/A-Adresse
SOUT<B> E/A-Adresse,Register

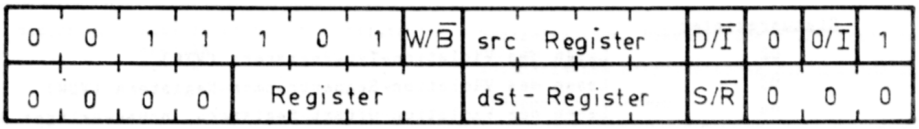
Bitschema:



Ausgabe/Eingabe (Output/Input)	**Datentyp** (Wort/Byte)	**Befehl**
Ausgabe (O/$\overline{I}$ = 1)	Wort (W/$\overline{B}$ = 1)	SOUT
	Byte (W/$\overline{B}$ = 0)	SOUTB
Eingabe (O/$\overline{I}$ = 0)	Wort (W/$\overline{B}$ = 1)	SIN
	Byte (W/$\overline{B}$ = 0)	SINB

Assembler-Syntax: **SIND<R><B>** dst-Register, src-Register, Register
SINI<R><B> dst-Register, src-Register, Register
SO<U>TD<R><B> dst-Register, src-Register, Register
SO<U>TI<R><B> dst-Register, src-Register, Register

Bitschema:



De-/Inkrementieren	**Modus** (Single/Repeat)	**Aus-/Eingabe** (Output/Input)	**Datentyp** (Wort/Byte)	**Befehl**
Dekrement (D/$\overline{I}$ = 1)	Einzelbyte (S/$\overline{R}$ = 1)	Ausgabe (O/$\overline{I}$ = 1)	Wort (W/$\overline{B}$ = 1)	SOUTD
			Byte (W/$\overline{B}$ = 0)	SOUTDB
		Eingabe (O/$\overline{I}$ = 0)	Wort (W/$\overline{B}$ = 1)	SIND
			Byte (W/$\overline{B}$ = 0)	SINDB
	Repetieren (S/$\overline{R}$ = 0)	Ausgabe (O/$\overline{I}$ = 1)	Wort (W/$\overline{B}$ = 1)	SOTDR
			Byte (W/$\overline{B}$ = 0)	SOTDRB
		Eingabe (O/$\overline{I}$ = 0)	Wort (W/$\overline{B}$ = 1)	SINDR
			Byte (W/$\overline{B}$ = 0)	SINDRB
Inkrement (D/$\overline{I}$ = 0)	Einzelbyte (S/$\overline{R}$ = 1)	Ausgabe (O/$\overline{I}$ = 1)	Wort (W/$\overline{B}$ = 1)	SOUTI
			Byte (W/$\overline{B}$ = 0)	SOUTIB
		Eingabe (O/$\overline{I}$ = 0)	Wort (W/$\overline{B}$ = 1)	SINI
			Byte (W/$\overline{B}$ = 0)	SINIB
	Repetieren (S/$\overline{R}$ = 0)	Ausgabe (O/$\overline{I}$ = 1)	Wort (W/$\overline{B}$ = 1)	SOTIR
			Byte (W/$\overline{B}$ = 0)	SOTIRB
		Eingabe (O/$\overline{I}$ = 0)	Wort (W/$\overline{B}$ = 1)	SINIR
			Byte (W/$\overline{B}$ = 0)	SINIRB

Bild 6.3: Syntax und Bitschema der Spezial-E/A-Befehle

Setz-/Rücksetzkommandos

(Rücksetzen der Register MR, DSCR oder VTR, der Flags SWW oder FATL und Setzen aller CPUI- oder DMAI-Flags)

Diese Kommandos setzen oder löschen bestimmte Flags oder Register in der MMU. Bei ihnen erfolgt keine Datenübergabe, und es ist nur der Spezial-E/A-Befehl "Spezialausgabe" (SOUT oder SOUTB) erforderlich. Das Ziel wird direkt adressiert, und diese direkte Adresse ist das MMU-Kommandowort. Die Quelle ist ein beliebiges Register. Der Inhalt dieses Registers wird zwar in der Datenphase des SOUT-Befehls auf dem Datenbus ausgegeben, von der MMU aber ignoriert.

Beispiel:

```
SOUT %11FC,R0      !Rücksetzen VTR der MMU1!
```

Das höherwertige Byte der Zieladresse ist der MMU-Befehlskode %11, also das Kommando "Rücksetzen des VTR". Das niederwertige Byte selektiert die MMU1. Der Inhalt von R0 erscheint zwar auf dem Datenbus, wird aber nicht ausgewertet.

Lese-/Schreibkommandos

Diese Kommandos sind komplizierter als die Setz-/Rücksetzkommandos, da neben dem Kommando Daten übertragen werden müssen. Die Kommandos werden der MMU wieder mit Hilfe von Spezial-E/A-Befehlen übergeben. Im Gegensatz zu den Setz-/Rücksetzkommandos ist bei den Lese/Schreibkommandos nicht nur das MMU-Kommando (die Spezial-E/A-Adresse) von Bedeutung, sondern auch der Inhalt von Quelle bzw. Ziel (die Daten).
Lesekommandos verwenden Eingabebefehle. Der Quelloperand bildet das MMU-Kommandowort, und im Zieloperand werden die von der MMU empfangenen Daten abgelegt. (Bei Registeradressierung erfolgt dies in einem Register, bei indirekter Adressierung auf der Stelle im Speicher, auf die das Register weist.)
Bei Schreibkommandos werden Ausgabebefehle verwendet. Hier bildet der Zieloperand das MMU-Kommandowort, während der Quelloperand die Daten enthält, die der MMU übergeben werden sollen. (Bei Registeradressierung befinden sich diese in einem Register, bei indirekter Adressierung auf der Stelle im Speicher, auf die das Register weist.)

Beispiel:

```
SOUTB %01FC,RH0    !Inhalt von RH0 in das SAR der MMU1 laden!
```

In diesem Beispiel soll RH0 %07 enthalten. Das höherwertige Byte der Ziel-E/A-Adresse ist der Befehlskode %01, also das Kommando "Lesen/Schreiben des SAR". Das niederwertige Byte wählt wie im vorhergehenden Beispiel die MMU1 aus, denn es wird weiter angenommen, das die einfache Dekodiermethode angewendet wird, bei der ein rückgesetztes Bit i die MMUi selektiert. Das Bitschema des Befehls ist im Bild 6.5 dargestellt (vgl. auch "CPU U8001/U8002 - Befehlsbeschreibung").

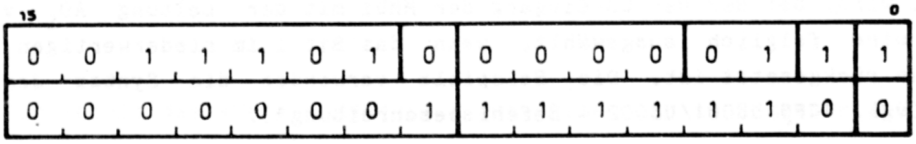


Bild 6.5: Bitschema des Befehls SOUTB %01FC,RH0

Das Beispiel soll anhand des in Bild 6.6 dargestellten zeitlichen Verlaufs eines Spezial-Ausgabebefehls in seinen Phasen erklärt werden.

1. Das $\overline{CS}$-Signal wird aus dem niederwertigen Byte auf den Adreß-/Datenleitungen gebildet. Es liegt an dem $\overline{CS}$-Pin während des T_1- und einem Teil des T_2-Zyklus an. Das Signal gibt die MMU für die Übernahme eines Kommandos frei.
2. Die Statusleitungen führen den Kode für einen Spezial-E/A-Befehl %3 (0011). Der Statuskode wird im T_1-Zyklus ausgegeben und bleibt bis zum Abschluß des T_3-Zyklus gültig.

3. Das höherwertige Byte des Adreß-/Datenbusses ($AD_8 \ldots AD_{15}$) führt den Befehlskode %01. Dieses Signal wird wie das $\overline{CS}$-Signal während des T_1-Zyklus und einem Teil von T_2 ausgegeben.
4. Die Adreß-/Datenleitungen ändern ihren Zustand und führen das Datenbyte (%07), daß in die MMU eingeschrieben werden soll. Dieses Signal ist bis Abschluß des T_3-Zyklus gültig.

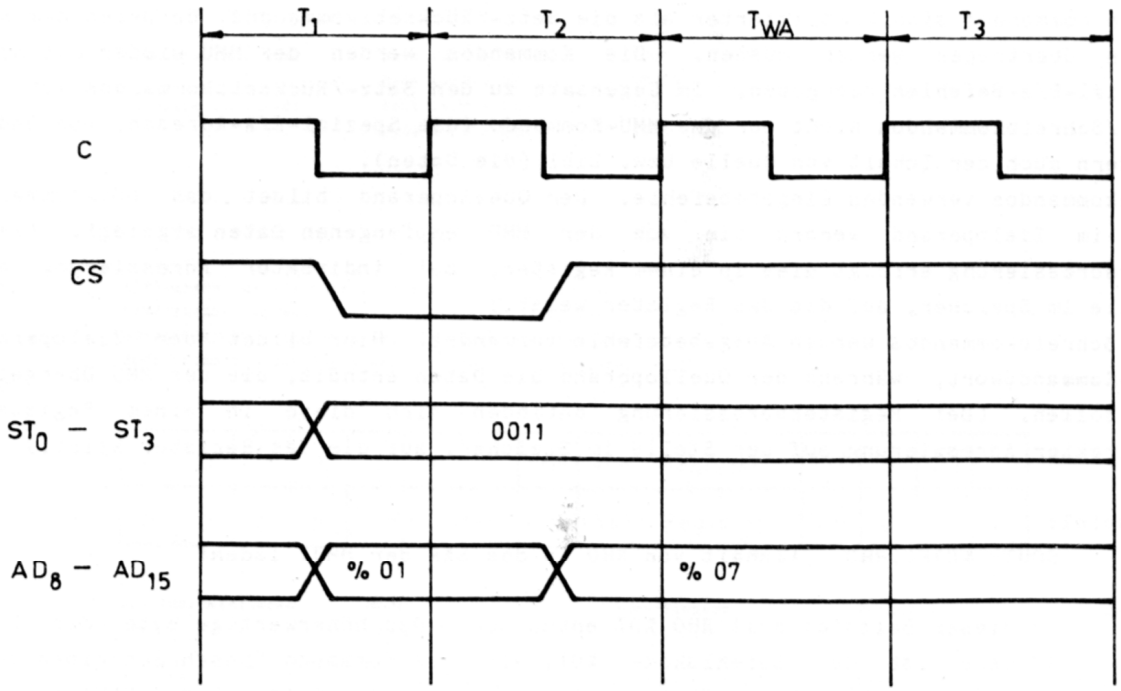


Bild 6.6: Zeitablauf des Befehls SOUTB %01FC,RH0

In den folgenden Unterpunkten werden die MMU-Kommandos einzeln beschrieben, wobei jeweils ein Beispiel angeführt wird. Bei den Beispielen wird wieder angenommen, daß eine Selektierung angewendet wird, bei der der $\overline{CS}$-Eingang der MMUi mit der Leitung AD_i verbunden ist. Die MMUi wird folglich ausgewählt, wenn das Bit i im niederwertigen Byte der Spezial-E/A-Adresse rückgesetzt ist. Die Beispiele verwenden die Syntax der U8000-Assemblersprache (vgl. "CPU U8001/U8002 - Befehlsbeschreibung").

6.2.1. Lesen/Schreiben des Modus-Registers (MR)

Befehlskode: %00

Dieses Kommando wird verwendet, um das 8-Bit-Modus-Register (MR) zu lesen oder zu beschreiben. Da bei diesem Kommando nur ein Datenbyte zu transportieren ist, wird normalerweise der Spezial-E/A-Befehl SINB bzw. SOUTB verwendet.

Beispiel:

Das Modus-Register der MMU2 soll mit dem Wert %E2 initialisiert werden.

```
LDB RH0,#%E2
SOUTB %00FA,RH0          !Befehlskode %00: Lesen/Schreiben des MR;
                          Auswahlkode %FA: Selektierung der MMU2!
```

(Mit dem Befehl SOUT %00FA,R0 könnte die gleiche Wirkung erzielt werden.)

6.2.2. Lesen/Schreiben des Segment-Adreß-Registers (SAR)

Befehlskode: **%01**
Dieses Kommando wird verwendet, um das 8-Bit-Segment-Adreß-Register (SAR) zu lesen oder zu beschreiben. Da auch bei diesem Kommando nur ein Datenbyte zu transportieren ist, wird der Spezial-E/A-Befehl SINB bzw. SOUTB verwendet.

Beispiel:

Der Inhalt des SAR der MMU1 soll ausgelesen und im Register RH0 abgelegt werden.

```
SINB RH0,%01FC           !%FC wählt die MMU1 aus!
```

(Mit dem Befehl SIN R0,%01FC könnte ebenfalls der Inhalt des SAR in RH0 abgelegt werden. Der Inhalt von RL0 würde dann aber mit verändert.)

6.2.3. Lesen des Violation-Typ-Registers (VTR)

Befehlskode: **%02**
Dieses Kommando wird verwendet, um das 8-Bit-Violation-Typ-Register (VTR) zu lesen. Das VTR kann nur gelesen und nicht beschrieben werden. Es ist aber möglich, dieses Register zu löschen (siehe 6.2.17. und 6.2.18.). Normalerweise wird bei diesem Kommando der Spezial-E/A-Befehl SINB verwendet.

Beispiel:

Der Inhalt des VTR der MMU3 soll ausgelesen und im Register RH0 abgelegt werden.

```
SINB RH0,%02F6           !%F6 wählt die MMU3 aus!
```

(Der Befehl SIN R0,%02F6 würde ebenfalls den Inhalt des VTR im Register RH0 ablegen. Der Inhalt von RL0 würde dabei aber mit verändert.)

6.2.4. Lesen des Violation-Segment-Nummer-Registers (VSN)

Befehlskode: **%03**
Dieses Kommando wird verwendet, um das 8-Bit-Violation-Segment-Nummer-Register (VSN) zu lesen. Das VSN ist ein weiteres Register, daß nur gelesen und nicht beschrieben werden kann. Normalerweise wird bei diesem Kommando der Spezial-E/A-Befehl SINB verwendet.

Beispiel:

Der Inhalt des VSN der MMU2 soll ausgelesen und im Register RH0 abgelegt werden.

```
SINB RH0,%03FA
```

6.2.5. Lesen des Violation-Offset-Registers (VOFF)

Befehlskode: **%04**
Dieses Kommando wird verwendet, um das 8-Bit-Violation-Offset-Register (VOFF) zu lesen. Das Register ist nur lesbar. Am besten wird hier auch der Spezial-E/A-Befehl SINB verwendet.

Beispiel:

Der Inhalt des VOFF der MMU1 soll ausgelesen und im Register RL0 abgelegt werden.

```
SINB RL0,%04FC
```

6.2.6. Lesen des Bus-Zyklus-Status-Registers (BCSR)

Befehlskode: **%05**
Dieses Kommando wird verwendet, um das 8-Bit-Bus-Zyklus-Status-Register (Bus Cycle Status Register, BCSR) zu lesen. Es handelt sich ebenfalls um ein nur lesbares Register und man sollte den Spezial-E/A-Befehl SINB verwenden.

Beispiel:

Der Inhalt des BCSR der MMU7 soll ausgelesen und im Register RH0 abgelegt werden.

```
SINB RH0,%057E
```

6.2.7. Lesen des Instruktion-Segment-Nummer-Registers (ISN)

Befehlskode: **%06**
Dieses Kommando wird verwendet, um das 8-Bit-Instruktion-Segment-Nummer-Register (ISN) zu lesen. Das Register ist nur lesbar und auch für dieses Kommando ist es am sinnvollsten den SINB-Befehl zu verwenden.

Beispiel:

Der Inhalt des ISN der MMU1 soll ausgelesen und im Register RH7 abgelegt werden.

```
SINB RH7,%06FC
```

6.2.8. Lesen des Instruktion-Offset-Register (IOFF)

Befehlskode: **%07**
Dieses Kommando wird verwendet, um das 8-Bit-Instruktion-Offset-Register (IOFF) zu lesen. Es ist wiederum nur lesbar und gewöhnlich wird hierfür der Spezial-E/A-Befehl SINB verwendet.

Beispiel:

Der Inhalt des IOFF der MMU2 soll ausgelesen und im Register RL1 abgelegt werden.

```
SINB RL1,%07FA
```

6.2.9. Lesen/Schreiben des Basisadreßfeldes im Deskriptor

Befehlskode: **%08**
Dieses Kommando wird verwendet, um das 16-Bit-Basisadreßfeld des Segment-Deskriptor-Registers, auf das das Segment-Adreß-Register zeigt, zu lesen oder zu beschreiben. Da zwei Datenbytes zu übertragen sind, sollte ein repetierender Spezial-E/A-Befehl (SINIRB, SOTIRB, SINDRB oder SOTDRB) verwendet werden. Natürlich wird zum Lesen ein Eingabebefehl und zum Schreiben ein Ausgabebefehl eingesetzt. Die MMU behandelt zuerst das höherwertige Byte der Basisadresse. Vor der Ausgabe des Kommandos muß garantiert sein, daß das DSCR rückgesetzt ist (DSCR = 0), d.h. es muß auf das höherwertige Byte zeigen. Nach der Übertragung des Byte erhöht die MMU das DSCR (DSCR = 1), so daß es auf das niederwertige Byte des Basisadreßfeldes zeigt. Bei korrekter Beendigung des Kommandos wird das DSCR automatisch wieder auf Null gebracht. Dies ist nicht der Fall, wenn nur ein Datenbyte übertragen wurde. (Im Abschnitt 7.4. sind dazu nähere Angaben zu finden.)
Soll das höherwertige Byte der 2-Byte-Basisadresse im Speicher auf der geraden Adresse abgelegt werden, so ist die inkrementierende Form der Spezial-E/A-Befehle zu verwenden. Das SAR bestimmt, auf welchen Deskriptor der Zugriff erfolgt. Es muß deshalb so initialisiert werden, daß es auf den gewünschten Deskriptor zeigt.

Beispiel:

Von der MMU1 soll der Inhalt des Basisadreßfeldes vom Segment-Deskriptor des Segments %25 ausgelesen werden. Es soll auf der Adresse BASIS beginnend im Speicher abgelegt werden. (Die Initialisierung des DSCR ist im Abschnitt 7.4. erläutert.)

```
LDB RL0,#%25            !Laden der Segment-Deskriptor-Nummer!
SOUTB %01FC,RL0         !Initialisieren des SAR!
LDA RR2,BASIS           !Zeiger auf Speicherfeld!
LD R1,#%08FC            !Laden des Kommandos "Lesen/Schreiben des Basisadreßfeldes
                        der MMU1"!
LD R0,#2                !Laden der Anzahl der zu transportierenden Bytes!
SINIRB@RR2,@R1,R0       !Lesen der Bytes des spezifizierten Basisadreßfeldes!
```

Nach Abarbeitung dieser Routine enthält der Speicherplatz BASIS das höherwertige Byte des Basisadreßfeldes vom Segment-Deskriptor %25 der MMU1. Auf dem Speicherplatz BASIS+1 befindet sich dessen niederwertiges Byte. Das SAR enthält %25, das Register R0 steht auf Null und RR2 zeigt auf BASIS+2.

6.2.10. Lesen/Schreiben des Limitfeldes im Deskriptor

Befehlskode: **%09**
Dieses Kommando wird verwendet, um das 8-Bit-Limitfeld des Segment-Deskriptor-Registers, auf das das Segment-Adreß-Register zeigt, zu lesen oder zu beschreiben. Da nur ein Datenbyte zu übertragen ist, wendet man am besten den SINB- oder SOUTB-Befehl an. Bei der Ausführung dieses Befehls setzt die MMU das DSCR automatisch, so daß es auf das Limitfeld (drittes Byte) des Segment-Deskriptors zeigt. Darum braucht sich der Anwender nicht zu kümmern. Jedoch muß er das SAR initialisieren, so daß es auf den gewünschten Deskriptor zeigt. Nach Beendigung des Kommandos wird das DSCR automatisch auf Null gestellt.

Beispiel:

Der Wert, den die Speicherstelle LIM enthält, soll in das Limitfeld des Segment-Deskriptors %08 der MMU2 eingetragen werden.

```
LDB RH0,LIM             !Laden des Limitwertes!
LDB RL0,#%08            !Laden der Segment-Deskriptor-Nummer!
SOUTB %01FA,RL0         !Initialisieren des SAR!
SOUTB %09FA,RH0         !Beschreiben des Limitfeldes!
```

6.2.11. Lesen/Schreiben des Attributfeldes im Deskriptor

Befehlskode: **%0A**
Dieses Kommando ist dem vorhergehenden sehr ähnlich. Es wird verwendet, um das 8-Bit-Attributfeld des Segment-Deskriptor-Registers, auf das das Segment-Adreß-Register zeigt, zu lesen oder zu beschreiben. Wieder wird nur ein Datenbyte übertragen, so daß am besten der SINB- oder SOUTB-Befehl verwendet wird. Der Anwender muß das SAR auf den gewünschten Deskriptor initialisieren. Das Attributfeld ist das vierte Byte im Segment-Deskriptor. Von der MMU wird das DSCR automatisch auf das Attributfeld gerichtet und nach Ausführung des Befehls wieder auf Null gestellt.

Beispiel:

Die Nummer des Segment-Deskriptors befinde sich auf der Speicherstelle SEG_DES_NR. Von der MMU3 soll der Inhalt des Attributfeldes dieses Segment-Deskriptors in das Register RH1 überführt werden.

```
LDB RL0,SEG_DES_NR      !Laden der Segment-Deskriptor-Nummer!
SOUTB %01F6,RL0         !Initialisieren des SAR!
SINB RH1,%0AF6          !Lesen des Attributfeldes!
```

6.2.12. Lesen/Schreiben aller Felder des Deskriptors

Befehlskode: **%0B**
Dieses Kommando wird verwendet, um das vollständige 4-Byte-Segment-Deskriptor-Register, auf das das SAR zeigt, zu lesen oder zu beschreiben. Da nicht nur ein Datenbyte, sondern vier zu übertragen sind, sollte ein repetierender Spezial-E/A-Befehl (SINIRB, SOTIRB, SINDRB oder SOTDRB) angewendet werden. Die MMU behandelt zuerst das höherwertige Byte des Basisadreßfeldes. Wenn dieses Byte auf der niederwertigen Speicherstelle abgelegt ist bzw. werden soll, so ist die inkrementierende Form des Spezial-E/A-Befehls

einzusetzen (SINIRB bzw. SOTIRB). Es wird der Deskriptor angesprochen, auf den das SAR zeigt, so daß dieses vor Ausführung des Kommandos initialisiert werden muß. Das DSCR muß vor Ausführung des Kommandos auf Null gestellt werden, damit es auf das höherwertige Byte des Basisadreßfeldes weist. Bei Abarbeitung des Kommandos wird das DSCR automatisch erhöht und zeigt in folgender Reihenfolge und mit folgenden Werten auf die Felder des Deskriptors:

0: höherwertiges Byte des Basisadreßfeldes
1: niederwertiges Byte des Basisadreßfeldes
2: Limitfeld
3: Attributfeld

Nach Beendigung des Kommandos wird unter der Voraussetzung, daß alle vier Datenbytes übertragen wurden, das DSCR automatisch auf Null gestellt (siehe Abschnitt 7.4).

Beispiel:

Beginnend auf der Adresse DESK sind im Speicher vier Datenbytes abgelegt. Diese sollen in das Segment-Deskriptor-Register 35 der MMU3 eingetragen werden (vgl. Abschnitt 7.4. bezüglich Initialisierung des DSCR).

```
LDB RH0,#35             !Laden der Segment-Deskriptor-Nummer!
SOUTB %01F6,RH0         !Initialisieren des SAR!
LDA RR2,DESK            !Laden des Zeigers auf die Datenbytes!
LD R1,#%0BF6            !Laden des Kommandos "Lesen/Schreiben des Deskriptors
                         der MMU3"!
LD R0,#4                !Laden der Anzahl der zu übertragenden Bytes!
SOTIRB @R1,@RR2,R0      !Übertragung des Deskriptors!
```

6.2.13. Lesen/Schreiben des Basisadreßfeldes und Erhöhen des SAR

Befehlskode: **%0C**

Dieses Kommando hat dieselbe Funktion wie das im Abschnitt 6.2.9. beschriebene. Der Unterschied besteht darin, daß hier nach dem Lesen oder Beschreiben des Basisadreßfeldes im Segment-Deskriptor das SAR erhöht wird, so daß es auf den nächsten Deskriptor zeigt. Wird eine repetierende Form der Spezial-E/A-Befehle verwendet, so ist es dem Anwender möglich, aufeinanderfolgende Deskriptor-Register anzusprechen. Dabei folgt dem Deskriptor 63 der Deskriptor 0. Mit einem Kommando könnten alle Basisadreßfelder der MMU beschrieben oder gelesen werden.
Das DSCR muß vor Ausgabe dieses Kommandos Null sein. Nach jeder Erhöhung des SAR stellt die MMU das DSCR so ein, daß es wieder auf das höherwertige Byte der Basisadresse zeigt. Nach Beendigung des Kommandos steht das DSCR auf Null, es sei denn, es wurde eine ungerade Anzahl von Datenbytes übertragen.

Beispiel:

Die 64 Basisadreßfelder der MMU1 sollen beschrieben werden. Es wird vorausgesetzt, daß die Daten im Speicher so zur Verfügung stehen, wie dies im Bild 6.7 dargestellt ist (vgl. Abschnitt 7.4. bezüglich Initialisierung des DSCR).

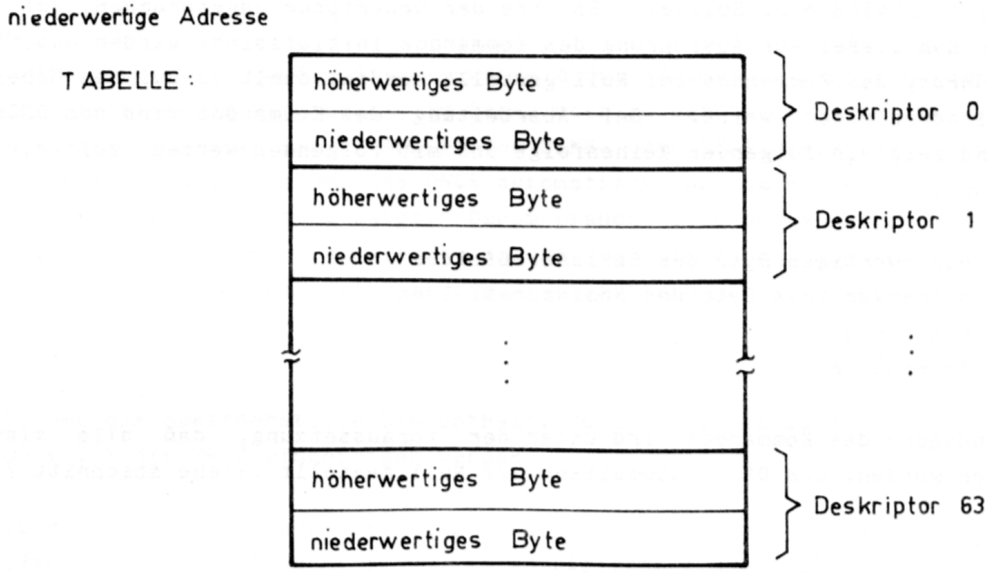


Bild 6.7: Anordung der Daten im Speicher für die Basisadreßfelder der MMU

```
CLR R0                  !Beginn mit Deskriptor 0!
SOUTB %01FC,RH0         !Initialisieren des SAR!
LDA RR2,TABELLE         !Zeiger auf Speicherfeld!
LD R1,#%0CFC            !Laden des Kommandos!
LD R0,#128              !Laden der Anzahl der Bytes!
SOTIRB @R1,@RR2,R0      !Beschreiben der Basisadreßfelder!
```

Nach der Ausführung haben die CPU-Register folgenden Zustand:
R0 = 0, RR2 = TABELLE+128, R1 = %0CFC

6.2.14. Lesen/Schreiben des Limitfeldes und Erhöhen des SAR

Befehlskode: **%0D**
Dieses Kommando wirkt wie das im Abschnitt 6.2.10. beschriebene mit der Ausnahme, daß nach dem Zugriff auf das Limitfeld eine Erhöhung des SAR stattfindet. Dem Deskriptor 63 folgt dabei der Deskriptor 0. Dieses Kommando ermöglicht in Verbindung mit einem repetierenden Spezial-E/A-Befehl den Zugriff auf alle Limitfelder der MMU. Die MMU sorgt neben der Erhöhung des SAR dafür, daß das DSCR auf das Limitfeld zeigt. Nach Beendigung des Kommandos steht das DSCR auf Null.

Beispiel:

Die ersten 30 Limitfelder der MMU3 sollen im Speicher beginnend auf der Adresse LIMIT abgelegt werden.

```
CLR R0                  !Beginn mit Deskriptor 0!
SOUTB %01F6,RL0         !Initialisieren des SAR!
LDA RR4,LIMIT           !Zeiger auf Speicherfeld!
LD R1,#%0DF6            !Laden des Kommandos!
LD R0,#30               !Anzahl der Bytes!
SINIRB @RR4,@R1,R0      !Lesen der Limitfelder!
```

Nach der Ausführung haben die CPU-Register folgenden Zustand:
R0 = 0, RR4 = LIMIT+30, R1 = %0DF6

6.2.15. Lesen/Schreiben des Attributfeldes und Erhöhen des SAR

Befehlskode: **%0E**
Mit dem Unterschied, daß dieses Kommando auf das Attributfeld und nicht auf das Limitfeld wirkt, entspricht es dem vorhergehenden.

Beispiel:

Alle 64 Attributfelder der MMU4 sollen mit den Werten beschrieben werden, die im Speicher beginnend auf der Adresse ATTRIB zu finden sind.

```
CLR R0                  !Beginn mit Deskriptor 0!
SOUTB %01EE,RH0         !Initialisieren des SAR!
LDA RR4,ATTRIB          !Zeiger auf Speicherfeld!
LD R1,#%0EEE            !Laden des Kommandos!
LD R0,#64               !Anzahl der Bytes!
SOTIRB @R1,@RR4,R0      !Beschreiben der Attributfelder!
```

Nach der Ausführung liegt folgender Zustand in den CPU-Registern vor:
R0 = 0, RR4 = ATTRIB+64, R1 = %0EEE

6.2.16. Lesen/Schreiben des Deskriptors und Erhöhen des SAR

Befehlskode: **%0F**
Dieses Kommando greift auf die vier Bytes des Deskriptors zu und erhöht danach das SAR, so daß es auf den nachfolgenden Deskriptor zeigt. (Deskriptor 0 folgt dem Deskriptor 63.) Wie bei den drei vorhergehenden Kommandos kann man auch hier, mit nur einem repetierenden E/A-Befehl, auf alle 64 Deskriptoren der MMU zugreifen.
Bevor dieses Kommando ausgegeben wird, muß das DSCR auf Null gestellt werden. Nach jeder Erhöhung des SAR bringt die MMU das DSCR wieder in den Zustand, daß es auf das höherwertige Byte des Basisadreßfeldes zeigt. Nach Beendigung des Kommandos steht damit das DSCR auf Null, es sei denn, nicht alle vier Bytes eines Deskriptors wurden behandelt (vgl. Abschnitt 7.4.).

Beispiel:

Alle 64 Deskriptor-Register der MMU1 sind zu initialisieren. Es wird vorausgesetzt, daß die Daten, wie in Bild 6.8 dargestellt, im Speicher angeordnet sind.

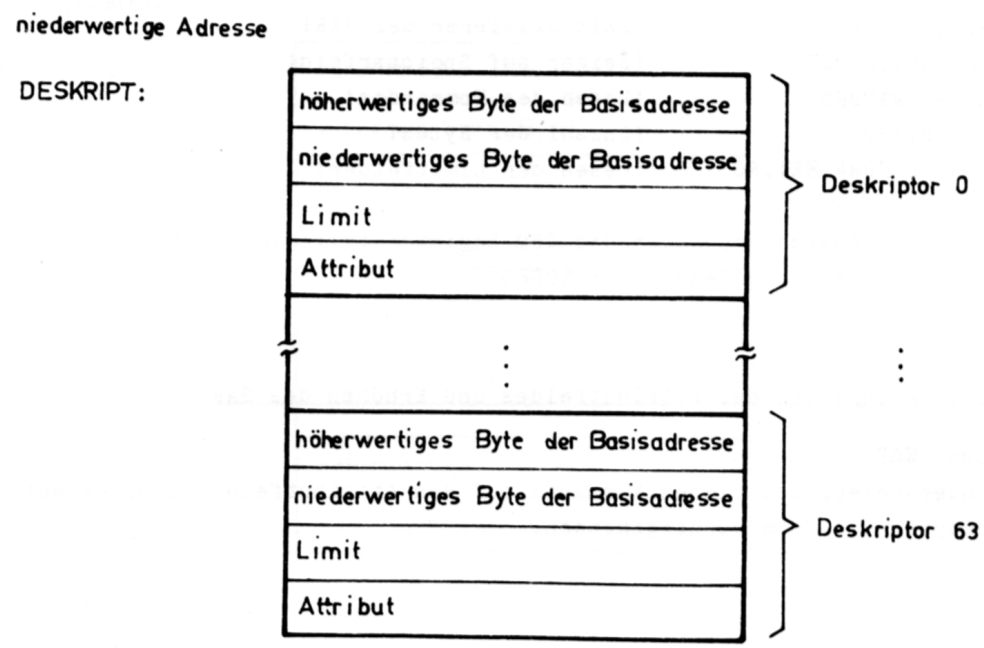


Bild 6.8: Anordnung der Daten für die Deskriptoren im Speicher

```
CLR R0                  !Beginn mit Deskriptor 0!
SOUT %01FC,R0           !Initialisieren des SAR!
LDA RR4,DESKRIPT        !Zeiger auf Datenfeld!
LD R1,#%0FFC            !Laden des Kommandos!
LD R0,#256              !Anzahl der Bytes!
SOTIRB @R1,@RR4,R0      !Beschreiben der Deskriptoren!
```

Nach der Ausführung liegt folgender Zustand vor:
R0 = 0, RR4 = DESKRIPT+256, R1 = %0FFC

Der SOTIRB-Befehl gibt alle 256 Bytes des Datenblockes an die MMU aus. Da das SAR auf Null gestellt wurde, werden mit dem Befehl "Lesen/Schreiben des Deskriptors und Erhöhen des SAR" die ersten vier Bytes des Datenblockes in das Deskriptor-Register 0 eingeschrieben. Danach wird das SAR erhöht und die nächsten vier Bytes werden in das Deskriptor-Register 1 gebracht. Dieser Prozeß wiederholt sich, bis alle 256 Bytes den 64 Deskriptor-Registern übergeben wurden. Das Deskriptor-Selektion-Counter-Register (DSCR) zeigt in folgender Reihenfolge und mit folgenden Werten auf die Bytes des Deskriptor-Registers:

0: höherwertiges Byte des Basisadreßfeldes
1: niederwertiges Byte des Basisadreßfeldes
2: Limitfeld
3: Attributfeld

Der Programmierer braucht sich um das DSCR nicht zu kümmern, denn es wird von der MMU automatisch verwaltet. Es muß aber beachtet werden, daß die repetierenden Befehle durch Interrupt unterbrochen werden können. Dies ist auch notwendig, denn allein bei diesem Beispiel würden 11 + (10 * 256) = 2571 Taktzyklen zur

Ausführung des SOTIRB-Befehls benötigt. Sollte in der Interruptserviceroutine eine Veränderung des DSCR erfolgen, so muß dessen Inhalt vorher gerettet und zum Abschluß wieder regeneriert werden, so daß der SOTIRB-Befehl nach Behandlung des Interrupt korrekt fortgesetzt werden kann.

6.2.17. Reset (Rücksetzen des MR, VTR und DSCR)

Befehlskode: **%10**
Dieses Kommando hat auf die MMU die Wirkung wie ein Hardware-Reset. Es bringt die MMU in einen definierten Zustand. Alle Bits im MR, VTR und DSCR werden auf Null gestellt. Bei diesem Kommando werden keine Daten zur MMU übertragen. Am zweckmäßigsten wendet man den SOUT- oder SOUTB-Befehl an.

Beispiel:

Die MMU1 soll rückgesetzt werden.

```
SOUT %10FC,R0
```

Das Register R0 wurde willkürlich als Quelle gewählt. Dessen Inhalt erscheint zwar während der Datenphase des SOUT-Befehls auf dem Datenbus, wird aber von der MMU ignoriert.

6.2.18. Rücksetzen des Violation-Typ-Registers (VTR)

Befehlskode: **%11**
Zum Löschen des Violation-Typ-Registers (VTR) wird dieses Kommando verwendet. Im VTR werden damit alle Bits auf Null gestellt. Bei diesem Kommando werden keine Daten zur MMU übertragen. Am zweckmäßigsten wendet man den SOUT- oder SOUTB-Befehl an. Dieses Kommando bringt die MMU in den Normalzustand (als ob keine Violation aufgetreten wäre).

Beispiel:

Das VTR der MMU1 soll rückgesetzt werden.

```
SOUT %11FC,R0
```

Das Register R0 wurde willkürlich als Quelle gewählt. Dessen Inhalt erscheint zwar während der Datenphase des SOUT-Befehls auf dem Datenbus, wird aber von der MMU ignoriert.

6.2.19. Rücksetzen des SWW-Flags im Violation-Typ-Register

Befehlskode: **%13**
Mit diesem Kommando wird das SWW-Flag im VTR gelöscht. Es werden keine Daten übergeben. Der SOUT- oder SOUTB-Befehl sollte verwendet werden.

Beispiel:

Das SWW-Flag der MMU2 ist rückzusetzen.

```
SOUTB %13FA,RH0
```

RH0 wurde wieder willkürlich gewählt. Sein Inhalt erscheint zwar auf dem Datenbus, wird aber von der MMU nicht ausgewertet.

6.2.20. Rücksetzen des FATL-Flags im Violation-Typ-Register

Befehlskode: **%14**
Dieses Kommando löscht das FATL-Flag im VTR. Ansonsten arbeitet es wie das vorhergehende Kommando.

Beispiel:

Das FATL-Flag der MMU1 ist zu löschen.

```
SOUTB %14FC,RH0
```

Welches Register als Quelle spezifiziert wird ist beliebig, da sein Inhalt von der MMU nicht ausgewertet wird.

6.2.21. Setzen aller CPUI-Flags

Befehlskode: **%15**
Dieses Kommando sorgt dafür, daß die CPU-Inhibit-Flags in allen Segment-Deskriptoren der MMU gesetzt werden. Der MMU müssen dazu keine Daten übergeben werden. Der SOUT- oder SOUTB-Befehl ist für dieses Kommando optimal. Dieses und das nächste Kommando sind sinnvoll für die Initialisierung von Adreßübersetzungstabellen oder zur Task-Umschaltung. Dies sei an folgendem Beispiel erläutert: In einem System verwalten zwei MMUs alle Anwendersegmente. Soll der Übergang von einer Task auf eine andere erfolgen, so können durch das Kommando "Setzen aller CPUI-Flags" Zugriffe der nachfolgenden Task auf Segmente der vorhergehenden verhindert werden. Dazu müssen erst Attributfelder von Segmenten der vorhergehenden Task gezielt verändert werden.

Beispiel:

Alle CPUI-Flags der MMU3 und der MMU4 sollen gesetzt werden.

```
SOUT %15E6,R0          !Auswahlkode %E6 selektiert gleichzeitig MMU3
                        und MMU4!
```

Register R0 wurde wieder willkürlich gewählt.

6.2.22. Setzen aller DMAI-Flags

Befehlskode: **%16**
Mit diesem Kommando wird bewirkt, daß alle DMA-Inhibit-Flags der MMU gesetzt werden. Auch hier werden keine Daten übertragen. Am sinnvollsten ist die Anwendung des SOUT- oder SOUTB-Befehls.

Beispiel:

Von der MMU1 sollen alle DMAI-Flags gesetzt werden.

```
SOUT %16FC,R0          !Setzen aller DMAI-Flags in der MMU1!
```

6.2.23. Lesen/Schreiben des Deskriptor-Selektion-Counter-Registers (DSCR)

Befehlskode: **%20**
Mit diesem Kommando ist es möglich, das 2-Bit-DSCR zu lesen oder zu beschreiben. Es muß nur ein Datenbyte übertragen werden, so daß am besten der Befehl SOUTB bzw. SINB verwendet wird.

Beispiel:

Der Inhalt des DSCR der MMU1 soll in das Register RH1 überführt werden.

```
SINB RH1,%20FC
```

7. Besonderheiten

7.0. Einführung

In diesem Kapitel sind verschiedene Besonderheiten zusammengefaßt, die bereits in vorangehenden Kapiteln erwähnt wurden, hier aber detailliert erläutert werden sollen.

7.1. Direkter Speicherzugriff (DMA)

Neben den Zugriffen der CPU U8001 können DMA-Zugriffe durch die MMU behandelt werden. Die MMU unterstützt DMA-Operationen sowohl im Systemmodus als auch im Normalmodus. Für jeden Speicherzugriff erfolgt eine Überprüfung der Segmentattribute und bei Feststellung einer Violation wird das Suppress-Signal ($\overline{SUP}$) erzeugt. Im Gegensatz zur Violation bei CPU-Zugriffen, wo in nachfolgenden Speicherzugriffen bis zum Ende des Befehls wiederholt das $\overline{SUP}$-Signal aktiviert wird, erfolgt bei Violation in DMA-Zugriffen nur in dem Zyklus selbst die Erzeugung von $\overline{SUP}$. Die DMA-Einheit oder externe Hardware sollte auf das $\overline{SUP}$-Signal reagieren und die entsprechenden Informationen aufzeichnen, so daß das Betriebssystem den Fehler korrigieren kann. Segmenttrap-Anforderungen werden bei DMA-Zugriffen nicht erzeugt, da die CPU nicht vor Beendigung des DMA-Zugriffes den Trap bestätigen könnte. Folglich werden auch keine Warnungen übermittelt. Die MMU markiert DMA-Zugriffe nicht im CHG- oder REF-Flag des Attributfeldes im Deskriptor-Register .
Der MMU wird der Beginn eines DMA-Zugriffs angezeigt, in dem die DMASYNC-Leitung auf L geht. Ein L-Pegel auf dieser Leitung verhindert, daß die MMU eine ungültige Segmentnummer von den Leitungen $SN_0 \ldots SN_6$ übernimmt. Bei Gültigkeit der logischen Speicheradresse der DMA-Einheit muß DMASYNC, zur ordnungsgemäßen Ausführung der Adreßübersetzungs- und Zugriffsschutzfunktion der MMU, bei der fallenden Flanke des Taktes H sein. Nach der steigenden Flanke des Taktes muß DMASYNC wieder auf L gehen, um der MMU mitzuteilen, daß ein DMA-Zugriff bearbeitet wird. Im Bild 7.1 ist ein DMA-Zugriff dargestellt. Nähere Informationen sind im Anhang B "Zeitverläufe" zu finden. DMA-Einheiten, die selbst die physische Adressen erzeugen, nutzen die MMU nicht. Liegt ein solches System vor und wird DMASYNC auch hier von der DMA-Einheit gebildet, so muß dafür gesorgt werden, daß DMASYNC während des DMA-Zyklus auf L gehalten wird. Nur so kann garantiert werden, daß die MMU-Adreßausgänge im Tri-State bleiben und die MMU nicht versucht, eine physische Adresse der DMA-Einheit zu übersetzen.
Nach Freigabe des Busses mit Abschluß des DMA-Zugriffs muß DMASYNC wieder auf H gehen. Zwei Taktzyklen danach setzt die MMU voraus, daß die CPU wieder die Steuerung des Busses übernommen hat und das nachfolgende Speicherzugriffe Zugriffe der CPU sind. Der erste CPU-Zugriff erfolgt zwei Taktzyklen nachdem die CPU die Steuerung des Busses zurückerhalten hat. Während des Ablaufs von Zyklen der CPU muß DMASYNC auf H-Pegel sein. Dies signalisiert der MMU, daß CPU-Zugriffe stattfinden.
Mit DMASYNC auf L ist keine ordnungsgemäße Programmierung der MMU möglich.

7.2. Rücksetzen (Reset)

Die MMU kann sowohl durch Hardware als auch mittels Software rückgesetzt werden. Dabei treten die nachfolgend beschriebenen Wirkungen auf. Ein Hardware-Reset wird mit einer fallenden Flanke am $\overline{RESET}$-Pin ausgelöst. Software-Reset wird dagegen durch Kommandos bewirkt.

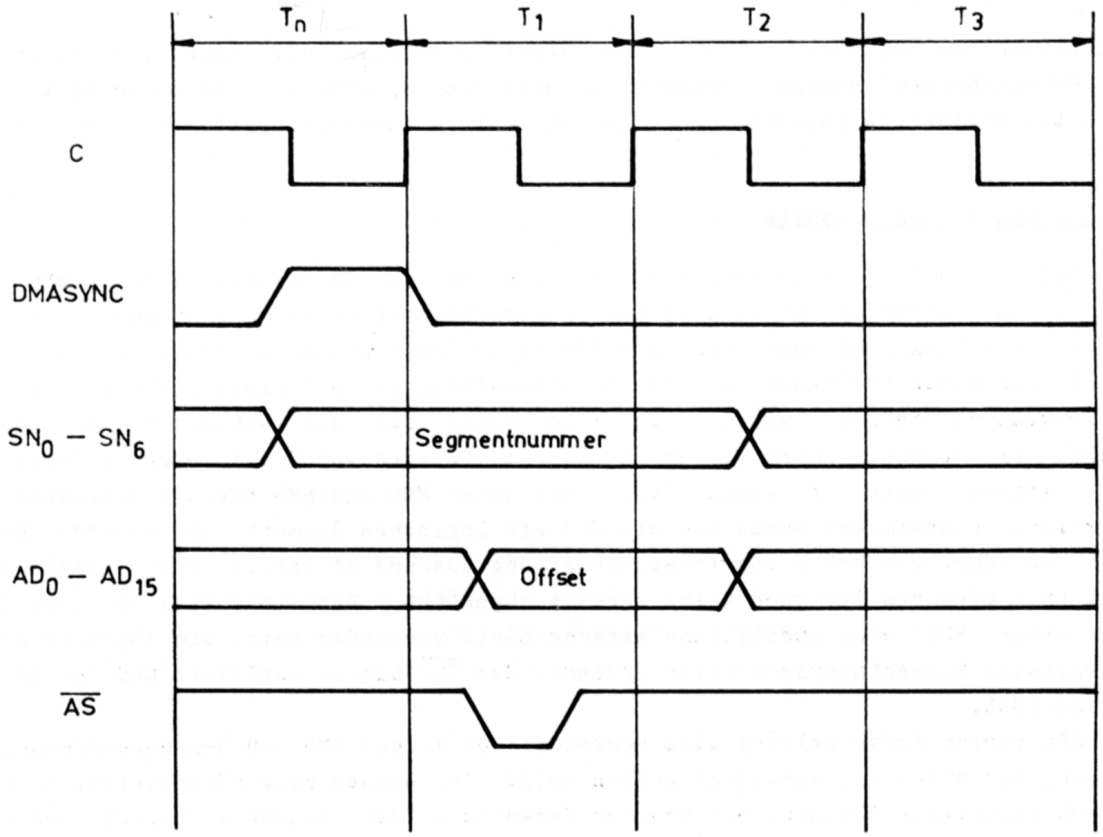


Bild 7.1: Ablauf eines DMA-Zugriffs

7.2.1. Hardware-Reset

Durch Hardware-Reset werden das Modus-Register (MR), das Violation-Typ-Register (VTR) und das Deskriptor-Selektion-Counter-Register (DSCR) gelöscht. Nach Reset sind die AD-Leitungen und die Adreßausgänge $A_8 \ldots A_{23}$ im Tri-State. Die MMU reagiert nicht auf Adressen auf ihren AD-Leitungen, da ihr MSEN-Flag rückgesetzt ist. Die Open-Drain-Ausgänge $\overline{SUP}$ und $\overline{SEGT}$ werden nicht getrieben. Um eine MMU nach einem Hardware-Reset freizugeben, ist ihr MSEN-Flag zu setzen. Dazu muß $\overline{CS}$ an der MMU aktiviert werden und das entsprechende Kommando ausgegeben werden.
Wird die Chip-Auswahl-Leitung ($\overline{CS}$) einer MMU während des Reset-Vorganges aktiviert, so wird im Modus-Register das Master-Enable-Flag (MSEN) gesetzt. Das Translate-Flag (TRNS) wird aber gelöscht. Dieser Zustand hat zur Folge, daß die von der CPU ausgegebene logische Adresse unbeeinflußt die MMU passiert und an deren Ausgängen $A_8 \ldots A_{23}$ ausgegeben wird ($AD_8 \ldots AD_{15}$ auf $A_8 \ldots A_{15}$, $SN_0 \ldots SN_7$ auf $A_{16} \ldots A_{22}$ und A_{23} ist L). Dies erlaubt die Einordnung des Anfangslader-ROMs in den verwalteten Speicher.

7.2.2. Software-Reset

Ein Rücksetzen mittels Software kann durch das Kommando "Reset" (%10) oder durch das Kommando "Rücksetzen des Violation-Typ-Registers" (%11) erfolgen. Diese beiden Kommandos und deren Wirkung sind im Abschnitt 6.2.17. und 6.2.18. beschrieben.

7.3. Übersetzungstabellen

Die Architektur der MMU U8010 erlaubt Systemkonfigurationen mit mehr als einer MMU. Mehrere MMU-Einheiten werden verwendet, um alle 128 Segmente der CPU zu verwalten (eine MMU verwaltet maximal 64 Segmente) oder um mit mehreren Übersetzungstabellen zu arbeiten.

7.3.1. Eine Übersetzungstabelle

Die CPU U8001 erzeugt logische Adressen, mit denen bis zu 128 Segmente angesprochen werden können. Da die MMU nur 64 Segment-Deskriptor-Register enthält, sind zwei MMUs erforderlich um Verwaltungsfunktionen für alle 128 logischen Segmente zu realisieren.
Systeme mit nur einer MMU haben noch die Leistungsfähigkeit und Flexibilität, die durch Speicherverwaltung geboten wird, doch sind Tasks auf die Nutzung von 64 Segmenten beschränkt. Die Segmentnummern müssen entweder im Bereich von 0...63 oder im Bereich von 64...127 liegen. Wenn in einem System mit einer MMU die MMU für die Übersetzung des einen Bereichs programmiert wurde und die CPU ein logisches Segment im anderen Bereich anspricht, so führt die MMU keine Adreßübersetzung aus und es erfolgt keine Adreßausgabe. In diesem Fall wird dem Speicher keine Adresse zugeführt. Deshalb sollte in einem System mit nur einer MMU eine zusätzliche externe Logik vorhanden sein, die Zugriffe auf den nichtverwalteten Segmentnummernbereich erkennt, das $\overline{\text{SUP}}$-Signal aktiviert und einen Segmenttrap auslöst.
Das Flag URS (Upper Range Select) wird verwendet, um in der MMU den Segmentnummernbereich festzulegen, der durch sie verwaltet werden soll. Ist dieses Flag rückgesetzt, so werden die Segment-Deskriptor-Register der MMU zur Verwaltung der Segmente 0...63 verwendet. Ist es dagegen gesetzt, so bearbeitet die MMU die Segmente 64...127. Das URS-Flag korrespondiert so mit dem höchstwertigen Bit der Segmentnummern, die von der MMU übersetzt werden. Da dieses Flag durch Software gesteuert werden kann, ist es möglich, den Segmentnummernbereich durch ein Systemprogramm zu ändern.

7.3.2. Mehrere Übersetzungstabellen

Die MMU-Architektur unterstützt auch mehrere Übersetzungstabellen. Mehrere Tabellen können verwendet werden, um die Zeit zu minimieren, die bei einer Task-Umschaltung notwendig ist. Dabei wird jeder Task eine eigene Tabelle zugeordnet. Bei Multitask-Aufgaben mit mehreren Übersetzungstabellen muß die Task-Umschaltung nicht durch Umprogrammierung der Segment-Deskriptor-Register realisiert werden, sondern erfolgt einfach durch Setzen des Master-Enable-Flags der entsprechenden MMU.
Mehrere Übersetzungstabellen können auch verwendet werden, um den logischen Adreßbereich über 8 MByte hinaus zu erweitern. Dies kann durch Aufteilung in System-/Normalspeicher und/oder Programm-/Daten-/Stack-Speicher erfolgen. Das MST- und das NMS-Flag im Modus-Register können in Verbindung mit der N/$\overline{\text{S}}$-Leitung verwendet werden, um die entsprechende Tabelle auszuwählen. Zusätzliche externe Hardware, die die CPU-Statusleitungen überwacht und den N/$\overline{\text{S}}$-Eingang der MMU steuert, kann verwendet werden, um eine Trennung der Adreßräume vorzunehmen. Das MST-Flag (Multi Segment Table) zeigt an ob in der Konfiguration mehrere Tabellen verwendet werden. Wenn dieses Flag gesetzt ist, dann vergleicht die MMU den N/$\overline{\text{S}}$-Eingang mit dem NMS-Flag (Normal Mode Select) bevor sie physische Adressen an ihren Adreßausgängen erzeugt. Stimmen der Zustand des Eingangs und das Flag überein, so ist die MMU freigegeben und eine Adreßübersetzung wird durchgeführt. (Vorausgesetzt, das URS-Flag entspricht dem höchstwertigen Bit der Segmentnummer.) Gibt es dagegen bei gesetztem MST-Flag keine Übereinstimmung zwischen dem Zustand des N/$\overline{\text{S}}$-Einganges und dem NMS-Flag, so bleibt die MMU passiv. Die Flags MST und NMS können durch

Systemsoftware gesteuert werden.
Die einfachste Konfiguration mit mehreren Übersetzungstabellen hat eine Tabelle für Normalmodus- und eine für Systemmoduszugriffe. Hier wird das MST-Flag in allen MMUs gesetzt und die Eingänge N/$\overline{S}$ der MMUs werden mit dem Ausgang N/$\overline{S}$ der CPU verbunden. Das NMS-Flag ist bei den MMUs, die die Deskriptoren für die Systemsegmente enthalten, rückgesetzt. Dagegen wird es bei den MMUs, die die Normalsegmente verwalten, gesetzt. Arbeitet die CPU U8001 z.B. im Normalmodus, so ist die N/$\overline{S}$-Leitung im H-Zustand. Es kommt zur Übereinstimmung mit dem NMS-Flag der MMUs, die die Deskriptoren für die Normalsegmente enthalten. Folglich werden diese MMUs aktiviert, um die Adreßübersetzung durchzuführen. In einer solchen Konfiguration sind Segmente des Betriebssystems von Segmenten der Nutzer getrennt. Wenn der U8001 vom Normalmodus in den Sytemmodus wechselt, wird automatisch die entsprechende Übersetzungstabelle ausgewählt. Ein komplexeres Beispiel mit mehreren Übersetzungstabellen wird im Kapitel 8 "Applikationen" vorgestellt.

7.4. Initialisierung des DSCR

Das Deskriptor-Selektion-Counter-Register (DSCR) wird bei allen Kommandos verwendet, die auf die Felder des Deskriptors zugreifen (Lesen/Schreiben des Basis-, Limit- oder Attributfeldes oder Lesen/Schreiben des vollständigen Deskriptors einschließlich aller inkrementierenden Versionen dieser Kommandos).
Erfolgt der Zugriff nur auf das Limit- oder Attributfeld, so stellt die MMU zu Beginn das DSCR automatisch auf das entsprechende Feld und zum Abschluß automatisch auf Null. Die Kommandos, die auf das Basisadreßregister oder den gesamten Deskriptor zugreifen, modifizieren das DSCR zu Beginn nicht. Dies ist notwendig, damit ein solches Kommando nach Interrupts innerhalb von Blocktransfers an der richtigen Position im Deskriptor fortgesetzt werden kann. Bei korrekter Beendigung setzen auch diese Kommandos zum Abschluß das DSCR automatisch auf Null.
Wird ein Kommando nicht korrekt abgearbeitet, so kann der Fall eintreten, daß zum Abschluß im DSCR nicht der Wert Null steht. Dies sei an folgendem Beispiel erläutert:

Beispiel:

```
LDB RL0,#%06
SOUTB %01FC,RL0          !Initialisieren des SAR!
LDA RR2,BASIS            !Laden der Zieladresse!
LD R1,%08FC              !Laden des Kommandos!
LD R0,#1                 !Laden der Anzahl der Bytes!
SINIRB @RR2,@R1,R0
```

Das Problem liegt in der vorletzten Programmzeile. Das Kommando %08 dient zum Lesen der **zwei** Bytes des Basisadreßfeldes. Das Register R0 wurde aber nur mit dem Wert 1 geladen, so daß nur **ein** Datenbyte durch den SINIRB-Befehl transportiert wird. Damit wird der Befehl durch die CPU beendet, ohne daß das MMU-Kommando zum Abschluß gebracht wurde. Das DSCR zeigt auf das zweite Byte des Basisadreßfeldes.

Dieses Problem kann immer dann auftreten wenn mehr als ein Byte eines Deskriptors transportiert wird. Dies geschieht bei den folgenden Kommandos:

- Lesen/Schreiben des Basisadreßfeldes (%08)
- Lesen/Schreiben des Deskriptors (%0B)
- Lesen/Schreiben des Basisadreßfeldes und Inkrementieren des SAR (%0C)
- Lesen/Schreiben des Deskriptors und Inkrementieren des SAR (%0F)

Existiert vor der Ausführung eines dieser Kommandos der Zustand, daß das DSCR nicht auf Null steht, so wird auf ein falsches Feld zugegriffen. Es wird deshalb vorgeschlagen, vor der Nutzung eines solchen Kommandos das DSCR auf Null zu setzen.

8. Applikationen

8.0. Einführung

In diesem Kapitel werden zwei U8001-U8010-Konfigurationen beschrieben. Die eine enthält zwei MMUs und eine Übersetzungstabelle und die andere 16 MMUs mit acht Übersetzungstabellen. Diese Beispiele werden so ausführlich dargestellt, daß die grundsätzlichen Ideen der Konstruktion von Speicherverwaltungssystemen mit der U8010-MMU erörtert werden können. Blockschaltbilder zeigen die grundsätzliche Hardware-Konfiguration und kurze Programme dienen der Illustration der Software-Techniken bei Anwendung der MMU.

8.1. Beispielsystem 1: Zwei MMUs, 128 Deskriptoren

Das erste Beispielsystem, dargestellt im Bild 8.1, ist eine Konfiguration mit zwei MMUs.

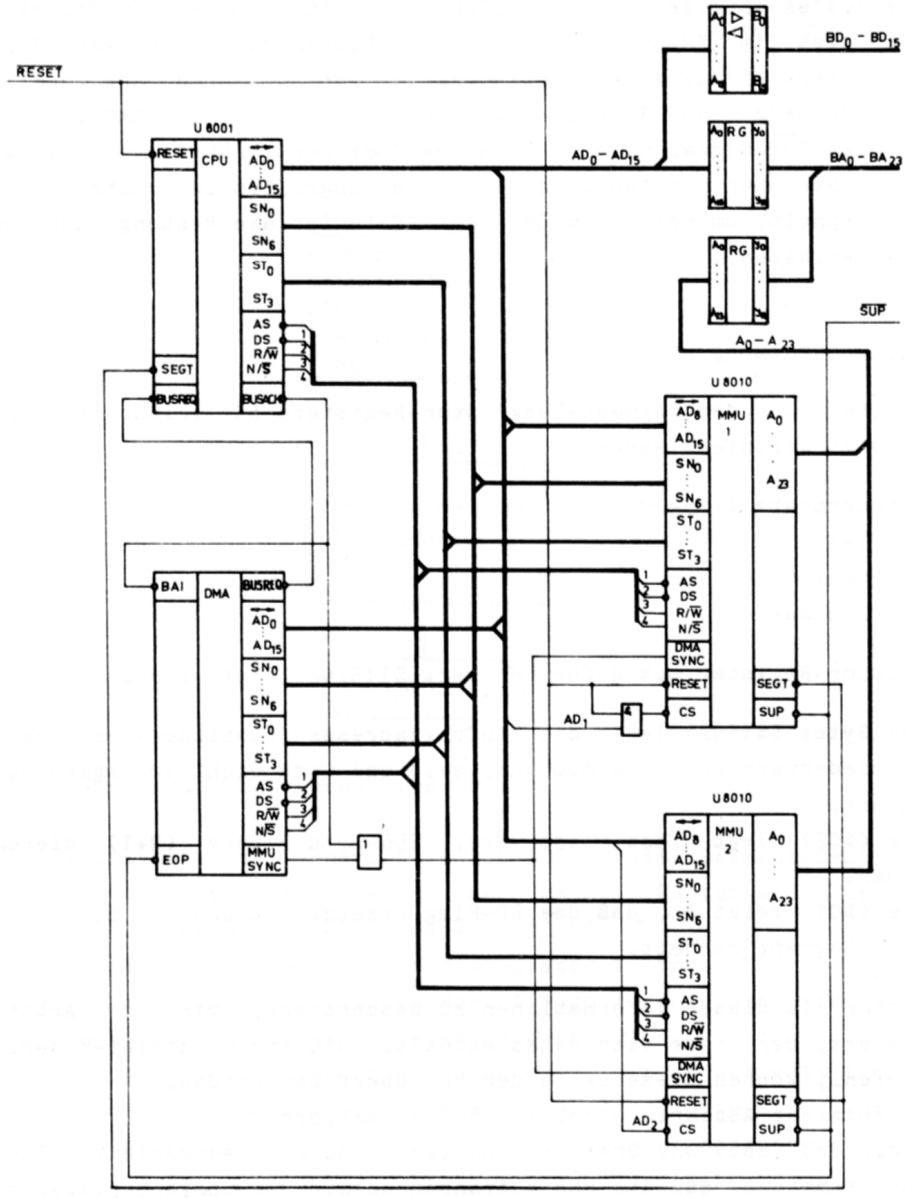


Bild 8.1: Beispielsystem 1

Sie werden mit MMU1 und MMU2 bezeichnet. In einem Kommandozyklus werden sie mit L-Pegel auf AD_1 bzw. AD_2 ausgewählt. (Die Annahme, daß das $\overline{CS}$-Signal durch L auf den entsprechenden AD-Leitungen aktiviert wird, wurde aus folgendem Grund getroffen: Nach Reset befindet sich der AD-Bus im Tri-State. Wenn ein aktives $\overline{CS}$-Signal aus dem H-Pegel auf der entsprechenden AD-Leitung abgeleitet würde, so würde bei Reset die Auswahl aller MMUs erfolgen.) Hat also das niederwertige Byte eines Kommandos den Wert %FC, so wird die MMU1 angesprochen. Dagegen wählt %FA die MMU2 aus, und mit %F8 werden beide MMUs selektiert. Bei der MMU2 erfolgt in der Reset-Phase keine Aktivierung des $\overline{CS}$-Einganges. Bei einem Reset ohne Aktivierung des $\overline{CS}$ wird das Modus-Register zurückgesetzt, also auch das MSEN-Flag auf Null gebracht, so daß keine Adressen durch die MMU ausgegeben werden können. Dagegen wird das $\overline{CS}$-Signal der MMU1 aus einer Verknüpfung von AD_1 und $\overline{RESET}$ gebildet. Dieses hat zur Folge, daß das MSEN-Flag bei der Systeminitialisierung gesetzt wird. Da das TRNS-Flag rückgesetzt wird, passieren die von der CPU erzeugten logischen Adressen unverändert die MMU in Richtung Speicher. So kann das Bootstrap-Programm (Anfangslader) auf den absoluten Speicherplätzen des physischen Speichers abgelegt sein. (Der Anfangslader kann sich damit in dem Speicher befinden, der der MMU zugeordnet ist.) Die Verwaltung des Busses erfolgt mittels $\overline{BUSREQ}$ und $\overline{BUSACK}$ in einer Prioritätenkette. Dataillierter ist dies im "Handbuch U8000: CPU U8001/U8002 Technische Beschreibung" ausgeführt. Es existiert ein direkter Verbindungsweg von CPU und DMA zum Systembus. Dieser Weg wird für E/A- und Refresh-Zugriffe verwendet, da die MMU in diesen Phasen nicht aktiv wird. Das $\overline{SUP}$-Signal wird in Richtung Speicher geführt, wo es verwendet werden kann, um den Speicher vor fehlerhaften Schreibzugriffen zu schützen. Zum anderen gelangt es zur DMA-Einheit, um bei einem DMA-Zugriffsfehler die Rettung der notwendigen Informationen zu veranlassen.

8.1.1. Laden eines Deskriptors

Es soll die Initialisierung des Segment-Deskriptor-Registers 65 erläutert werden. Ein Segment soll wie folgt definiert werden:

- Beginn auf Speicherplatz %115200
- Länge von 768 Bytes
- nur lesbares Segment
- Zugriffe im Normalmodus

Das Segment-Deskriptor-Register muß dafür den Wert %11520201 enthalten.

- Die ersten zwei Bytes (%1152) legen die Anfangsadresse (Basisadresse) des Segments fest. Deren niederwertiges Byte muß Null sein und wird nicht im Segment-Deskriptor-Register definiert.
- Das dritte Byte (%02) legt fest, das drei 256-Byte-Blöcke (N+1) diesem Segment zugeordnet wurden.
- Das vierte Byte (%01) zeigt an, daß das RD-Flag gesetzt ist und es sich folglich um ein nur lesbares Segment handelt.

Um den MMU-Deskriptor mit diesen Informationen zu beschreiben, wird im Arbeitsspeicher ein Bereich angelegt, der diese vier Bytes enthält. Mit einem Spezial-E/A-Blockbefehl, z.B. dem SOTIRB-Befehl, können diese Werte der MMU übergeben werden.
Dieser Befehl hat folgende Assembler-Syntax: SOTIRB dst,src,r
Dabei werden Zieloperand (dst) und Quelloperand (src) indirekt adressiert. Das Zielregister enthält das Kommando, daß die MMU ausführen soll. Der Speicherplatz, auf den das Quellregister zeigt, enthält das erste Byte, daß der MMU übergeben werden soll und im Register (r) wird die Anzahl der zu übertragenden Bytes definiert.

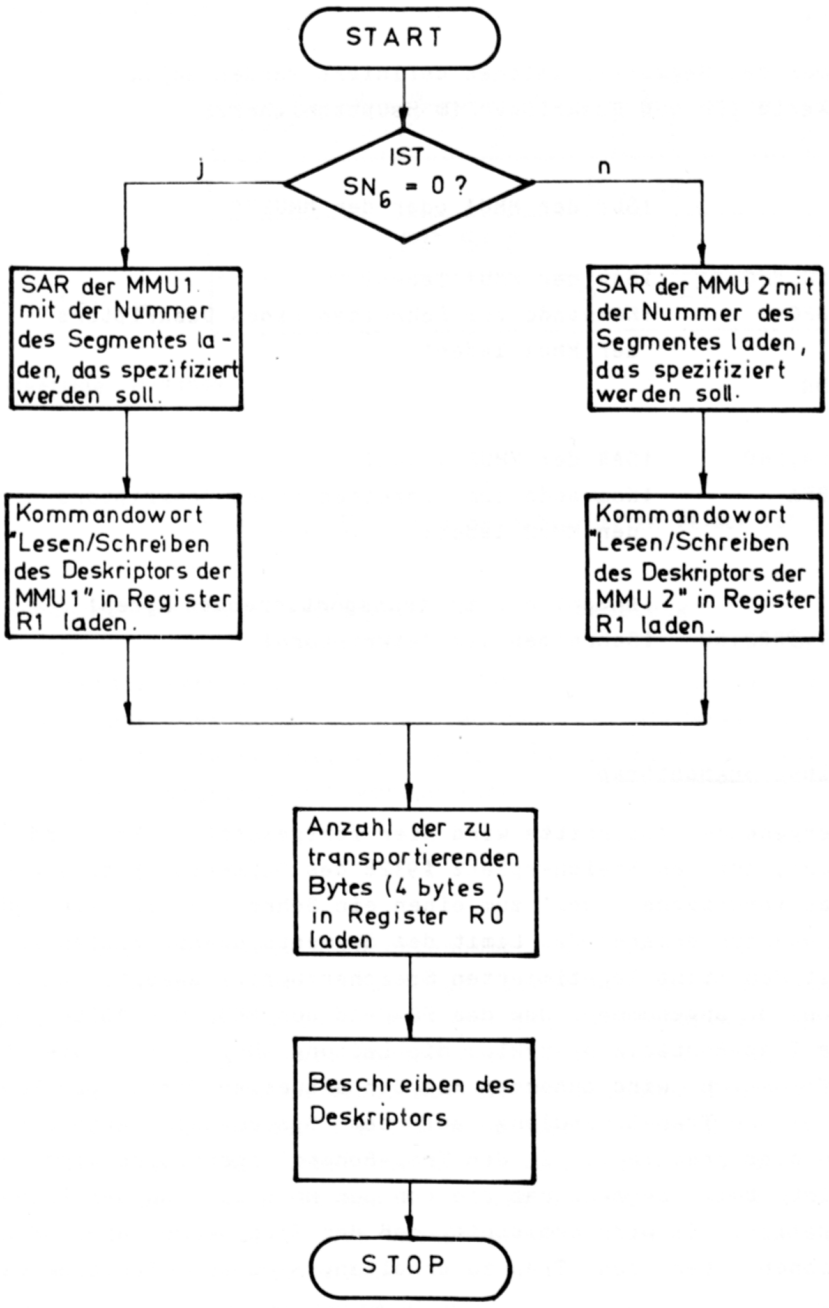


Bild 8.2: Programmablaufplan "Laden eines Deskriptors"

Der Befehlskode zum Laden des Deskriptors ist %0B. Das Segment-Deskriptor-Register 65 ist das Segment-Deskriptor-Register 1 der MMU2. Es ist also der Auswahlkode %FA zu verwenden. Somit lautet das MMU-Kommandowort %0BFA.
Um festzulegen, daß das Segment-Deskriptor-Register 1 der MMU2 beschrieben wird, ist deren Segment-Adreß-Register mit dem Wert %01 zu laden. Der Befehlskode zum Laden des SAR ist %01. Somit lautet dieses MMU-Kommandowort %01FA.
In dem Beispielprogramm ist die Nummer des zu behandelnden Segments (in diesem Fall 65) als Argument im Register R0 abgelegt. Als weitere Argumente sind diesem Programm die in das Segment-Deskriptor-Register einzutragenden Werte zu übergeben. Dies erfolgt über RR2, welches auf den Speicherbereich zeigt, in dem diese Werte abgelegt sind.
Im Bild 8.2 ist der Programmablaufplan für dieses Beispiel gezeigt, und nachfolgend ist

das Programm zur Initialisierung des Segment-Deskriptors angegeben. Es werden die Register R0...R3 verwendet.

- R0 enthält die Nummer des Segments, welches definiert werden soll.
- RR2 zeigt auf die Werte für den Deskriptor im Hauptspeicher.

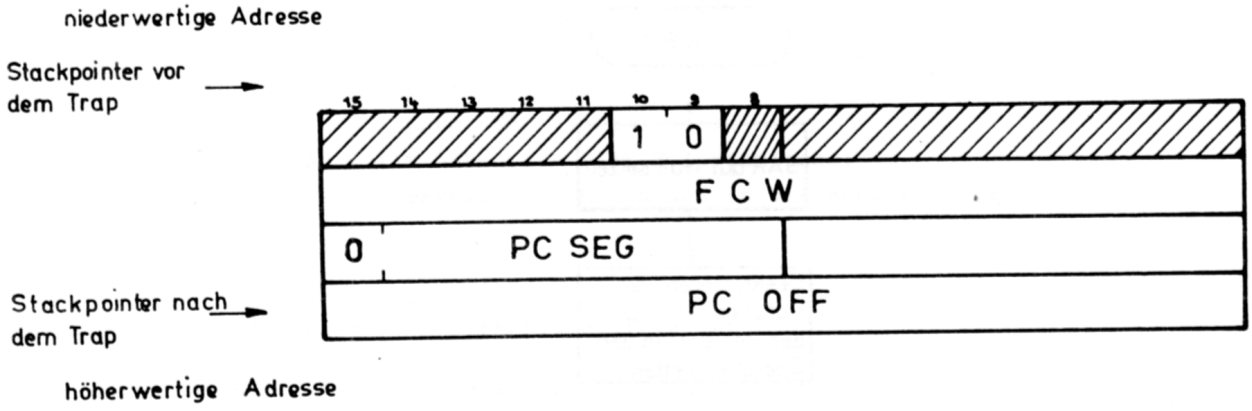
```
            BIT R0,#6           !SDR der MMU1 oder der MMU2?!
            JR NZ,MMU2
            SOUTB %01FC,RH0     !SAR der MMU1 laden!
            LD R1,#%0BFC        !Kommando zum Schreiben eines Deskriptors
                                 der MMU1 laden!
            JR LADE_SDR
MMU2:
            SOUTB %01FA,RH0     !SAR der MMU2 laden!
            LD R1,#%0BFA        !Kommando zum Schreiben eines Deskriptors
                                 der MMU2 laden!
LADE_SDR:
            LD R0,#4            !Anzahl der zu transportierenden Bytes!
            SOTIRB @R1,@RR2,R0  !Schreiben des Deskriptors!
```

8.1.2. Verarbeitung eines Segmenttrap

Das Beispiel des vorhergehenden Abschnitts wird hier fortgesetzt. Es wird angenommen, daß der Nutzer versucht, auf den Speicherplatz %9328 des Segments 65 zu schreiben. Dies würde einen Segmenttrap verursachen, weil zum einen ein Schreiben in ein nur lesbares Segment versucht wird und zum anderen das Limit des Segments überschritten wird.
Am Ende des Befehls mit dem nicht legetimierten Speicherzugriff akzeptiert die CPU den Trap. Im Beispiel wird angenommen, daß das ID-Feld der MMU2 mit "010" geladen wurde. Damit wird während des Trap-Bestätigungszyklus die Leitung AD_{10} durch die MMU2 auf H gebracht. Diese Information wird zusammen mit den aktuellen Werten von FCW, PC-Offset und PC-Segmentnummer bei der Trap-Behandlung auf dem Systemstack abgelegt. Bild 8.3 zeigt, in welcher Form dies geschieht. In der Trap-Behandlungsroutine wird das Kennwort ausgewertet. Dies ist beim Segmenttrap die von den MMUs während der Trap-Bestätigung ausgesendete Identifikation. Es wird ermittelt, daß der Trap durch die MMU2 ausgelöst wurde. Um Informationen über den Trap zu erhalten, sind also Register dieser MMU zu lesen und auszuwerten.

- Das Violation-Typ-Register (VTR) enthält %05, womit angezeigt wird, daß sowohl eine Überschreitung der Segmentlänge als auch ein Schreiben in ein nur lesbares Segment aufgetreten ist.
- Im Bus-Zyklus-Status-Register (BCSR) ist der Wert %28 zu finden. Dieser dokumentiert in seinem höherwertigen Nibble, daß eine Schreiboperation im Normalmodus abgearbeitet wurde und in seinem niederwertigen Nibble, daß es sich um einen Datenzugriff handelte.
- Im Violation-Segment-Nummer-Register (VSN) steht %01. Dadurch erhält man die Information, daß die Violation im Segment 1 der MMU2, also im Segment 65 erfolgte.
- Der Wert %93 im Violation-Offset-Register (VOFF) zeigt das höherwertige Byte des Offset der logischen Adresse, bei der der Fehler aufgetreten ist.

Das Betriebssystem könnte dem Nutzer eine Fehlermeldung übergeben mit dem Inhalt, daß auf das Segment 65 ein nicht erlaubter Zugriff versucht wurde (Schreiben in ein nur lesbares Segment und Überschreitung der Segmentlänge). Eine Information über den nächsten



Bemerkung: Der Inhalt der schraffierten Felder ist nicht definiert

Bild 8.3: Anordnung der Programmstatuswerte im Systemstack

auszuführenden Befehl kann man erhalten, indem der PC ausgewertet wird, der bei der Trap-Bestätigung auf den Stack gerettet wurde. Das niederwertige Byte der Adresse, bei der die Violation aufgetreten ist, geht verloren. Im nächsten Beispiel wird beschrieben, wie auch dieses Byte aufgezeichnet werden kann.

8.1.3. Context-Swap

In Fortsetzung dieses Beispiels wird angenommen, daß zwei Nutzer durch das Rechnersystem bedient werden. Beide können alle 128 Segmente des U8001 nutzen. Da die 23-Bit-logische-Adresse durch die MMU in eine 24-Bit-physische-Adresse übersetzt wird, ist im physischen Speicher Platz für zwei separate 2^{23}-Byte-Adreßräume. Da immer nur ein Nutzer die Verfügung über die CPU haben kann, muß ein sogenanntes Context-Swap durchgeführt werden. Dabei übernimmt ein Nutzer die CPU nach festgelegter Zeit vom anderen. In den zwei MMUs können zu gegebener Zeit immer nur die Deskriptoren eines Nutzers abgelegt sein. Unter einem Context-Swap versteht man hier, das Retten der Deskriptoren des einen Nutzers von den MMUs in den Speicher und das nachfolgende Laden der Deskriptoren des anderen Nutzers aus dem Speicher in die MMUs.
Der im Bild 8.4 dargestellte Programmablaufplan und das nachfolgend aufgeführte Programm zeigen, wie ein solches Context-Swap realisiert werden könnte. Das äußerst leistungsfähige MMU-Kommando "Lesen/Schreiben des Deskriptors und Erhöhen des SAR" ist Kernstück in diesem Programm. Es wird angenommen, daß die Deskriptoren des Nutzers 1 beginnend auf der Adresse DESK_1 und die des Nutzers 2 beginnend auf der Adresse DESK_2 im Speicher abgelegt sind. Weiter wird vorausgesetzt, daß das Segment 127 immer für das Betriebssystem reserviert ist. Dort sind auch die Daten und das Programm der Context-Swap-Routine untergebracht.

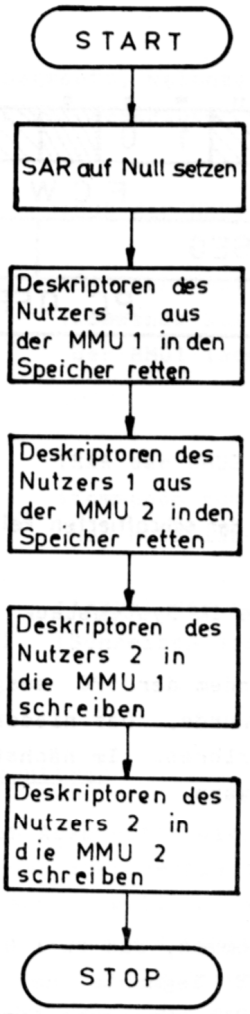


Bild 8.4: Programmablaufplan zum Context-Swap

```
CLR R0                  !Beginne mit Deskriptor 0!
SOUTB %01F8,RH0         !SAR in MMU1 und MMU2 initialisieren!
LDA RR2,DESK_1          !Anfangsadresse des Datenfeldes DESK_1 laden!
LD R1,#%0FFC            !Kommando "Lesen/Schreiben des Deskriptors
                         der MMU1" laden!
LD R0,#256              !Anzahl der Bytes laden!

SINIRB @RR2,@R1,R0      !Deskriptoren aus der MMU1 lesen und in dem
                         Datenfeld DESK_1 ablegen!
LD R1,#%0FFB            !Kommando "Lesen/Schreiben des Deskriptors der
                         MMU2" laden!
LD R0,#252              !Anzahl der Bytes laden!
SINIRB @RR2,@R1,R0      !Deskriptoren aus der MMU2 lesen und
                         fortsetzend im Datenfeld DESK_1 ablegen!
LDA RR2,DESK_2          !Anfangsadresse des Datenfeldes DESK_2 laden!
LD R1,#%0FFC            !Kommando "Lesen/Schreiben des Deskriptors
                         der MMU1" laden!
LD R0,#256              !Anzahl der Bytes laden!
```

```
SOTIRB @R1,@RR2,R0    !Deskriptoren aus dem Datenfeld DESK_2 holen
                       und in die MMU1 eintragen!
LD R1,#%0FFB          !Kommando "Lesen/Schreiben des Deskriptors
                       der MMU2" laden!
LD R0,#252            !Anzahl der Bytes laden!
SOTIRB @R1,@RR2,R0    !im Datenfeld DESK_2 fortsetzend;
                       Deskriptoren holen und in die MMU2 eintragen!
```

8.2. Beispielsystem 2: 16 MMUs, 8 Übersetzungstabellen

Bild 8.5 zeigt das Blockschaltbild des zweiten zu diskutierenden Systems. Diese Konfiguration enthält 16 MMUs. Als hauptsächliche Veränderungen gegenüber dem vorherigen Beispiel sind zu registrieren:

- zusätzliche Hardware zur Generierung der $\overline{CS}$-Signale
- zusätzliche Latches zur Aufzeichnung des niederwertigen Teils des Offsets bei einer Violation

Die erste Veränderung ist erforderlich, da maximal sieben MMUs ohne zusätzliche Hardware selektiert werden können. (AD_0 kann nicht an eine achte MMU vergeben werden, weil Bytetransfers im Spezial-E/A-Raum auf geraden Adressen zu erfolgen haben. Bei einer Byte-Leseoperation mit ungerader Adresse im Spezial-E/A-Raum würden die Daten von $AD_0 \ldots AD_7$ übernommen. Diese Leitungen werden aber von der MMU gar nicht versorgt, so daß dieser Lesezugriff ein undefiniertes Ergebnis hätte.)
Die Latches dienen dazu, den niederwertigen Teil des Offsets aufzuzeichnen, wenn eine MMU einen Segmenttrap auslöst. Ein Latch enthält dann das niederwertige Byte der Adresse, bei der die Violation auftrat und im anderen Latch ist das niederwertige Byte der Adresse des Befehls zu finden, der die Violation verursachte. In der Trap-Behandlungsroutine können diese Latches gelesen werden. Damit kann die Ursache der Violation konkret ermittelt werden und evtl. eine Korrektur und Wiederaufnahme des Befehls erfolgen.

Tabellenumschaltung im System mit 16 MMUs

Dieses System besitzt eine Speicherverwaltung bei der zwei MMUs ständig dem Betriebssystem zugeordnet sind. Von den restlichen MMUs sind jeweils zwei einer Anwender-Task zugeeignet. So sind die Übersetzungstabellen von sieben Anwender-Tasks in 14 MMUs abgelegt und die Umschaltung auf eine Task erfordert nur die Freigabe der zugeordneten MMUs. Dieser Freigabeprozeß kann durch Steuerung der Master-Enable-Flags (MSEN) in den Modus-Registern (MR) der entsprechenden MMUs erfolgen. Das Programm führt Task-Swap durch selektives Freigeben von MMUs durch. Dies geschieht in folgender Weise:

- Alle Anwender-MMUs werden gesperrt. Damit werden auch die MMUs der letzten Task gesperrt. (Die beiden dem Betriebssystem zugeordneten MMUs sind ständig freigegeben.)
- Das der aktuellen Task zugeordnete MMU-Paar wird freigegeben. Dabei enthält das Register R1 die Nummer der aktuell zu bearbeitenden Task (%n wenn Task n behandelt werden soll).

Folgende Besonderheiten sollten beachtet werden: Alle Anwender-MMUs, die den unteren Segmentnummernbereich verwalten haben die gleiche Identifikation (%2). Ebenso stimmen die Anwender-MMUs des höheren Segmentnummernbereichs in ihrer Identifikation (%3) überein. Dies muß in der Trap-Behandlungsroutine beachtet werden.

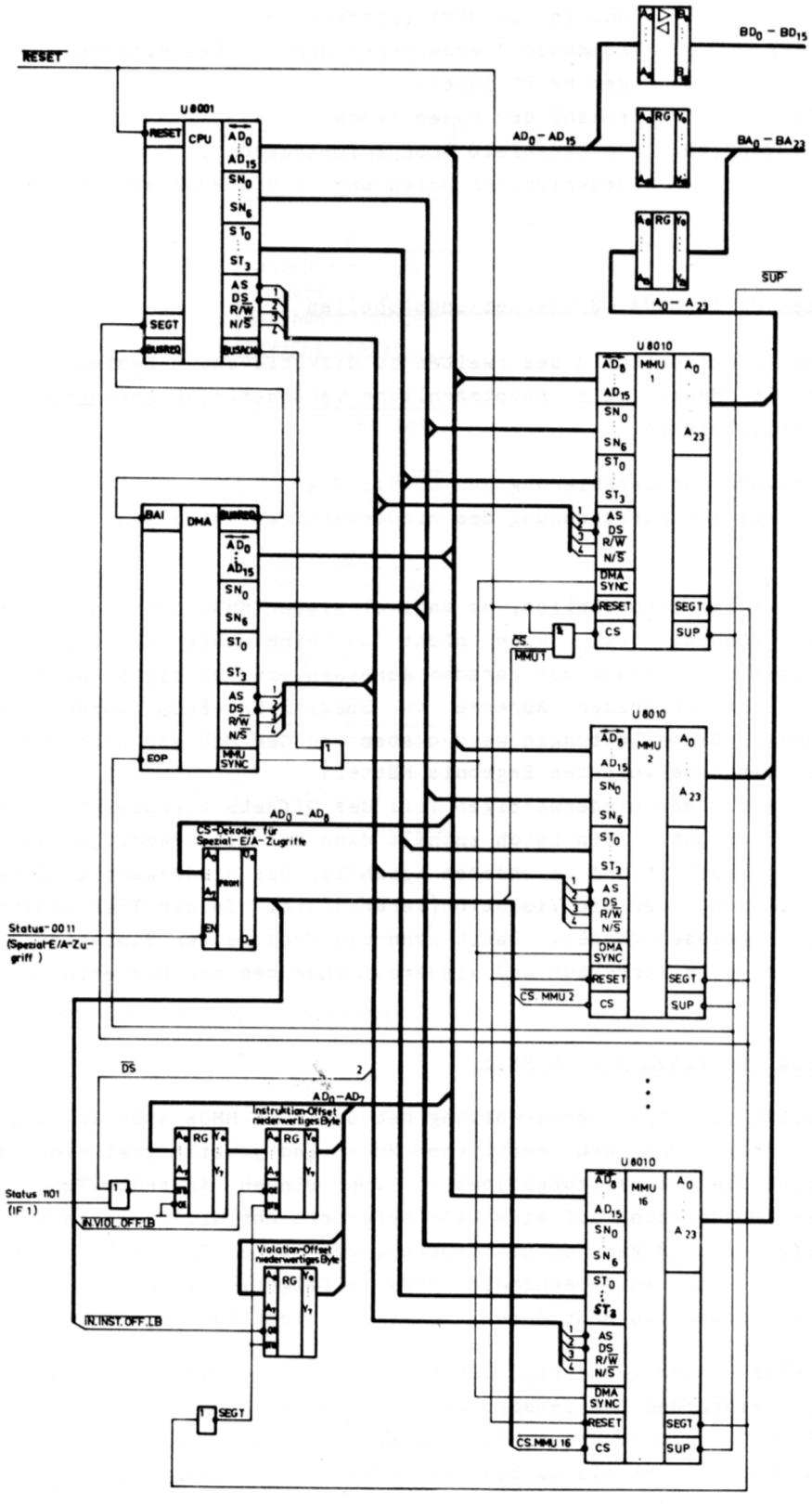


Bild 8.5: Beispielsystem 2

Die Tabelle 8.1 zeigt mit welchem Kode die MMUs selektiert werden und welche Identifikation festgelegt ist. Im Bild 8.6 ist der Programmablaufplan für dieses Beispiel dargestellt und nachfolgend ist das Programm aufgeführt.

Task	MMUi	SN-Bereich	Auswahlkode	Identifikation
System	MMU1	URS=0	%02	%0
	MMU2	URS=1	%04	%1
Task0	MMU3	URS=0	%08	%2
	MMU4	URS=1	%10	%3
Task1	MMU5	URS=0	%18	%2
	MMU6	URS=1	%20	%3
Task2	MMU7	URS=0	%28	%2
	MMU8	URS=1	%30	%3
		.		
		.		
		.		
Task6	MMU15	URS=0	%68	%2
	MMU16	URS=1	%70	%3

Außerdem wird angenommen, daß Auswahlkode
%F8 alle Anwender-MMUs (MMU3...MMU15) selektiert.

Tabelle 8.1: Auswahlkodes und Identifikation der MMUs im System 2

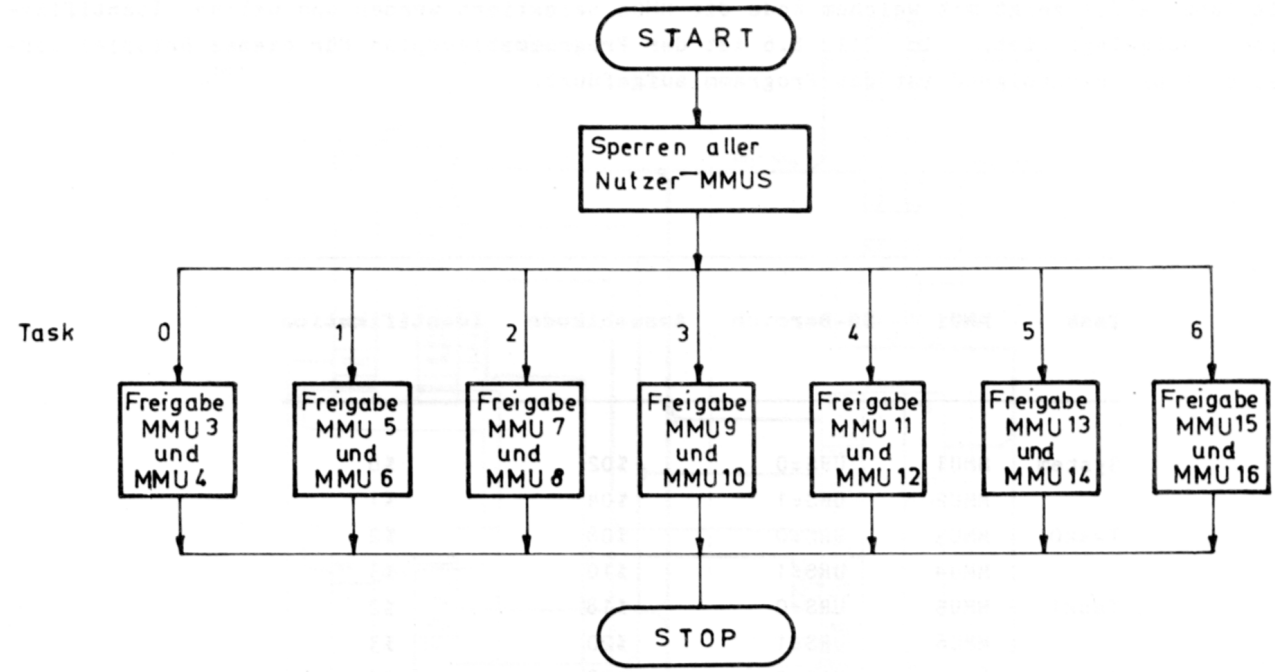


Bild 8.6: Programmablaufplan zur Tabellenumschaltung

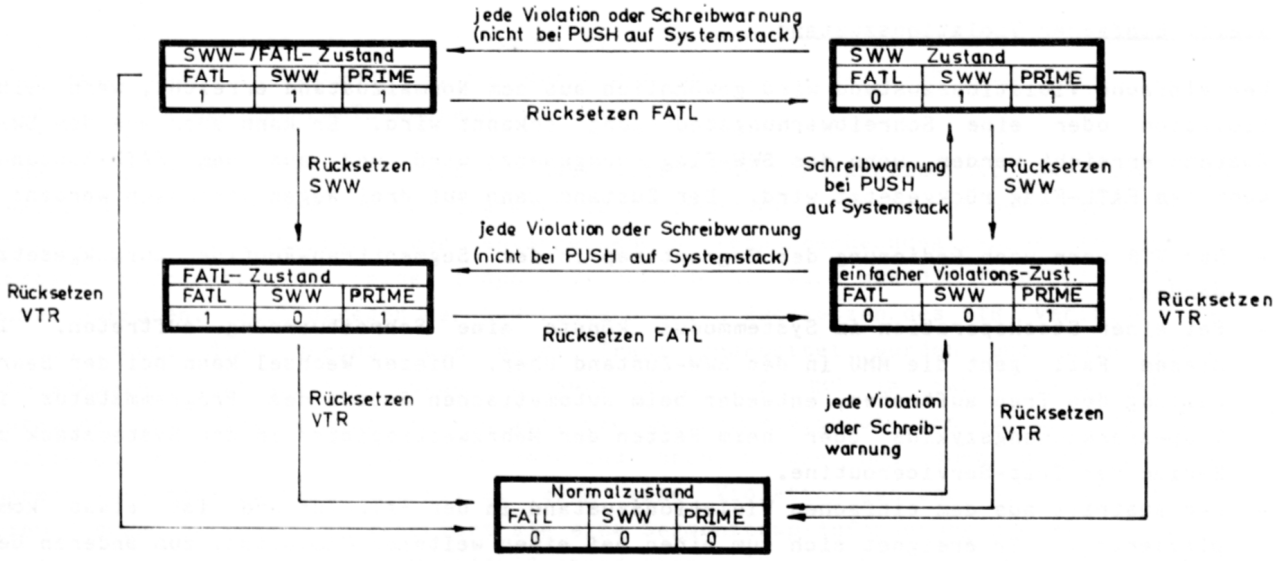
```
            CLR R0                  !Kode für MR laden!
            SOUT %00FA,R0           !alle Anwender-MMUs durch Rücksetzen
                                     des MR sperren!
            SLA R1,#1               !Multiplizieren der Task-Nr. mit 2!
            LD R1,KOM_TAB(R1)       !Kommandowort laden
                                     (Auswahlkode der MMU; Kommandokode %0)!
            LDA RR2,MR_INIT         !RR2 zeigt auf Initialisierungswert für MR
                                     der MMU des unteren SN-Bereiches!
            SOUTIB @R1,@RR2,R0      !MMU des unteren SN-Bereiches initialisieren;
                                     RR2 auf MR_INIT+1 richten!
            INC R1,#8               !Auswahlkode für MMU des oberen
                                     SN-Bereiches einstellen!
            SOUTIB @R1,@RR2,R0      !MMU des oberen SN-Bereiches initialisieren!
ENDE:

MR_INIT:
            BYTES(%DA,%FB)

KOM_TAB:
            WORDS(%8,%18,%28,%38,%48,%58,%68)
```

A. Interne MMU-Zustände

Wenn die MMU Zugriffsverletzungen und Schreibwarnungsbedingungen ermittelt, hängt ihre Reaktion von den vorausgegangenen Ereignissen ab. Die Vorgeschichte zum Verletzungs- oder Warnungsereignis wird am internen Zustand der MMU erkennbar. Es gibt fünf interne Zustände. Die MMU führt maximal einen Zustandswechsel bei Abarbeitung eines Befehls aus. DMA- und Refresh-Operationen sowie CPU-Operationen, die von der MMU ignoriert werden (z.B. Standard-E/A-Operationen) haben keinen Einfluß auf die internen Zustände und werden auch nicht von diesen internen Zuständen beeinflußt. Bild A.1 stellt alle möglichen Zustände und deren Übergänge dar. Die internen Zustände der MMU werden durch die Kombination der gesetzten Violationsflags im VTR bestimmt. Zum Zweck der Definition dieser internen Zustände werden die Flags in drei Gruppen eingeteilt: Die erste Gruppe enthält die Flags CPUIV, SLV, SYSV, PWW, RDV und EXCV. Sie werden primäre Flags genannt. Die zweite Gruppe enthält das SWW-Flag, und die dritte Gruppe wird vom FATL-Flag gebildet.



Bemerkung: PRIME = PWW v EXCV v CPU IV v SLV v SYSV v RDV
(v steht für logisches ODER)

Bild A.1: Interne MMU-Zustände und deren Übergänge

A.1. Normalzustand

- Der erste interne Zustand ist der Normalzustand, in dem noch kein Flag im VTR gesetzt ist. Dieser Zustand wird immer erreicht, wenn die MMU durch das $\overline{\text{RESET}}$-Signal zurückgesetzt wird oder wenn das VTR durch MMU-Steuerkommandos (Befehlskode %10 oder

%11) zurückgesetzt wird. Wenn sich die MMU in diesem Zustand befindet, wurde seit dem letzten Rücksetzen keine Violation oder Schreibwarnung erkannt.

A.2. Fehlerzustände

Die restlichen vier Zustände resultieren aus dem Erkennen einer Violation oder Schreibwarnung. Diese vier Zustände werden dadurch unterschieden, ob die Flags SWW und FATL gesetzt sind oder nicht. Diese beiden Flags können nur gesetzt werden, wenn mindestens eines der primären VTR-Flags gesetzt ist.

- Der zweite interne Zustand ist der einfache Violationszustand, in dem ein oder auch mehrere primäre VTR-Flags gesetzt sind, aber sowohl das FATL-Flag als auch das SWW-Flag rückgesetzt sind.
- Der dritte Zustand ist der SWW-Zustand, in dem ein oder auch mehrere VTR-Flags und das SWW-Flag gesetzt sind, das FATL-Flag jedoch rückgesetzt ist.
- Der vierte Zustand ist der FATL-Zustand, in dem ein oder auch mehrere VTR-Flags und das FATL-Flag gesetzt sind, das SWW-Flag jedoch rückgesetzt ist.
- Der fünfte Zustand ist der SWW-/FATL-Zustand, in welchem das FATL- und das SWW-Flag, sowie ein oder mehrere VTR-Flags gesetzt sind.

A.2.1. Einfacher Violationszustand

Der einfache Violationszustand wird gewöhnlich aus dem Normalzustand erreicht, wenn eine Violation oder eine Schreibwarnungsbedingung erkannt wird. Er kann auch aus dem SWW-Zustand erreicht werden, wenn das SWW-Flag rückgesetzt wird und aus dem FATL-Zustand, wenn das FATL-Flag rückgesetzt wird. Der Zustand kann auf drei Wegen verlassen werden:

- Das VTR kann nach Bedienung des Segmenttrap in der Segmenttrap-Routine zurückgesetzt werden.
- Bei einer Stackoperation im Systemmodus könnte eine Schreibwarnung auftreten. In diesem Fall geht die MMU in den SWW-Zustand über. Dieser Wechsel kann bei der Bearbeitung des Trap auftreten, entweder beim automatischen Retten des Programmstatus im Trap-Anerkennungszyklus oder beim Retten der Mehrzweckregister in den Systemstack zu Beginn der Trap-Serviceroutine.
- Der Austritt aus dem einfachen Violationszustand in den FATL-Zustand ist etwas komplizierter. Er ereignet sich zum einen bei einer weiteren Violation, zum anderen bei einer Schreibwarnung im Normalmodus oder einer Schreibwarnung im Systemmodus, bei der nicht auf den Stack geschrieben wird (Status nicht %9).

A.2.2. FATL-Zustand

Der FATL-Zustand kann sowohl aus dem einfachen Violationszustand, wie im vorangegangenen Abschnitt beschrieben, als auch aus dem SWW-/FATL-Zustand, durch Rücksetzen des SWW-Flag, erreicht werden. Der FATL-Zustand kann entweder durch Rücksetzen des FATL-Flag verlassen werden, wodurch die MMU in den einfachen Violationszustand gebracht wird oder durch Rücksetzen des VTR, wodurch die MMU in den Normalzustand zurückkehrt.
Solange sich die MMU im FATL-Zustand befindet, generiert sie bei weiteren Violationen Suppress-Signale, aber keine Segmenttrap-Anforderungen. Der FATL-Zustand bedeutet gewöhnlich, daß die MMU einen Fehler in der Trap-Serviceroutine des Betriebssystems

erkannt hat. Entweder erfolgte bei der Routine selbst ein nicht korrekter Zugriff auf ein Segment, oder es wird zum Normalzustand zurückgekehrt, ohne daß das VTR zurückgesetzt worden ist. Über die Ursache dieser fatalen Violation wird keine Information in der MMU aufgezeichnet. Bei der zweiten oder weiteren Violationen werden keine Violationsflags gesetzt und keine Adreßinformationen gespeichert. Das ist durch die begrenzte Anzahl von Statusregistern in der MMU bedingt, welche immer nur auf die erste erkannte Violation bzw. Schreibwarnung hinweisen.
Wenn das FATL-Flag einmal gesetzt worden ist, ist es höchst unwahrscheinlich, daß das Betriebssystem einen solchen Fehler korrigieren kann. In dieser Situation ist möglichst der Prozessorzustand zu retten und anzuhalten.

A.2.3. SWW-Zustand

Der SWW-Zustand zeigt an, daß bei gesetzten VTR-Flags ein Schreiben auf den Systemstack eine Schreibwarnungsbedingung erzeugt hat. Im allgemeinen bedeutet dies, daß der Systemstack während der Segmenttrap-Anerkennungssequenz bis in die letzten 256 Bytes seines zugewiesenen Speicherraums gewachsen ist. Dieser Zustand kann auf drei Wegen verlassen werden:

- Das VTR wird zurückgesetzt, wodurch die MMU in den Normalzustand zurückkehrt.
- Das SWW-Flag wird zurückgesetzt, wodurch die MMU in den einfachen Violationszustand zurückkehrt.
- Die MMU erkennt eine weitere Violation, eine Schreibwarnung im Normalmodus oder eine Schreibwarnung im Systemmodus, bei der nicht auf den Stack geschrieben wird. Dadurch erreicht sie den SWW-/FATL-Zustand.

Gewöhnlich erfolgt der Austritt aus dem SWW-Zustand durch Rücksetzen des SWW-Flag. Das SWW-Flag wird bei einer zweiten Segmenttrap-Anforderung gesetzt, und der Programmstatus für beide Traps wird während der entsprechenden Trap-Anerkennungszyklen in den Systemstack gebracht. Um beide Traps zu behandeln, wird man die Trap-Serviceroutine zweimal aufrufen. Soll dagegen der SWW-Zustand durch Rücksetzen des VTR verlassen werden, so setzt dies voraus, daß beide Traps bearbeitet wurden und auch der Systemstackpointer aktualisiert wurde. Solange sich die MMU im SWW-Zustand befindet, verursachen nachfolgende Schreibwarnungen, die aus Schreiboperationen auf den Systemstack resultieren, keine Trap-Anforderungen mehr und bringen die MMU auch nicht in einen anderen internen Zustand. Ereignet sich eine Violation während sich die MMU in diesem Zustand befindet, so generiert sie eine Segmenttrap-Anforderung und ein Suppress-Signal und wechselt in den SWW-/FATL-Zustand über. (Eine Schreibwarnung, bei der nicht auf den Systemstack geschrieben wird, generiert einen Trap, aber kein Suppress und bewirkt ebenfalls den Übergang zum SWW-/FATL-Zustand.)

A.2.4. SWW-/FATL-Zustand

Gewönlich bedeuten der SWW-/FATL-Zustand und ebenso der FATL-Zustand, daß die MMU einen Fehler in der Trap-Serviceroutine des Betriebssystems festgestellt hat. Über die fatale Violation wird keine Information in der MMU gespeichert. Wenn dieser Zustand einmal erreicht wurde, ist es daher unwahrscheinlich, daß das Betriebssystem sich regenerieren kann. Befindet sich die MMU in diesem Zustand, so generieren alle nachfolgenden Violationen Suppress-Signale, aber keine Trap-Anforderungen. Schreibwarnungen verursachen weder Trap-Anforderungen noch Suppress-Signale. Dieser Zusstand kann auf drei Wegen verlassen werden:

- Die MMU erreicht den FATL-Zustand durch Rücksetzen des SWW-Flag.
- Sie erreicht den SWW-Zustand durch Rücksetzen des FATL-Flag.
- Es erfolgt ein Übergang in den Normalzustand durch Rücksetzen des VTR.

A.3. "Unechter" Befehlsholezyklus

Bei den meisten U8001-Befehlen erfolgen in den letzten zwei oder drei Taktzyklen interne Operationen, welche keinen Speicherzugriff erfordern. Der nächste Befehl wird bereits geholt, während diese Befehle ihre internen Operationen ausführen (Prinzip der Befehlsvorausschau). Wenn die MMU in der Speicherzugriffsphase eines dieser Befehle eine Segmenttrap-Anforderung generiert, wird trotzdem, noch vor der Trap-Anerkennung durch die CPU, das erste Wort des nächsten Befehls geholt (da der Trap erst nach der Beendigung des Befehls anerkannt wird). Die MMU muß auf diesen "unechten" Befehlsholezyklus reagieren. Im Einzelnen heißt das, wenn sich in einem "unechten" Befehlsholezyklus eine Violation ereignet, wird ein Suppress-Signal generiert (die Segmenttrap-Anforderung von der vorhergehenden Violation oder Schreibwarnung bleibt aktiv). Verursacht dieser "unechte" Befehlsholezyklus keine Violation, wird auch kein Suppress-Signal generiert. Auf jeden Fall bricht die CPU diesen Befehl zu Beginn des Trap-Anerkennungszyklus ab und führt nach Rückkehr aus der Trap-Serviceroutine einen erneuten Befehlsholezugriff auf diesen Befehl aus. "Unechte" Befehlsholezyklen, die Violationen verursachen, bewirken keine Übergänge zwischen internen MMU-Zuständen und auch nicht das Setzen von Flags im VTR.

B. Zeitabläufe

B.0. Einführung

Die MMU U8010 übersetzt Adressen und überprüft Zugriffsrechte, indem sie Folgen von Basiszyklen durchläuft. Diese korrespondieren mit der Struktur der Buszugriffe der CPU U8001.
Die Diagramme im folgenden Abschnitt zeigen die prinzipiellen zeitlichen Verläufe der MMU-Signale bei ihren grundsätzlichen Operationen

- Lesen/Schreiben im Speicher
- Kommandoübergabe
- Segmenttrap-Bestätigung

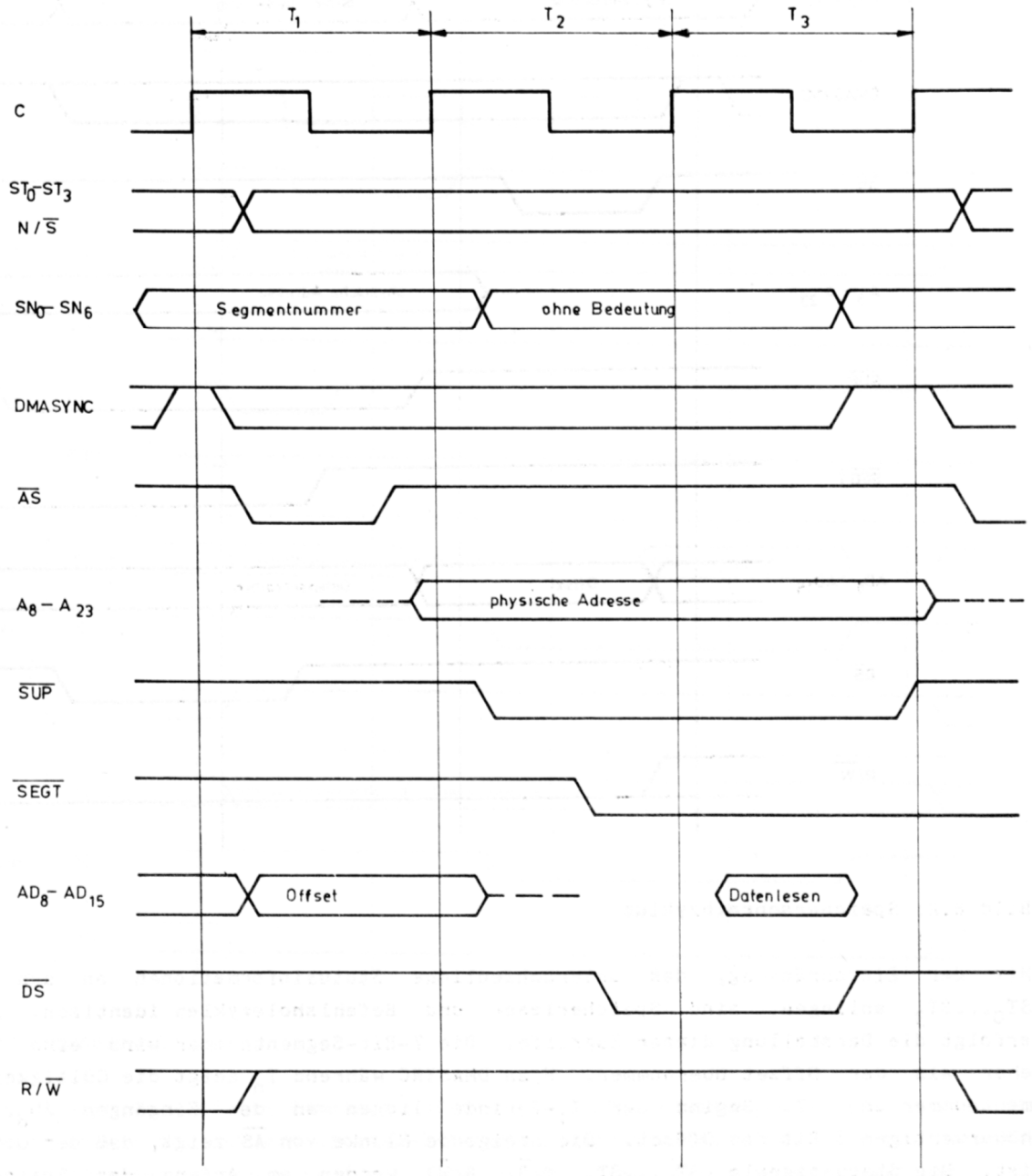


Bild B.1: Speicherlesezyklus

B.1. Zeitlicher Ablauf der Speicherlese- und Speicherschreibzugriffe

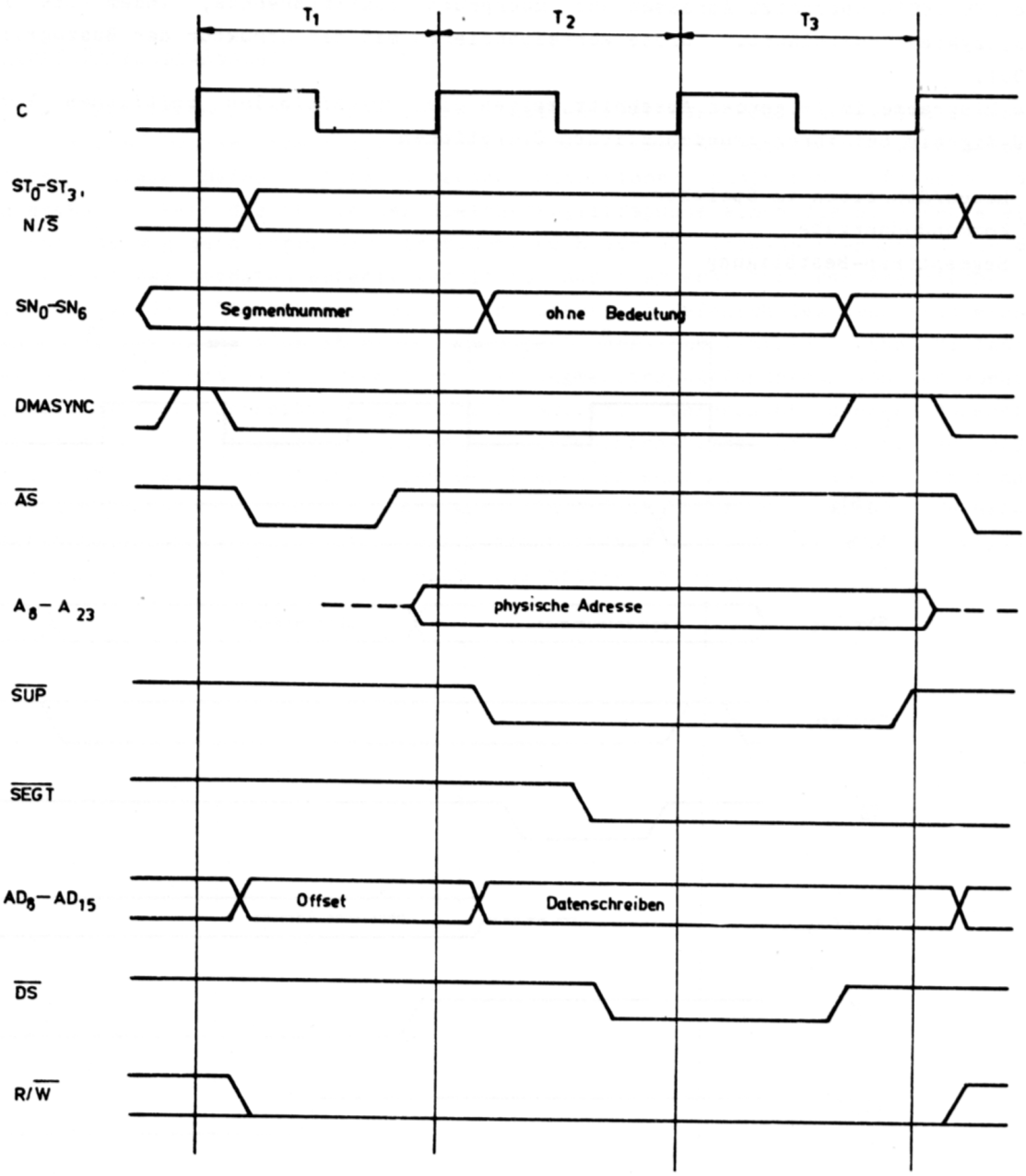


Bild B.2: Speicherschreibzyklus

Mit der Einschränkung, daß unterschiedliche Statusinformationen an den Eingängen $ST_0 \ldots ST_3$ anliegen, sind Speicherlese- und Befehlsholezyklen identisch. Im Bild B.1 erfolgt die Darstellung dieser Zugriffe. Die 7-Bit-Segmentnummer wird eine Taktperiode eher als der Offset übernommen. H an DMASYNC während T_n zeigt die Gültigkeit der Segmentnummer an. Zu Beginn der T_1-Periode liegen an den Eingängen $AD_8 \ldots AD_{15}$ die höherwertigen 8 Bit des Offset. Die steigende Flanke von $\overline{AS}$ zeigt, daß der Offset gültig ist. Die Statussignale ($ST_0 \ldots ST_3$, $N/\overline{S}$, $R/\overline{W}$) werden am Anfang des Speicherzugriffs gültig und bleiben während des gesamten Zugriffs stabil. Die höherwertigen 16 Bit der Adresse (die physische Speicheradresse) bleiben bis zum Ende von T_3 gültig. Die Signale

$\overline{SEGT}$ und $\overline{SUP}$ werden bei festgestellten Zugriffsverletzungen im Zylus T_2 aktiviert. Der Segmenttrap-Ausgang bleibt bis zum Empfang der Segmenttrap-Bestätigung auf L-Pegel. Das Suppress-Signal wird dagegen nur während des laufenden Maschinenzyklus ausgegeben und im T_3-Zyklus inaktiv. Es wird aber bei jedem weiteren CPU-Speicherzugriff wiederholt aktiv, bis der laufende Befehl beendet ist.
Speicherschreibzyklen sind identisch mit Speicherlesezyklen, außer daß die R/$\overline{W}$-Leitung im Lesezyklus H-Pegel und im Schreibzyklus L-Pegel führt. Der Ablauf von Speicherschreibzyklen ist im Bild B.2 dargestellt.

B.2. Zeitlicher Ablauf der Kommandoübergabe

Das Bild B.3 zeigt die Übergabe eines Kommandos an die MMU.

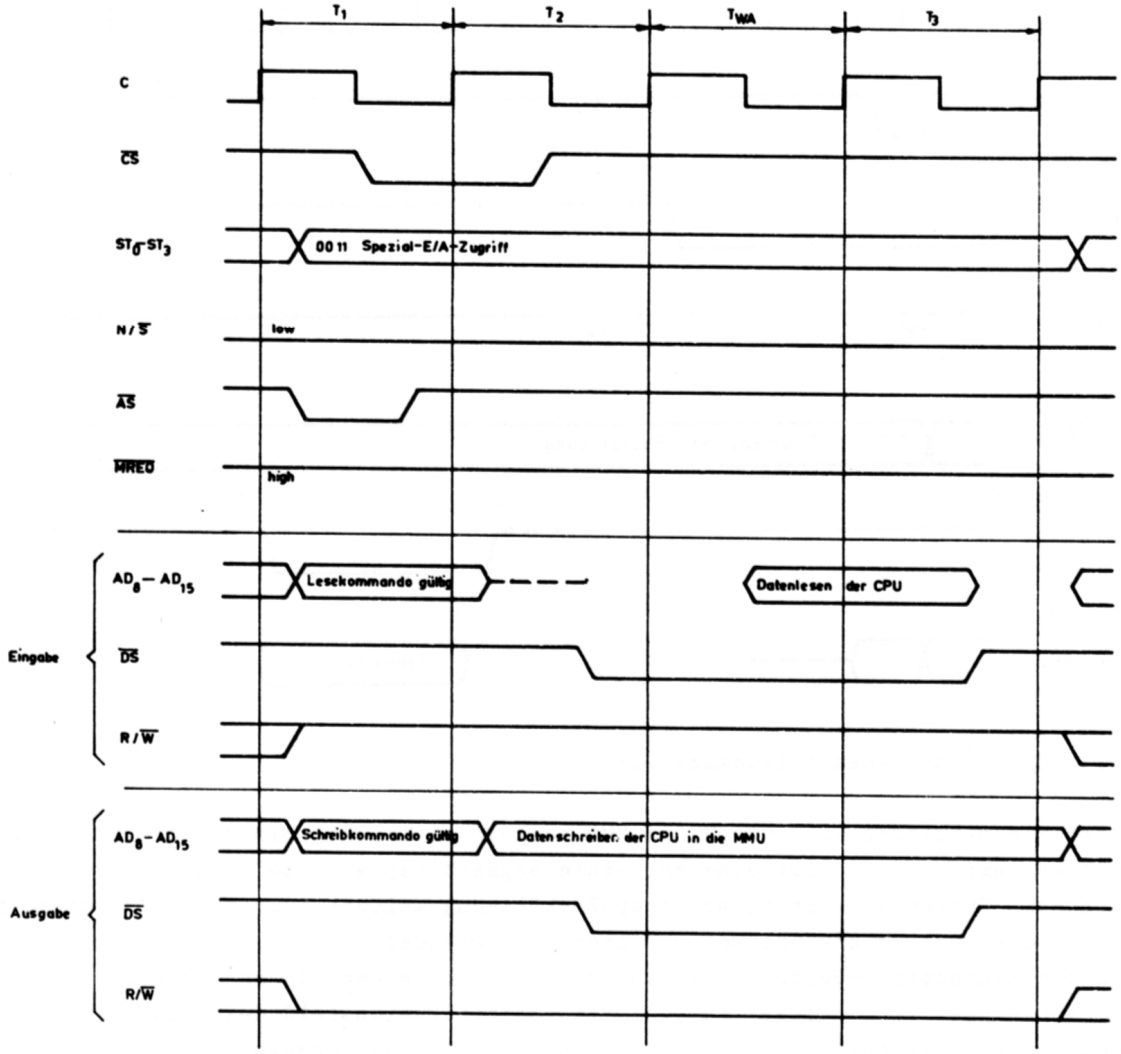


Bild B.3: Kommandozyklus

Während des T_1-Zyklus wird das Kommando auf die Adreß-/Datenleitungen gelegt. Die Statusleitungen zeigen an, daß ein Spezial-E/A-Befehl abgearbeitet wird. Die zur Übernahme des Kommandos vorgesehene MMU wird durch Aktivierung ihres $\overline{CS}$-Eingangs selektiert. Daten, die in ein Register der MMU eingeschrieben werden sollen, müssen am Ende des T_2-Zyklus auf den AD-Leitungen gültig sein. Daten, die aus der MMU gelesen werden sollen,

werden am Ende des T_{WA}-Zyklus auf den AD-Leitungen ausgegeben.

B.3. Zeitlicher Ablauf der Segmenttrap-Anerkennung

Die MMU erzeugt immer dann einen Segmenttrap, wenn sie eine Zugriffsverletzung feststellt oder wenn ein Schreiben in den letzten Block eines Stacksegmentes (Segment mit gesetztem DIRW-Flag) erfolgt. Bild B.4 zeigt den Ablauf bei einer Segmenttrap-Bestätigung.

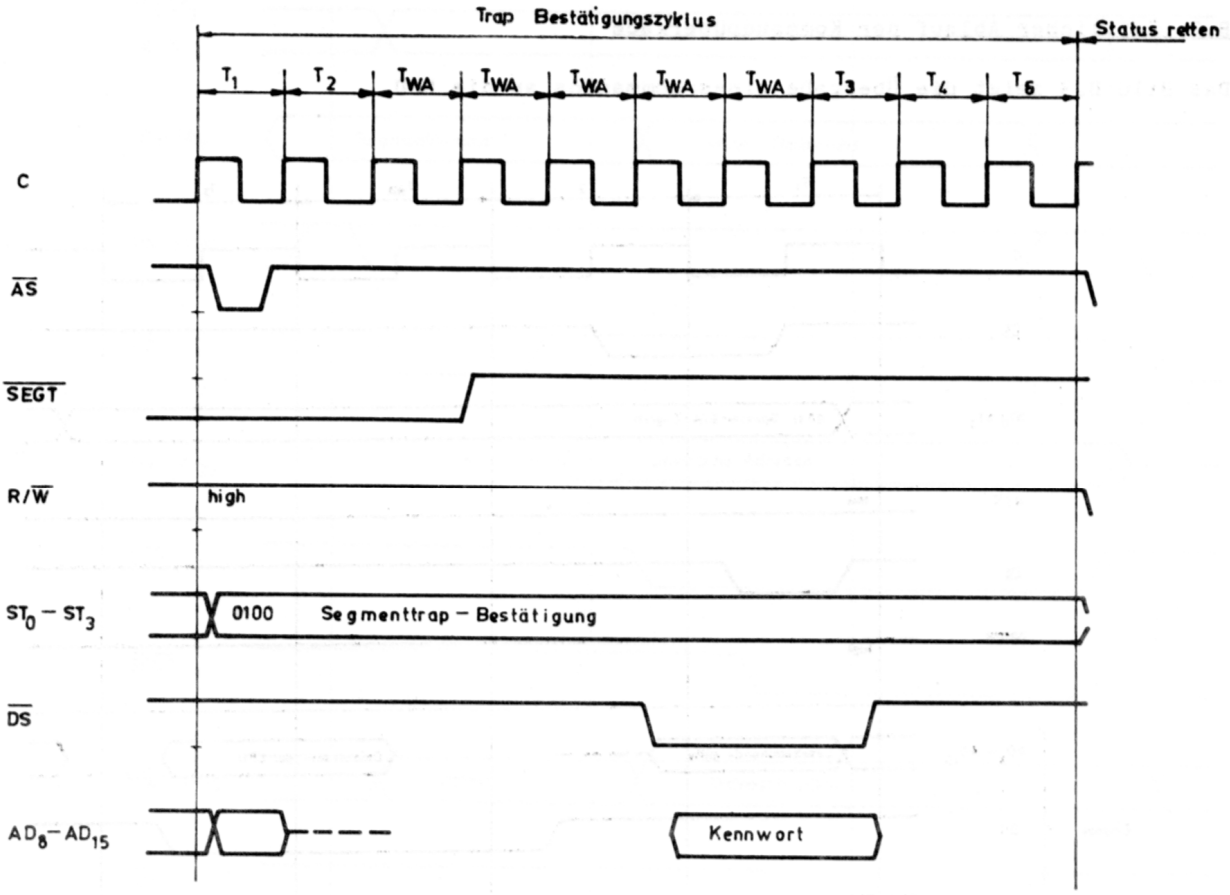


Bild B.4: Segmenttrap-Bestätigungszyklus

Die Segmenttrap-Leitung wird durch die MMU solange aktiviert, bis sie die Bestätigung des Segmenttrap empfängt. Gibt eine MMU einen Segmenttrap aus, so erwartet sie auch dessen Bestätigung. Treten vor der Segmenttrap-Bestätigung weitere Zugriffsverletzungen auf, so werden diese ebenfalls erkannt und entsprechend behandelt.
Während der Segmenttrap-Bestätigung gibt die MMU auf einer ihrer AD-Leitungen H-Pegel aus. Festgelegt wird diese Leitung im Identifikationsfeld des Modus-Registers (MR). Nach dem Segmenttrap-Bestätigungszyklus bringt die CPU das Kennwort (Identifikation der MMUs) und weitere Statusinformationen auf den Systemstack. Dann beginnt die Routine zur Behandlung des Segmenttrap. Im technischen Handbuch "CPU U8001/U8002 - Technische Beschreibung" sind detaillierte Informationen zum Retten des Status enthalten.

B.4. Definition der dynamischen Kennwerte

Diese Angaben haben informativen Charakter.
Verbindliche Angaben sind dem Fachbereichstandard TGL 43 020 zu entnehmen.

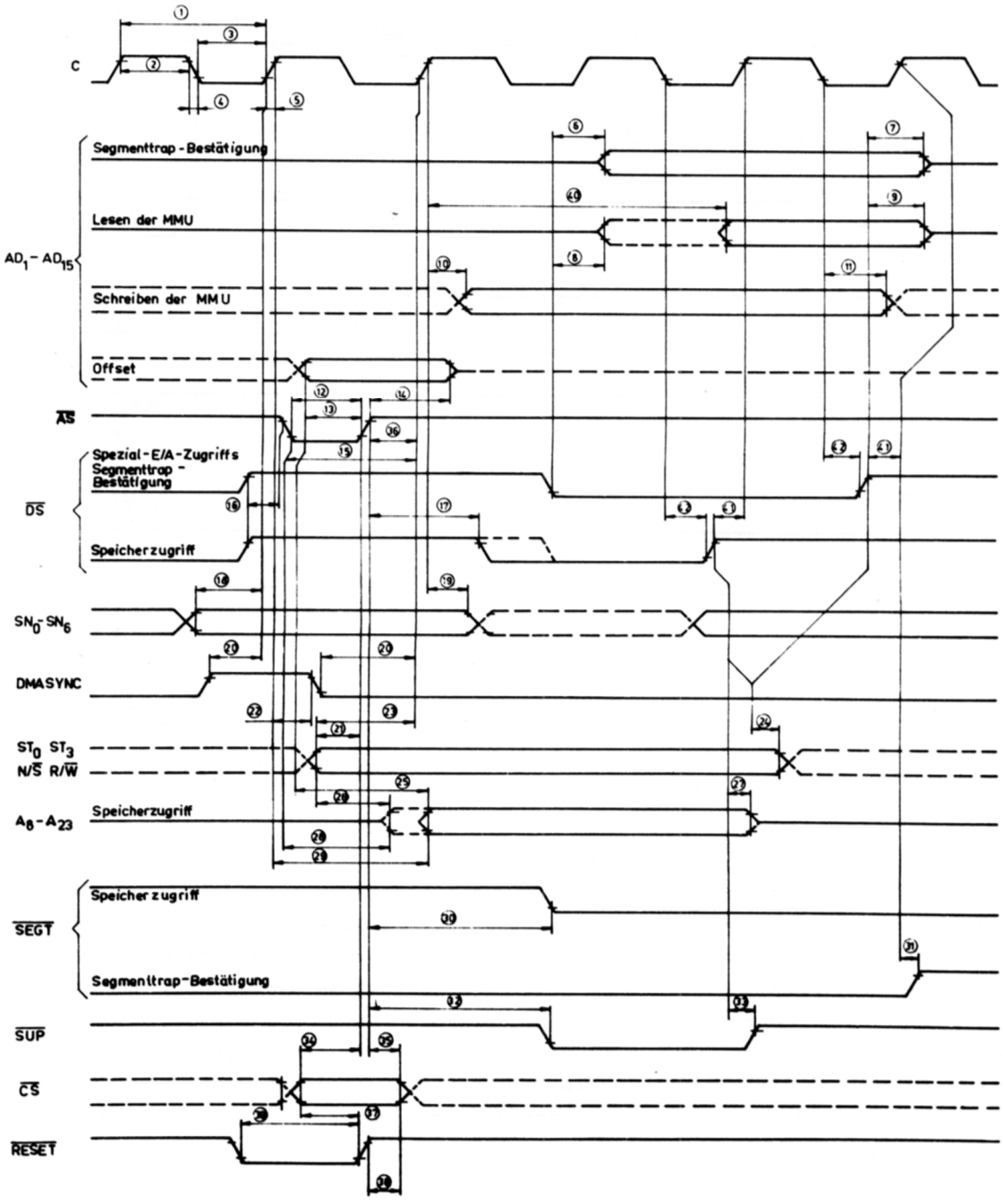


Bild B.5: Zeitdiagramm

Nr.	Symbol	Parameter	min [ns]	max [ns]
1	TcC	Zykluszeit des Taktes	250	
2	TwCh	Breite der High-Phase des Taktes	105	
3	TwCl	Breite der Low-Phase des Taktes	105	
4	TfC	Flankenanstiegszeit des Taktes (HL)		20
5	TfC	Flankenabfallzeit des Taktes (LH)		20
6	TdDSA(RDv)	Verzögerungszeit von $\overline{DS}$ (HL) bis Gültigkeit der Daten im Trap-Bestätigungszyklus		100
7	TdDSA(RDf)	Verzögerungszeit von $\overline{DS}$ (LH) bis Floaten von AD8-AD15 im Trap-Bestätigungszyklus		75
8	TdDSR(RDv)	Verzögerungszeit von $\overline{DS}$ (HL) bis Treiben der AD-Ausgänge im Lesezyklus		100
9	TdDSR(RDf)	Verzögerungszeit von $\overline{DS}$ (LH) bis Floaten von AD8-AD15 im Lesezyklus		75
10	TdC(WDv)	Verzögerungszeit von C (LH) bis Gültigkeit der Daten im Schreibzyklus		125
11	ThC(WDn)	Haltezeit der Daten nach C (HL) im Schreibzyklus	30	
12	TwAS	Breite von $\overline{AS}$	60	
13	TsOFF(AS)	Setzzeit des Offset vor $\overline{AS}$ (LH)	45	
14	ThAS(OFFn)	Haltezeit des Offset nach $\overline{AS}$ (LH)	60	
15	TdAS(C)	Verzögerungszeit von $\overline{AS}$ (HL) bis C (LH)	110	
16	TdDS(AS)	Verzögerungszeit von $\overline{DS}$ (LH) bis $\overline{AS}$ (HL)	50	
17	TdAS(DS)	Verzögerungszeit von $\overline{AS}$ (LH) bis $\overline{DS}$ (HL)	50	
18	TsSN(C)	Setzzeit von SN0-SN6 vor C (LH)	100	
19	ThC(SNn)	Haltezeit von SN0-SN6 nach C (LH)	0	
20	TdDMAS(C)	Verzögerungszeit von DMASYNC (LH) bis C (LH)	120	
21	TdSTNR(AS)	Verzögerungszeit von Gültigkeit des Status (ST0-ST3, N/$\overline{S}$, R/$\overline{W}$) bis $\overline{AS}$ (LH)	50	
22	TdC(DMA)	Verzögerungszeit von C (LH) bis DMASYNC (HL)	20	
23	TdST(C)	Verzögerungszeit von Gültigkeit des Status (ST0-ST3) bis C (LH)	100	
24	TdDS(STn)	Verzögerungszeit von $\overline{DS}$ (LH) bis Ungültigkeit des Status	0	
25	TdOFF(Av)	Verzögerungszeit von Gültigkeit des Offset bis Gültigkeit der Adreßausgabe		175
26	TdST(Ad)	Verzögerungszeit von Gültigkeit des Status bis Treiben der Adreßausgänge		155
27	TdDS(Af)	Verzögerungszeit von $\overline{DS}$ (LH) bis Floaten der Adreßausgänge		160
28	TdAS(Ad)	Verzögerungszeit von $\overline{AS}$ (HL) bis Treiben der Adreßausgänge		145
29	TdC(Av)	Verzögerungszeit von C (LH) bis Gültigkeit der Adreßausgabe		255
30	TdAS(SEGT)	Verzögerungszeit von $\overline{AS}$ (LH) bis $\overline{SEGT}$ (HL)		160
31	TdC(SEGT)	Verzögerungszeit von C (LH) bis $\overline{SEGT}$ (LH)		300
32	TdAS(SUP)	Verzögerungszeit von $\overline{AS}$ (LH) bis $\overline{SUP}$ (HL)		150
33	TdDS(SUP)	Verzögerungszeit von $\overline{DS}$ (LH) bis $\overline{SUP}$ (LH)		155
34	TsCS(AS)	Setzzeit von $\overline{CS}$ vor $\overline{AS}$ (LH)	10	
35	ThAS(CSn)	Haltezeit von $\overline{CS}$ nach $\overline{AS}$ (LH)	60	
36	TdAS(C)	Verzögerungszeit von $\overline{AS}$ (LH) bis C (LH)	0	
37	TsCS(RST)	Setzzeit von $\overline{CS}$ vor $\overline{RESET}$ (LH)	150	
38	ThRST(CS)	Haltezeit von $\overline{CS}$ nach $\overline{RESET}$ (LH)	0	
39	TwRST	Breite von $\overline{RESET}$ (Low)	2TcC	
40	TdC(RDv)	Verzögerungszeit von C (LH) bis Gültigkeit der Daten im Lesezyklus		460
41	TdDS(C)	Verzögerungszeit von $\overline{DS}$ (LH) bis C (LH)	30	
42	TdC(DS)	Verzögerungszeit von C (HL) bis $\overline{DS}$ (LH)	0	

Bemerkung: (LH) symbolisiert die Flanke vom Low- zum High-Zustand (steigende Flanke)
(HL) symbolisiert die Flanke vom High- zum Low-Zustand (fallende Flanke)

Dynamische Kennwerte bei ϑ_a = 0°C bis 70°C, U_{CC} = 5V ± 0,25V, Testlast C_L = 100pF

Tabelle B.1: Dynamische Kennwerte

C. Anschlußbeschreibung

$A_8 \ldots A_{23}$ Adreßbus (Ausgänge, H-aktiv, Tri-State)
Diese Leitungen führen die 16 höherwerigen Bits der physischen Speicheradresse.

$AD_8 \ldots AD_{15}$ Adreß-/Datenbus (Ein-/Ausgänge, H-aktiv, Tri-State)
Diese multiplexten Adreß-/Datenleitungen werden für die Übertragung der Kommandos, deren Daten und der zu übersetzenden logischen Adressen verwendet.

$\overline{AS}$ Adreß-Strobe (Eingang, L-aktiv)
Die steigende Flanke von $\overline{AS}$ zeigt die Gültigkeit von $AD_0 \ldots AD_{15}$, ST_0 - ST_3, $\overline{CS}$, R/$\overline{W}$ und N/$\overline{S}$ an.

C Systemtakt (Clock) (Eingang)
Dieser 5V-Einphasentakt dient als Zeitbasis für die CPU und die MMU.

$\overline{CS}$ Chip-Auswahl (Chip Select) (Eingang, L-aktiv)
Über diesen Anschluß wird die MMU zur Übernahme eines Kommandos ausgewählt.

DMASYNC DMA-Segmentnummer-Synchronisations-Strobe (Eingang, H-aktiv)
L-Pegel auf dieser Leitung signalisiert, daß ein DMA-Zugriff erfolgt. H-Pegel zeigt die Gültigkeit der Segmentnummer an. Bei CPU-Zyklen muß diese Leitung stets H sei.

$\overline{DS}$ Daten-Strobe (Eingang, L-aktiv)
Diese Leitung sorgt für die zeitliche Steuerung des Datentransfers zwischen CPU und MMU.

N/$\overline{S}$ Normal-/Systemmodus (Eingang, L bei Systemmodus)
Über diese Leitung wird der MMU mitgeteilt, ob die CPU oder DMA-Einheit im Normal- oder im Systemmodus arbeitet. Der Eingang kann auch verwendet werden, um die MMU einem bestimmten Adreßraum zuzuordnen.

$\overline{RESET}$ Rücksetzen (Reset) (Eingang, L-aktiv)
Bei L-Pegel auf dieser Leitung wird die MMU zurückgesetzt.

R/$\overline{W}$ Lesen/Schreiben (Read/Write) (Eingang, L für Schreiben)
Diese Leitung zeigt an, ob von der CPU oder DMA-Einheit eine Lese- oder Schreiboperation ausgeführt wird.

$\overline{SEGT}$ Segmenttrap (Ausgang, L-aktiv, Open-Drain)
Wenn die MMU eine Zugriffsverletzung ermittelt oder eine Schreibwarnung erzeugt, wird der CPU über diese Leitung eine Unterbrechungsanforderung gesendet.

$SN_0 \ldots SN_6$ Segmentnummer (Eingänge, H-aktiv)

Die Leitungen $SN_0 \ldots SN_5$ werden verwendet, um eines der 64 Segmente in der MMU zu adressieren. Die Leitung SN_6 dient zur selektiven Freigabe der MMU.

$ST_0 \ldots ST_3$ Status (Eingänge, H-aktiv)

Auf diesen Leitungen wird der MMU der Status der CPU U8001 übermittel. Die Kodierung der Statusinformation zeigt Tabelle C.1.

$\overline{SUP}$ Suppress (Ausgang, H-aktiv, Open-Drain)

Dieses Signal wird während eines Buszyklus aktiviert, wenn eine Violation festgestellt wurde. Eine Schreibwarnung hat keinen Einfluß auf dieses Signal.

RSV Reserviert

Dieser Anschluß darf nicht beschaltet werden.

Art des Zugriffs	ST_3 ... ST_0 hex	binär	Definition
Interne Operation	%0	0000	
Refresh	%1	0001	
E/A-Zugriff	%2	0010	Standard-E/A
	%3	0011	Spezial-E/A
Interrupt-/Trap-Bestätigung	%4	0100	Segmenttrap
	%5	0101	Nichtmaskierbarer Interrupt
	%6	0110	Nichtvektorisierter Interrupt
	%7	0111	Vektorisierter Interrupt
Speicherzugriff	%8	1000	Datenadreßraum
	%9	1001	Stackadreßraum
	%A	1010	Datenadreßraum (EPU)
	%B	1011	Stackadreßraum (EPU)
	%C	1100	Programmadreßraum, n-tes Wort (IFn) eines Befehls
	%D	1101	Programmadreßraum, erstes Wort (IF1) eines Befehls
EPU-Transfer	%E	1110	
Reserviert	%F	1111	

Tabelle C.1: Statuskodierung

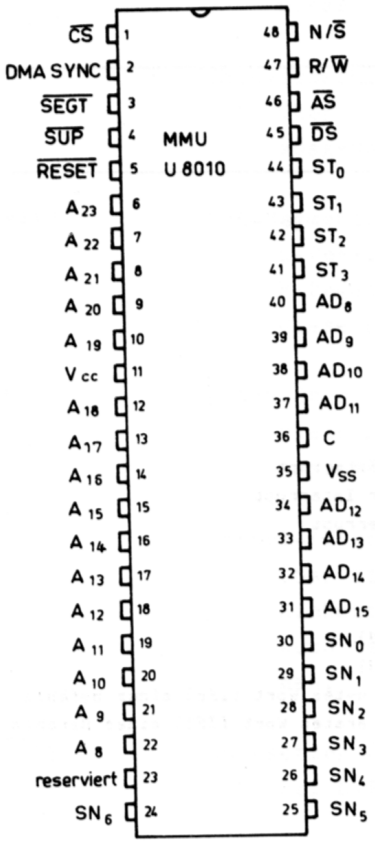


Bild C.1: Anschlußbelegung

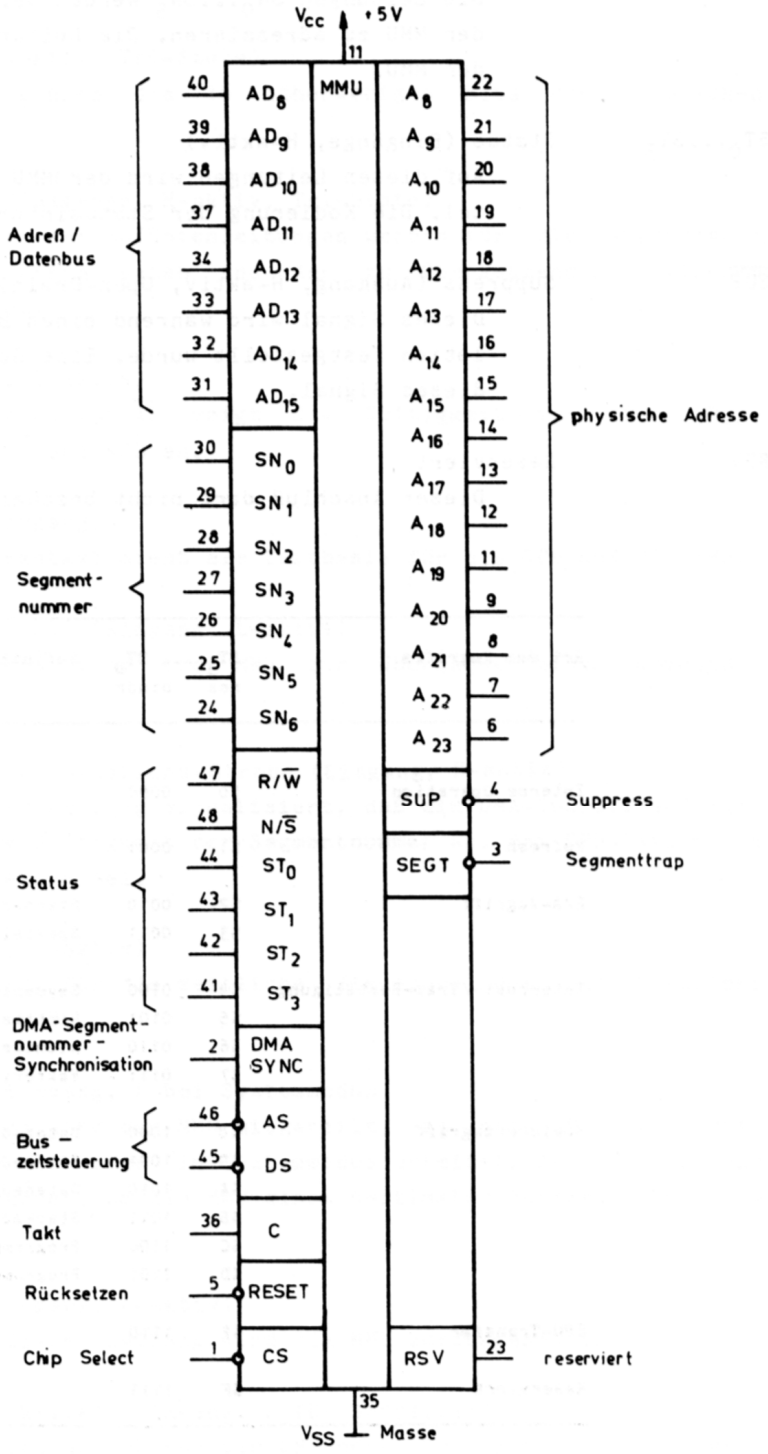


Bild C.2: Schaltbild MMU U8010

D. Zusammenstellung der Register und Flags

D.1. Segment-Deskriptor-Register (SDR)

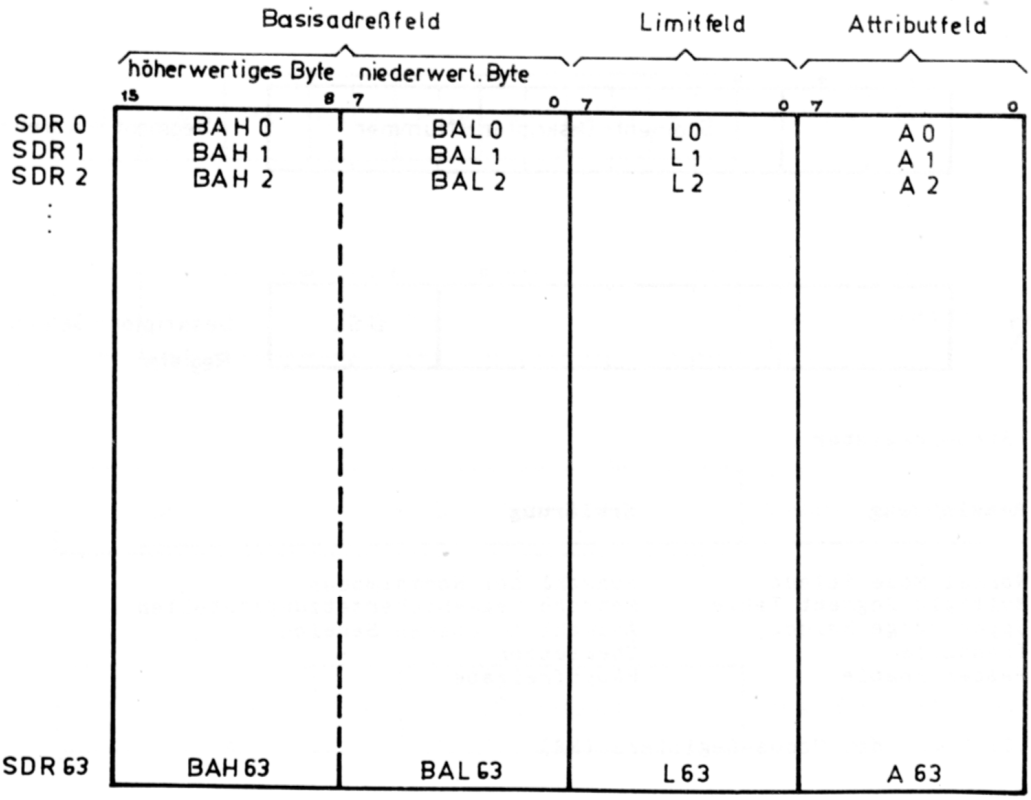


Bild D.1: Segment-Deskriptor-Register

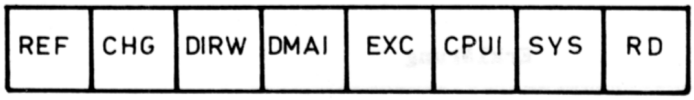


Bild D.2: Flags des Attributfeldes

Symbol	**Bezeichnung**	**Erklärung**
RD	Read-Only	nur Lesezugriffe erlaubt
SYS	System-Only	nur Systemmoduszugriffe erlaubt
CPUI	CPU-Inhibit	CPU-Zugriffe verboten
EXC	Execute-Only	nur Befehlsholezugriffe erlaubt
DMAI	DMA-Inhibit	DMA-Zugriffe verboten
DIRW	Direction and Warning	Stacksegment (Richtung und Warnung)
CHA	Changed	Veränderung
REF	Referenced	Benutzung

Tabelle D.1: Flags des Attributfeldes

D.2. Steuerregister

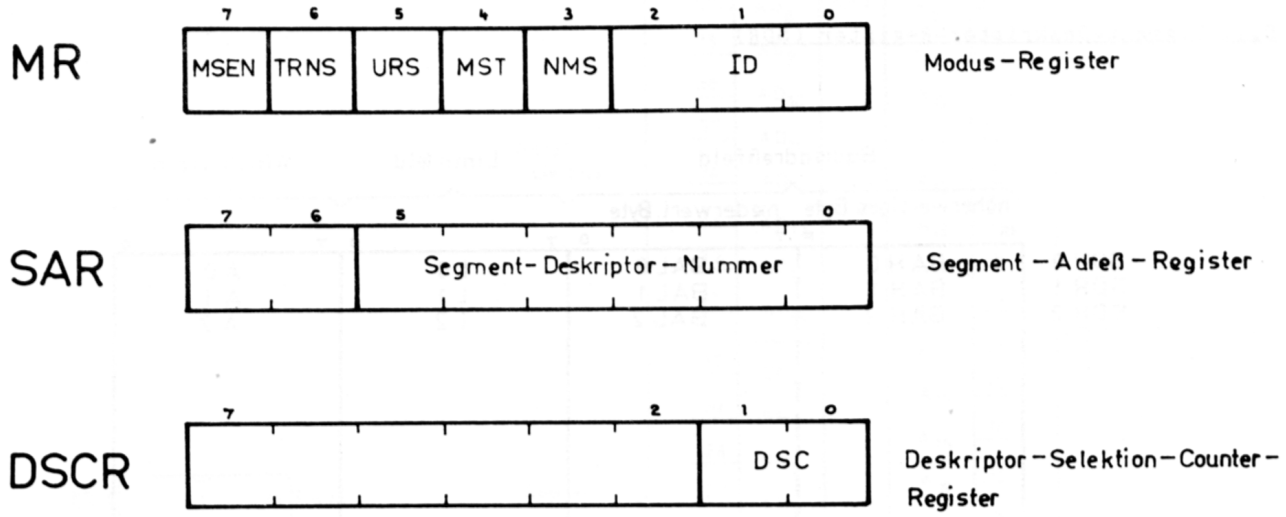


Bild D.3: Steuerregister

Symbol	Bezeichnung	Erklärung
NMS	Normal Mode Select	Auswahl bei Normalmodus
MST	Multiple Segment Table	mehrere Segmentübersetzungstabellen
URS	Upper Range Select	Auswahl im oberen Bereich
TRNS	Translate	Übersetzen
MSEN	Master Enable	Hauptfreigabe

Tabelle D.2: Flags des Modus-Registers (MR)

D.3. Statusregister

Symbol	Bezeichnung	Erklärung
FATL	Fatal Condition	fataler Fehler
SWW	Secondary Write Warning	sekundäre Schreibwarnung
PWW	Primary Write Warning	primäre Schreibwarnung
RDV	Read-Only Violation	Verletzung eines nur lesbaren Segments
EXCV	Execute-Only Violation	Verletzung eines nur ausführbaren Segments
CPUIV	CPU-Inhibit Violation	Verletzung eines CPUI-Segments
SLV	Segment Length Violation	Verletzung der Segmentlänge
SYSV	System Violation	Verletzung eines Systemsegments

Tabelle D.3: Flags des VTR

Symbol	Bezeichnung	Erklärung
N/$\overline{S}$	Normal/System	Normal-/Systemmodus
R/$\overline{W}$	Read/Write	Lese-/Schreibzugriff

Tabelle D.4: Flags des BCSR

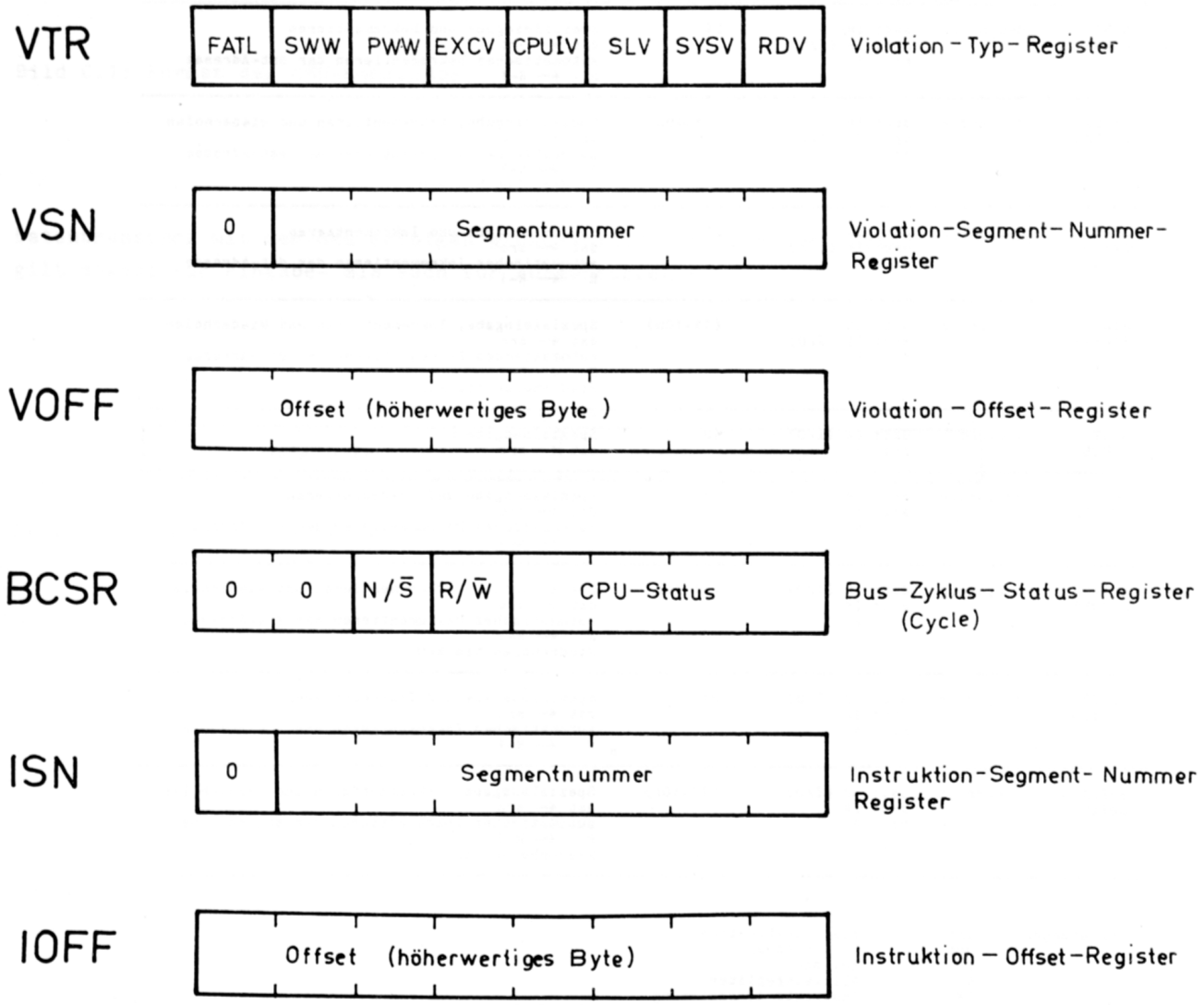


Bild D.4: Statusregister

E. Zusammenstellung der Kommandos

E.1. Spezial-E/A-Befehle

Mnemonik	Operanden	Adressierungsarten	Taktzyklen	Wirkung
SIN SINB	dst,src	dst: R src: DA (I/O)	12	Spezialeingabe dst ← src
SIND SINDB	dst,src,r	dst: IR src: IR (I/O) r: R	21	Spezialeingabe und Dekrementieren dst ← src automatisches Dekrementieren der dst-Adresse R ← R-1
SINDR SINDRB	dst,src,r	dst: IR src: IR (I/O) r: R	(11+10n)	Spezialeingabe, Dekrementieren und Wiederholen dst ← src automatisches Dekrementieren der dst-Adresse R ← R-1 Wiederholen bis R=0
SINI SINIB	dst,src,r	dst: IR src: IR (I/O) r: R	21	Spezialeingabe und Inkrementieren dst ← src automatisches Inkrementieren der dst-Adresse R ← R-1
SINIR SINIRB	dst,src,r	dst: IR src: IR (I/O) r: R	(11+10n)	Spezialeingabe, Inkrementieren und Wiederholen dst ← src automatisches Inkrementieren der dst-Adresse R ← R-1 Wiederholen bis R=0
SOUT SOUTB	dst,src	dst: DA (I/O) src: R	12	Spezialausgabe dst ← src
SOUTD SOUTDB	dst,src,r	dst: IR (I/O) src: IR r: R	21	Spezialausgabe und Dekrementieren dst ← src automatisches Dekrementieren der src-Adresse R ← R-1
SOTDR SOTDRB	dst,src,r	dst: IR (I/O) src: IR r: R	(11+10n)	Spezialausgabe, Dekrementieren und Wiederholen dst ← src automatisches Dekrementieren der src-Adresse R ← R-1 Wiederholen bis R=0
SOUTI SOUTIB	dst,src,r	dst: IR (I/O) src: IR r: R	21	Spezialausgabe und Inkrementieren dst ← src automatisches Inkrementieren der src-Adresse R ← R-1
SOTIR SOTIRB	dst,src,r	dst: IR (I/O) src: IR r: R	(11+10n)	Spezialausgabe, Inkrementieren und Wiederholen dst ← src automatisches Inkrementieren der src-Adresse R ← R-1 Wiederholen bis R=0

Erklärungen:

dst	Ziel (destination)
src	Quelle (source)
r	Register
R	Allzweckregister
IR	indirekte Adressierung
IR (I/O)	indirekte Adressierung einer E/A-Einheit (MMU-Kommandowort)
DA (I/O)	direkte Adressierung einer E/A-Einheit (MMU-Kommandowort)
n	Anzahl der transportierten Datenelemente

Tabelle C.1: Spezial-E/A-Befehle

E.2. Format der MMU-Kommandos

Kommandos werden der MMU durch die E/A-Adresse des Spezial-E/A-Befehls übergeben. Die Kommandos haben folgendes Format:

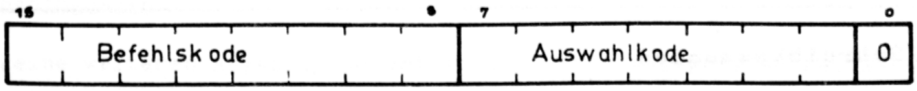


Bild C.1: Format der MMU-Kommandos

Datentransfers mit der MMU erfolgen grundsätzlich in Form von Bytes. Folgendes Format gilt sowohl für Eingabe- als auch für Ausgabebefehle.

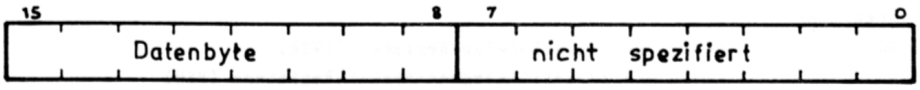


Bild C.2: Format der MMU-Daten

E.3. MMU-Befehlskodes

Befehlskode	Wirkung
Lese-/Schreibkommandos	
Segment-Deskriptor-Register	
08	Lesen/Schreiben des Basisfeldes im Deskriptor
09	Lesen/Schreiben des Limitfeldes im Deskriptor
0A	Lesen/Schreiben des Attributfeldes im Deskriptor
0B	Lesen/Schreiben des Deskriptors (alle Felder)
0C	Lesen/Schreiben des Basisfeldes und Erhöhen des SAR
0D	Lesen/Schreiben des Limitfeldes und Erhöhen des SAR
0E	Lesen/Schreiben des Attributfeldes und Erhöhen des SAR
0F	Lesen/Schreiben des Deskriptors (alle Felder) und Erhöhen des SAR
Steuerregister	
00	Lesen/Schreiben des Modus-Registers (MR)
01	Lesen/Schreiben des Segment-Adreß-Registers (SAR)
20	Lesen/Schreiben des Deskriptor-Selektor-Counter-Registers (DSCR)
Statusregister	
02	Lesen des Violation-Typ-Registers (VTR)
03	Lesen des Violation-Segmentnummer-Registers (VSN)
04	Lesen des Violation-Offset-Registers, höherwertiges Byte (VOFF)
05	Lesen des Bus-Zyklus-Status-Registers (BCSR)
06	Lesen des Instruktion-Segmentnummer-Registers (ISN)
07	Lesen des Instruktion-Offset-Registers, höherwertiges Byte (IOFF)
Setz-/Rücksetzkommandos	
10	Reset (Rücksetzen von MR, VTR und DSCR)
Violation-Status-Register	
11	Rücksetzen Violation-Typ-Register (VTR)
13	Rücksetzen SWW-Flag im VTR
14	Rücksetzen FATL-Flag im VTR
Segment-Deskriptor-Register	
15	Setzen aller CPU-Inhibit-Flags
16	Setzen aller DMA-Inhibit-Flags
Reservierte Kommandos	
12	nicht benutzen
17-1F	nicht benutzen
21-FF	nicht benutzen

F. Elektrische Kennwerte

Diese Angaben haben informativen Charakter.
Verbindliche Angaben sind dem Fachbereichstandard TGL 43 020 zu entnehmen.

F.1. Testbedingungen

Erfolgen keine weiteren Angaben, so gelten die unten angegebenen Kennwerte unter folgenden Testbedingungen:

$4{,}75\,V \leqq U_{CC} \leqq 5{,}25\,V$; $U_{SS} = 0\,V$; $0\,^{o}C \leqq \vartheta_a \leqq 70\,^{o}C$

Folgende Testschaltung wird verwendet:

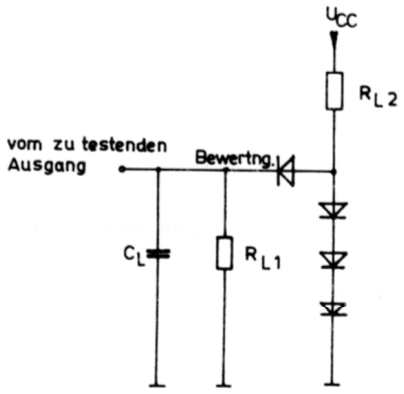


Bild F.1: Testlast

F.2. Statische Kennwerte

Symbol	Parameter	Min	Max	Einheit	Bedingungen
U_{CH}	Taktpegel (H)	U_{CC}-0,4	U_{CC}+0,3	V	externer Taktgenerator
U_{CL}	Taktpegel (L)	-0,5	+0,45	V	externer Taktgenerator
U_{IH}	Eingangspegel (H)	2	U_{CC}+0,3	V	
U_{IL}	Eingangspegel (L)	-0,5	0,8	V	
U_{OH}	Ausgangspegel (H)	2,4		V	I_{OH} = 250 µA
U_{OL}	Ausgangspegel (L)		0,4	V	I_{OL} = 2,0 mA
I_{IL}	Eingangsreststrom		± 10	µA	$0{,}4 \leqq U_{IN} \leqq 2{,}4$ V
I_{OL}	Ausgangsreststrom		± 10	µA	$0{,}4 \leqq U_{IN} \leqq 2{,}4$ V
I_{CC}	Gesamtstromaufnahme		300	mA	

Kennwerte bei: U_{CC} = 5 V ± 5%; ϑ_a = 0 ... 70°C

F.3. Grenzwerte

Es sind folgende Grenzwerte einzuhalten:

Betriebsspannung und Spannung an allen Ein- und Ausgängen: $-0{,}5V \leqq U \leqq 7{,}0V$

Lagerungstemperaturbereich: $-55\ ^{o}C \leqq \vartheta_{stg} \leqq 125\ ^{o}C$

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
5020 Erfurt Rudolfstr. 47

Bemerkungen und Verbesserungsvorschläge des Lesers

Beschreibung:

Titel: Technische Beschreibung U8010
Ausgabe: 11/85

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 061 306

Rs 1205/86 V 7 1 1593 N 2