

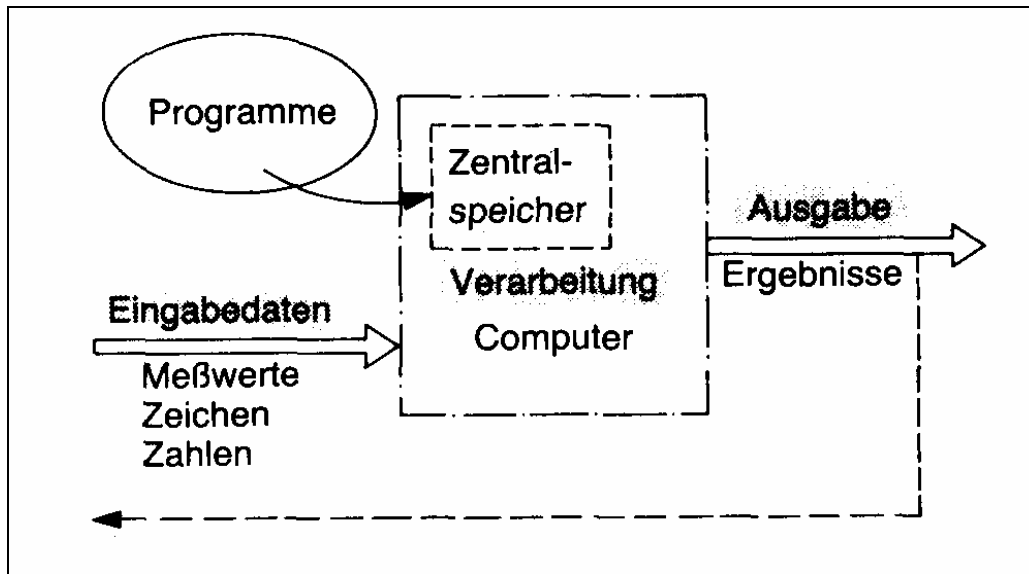
<b>1 Einführung:</b> .....	<b>4</b>
1.1 Prinzip der Datenverarbeitung .....	4
1.1.1 Binäre Daten: .....	4
1.1.2 Verarbeitung binärer Daten.....	4
1.2 Blockschaltbild eines Mikrocontrollers .....	5
1.2.1 Zentraleinheit .....	5
1.2.2 Zentralspeicher.....	6
1.2.3 Ein-/Ausgabe- Bausteine .....	6
1.2.4 Busleitungen .....	6
1.3 Prinzipieller Ablauf eines Programms .....	7
<b>2 Microcontroller der Serie 8051</b> .....	<b>8</b>
2.1 Die Architektur des MAB 8051 AH .....	9
2.1.1 Überblick.....	9
2.1.2 Speicheraufbau.....	10
2.1.3 Akkumulator .....	10
2.1.4 Register B.....	10
2.1.5 Programmstatuaregister .....	11
2.1.6 Stack pointer .....	11
2.1.7 Datenzeiger .....	11
2.1.8 Ports 0 bis 3.....	11
2.1.9 Zeitgeber-Register.....	12
2.1.10 Serieller Datenpuffer.....	12
2.1.11 Steuerregister .....	12
<b>3 Überblick über den 80C517/80C517A</b> .....	<b>13</b>
3.1 Speicherorganisation.....	15
3.1.1 Allgemeines .....	15
3.1.2 Interner Datenspeicher .....	15
3.1.3 Die Code-Data-Struktur des 8051 .....	16
<b>4 C-Programme für den 8051</b> .....	<b>18</b>
4.1 Allgemeines .....	18
4.2 Zugriff auf Special Function Register.....	18
<b>5 Standardbeschaltung des Microcontrollers 80C537</b> .....	<b>20</b>
<b>6 Serielle Schnittstelle(n)</b> .....	<b>21</b>
6.1 RS232-Schnittstelle an einem Personal Computer: .....	23
6.2 Andere serielle Schnittstellen: .....	24
6.3 Die seriellen Schnittstellen des 80C517 .....	26
6.3.1 Serielle Schnittstelle 0.....	26
6.3.2 Serielle Schnittstelle 1: .....	29
<b>7 Speicher</b> .....	<b>31</b>
7.1 Allgemeines .....	31
7.2 Speicherorganisation:.....	32
7.3 Statische RAMs: .....	33
7.3.1 TTL-Speicherzelle: .....	33
7.3.2 CMOS-Speicherzelle: .....	35
7.3.3 Symbole für Impuls- bzw. Zeitdiagramme: .....	36
7.3.4 Zeitbedingungen: .....	36
7.3.5 Beispiel aus dem Datenbuch:.....	37
7.4 Dynamische RAMs:.....	39
7.4.1 Speicherzellenaufbau: .....	40
7.4.2 Dynamic RAM Controller: .....	41

7.5 FIFO-Speicher (First In First Out Memories).....	45
7.6 Festwertspeicher (ROM).....	45
7.6.1 Masken-ROMs:.....	46
7.6.2 PROM (Programmierbare Festwertspeicher): .....	46
7.7 UV-löschbare Festwertspeicher (EPROM): .....	47
7.8 Elektrisch löschbare Festwertspeicher (EEPROMs): .....	49
7.9 Flash-EEPROMs:.....	50
7.10 Zweitortspeicher: .....	50
<b>8 Anschluß eines LCDs .....</b>	<b>52</b>
8.1 Allgemeines .....	52
8.2 Anschluß an die Druckerschnittstelle .....	52
8.3 Beispiel für den Anschluß an einen 8031er Adresse F000H: .....	53
8.4 Anschluß direkt an einen Port:.....	53
8.5 Befehlssatz .....	54
<b>9 Reset .....</b>	<b>57</b>
<b>10 Parallele Eingabe-Ausgabe-Ports .....</b>	<b>58</b>
10.1 Digitale Ein-Ausgabe-Ports .....	58
10.2 Analog/Digitale Eingabeports.....	61
10.3 Alternative Port-Funktionen .....	62
<b>11 Timer .....</b>	<b>65</b>
11.1 Modus 0 .....	66
11.2 Modus 1 .....	68
11.3 Modus 2 .....	69
11.4 Modus 3 .....	70
<b>12 Capture/Compare-Einheit .....</b>	<b>71</b>
12.1 Begriffserklärung .....	71
12.1.1 Compare .....	71
12.1.2 Reload .....	71
12.1.3 Capture.....	71
12.2 Anwendungsbeispiel PWM.....	76
<b>13 Analog - Digital - Umsetzer .....</b>	<b>79</b>
13.1 Allgemeines .....	79
13.1.1 Parallelverfahren (flash converter): .....	80
13.1.2 Wägeverfahren (successive approximation) .....	81
13.1.3 Zählverfahren (incremental converter) .....	82
13.1.4 Tracking ADC (Nachlaufender ADU):.....	83
13.1.5 Zwei-Rampenverfahren (dual slop, dual ramp):.....	84
13.2 Der ADU des 80C517 .....	87
13.3 Erklärungen.....	87
13.4 Blockschaltbild ADU 80C517 .....	88
13.5 Initialisierung und Wahl des Eingangskanals .....	89
13.6 Erhöhung der Umsetzerauflösung von 8 Bit auf 10 Bit:.....	90
13.7 Anpassen der Referenzspannungen an die Eingangsspannung: .....	90
13.7.1 Interne einstellbare Referenzspannungen: .....	90
<b>14 Sicherheitsmechanismen .....</b>	<b>92</b>
14.1 Watchdog-Timer (WDT) .....	92
14.2 Oszillator-Watchdog (OWD).....	94
14.3 Oszillator-Watchdog im 80C517 .....	95
14.4 Oszillator-Watchdog im 80C517A .....	95
<b>15 Oszillator und Taktversorgung .....</b>	<b>97</b>

<b>16 Interruptsystem.....</b>	<b>99</b>
16.1 Einlesen eines Ports .....	99
16.2 Das Polling Verfahren.....	99
16.3 Die Programmierung von Interrupt Routinen .....	100
16.3.1 Allgemeines .....	100
16.4 Vektoradressen.....	100
16.5 Interruptsystem 80C517.....	101
16.6 Special Function Register für Interrupts .....	103
16.7 Prioritätssteuerung .....	108
16.8 Eine Interruptgesteuerte Zeitschleife .....	109
16.9 Interruptquellen und -nummern .....	110
<b>17 Literaturnachweis .....</b>	<b>111</b>

# 1 Einführung

## 1.1 Prinzip der Datenverarbeitung



**Eingabe:** Eingabegeräte stellen Daten zur Verarbeitung bereit

**Beispiele:** Sensoren, Taster, Datenspeicher, Bussystem

**Verarbeitung:** Diese Daten werden nun von der eigentlichen Verarbeitungseinheit, dem Computer, gelesen und entsprechend einer im Computer vorhandenen Arbeitsanweisung, dem Programm, verarbeitet.

**Ausgabe:** Die Ergebnisse dieser Verarbeitung müssen ausgegeben werden.

**Beispiele:** Speicher, Display, Bussystem, Drucker, Steuerelektronik

### 1.1.1 Binäre Daten

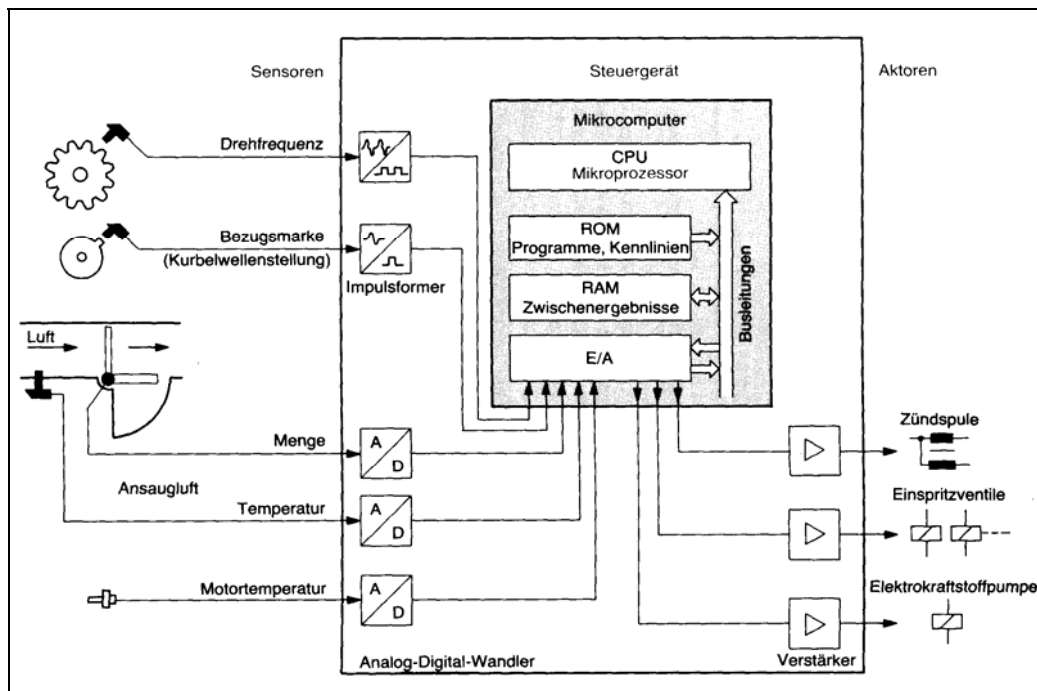
Die Daten werden in binärer Form verarbeitet, d.h. analoge Signale müssen zur weiteren Verarbeitung erst in digitale Signale umgeformt werden. Die digitale Verarbeitung hat sich aus mehreren Gründen durchgesetzt:

- Daten, z.B. analoge Meßdaten, können nur in digitalisierter Form über längere Zeit gespeichert werden.
- Die Genauigkeit der Verarbeitung kann man mit relativ geringem Aufwand praktisch beliebig steigern.
- Digitalisierte Daten sind bei Übertragung störungsempfindlicher als analoge Signale.

### 1.1.2 Verarbeitung binärer Daten

Bei einem Computer handelt es sich nicht um eine Schaltung, die speziell für eine bestimmte Aufgabe konstruiert ist (wie GAL), sondern um eine universelle binäre Schaltung, die eingelesene Daten entsprechend den Befehlen des im Computer gespeicherten Programms bearbeiten.

**Beispiel: Datenverarbeitung bei der Steuerung eines Ottomotors**



**1.2 Blockschaltbild eines Mikrocontrollers**

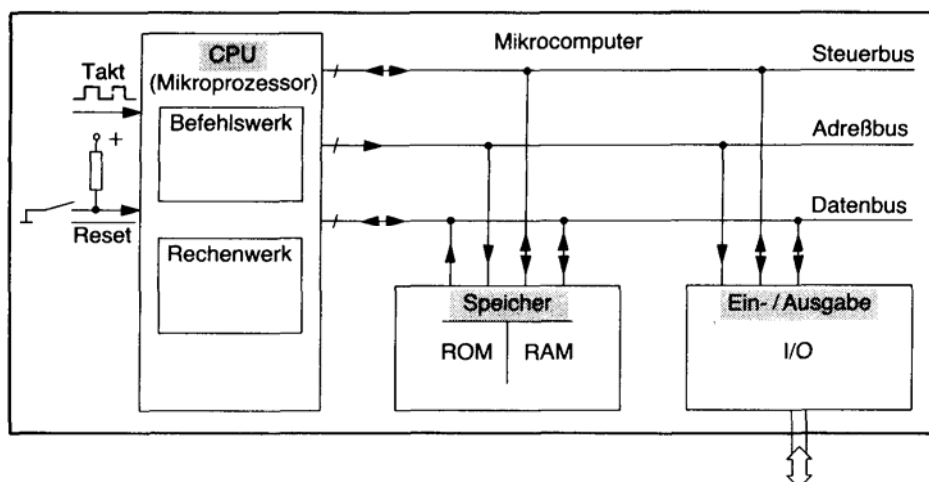
Jeder digital arbeitende Computer besteht aus 3 Baugruppen

- ➔ Zentraleinheit (Central Processing Unit, CPU)
- ➔ Zentralspeicher (Memory)
- ➔ Ein-/Ausgabe - Einheiten (Input/Output, I/O)

**1.2.1 Zentraleinheit**

Die CPU hat zwei Aufgaben:

- a) Sie steuert den gesamten Ablauf
- b) Sie bearbeitet Daten, kann also arithmetische und logische Operationen ausführen.



Blockschaltbild eines Mikrocontrollers

### 1.2.2 Zentralspeicher

Zentralspeicher werden von der CPU direkt angesteuert, im Gegensatz zu peripheren Speichern, z.B. Festplatten, Disketten, Bussysteme, die über spezielle E/A Bausteine angesteuert werden müssen.

#### →Nichtflüchtige Speicher

(Non Volatile Memory) behalten ihre Information auch beim Ausschalten der Versorgungsspannung. Sie sind geeignet für Programme, Codetabellen usw. (Bausteine: ROM, EPROM, EEPROM)

#### →Flüchtige Speicher

verlieren ihre Information beim Ausschalten der Versorgungsspannung, dafür lassen sie sich einfach ändern. Sie werden zum Zwischenspeichern von Eingabedaten und Ergebnissen benötigt.

(Bausteine: RAM (Random Access Memory), FIFO (First in first out) und LIFO)

### 1.2.3 Ein-/Ausgabe- Bausteine

Sie stellen die Verbindung zur Außenwelt des Computers, zur Peripherie, her. Sie werden oft als Ports bezeichnet.

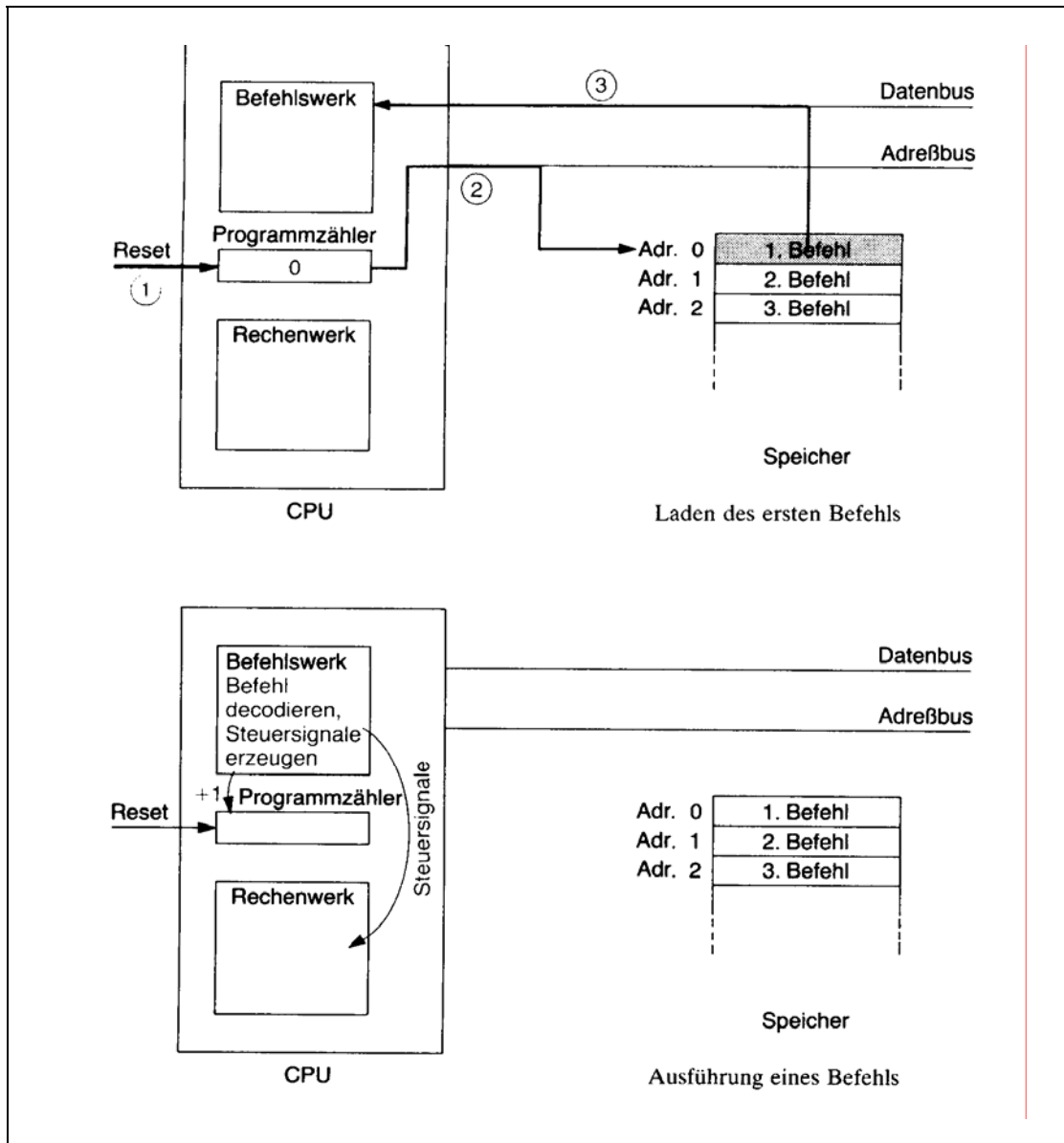
### 1.2.4 Busleitungen

Der Bus besteht aus mehreren Leitungen, wobei an jede Leitung parallel mehrere Bausteine angeschlossen sind. Zur korrekten Abwicklung des Datenflusses werden im Systembus eines Computers entsprechend ihrer Aufgabe drei Gruppen von Busleitungen unterschieden:

- a) **Datenbus**
- b) **Adreßbus**
- c) **Steuerbus**

Da an einer Datenleitung immer mehrere Sender angeschlossen sind, spricht man hier von einem bidirektionalen Bus.

### 1.3 Prinzipieller Ablauf eines Programms



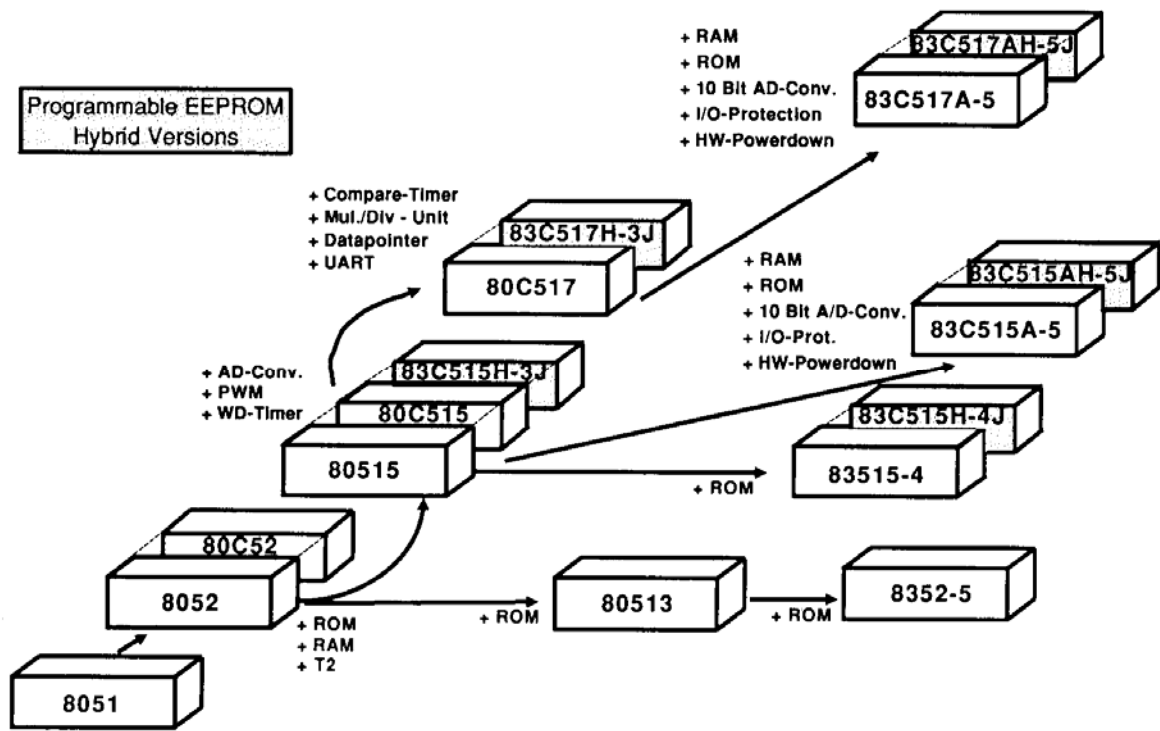
Prinzipieller Ablauf eines Programms in einem Mikrokontrollersystem

## 2 Microcontroller der Serie 8051

Als Mikrocontroller bezeichnet man Integrierte Schaltungen, die auf einem einzigen Chip ein vollständiges Mikrocomputersystem enthalten.

Mikrocontroller werden verwendet, um Regelungen und Steuerungen aufzubauen, um mechanische Aktoren zu bedienen, Sensoren einzulesen und Echtzeitberechnungen durchzuführen.

Aufbauend auf dem 8051, der von Intel entwickelt wurde, wurden zahlreiche Derivate von verschiedenen Firmen entwickelt. Als Vertreter solcher Derivate seien der Microcontroller 83C552 von Philips und der 80C517 von Siemens genannt. Beide sind leistungsfähige Erweiterungen des 8051. Sie enthalten den kompletten 8051, besitzen eine Menge von zusätzlichen Peripheriefunktionen und sind völlig softwarekompatibel.



Überblick der 8051-Familie von Siemens



## 2.1 Die Architektur des MAB 8051 AH

### 2.1.1 Überblick

Die wesentlichen Merkmale des Mikrocontrollers MAB 8051 AH sind:

- 8bit-Zentraleinheit (CPU),
- 4KByte-Programmspeicher (ROM),
- 128Byte-Datenspeicher (RAM),
- 21 spezielle Funktionsregister,
- 32 E/A-Leitungen,
- 64KByte-Adressierbereich für den externen Datenspeicher,
- 64KByte-Adressierbereich für den Programmspeicher (extern u. intern zusammen),
- zwei 16bit-Zeitgeber/Zähler,
- Interrupts aus 5 Quellen mit 2 Prioritäten,
- serieller Port (Voll duplex),
- Boolescher Prozessor,
- Oszillator- und Taktgeberschaltungen (auf dem Chip),
- Multiplikation und Division durch je einen Befehl möglich,
- nicht pageorientierter Programmspeicher.

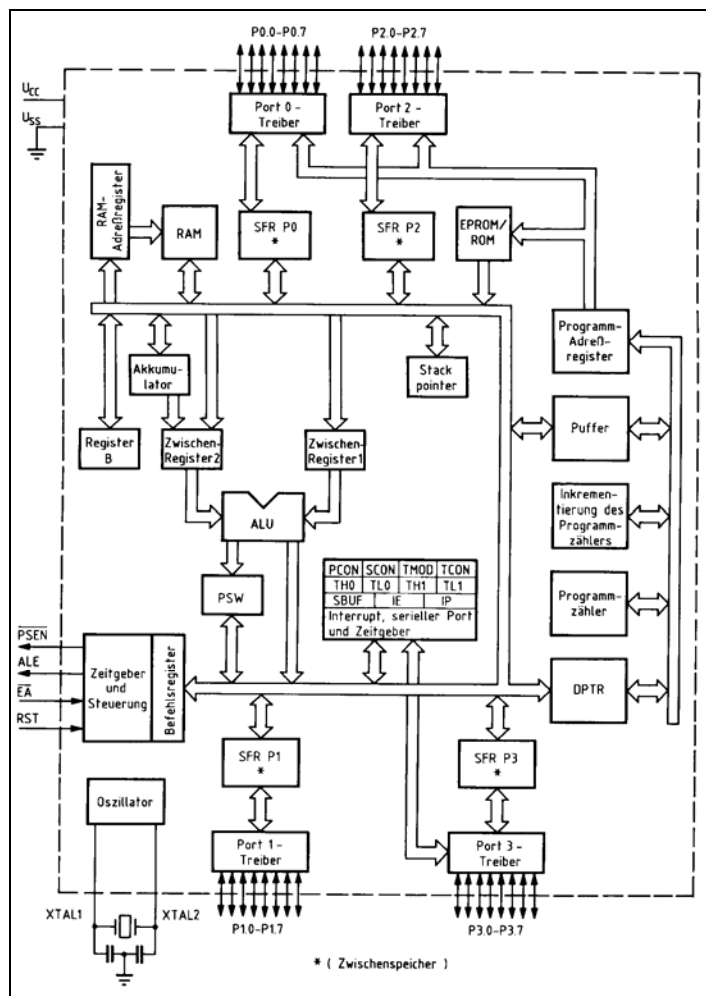


Abb. Blockschtung des MAB 8051 AH.

### 2.1.2 Speicheraufbau

Der MAB 8051 AH und der MAB 8031 AH verfügen über getrennte Adressierbereiche für den Programmspeicher und den Datenspeicher. Der externe Programmspeicher kann bis zu 64 KBytes umfassen; außerdem befinden sich im ROM auf dem Chip des MAB 8051 AH weitere 4 KBytes. Der Datenspeicher besteht aus einem auf dem Chip befindlichen RAM von 128 Bytes. In einem weiteren Adressierbereich von 128 Bytes sind nur 21 Bytes mit speziellen Funktionsregistern belegt. Außerdem kann die Schaltung auf bis zu 64 KBytes eines externen Datenspeichers zugreifen.

In der vorrigen Abbildung sind die 21 speziellen Funktionsregister mit ihren Kurzbezeichnungen sowie Speicherplätzen aufgeführt und anschließend kurz beschrieben. Die mit einem Stern versehenen Register sind sowohl byte- als auch bitadressierbar (11 Register mit Adressen, die durch 8 ohne Rest teilbar sind).

### 2.1.3 Akkumulator

Das Register ACC ist der Akkumulator. In den mnemonischen Kurzbezeichnungen, die für Befehle mit Bezug auf den Akkumulator verwendet werden, wird der Akkumulator nur mit A bezeichnet.

### 2.1.4 Register B

Das Register B wird beim Multiplizieren und Dividieren benötigt. Bei der Abarbeitung anderer Befehle kann es als weiterer schneller Hilfsspeicher dienen.

**Tabelle: Spezielle Funktionsregister des MAB 8051 AH**

Kurzbezeichnung für Assembler	Register	Adresse hexa- dezimal
	dezimal	
ACC*)	Akkumulator	224 EO
B*)	Register B	240 FO
PSW*)	Programmstatuswort	208 DO
SP	Stack pointer	129 81
DPH	Datenzeiger (DPTR) oberes Byte	131 83
DPL	Datenzeiger (DPTR) unteres Byte	130 82
PO*)	Port 0	128 80
PI *)	Port 1	144 90
P2*)	Port 2	160 AO
P3*)	Port 3	176 BO
IP*)	Interrupt-Prioritäten-Register	184 B8
IE*)	Interrupt-Freigabe-Register	168 A8
TMOD	Zeitgeber/Zähler-Betriebsart-Register	137 89
TCON*)	Zeitgeber/Zähler-Steuerregister	136 88
THO	Zeitgeber/Zähler 0 (oberes Byte)	140 8C
TLO	Zeitgeber/Zähler 0 (unteres Byte)	138 8A
TH1	Zeitgeber/Zähler 1 (oberes Byte)	141 80
TL1	Zeitgeber/Zähler 1 (unteres Byte)	139 8B
SCHON*)	serielles Steuerregister	152 98
SBUF	serieller Datenpuffer	153 99
PCON	Energie-Steuerregister	135 87

\*) auch bitadressierbar

### 2.1.5 Programmstatusregister

Das Programmstatusregister (PSW) enthält das in der nachstehenden näher beschriebene Programmstatuswort.

### 2.1.6 Stack pointer

Der 8 bit breite Stack pointer (SP) wird inkrementiert, bevor Daten während der Ausführung eines PUSH- oder CALL-Befehls gespeichert werden. Während der Stack im allgemeinen irgendwo im RAM (auf dem Chip) untergebracht sein kann, zeigt der Stack pointer nach einem Rücksetz-Vorgang auf den Speicherplatz 07H. Dies veranlaßt den Stack, mit Speicherplatz 08H zu starten.

### 2.1.7 Datenzeiger

Der 16bit-Datenzeiger (DPTR) besteht aus den Registern DPH (oberes Byte) und DPL (unteres Byte). Er enthält eine 16bit-Adresse und kann entweder als 16bit-Register oder als zwei unabhängige 8bit-Register arbeiten.

### 2.1.8 Ports 0 bis 3

Die speziellen Funktionsregister PO, PI, P2 und P3 sind die Zwischenspeicher für die Ports PO, PI, P2 bzw. P3.

## Programmstatusregister

höchstwertiges Bit		niedrigstwertiges Bit	
CY	AC	F0	RS1
RS0	OV	—	P
Symbol	Bit-Speicherstelle	Name und Bedeutung	
CY	PSW.7	Übertragsbit	
AC	PSW.6	Hilfsübertragsbit (für BCD-Operationen)	
F0	PSW.5	Kennzeichnungsbit 0 (steht dem Anwender für allgemeine Zwecke zur Verfügung)	
RS1	PSW.4	Registerbank-Auswahlbits 1 bzw. 0. Werden durch Software gesetzt oder gelöscht, um die Registerbank auszuwählen, in der gearbeitet werden soll.*)	
RS0	PSW.3		
OV	PSW.2	Überlaufbit	
—	PSW.1	in Reserve	
P	PSW.0	Paritätsbit. Wird durch Hardware bei jedem Befehlszyklus gesetzt bzw. gelöscht, um eine ungerade bzw. gerade Anzahl von Einsen im Akkumulator anzuzeigen (d. h. gerade Parität).	
*) Durch RS1 und RS0 wird diese Registerbank wie folgt festgelegt:			
RS1	RS0	Bank	Speicherplätze
0	0	0	00H—07H
0	1	1	08H—0FH
1	0	2	10H—17H
1	1	3	18H—1FH

### **2.1.9 Zeitgeber-Register**

Die Registerpaare TH0 und TLO sowie TH1 und TL1 sind 16bit-Zählregister für die Zeitgeber/Zähler 0 bzw. 1.

### **2.1.10 Serieller Datenpuffer**

Der serielle Datenpuffer (SBUF) besteht eigentlich aus zwei separaten Registern, nämlich einem Sendepuffer- und einem Empfangspufferregister. Wenn Daten nach SBUF transportiert werden, so erfolgt dies in das Sendepufferregister; der Transport eines Bytes nach SBUF löst dann die Übertragung in eine externe Einheit aus. Werden Daten von SBUF geholt (von einer Einheit auf dem Chip), so kommen sie aus dem Empfangspufferregister.

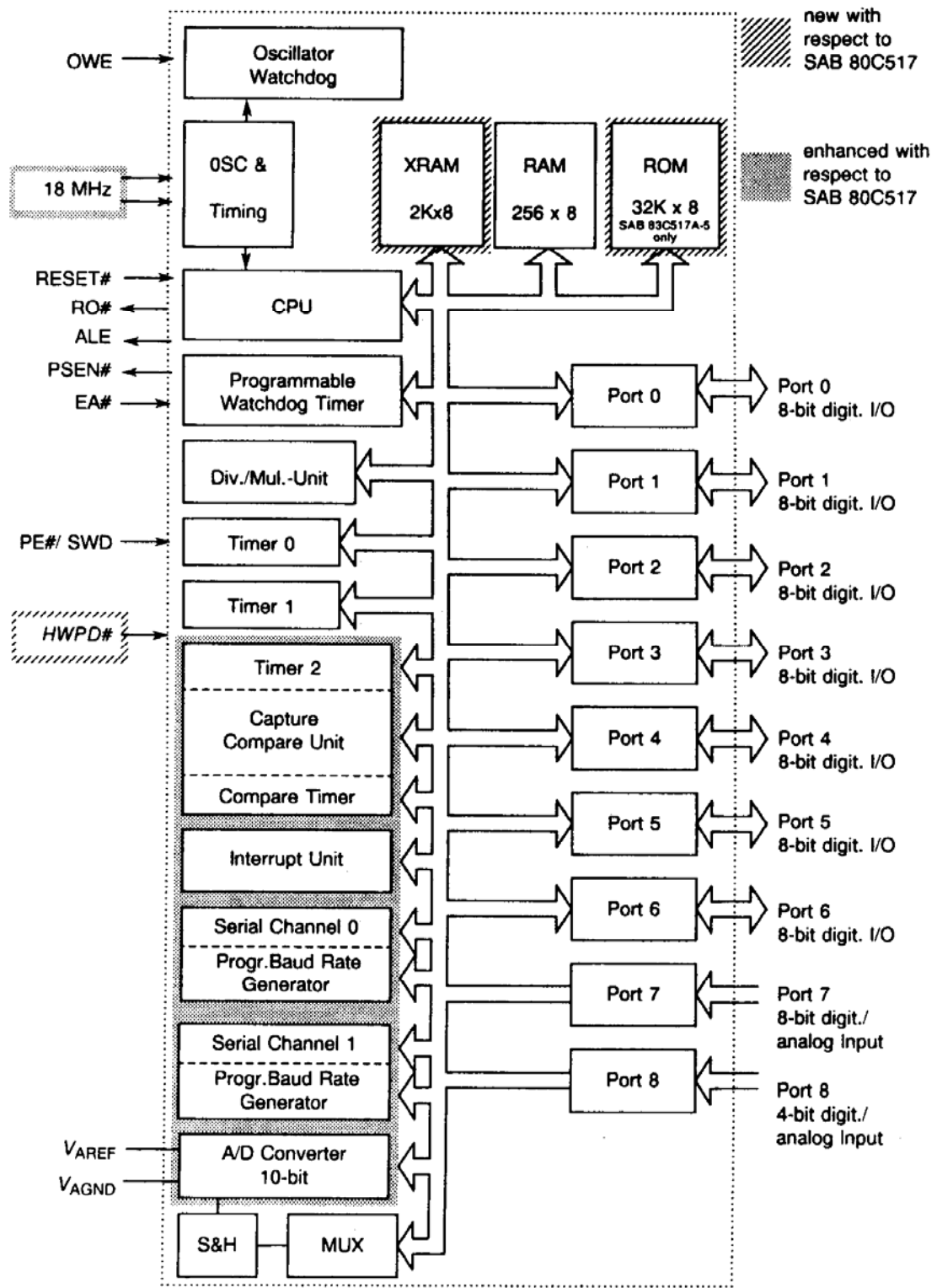
### **2.1.11 Steuerregister**

Die speziellen Funktionsregister IP, IE, TMOD, TCON, SCON und PCON enthalten Steuer- und Statusbits für das Interruptsystem, die Zeitgeber/Zähler und den seriellen Port. Sie werden in späteren Abschnitten beschrieben.

**Notizen:**

### 3 Überblick über den 80C517/80C517A

- ⊗ 8 Kbyte On-Chip-ROM beim 80C517
- ⊗ 32 Kbyte On-Chip-ROM beim 83C517A-5
- ⊗ ROM-lose Varianten sind auch erhältlich (80C537 und 80C517A)
- ⊗ Aufwärtskompatibel zum 8051 und 80515
- ⊗ 256 Byte On-Chip-RAM
- ⊗ Zusätzliche 2 Kbyte On-Chip-RAM beim 80C517A
- ⊗ Einzelbitverarbeitung
- ⊗ Versionen mit verschiedenen Taktgeschwindigkeiten erhältlich
- ⊗ Externe Erweiterbarkeit von Programm- und Datenspeicher (je 64 Kbyte)
- ⊗ On-Chip-A/D-Wandler (8 bit- Version beim 80C517, 10 bit- Version beim 80C517A) 12 Eingänge, wahlweise interner oder externer Start.
- ⊗ Zwei 16-bit Timer, voll kompatibel zum 80(C)51.
- ⊗ Universelle Compare/Capture-Einheit mit eigenen 16-bit Timern; unterschiedlich konfigurierbare 16-bit-Compare- und Capture-Register für zeitliche Auflösungen.
- ⊗ Arithmetik-Einheit für 16-bit-Arithmetik (Multiplikation, Division, Schiebeoperationen)
- ⊗ Integrierte Systemüberwachung: Programmierbarer 16-bit-Watchdog-Timer, Oszillator-Watchdog.
- ⊗ Neun Eingabe-Ausgabe-Ports, unterschiedlich konfigurierbar
- ⊗ Zwei unabhängige serielle Schnittstellen mit eigener Baudratenerzeugung
- ⊗ Interrupt-System mit 14 (beim 80C517) bzw. 17 (beim 80C517A) Interrupt-Vektoren und 4 Prioritätsebenen.
- ⊗ Betriebsarten zur Reduzierung der Stromaufnahme:  
Slow-Down-Modus      Idle- Modus      Power-Down-Modus
- ⊗ Gefertigt in CMOS-Technologie
- ⊗ Unterschiedliche Gehäusevarianten erhältlich: PLCC84, PQFP 100



Funktionsdiagramm des 80C517/80C517A

### 3.1 Speicherorganisation

#### 3.1.1 Allgemeines

Der Speicherraum ist logisch und physikalisch in vier verschiedene Bereiche unterteilt.

- a) bis zu 64 Kbyte Programmspeicher adressierbar.
- b) bis zu 64 Kbyte externer Datenspeicher adressierbar.
- c) 256 Byte interner Datenspeicher
- d) ein 128 Byte großer Bereich für Special-Function-Register (SFR)

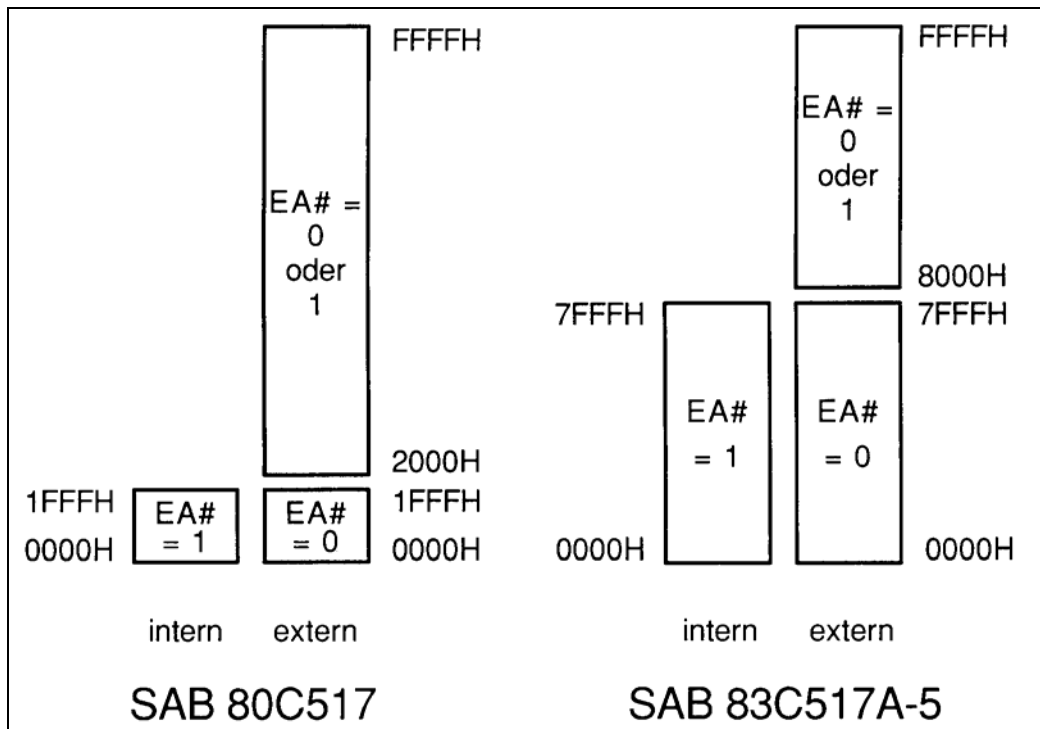
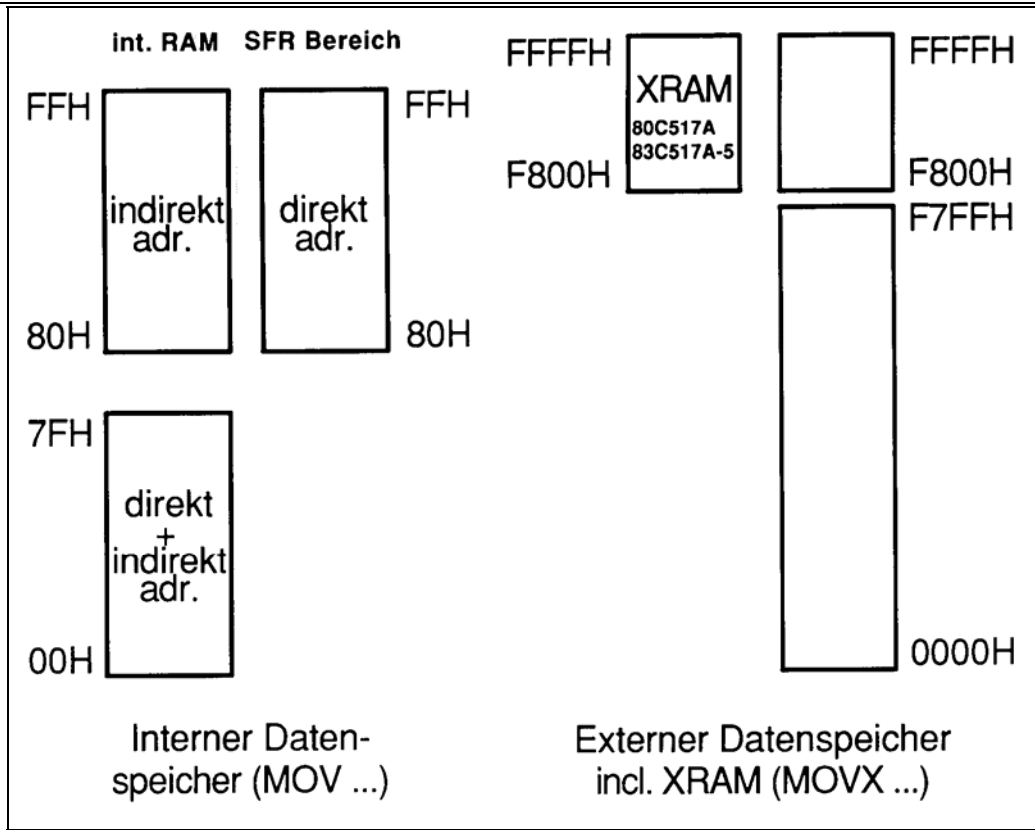


Abb. 3: Programmspeicheraufteilung

Im Bereich von 03H bis 93H liegen beim 80C517 des Programmspeichers die Einsprungadressen für die Interrupt-Routinen. Wenn einzelne Interrupts nicht benutzt werden, können aber diese Bereiche selbstverständlich für normalen Code verwendet werden.

#### 3.1.2 Interner Datenspeicher

Speicherbereich	Adresse	Adressierungsart
Untere 128 Byte RAM	00H bis 7FH	direkt/indirekt
Obere 128 Byte RAM	80H bis 0FFH	indirekt
SFR	80H bis 0FFH	direkt

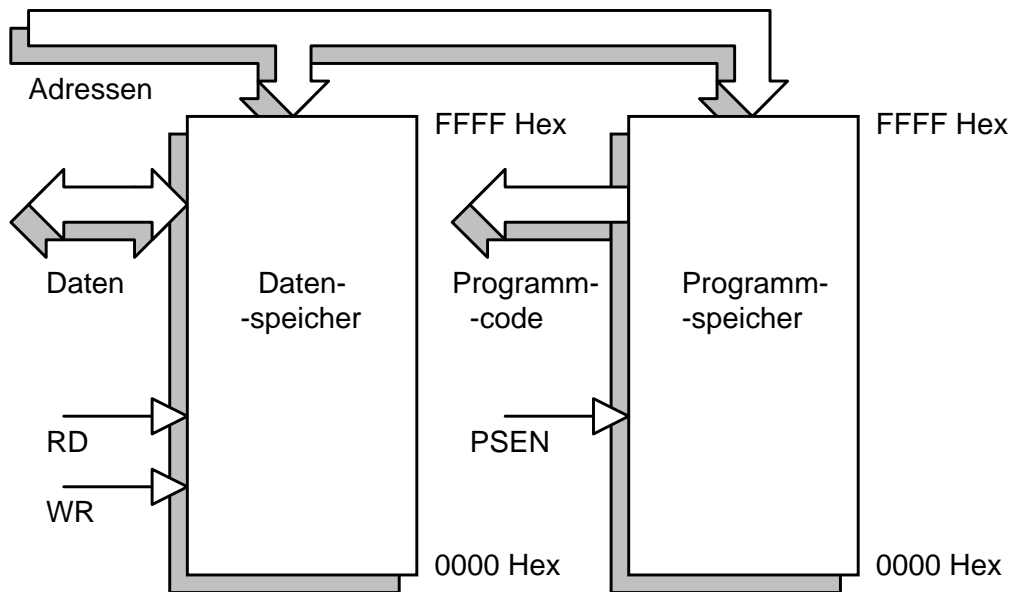


Aufteilung des internen Datenspeichers

### 3.1.3 Die Code-Data-Struktur des 8051

Alle Microcontroller der Familie 8051 verfügen über getrennte Adressierbereiche für den Programmspeicher und den Datenspeicher. Der externe Programmspeicher und der externe Datenspeicher kann bis zu 64 kByte umfassen.

Der Zugriff auf den externen Programmspeicher wird über das Signal PSEN gesteuert. Für den Zugriff auf den Datenspeicher stehen die Signale RD und WR zur Verfügung. Die Struktur eines solchen Speichersystems wird Harvard Struktur genannt.

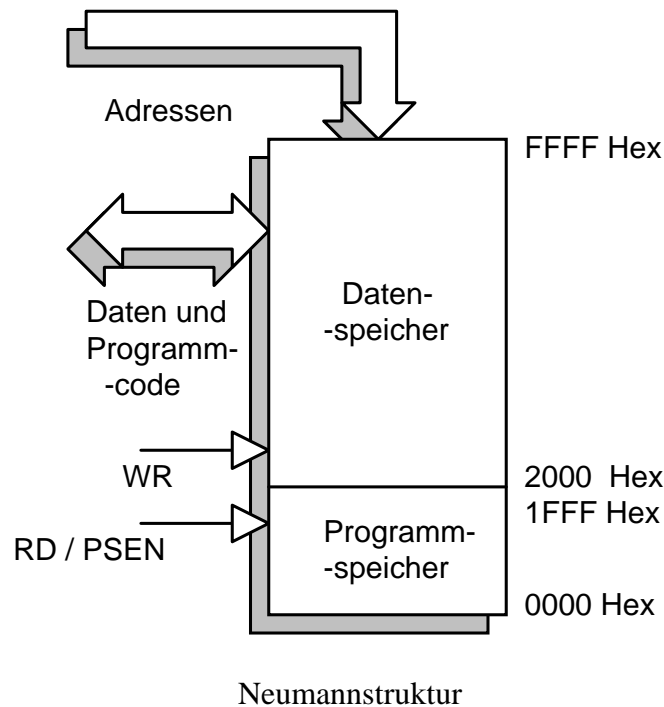


Harvard Struktur



Die Harvard Struktur wird während des normalen Betriebs des Mikrocontrollers verwendet. Zur Programmentwicklung wird jedoch eine andere Speicherstruktur verwendet. Sie wird Neumann Struktur genannt. Bei der Neumann Struktur liegt der Datenspeicher und der Programmspeicher in einem gemeinsamen Adressbereich. Dazu muß das Signal RD, welches aus dem RAM liest, und das Signal PSEN, welches aus dem ROM liest, verknüpft werden. Die UND Verknüpfung der beiden Signale ergibt ein Lesesignal für den gesamten Speicher.

Nötig ist diese Zusammenschaltung aus folgendem Grund: Während der Programmentwicklung läuft im ROM ein Monitorprogramm. Dieses kommuniziert über die serielle Schnittstelle mit dem PC. Auf dem PC wird das Anwenderprogramm geschrieben, kompiliert, gelinkt und in ein INTEL Hex Format gebracht. Das Anwenderprogramm wird mit Hilfe des Monitors in das RAM des 8051 geladen und vom Monitorprogramm gestartet. Das Programm kann bequem getestet und auf Fehlfunktionen überprüft werden.



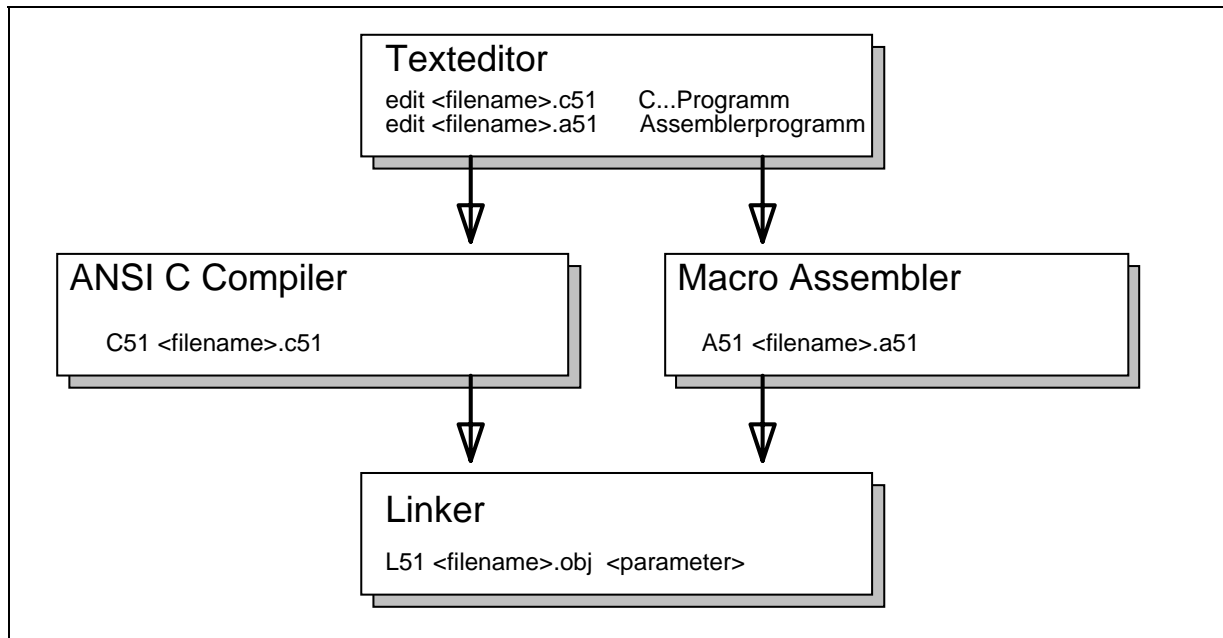
Erst wenn das Programm einwandfrei funktioniert, wird es in ein EPROM gebrannt. Dann wird wieder die Harvard Stuktur mit getrenntem Code- und Datenbereich verwendet.

## 4 C-Programme für den 8051

### 4.1 Allgemeines

Zur Programmierung der Mikrocontroller der Familie 8051 steht ein C-Compiler der Firma Keil zur Verfügung.

Der Ablauf bei der Erstellung eines C-Programms sieht folgendermaßen aus:



Mit dem Texteditor wird der C Source Code eingegeben und mit dem C Compiler übersetzt. Dieser liefert einen absoluten Object Modul für das Anlaufen des Programms an der Adresse 0 und einige verschiebbare (relocatable) Programm Module, denen erst beim Link Vorgang absolute Adressen zugewiesen werden. Beim Link Vorgang werden dann auch die mit dem C Compiler mitgelieferten Library Module, die von den Funktionsaufrufen angesprochen werden, in den Maschinencode miteingebunden.

Der Linker liefert den lauffähigen Code, der für die Datenübertragung mittels des Monitorprogramms noch mit einem Object Hex Konvertierungsprogramm in ASCII umgewandelt wird.

### 4.2 Zugriff auf Special Function Register

Für ein Beispielprogramm sollen die acht Leuchtdioden auf der Zusatzplatine blinken. Die Leuchtdioden sind über einen Treiber an den Port 1 des 8051 angeschlossen. Das Programm muß folgende Aufgaben erfüllen:

Zuerst soll der Port 1 rückgesetzt werden, so daß keine LED leuchtet. Anschließend soll ein bestimmtes Bitmuster am Port 1 ausgegeben werden. Das Bitmuster soll abwechselnd **01010101** und **10101010** lauten, dies entspricht den beiden HexZahlen 0x55 und 0xAA.

Die im Hauptprogramm verwendete Bezeichnung P1 bezeichnet den Port 1 des Microcontrollers. Die zugehörige Datendefinition ist in der Datei REG517.H abgespeichert und lautet `sfr P1 = 0x90`. Die Abkürzung *sfr* steht für *special function register* und bezeichnet einen Datentypen zum Zugriff auf ein special function register des 8051. Die Zeile `sfr P1 = 0x90` sagt aus, daß das special function register, über welches man auf Port 1 zugreifen kann, im Programm **P1** heißt und intern im Speicher des 8051 die Adresse **90Hex ( 0x90 )** hat.

Der C Source Code wird mit dem Texteditor von MS DOS eingegeben. Dieser wird mit dem Befehl `C:\>edit` gestartet.

```
/*
 *          PORT.C
 *          Erstes Testprogramm für BULME - Graz
 */

/****  Steueranweisungen an den Compiler  ****/
#pragma noiv
#pragma mod517
#pragma code debug pl(61)

/****  Angabe der Include Dateien  ****/
#include <reg517.h>

/****  Prototypen der Funktionen  ****/
void warte(unsigned int msec);

/****  Funktionen  ****/
void warte(unsigned int msec)

{ data char i;
  for (msec; msec!=0; msec--) for (i=0; i<83; i++);
}

/****  Hauptprogramm  ****/

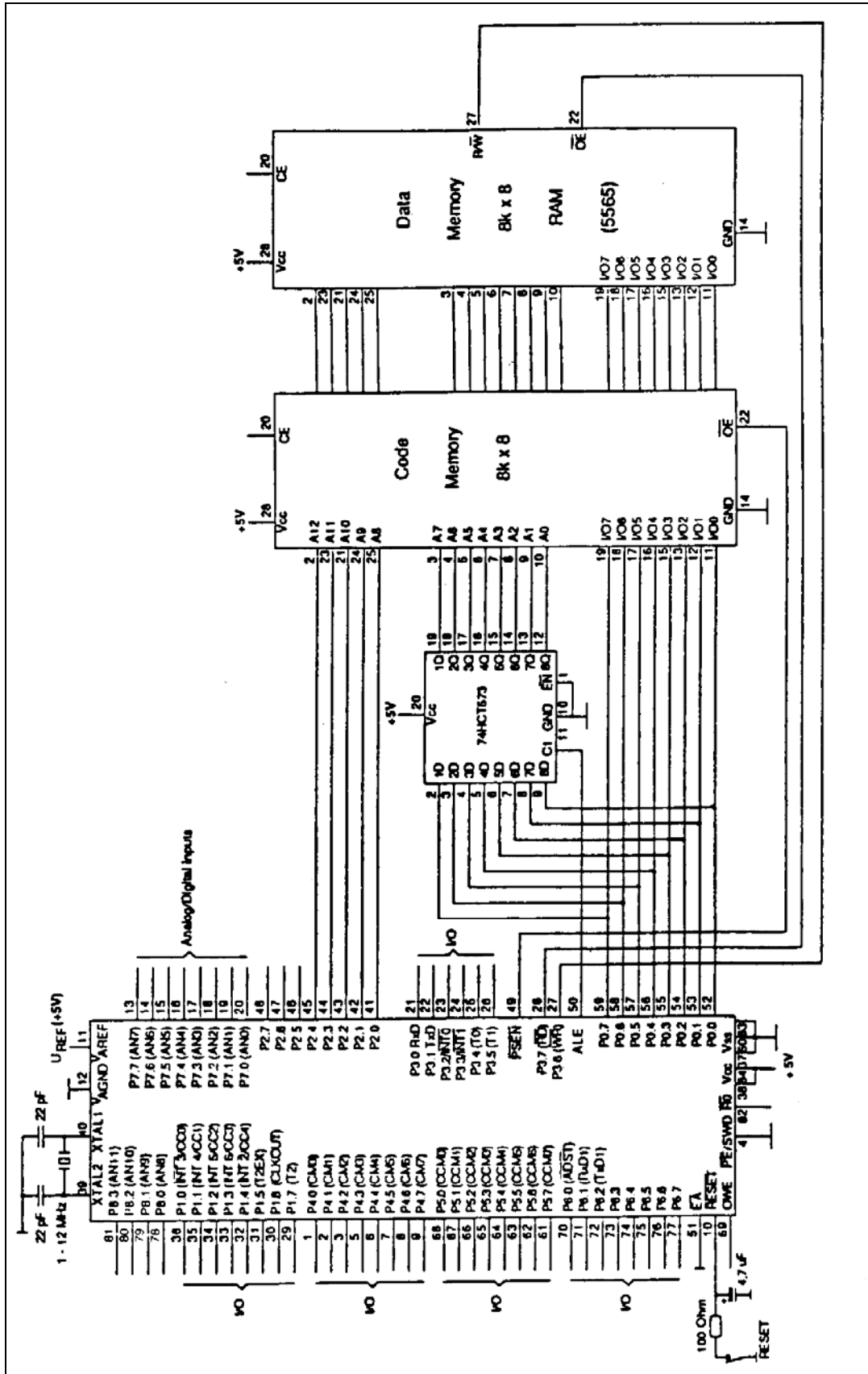
main()
{
    P1 = 0x00;
    warte (1000);
    P1 = 0xAA;
    warte (1000);
    P1 = 0x55;
    warte (1000);
    P1 = 0xAA;
    warte (1000);
    P1 = 0x00;
}
```

Der eingegebene Code wird unter dem Namen **PORT.C** abgespeichert und der Editor beendet.

Die Compilierung des C-Source Codes erfolgt mit dem Aufruf:

**C51 PORT.C**

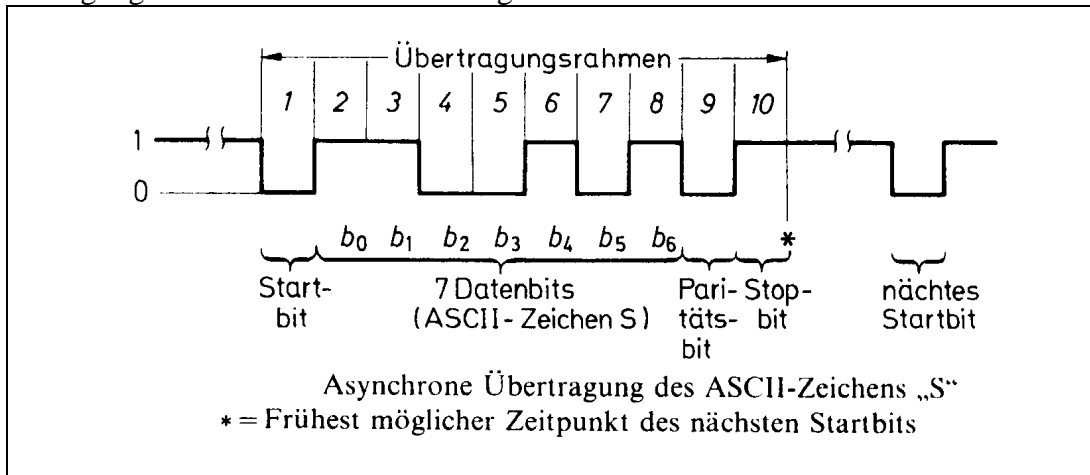
5 Standardbeschaltung des Microcontrollers 80C537



## 6 Serielle Schnittstelle(n)

Die RS232E-Schnittstelle dient zur asynchronen seriellen Punkt zu Punkt Datenübertragung. Ein Startbit eröffnet den Übertragungsrahmen eines Zeichens. Darauf folgen die 7 Bit des ASCII-Kodes, oder die 8 Bit bei erweitertem Zeichenvorrat, gefolgt von einem Paritäts- und einem Stopbit, welches den Übertragungsrahmen abschließt.

Ein Übertragungsrahmen sieht dabei wie folgt aus:



Das Startbit löst im Empfänger einen zeitlichen Ablauf aus, der die einzelnen Elemente des gesendeten Codes indiziert. Daher muß der Empfänger über die zeitliche Dauer der Elemente a priori informiert sein.

Als Erinnerung an den Fernschreiber ist der Ruhezustand des Schnittstellensignals die logische Eins. Der Linienstrom läßt erkennen, ob die Fernschreibleitung unterbrochen ist.

Das Paritätsbit dient der Fehlererkennung. Der Wert des Paritätsbits wird von der Zahl der Einsen im Datenwort bestimmt. Er ist 0 wenn bei gerader (even) Parität die Zahl der Einsen gerade ist oder bei ungerader (odd) Parität ungerade ist. Die gerade Parität kann daher auch als modulo 2-Summe der Datenbits erhalten werden.

Der Übertragungsrahmen (Frame) wird durch ein, im Sonderfall zwei, Stopbits abgeschlossen.

Die Übertragungsgeschwindigkeit wird in Baud angegeben. Ein Baud entspricht einer Rate von 1 bit/s (bps). Die Übertragungsgeschwindigkeiten sind genormt. Sie müssen beim Sender und Empfänger gleich eingestellt werden. Gängige Übertragungsgeschwindigkeiten sind

**75, 150, 300, 600, 1200, 2400, 4800, 9600, 19200 baud.**

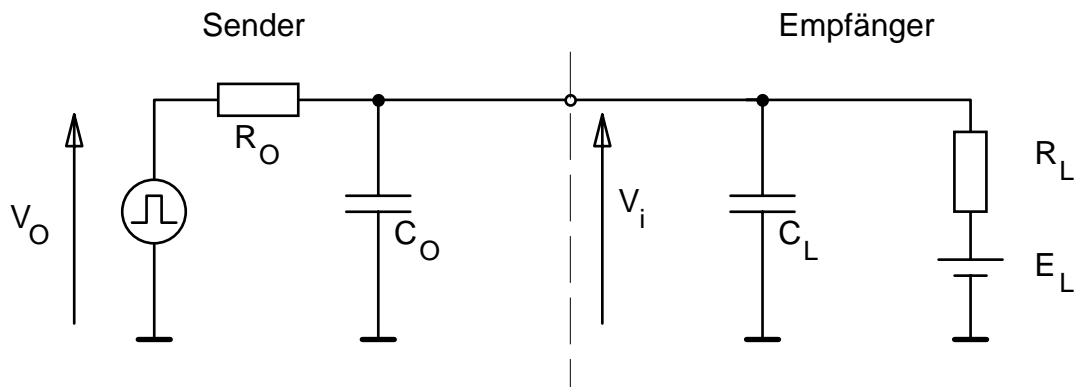
### Elektrische Eigenschaften der asynchronen seriellen Schnittstelle RS232E:

Die elektrischen Eigenschaften der Spannungsschnittstelle (CCITT-Empfehlung<sup>1</sup> V.24; EIA/TIA-Standard<sup>2</sup> RS232E (1990); DIN 66020) gültig für Datenraten < 20000 baud werden

<sup>1</sup> CCITT International Telegraph and Telephone Consultative Committee

<sup>2</sup> EIA Electronics Industries Association  
TIA Telecommunications Industry Association

anhand des Ersatzschaltbildes einer Zweidrahtschnittstelle (ohne Potentialtrennung) beschrieben.



Ersatzschaltung für die RS232E Schnittstelle nach EIA-Standard

Die Leerlaufspannung des Senders  $V_O$  soll kleiner als 25 V sein. Der Innenwiderstand des Senders  $R_O$  soll so bemessen sein, daß bei Kurzschluß in der Übertragungsleitung (auch zu benachbarten Adern) der Strom auf 0,5 A begrenzt wird. Im ausgeschalteten Zustand des Senders soll die Schnittstellenspannungen kleiner 2 V und  $R_O > 300 \Omega$  sein. Daher ist in manchen Senderschaltungen ein ohmscher Widerstand dieser Größe zur Strombegrenzung vorgesehen.

Die Ausgangskapazität  $C_O$  wird meist mit der Lastkapazität für den Sender  $C_L$  (Eigenkapazität des Empfängers und Kapazität der Leitung) gemeinsam betrachtet. Die Gesamtkapazität soll dabei nicht größer als 2500 pF sein. Dadurch wird die Länge der Leitung auf etwa 10 m begrenzt.

Der Lastwiderstand  $R_L$  (Eingangswiderstand des Empfängers) muß zwischen 3 und 7 k $\Omega$  liegen. Eine eventuelle innere Spannung des Empfängers (meist Basisemitterspannungen)  $< 2$  V sein.

Im Normalbetrieb ( $E_L=0$ ) muß  $V_O$  so bemessen sein, daß die Schnittstellenspannung  $V_i$  zwischen 5 und 15 V beträgt. Positive Werte für die logische Null, negative für die logische Eins. Das bedeutet implizit, daß für den Sender zwei Betriebsspannungen deutlich größer als  $\pm 5$  V notwendig sind. Die Verwendung der TTL Versorgungsspannungen als Betriebsspannung für die Schnittstelle stellt nur eine Behelfslösung dar.

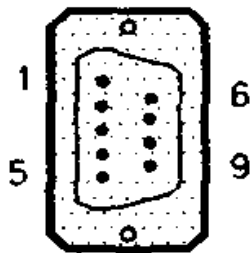
Der Empfänger soll Spannungen zwischen 3 und 25 V entsprechend dem Vorzeichen richtig erkennen können. Der Übergangsbereich zwischen -3 und +3 V dient zur Erkennung eines Senderausfalles oder einer Leitungsunterbrechung. Gute Empfängerbausteine signalisieren diesen Zustand getrennt vom Datenausgang. Das Schnittstellensignal soll den Übergangsbereich von einem Signalzustand zum anderen durchqueren, dabei seine Richtung nicht ändern und erst nach der nächsten Änderung des logischen Zustandes wieder in den Übergangsbereich eintreten. Die Zeit für das Passieren des Übergangsbereiches soll  $< 1$  ms sein, zusätzlich auch  $< 4\%$  der Dauer eines Signalelementes bei Daten- und Taktsignalen. Die zeitliche Änderung der Schnittstellenspannung muß immer  $< 30$  V/ms sein (Schwarzsender?).

Beispiel für einen Schnittstellenbaustein für die wechselseitige Pegelumsetzung zwischen 5 V CMOS- (auch TTL-) und RS232-Pegeln. Der eingebaute Spannungswandler erzeugt die Betriebsspannungen für die Schnittstellentreiber intern nach dem Prinzip der Ladungspumpe aus der 5 V Versorgung.

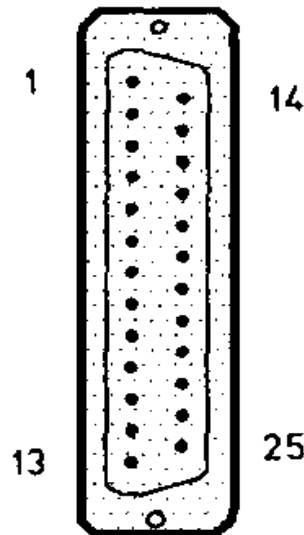
### 6.1 RS232-Schnittstelle an einem Personal Computer:

Bei der heute gängigen Schnittstelle zur Verbindung eines Personalcomputers mit einem beliebigen peripheren Gerät.

9-poliger Stecker



25-poliger Stecker



Steckerbelegung an einem Personalcomputer

9-pol.	25-pol.	Abk.	Bedeutung	
1	8	DCD	Carrier Detect	Ausgang
2	3	RxD	Received Data	Eingang
3	2	TxD	Transmitted Data	Ausgang
4	20	DTR	Data Terminal Ready	Eingang
5	7	GND	Signal Ground	
6	6	DSR	Data Set Ready	Ausgang
7	4	RTS	Request to Send	Ausgang
8	5	CTS	Clear to Send	Eingang
9	22	RI	Ring Indicator	Eingang

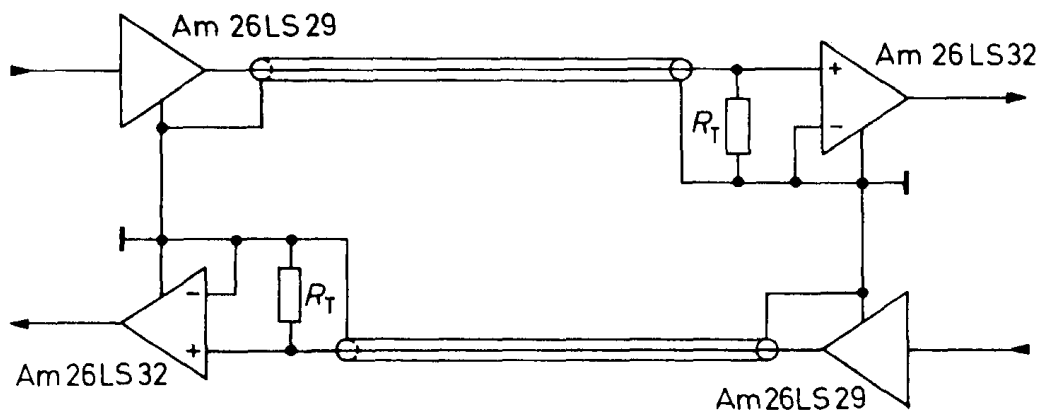
## 6.2 Andere serielle Schnittstellen:

### RS423 (V.10 in Europa):

Diese Schnittstelle hat einen kleineren Spannungshub (swing).

Schaltsschwellen:
Sender: +/- 3,6 Volt
Empfänger: +/- 0,2 Volt

Sie ist eine unsymmetrische Schnittstelle. Die Massen müssen nicht miteinander verbunden werden. Die RS423 stellt eine echte Punkt-zu-Punkt-Verbindung (wegen Koaxkabel) dar. Die Leitung muß mit dem Wellenwiderstand abgeschlossen sein. Die maximale Datenrate beträgt 300 kbaud bei 30m Leitungslänge und reduziert sich bis auf 15 kbaud bei 600m Leitungslänge.



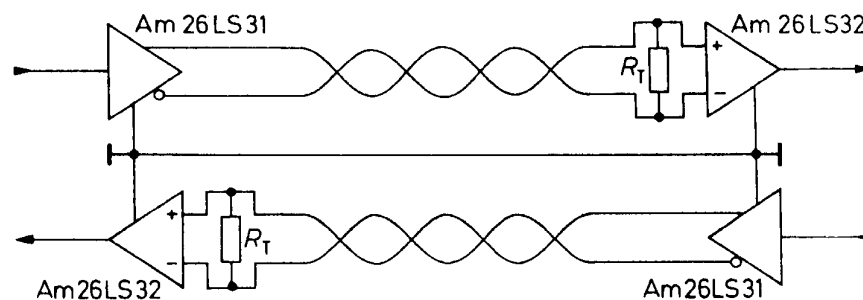
Anordnung einer RS 423-Schnittstelle

### RS422 (V.11 in Europa):

Schaltsschwellen:

Sender: +/- 3,6 Volt

Empfänger: +/- 0,2 Volt



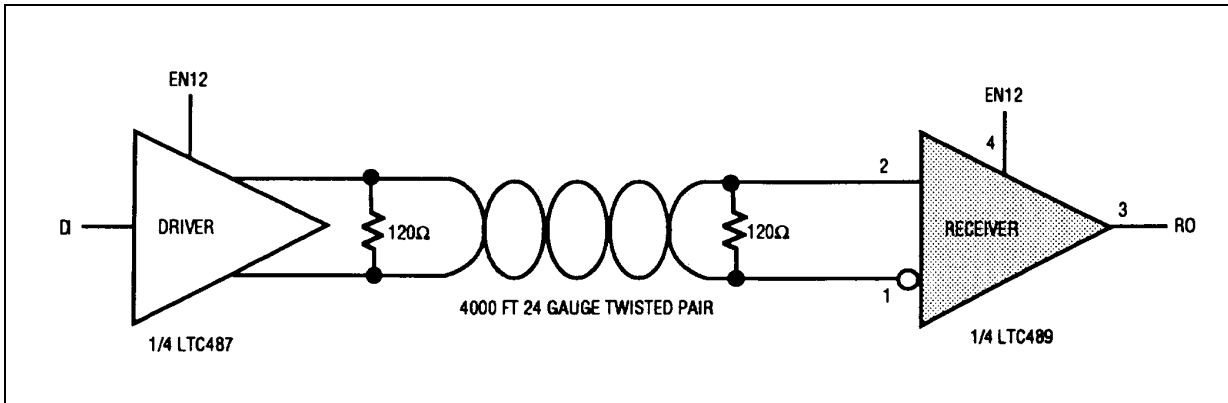
Anordnung einer RS 422-Schnittstelle

Leitung: verdrehtes Leitungspaar mit einer maximalen Länge von 1200m

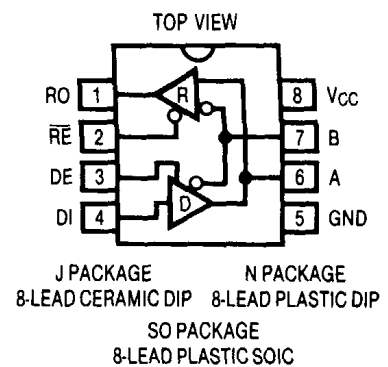


Zwei Eintakttreiber werden im Gegentakt angesteuert. Die Ausgangsspannungen sind zueinander komplementär. Gleichtaktstörungen werden dadurch "wegsubtrahiert". Übertragungsraten von maximal 2Mbaud bei einer maximalen Leitungslänge von 60m sind erreichbar. Bei größeren Leitungslängen reduziert sich die Datenrate bis auf 100kbaud.

**RS485:**



- Supply Voltage ( $V_{CC}$ ) ..... 12V
- Control Input Voltages .....  $-0.5V$  to  $V_{CC} + 0.5V$
- Driver Input Voltage .....  $-0.5V$  to  $V_{CC} + 0.5V$
- Driver Output Voltages .....  $\pm 14V$
- Receiver Input Voltages .....  $\pm 14V$
- Receiver Output Voltage .....  $-0.5V$  to  $V_{CC} + 0.5V$
- Operating Temperature Range
- LTC485I .....  $-40^{\circ}C \leq T_A \leq 85^{\circ}C$
- LTC485C .....  $0^{\circ}C \leq T_A \leq 70^{\circ}C$



## 6.3 Die seriellen Schnittstellen des 80C517

### 6.3.1 Serielle Schnittstelle 0

Die serielle Schnittstelle 0 arbeitet in vier Betriebsarten. Das dazugehörige Steuerregister ist S0CON.

**S0CON (98H), bitadressierbar**

MSB								LSB
SM0	SM1	SM20	REN0	TB80	RB80	TIO	RI0	
9FH	9EH	9DH	9CH	9BH	9AH	99H	98H	
Steuerregister der seriellen Schnittstelle 0 (Serial Interface 0 Control). Es enthält Steuerbits für die serielle Schnittstelle 0. Reset-Wert: 00H								
Bitsymbol		Funktion						
SM0	SM1	Serieller Modus Bit 0/1						
0	0	Modus 0: Schieberegister-Modus						
0	1	Modus 1: 8-Bit-UART, flexible Baudrate						
1	0	Modus 2: 9-Bit-UART, feste Baudrate						
1	1	Modus 3: 9-Bit-UART, flexible Baudrate						
SM20		Aktiviert den Multiprozessor-Kommunikationsbetrieb in Modus 2 und 3. Mit SM20 = 1 in Modus 2 und 3 wird RI0 nicht gesetzt, wenn das 9. empfangene Datenbit 0 ist. Mit SM20 = 1 in Modus 1 wird RI0 nicht gesetzt, wenn kein gültiges Stoppbit empfangen wurde. In Modus 0 muß SM20 immer 0 sein.						
REN0		Empfängeraktivierung (Receive Enable). wenn REN0 auf 1 gesetzt ist, ist serieller Empfang möglich, mit REN0 = 0 ist serieller Empfang abgeschaltet. Muß durch Software gesetzt/rückgesetzt werden.						
TB80		Sende-bit 8 (Transmitter Bit 8). Dies ist das 9. Datenbit, das in Modus 2 und 3 ausgesendet wird. Muß durch Software gesetzt/rückgesetzt werden.						
RB80		Empfangsbit 8 (Receiver Bit 8). Dies ist das 9. Datenbit, das in Modus 2 und 3 empfangen wird. In Modus 1 (mit SM20 = 0) ist RB80 das empfangene Stoppbit. In Modus 0 wird RB80 nicht benutzt.						
TIO		Sende-Interrupt (Transmitter Interrupt). Dies ist das Interrupt-Request-Flag für den Sender. TIO wird von der Hardware in Modus 0 am Ende des 8. Datenbits gesetzt, in den anderen Modi bei Beginn des Stoppbits. TIO muß durch Software rückgesetzt werden.						
RI0		Empfänger-Interrupt (Receiver Interrupt). Dies ist das Interrupt-Request-Flag für den Empfänger. RI0 wird in Modus 0 von der Hardware am Ende des 8. Datenbits gesetzt, in den anderen Modi während des Stoppbits. RI0 muß durch Software rückgesetzt werden.						

Bild 8-2: Special-Function-Register S0CON (98H)

**PCON (87H), nicht bitadressierbar**

MSB						LSB	
SMOD	PDS	IDLS	SD	GF1	GF0	PDE	IDLE
Steuerregister für die Stromaufnahme (Power Control Register). Es enthält Steuerbits für die Stromaufnahme und die serielle Schnittstelle 0. Reset-Wert: 00H							
Bitsymbol	Funktion						
SMOD	Bei gesetztem Bit SMOD verdoppelt sich die Baudrate der seriellen Schnittstelle 0 in den Modi 1,2 und 3						

Bild 8-3: Special-Function-Register PCON (87H)

**ADCON0 (D8H), bitadressierbar**

MSB						LSB	
BD	CLH	ADEX	BSY	ADM	MX2	MX1	MX0
DFH	DEH	DDH	DCH	DBH	DAH	D9H	D8H
A/D-Wandler-Steuerregister 0 (A/D Converter Control Register 0). Es enthält Steuerbits für den A/D-Wandler und die serielle Schnittstelle 0. Reset-Wert: 00H.							
Bitsymbol	Funktion						
BD	Freigabebit für den Baudratengenerator der seriellen Schnittstelle 0. Wenn es gesetzt ist, wird die Baudrate in den Modi 1 und 3 vom Baudratengenerator der seriellen Schnittstelle 0 abgeleitet. Damit können die Standardbaudraten 4,8 Kbaud bzw. 9,6 Kbaud bei 12 MHz Oszillatorfrequenz erzeugt werden.						

Bild 8-4: Special-Function-Register ADCON0 (D8H)

In Bild 8-7 sind einige repräsentative Betriebsfälle bei Baudratenerzeugung mit Timer 1 herausgegriffen.

Baudrate in Modus 1 u. 3	f <sub>osz</sub> [MHz]	SMOD	Timer 1		
			C/T#	Modus	Reload
62,5 Kbaud	12,000	1	0	2	FFH
19,2 Kbaud	11,059	1	0	2	FDH
9,6 Kbaud	11,059	0	0	2	FDH
4,8 Kbaud	11,059	0	0	2	FAH
2,4 Kbaud	11,059	0	0	2	F4H
1,2 Kbaud	11,059	0	0	2	E8H
110 Baud	6,000	0	0	2	72H
110 Baud	12,000	0	0	1	FEEBH

Bild 8-7: Baudratenerzeugung mit Timer 1

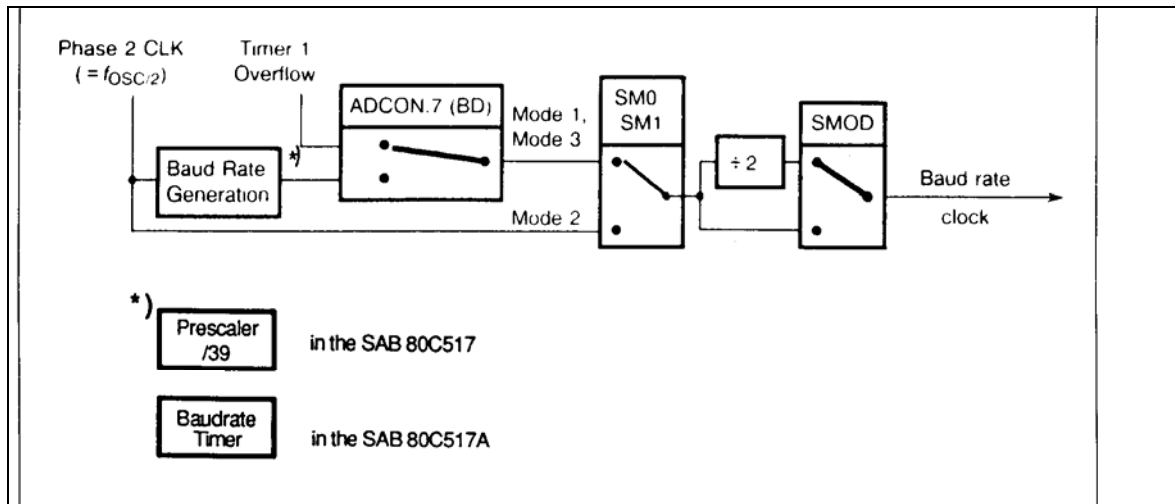


Bild 8-8: Baudratenerzeugung für die serielle Schnittstelle 0

Baudratentakt abgeleitet von	Modus	Baudrate
Timer 1 in Modus 1 (siehe Bild 7-7)	1 u. 3	$\frac{2^{SMOD}}{2} * \frac{1}{16} * (\text{Timer 1-Überlaufrate})$
Timer 1 in Modus 2	1 u. 3	$\frac{2^{SMOD}}{2} * \frac{1}{16} * \frac{f_{osz}}{12 * (256 - (TH1))}$
Oszillator	2	$\frac{2^{SMOD}}{2} * \frac{1}{16} * \frac{f_{osz}}{2}$
Baudratengenerator im 80C517	1 u. 3	$\frac{2^{SMOD}}{2} * \frac{f_{osz}}{1250}$
Baudratengenerator im 80C517A	1 u. 3	$\frac{2^{SMOD}}{2} * \frac{1}{32} * \frac{f_{osz}}{2^{10} - S0REL}$ (mit S0REL = S0RELH.1-0, S0RELL.7-0)

Bild 8-9: Formeln für Baudraten für die serielle Schnittstelle 0

### 6.3.2 Serielle Schnittstelle 1

Die serielle Schnittstelle 1 hat lediglich asynchrone Betriebsarten und kann ähnlich wie die serielle Schnittstelle 0 in den Modi 8- oder 9-bit-Datenübertragung arbeiten. Das dazugehörige SFR ist S1CON.

#### S1CON (9BH), nicht bitadressierbar

MSB						LSB	
SM	-	SM21	REN1	TB81	RB80	TI1	RI1
Steuerregister der seriellen Schnittstelle 1 (Serial Interface 1 Control). Es enthält Steuerbits für die serielle Schnittstelle 1. Reset-Wert: 0x00 0000B							
Bitsymbol	Funktion						
SM	SM = 0: serielle Betriebsart A; 9-Bit UART SM = 1: serielle Betriebsart B; 8-Bit UART						
SM21	Aktiviert den Multiprozessor-Kommunikation-Betrieb in Modus A. Mit SM21 = 1 wird RI1 nicht gesetzt, wenn das 9. empfangene Datenbit 0 ist. Mit SM21 = 1 in Modus B wird RI1 nicht gesetzt, wenn kein gültiges Stoppbit empfangen wurde.						
REN1	Receive Enable; Enable für den Empfänger; wenn auf 1 gesetzt, ist serieller Empfang möglich, mit REN1 = 0 ist serieller Empfang abgeschaltet. Es muß durch Software gesetzt/rückgesetzt werden.						
TB81	Transmitter Bit 8; Dies ist das 9. Datenbit, das in Modus A ausgesendet wird. Es muß durch Software gesetzt/rückgesetzt werden.						
RB81	Receiver Bit 8; Dies ist das 9. Datenbit, das in Modus A empfangen wird. In Modus B (mit SM21 = 0) ist RB81 das empfangene Stoppbit.						
TI1	Transmitter Interrupt; Dies ist das Interrupt-Request-Flag für den Sender. TI1 wird von der Hardware bei Beginn des Stoppbits gesetzt. TI1 muß durch Software rückgesetzt werden.						
RI1	Receiver Interrupt; Dies ist das Interrupt-Request-Flag für den Empfänger. TI1 wird während des Stoppbits gesetzt. RI1 muß durch Software rückgesetzt werden.						

Bild 8-10: Special-Function-Register S1CON (9BH)

#### S1BUF (9CH), nicht bitadressierbar

MSB						LSB	
S1BUF.7	S1BUF.6	S1BUF.5	S1BUF.4	S1BUF.3	S1BUF.2	S1BUF.1	S1BUF.0
Empfangs- und Sendepuffer der seriellen Schnittstelle 1 (Serial Interface 1 Buffer). Ein Schreibbefehl auf S1BUF lädt den Sendepuffer und startet eine Übertragung. Ein Lesebefehl auf S1BUF greift auf das physikalisch getrennte Empfangsregister zu. Reset-Wert: XXH							

Bild 8-11: Special-Function-Register S1BUF (9CH)

Baudraten:

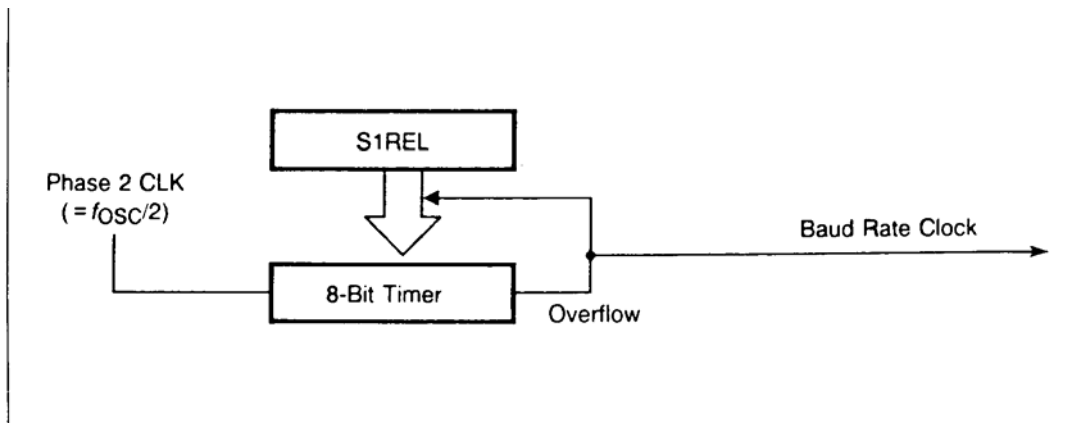


Bild 8-12: Baudratengenerator für serielle Schnittstelle 1 im 80C517

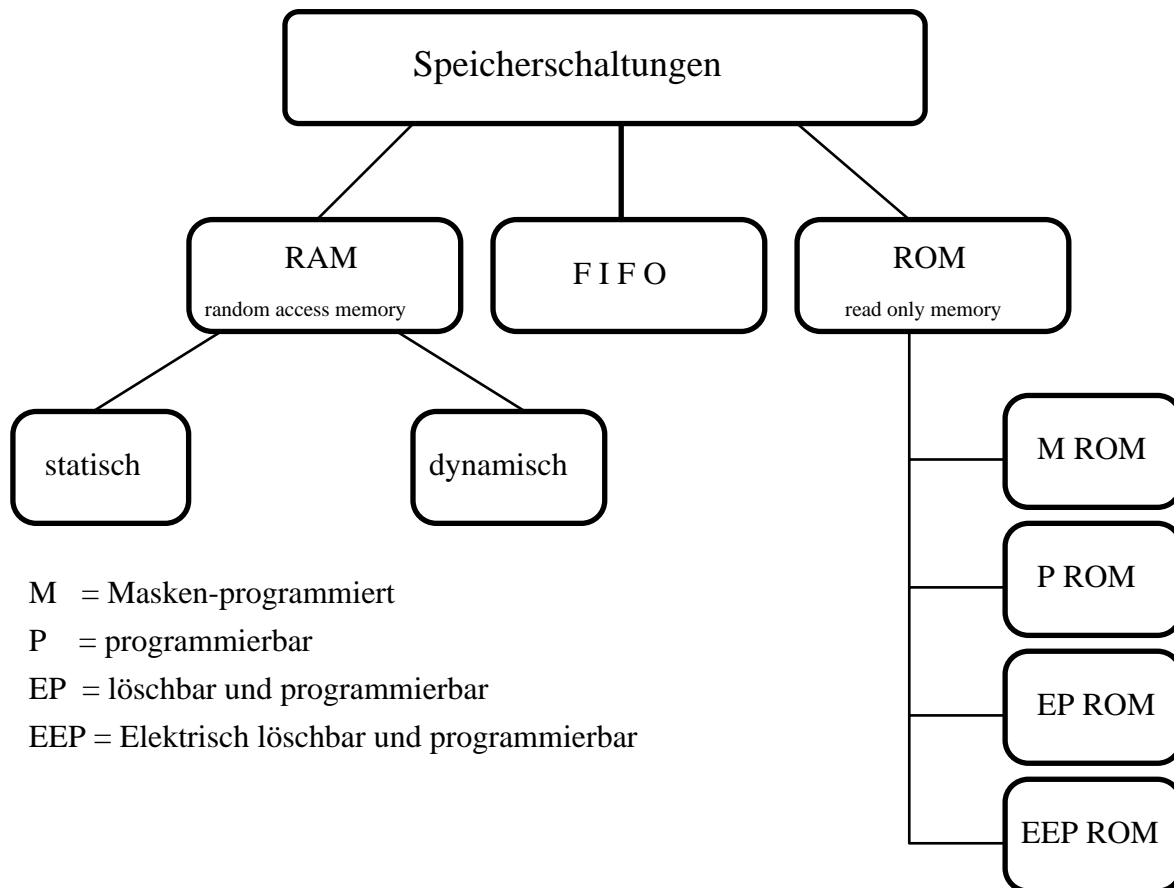
**S1REL (9DH), nicht bitadressierbar**

MSB							LSB
S1REL.7	S1REL.6	S1REL.5	S1REL.4	S1REL.3	S1REL.2	S1REL.1	S1REL.0
Reload-Register (8 bit) für den Baudratengenerator der seriellen Schnittstelle 1 im 80C517. Nach Reset ist die Baudrate 1,5 Kbaud bei 12 MHz Oszillatorfrequenz. Reset-Wert: 00H							

## 7 Speicher

### 7.1 Allgemeines

Halbleiter-Speicherschaltungen zum Schreiben und Lesen von Daten ("random access memory") lassen sich in zwei unterschiedliche Hauptgruppen einteilen. Dies sind der statische Schreib-Lese-Speicher, der eine FF-Struktur besitzt, und der dynamische Schreib-Lese-Speicher mit einem integrierten Speicherkondensator, dessen Ladung zyklisch aufgefrischt werden muß ("refresh").



Statische Schreib-Lese-Speicher, SRAM genannt ("static random access memory" = Speicher mit wahlfreiem Zugriff), weisen je nach Technologie zwei bis sechs Transistoren pro Speicherzelle auf. Bei der 6-Transistorzelle werden zwei Transistoren als Ersatz für die Lastwiderstände verwendet, weil Widerstände in integrierten Schaltungen einen großen Platzbedarf aufweisen. Der Begriff "Speicher mit wahlfreiem Zugriff" bedeutet, daß man auf jedes Wort zu jeder Zeit zugreifen kann, im Unterschied zum Schieberegisterspeicher (siehe später), bei dem die Daten nur in derselben Reihenfolge ausgelesen werden können, in der sie eingeschrieben wurden. Die Bedeutung von Schieberegisterspeicher ist in letzter Zeit gesunken. Deshalb ist der Begriff RAM zur generellen Bezeichnung für Schreib-Lese-Speicher geworden. Das ist insofern etwas irreführend, als die ROMs ebenfalls einen wahlfreien Zugriff auf jedes Datenwort haben.

Die dynamischen Schreib-Lese-Speicher (DRAM = "dynamic random access memory") benötigen zusätzlich zu den Speicherzellen aufwendige Hilfsschaltungen ("Refresh"-Verstärker und Ladungspumpen zur Erzeugung interner Hilfsspannungen), die einen nicht zu vernachlässigenden Flächenbedarf haben.

Ausgehend von gleicher Speicher-Bausteinfläche und Technologie liegt die Speicherkapazität bei DRAMs ca. um den Faktor vier höher als bei SRAMs.

ROM ist die Abkürzung des englischen Begriffs "Read Only Memory", d.h. Festwertspeicher. Damit werden Speicher ICs bezeichnet, die ihre Daten auch dann behalten, wenn sie ohne jede Versorgungsspannung, also auch ohne Hilfsbatterie, betrieben werden. Im Normalbetrieb werden sie nur gelesen. Die Speicherung der Daten erfordert in der Regel spezielle Geräte. Man bezeichnet den Speichervorgang in diesem Fall als Programmierung (siehe später).

## 7.2 Speicherorganisation:

Ein Speicherbaustein hat für seine **interne Datenorganisation** Anschlüsse für:

- o Adresse (**n** für  $2^n$  Adressen (je nach Speichertiefe)
- o Daten (1, 4, 8, 16 bit )

**SRAM-Beispiel:** SRM-20256LC-10 (32KBx8)

SRM-20... Firmenbezeichnung  
 256 ... 256 kbit Speichertiefe  
 LC ... CMOS  
 10 ... Zugriffszeit = 100ns  
 32 KByte ... Speichertiefe

Die Organisation für diesen Speicher ist 8 bit Datenbreite (Byte-Organisation)

8 bit Datenbreite = **BYTE**

16 bit Datenbreite = **WORD**

Dezimal

A D R E S S E

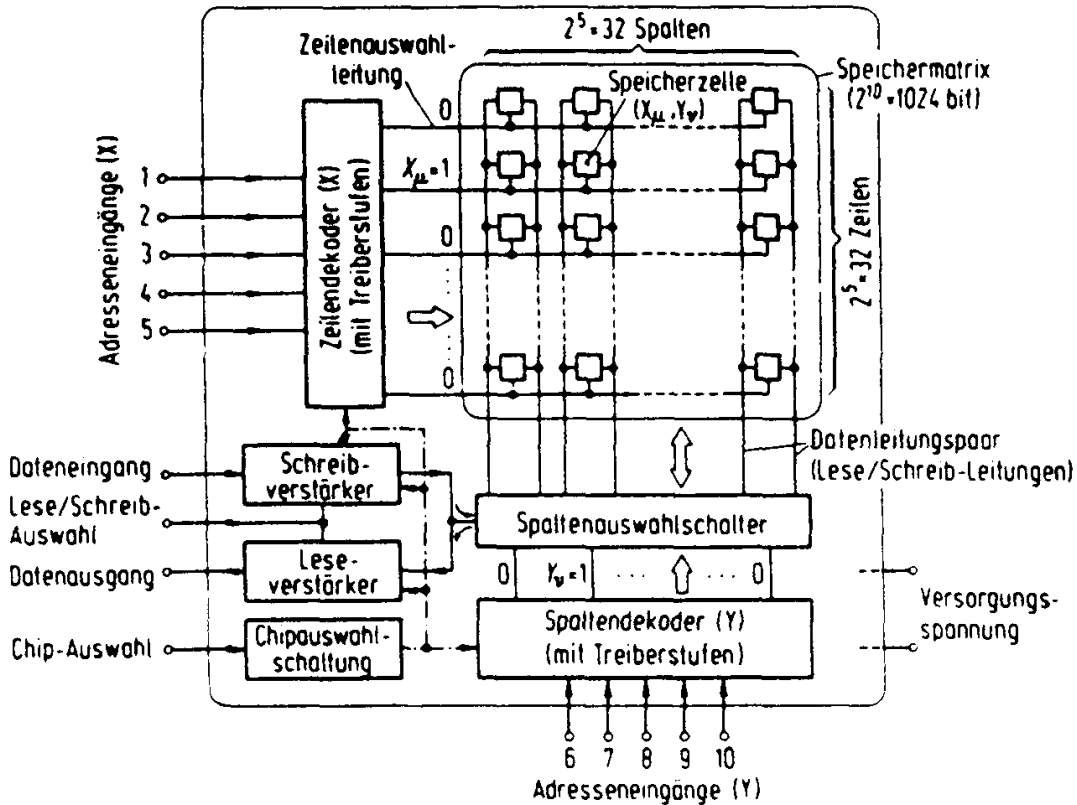
D A T E N W O R T

Dezimal	$2^{14}$	$2^{13}$	...	$2^1$	$2^0$	MSB	D <sub>6</sub>	...	D <sub>1</sub>	LSB
	A <sub>14</sub>	A <sub>13</sub>		A <sub>1</sub>	A <sub>0</sub>	D <sub>7</sub>				D <sub>0</sub>
0	0	0		0	0					
1	0	0		1	0					
2	0	0		1	1					
3	0	0		0	0					
...										
...										
...										
32765	1	1		0	0					
32766	1	1		1	0					
32767	1	1		1	1					



### 7.3 Statische RAMs:

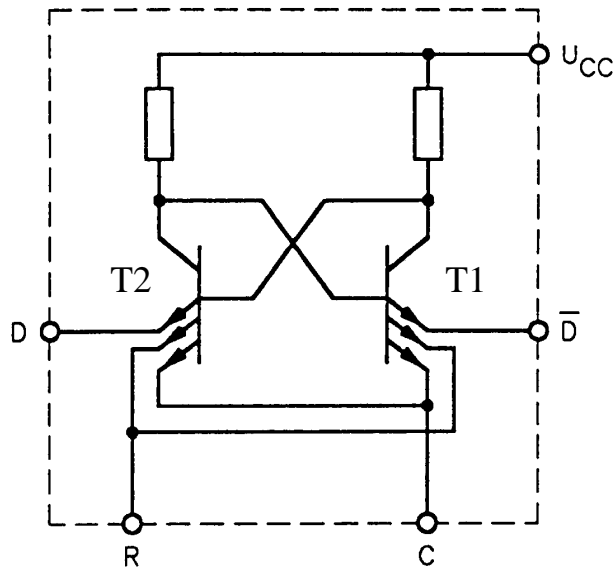
Ein RAM ist ein Speicher, bei dem man nach Vorgabe einer Adresse Daten abspeichern und unter dieser Adresse wieder auslesen kann (wahlfreier Zugriff). Aus technologischen Gründen werden die einzelnen Adressen nicht linear, sondern in einer quadratischen Matrix angeordnet. Zur Auswahl einer bestimmten Speicherstelle wird die Adresse A von einem Spalten- bzw. Zeilendecoder dekodiert.



Beispiel für eine Organisation eines 1024x1bit Speicherbausteins

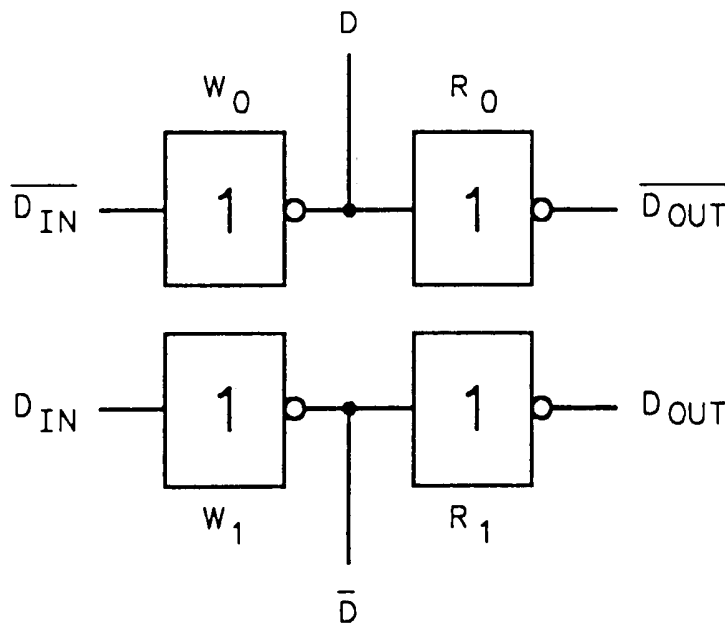
#### 7.3.1 TTL-Speicherzelle:

Die Auswahl jeder Speicherzelle erfolgt über die jeweilige Zeile (row) und der jeweiligen Spalte (column). Jeder Transistor (Multiemittertransistor) verfügt über eine weitere Emitterleitung ( $\bar{D}$  bzw. D), die alle mit entsprechenden Schreib- und Leseverstärkern verbunden sind. Über diese Leitungen werden Daten ein- oder ausgegeben. Die Adressierung des Speichers erfolgt durch die Dekodierung der Adressen A1 bis A10. Die Betriebsart (Schreiben oder Lesen) wird üblicherweise durch ein Steuersignal  $R/\bar{W}$  festgelegt. Beim Schreiben werden die Schreibverstärker W0 und W1 (siehe nächstes Bild) freigegeben und die Leseverstärker R0 und R1 gesperrt. Beim Lesen werden dann entsprechend die Leseverstärker freigegeben und die Schreibverstärker gesperrt.



TTL-Speicherelement

Die FF-Zelle wird angesprochen, wenn die Leitungen R und C hi-Pegel führen. Dadurch fließen die Emitterströme  $I_{EX}$  beim leitenden und  $I_{EY}$  beim gesperrten Transistor zu den angeschlossenen Leseverstärkern. Bei allen anderen FF-Zellen fließen die Emitterströme ausschließlich durch die Schalttransistoren der Dekoder ab.



Anordnung der Schreib/Lese-Verstärker bei statischen TTL-RAMs

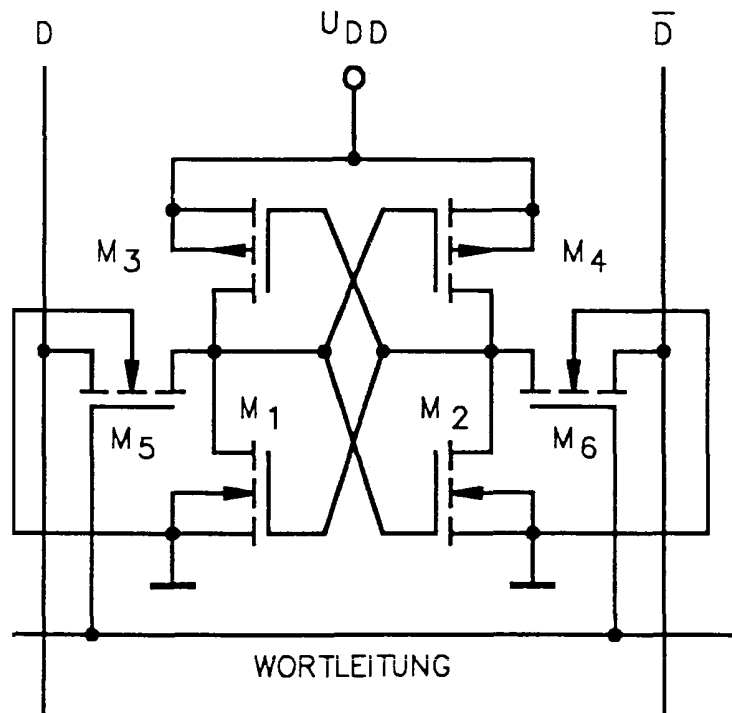
Durch Aktivierung der Leseverstärker kann ein Datum  $D_{OUT}$  oder  $\overline{D_{OUT}}$  ausgegeben werden.

Für den Schreibvorgang werden die Schreibverstärker aktiviert und in Abhängigkeit des Eingangsdatums die eine Emitterleitung auf hi- und die andere auf lo-Pegel gelegt. Ausgehend von einem gesperrten Transistor T2 und leitendem T1 kippt dieses FF, wenn am Emitter von T2 (Anschluß D) ein lo-Potential anliegt. Durch den hi-Pegel an allen drei Emitteranschlüssen des

Transistors T1 sperrt dieser. Dadurch stellt sich ein hohes Basispotential an T2 ein, folglich leitet jetzt dieser Transistor.

Bipolare TTL-RAMs wurden in der Vergangenheit wegen der kurzen Zugriffszeiten (<100ns) bei schnellen Schaltungen eingesetzt. Durch den großen Flächenbedarf der Bipolartransistoren und die hohe Verlustleistung ist die Speicherkapazität begrenzt (<4kbit).

### 7.3.2 CMOS-Speicherzelle:

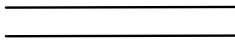
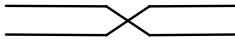

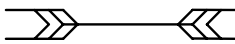


CMOS - RAM mit sechs Transistoren

Bei einem CMOS-RAM findet man im allgemeinen sechs Transistor-Speicherzellen. Die Speicherzelle wird durch die beiden rückgekoppelten CMOS-Inverter mit den Transistoren M1 bis M4 gebildet. Wenn M1 und M4 leiten, sperren M2 und M3. Durch die anliegende Adresse wird vom Zeilendekoder eine Zeile, die als "Wortleitung" bezeichnet wird, aktiviert. Die von der Wortleitung mit einem hi-Pegel angesteuerten Schalttransistoren M5 und M6 legen beim Lesen die Drainpotentiale der N-Kanal-Transistoren auf die beiden Spaltenleitungen, die als Bitleitungen (D und  $\bar{D}$ ) bezeichnet werden. Diese Bitleitungen werden einem Spaltenmultiplexer zugeführt, der das durch die Adresse selektierte Bitleitungspaar auf den Leseverstärker legt und zum Ausgang führt.

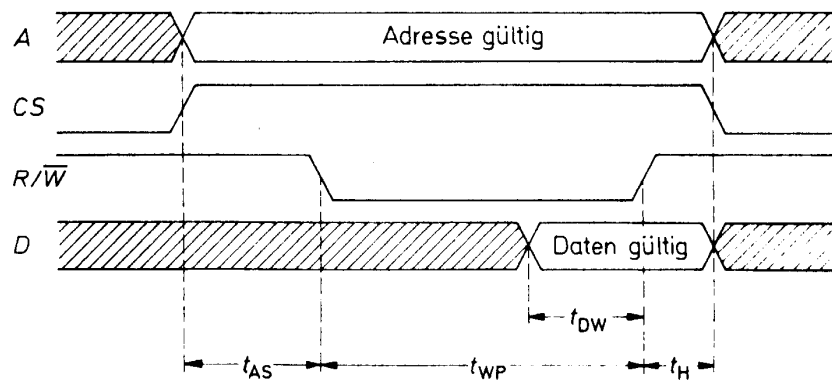
Beim Schreibvorgang arbeitet der Multiplexer als Dekoder/Demultiplexer. Das Datum liegt in identischer und invertierter Form an den beiden selektierten Bitleitungen. Die gewünschte Speicherzelle wird durch Dekodierung der Wortleitung selektiert. Liegt z.B. an der Bitleitung D ein lo-Pegel und sperrt M1, so erzwingt der eingeschaltete Transistor M5, daß der zuvor leitende Transistor M2 sperrt und M1 leitet.

**7.3.3 Symbole für Impuls- bzw. Zeitdiagramme:**

SYMBOL	EINGANG	AUSGANG
	Muß konstant H oder L sein	Ist konstant H oder L
	Wechsel ist erlaubt	Wechsel findet statt
	H/L - Wechsel ist erlaubt	H/L - Wechsel findet statt
	L/H - Wechsel ist erlaubt	L/H - Wechsel findet statt
	Ohne Bedeutung	Unbekannt oder wechselnd
	-----	Tristate (hochohmiger Zustand)

**7.3.4 Zeitbedingungen:**

Um die einwandfreie Funktion eines Speichers zu gewährleisten, müssen einige zeitliche Randbedingungen eingehalten werden. Um zu verhindern, daß die Daten in eine falsche Zelle geschrieben werden, darf der Schreibbefehl erst eine gewisse Zeit nach der Adresse angelegt werden. Diese Zeit heißt Address Setup Time  $t_{AS}$ . Die Dauer eines Schreibimpulses darf den Minimalwert  $t_{WP}$  (Write Pulse Width) nicht unterschreiten.



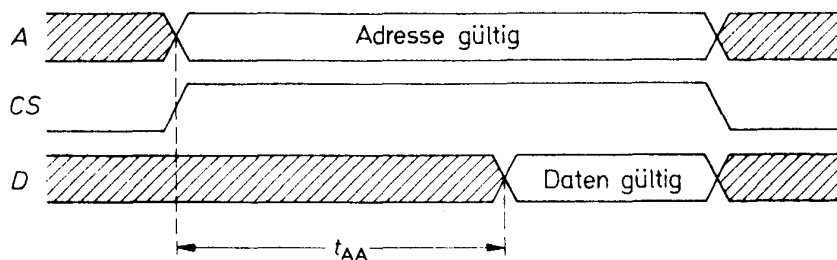
Zeitlicher Ablauf eines Schreibvorganges

- $t_{AS}$  : Address Setup Time
- $t_{WP}$  : Write Pulse Width
- $t_{DW}$  : Data Valid to End of Write Time
- $t_H$  : Hold Time

Die Daten werden am Ende des Schreibimpulses eingelesen. Sie müssen eine bestimmte Mindestzeit vorher gültig, d.h. stabil sein. Diese Zeit heißt  $t_{DW}$  (Data Valid to End of Write). Bei vielen Speichern müssen die Daten bzw. Adressen noch eine gewisse Zeit  $t_H$  nach dem Ende des Schreibimpulses anliegen (Hold Time). Wie man aus dem Bild erkennt, ergibt sich für die Durchführung eines Schreibvorganges die Zeit:

Dies Summe dieser 4 Zeiten wird als Schreib-Zyklus-Zeit (Write Cycle Time) bezeichnet.

Der Lesevorgang ist im nächsten Bild dargestellt. Nach dem Anlegen der Adresse muß man die Zeit  $t_{AA}$  abwarten, bis die Daten am Ausgang gültig sind. Diese Zeit heißt Lese-Zugriffszeit (Adress Access Time) oder einfach Zugriffszeit.



Zeitlicher Ablauf eines Lesevorganges  
 $t_{AA}$ : Address Access Time

### 7.3.5 Beispiel aus dem Datenbuch:

Lesezyklus:

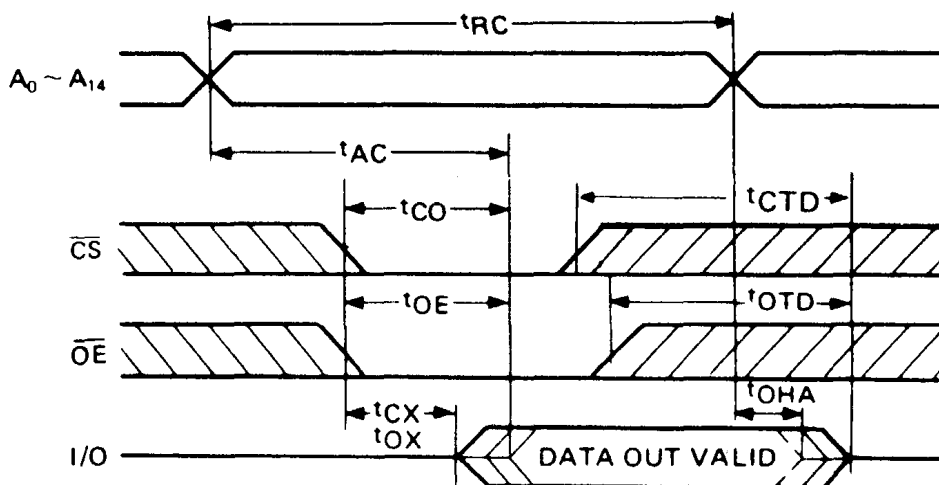
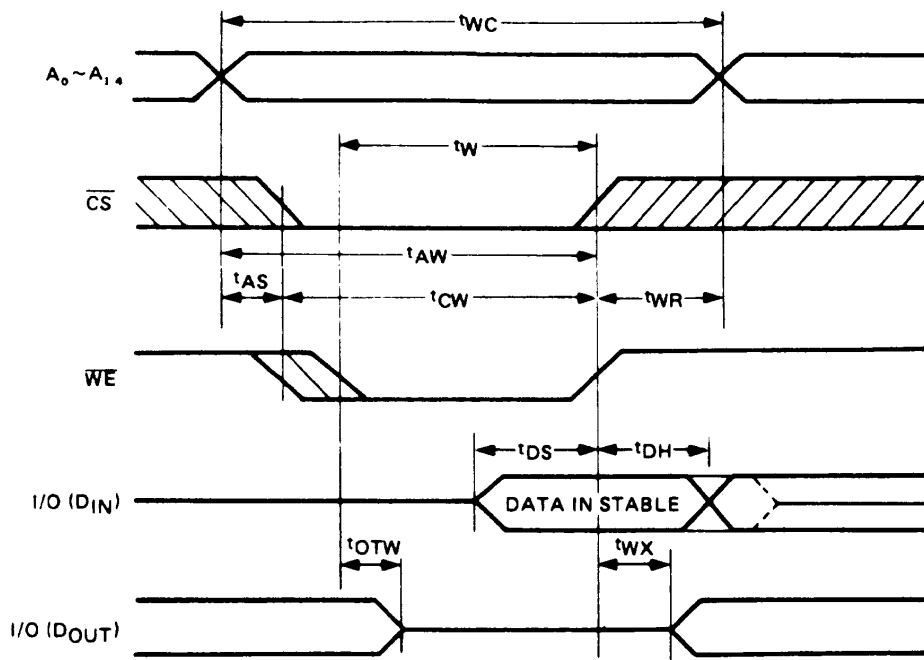


Tabelle für einen Lesezyklus:

Parameter	Symbol	MSM51257ALL-85		MSM51257ALL-10		MSM51257ALL-12		Unit
		Min.	Max.	Min.	Max.	Min.	Max.	
Read Cycle Time	$t_{RC}$	85		100		120		ns
Address Access Time	$t_{AC}$		85		100		120	ns
Chip Enable Access Time	$t_{CO}$		85		100		120	ns
Output Enable to Output Valid	$t_{OE}$		45		50		60	ns
Chip Selection to Output Active	$t_{CX}$	10		10		10		ns
Output Hold Time From Address Change	$t_{OHA}$	5		10		10		ns
Output 3-state from Output Disable	$t_{OTD}$		30		35		35	ns
Output 3-state from Chip Deselection	$t_{CTD}$		30		35		35	ns
Output Enable to Output Active	$t_{OX}$	5		5		5		ns

Schreibzyklus:



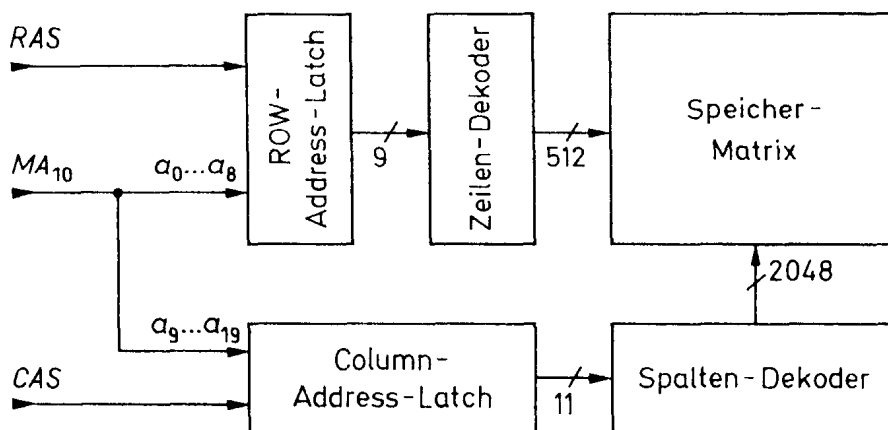
Item	Symbol	MSM51257ALL-85		MSM51257ALL-10		MSM51257ALL-12		Unit
		Min.	Max.	Min.	Max.	Min.	Max.	
Write Cycle Time	t <sub>WC</sub>	85		100		120		ns
Address to Write Setup Time	t <sub>AS</sub>	0		0		0		ns
Write Time	t <sub>W</sub>	70		75		90		ns
Write Recovery Time	t <sub>WR</sub>	5		10		10		ns
Data Setup Time	t <sub>DS</sub>	40		40		50		ns
Data Hold from Write Time	t <sub>DH</sub>	0		0		0		ns
Output 3-State from Write	t <sub>OTW</sub>	0	30	0	35	0	35	ns
Chip Selection to End of Write	t <sub>CW</sub>	75		90		100		ns
Address Valid to End of Write	t <sub>AW</sub>	75		90		100		ns
Output Active from End of Write	t <sub>WX</sub>	5		5		5		ns

### 7.4 Dynamische RAMs:

Dynamische RAMs weisen sehr hohe Speicherkapazitäten auf. Die hohe Integrationsdichte wird durch einen einfachen Datenspeicher, bestehend aus einer MOS-Kapazität von einigen Femtofarad Größe und einem Schalttransistor für die Bitleitung erreicht ("1 Transistorspeicher"). Die aufgebrauchte Ladung in der Gate-Source-Kapazität dient zur Darstellung von einem bit. Allerdings bleibt diese Ladung nur für kurze Zeit erhalten. Deshalb muß der Kondensator regelmäßig (ca. alle 2..8ms) nachgeladen werden. Diesen Vorgang bezeichnet man als refresh, die Speicher als dynamische RAMs.

Diesem Nachteil stehen mehrere Vorteile gegenüber. Auf derselben Chipfläche, bei derselben Stromaufnahme und mit denselben Kosten läßt sich mit dynamischen RAMs ungefähr die vierfache Speicherkapazität realisieren.

Um Anschlüsse einzusparen, wird die Adresse bei dynamischen RAMs in zwei Schritten eingegeben und im IC zwischengespeichert.



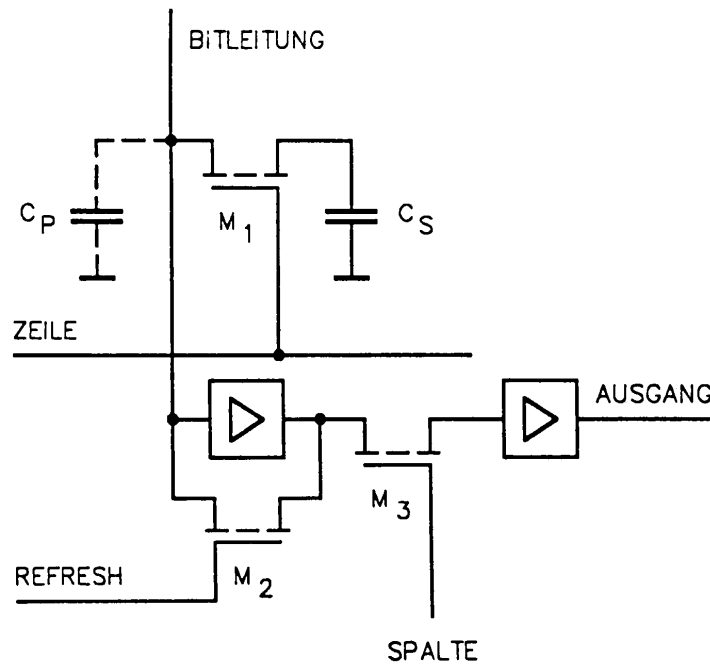
Adreß-Dekodierung in einem dynamischen 1 Mbit-Speicher

RAS: Row Address Strobe (gleichzeitig Chip Enable)

CAS: Column Address Strobe

Im ersten Schritt werden die Adressbits  $a_0 \dots a_8$  mit dem RAS-Signal in das Row-Adress-Latch gespeichert, und gleichzeitig das Bit  $a_9$  im Column-Adress-Latch. Im zweiten Schritt werden die Adressbits  $a_{10} \dots a_{19}$  mit dem CAS-Signal in das Column-Adress-Latch geladen. Dadurch ist es möglich, einen 1Mbit-Speicher in einem 18poligen Gehäuse unterzubringen.

### 7.4.1 Speicherzellenaufbau:



Ist der Speicherkondensator  $C_S$  geladen, entspricht dies einer logischen "1" (hi-Pegel), ein nichtgeladener Kondensator einer logischen "0" (lo-Pegel). Der Ladungsunterschied zwischen einem geladenen und einem ungeladenen Kondensator liegt bei etwa  $10^6$  Elektronen.

Über den Adreßdekoder wird nun eine Zeile (=Wortleitung) selektiert. Der hi-Pegel auf der Wortleitung steuert den Schalttransistor  $M_1$  durch und entlädt  $C_S$  wegen der hohen Parasitärkapazität  $C_P$  ( $C_P$  ca.  $10 \cdot C_S$ ). Dieser Ladungsausgleich bewirkt, daß die gespeicherte Information einer ganzen Zeile zerstört wird. Über einen Leseverstärker läßt sich das Signal an  $C_P$  verstärken.

Zum Auslesen wird über die Zeilenleitung der Transistor  $M_1$  durchgeschaltet. Der Ausgang des Leseverstärkers wird durch Schalten des selektierten Spaltentransistors  $M_3$  auf den Ausgangsverstärker geführt. Wegen der Ladungszerstörung ist eine Rückkopplung über  $M_2$  erforderlich, die das Signal am Kondensator wieder auffrischt (refresh). Der Speicherkondensator  $C_S$  wird über den Schalttransistor  $M_1$  von  $C_P$  geladen und somit auch aufgefrischt.

Während eines Refresh-Vorganges wird  $M_3$  nicht aktiviert. Bei einem Schreibvorgang wird eine Bitleitung durch den Adreßdekoder selektiert und dadurch die zugehörige parasitäre Kapazität



$C_p$  geladen. Über die selektierte Wortleitung (Zeile) wird die Ladung über  $M_1$  dem Speicherkondensator  $C_S$  zugeführt.

### 7.4.2 Dynamic RAM Controller:

Der Betrieb von dynamischen RAMs erfordert zusätzliche Schaltungen. Bei einem normalen Speicherzugriff muß die Adresse in zwei aufeinanderfolgenden Schritten in das RAM geladen werden. Um einen Datenverlust zu vermeiden, ist es erforderlich, alle Zeilenadressen in (normalerweise) 8 ms mindestens einmal aufzurufen. Wenn der Speicherinhalt nicht zyklisch ausgelesen wird, sind Schaltungszusätze notwendig, die eine zyklische Adressierung zwischen den normalen Speicherzugriffen bewirken. Man bezeichnet solche Zusatzschaltungen als "DYNAMIC-RAM-Controller".

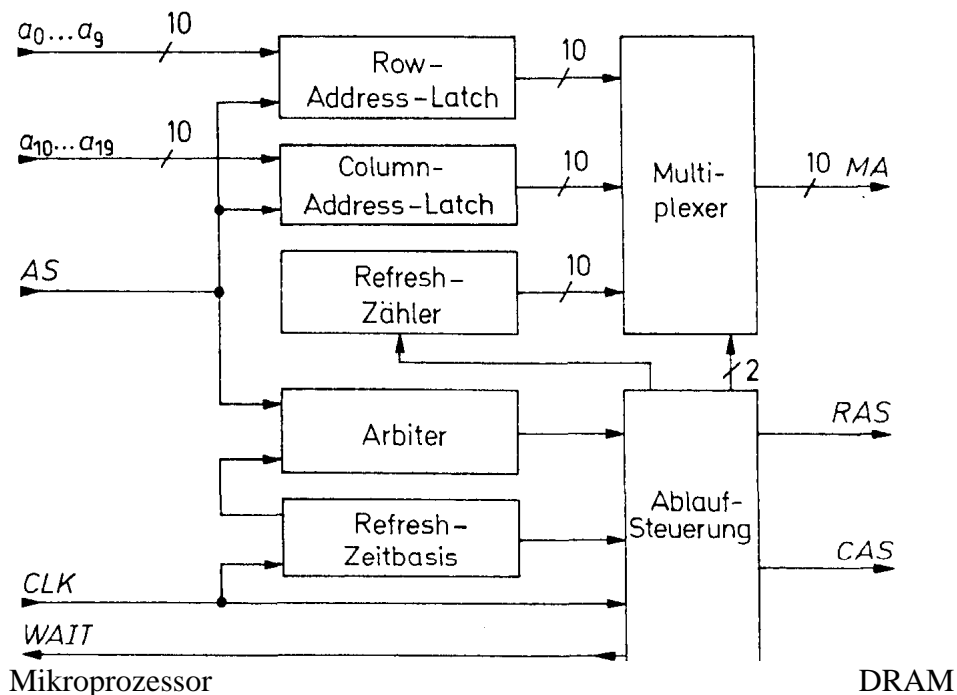


Bild: Blockschaltbild eines DRAM-Controllers

Bei einem normalen Speicherzugriff wird die außen angelegte Adresse im Row- bzw. Column-Adress-Latch eingespeichert, wenn Adress-Strobe AS "hi" wird und damit anzeigt, daß die Adresse gültig ist. Gleichzeitig wird in der Ablaufsteuerung ein Zugriffszyklus ausgelöst. Dabei wird zunächst die Zeilenadresse  $a_0 \dots a_9$  über den Multi-plexer an den Speicher ausgegeben. Dann wird der Row-Adress-Strobe gleich "hi" und bewirkt die Übernahme in den Speicher. Anschließend wird die Spaltenadresse  $a_{10} \dots a_{19}$  ausgegeben und mit dem Column-Address-Strobe ebenfalls in den Speicher eingelesen. Nach der Adresseneingabe muß das Adress-Strobe-Signal so lange auf "hi" bleiben, bis die Datenübertragung abgeschlossen ist. Der nächste Speicherzugriff darf nicht sofort erfolgen, sondern erst nach der "Precharge Time", die in derselben Größenordnung liegt wie die Zugriffszeit (Address Access Time). Zur Durchführung des Refreshs muß man die niedrigsten 512 Adressen alle 8 ms einmal anlegen. Bei einer "Refresh-Cycle-Time" von 300 ns ist dazu eine Gesamtzeit von ca. 150  $\mu$ s erforderlich. Die Verfügbarkeit des Speichers reduziert sich dadurch also um weniger als 2%. Bei der zeitlichen Aufteilung des Refreshs unterscheidet man drei verschiedene Methoden:

❶ Burst Refresh:

Bei dieser Betriebsart wird nach jeweils 8 ms der Normalbetrieb unterbrochen und ein Refresh für alle Speicherzellen durchgeführt. In vielen Fällen ist jedoch störend, daß der Speicher für 150 us blockiert ist.

❷ Cycle Stealing:

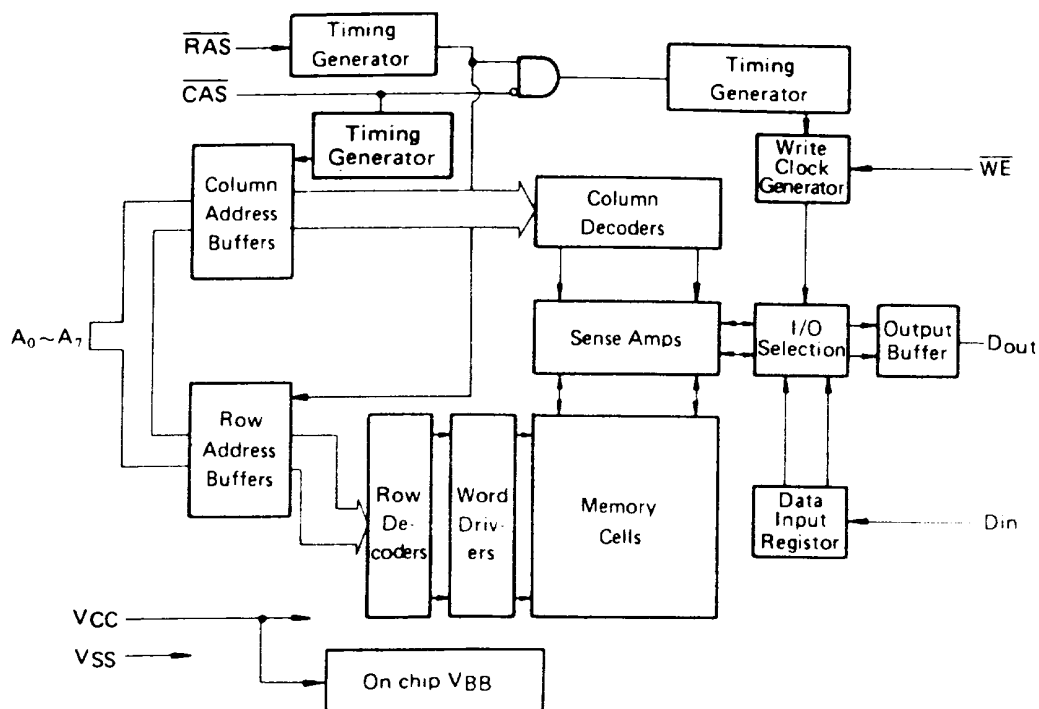
Um den Nachteil der zusammenhängenden Blockierung des Speichers zu vermeiden, kann man den Refreshvorgang gleichmäßig auf 8 ms verteilen. Wenn man den Zählerstand des Refreshzählers alle 15 us um eins erhöht, hat man nach  $512 \cdot 15\text{us}$  alle 8 ms jede Zeilenadresse einmal angelegt. Beim Cycle Stealing hält man dazu den Prozessor alle 15 us für einen Zyklus an und führt einen Refresh-Schritt aus. Zur Durchführung des Cycle Stealing ist im Bild "DRAM-Controller" eine Refresh-Zeitbasis eingezeichnet, die den Takt CLK so herunterteilt, daß die Ablaufsteuerung alle 15 us einen Refresh-Befehl gibt.

Bei einem Refresh-Zyklus wird der Zählerstand des Refresh-Zählers über den Multiplexer an den Speicher ausgegeben und das RAS-Signal vorübergehend auf "1" gesetzt. Anschließend wird der Zählerstand um eins erhöht. Während des Refresh-Zyklus wird der Benutzer des Speichers über ein Wait-Signal angehalten. Dadurch wird der laufende Prozeß alle 15 us für 0,3 us angehalten, also ebenfalls um 2% verlangsamt.

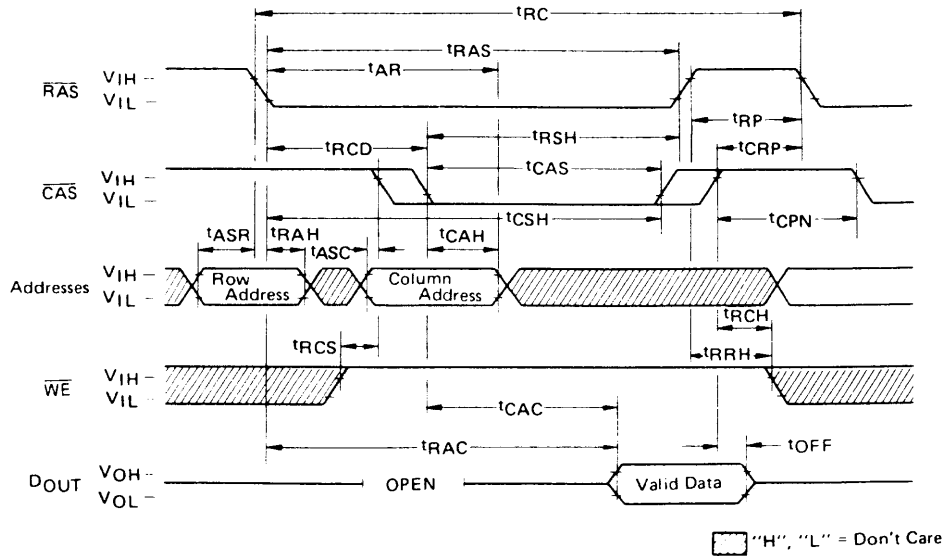
❸ Transparent- bzw. Hidden Refresh:

Bei diesem Verfahren führt man ebenfalls alle 15 us einen Refresh Zyklus aus. Man synchronisiert den Refresh Controller jedoch so, daß der Benutzer des Speichers nicht angehalten wird, sondern der Refresh genau dann ausgeführt wird, wenn der Benutzer ohnehin nicht auf den Speicher zugreift. Dadurch wird der Zeitverlust gleich Null. Wenn sich eine Überlappung eines externen Zugriffs mit einem Refresh-Zyklus nicht ganz ausschließen läßt, kann man einen zusätzlichen Prioritäts-Decoder (Arbiter), wie im "DRAM-Controller" zu erkennen ist, einsetzen. Er quittiert eine externe Anforderung mit einem Wait-Signal, bis der laufende Refresh-Zyklus abgeschlossen ist und führt sie im Anschluß daran aus.

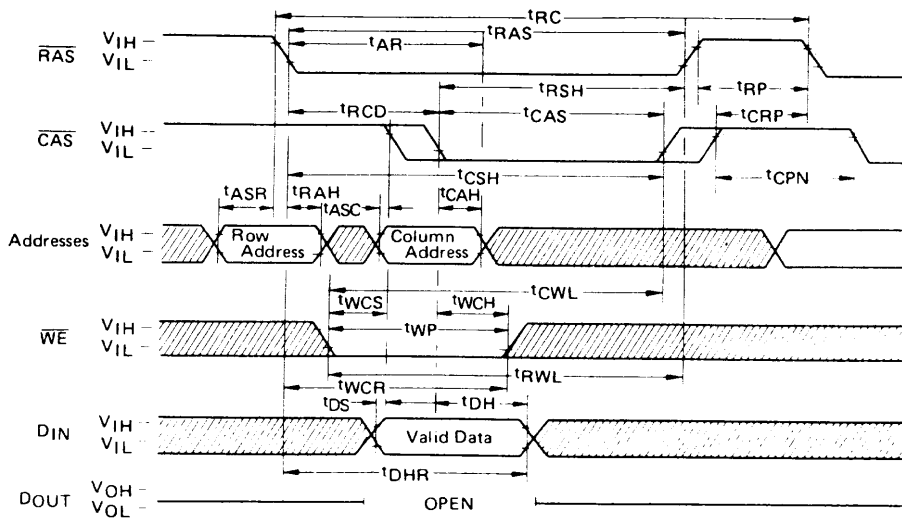
**Beispiel eines DRAM: OKI MSM3764 (65536\*1bit)**



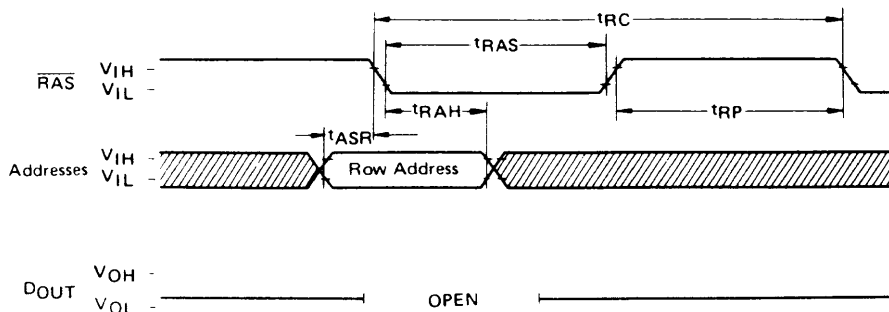
Lesezyklus:



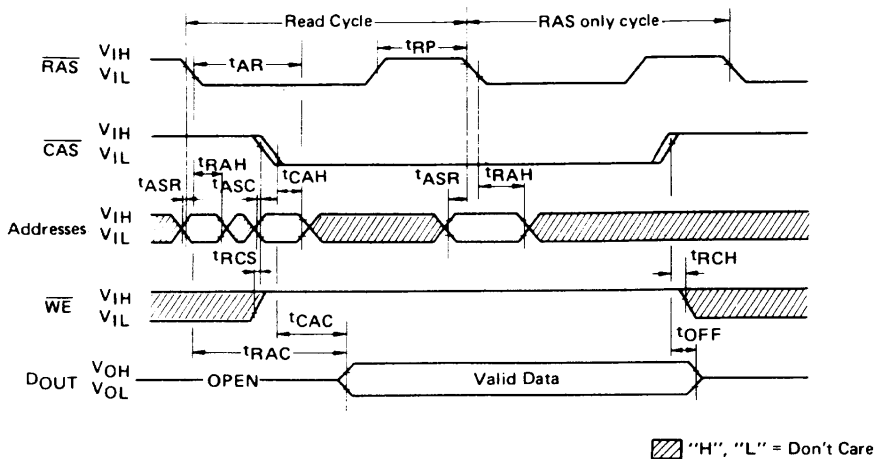
Schreibzyklus:



RAS ONLY REFRESH TIMING



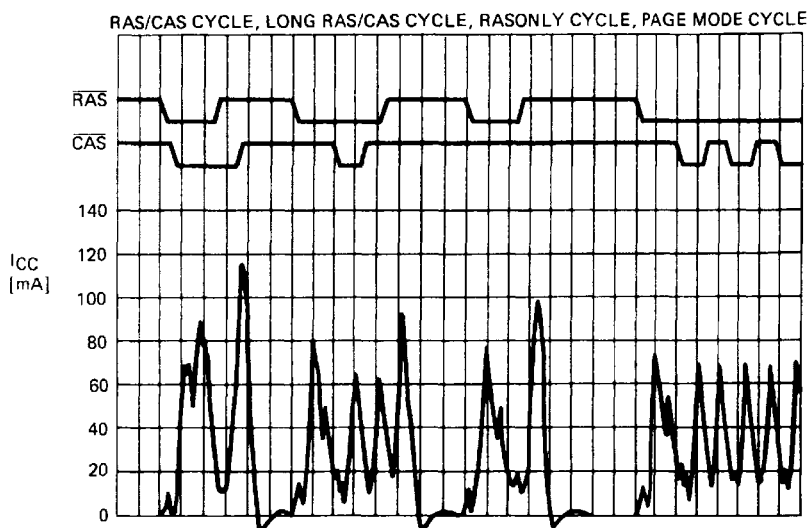
HIDDEN REFRESH



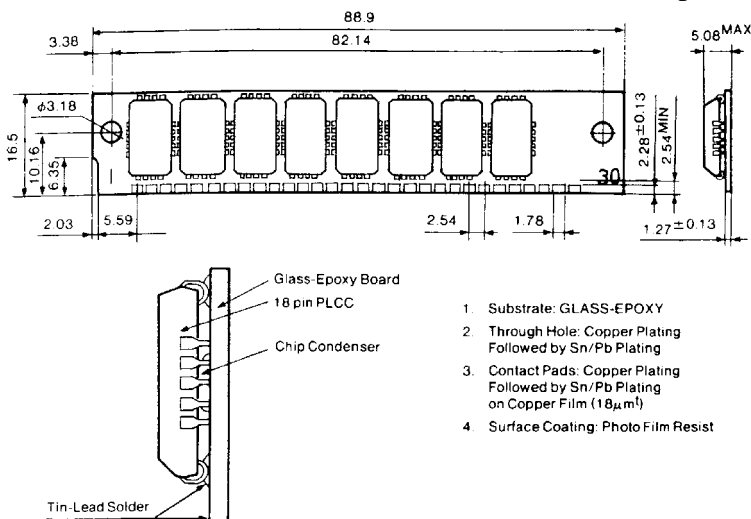
“H”, “L” = Don't Care

Stromverbrauch während der einzelnen Arbeitsphasen:

$V_{CC} = 5.5V$   
 $T_a = 25^\circ C$   
 50 ns/div



Beispiel eines SIMM-Modules für einen PC mit 4Mbit\*9 Speichertiefe:



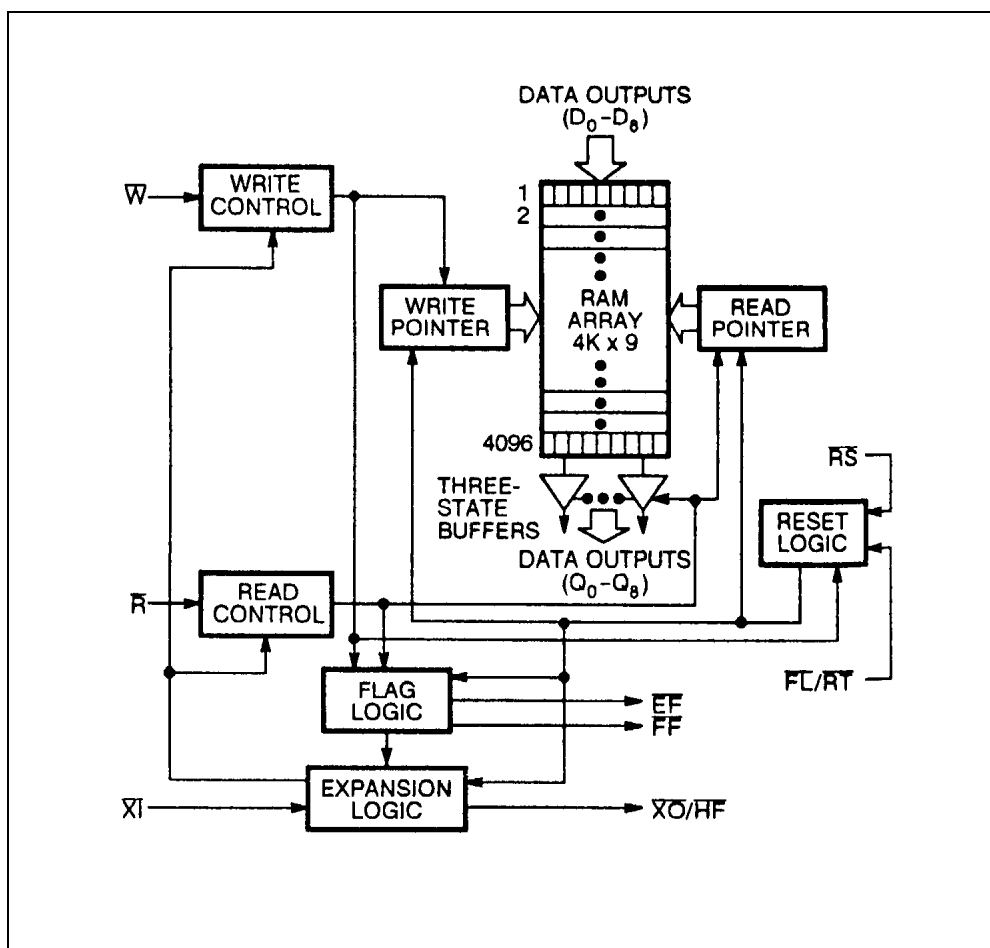
## 7.5 FIFO-Speicher (First In First Out Memories)

Ein FIFO ist eine besondere Form eines Schieberegisters. Das gemeinsame Merkmal ist, daß Daten in derselben Reihenfolge am Ausgang erscheinen, wie sie eingegeben wurden: das zuerst eingelesene Wort (First In) wird auch wieder zuerst ausgelesen. Bei einem FIFO kann dieser Vorgang völlig asynchron erfolgen, d.h. der Auslesetakt ist unabhängig vom Einlesetakt. Deshalb benutzt man FIFOs zur Kopplung asynchroner Systeme.

Die Funktion ist ähnlich einer Warteschlange: Die Daten wandern nicht mit einem festen Takt vom Eingang zum Ausgang, sondern warten solange in einem Register, bis alle vorhergehenden Daten ausgegeben sind.

Bei den FIFOs der letzten Generation werden nicht mehr wie früher die Daten verschoben, sondern lediglich zwei Zeiger aktualisiert, die die Eingabe- bzw. Ausgabe-Adresse in einem RAM angeben.

Beispiel eines FIFO Speichers:



## 7.6 Festwertspeicher (ROM)

Unter ROMs versteht man Tabellenspeicher, die im Normalfall nur gelesen werden. Sie eignen sich daher zur Speicherung von Tabellen und Programmen. Vorteilhaft ist hier, daß der Speicherinhalt beim Abschalten der Betriebsspannung erhalten bleibt. Nachteilig ist, daß die Eingabe der Tabelle sehr viel mühsamer ist als bei RAMs.

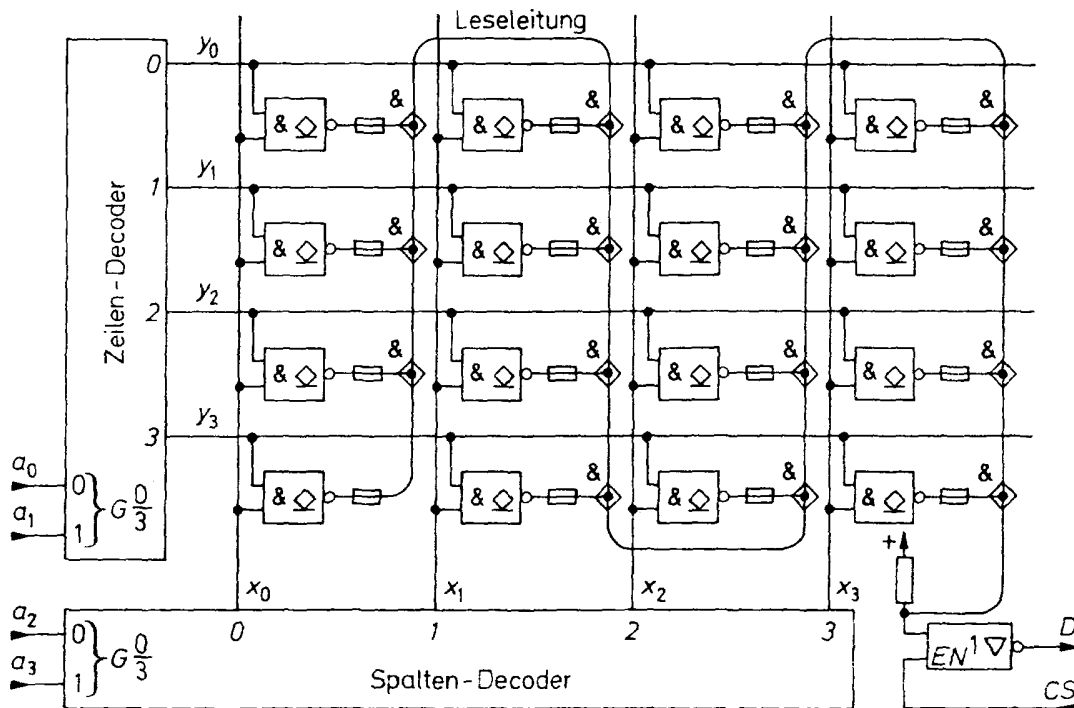
### 7.6.1 Masken-ROMs:

Bei Maskenprogrammierten ROMs (MROM) wird der Speicherinhalt vom Hersteller im letzten Herstellungsschritt mit einer spezifischen Metallisierungsmaske eingegeben. Dieses Verfahren ist nur für größeren Stückzahlen (ab ca. 10000 Stück) kostengünstig und erfordert meist mehrere Monate zur Realisierung, jedoch bietet diese Methode die größte Datensicherheit.

### 7.6.2 PROM (Programmierbare Festwertspeicher):

Unter PROMs versteht man Festwertspeicher, deren Inhalt vom Anwender programmierbar ist. Als programmierbare Bauelemente werden hier meist Schmelzsicherungen verwendet, die in den integrierten Schaltungen durch besonders dünne Metallisierungsbrücken realisiert werden. Daneben werden auch Dioden eingesetzt, die man durch Überlastung in Sperrichtung in einen Kurzschluß umwandeln kann. Die neuesten programmierbaren Bauelemente für PROMs sind spezielle MOSFETs, die ein zusätzliches "floating gate" besitzen. Dieses wird beim Programmieren aufgeladen und verschiebt dadurch die Schwellspannung des MOSFETs. Da das floating gate ringsum mit  $\text{SiO}_2$  isoliert ist, kann der Ladungserhalt für 10 Jahre garantiert werden.

Der innere Aufbau eines PROMs soll am Beispiel des Sicherungs-PROMs erklärt werden.



Interner Aufbau eines PROMs. Beispiel für 16 bit Speicherkapazität

Aus technologischen Gründen werden die einzelnen Speicherzellen nicht linear sondern in einer quadratischen Matrix angeordnet. Die Adressierung einer bestimmten Speicherzelle erfolgt dadurch, daß an die entstehende Spalten- bzw. Zeilenleitung je eine logische "1" gelegt wird. Zu

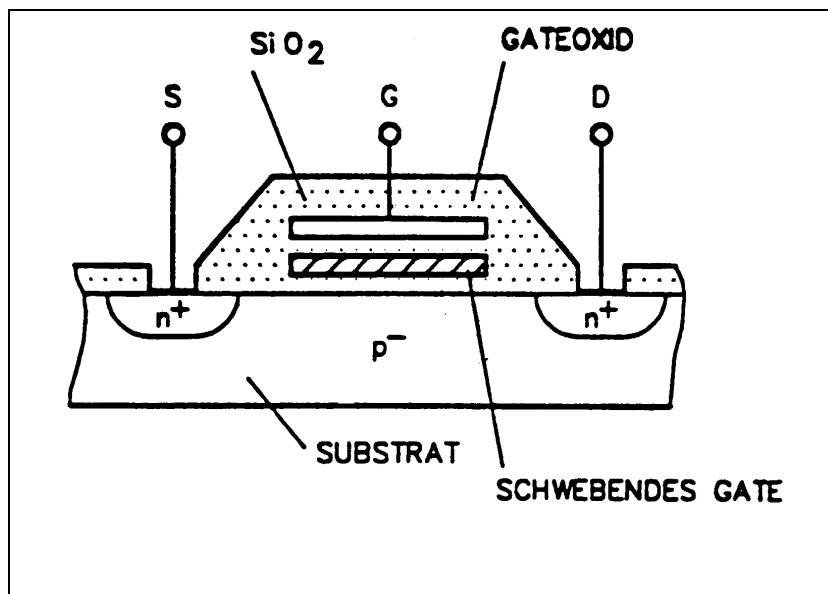
diesem Zweck muß der von außen angelegte Adressenvektor entsprechend dekodiert werden. Dazu dienen die Spalten- bzw. Zeilendekoder, die als 1-aus-n Dekoder arbeiten.

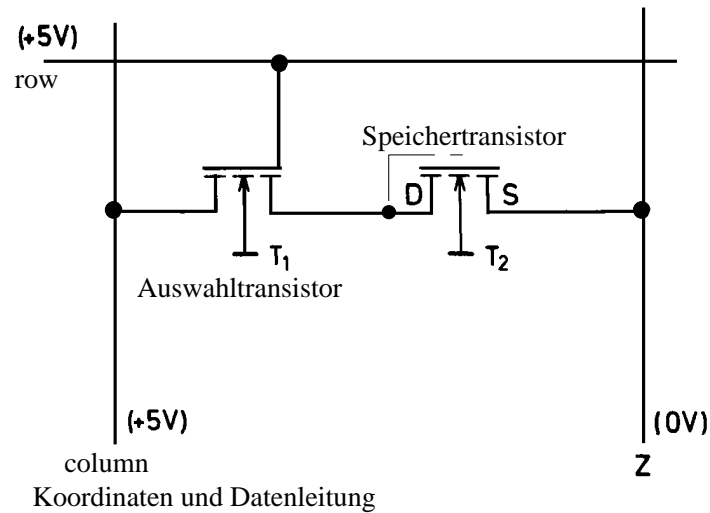
Die ausgewählte Speicherzelle wird durch das UND-Gatter am Kreuzungspunkt der selektierten Spalten- bzw. Zeilenleitung aktiviert. Die ODER-Verknüpfung aller Speicherzellenausgänge ergibt das Ausgangssignal D. Um dazu nicht ein Gatter mit  $2^n$  Eingängen zu benötigen, verwendet man eine "Wired-OR"-Verknüpfung.

Im Urzustand erzeugt jede adressierte Speicherzelle das Ausgangssignal  $D=1$ . Zur Programmierung einer "0" wird die Sicherung am Ausgang der gewünschten Zelle durchgebrannt. Dazu wird die Adresse der entsprechenden Zelle angewählt und damit der Ausgangstransistor des NAND-Gatters leitend gemacht. Dann prägt man in die Leseleitung einen kräftigen Stromimpuls ein, der gerade so groß ist, daß die Sicherung am Ausgang des NAND-Gatters durchbrennt. Dabei muß ein vom Hersteller genau vorgeschriebener Zeitablauf eingehalten werden.

### 7.7 UV-löschbare Festwertspeicher (EPROM):

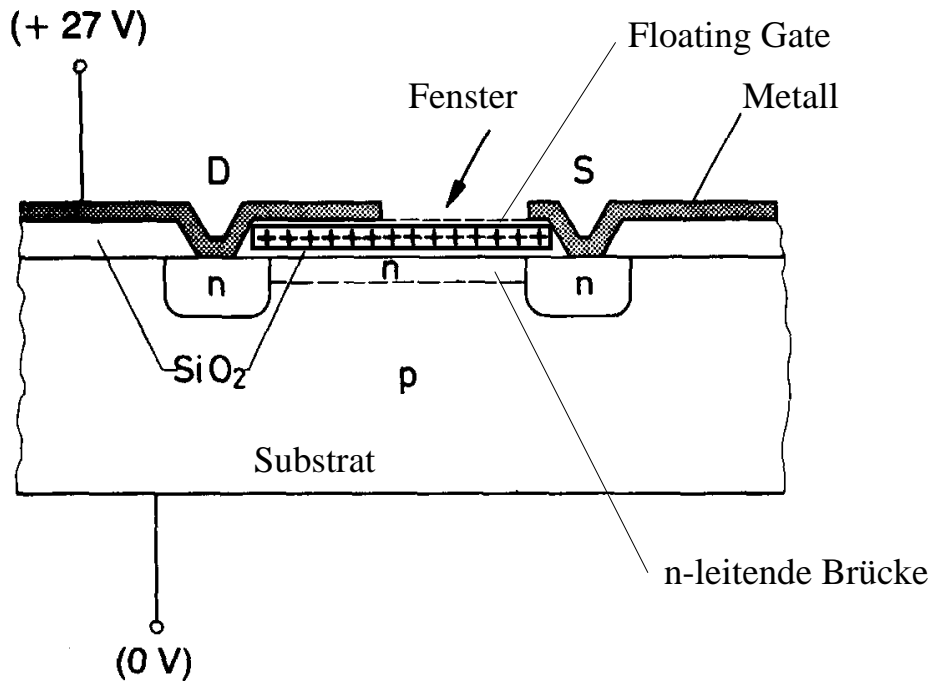
Unter einem EPROM (Erasable PROM) versteht man einen Festwertspeicher, der sich nicht nur vom Anwender programmieren läßt, sondern der auch mit ultraviolettem Licht löschen läßt. Als Speicherelement verwendet man hier ausschließlich MOSFETs mit einem zusätzlichen "floating gate". Diese Transistoren werden auch FAMOS (floating gate avalanche MOS) genannt. Das floating gate wird beim Programmieren aufgeladen und verschiebt dadurch die Schwellspannung des MOSFETs. Diese Ladung kann mittels UV-Strahlung (ca. 20 Minuten) wieder entfernt werden. Um dies zu ermöglichen, besitzen diese Gehäuse über dem Chip ein Fenster aus Quarzglas.





Das Gate des Speichertransistors T2 ist von hochisolierendem Werkstoff umgeben. Es ist nirgendwo angeschlossen ("floating gate"). Im gelöschten Zustand ist das Floating Gate ohne Ladung. Der Transistor T2 ist also gesperrt. Legt man jetzt an die X-Koordinatenleitung (row) und an die Y-Koordinatenleitung (column) jeweils +5V, so wird der Transistor T1 durchgeschaltet. Der Transistor T2 ist aber gesperrt, so daß die Y-Leitung, die gleichzeitig Datenleitung ist, nicht auf lo-Pegel gezogen wird. Die Y-Leitung bleibt auf hi-Pegel. Dies entspricht informatorisch "1" und stellt die Information für ein gelöschtes EPROM dar.

Beim Programmieren einer Information werden bestimmte Speicherelemente auf "0" gesetzt. Es werden also "Nullen programmiert". Wird ein Speicherelement mit durchgeschaltetem Speichertransistor T2 abgefragt, wird die Y-Leitung auf lo-Pegel gezogen, da die Transistoren T1 und T2 leiten.





### Aufladung des Floating Gates:

Zwischen D und Substrat wird eine verhältnismäßig hohe Spannung angelegt (Beispiel 27V). Da das Floating Gate und die Isolierschichten sehr dünn sind, entsteht ein sehr starkes elektrisches Feld. Unter dem Einfluß dieses starken Feldes wandern Elektronen vom Floating Gate zum Drain ab (Elektronenwanderung entgegen der Feldlinienrichtung). Dieser Vorgang wird Floating Gate Avalanche-Injection (lawinenartige Aufladung des schwimmenden Gates) genannt.

## 7.8 Elektrisch löschbare Festwertspeicher (EEPROMs):

Unter einem EEPROM (Electrically Erasable PROM) versteht man ein PROM, das sich im Gegensatz zum EPROM auch elektrisch löschen läßt. Bei den neueren Typen sind der Spannungswandler zur Erzeugung der Programmierspannung und der Timer zur Festlegung der Programmierimpulsdauer auf einem Chip integriert. Um ein Byte zu programmieren, muß man lediglich Adresse und Daten anlegen. Wenn man dann die Programmierung mit einem Schreibbefehl auslöst, speichert das EEPROM die Adresse und Daten intern und gibt die Adreß- und Datenleitung sofort wieder frei. Der weitere Vorgang läuft auf dem Chip autonom ab. Zuerst wird das alte Byte gelöscht, und dann das neue programmiert. Dieser Vorgang wird intern überwacht, um sicherzustellen, daß die programmierte Ladung ausreicht.

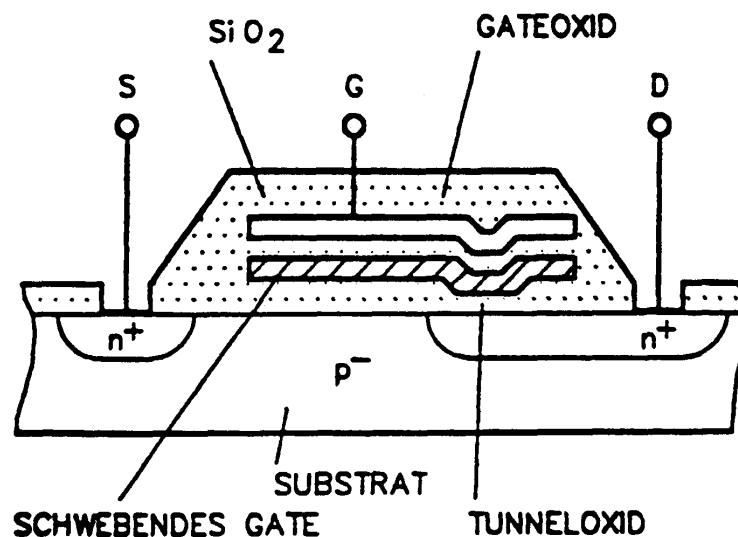


Bild: Prinzipieller Aufbau eines FAMOS-Transistors für eine EEPROM-Zelle

Die Zahl der möglichen Schreibzyklen ist begrenzt ( $10^4$  ...  $10^6$  je nach Typ). Wie bei den meisten Speichern reduziert sich auch bei vielen EEPROMs die Verlustleistung, wenn sie nicht selektiert werden. Die geringste Verlustleistung ergibt sich natürlich, wenn man die Betriebsspannung ganz abschaltet. Die Daten gehen dadurch, wie bei allen ROMs, nicht verloren. Allerdings ergibt sich nach dem Anlegen der Betriebsspannung eine erhöhte Zugriffszeit, da zunächst die Leseverstärker einschwingen müssen. Aus diesem Grund ist es nicht ratsam, die Betriebsspannung erst bei einem Speicherzugriff einzuschalten.

## 7.9 Flash-EEPROMs:

Die Flash-EEPROMs stellen ein Mittelding zwischen den EPROMs und den EEPROMs dar. Sie sind zwar wie die EEPROMs elektrisch löscher, aber wie die EPROMs nicht byteweise sondern nur der ganze Chip (oder Blöcke) auf einmal; daher kommt der Name Flash-EEPROM. Die Löschung ist viel einfacher als bei EPROMs: Sie erfolgt mit einem einzigen Löschimplus, der einige Sekunden lang ist.

## 7.10 Zweitortspeicher:

Zweitortspeicher sind spezielle RAMs, die es zwei unabhängigen Prozessen ermöglichen, auf gemeinsame Daten zuzugreifen. Dies ermöglicht einen Datenaustausch zwischen zwei Prozessoren (Beispiel VIDEO-RAM in einem Personalcomputer, Speicher in einem LCD-System).

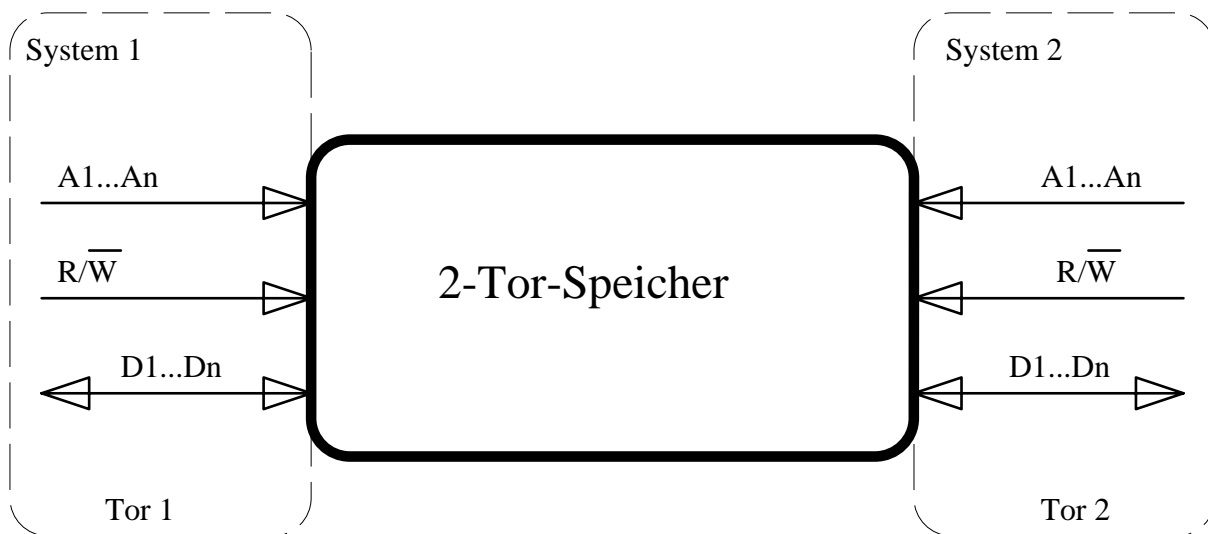


Bild: Prinzipschaltbild eines Zweitortspeichers

Dazu muß der Zweitortspeicher zwei getrennte Sätze von Adreß-, Daten- und Steuerleitungen besitzen. Dieses Prinzip läßt sich nicht ohne Einschränkungen realisieren, da es prinzipiell unmöglich ist, gleichzeitig von beiden Toren in dieselbe Speicherzelle zu schreiben.

Dieses Problem wird bei den "Read-While-Write-Speichern" dadurch umgangen, daß an einem der beiden Tore nur geschrieben wird und am anderen nur gelesen.

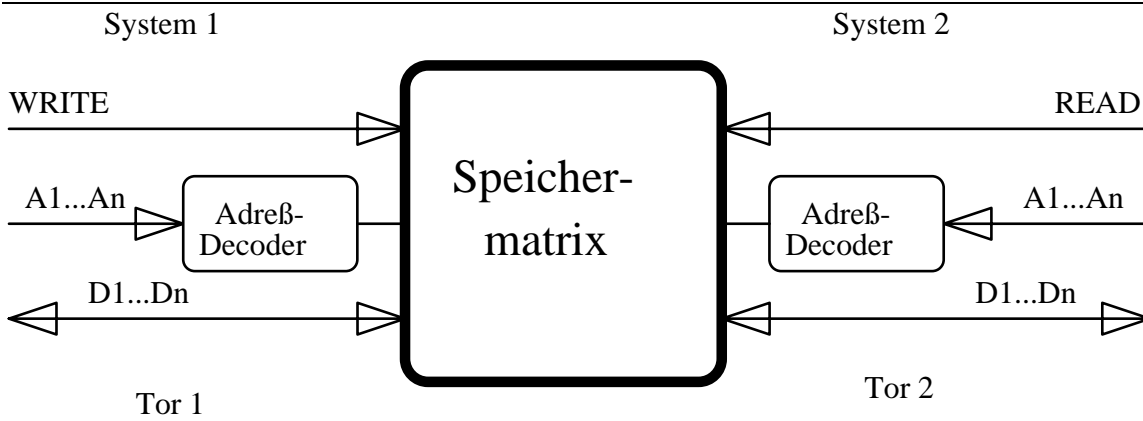
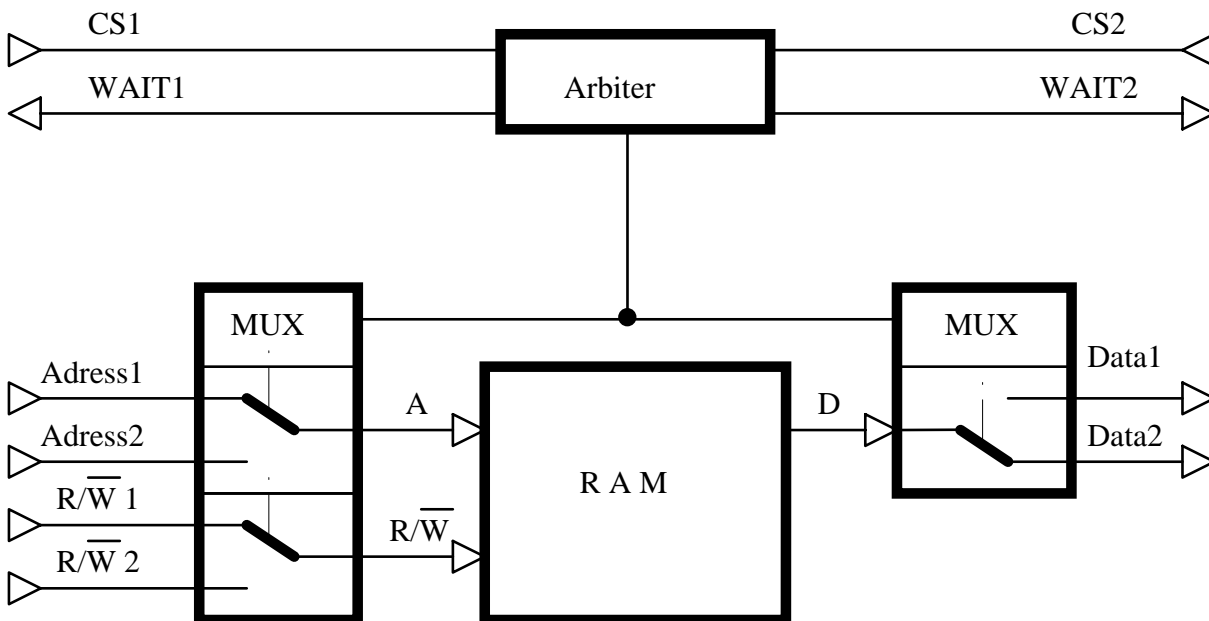


Bild: Aufbau eines Read-While-Write-Speichers mit getrennten Adreß-Eingängen

Diese Speicher besitzen zwei getrennte Adreß-Decoder, die es ermöglichen, gleichzeitig auf eine Adresse zu schreiben und einer anderen zu lesen.

Wenn an beiden Toren eines Zweitortspeichers gelesen und geschrieben werden soll, läßt sich ein Zugriffskonflikt im allgemeinen nur dadurch umgehen, daß man gleichzeitige Speicherzugriffe verhindert. Dazu kann man, wie im nächsten Bild erkennbar ist, Adreß-, Daten- und Steuerleitungen über einen MUX dem angesprochenen Tor zur Verfügung stellen. In vielen Fällen lassen sich die beiden auf den Speicher zugreifenden Prozesse so miteinander synchronisieren, daß ein gleichzeitiger Speicherzugriff ausgeschlossen ist. Wenn dies nicht möglich ist, kann man einen Prioritätsdecoder einsetzen, der bei überlappenden Speicherzugriffen einen der beiden Prozesse über ein Wait-Signal vorübergehend anhält.

Integrierte Zweitortspeicher (74AS870 16x8, IDT7133 2kx8....) sind erhältlich, haben aber eine geringe Speichertiefe. Um Zweitortspeicher mit einer großen Speichertiefe zu realisieren, ist es zweckmäßig, normale RAMs zusammen mit einem Dual-Port-RAM-Controller einzusetzen. In diesem Fall ist der Baustein 74LS764 besonders vorteilhaft, weil er den Betrieb von dynamischen RAMs als Zweitortspeicher unterstützt.



## 8 Anschluß eines LCDs

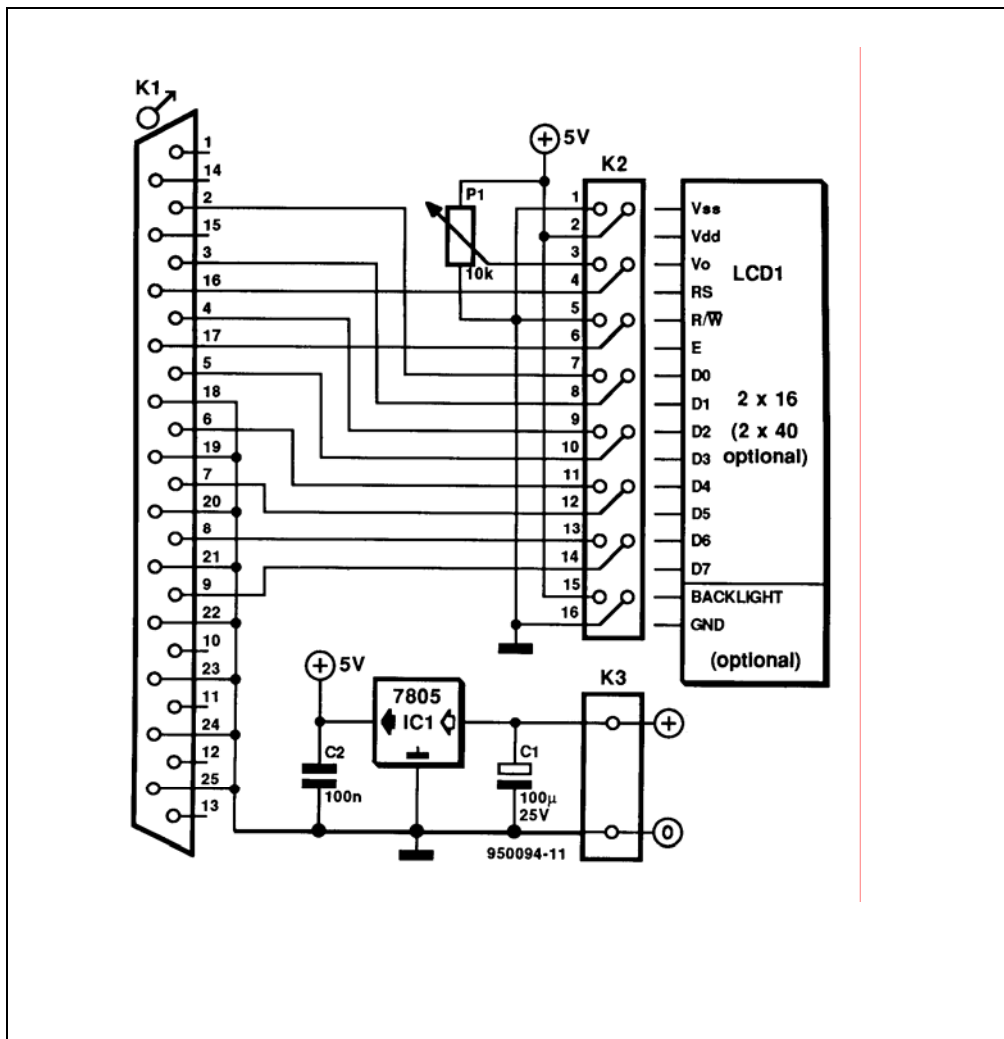
### 8.1 Allgemeines

Eine sehr oft verwendete Anzeigeeinheit in Verbindung mit einem Mikrocontroller ist eine LCD Anzeige. Neben normalen Sieben Segment Displays ohne integrierte Ansteuerelektronik sind auch komplette Anzeigemodule erhältlich. Hier unterscheidet man Punkt Matrix LCD Module für Schriftdarstellung und Punkt Matrix LCD Module, mit welchen Grafik dargestellt werden kann.

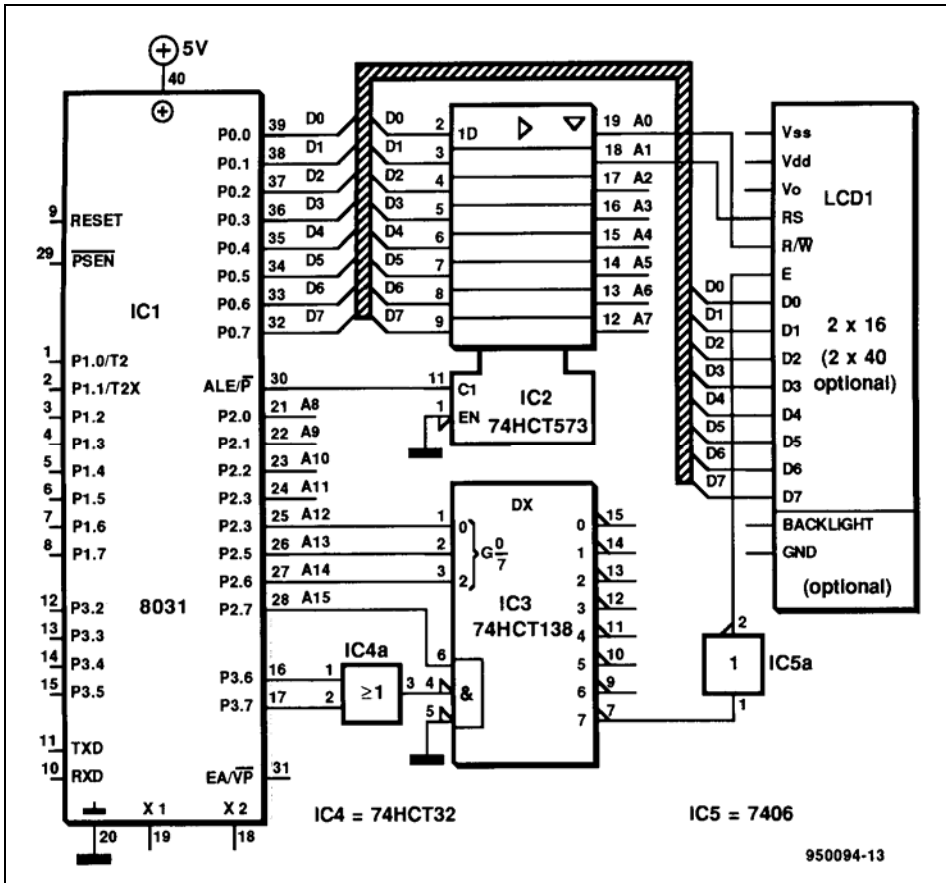
Punkt Matrix Module für Schriftdarstellung sind in verschiedenen Größen erhältlich. Sie verwenden alle eine Zeichenmatrix mit 5x7 Punkten zur Darstellung der Zeichen. Die auf dem Modul enthaltene Ansteuerelektronik ist meist mit dem Controller/Treiberbaustein HD44780 aufgebaut, welcher einen integrierten Zeichengenerator und ein Display RAM enthält.

Auf der Übungsplatine ist ein Punkt Matrix Modul mit 2 Zeilen zu je 20 Zeichen aufgebaut. Dieses ist über den Port 6 an den Mikrocontroller 80C517 angeschlossen. Das Modul benötigt drei Steuerleitungen und vier Datenleitungen.

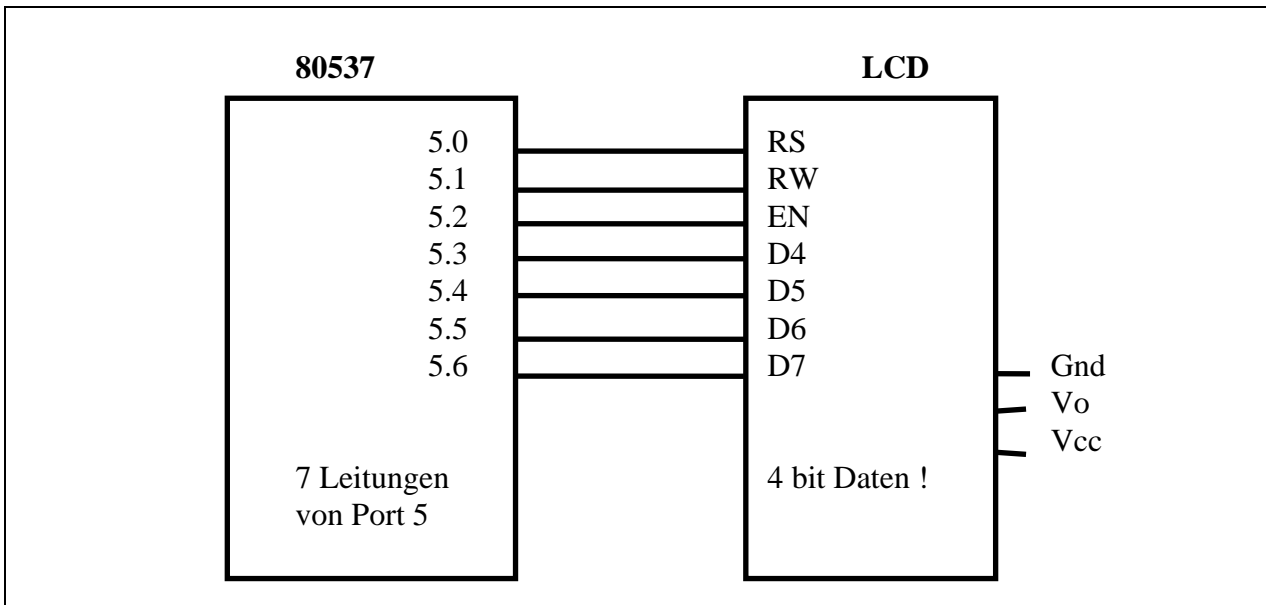
### 8.2 Anschluß an die Druckerschnittstelle



### 8.3 Beispiel für den Anschluß an einen 8031er Adresse F000H:



### 8.4 Anschluß direkt an einen Port:



Der im ROM des Controllers verfügbare Zeichensatz. Es können auch noch maximal 8 Zeichen vom Anwender definiert werden.

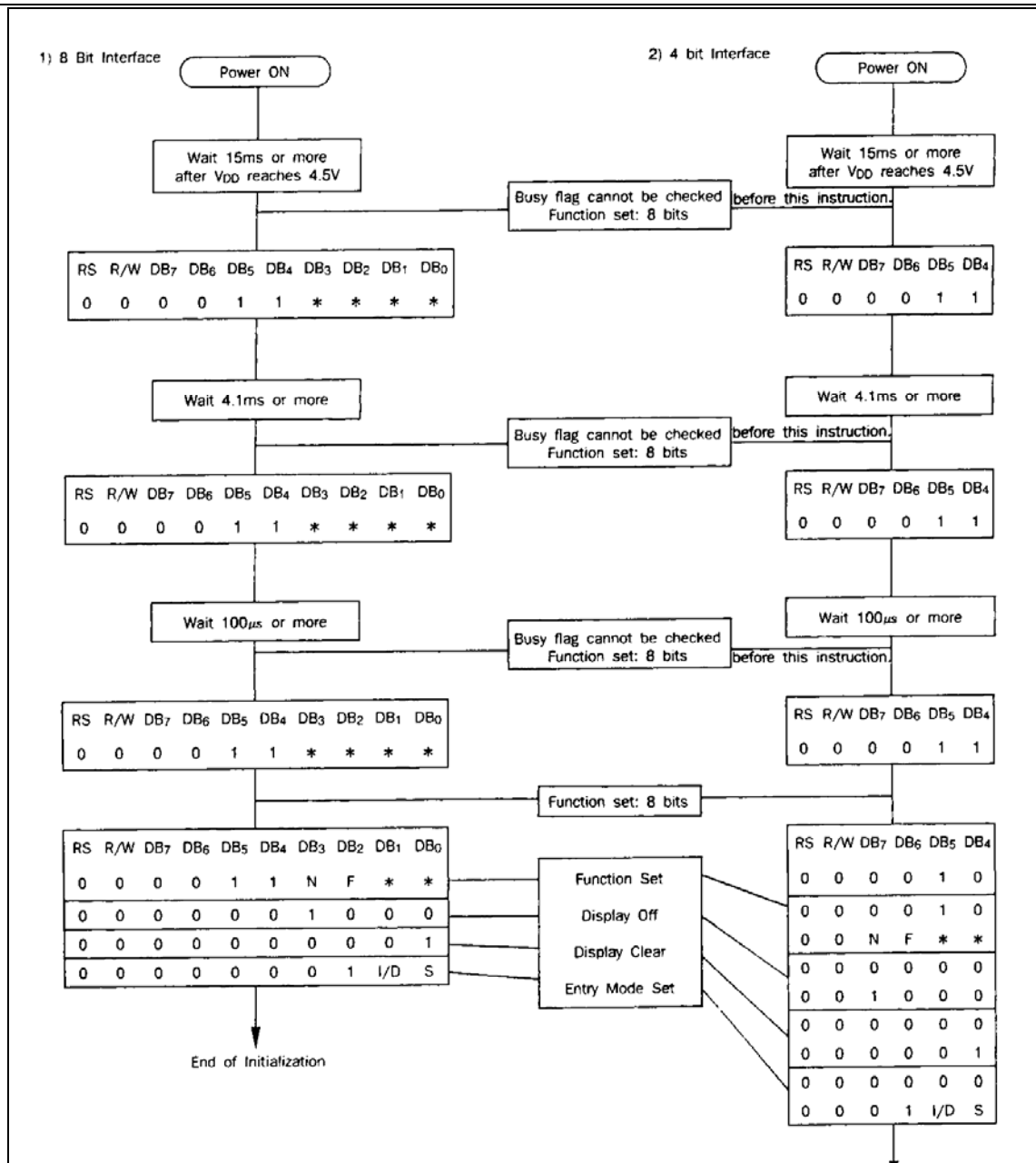
### 8.5 Befehlssatz

Higher 4 bit Lower 4 bit	0000	0010	0011	0100	0101	0110	0111	1010	1011	1100	1101	1110	1111
xxxx0000	CG RAM (1)		0	a	P	´	P	—	9	E	α	P	
xxxx0001	(2)	!	1	A	0	a	9	a	7	9	4	ä	g
xxxx0010	(3)	"	2	B	R	b	r	´	ı	ı	x	P	θ
xxxx0011	(4)	#	3	C	S	c	s	ı	o	t	e	e	∞
xxxx0100	(5)	\$	4	D	T	d	t	ı	I	T	†	P	α
xxxx0101	(6)	%	5	E	U	e	u	ı	ı	ı	ı	ı	ı
xxxx0110	(7)	&	6	F	V	f	v	ı	ı	ı	ı	P	Σ
xxxx0111	(8)	´	7	G	W	g	w	ı	ı	ı	ı	ı	ı
xxxx1000	(1)	(	8	H	X	h	x	ı	ı	ı	ı	ı	ı
xxxx1001	(2)	)	9	I	V	i	v	ı	ı	ı	ı	ı	ı
xxxx1010	(3)	*	#	J	Z	j	z	ı	ı	ı	ı	ı	ı
xxxx1011	(4)	+	;	K	L	k	l	ı	ı	ı	ı	ı	ı
xxxx1100	(5)	,	<	L	*	I	I	ı	ı	ı	ı	ı	ı
xxxx1101	(6)	—	=	M	I	m	)	ı	ı	ı	ı	ı	ı
xxxx1110	(7)	.	>	N	^	n	ı	ı	ı	ı	ı	ı	ı
xxxx1111	(8)	/	?	0	_	o	ı	ı	ı	ı	ı	ı	ı

Elektor 10/95

27

Initialisierungsroutine des Displays im Flußdiagramm. Dieser Prozeß muß nach jedem Einschalten durchlaufen werden, er dauert aber nur wenige Millisekunden.



Der Befehlssatz des Hitachi-Displaycontrollers, der in den meisten alphanumerischen, ein- bis zweizeiligen LC-Anzeigen verwendet wird.

Instruction	Code											Description	Execution Time (max) (when fcp or fosc is 250 KHz)
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0			
Clear Display	0	0	0	0	0	0	0	0	0	0	1	Clears entire display and sets DD RAM address 0 in address counter.	1.64 ms
Return Home	0	0	0	0	0	0	0	0	0	1	*	Sets DD RAM address 0 in address counter. Also returns display being shifted to original position. DD RAM contents remain unchanged.	1.64 ms
Entry Mode Set	0	0	0	0	0	0	0	0	1	I/D	S	Sets cursor move direction and specifies shift of display. These operations are performed during data write and read.	40 μs
Display On / Off Control	0	0	0	0	0	0	0	1	D	C	B	Sets ON/OFF of entire display (D), Cursor ON/OFF (C), and blink of cursor position character (B).	40 μs
Cursor or Display Shift	0	0	0	0	0	1	S/C	R/L	*	*		Moves cursor & shifts display without changing DD RAM contents.	40 μs
Function Set	0	0	0	0	1	DL	N	F	*	*		Sets interface data length (DL), number of display lines (L) and character fonts (F).	40 μs
Set CG RAM Address	0	0	0	1	ACG							Sets CG RAM address. CG RAM data is sent and received after this setting.	40 μs
Set DD RAM Address	0	0	1	ADD							Sets DD RAM address. CG RAM data is sent and received after this setting.	40 μs	
Read Busy Flag and Address	0	1	BF	AC							Reads Busy flag (BF) indicating internal operation is being performed and reads address counter contents.	0 μs	
Write Data to CG or DD RAM	1	0	Write Data							Writes data into DD RAM or CG RAM	40 μs		
Read Data from CG or DD RAM	1	1	Read Data							Reads data into DD RAM or CG RAM	40 μs		
	I/D = 1 : Increment I/D = 0 : Decrement S = 1 : Accompanies display shift S/C = 1 : Display shift S/C = 0 : Cursor move R/L = 1 : Shift to the right R/L = 0 : Shift to the left DL = 1 : 8 bits, DL = 0 : 4 bits N = 1 : 2 lines, N = 0 : 1 line F = 1 : 5 × 10 dots, F = 0 : 5 × 7 dots FB = 1 : Internally operating FB = 0 : Can accept instruction											DD RAM : Display data RAM CG RAM : Character generator RAM ACG: CG RAM address ADD: DD RAM Address : Corresponds to cursor address AC: Address counter used for both DD and CG RAM address.	Execution time changes when frequency changes Example: When fcp or fosc is 270 kHz: $40 \mu s \times \frac{250}{270} = 37 \mu s$

\* No effect

950094-1-16



## 9 Reset

Die Hardware-Reset-Funktion bei Microprozessoren und -controllern wird benötigt, um den Baustein nach dem Einschalten in einen definierten Startzustand zu bringen. Aber auch während des Betriebs kann über die Reset-Funktion ein Neustart des Bausteins erzwungen werden. Ein besonderer Anwendungsfall für einen Hardwarereset ist die Beendigung des SW-Power-Down-Modi, aus dem kein anderer Rückweg als ein Reset existiert.

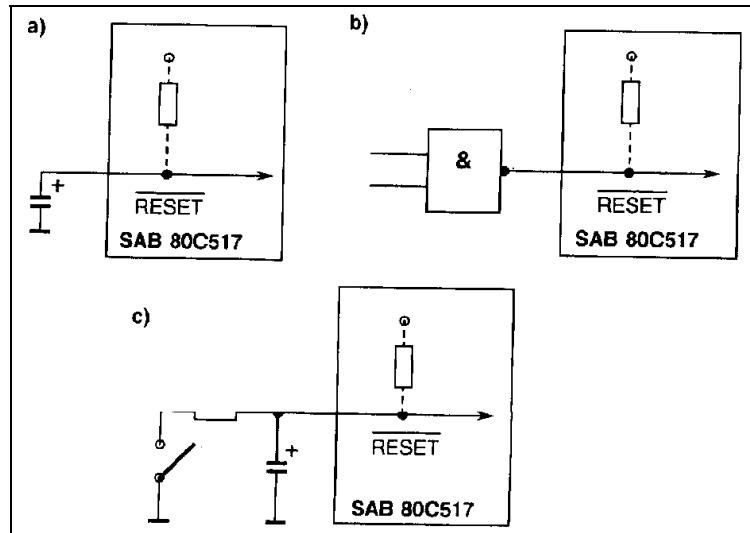


Abb. Reset-Beschaltungen für den 80C517/C517A

Register	Wert	Register	Wert
PC	0000H	ACC	00H
ADCON0	00H	ADCON1	xxxx 0000B
ADDAT	00H	ARCON	0xxx xxxxB
B	00H	CC4EN	00H
CCEN	00H	CCH1-4	00H
CCL1-4	00H	CMEN	00H
CMH0-7	00H	CML0-7	00H
CMSSEL	00H	CRCL, CRCH	00H
CTCON	xxxx 0000B	CTRELL, CTRELH	00H
DAPR	00H	DPSEL	xxxx x000B
DPTR0-7	0000H	IEN0, IEN1	00H
IEN2	xxxx 0xx0B	IP0, IP1	00H
IRCON	00H	MD0-5	XXH
P0-P6	0FFH	P7, P8	XXH
PCON	00H	PSW	00H
S0BUF, S1BUF	XXH	S0CON	00H
S1CON	0x00 0000B	S1REL	00H
SP	07H	T2CON	00H
TCON	00H	TL0, TH0	00H
TL1, TH1	00H	TL2, TH2	00H
TMOD	00H	WDTRL	00H

Es ist wichtig zu wissen, daß der Inhalt des internen RAM vom Reset-Vorgang nicht berührt wird. Dies bedeutet zum einem, daß der RAM-Inhalt nach einem Power-On-Reset vollständig undefiniert ist, daß aber nach einem Reset bei bereits laufendem Prozessor nichts am vorigen RAM-Inhalt verändert wurde.

## 10 Parallele Eingabe-Ausgabe-Ports

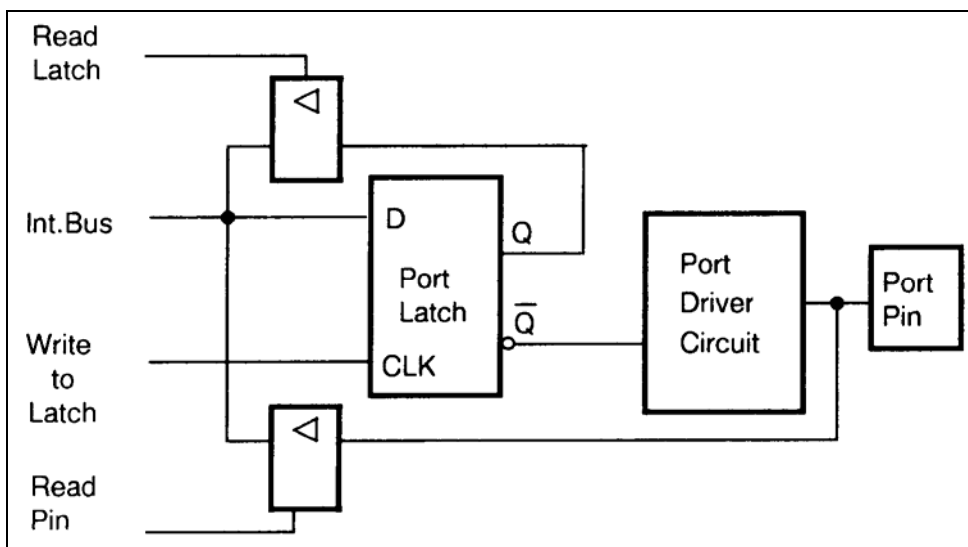
Der 80C517/C517A besitzt sieben Eingabe-Ausgabe-Ports (I/O-Ports) und zwei Ports, die nur für Eingangsfunktionen zur Verfügung stehen.

### 10.1 Digitale Ein-Ausgabe-Ports

56 Portleitungen sind zu sieben, mit je 8 Leitungen, Ports zusammengefaßt. All diese Anschlüsse, die an einzelnen Portpins am Baustein zur Verfügung stehen, sind wahlweise als Eingang oder Ausgang für digitale Signale zu betreiben. Jeder Pin hat sein eigenes Latch, eine spezielle Ausgangstreiberschaltung, sowie einen Eingangstreiber.

#### Port 0 bis Port 5 sind bitadressierbar, P6 ist nicht bitadressierbar

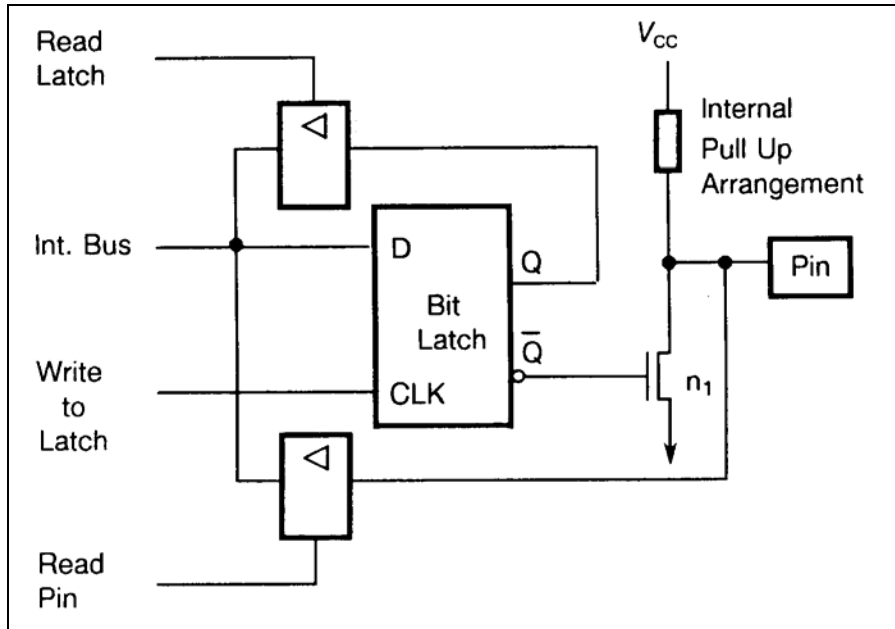
Nachfolgende Abbildung zeigt ein Blockschaltbild der grundlegenden Struktur aller Eingabe-Ausgabe-Ports. Es ist ein einzelnes Bit eines Ports abgebildet; je 8-bit-Port ist diese Struktur also achtmal vorhanden. Der Kern der Struktur ist das Port-Latch, das durch ein D-Flip-Flop gebildet wird. Ein Schreibvorgang in das entsprechende Port-Register PO bis P6 (z.B. MOV P3,#55H) übernimmt den entsprechenden Bitwert in alle acht Latches eines Ports durch Betätigen der internen CLK-Leitung am Latch. Bis zu einem weiteren Schreibvorgang auf das Latch bleibt dieser Wert dann gespeichert. Der nicht-invertierte Ausgang des Latch kann durch bestimmte Befehle, die den Read-Latch-Treiber aktivieren, wieder zurückgelesen werden. Der invertierte Ausgang des Latch dient zur Ansteuerung der Treiberschaltung für den Portpin, wo ein entsprechender Pegel erzeugt wird. Der Pegel am Pin selbst kann ebenfalls durch eine eigene Treiberschaltung (Read Pin) wieder eingelesen werden.



Im der vorigen Abbildung ist die eigentliche Porttreiberschaltung noch sehr allgemein dargestellt. In der nächsten Abbildung ist eine weitere Detaillierung durchgeführt, wie sie bei den Ports 1 bis 6 gilt. Diese Ports werden quasibidirektional genannt. Sie erlauben nämlich den wahlweisen Betrieb jedes einzelnen Pins als Ein-oder Ausgang. Im Gegensatz zu einem echten bidirektionalen Ausgang sind allerdings gewisse Besonderheiten zu beachten.

Die Struktur benutzt im Ausgangstreiber einen starken N-Kanal-Feldeffekttransistor n1 zur Erzeugung des Low-Pegels. Der High-Pegel wird durch ein aus mehreren Transistoren

bestehendes Arrangement, das als Pull-Up-Widerstand wirkt, erzeugt. Wenn sich im Port-Latch eine 0 befindet, ist der Ausgang Q# auf 1; damit wird der Transistor n1 eingeschaltet und zieht den Portpin auf Low-Pegel. Wenn sich im Port-Latch eine 1 befindet, ist der Ausgang Q# auf 0; damit wird der Transistor n1 nicht aktiviert und der Pull-Up zieht den Portpin auf eine schwache 1 (High-Pegel). Somit sind die Ausgabe von 0 und 1 entsprechend der Werte im Port-Latch gewährleistet.



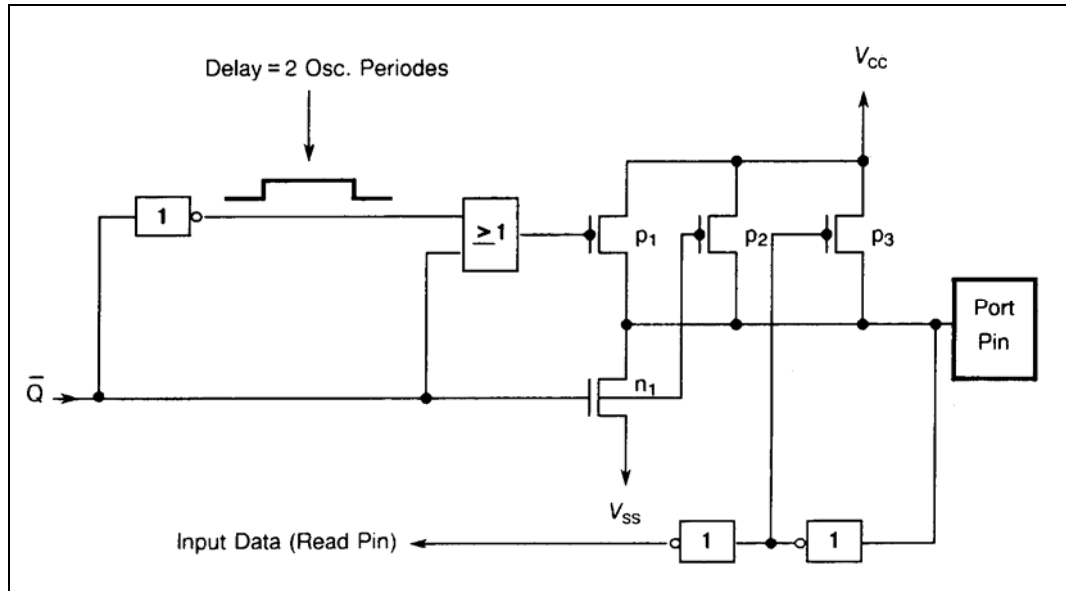
Darüber hinaus ist es nun aber möglich, den Portpin von außen auf Low-Pegel zu ziehen, während im Latch eine 1 steht. Genau dies wird getan, wenn der Pin als Eingang benutzt wird. Das Pull-Up-Arrangement kann dabei von externer Beschaltung problemlos auf Low-Pegel heruntergezogen werden. Der Pegel am Pin unterscheidet sich dann vom Inhalt des Latch. Daher kann der Pin-Pegel wie oben schon erwähnt auch getrennt vom Latch eingelesen werden.

Es sei nochmals ausdrücklich darauf hingewiesen, daß durch die quasi-bidirektionale Struktur keine gesonderte Programmierung der Ports für die Eingabe- oder Ausgabefunktion nötig ist, wie es bei verschiedenen externen Peripheriebausteinen mit parallelen Ports nötig ist. Beim bekannten 8255 muß z.B. für einen ganzen Port einheitlich festgelegt werden, ob er zu Ein- oder Ausgabe genutzt werden soll. Beim 80C517/80C517A (und auch den anderen Bausteinen der 8051-Familie) ist dies nicht nötig. Hier muß nur eine 1 bei den Pins im Latch stehen, die als Eingang verwendet werden. Somit ist es auch möglich, Ein- und Ausgabe beliebig zu mischen (auch innerhalb desselben Ports).

Zusammenfassend liegen damit für die Ports 1 bis 6 folgende Möglichkeiten vor:

- ➔ **Betrieb als Ausgang:** Am Ausgang erscheinen die in das Port-Latch programmierten Werte. Allerdings sind die Treiberleistungen für die Low- und High-Pegel sehr unterschiedlich.
- ➔ **Betrieb als Eingang:** Im Port-Latch muß sich eine 1 befinden. Dann kann von außen Low oder High-Pegel angelegt werden. Während sich der High-Pegel durch die Pull-Up-Struktur von selbst ergibt, ist zur Erreichung des Low-Pegels ein gewisser Treiberstrom erforderlich.

Die bisher global beschriebene Funktion der Ausgangstreiberschaltung im Port 1 bis 6 ist in der nächsten Abbildung nun ganz detailliert aufgeschlüsselt und wird im folgenden beschrieben. Es werden auch die von diesen Transistoren erzeugten Treiberströme genannt, auf die im Datenblatt des Bausteins Bezug genommen wird. Die Kenntnis dieser Zusammenhänge erlaubt dann eine Dimensionierung externer Beschaltungen an den Portpins.



Die Pull-Up-Struktur der Ports 1 bis 6 setzt sich aus drei Feldeffekttransistoren p1, p2 und p3 zusammen. Für die Erzeugung des Low-Pegels am Pin ist der Transistor n1 zuständig. Die Funktion und die Bedingungen, wann die Transistoren aktiviert werden, sind anschließend einzeln aufgeführt.

- Der FET n1 ist ein N-Kanal-Feldeffekttransistor vom Anreicherungstyp. Er ist leitend, wenn High-Pegel am Gate angelegt wird; bei Low-Pegel am Gate ist er gesperrt. Dieser Transistor ist groß dimensioniert und kann daher hohe Ströme nach Masse ziehen ( $I_{OL}$ , d.h.  $I_{Output\ Low}$ ;). Er wird nur dann eingeschaltet, wenn sich eine 0 im zugehörigen Latch befindet. In diesem Zustand muß allerdings auf jeden Fall ein Kurzschluß des Pins zur positiven Versorgungsspannung  $V_{CC}$  (= 5V) vermieden werden, da der Kurzschlußstrom (bis zu 100 mA!) den Baustein beschädigen könnte.
- Der FET p1 ist ein P-Kanal-Feldeffekttransistor vom Anreicherungstyp. Er verhält sich polaritätsmäßig genau andersherum wie der oben beschriebene N-Kanal-Transistor, d.h. bei High-Pegel am Gate sperrt er, bei Low-Pegel ist er durchgeschaltet. Dieser Transistor wird nur für die Dauer von zwei Oszillatortaktperioden aktiviert, wenn ein 0-1-Übergang am Portpin erzeugt werden soll. Das bedeutet, daß p1 nur dann einschaltet, wenn eine 1 in das Port-Latch geschrieben wird, das vorher eine 0 enthielt. Wird eine 1 auf eine bereits vorhandene 1 geschrieben, geschieht nichts. Die Treiberleistung dieses Transistors ist ähnlich dem Transistor n1. Diese Schaltung dient dazu, daß der Port auch bei der Ausgabe einer steigenden Flanke mit einer kurzen Anstiegszeit operiert, besonders wenn eine kapazitive Last anliegt.
- Der FET p2 ist ebenfalls ein P-Kanal-Feldeffekttransistor. Im Gegensatz zu p1 ist er allerdings schwach dimensioniert. Er ist immer dann aktiviert, wenn sich eine 1 im Port-

Latch befindet. Es ist problemlos möglich, diesen Transistor dauernd nach Masse kurzzuschließen; genau dies wird ja auch gemacht, wenn der Port als Eingang benutzt wird und Low-Pegel von extern angelegt wird. Der relativ geringe Strom, der dann fließt, ist  $I_{IL}$  (I Input Low). Typisch ist  $I_{IL}$  kleiner als  $30 \mu A$ .

- Der FET p3 ist ebenfalls ein P-Kanal-Feldeffekttransistor. Er ist dann eingeschaltet, wenn die Spannung am Portpin höher als etwa 1,0 bis 1,5 Volt ist. Damit unterstützt er den Transistor p2 in seiner Pull-Up-Wirkung, solange die Spannung am Pin die erwähnte Schwelle von 1,0 bis 1,5 Volt nicht unterschritten wird. Solange p3 aktiviert ist, kann der Strom  $I_{TL}$  (I Transition Low) nach Masse abgeleitet werden, ohne daß der Pegel absinkt. Dieser Strom ist deutlich höher als der Strom  $I_{IL}$ , der von p2 alleine aufgebracht werden kann (typisch ca  $250 \mu A$ ). Der Sinn dieser Schaltung liegt darin, daß beim Betrieb des Ports als Output auch der High-Pegel eine etwas höhere Ausgangstreiberleistung zur Verfügung steht. Wenn aber der Port als Input verwendet wird und Low-Pegel angelegt wird, ist nur noch p2 aktiv und es fließt ein deutlich niedriger Strom. Somit kann der Stromverbrauch der Gesamtschaltung verringert werden, da ja auch die Pull-Up-Ströme letztlich von der Stromversorgung aufgebracht werden müssen. Bei den in NMOS-Technologie gefertigten Varianten der 8051-Familie (alle Typen ohne C im Namen), ist übrigens diese Funktion nicht implementiert; bei diesen Bausteinen ist die zum Transistor p2 äquivalente Schaltung einfach deutlich stärker dimensioniert, während p3 entfällt. Damit ist dann die Treiberleistung bei High ähnlich wie bei den CMOS-Typen; beim Kurzschluß nach Masse fließt dann allerdings ein deutlich höherer Strom. Da die NMOS-Typen aber selbst schon einen wesentlich höheren Eigenstromverbrauch haben, ist dies nicht so störend.

Die obige Beschreibung trifft wie gesagt nur auf die Eingabe-Ausgabe-Ports 1 bis 6 zu. Port 0 und Port 2 weisen einige Besonderheiten auf. Im Gegensatz zu den anderen Ports wird hier streng zwischen I/O-Betrieb and Betrieb als Daten/Adreßbus unterschieden. Ein Multiplexer wird dazu von einem internen Steuersignal umgeschaltet, das immer aktiv wird, wenn ein externer Buszugriff erfolgen soll. Ist dieses Signal nicht aktiv (somit I/O-Betrieb) ist keinerlei Pull-Up aktiv. Damit liegt ein echt bidirektionaler Port vor, da er als Eingang (mit 1 im Latch) völlig hochohmig ist. Soll Port 0 dagegen als Ausgang benutzt werden, muß der High-Pegel durch externe Pull-Up-Widerstände erzeugt werden. Auch der Port 2 weist bei Benutzung als Daten/Adreßbus Besonderheiten auf-, siehe dazu Abschnitt 2 in diesem Kapitel.

## 10.2 Analog/Digitale Eingabeports

Die Ports 7 und 8 sind nur zu Eingabezwecken zu benutzen, können dabei aber sowohl für digitale als auch analoge Signale verwendet werden.

Generell sind diese Ports hochohmig, enthalten also keine Pull-Up-Schaltungen wie die EingabeAusgabe-Ports 1 bis 6. Beim Betrieb als digitaler Eingang können die angelegten Pegel (Low oder High) aus den entsprechenden Special-Function-Registern P7 und P8 gelesen werden. Ein Schreibvorgang auf diese Ports, die ja kein Latch enthalten, bleibt ohne Wirkung (siehe Bild 7-5).

<b>P7 (DBH), nicht bitadressierbar</b>							
<b>P8 (DDH), nicht bitadressierbar</b>							
MSB							LSB
<b>P7.7</b>	<b>P7.6</b>	<b>P7.5</b>	<b>P7.4</b>	<b>P7.3</b>	<b>P7.2</b>	<b>P7.1</b>	<b>P7.0</b>
				<b>P8.3</b>	<b>P8.2</b>	<b>P8.1</b>	<b>P8.0</b>

Die Portregister P7 und P8 dienen zum Lesen von den Portpins. Ein Schreibvorgang auf diese Register bleibt ohne Wirkung.

Gleichzeitig bilden diese Ports auch die Eingänge zum integrierten Analog/Digital-Wandler. Wenn Analogspannungen gemessen werden sollen, muß der entsprechende Eingangskanal durch Programmierung der Register **ADCON0** und **ADCON1** selektiert werden. Hierbei ist zu beachten, daß zwar der Port selbst nach wie vor hochohmig ist, der intern angeschlossene A/D-Wandler aber während seiner Sampling-Zeit eine gewisse kapazitive Last darstellt, die von der analogen Spannungsquelle umgeladen werden muß.

Die analoge oder digitale Funktion dieser Ports muß nicht eigens programmiert werden. Ob ein Portpin als Analog-Eingang benutzt wird oder sein digitaler Wert aus P7 oder P8 gelesen wird, ist daher nicht gesondert festzulegen. Es ist nur zu beachten, daß Eingangsspannungen, die zwischen den definierten digitalen High- und Low-Pegeln liegen, kein vorhersagbares Ergebnis für diese Bits in P7 und P8 erzeugen. Je nach tatsächlicher Schaltschwelle der Digitallogik entsteht dann eine 0 oder 1. Zwar ist es nach dem Gesagten grundsätzlich problemlos möglich, an die Ports 7 und 8 wahllos gemischt Pin für Pin analoge und digitale Eingangssignale anzulegen. Es ist aber dabei zu beachten, daß aus Gründen des Übersprechens besonders von digitalen auf (relativ stöempfindliche) analoge Signale, diese gesondert und nach Möglichkeit abgeschirmt an die entsprechenden Pins geführt werden; damit wird sich auch meist eine Separierung in eine Gruppe von nebeneinander liegenden digitalen un analogen Eingangspins ergeben.

### 10.3 Alternative Port-Funktionen

Viele Pins der Ports 1, 3, 4, 5 und 6 haben über die normale I/O-Funktion eine Erweiterung, um bei Bedarf für die auf dem Chip integrierten Peripheriekomponenten ebenfalls Eingabe- oder Ausgabefunktionen zur Verfügung zu stellen. Man nennt dies die alternativen Funktionen (alternate functions). Welcher Pin für welche Peripheriefunktion verwendet werden kann, zeigt nachfolgende Abbildung.

Den Aufbau der betreffenden Ports zeigt nachfolgende Abbildung. Es handelt sich hier nur um eine Erweiterung der oben diskutierten Portstruktur; es wird lediglich ein AND-Gatter in den Ansteuerpfad für die Ausgangstreiber eingefügt. Der eine Eingang dieses Gatters ist wie gewohnt der Ausgang des PortLatch. Der andere Eingang wird aus dem Ausgangssignal der jeweiligen Peripheriekomponente gespeist. Benötigt die Peripheriekomponente ein Eingangssignal, wird dies direkt hinter der Eingangsschaltung abgegriffen.

Solange die Peripheriekomponente nicht aktiv ist, ist ihr Ausgangssignal auf High-Pegel, somit steuert das Port-Latch über das AND-Gatter wie gehabt die Ausgangsstruktur. Wird durch entsprechende Programmierung die Peripheriekomponente eingeschaltet, wird sie ihre Ausgangssignale ebenfalls über das AND-Gatter an die Treiber schicken können, solange das

Port-Latch eine 1 enthält. Somit wird klar, daß keine gesonderte Umprogrammierung des Ports für die Alternativ-Funktion nötig ist. Es muß nur eine 1 im Port-Latch sein, wenn die entsprechende Peripheriekomponente aktiviert wird. Andernfalls bleibt der Pin dauernd auf 0. Die 1 im Port-Latch ist übrigens auch der Zustand nach Reset. Damit ist die Alternativfunktion also ohne weitere Programmierung des Ports möglich. Eine Besonderheit weisen die Alternativ-Funktionen der Compare-Ausgangspins auf. Hier muß nicht zwingend eine 1 im Latch sein. Die Peripheriefunktion programmiert das Latch selbst um.

Keine Kollision ist beim Eingangssignal möglich. Hier können sowohl die Peripheriekomponente als auch die normale Input-Funktion bedient werden (sogar gleichzeitig).

Portpin	Alternativfunktion
P1.0/INT3#/CC0	Ext. Interrupt 3, Comp./Capt.-Pin 0
P1.1/INT4/CC1	Ext. Interrupt 4, Comp./Capt.-Pin 1
P1.2/INT5/CC2	Ext. Interrupt 5, Comp./Capt.-Pin 2
P1.3/INT6/CC3	Ext. Interrupt 6, Comp./Capt.-Pin 3
P1.4/INT2#/CC4	Ext. Interrupt 2, Comp./Capt.-Pin 4
P1.5/T2EX	Externe Reload-Anforderung für Timer 2
P1.6/CLKOUT	Systemtakt-Ausgang
P1.7/T2	Externer Eingang für Timer 2
P3.0/RXD0	Eingang der ser. Schnittstelle 0
P3.1/TXD0	Ausgang der ser. Schnittstelle 0
P3.2/INT0#	Ext. Interrupt 0
P3.3/INT1#	Ext. Interrupt 1
P3.4/T0	Externer Eingang für Timer 0
P3.5/T1	Externer Eingang für Timer 1
P3.6/WR#	Bussteuersignal Write für externen Datenspeicher
P3.7/RD#	Bussteuersignal Read für externen Datenspeicher
P4.0/CM0	Comp.-Ausgang von Register CM0
P4.1/CM1	Comp.-Ausgang von Register CM1
P4.2/CM2	Comp.-Ausgang von Register CM2
P4.3/CM3	Comp.-Ausgang von Register CM3
P4.4/CM4	Comp.-Ausgang von Register CM4
P4.5/CM5	Comp.-Ausgang von Register CM5
P4.6/CM6	Comp.-Ausgang von Register CM6
P4.7/CM7	Comp.-Ausgang von Register CM7
P5.0/CCM0	Concurrent-Compare-Ausgang 0
P5.1/CCM1	Concurrent-Compare-Ausgang 1
P5.2/CCM2	Concurrent-Compare-Ausgang 2
P5.3/CCM3	Concurrent-Compare-Ausgang 3
P5.4/CCM4	Concurrent-Compare-Ausgang 4
P5.5/CCM5	Concurrent-Compare-Ausgang 5
P5.6/CCM6	Concurrent-Compare-Ausgang 6
P5.7/CCM7	Concurrent-Compare-Ausgang 7
P6.0/ADST#	Externer Start des A/D-Wandlers
P6.1/RXD1	Eingang der ser. Schnittstelle 1
P6.2/TXD1	Ausgang der ser. Schnittstelle 1

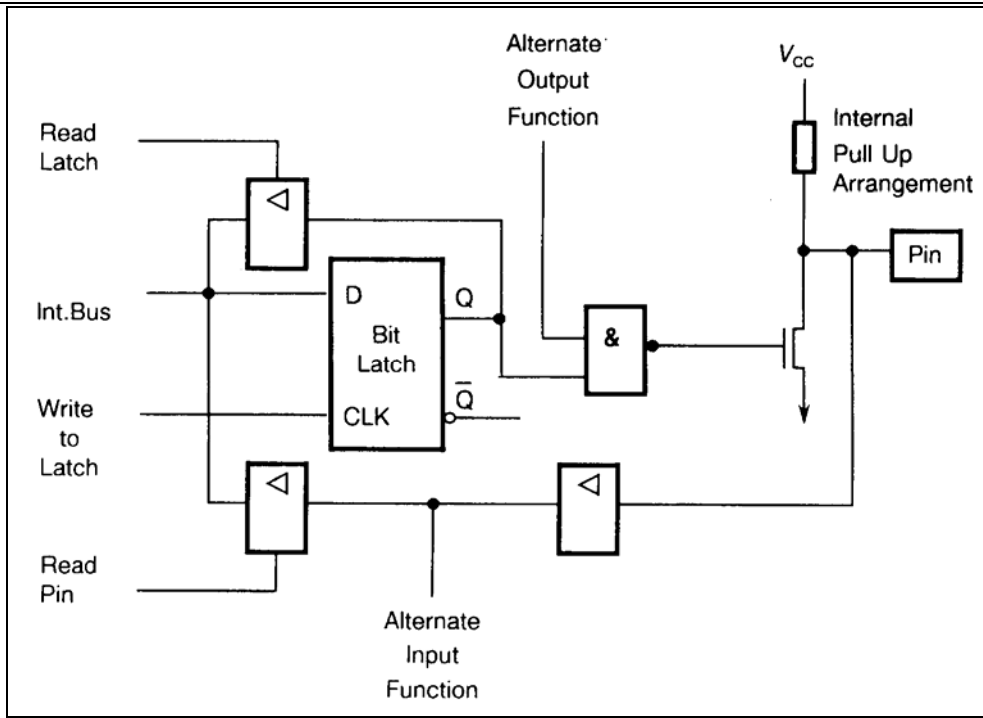


Abb. Portstruktur bei Alternativfunktion



## 11 Timer

Der 80C517/80C517A hat mehrere Timer: Timer 0 und 1, sowie Timer 2 und den Compare Timer. Die beiden hier besprochenen Timer 0 und 1 sind voll kompatibel zu den beiden Timern im 8051 und können daher mit derselben Programmierung in denselben Betriebsarten genutzt werden.

Beide Timer können grundsätzlich in allen Betriebsarten entweder als "Timer" oder als "Counter" benutzt werden. (Die Begriffe sind leider etwas verwirrend, da "Timer" einmal als Bezeichnung für die Einheit verwendet wird, und zum anderen damit eine von zwei Funktionsmöglichkeiten bezeichnet wird. Die manchmal verwendete Übersetzung Zähler/Zeitgeber ist allerdings auch keine glücklichere Lösung.) Die Timer werden generell inkrementiert, d.h. sie zählen aufwärts.

- ➔ Timer-Funktion bedeutet, daß das Zählregister vom entsprechend geteilten Oszillortakt regelmäßig inkrementiert wird. Bei Timer 0 und 1 beträgt der Teilfaktor 12. Damit erfolgt praktisch eine Zählung der Maschinenzyklen, da die Zählrate  $1/12$  des Oszillortakts ist. Ein Sonderfall ist die Gated-Timer-Funktion; hier arbeitet die Schaltung wie ein Timer, allerdings nur, während durch die Aktivierung eines externen Pins ein Gatter freigeschaltet wird, das sonst das Zählen unterdrücken kann.
- ➔ Counter-Funktion bedeutet, daß das Zählregister immer dann inkrementiert wird, wenn ein 1-0-Pegelwechsel (fallende Flanke) an den entsprechenden Zählleitungen (TO-Alternativfunktion von P3.4 bzw. TI-Alternativfunktion von P3.5) erkannt wurde. Zur Flankenerkennung wird der Pin jeweils zum Zeitpunkt S5P2 abgetastet. Wenn die Abtastungen einen High-Pegel und im darauffolgenden Zyklus einen Low-Pegel ergeben, wird der Zählerstand inkrementiert. Der neue Wert erscheint dann zu S3P1 des folgenden Maschinenzyklus im Zählregister. Es erfordert mindestens zwei Maschinenzyklen (24 Oszillortaktperioden), um einen 1-0-Pegelwechsel zu erkennen (High in einem Zyklus, Low im darauffolgenden Zyklus). Daher ist die höchstmögliche Eingangsfrequenz für die Counter-Funktion  $1/24$  der Oszillatorfrequenz. Andernfalls können Zählsignale verlorengehen. Das Tastverhältnis des Eingangssignals spielt keine Rolle, solange jeder Pegel mindestens einen Maschinenzyklus anliegt, damit er sicher erkannt wird. Ein Sonderfall ähnlich wie die schon erwähnte Gated-Timer-Funktion ist eine allerdings selten genutzte Gated-Counter-Funktion; hier arbeitet die Schaltung wie ein Counter, allerdings nur, während durch die Aktivierung eines externen Pins ein Gatter freigeschaltet wird, das sonst das Zählen unterdrücken kann.

Beide Timer können in vier Betriebsarten (Modi) benutzt werden. Jeder Timer besteht aus zwei 8bit-Registern: TLO (Timer 0 Low; 8AH) und THO (Timer 0 High; 8CH) bei Timer 0, TLI (Timer 1 Low; 8BH) und THI (Timer 1 High; 8DH) bei Timer 1; diese Register werden abhängig von den Betriebsarten unterschiedlich verwendet. Die Modusauswahl und Steuerung beider Timer erfolgt über die Special-Function-Register TCON (Timer Control, 88H) und TMOD (Timer Modus, 89H).

Da die Modi 0, 1 und 2 in beiden Timern völlig identisch arbeiten, wird in den folgenden Beschreibungen nur Timer 0 dargestellt und beschrieben. In Modus 3 bestehen Unterschiede, deshalb wird in diesem Fall getrennt beschrieben.

**TCON (88H), bitadressierbar**

MSB				LSB			
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
8FH	8EH	8DH	8CH	8BH	8AH	89H	88H
Timer-Steuerregister (Timer Control). Es enthält Steuerbits für die Timer 0 und 1 sowie für die externen Interrupts. Reset-Wert: 00H							
Bitsymbol	Funktion						
TF1	Timer 1-Überlaufflag. Es wird von der Hardware bei einem Überlauf des Timers 1 gesetzt. Beim Einsprung in die zugehörige Interrupt-Adresse wird es automatisch gelöscht.						
TR1	Freigabebit für Timer 1 (Timer 1 Run Flag). Es muß durch Software gesetzt oder gelöscht werden, um den Timer 1 zu starten bzw. anzuhalten.						
TF0	Timer 0 Überlaufflag. Es wird von der Hardware bei einem Überlauf des Timers 0 gesetzt. Beim Einsprung in die zugehörige Interrupt-Adresse wird es automatisch gelöscht.						
TR0	Freigabebit für Timer 0 (Timer 0 Run Flag). Es muß durch Software gesetzt oder gelöscht werden, um den Timer 0 zu starten bzw. anzuhalten.						

### 11.1 Modus 0

Wählt man den Modus 0 für Timer 0 oder 1, so wird er als 8 bit breiter Timer mit einem Vorteiler/32 betrieben. Dies in einem Funktionsschaltbild dargestellt. Der Grund, warum dieser auf den ersten (und auch zweiten) Blick etwas eigenartige Modus implementiert ist, liegt in der von den 8051 -Entwicklern gewünschten Kompatibilität zur Vorgängerfamilie, der 8048-Familie. Dort war ein (einziger!) 8-bit-Timer mit entsprechender Vorteilung integriert. Die Absicht war gewesen, ein möglichst einfaches Portieren von 8048-Programmen auf den 8051 zu ermöglichen.

**TMOD (89H), nicht bitadressierbar**

MSB				LSB			
Timer 1				Timer 0			
Gate	C/T#	M1	M0	Gate	C/T#	M1	M0
Modusregister für Timer 0 und 1 (Timer Mode). Es enthält die Bits zur Einstellung der Betriebsarten für die Timer 0 und 1. In den Erläuterungen steht x für 0 oder 1, je nachdem, welcher Timer verwendet wird. Reset-Wert: 00H							
Bitsymbol		Funktion					
Gate		Wenn dieses Bit gesetzt ist, läuft der Timer nur dann, wenn er mit TRx freigegeben ist und gleichzeitig der Portpin P3.2/INT0# (Timer 0) bzw. P3.3/INT1# (Timer 1) auf High-Pegel ist.					
C/T#		Dieses Bit legt fest, ob Timer- oder Counter-Modus verwendet wird. C/T# = 0 wählt Timer- und C/T# = 1 wählt Counter-Modus.					
M0	M1	Arbeitsmodi					
0	0	THx dient als 8-bit-Timer; TLx bildet einen 5-bit-Vorteiler.					
0	1	THx und TLx bilden einen 16-bit-Timer.					
1	0	Auto-Reload-Timer (8 bit). Der Inhalt von THx wird beim Timerüberlauf nach TLx kopiert. THx selbst bleibt unverändert.					
1	1	Timer 0: Beide Register THx und TLx arbeiten als eigenständige 8-bit-Timer. TLx wird durch die Steuerbits von Timer 0, THx durch die Steuerbits von Timer 1 eingestellt. Timer 1: Timer 1 stoppt in dieser Betriebsart.					

Die Timerregister werden zu einem 13 bit breiten Register kombiniert. Dazu werden alle 8 Bit von TH0 und die unteren 5 Bit von TLO verwendet. Die oberen 3 Bit von TLO sind in diesem Modus undefiniert und sollten nicht durch den Programmierer verwendet werden.

Die Unterscheidung zwischen Timer- und Counterfunktion wird durch das Steuerbit C/T# (im Register TMOD) getroffen; eine 1 in C/T# wählt Counter aus, während eine 0 die Timer-Funktion selektiert. Der Timer kann (in beiden Funktionen) zählen, wenn das Timer-Run-Bit TRO (in Register TCON) auf 1 gesetzt ist; das Setzen von TRO gibt den Timer frei, löscht aber nicht die Zählregister. Damit läßt sich mit diesem Bit der Zählvorgang beliebig unterbrechen und wieder aufnehmen.

Die Gated-Timer/Counter-Funktion wird durch das Steuerbit GATE in Register TMOD aktiviert; ist Gate = 1, so kann nur dann gezählt werden, wenn der externe Pin P3.2/INT0# auf 1 gehalten wird. Die Bezeichnung INTO# (Alternativfunktion von P3.2) soll hier nicht verwirren; dies zeigt lediglich die Vielfachbelegung der Portpins an; obwohl die hier beschriebene Funktion für das Gate nicht direkt mit den Interrupts zu tun hat, kann übrigens trotzdem der Pin gleichzeitig für das Gate und die Interrupt-Auslösung benutzt werden, wenn das in der Anwendung sinnvoll ist.

Wenn der Zähler von seinem Maximalwert (1 in allen Bits) auf OOH überläuft, setzt der Timer das Überlaufflag TFO. Dieses Flag wird auch von der Interrupt-Logik als Interrupt-Request-Flag verwendet (siehe Kapitel 17 für Details über die Interrupts).

Wie schon erwähnt, ist die Betriebsart 0 auch für Timer 1 in gleicher Weise verfügbar. Es müssen natürlich die entsprechenden Steuer- und Statusflags von Timer 1 verwendet werden (TRI, TFI, TL 1, TH 1, P3.3/INT 1 # sowie C/T# und GATE für Timer 1).

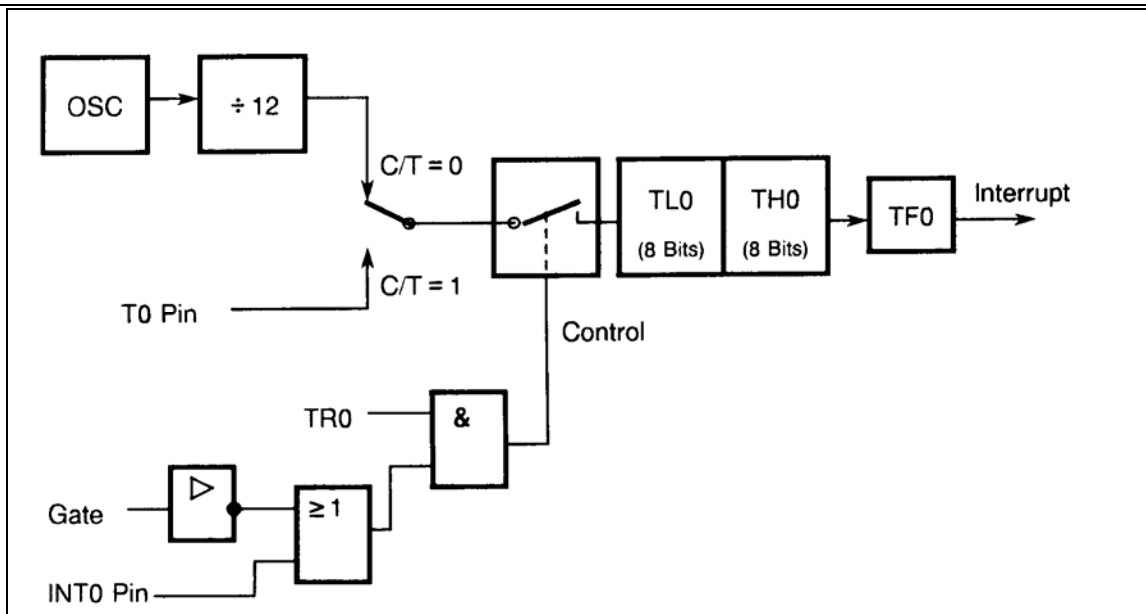


Abb. Timer 0/1, Modus 0: 13-bit-Timer/Counter

## 11.2 Modus 1

In Modus 1 wird der Timer genauso betrieben wie im eben beschriebenen Modus 0. Der einzige Unterschied liegt in der vollen Ausnutzung der beiden Timerregister TLO und THO, die zu einem einzigen 16-bit-Zählerregister zusammengeschaltet werden. Damit steht dann ein 16-bit-Timer zur Verfügung. zeigt das Funktionsschaltbild.

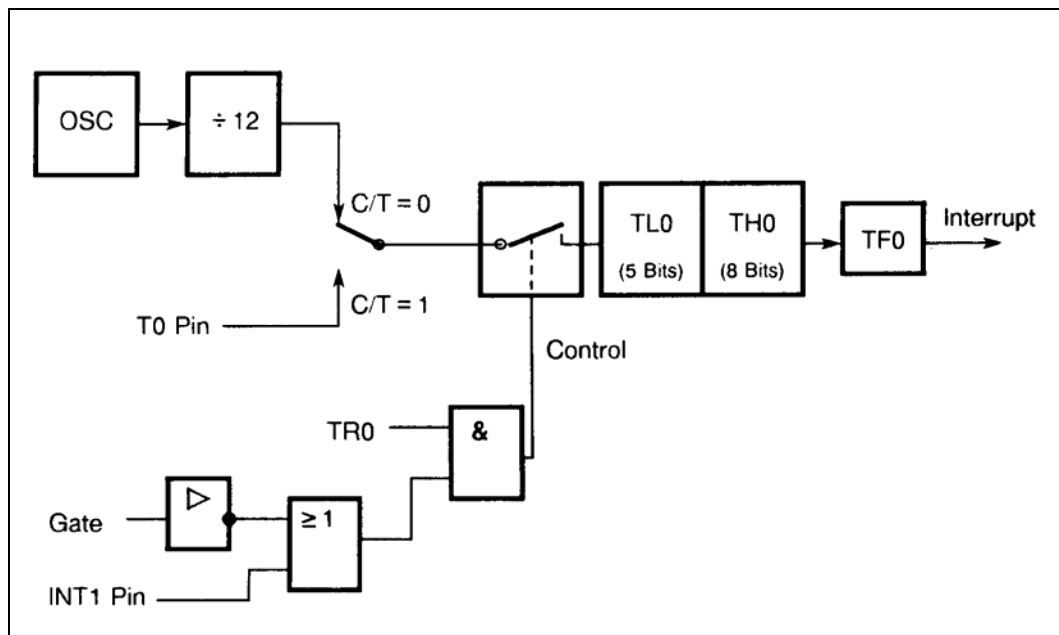


Abb. Timer 0/1, Modus 1: 16-bit-Timer/Counter

Es ist übrigens generell möglich, die Register TLO und THO auch während des Betriebs zu schreiben und zu lesen (in allen Modi). Man muß dann allerdings genau auf verschiedene Nebeneffekte achten, um auch das gewünschte Ergebnis zu erzielen; das Problem liegt darin, daß man nur mit 8-bit-Schreib- oder Leseoperationen auf das 16 bit breite Timerregister arbeiten kann. Ein Beispiel soll den Vorgang verdeutlichen: Man will den 16-bit-Wert aus dem

laufenden Timer 0 lesen. Zuerst liest man das TH0-Register und legt den Inhalt irgendwo ab. Anschließend liest man das TLO-Register und legt den Inhalt ebenso ab. Dabei kann aber der Timer zwischen den beiden Lesevorgängen einen Übertrag vom Low- ins High-Register gehabt haben; damit gehört der gelesene High-Wert gar nicht mehr zum später gelesenen Low-Wert! Ähnliche Probleme können auch beim Schreiben in den laufenden Timer auftreten, so daß man diesen Schwierigkeiten am besten dadurch aus dem Weg geht, daß man den Timer vorher stoppt (wenn in der Applikation möglich). Sonst muß man sich durch verschiedene Tricks behelfen, wie es auch einen für das eben geschilderte Beispiel gibt: Man führt das Lesen durch wie oben beschrieben; nachträglich liest man aber nochmals das TH0-Register und vergleicht den Wert mit dem anfangs gelesenen. Sind die Werte unterschiedlich, ist genau der Fehler eingetreten; dann muß man den gesamten Vorgang wiederholen, in der Hoffnung, daß man diesmal keinen Übertrag zwischen den Registern erwischt.

Die Betriebsart 1 ist analog auch in Timer 1 mit dessen eigenen Steuer- und Statusflags vorhanden.

### 11.3 Modus 2

In Modus 2 werden die beiden Register TLO und TH0 nicht als ein Zählregister zusammenschaltet. Vielmehr wird nur TLO als 8-bit-Zählregister verwendet. TH0 dient als Speicher für einen 8-bit Reloadwert. Beim Überlauf von TLO wird der dort liegende Wert wieder nach TLO eingeschrieben; damit läßt sich der Zeitraum zwischen den Überläufen entsprechend verkürzen. Die sonstigen Funktionen des Timers (Timer/Counter-Funktion, Gate, Interrupt-Request-Flag, etc.) entsprechen den Modi 0 und 1. Üblicherweise wird Modus 2 dazu verwendet, regelmäßig einen Interrupt zu erzeugen, um darin dann irgendwelche Aktionen auszuführen, die in gleichmäßigen Abständen benötigt werden (z.B. Tatstaturabfrage). Die geeignete Wahl des Reloadwerts erlaubt es dann, den Abstand zwischen solchen Aktionen festzulegen.

Die Betriebsart 2 ist analog auch in Timer 1 mit dessen eigenen Steuer- und Statusflags vorhanden.

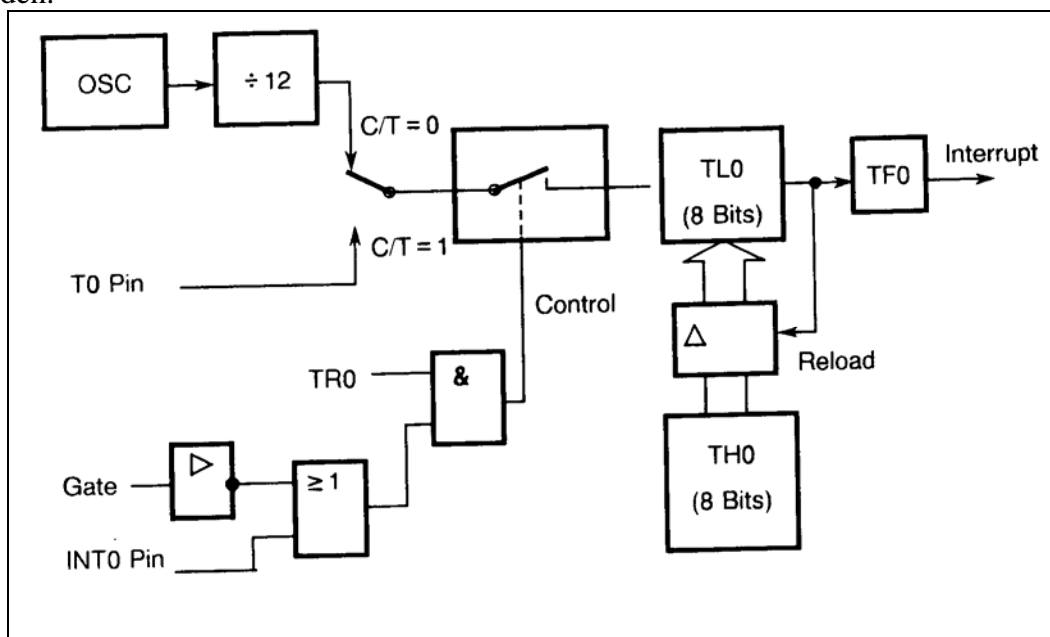


Abb. Timer 0/1, Modus 2: 8-bit-Timer/Counter mit Reload

## 11.4 Modus 3

In Modus 3 verhalten sich Timer 0 und 1 unterschiedlich, siehe nachfolgende Abbildung..

### Timer 0

In Modus 3 wird Timer 0 in zwei unabhängige 8-bit-Timer zerlegt. Das Register TLO arbeitet mit den von Modus 0, 1 und 2 bekannten Möglichkeiten (Timer/Counter-Funktion, Gate, Interrupt-Request-Flag TFO). Das Register TH0 wird fest in Timer-Funktion (Zählen der internen Maschinentzyklen) betrieben; als Steuerflag wird TRI verwendet, das sonst Timer 1 zugeordnet ist. Als InterruptRequest-Flag wird TFI benutzt (sonst auch Timer 1 zugeordnet). Damit kann TH0, wenn auch eingeschränkt, wie ein 8 bit breiter Timer 1 betrieben werden.

### Timer 1

Timer 1 stoppt, wenn er in Modus 3 versetzt wird. Die Wirkung ist dieselbe, als wenn TRI auf 0 rückgesetzt würde.

Was ist die Anwendung dieser Betriebsart? Sie wurde erfunden, als zu Zeiten des 8051 die Timer extrem knapp waren. Man gewinnt in der Betriebsart 3 einen dritten Timer aus dem Timer 0/1 Block (zwar nur 8 bit breit, aber besser als gar nichts). Allerdings hat dann Dimer 1 selbst keinen Interrupt mehr, aber wenn man ihn zur Baudratengenerierung braucht, verwendet man sowieso den Interrupt nicht. Das Problem des fehlenden Run-Bits TRI wird durch den Modus 3 für Timer 1 gelöst, der damit angehalten werden kann.

Diese Problematik ist beim 80C517 inzwischen entschärft, da hier viel mehr Timer zur Verfügung stehen und außerdem zur Baudratenerzeugung oft auf eigene Generatoren zurückgegriffen werden kann. Aus Kompatibilitätsgründen ist aber alles wie beim 8051 implementiert, was die Übernahme bestehender Programme stark vereinfacht.

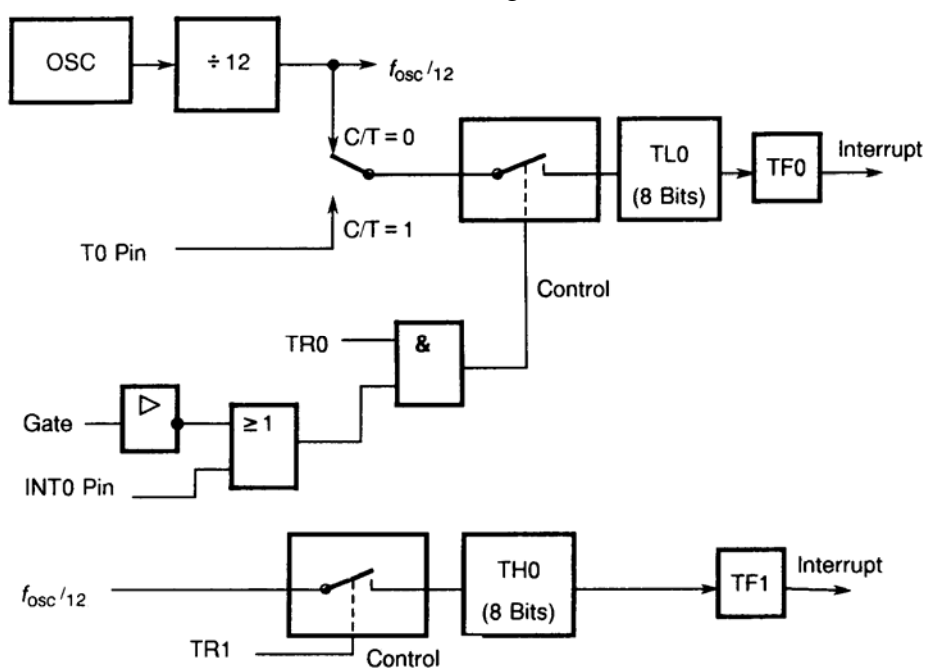


Abb. Timer 0, Modus 3: Zwei 8-bit-Timer/Counter

## 12 Capture/Compare-Einheit

### 12.1 Begriffserklärung

Im wesentlichen besteht die Compare/Capture-Einheit aus zwei Timer-Einheiten und insgesamt 13 Compare/Capture Registern. Ein Satz von 6 Steuerregistern erlaubt die Konfiguration der Compare/Capture-Einheit für die jeweilige Anwendung. Die 13 Register sind folgend eingeteilt:

- ⊗ ein 16-bit-Compare/Reload/Capture-Register (CRC)
- ⊗ drei 16-bit-Compare/Capture-Register (CC1 ... CC3)
- ⊗ ein 16-bit-Compare/Capture-Register (CC4) mit zusätzlicher  
Concurrent-Compare-Betriebsart.
- ⊗ acht 16-bit-Compare Register (CM0 ... CM7)

Je nach Konfiguration bieten die den beiden Timern T2 und CT (Compare Timer) angeschlossenen Register unterschiedliche Funktionen.

#### 12.1.1 Compare

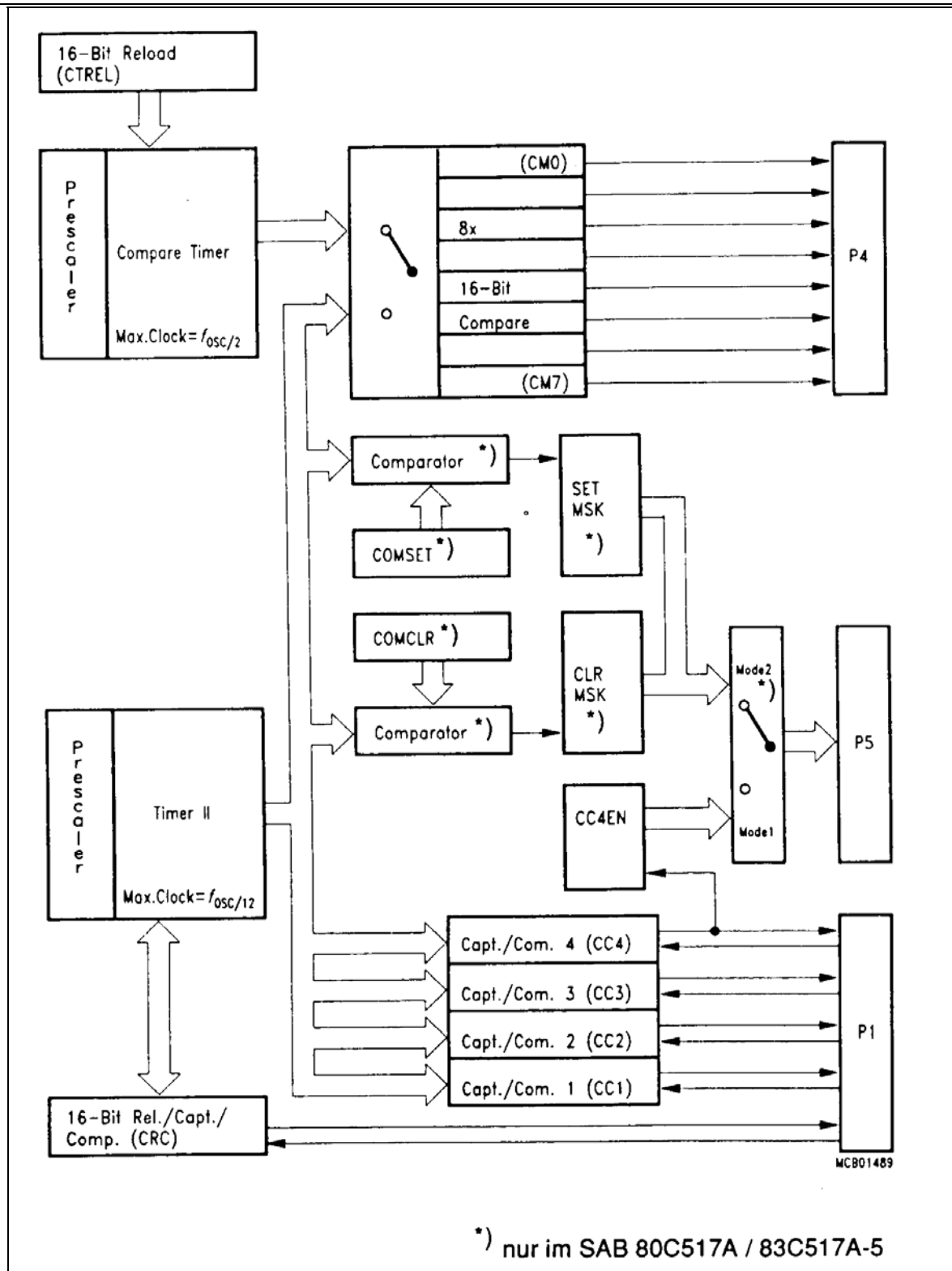
Der Inhalt des zugeordneten Timers wird ständig mit dem Inhalt des Compare-Registers verglichen. Bei Übereinstimmung werden ein oder mehrere Ereignisse (je nach Konfiguration) ausgelöst. Damit kann durch geeignete Wahl des Compare-Werts der Zeitpunkt für den Eintritt des (der) Ereignisse(s) präzise festgelegt werden.

#### 12.1.2 Reload

Nach einem Überlauf läuft der Timer nicht mit 0000H weiter, sondern mit dem im jeweiligen Reload-Register abgelegten Nachladewert. Dadurch kann die Timer-Periode (die Zeit von Überlauf zu Überlauf) über den Reload-Wert variiert werden.

#### 12.1.3 Capture

Auf ein bestimmtes Ereignis hin wird der Inhalt des laufenden Timers in das Capture-Register übernommen. Dadurch läßt sich der Zeitpunkt des Ereigniseintritts exakt festhalten.



Die Compare/Capture-Einheit des 80C517/80C517A



Konfigurationsmöglichkeiten der Capture/Compare-Einheit:

Timer	Compare-Register	Compare-Ausgang	Compare-Modi
Timer 2	CRCH/CRCL	P1.0/INT3#/CC0	Modus 0 und 1, Reload
	CCH1/CCL1	P1.1/INT4/CC1	Modus 0 und 1
	CCH2/CCL2	P1.2/INT5/CC2	Modus 0 und 1
	CCH3/CCL3	P1.3/INT6/CC3	Modus 0 und 1
	CCH4/CCL4	P1.4/INT2#/CC4	Modus 0 und 1
	CCH4/CCL4	P5.0/CCM0 bis P5.7/CCM7	Concurrent-Compare-Modus
	COMSET/COMCLR	P5.0/CCM0 bis P5.7/CCM7	Set/Reset-Modus (nur im 80C517A)
	CMH0/CML0	P4.0/CM0	Modus 1
	CMH1/CML1	P4.1/CM1	Modus 1
	CMH2/CML2	P4.2/CM2	Modus 1
	CMH3/CML3	P4.3/CM3	Modus 1
	CMH4/CML4	P4.4/CM4	Modus 1
	CMH5/CML5	P4.5/CM5	Modus 1
	CMH6/CML6	P4.6/CM6	Modus 1
CMH7/CML7	P4.7/CM7	Modus 1	
Compare-Timer	CMH0/CML0	P4.0/CM0	Modus 0
	CMH1/CML1	P4.1/CM1	Modus 0
	CMH2/CML2	P4.2/CM2	Modus 0
	CMH3/CML3	P4.3/CM3	Modus 0
	CMH4/CML4	P4.4/CM4	Modus 0
	CMH5/CML5	P4.5/CM5	Modus 0
	CMH6/CML6	P4.6/CM6	Modus 0
	CMH7/CML7	P4.7/CM7	Modus 0

Bild 11-9: Konfigurationsmöglichkeiten der CCU

**CTCON (E1H), nicht bitadressierbar**

MSB				LSB			
T2PS1	-	ICR/-	ICS/-	CTF	CLK2	CLK1	CLK0
Compare-Timer-Steueregister (Compare Timer Control Register). Es enthält die Auswahlbits für den Zähltakt des Compare-Timers und des Timers 2 sowie mehrere Interrupt-Request-Flags. Reset-Wert: 0xxx 0000B (80C517), 0x00 0000B (80C517A)							
Bitsymbol	Funktion						
T2PS1	Timer-2-Vorteilerbit 1 (Timer 2 Prescaler Bit 1). Zusammen mit T2PS (T2CON.7) läßt sich die Eingangsfrequenz und damit der Zähltakt des Timer 2 nach der Tabelle in Bild 11-5 einstellen.						
CTF	Compare-Timer-Überlaufflag. Dieses muß durch Software gelöscht werden. Falls der Interrupt des Compare-Timers freigegeben ist, löst CTF = 1 einen Interrupt aus.						
CLK2 CLK1 CLK0	Auswahl des Zähltakts für den Compare-Timer (siehe nachstehende Tabelle):						
	CLK2	CLK1	CLK0	Funktion			
	0	0	0	Zähltakt für den Compare-Timer ist $f_{osz}/2$			
	0	0	1	Zähltakt für den Compare-Timer ist $f_{osz}/4$			
	0	1	0	Zähltakt für den Compare-Timer ist $f_{osz}/8$			
	0	1	1	Zähltakt für den Compare-Timer ist $f_{osz}/16$			
	1	0	0	Zähltakt für den Compare-Timer ist $f_{osz}/32$			
	1	0	1	Zähltakt für den Compare-Timer ist $f_{osz}/64$			
	1	1	0	Zähltakt für den Compare-Timer ist $f_{osz}/128$			
	1	1	1	Zähltakt für den Compare-Timer ist $f_{osz}/256$			

Bild 11-8: Special-Function-Register CTCON (E1H)

Compare-Modus mit Timer 2

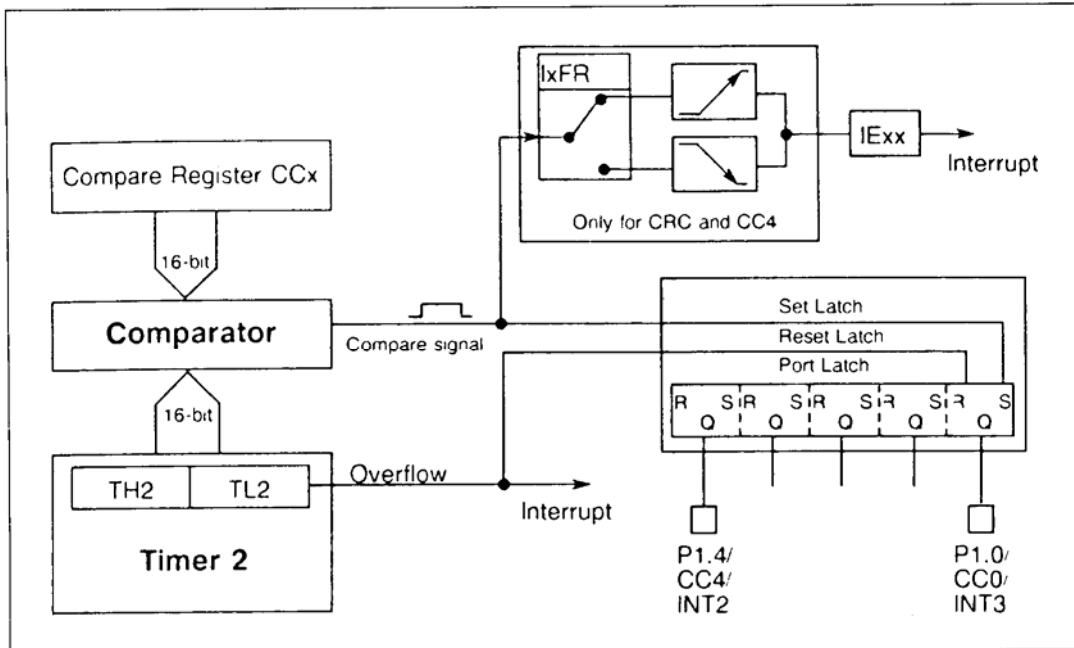


Bild 11-12: Timer 2 mit Registern CRC/CC1 bis CC4 im Compare-Modus 0

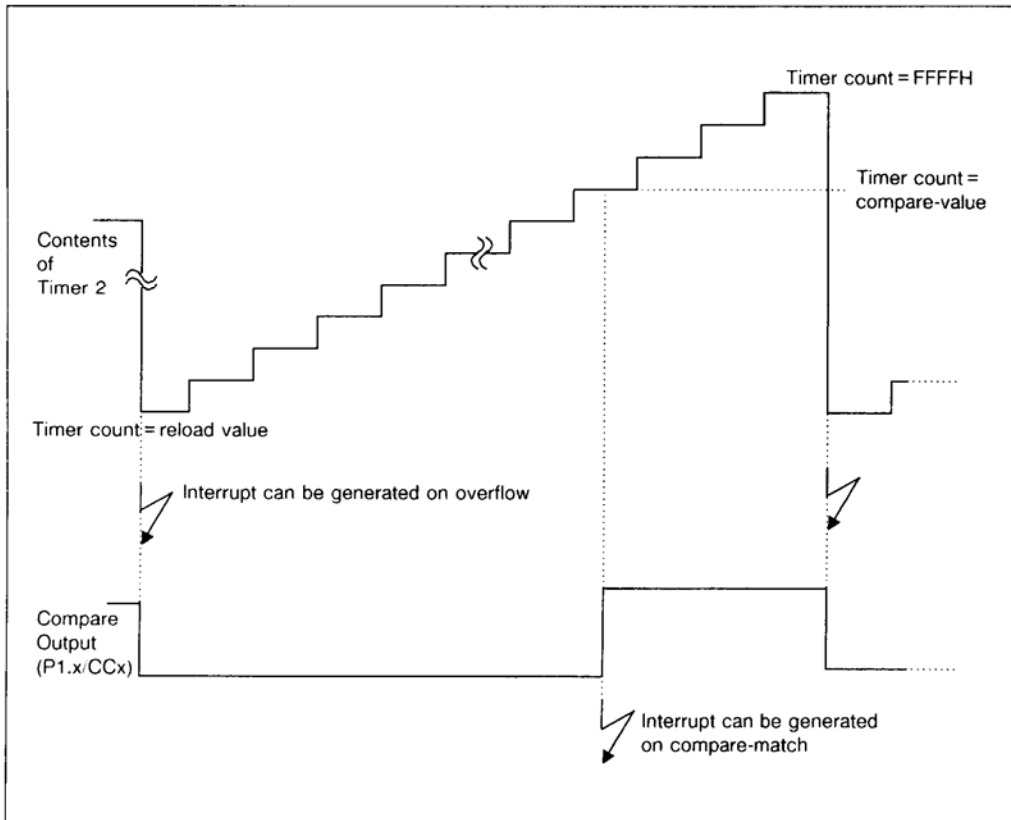


Bild 11-11: Funktion des Compare-Modus 0

Compare-Modus 0 mit Compare-Timer und Register CMx

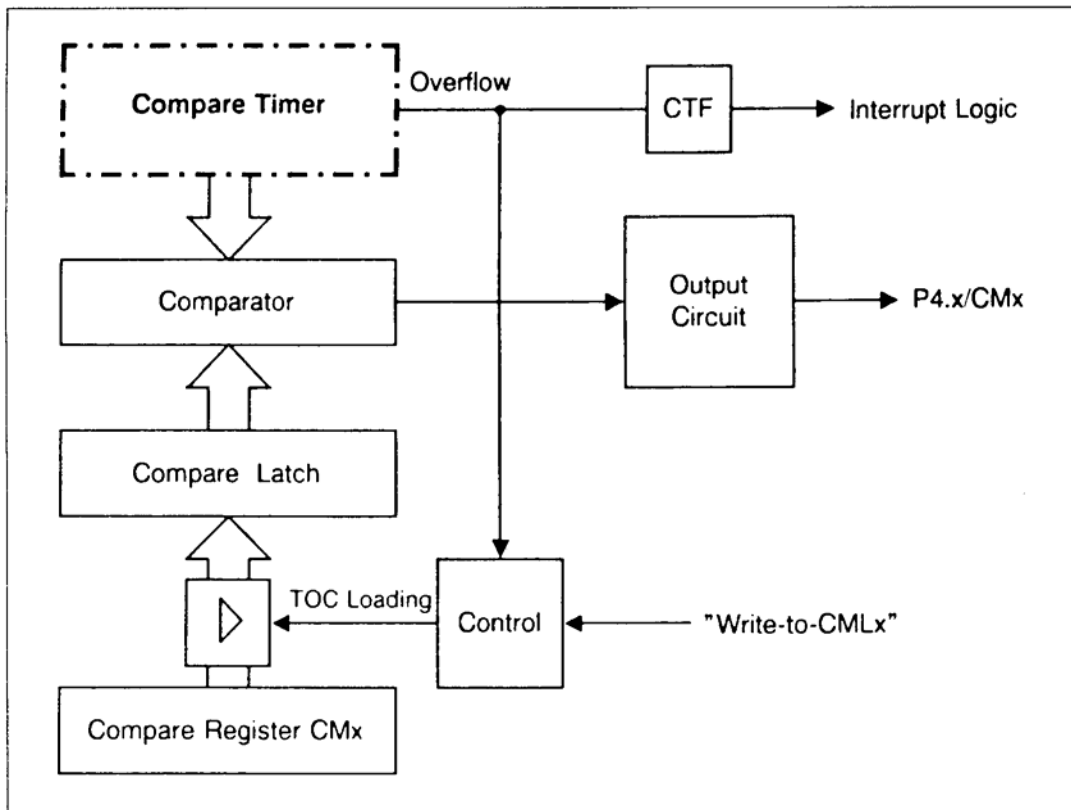


Bild 11-17: Compare-Modus 0 mit Compare-Timer und Register CMx

## 12.2 Anwendungsbeispiel PWM

Mit Hilfe der Capture Compare Einheit läßt sich einfach eine Pulsweitenmodulation aufbauen. Dazu verwendet man den Compare Timer.

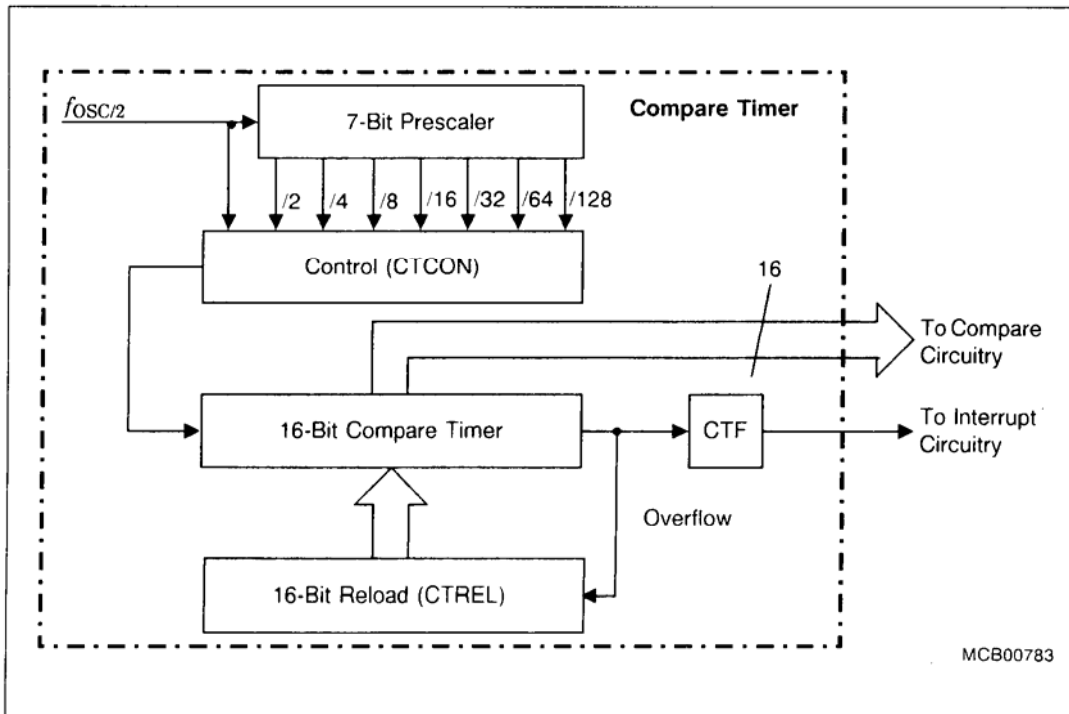


Bild 11-7: Blockschaltbild des Compare-Timers

### Compare Timer

Mit Hilfe des sfr **CTCON** wird die Zählfrequenz eingestellt. In das sfr **CTRELL** und **CTRELH** werden die reload Werte eingeschrieben. Für eine Genauigkeit von 10 bit muß **CTRELH = 0xFC** eingegeben werden. **CTRELL = 0x00**. Damit lassen sich 1024 verschiedene Pulsweiten einstellen.

Der Zählerstand wird laufend mit einem anderen Register, dem Register **CMx** ( $x = 0..7$ ), verglichen. Ist der Zählerstand des Registers kleiner als der des Compare Timers, so wird am Portpin **P4\_x** eine 1 ausgegeben.

Das Register wird über das sfr **CMSEL** ausgewählt. Das Übungsboard soll auf Pin P4\_0 ausgeben, daher ist **CMSEL = 0x01**.

Die Initialisierung für die PWM ist in der Routine **PWM\_INIT** programmiert. Das setzen eines neuen Werts wird mit der Funktion **PWM\_VAL** durchgeführt.

Verwendete SFR:

**CMEN (F6H), nicht bitadressierbar**

MSB							LSB
CMEN7	CMEN6	CMEN5	CMEN4	CMEN3	CMEN2	CMEN1	CMEN0
Freigabe für CM-Register (CM-Register Enable). Bei Setzen des entsprechenden Bits wird die Compare-Funktion des jeweiligen CM-Registers freigegeben und dessen Ausgangssignal an den Portpin geleitet. Reset-Wert: 00H							
Bitsymbol	Funktion						
CMEN7	Freigabe-Bit für Register CM7						
CMEN6	Freigabe-Bit für Register CM6						
CMEN5	Freigabe-Bit für Register CM5						
CMEN4	Freigabe-Bit für Register CM4						
CMEN3	Freigabe-Bit für Register CM3						
CMEN2	Freigabe-Bit für Register CM2						
CMEN1	Freigabe-Bit für Register CM1						
CMEN0	Freigabe-Bit für Register CM0						

Bild 11-16: Special-Function-Register CMEN (F6H)

**CMSEL (F7H), nicht bitadressierbar**

MSB							LSB
CMSEL7	CMSEL6	CMSEL5	CMSEL4	CMSEL3	CMSEL2	CMSEL1	CMSEL0
Auswahl der CM-Register (CM Register Selection). Das Register legt fest, mit welchem Timer das jeweilige CM-Register arbeiten soll. CMSELx = 1 schlägt das Register CMx dem Compare-Timer zu und läßt es im Compare-Modus 0 arbeiten (PWM-Modus). CMSELx = 0 ordnet das Register CMx dem Timer 2 zu und läßt es im Compare-Modus 1 arbeiten. Reset-Wert: 00H							
Bitsymbol	Funktion						
CMSEL7	Auswahlbit für Register CM7						
CMSEL6	Auswahlbit für Register CM6						
CMSEL5	Auswahlbit für Register CM5						
CMSEL4	Auswahlbit für Register CM4						
CMSEL3	Auswahlbit für Register CM3						
CMSEL2	Auswahlbit für Register CM2						
CMSEL1	Auswahlbit für Register CM1						
CMSEL0	Auswahlbit für Register CM0						

Bild 11-15: Special-Function-Register CMSEL (F7H)

```
void pwm_init(unsigned int val) /*Compare mode 0, val = 0 - 1023*/
{
    CTCON = 0x00;          /* f_OSC/2 (default)          */
    CMSEL = 0x01;          /* Assign CM0 to the compare timer */
    CTRELH = 0xFC;         /* 10 bit reload -> 1024 steps     */
    CTRELL = 0x00;         /* Start TOC loading               */

    if (val >= 1024) {
        CMH0 = 0xFF;
        CML0 = 0xFF;
    } else if (val <= 0) {
        CMH0 = CTRELH;
        CML0 = CTRELL;
    } else {
        CMH0 = CTRELH + (unsigned char)(val / 256);
        CML0 = CTRELL + (unsigned char)(val % 256);
    }
    CMEN = 0x01;
}

void pwm_val(unsigned int val)
{
    if (val >= 1024) {
        CMH0 = 0xFF;
        CML0 = 0xFF;
    } else if (val <= 0) {
        CMH0 = CTRELH;
        CML0 = CTRELL;
    } else { /* load new PWM value into CM0 */
        CMH0 = CTRELH + (unsigned char)(val / 256);
        CML0 = CTRELL + (unsigned char)(val % 256);
    }
    CMEN = 0x01;
}
```

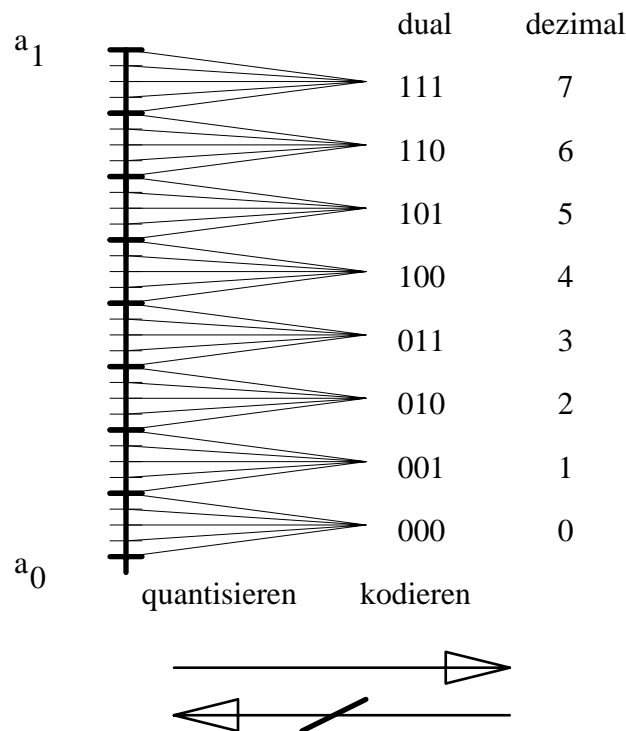
## 13 Analog - Digital - Umsetzer

### 13.1 Allgemeines

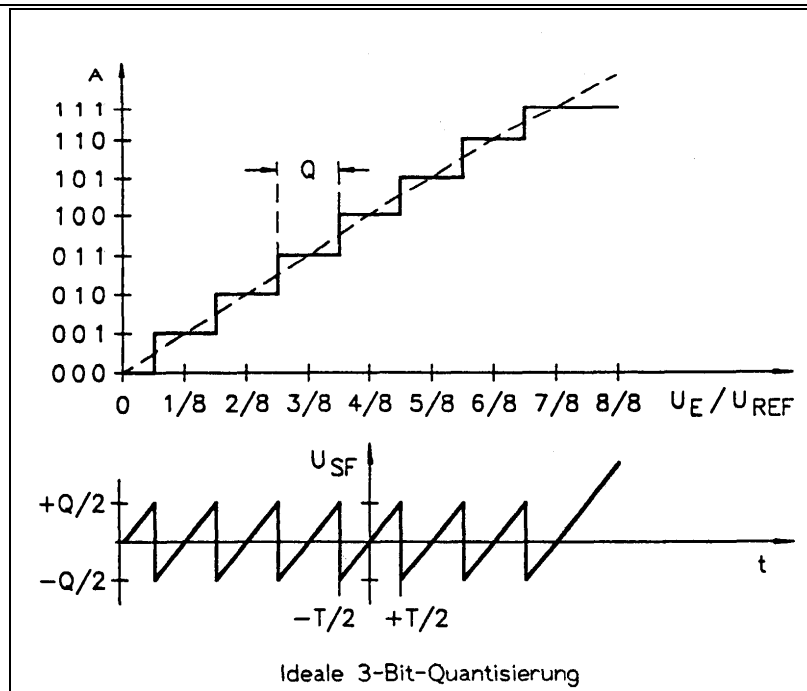
Ein ADU (oder **ADC** Analog Digital Converter) mißt das Verhältnis eines analogen Eingangsgröße  $a$  zu einer Referenzgröße  $a_r$  und gibt dieses in Form eines digitalen Wortes an. Dazu wird der mögliche Bereich der Eingangsgröße in  $n$  gleich große Intervalle zerlegt und festgestellt, welchem Intervall die Eingangsgröße zuzuordnen ist. Aus dem Ordnungsbegriff des Intervalles kann jeder beliebige digitale Kode abgeleitet werden. Die Umsetzung besteht also grundsätzlich aus zwei Schritten:

- ¶ Quantisieren
- Kodieren

Man beachte, daß durch die Quantisierung ein Informationsverlust entsteht, der nicht mehr rückgängig gemacht werden kann. Als Beispiel diene ein ADU mit der Auflösung von 8 Quantisierungsintervallen ( $3 \text{ bit } 2^3=8$ ).



Die Umsetzkennlinie ist eine Treppenfunktion, die angenähert als eine durch den Nullpunkt verlaufende Gerade dargestellt werden kann. Unter Quantisierungsfehler versteht man die Breite einer analogen Treppenstufe.



Man unterscheidet drei grundsätzlich verschiedene Umsetzprinzipien:

- ❶ Parallelverfahren (flash converter)  
 Die Umsetzung wird in einem einzigen Arbeitsschritt durchgeführt; es werden n innere Referenzgrößen benötigt. Das Verfahren ist sehr schnell (Videofrequenzen).
- ❷ Wägetverfahren (successive approximation)  
 Es werden  $N = \lg n$  Arbeitsschritte und N innere Referenzgrößen benötigt. Es ergibt sich ein günstiger Kompromiß zwischen Umsetzzeit und Aufwand.
- ❸ Zählverfahren (incremental converter)  
 n Arbeitsschritte, nur eine innere Referenzgröße. Dieses Verfahren ist langsam, es weist eine sehr gute Linearität auf.

### 13.1.1 Parallelverfahren (flash converter):

Beim Parallel - ADU stellen n Komperatoren fest, welchem Intervall die Eingangsgröße zuzuordnen ist. Die den einzelnen Intervallgrenzen entsprechenden Spannungen werden in einem Spannungsteiler aus n Widerständen erzeugt. n D-FFs synchronisieren den Zustand der Komperatoren damit sichergestellt wird, daß ein bestimmter Zeitpunkt für den Umsetzvorgang repräsentativ ist. Ein Prioritätskodierer reduziert die Zahl der Datenleitungen von n auf N. Derartige Umsetzer erreichen Auflösungen bis zu 8 bit (256 Komperatoren) bei Umsetzzeiten von 10ns. Ein großes Problem stellen die vielen parallel geschalteten Komperatoreingänge dar, die einen hohen dynamischen Eingangsstrom und eine hohe Eingangskapazität verursachen. Die Genauigkeit derartiger Umsetzer hängt vom Spannungsteiler und den Eingangsfehlspannungen der Komperatoren ab.



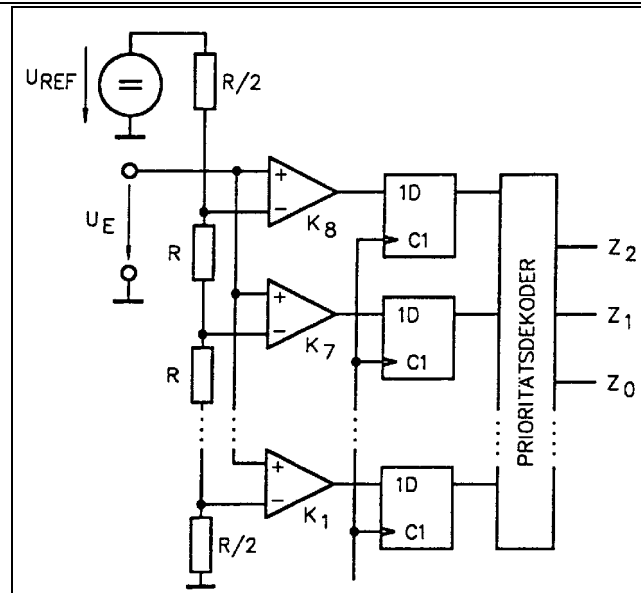


Bild: Prinzipschaltung eines Parallel - ADUs

### 13.1.2 Wägeverfahren (successive approximation)

Der klassische serielle ADU arbeitet nach dem Wägeverfahren mit sukzessiver Approximation. Die Schaltung besteht aus einer Abtast/Halte-Schaltung (Sample & Hold), einem Komperator, einem DAU und einem speziellen Register zur sukzessiven Approximation ("SAR").

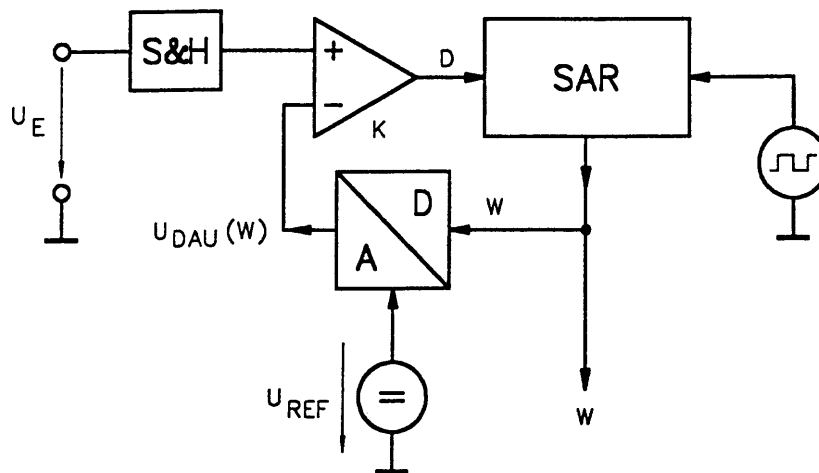


Bild: ADU mit sukzessiver Approximation

Bei diesem ADU wird die Eingangsspannung  $U_E$  mit der Ausgangsspannung des internen DAUs ( $U_{DAU}$ ) über einen Komperator (K) verglichen. Zu Beginn einer Umsetzung wird die Zahl  $W$  auf "0" gesetzt und das höchstwertigste Bit (MSB) auf "1" gesetzt. Ist  $U_{DAU}(W) > U_E$ , so bleibt das MSB auf "1" (Komperatorausgang  $D = "0"$ ), andern-falls ( $D = "1"$ ) wird es gelöscht., d.h. MSB = "0". Dieser Wägevorgang wird anschließend für alle niederwertigeren Bits ausgeführt. Nach Festlegung des LSBs ist der Umsetzungvorgang abgeschlossen. Im Register steht dualkodiert die Zahl  $W$ .

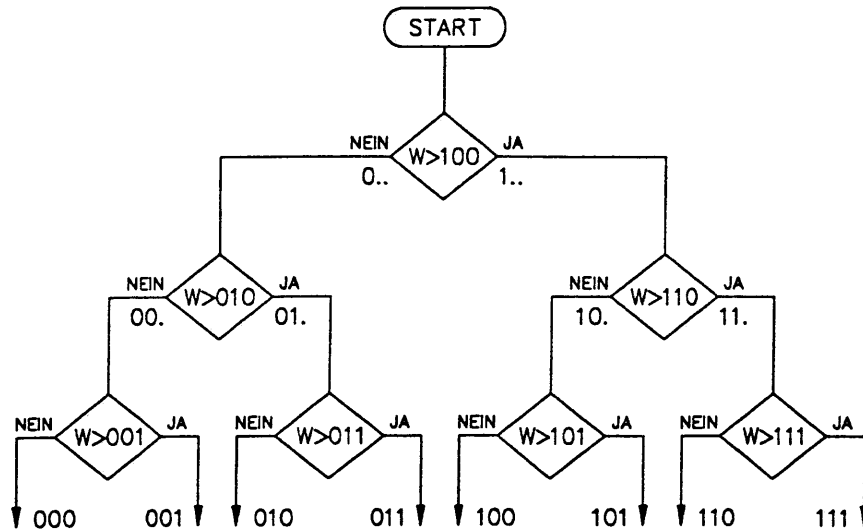


Bild: Flußdiagramm für einen 3 Bit ADU mit sukzessiver Approximation

Die Umsetzzeit liegt bei N Taktperioden. In jeder Taktperiode wird ein bit bestimmt. Umsetzer mit Auflösungen bis 15 Bit sind verfügbar.

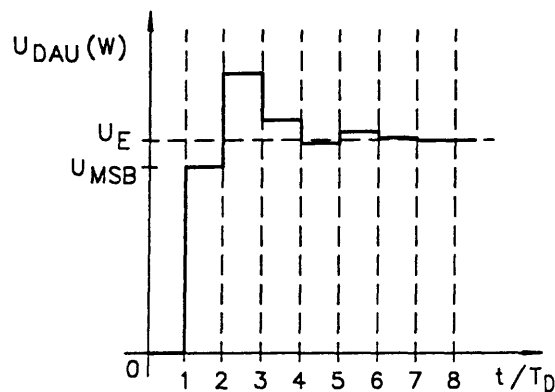
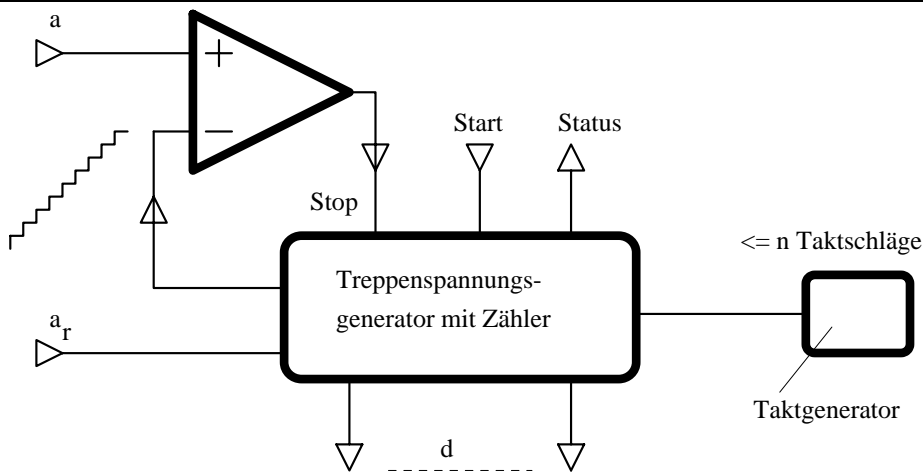


Bild: Zeitlicher Verlauf der Umsetzung bei der sukzessiven Approximation

### 13.1.3 Zählverfahren (incremental converter)

Die Funktion eines ADUs nach dem Zählverfahren kann mit der einer Waage verglichen werden, die anstelle eines Gewichtssatzes nur über viele gleich große Gewichte verfügt. Bei diesem Umsetzer erzeugt ein Treppenspannungsgenerator n Treppenstufen. Ein Komperator vergleicht die Eingangsspannung und die Treppenspannung. Er stellt fest, welche Treppenstufe der Eingangsgröße am nächsten liegt. Die digitale Ausgangsgröße ist identisch mit der Nummer der entsprechenden Treppenstufe.

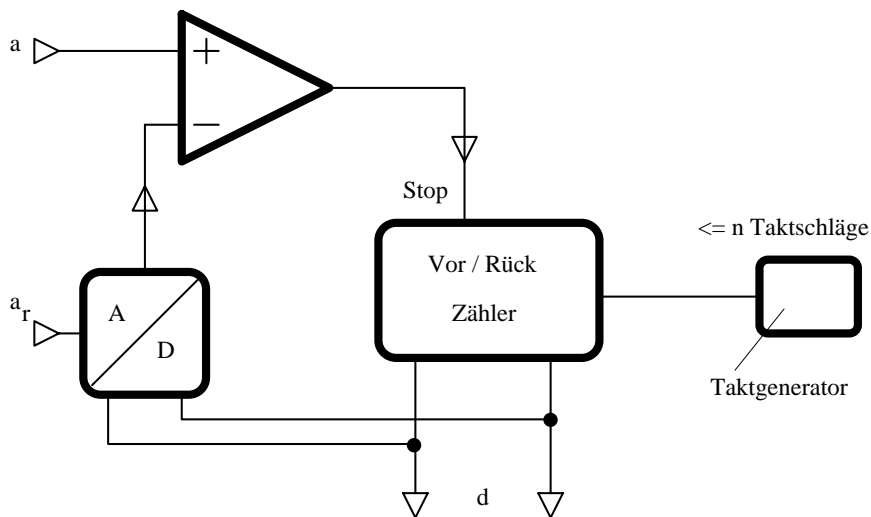
Für die Funktion des Umsetzers ist es gleichgültig wie breit die einzelne Treppenstufe ist. Die Treppe muß also nicht durch eine zeitlineare Rampe angenähert werden. Der Takt-generator hat daher nur die Indexfunktion.



Einfacher ADU nach dem Zählverfahren

### 13.1.4 Tracking ADC (Nachlaufender ADU):

Dieser Umsetzer bildet eine Gegenkopplungsschleife nach. Ein laufend getakteter Zähler, dessen Ausgänge mit den Digitaleingängen eines D/A-Umsetzers verbunden sind, wird in seiner Zählrichtung vom Komperator umgeschaltet. Damit ergibt sich, daß der augenblickliche Zählerstand der Eingangsgröße nachläuft. Dieser Umsetzer kann mit sehr schnellen Zählern (10MHz) betrieben werden. Er eignet sich daher gut für dynamische Anwendungen, wobei allerdings darauf geachtet werden muß, daß er zeitlichen Änderungen der Eingangsspannung nur bis zu einem bestimmten Wert folgen kann. Dieser Umsetzer eignet sich sehr gut für die Darstellung von niederfrequenten Sinussignalen in Echtzeitanwendungen (digitale Leistungsmessung).



Nachlaufender ADU

### + Sägezahnverfahren:

Das älteste elektrisch implementierte Umsetzverfahren ist das klassische Sägezahnverfahren.

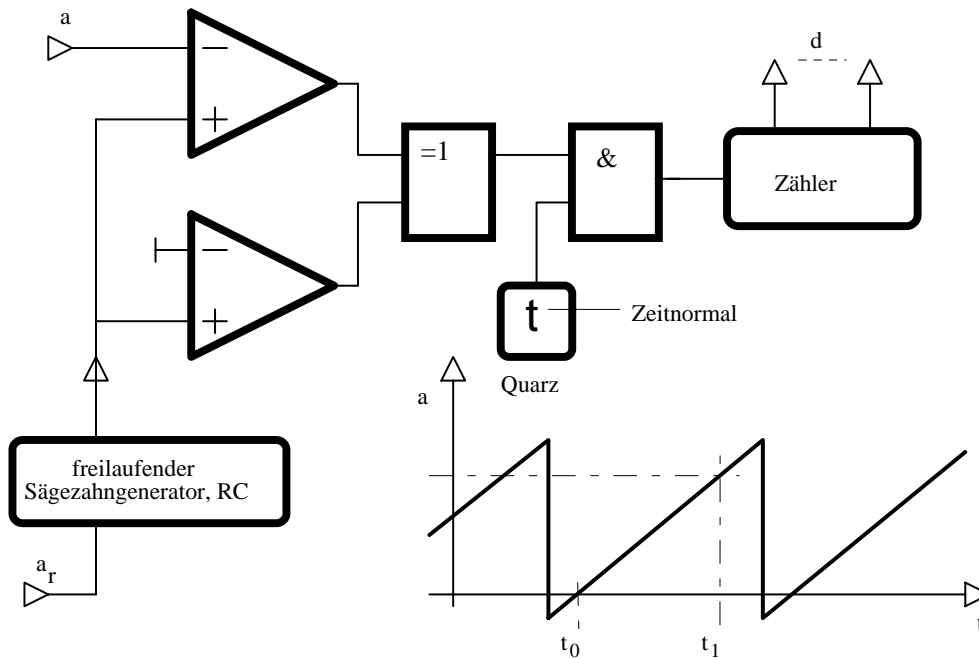


Bild: Klassisches Sägezahnverfahren

Ein freilaufender Sägezahngenerator, dessen Amplitude auf eine Referenzgröße zurückgeführt wird, erzeugt eine zeitlineare Rampenspannung. Diese Spannung wird sowohl mit 0 Volt als auch mit der unbekannt Eingangsspannung verglichen und ein Zählertor geöffnet während  $0 < \text{Rampenspannung} < a$  ist.

Während dieser Zeit zählt ein Zähler eine bestimmte Zahl von Daktschlägen. Der Einfachheit dieses Verfahrens stehen viele **Nachteile** gegenüber, vor allem, daß das RC-Produkt im Sägezahngenerator direkt in das Resultat eingeht. In letzter Zeit wurde dieses Verfahren sehr verbessert, weil es mit Hilfe von Mikroprozessoren möglich ist, Nullpunkts- und Steigungsfehler automatisch zu korrigieren. Eine inhärent kleine differentielle Nichtlinearität ist aus dem Verfahren sofort ersichtlich. Man beachte, daß bei diesem Verfahren der Quantisierung und Kodierung die Zeit als Informationsparameter in Erscheinung tritt.

### 13.1.5 Zwei-Rampenverfahren (dual slop, dual ramp):

Das Zwei-Rampenverfahren ist das älteste integrierende ADU-Verfahren. Es ist im Meßgerätebau weit verbreitet.

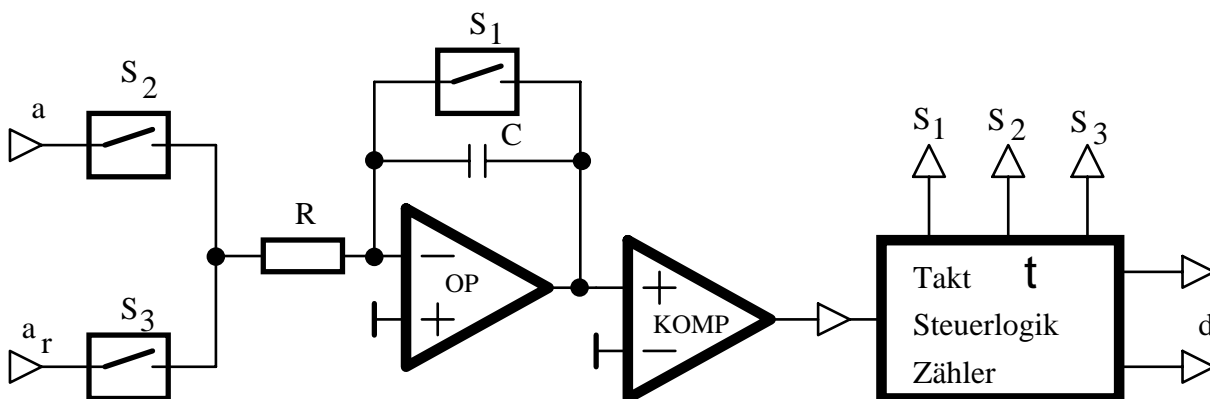
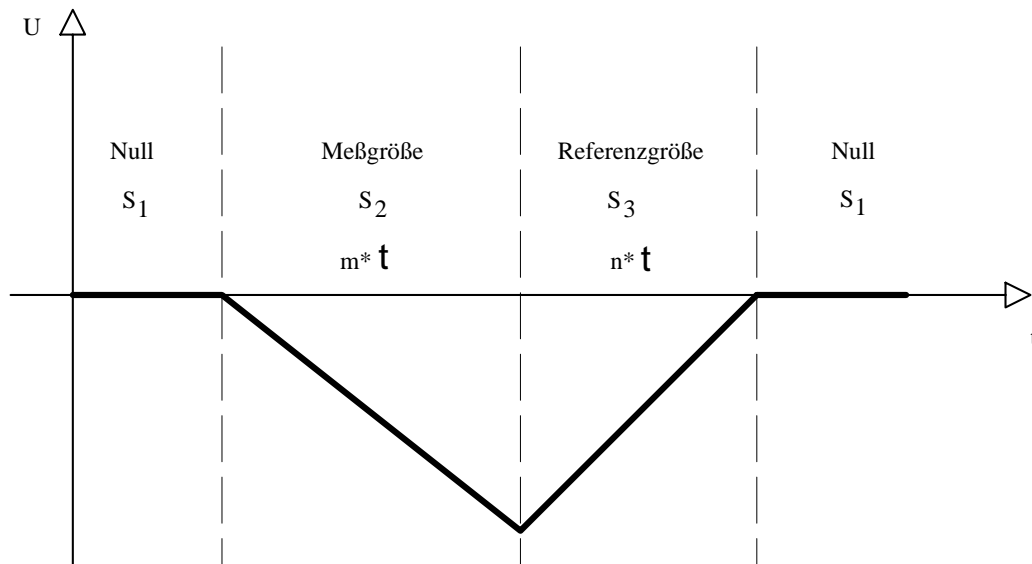


Bild: ADU nach dem Zwei-Rampenverfahren



Das Verfahren läuft in drei Abschnitten. Während der Phase Null ( $S_1$  geschlossen) wird der Integrationskondensator entladen. In der Phase "Meßgröße" ( $S_2$  geschlossen) wird das Zeitintegral der Meßgröße  $a$  während einer festen Zahl  $m$  von Taktschlägen der Dauer  $\tau$  gebildet. Danach wird in der Phase "Referenzgröße" ( $S_3$  geschlossen) zu dem gebildeten Integral das Zeitintegral der Referenzgröße hinzugezählt. Da die Referenzgröße umgekehrtes Vorzeichen hat, wird zu einem bestimmten Zeitpunkt der Betrag des Integrals Null sein.

Zu diesem Zeitpunkt unterbricht der Komparator die Referenzintegration. Der Zähler in der Steuerlogik zeigt die Dauer der Referenzintegration als Ergebnis an. Wie man sieht kürzt sich aus dem Resultat sowohl die Periode des Taktgenerators  $\tau$  als auch das RC-Produkt des Integrators heraus. Diese beiden Tatsachen sind der Grund für die große Beliebtheit des Verfahrens.

Nachteile des Verfahrens sind:

- ➔ die Tatsache, daß die "Beobachtung" des Eingangssignales durch die Phasen Null und Ref. unterbrochen wird,
- ➔ daß beim zeitlichen Übergang von "Meßgröße" auf "Referenzgröße" die gesamte Information im Integrationskondensator steckt und dessen dielektrische Absorption den Genauigkeitsengpaß darstellt und
- ➔ daß im Eingang  $a$  ein elektronischer Schalter  $S_2$  notwendig ist, der es verbietet, höhere Spannung direkt zu verarbeiten.

Da die Schalter  $S_1$  und  $S_3$  nur einmal pro Konversion betätigt werden, sind ihre Zeitfehler wenig kritisch.

Wird die Referenzspannung während der Referenzintegration von der Stellung des Zählers abhängig gemacht, kann der Umsetzer Sensorfunktionen implizit lösen. Die Abhängigkeit der Referenzspannung wird erzeugt, indem diese aus einem DAU entnommen wird, dessen Eingangssignale aus einem ROM stammen, welches vom Zähler adressiert wird, wenn während der Referenzphase der Widerstand R z.B. mit Hilfe eines parallelgeschalteten Leiternetzes verändert wird.

Die eben gezeigte Schaltung hat nur prinzipielle Bedeutung, weil einerseits der Eingangswiderstand klein ist und andererseits die Eingangsfehlspannung des Operationsverstärkers stört.

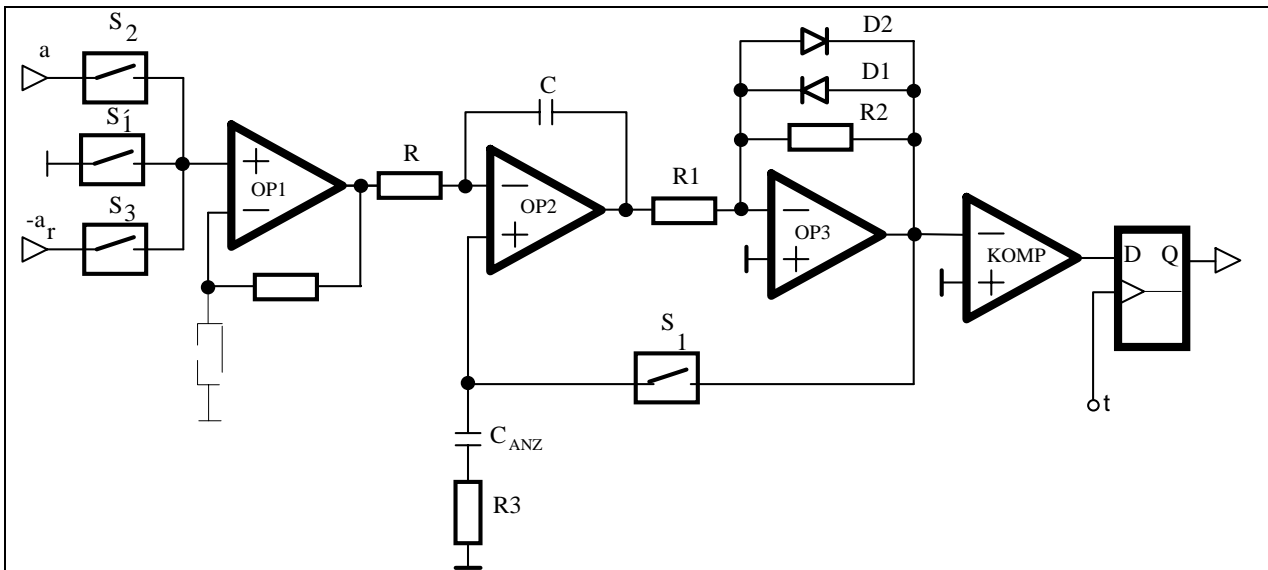


Bild: Praktische Ausführung des Zwei-Rampen ADUs

Tatsächlich eingesetzte Wandler verwenden einen Folger im Eingang, einen Nachverstärker zur schnellen Ansteuerung des Komparators und einem automatischen Nullabgleich (Autozero). In der Phase Null sind jetzt die Schalter S1 und S'1 geschlossen. Am nichtinvertierenden Eingang von OP2 wird sich jene Spannung einstellen, die notwendig ist, daß in den Integrationskondensator kein Strom fließt. Da gleichzeitig der Eingang von OP1 auf Masse liegt ist somit jene Spannung in  $C_{AZ}$  gespeichert, die die Fehlspannungen der ersten drei Operationsverstärker ausgleicht. Wenn S1 danach geöffnet wird, hält der Kondensator  $C_{AZ}$  diese Spannung während der beiden anderen Phasen konstant. Durch diese Maßnahme wird sowohl die Offsetspannung der Operationsverstärker als auch deren zeitliche Veränderung (Alterung) vollkommen ausgeglichen. Der Widerstand R3 verhindert die kapazitive Belastung von OP3 durch den großen (ca. 1mF) Kondensator  $C_{AZ}$ . Durch den begrenzenden Nachverstärker wird die Steigung der Rampe in der Nähe des Nulldurchganges vergrößert und somit die zeitliche Unsicherheit des Komparators reduziert. Wenn dieses analoge System mit kritischen Spannungen  $<1\text{mV}$  in unmittelbarer Nähe des digitalen Steuerwerkes arbeiten soll (auch auf dem gleichen Halbleiter-Chip), dann ist es vorteilhaft, mit Hilfe eines Synchronisations-FFs die Stellung des Komparatorausganges zu einem Zeitpunkt abzufragen, zu dem das Digitalsystem in Ruhe ist.

Mit derartigen Maßnahmen ist es möglich, Umsetzer mit Auflösungen bis zu 20 bit und Empfindlichkeiten von 1mV zu konstruieren. Bei den hohen Auflösungen stört oft die dabei entstehende lange Meßzeit. Sie kann dadurch verkürzt werden, daß während der Meßphase in Anlehnung an das charge balance-Verfahren der Integrationskondensator immer wieder entladen

wird und auf diese Weise die höherwertigen Stellen des Meßresultates ermittelt werden. Nach Ende der Meßphase wird dann entsprechend kürzer zur Ermittlung der fehlenden Stellen referenzintegriert.

Auch in der Referenzintegration kann zur Verkürzung der Zeit mit dezimalgewichteten Strömen gearbeitet werden (multiple slope). Alle diese verkürzenden Maßnahmen machen den Vorteil, daß der Betrag des Integratorwiderstandes nicht eingeht, zunichte.

### 13.2 Der ADU des 80C517

**Der interne ADU des 80C517 (Siemens) besitzt folgende Merkmale:**

- 8 Bit Sampling ADU nach dem Prinzip der Successiven Approximation
- 12 gemultiplexte Analogeingänge die auch als digitale Eingänge benutzt werden können.
- programmierbare Referenzspannungsquellen ( in 16 Stufen )
- 13µs Umsetzzeit bei 12Mhz Taktfrequenz
- der Start der Umsetzung kann intern oder extern ausgelöst werden
- Generierung eines Interrupt nach jeder beendeten Umsetzung

### 13.3 Erklärungen

- *Sampling* das Eingangssignal wird abgetastet, zwischengespeichert und dem ADU zugeführt.
- *gemultiplext* Es steht für die 12 Eingänge nur ein Umsetzer zur Verfügung, der Multiplexer übernimmt die Zuschaltung des jeweils angewählten Eingangspins (Kanals) auf den ADU.
- *Successiven Approximation*  
auch *Wägeverfahren* genannt, ist eines der 3 Verfahren der AD Umsetzung. ( Paralellverfahren, Wägeverfahren, Zählverfahren)  
Die (gesamplete) abgetastete Eingangsspannung wird mittels DA-Umsetzer und Komparator mit dem vorigen Wandlungsergebnis verglichen, und entsprechend dem Ergebnis, (größer oder kleiner als der vorige Wert) wird ein Bit im *Successiven Approximation Register* gesetzt bzw. rückgesetzt. Das wird vom höchstwertigen Bit (MSB) bis zum niederwertigsten Bit in gleicher Weise durchgeführt. Am Ende dieser Vergleiche steht im Register eine Zahl, die DA-Umgesetzt (innerhalb der Auflösungsgrenzen) der Eingangsspannung entspricht.

### 13.4 Blockschaltbild ADU 80C517

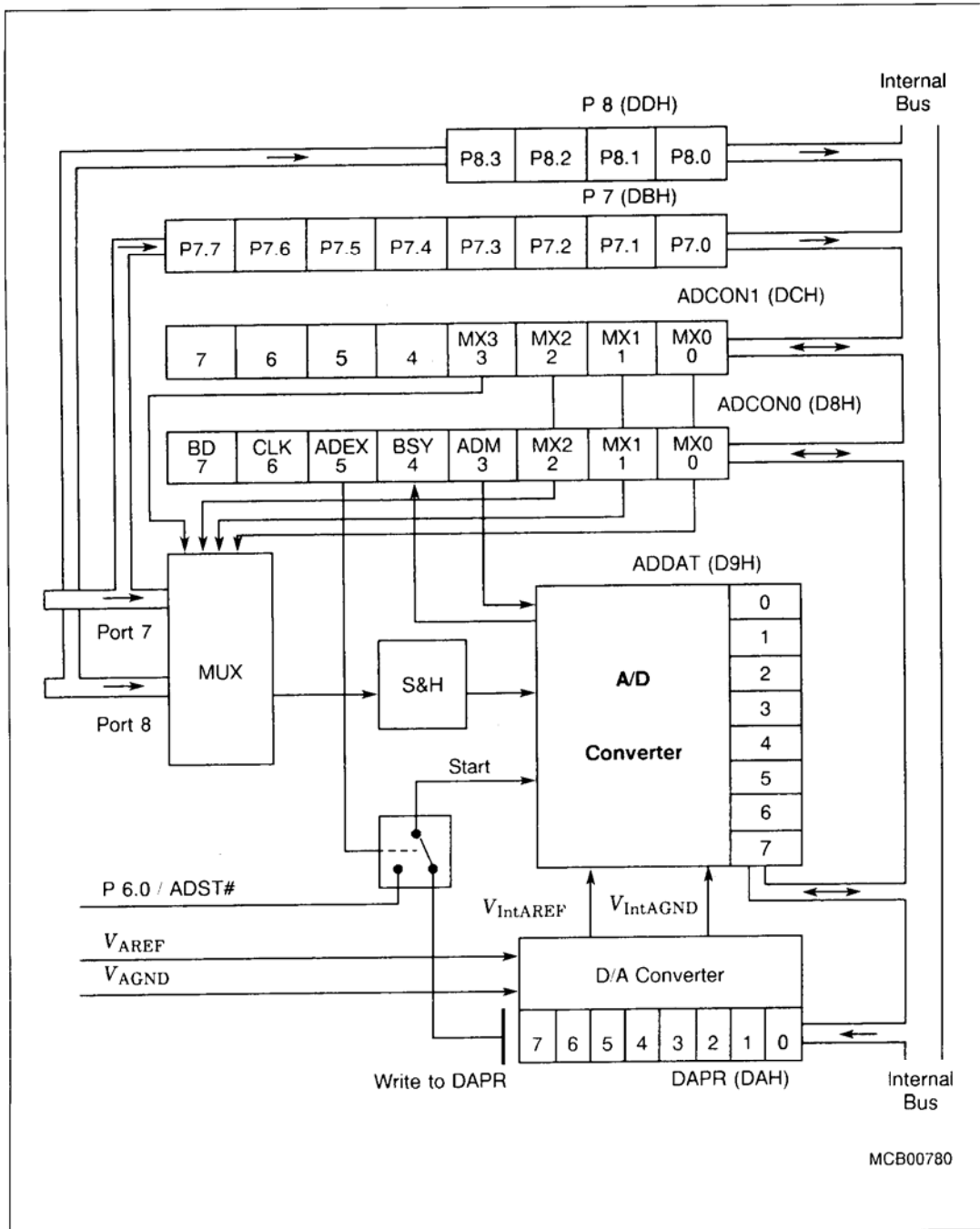


Bild 10-1: Funktionsschaltbild des A/D-Wandlers im 80C517



### 13.5 Initialisierung und Wahl des Eingangskanals

Es existieren beim 80C517 zwei *Special Function Register (SFR)* **ADCON0** und **ADCON1**, durch deren Beschreiben es möglich ist, zwei Arbeitsmodi auszuwählen, den Status zu lesen und den gewünschten Eingangspin (Portpin) auf den ADU zu schalten.

#### ADCON0:



- mit den 3 BIT *MX0 MX1 MX2* kann nun einer der 8 Eingangskanäle (Kanal 0 - 7) ausgewählt werden.
- **ADM** (ADU Mode) ist dieses Bit gesetzt (= 1) so befindet sich der ADU in einem Modus in dem er kontinuierlich umsetzt.  
Ist dieses Bit 0, so stoppt der ADU nach jeder Umsetzung.
- **BSY** (Busy Flag) dieses Bit zeigt an ob sich der ADU in Ruhe befindet oder gerade eine Konversion durchführt.  
BSY = 1 Umsetzung wird gerade durchgeführt  
Ist die Umsetzung beendet und liegt das Ergebnis vor, so wird das Bit von der Hardware zurückgesetzt.
- **ADEX** (*ADU extern*) Interner bzw. externer Start einer AD Umsetzung  
Wenn das ADEX Bit gesetzt ist, kann die Konversion am P6.0 /ADST# gestartet werden.

#### ADCON1 :



Die verbleibenden 4 Kanäle beziehungsweise alle 12 Kanäle können durch die 4 Bit *MX0 - MX3* als aktueller Eingang im *SFR ADCON1* gewählt werden.

Es kann also entweder jeder der 12 Kanäle in ADCON1 gewählt werden, oder die unteren 8 Kanäle in ADCON 0.

Liegt nun ein Ergebnis einer Umsetzung vor, so wird dieses Ergebnis im Register **ADDAT** gespeichert.

Dieses Ergebnis ist nun eine Umsetzung des 0 - 5 Volt Eingangssignals mit einer Auflösung von 8 Bit (Spannungsauflösung 5 mV).

Durch einen Trick kann beim 80C517 die Auflösung von 8 Bit auf maximal 10 Bit erhöht werden. Der Mikrokontroller 80C517A hat bereits einen 10 Bit ADU mit 12 gemultiplexten Eingangskanälen integriert.

Ein weiterer  $\mu$ C mit 10 Bit ADU ist der 80C552 der neben einem 8 Kanal ADU auch ein I<sup>2</sup>C Bus Interface besitzt.

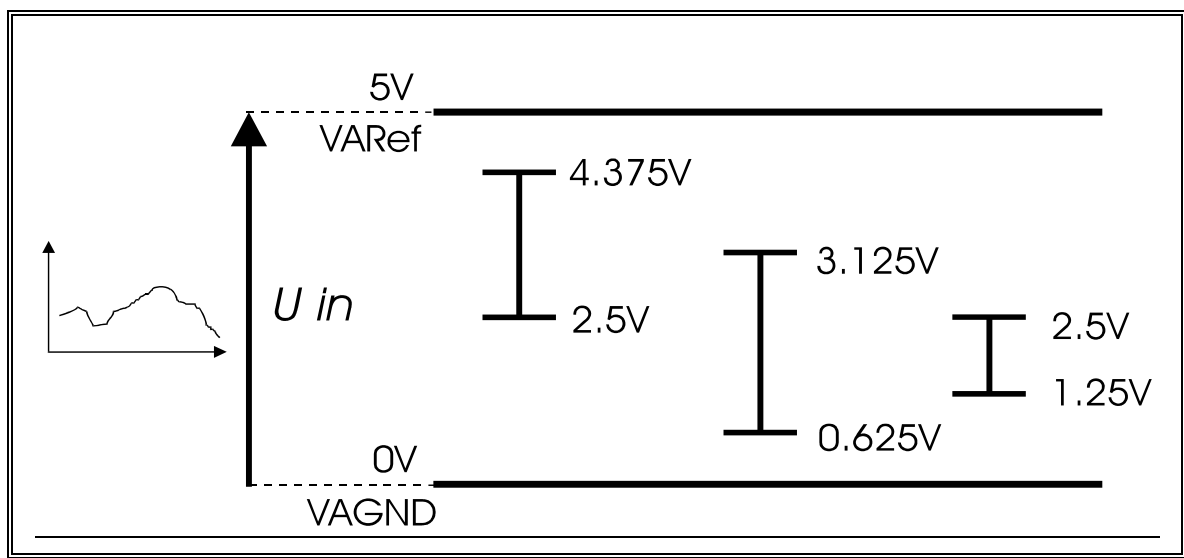
### 13.6 Erhöhung der Umsetzerauflösung von 8 Bit auf 10 Bit:

1. man bestimmt zuerst den Wert der Eingangsspannung mit Referenzspannungen von 0 und 5V.
2. man verschiebt je nach Größe der Eingangsspannung die Referenzspannungen und reduziert somit den Umsetzungsbereich.
3. man führt eine zweite Umsetzung in diesem eingegrenzten Bereich durch und erhält in diesem eingegrenzten Bereich wieder 8 Bit Auflösung.

Die kleinste Intervallgröße ist 1/4 des Referenzspannungsbereichs also 1,25 V.

- Bei 8 Bit und 5V Referenzspannung beträgt die Auflösung (engl. resolution) demnach  $5 / 2^8 = 19,5 \text{ mV}$ .
- Bei 8 Bit und 1.25V Referenzspannung beträgt die Auflösung  $1.25\text{V} / 2^8 = 4,88 \text{ mV}$ . Das entspricht 10 Bit Auflösung bei 5V Referenzspannung, denn  $5\text{V} / 2^{10} = 4,88 \text{ mV}$ .

### 13.7 Anpassen der Referenzspannungen an die Eingangsspannung:



#### 13.7.1 Interne einstellbare Referenzspannungen:

Die internen Referenzspannungen **VintAGND** und **VintAREF** können in 1/16 Schritten im Bereich der äußeren Referenzspannungen VAREF - VAGND programmiert werden. In unserem Fall wäre dies 5V - 0V. 1/16 Schritt entspricht also wie aus der nachfolgenden Tabelle zu entnehmen 0,3125V.

Der minimale Abstand der beiden Referenzspannungen muß allerdings mindestens 1V betragen um eine einwandfreie Funktion des AD Umsetzers zu gewährleisten.

Das entspricht also bei 5V mindestens 4 (1/16) Schritte Unterschied.  
 Durch Schreiben ins Register *DAPR* werden diese Referenzspannungen programmiert.

	Bits 4 - 7	Bits 0 -3	
	<b>VintAREF</b>	<b>VintAVGND</b>	
<b>Schritt</b>	<b>DAPR (.3 - .0) DAPR (.7 - .4)</b>	<b>Vint AGND</b>	<b>Vint AREF</b>
<i>0</i>	<i>0000</i>	<i>0.0</i>	<i>5.0</i>
<i>1</i>	<i>0001</i>	<i>0.3125</i>	-
<i>2</i>	<i>0010</i>	<i>0.625</i>	-
<i>3</i>	<i>0011</i>	<i>0.9375</i>	-
<i>4</i>	<i>0100</i>	<i>1.25</i>	<i>1.25</i>
<i>5</i>	<i>0101</i>	<i>1.5625</i>	<i>1.5625</i>
<i>6</i>	<i>0110</i>	<i>1.875</i>	<i>1.875</i>
<i>7</i>	<i>0111</i>	<i>2.1875</i>	<i>2.1875</i>
<i>8</i>	<i>1000</i>	<i>2.5</i>	<i>2.5</i>
<i>9</i>	<i>1001</i>	<i>2.8125</i>	<i>2.8125</i>
<i>10</i>	<i>1010</i>	<i>3.125</i>	<i>3.125</i>
<i>11</i>	<i>1011</i>	<i>3.4375</i>	<i>3.4375</i>
<i>12</i>	<i>1100</i>	<i>3.75</i>	<i>3.75</i>
<i>13</i>	<i>1101</i>	-	<i>4.0625</i>
<i>14</i>	<i>1110</i>	-	<i>4.375</i>
<i>15</i>	<i>1111</i>	-	<i>4.68754</i>

## 14 Sicherheitsmechanismen

Der 80C517/80C5I7A bietet zwei integrierte Überwachungseinheiten, die bei Fehlfunktionen der Hard- oder Software einen definierten Zustand des Bausteins herstellen und ein "Versagen nach der sicheren Seite" gewährleisten sollen:

- ➔ Einen programmierbaren Watchdog-Timer (WDT) mit variabler Reaktionszeit (von 341ms bis ca. 730 ms bei 18 MHz Taktfrequenz). Der Watchdog-Timer des 80C517/80C5I7A ist eine Übermenge desjenigen im 80(C)515. Nach dem Reset ist der WDT so konfiguriert, daß er sich wie der des 80(C)515 verhält.
- ➔ Einen Oszillator-Watchdog (OWD), der die Taktversorgung überwacht und bei Versagen einen Reset auslöst.

### 14.1 Watchdog-Timer (WDT)

Die Aufgabe des Watchdog-Timers ist, prinzipiell sicherzustellen, daß das ablaufende Programm korrekt arbeitet. Insbesondere soll er erkennen, wenn es aufgrund irgendeiner Störung aus seiner Bahn geworfen wird. Beispielsweise könnte durch elektromagnetische Einstrahlung auf dem externen Bus ein falscher Programmcode eingelesen werden, der dann einen Sprung in einen nicht definierten Programmbereich bewirkt. Nun ist es aber dem Watchdog-Timer nicht möglich, zwischen dem, was der Softwaredesigner beabsichtigt (sinnvoll), und einem Fehlverhalten (unsinnig) zu unterscheiden, da in beiden Fällen die CPU irgendeinen Programmcode abarbeitet. Daher muß man sich die Arbeitsweise des Watchdog-Timers wie folgt vergegenwärtigen:

Wenn der WDT einmal gestartet worden ist, kann man ihn nicht mehr anhalten (außer durch einen externen Reset). Sobald er überläuft, löst er einen Hardware-Reset aus, der den Controller wieder definiert beginnen läßt. Dieser WDT-Reset unterscheidet sich vom externen Reset nur dadurch, daß der WDT weiterläuft und sein Statusflag setzt. Um den WDT vom Überlaufen abzuhalten, muß ihn die CPU immer wieder zurücksetzen und von vorne zählen lassen. Der Software-Entwickler muß also sicherstellen, daß im normalen Programmlauf der WDT immer wieder zurückgesetzt wird. Falls dann die CPU außer Kontrolle geraten sollte, wird der WDT nicht mehr zurückgesetzt und löst nach der voreingestellten Überlaufzeit einen Reset aus. Der Mikrocontroller beginnt danach wieder von einem definierten Zustand an zu arbeiten. Dabei ist zu beachten, daß das Rücksetzen des WDT nicht in einer immer wiederkehrenden Interrupt-Routine erfolgt, da diese Routine ja stets ausgeführt wird, unabhängig davon, wo sich die CPU gerade befindet. Somit würde im Falle des Programmabsturzes zwar der WDT immer wieder zurückgesetzt, das Programm würde anschließend aber weiterhin Unsinn machen. Vielmehr ist zu empfehlen, den WDT im Hauptprogramm an Stellen zurückzusetzen, die im normalen Betrieb immer wieder durchlaufen werden.

Außer zu Sicherheitsaufgaben kann der Watchdog-Timer aber auch während der Software-Entwicklung als Testmittel eingesetzt werden. Springt die CPU in nicht beabsichtigte Bereiche oder durchläuft sie nicht vorgesehene Pfade, wird der WDT nicht zurückgesetzt, und der Entwickler bekommt das durch das Auftreten eines Reset mitgeteilt (RO#-Pin).

Der WDT im 80C517/80C5I7A ist ein 15 bit breiter Timer, der je nach gewähltem Vorteiler alle zwei bzw. 32 Maschinenzyklen inkrementiert wird (Zähltakt  $f_{OSZ}/24$  oder  $f_{OSZ}/384$ ). Der Vorteiler/2 ist fest implementiert, während der nachgeschaltete zweite Vorteiler/16 optional

durch Setzen von Bit 7 im SFR WDTREL (86H) aktiviert werden kann. Zusammen mit dem Nachladewert, der in den unteren sieben Bits von WDTREL eingestellt wird, ist die Überlaufzeit variabel programmierbar.

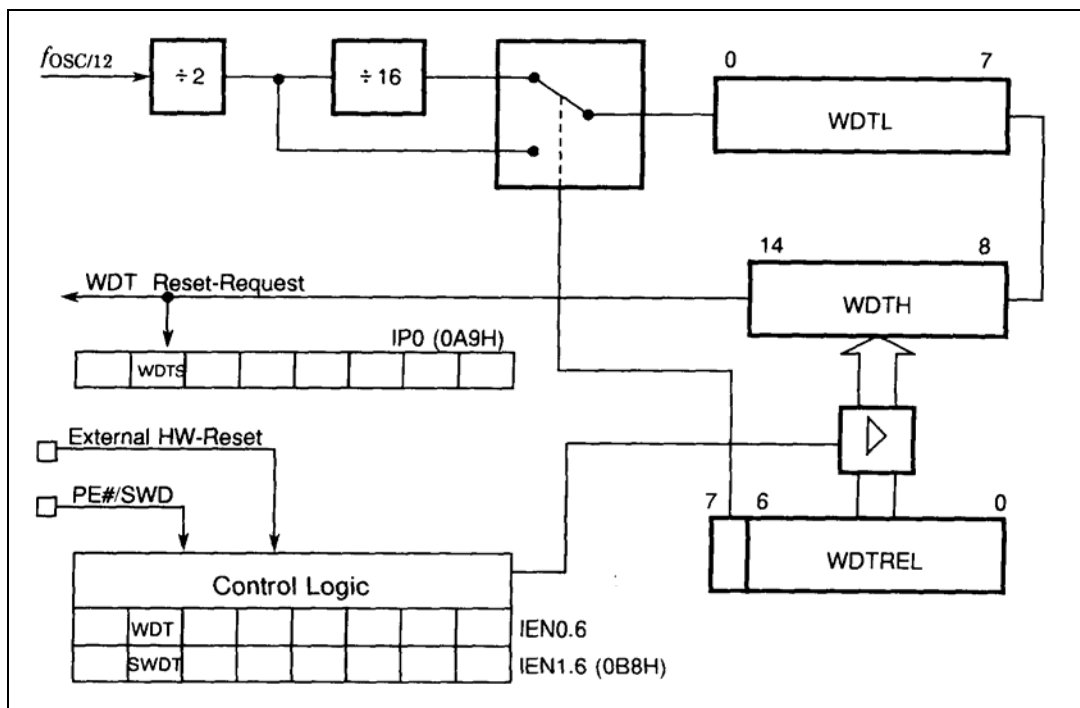


Abb. Funktionsschaltbild des Watchdog-Timers

Sofort nach dem Start wird der WDT mit dem Nachladewert in WDTREL.0-WDTREL.7 initialisiert. Bei einem externen Reset wird WDTREL wieder mit seinem Reset-Wert 00H belegt. Es kann jederzeit durch Software beschrieben werden.

MSB							LSB
WDTREL.7	WDTREL.6	WDTREL.5	WDTREL.4	WDTREL.3	WDTREL.2	WDTREL.1	WDTREL.0
Watchdog-Timer-Nachladeregister (Watchdog Timer Reload Register). Es enthält den Nachladewert für die höheren sieben Bits des Watchdog-Timers und ein Vorteilerbit. Reset-Wert: 00H							
Bitsymbol	Funktion						
WDTREL.7	Bit zur Einstellung des Eingangstaktvorteilers für den Watchdog-Timer. WDTREL.7 = 1 schaltet einen zusätzlichen Vorteiler/16 ein (siehe Bild 14-1).						
WDTREL.6	7 bit breiter Reload-Wert für den höherwertigen Teil des Watchdog-Timers.						
WDTREL.5	Dieser Wert wird in den Watchdog-Timer geladen, wenn dieser gestartet bzw. durch aufeinanderfolgendes Setzen der Bits WDT und SWDT zurückgesetzt wird.						
WDTREL.4							
WDTREL.3							
WDTREL.2							
WDTREL.1							
WDTREL.0							

Abb. Special-Function-Register WDTREL (86H)

WDTREL	Überlaufzeit	Bemerkungen
00H	65,5 ms	Dies ist der Default-Wert nach Reset; er stimmt mit der Überlaufzeit des 80(C)515 überein.
80H	1,1 s	Maximale Überlaufzeit
7FH	512 $\mu$ s	Minimale Überlaufzeit

Abb. Beispiele für Überlaufzeiten des Watchdog-Timers

## 14.2 Oszillator-Watchdog (OWD)

Ein kritischer Fall tritt dann ein, wenn die Taktversorgung des Mikrocontrollers ausbleibt. Was passiert, wenn der interne Oszillator aufhört zu schwingen, z.B. weil der Quarzkristall beschädigt ist oder eine Verbindung zum Kristall unterbrochen wird? Selbst der Watchdog-Timer kann hier nicht seine Funktion erfüllen, da auch er nicht mehr getaktet wird. Abhängig vom Chipdesign werden manche Mikrocontroller einfach stehenbleiben und ihren Zustand exakt beibehalten (statisches Design), andere werden in einen undefinierten Zustand übergehen. Beides ist in sicherheitskritischen Anwendungen untragbar, man denke nur an ein Anti-Blockiersystem im KFZ.

Beim 80C517/80C517A hingegen sorgt der integrierte Oszillator-Watchdog für einen internen Reset, sobald er einen Abfall der Taktfrequenz unterhalb eines bestimmten Werts entdeckt. Er hält den Mikrocontroller in diesem Zustand, bis die Taktversorgung wiederhergestellt ist. Damit ist das System auch in erschütterungsreichen Anwendungen (z.B. Automobil, Zentrifugen etc.) oder in Applikationen mit extremen Temperaturschwankungen gesichert, in denen die Anschlüsse des Quarzes sich lösen können oder der Quarz selbst beschädigt werden kann. Der OWD kann daher als Hardwareüberwachung verstanden werden, im Gegensatz zum Watchdog-Timer, der eher den Programmablauf kontrolliert. Im Reset-Zustand geben alle Portpins des 80C517/80C517A einen High-Pegel aus. Beim Entwurf der Anwenderschaltung sollte deshalb darauf geachtet werden, daß damit ein sicherer Zustand gewährleistet ist.

Der Oszillator-Watchdog besteht aus einem einfachen internen RC-Oszillator, der bei einem Ausfall des Haupttakts die Taktversorgung übernimmt, und einem Frequenzkomparator, der diesen Hilfstakt mit dem normalen Oszillatortakt vergleicht. Der OWD muß per Hardware am Pin OWE (Oscillator Watchdog Enable) freigegeben werden (OWE auf High-Pegel). Wird der Pin offengelassen, zieht ihn ein interner Pull-Up auf High-Pegel, so daß auch im Fall einer Leitungsunterbrechung der OWD funktioniert. Bei Anlegen eines Low-Pegels an Pin OWE ist der OWD ausgeschaltet.

Es stellt sich jetzt die Frage, wozu der interne RC-Takt überhaupt benötigt wird und warum er nicht dazu verwendet wird, den Baustein weiterlaufen zu lassen. Dazu muß man wissen, daß die Durchführung eines Reset nur bei Vorhandensein eines Takts möglich ist, weil die Reset-Sequenz im Grunde nichts anderes als ein Beschreiben von Special-Function-Registern mit definierten Werten ist. Andererseits kann der Hilfstakt nicht als normaler Betriebstakt benutzt werden, da dann wegen der niedrigen Frequenz das Echtzeitverhalten des 80C517/80C517A nicht mehr gewährleistet wäre (sämtliche Signale würden ja viel langsamer sein). Zudem ist die Erzeugung eines schnellen und genauen Takts mit einem integrierten RC-Glied technologisch nicht möglich.

Wie der Watchdog-Timer, hat auch der OWD ein Statusbit, das der Software signalisiert, daß der Reset durch den OWD ausgelöst wurde. Dieses Flag OWDS (Oscillator Watchdog Status) ist das höchstwertige Bit im Register IPO (A9H) und wird nach einem OWD-Reset von der Hardware gesetzt. Es kann durch die Software gelesen und auch geschrieben werden. Bei einem externen Reset wird es gelöscht. Wird der Reset durch den Oszillator-Watchdog ausgelöst, bleibt das Statusbit des Watchdog-Timers WDTS unbeeinflußt, d.h. es wird nicht gelöscht, falls es bereits gesetzt war. Der Watchdog-Timer selbst wird aber angehalten.

Im Detail bestehen Unterschiede in Aufbau und Funktion des OWD im 80C517 und 80C517A, weswegen sie im weiteren getrennt betrachtet werden.

### 14.3 Oszillator-Watchdog im 80C517

Typischerweise läuft der OWD im 80C517 mit ungefähr 300 kHz. Er ist unabhängig von der externen Beschaltung (die Versorgungsspannung muß selbstverständlich vorhanden sein). Die Grenze für die normale Taktpeisung, unterhalb derer er anspricht, wurde auf 1 MHz spezifiziert. Da der Frequenzkomparator den internen RC-Takt direkt mit dem Haupttakt vergleicht, kann selbst bei 1 MHz Taktfrequenz der Slow-Down-Modus verwendet werden (siehe Kapitel 13). Sobald der Haupttakt niedrigere Frequenz als der RC-Oszillator hat, schaltet der Komparator die Taktversorgung auf den internen RC-Oszillator um und führt einen Reset durch. Wenn sich die Taktversorgung nach einem Ansprechen des Oszillator-Watchdogs wieder erholt hat, hält der OWD den 80C517 noch für mindestens drei Maschinenzyklen im Reset, um sicherzustellen, daß der Hauptoszillator sich stabilisiert hat. Bild 14-8 zeigt das Prinzipschaltbild des Oszillator-Watchdog im 80C517.

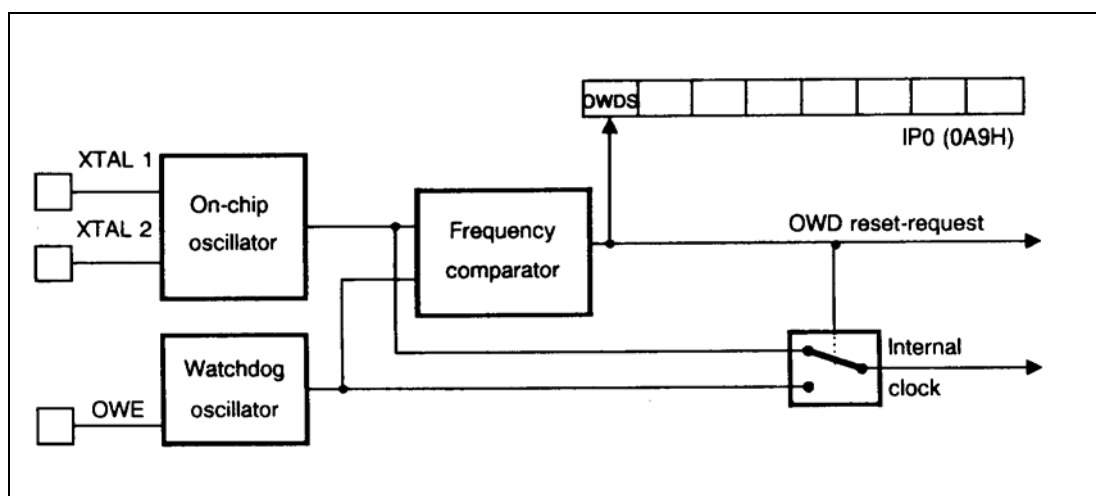


Abb. Funktionsschaltbild des Oszillator-Watchdogs im 80C517

### 14.4 Oszillator-Watchdog im 80C517A

Im 80C517A hat der Oszillator-Watchdog insgesamt drei Aufgaben.

- ➔ Die Frequenzüberwachung: Dabei funktioniert der OWD prinzipiell wie im 80C517.
- ➔ Neustart nach dem Hardware-Power-Down-Modus: Sobald der HWPDModus verlassen wird, sorgt der Oszillator-Watchdog für einen schnellen, korrekten Reset, bis der Hauptoszillator stabil arbeitet. Im Grunde ist dies nur eine Abwandlung der Frequenzüberwachung.

➔ Schneller Reset nach Power-On:

Der 80C517A sorgt unmittelbar nach Einschalten der Versorgungsspannung für einen internen Reset, noch bevor sich der Hauptoszillator stabilisiert hat und eine normale Resetsequenz durchführen kann. Auch hier arbeitet der OWD eigentlich als Frequenzüberwacher. Details zum schnellen Reset nach Power-On sind in Kapitel 6 enthalten.

Um diese Funktionen erfüllen zu können, muß der OWD durch High-Pegel am Pin OWE freigegeben sein.

Die Frequenz des internen RC-Oszillators (ca. 1.5 MHz) wird auf ca. 500 kHz heruntergeteilt und

vom Frequenzkomparator mit dem Haupttakt verglichen. Sobald der Haupttakt niedrigere Frequenz als der RC-Oszillator hat, schaltet der Komparator die Taktversorgung auf den internen RCOszillator um und führt einen Reset durch. Wie beim 80C517 kann auch beim 80C517A der SlowDown-Modus mit einem externen Takt von 1 MHz noch verwendet werden, da der Frequenzkomparator den Vergleichstakt direkt am Oszillator abgreift. Entdeckt der OWD, daß der Haupttakt wiederhergestellt worden ist, hält er den 80C517A noch für typischerweise 1 ms im Reset, bevor er auf die normale Taktversorgung zurückschaltet. Durch diese Wartezeit stellt er sicher, daß der Hauptoszillator sich stabilisiert hat. Anschließend entläßt er den Mikrocontroller aus dem Reset. Falls keine externe Reset-Anforderung anliegt, beginnt der Baustein sofort mit der Programmabarbeitung, andernfalls bleibt er im Reset, solange der Resetpin auf Low-Pegel gehalten wird. Nachfolgende Abbildung zeigt das Blockschaltbild des Oszillator-Watchdog im 80C517A.

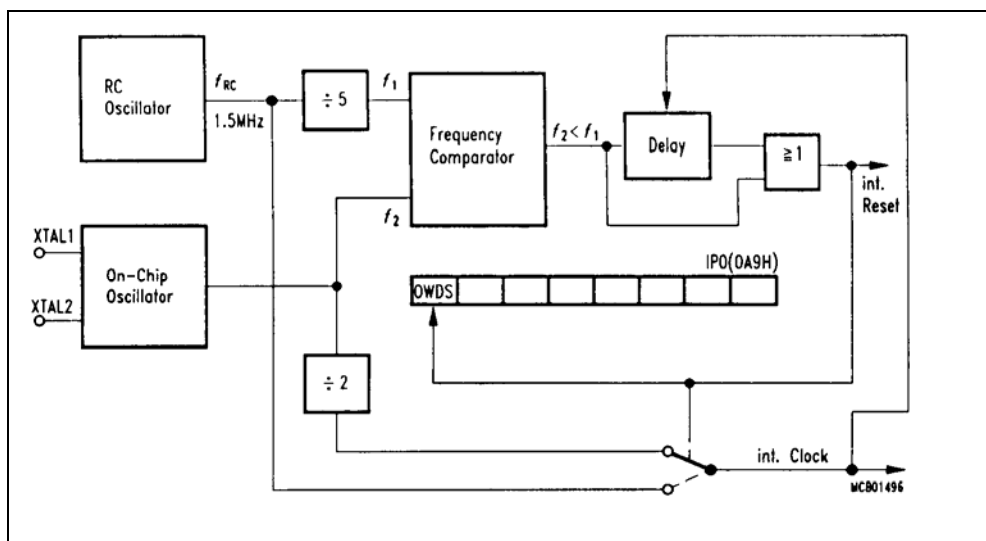


Abb. Funktionsschaltbild des Oszillator-Watchdogs im 80C517A



## 15 Oszillator und Taktversorgung

Die beiden Pins XTAL1 und XTAL2 sind der Eingang bzw. der Ausgang eines einstufigen, integrierten Inverters, der mit externen Bauteilen als Pierce-Oszillator konfiguriert werden kann. Dieser Oszillator treibt den internen Taktgenerator, der die Frequenz für die interne Taktversorgung durch zwei teilt. Der interne Systemtakt des 80C517/80C517A beträgt also  $f_{osz}/2$ . Damit werden die Phases 1 und 2, die States 1 bis 6 und die Maschinenzyklen festgelegt. Siehe Kapitel 3.

Nachstehendes Bild zeigt ein Schaltungsbeispiel unter Verwendung eines Quarzkristalls. In dieser Anwendung wird der integrierte Inverter als quarzgesteuerter Oszillator mit einer positiven Reaktanz beschattet. Kapazitäten von ca. 20 bis 30 pF zusammen mit einem für Mikroprozessoren empfohlenen Quarz haben sich bewährt. Unter Umständen kann auch in kostensensitiven Anwendungen ein keramischer Schwinger anstelle des Quarzkristalls eingesetzt werden. Dann sollten die die Werte der Kondensatoren C1 und C2 etwas höher gewählt werden, typischerweise 47pF. In jedem Fall sind Empfehlungen des Quarz- oder Keramischwingerherstellers zu beachten und die Schaltung gründlichen Tests unter verschiedenen Betriebsbedingungen zu unterwerfen (Variation von Spannung, Temperatur, Bauteiltoleranzen, etc.)

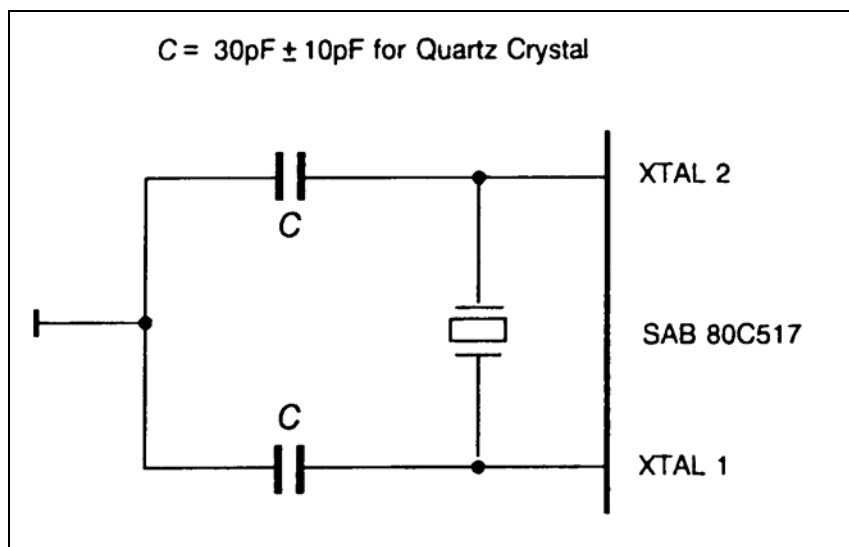


Abb. Empfohlene Oszillatorbeschaltung

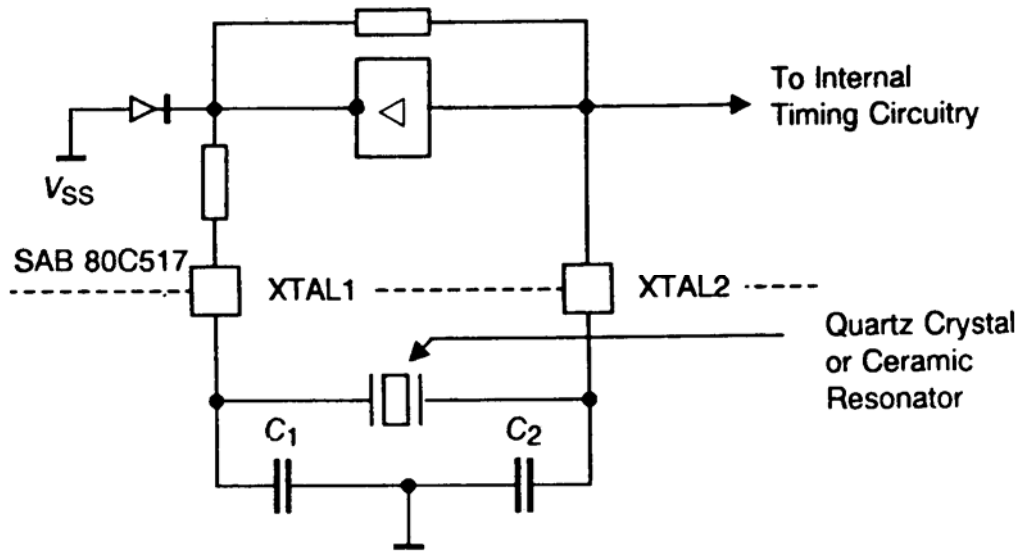


Abb. Detailliertes Schaltbild des On-Chip-Oszillators

Soll der Takt extern eingespeist werden, wird der interne Oszillator nicht verwendet. In diesem Fall muß der Takt an XTAL2 angelegt werden (Eingang des Oszillatorinverters und des Bausteintakts), während XTAL1 (Ausgang des Inverters) offen gelassen werden muß, siehe Bild 15-3. Um die Rauschempfindlichkeit zu mindern, wird dabei ein externer Pull-Up-Widerstand an XTAL2 empfohlen. Er ist aber nicht notwendig, wenn die Spannungspegel des Takts den Spezifikationen  $V_{IL}$  und  $V_{IH2}$  von XTAL2 entsprechen (siehe DC Characteristics im Datenblatt).

## 16 Interruptsystem

### 16.1 Einlesen eines Ports

Mikrocontroller stellen einige Ports zur Verfügung, über welche Daten ausgegeben und eingelesen werden können. Ports können verwendet werden, um Steuersignale auszugeben oder um Peripheriebausteine, wie z. B. eine LCD Anzeige, anzusprechen. Andererseits können über Portleitungen verschiedenste Geber und Melder, Sensoren und externe Peripheriebausteine wie zusätzliche A/D Umsetzer, Tastaturdekoder, Uhrenbausteine usw. eingelesen werden.

Als Beispiel für einen Peripheriebaustein soll ein Tastaturdekoder behandelt werden. Mit Hilfe dieses Tastaturdekoders wird eine Tastatur mit 20 Tasten, welche auf der Übungsplatine aufgebaut ist, abgefragt. Wenn eine Taste gedrückt wird, erzeugt der Dekoder eine 5 bit breite dual codierte Zahl, welche am 5bit breiten Datenbus des Dekoders ausgegeben wird. Zusätzlich dazu erzeugt der Dekoder ein Steuersignal, mit welchem er anzeigt, daß die Daten am Datenbus gültig sind. Abbildung 1 zeigt, wie die Tastatur über den Tastaturdekoder an den Mikrocontroller angeschlossen ist.

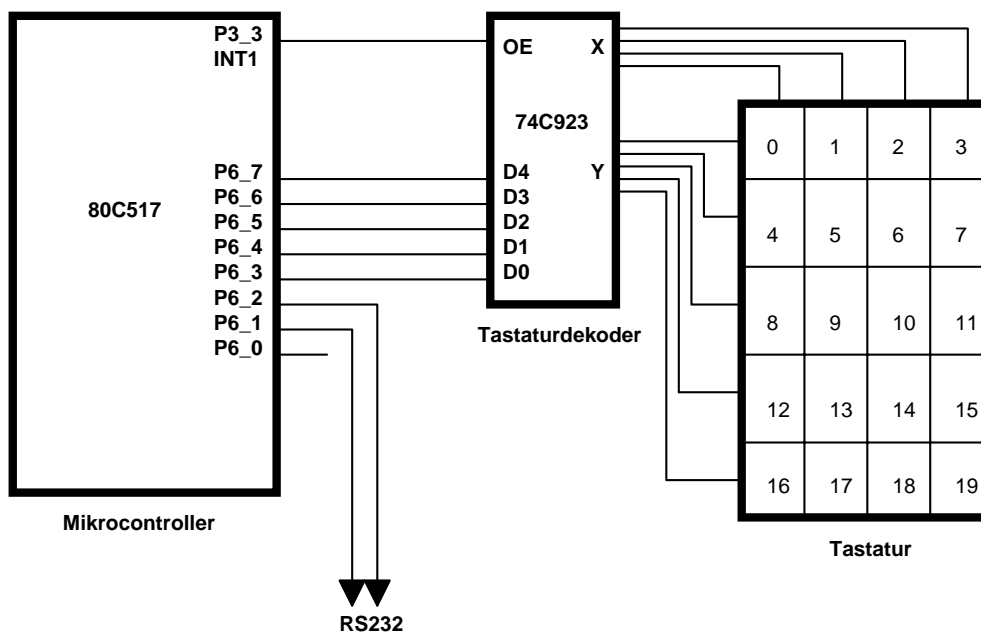


Abb. 1: Anschluß des Tastaturdekoders an den Mikrocontroller

Es gibt nun verschiedene Möglichkeiten, um den Wert des Tastaturdekoders in den Mikrocontroller einzulesen. Zwei Verfahren sollen besprochen werden:

- a) **Das Polling Verfahren**
- b) **Die Abfrage mit Hilfe einer Interrupt Service Routine**

### 16.2 Das Polling Verfahren

Bei diesem Verfahren fragt der Mikrocontroller den Port zyklisch ab, ob Daten anliegen. Die zyklische Abfrage kann auf verschiedene Arten realisiert werden. Die einfachste Möglichkeit ist, eine Schleife zu programmieren. Bei jedem Schleifen-durchlauf wird der Port 6 abgefragt und daraus der Tastenwert berechnet.

Der Nachteil dieser Methode ist, daß der Controller durch die zyklische Abfrage jedoch ständig beschäftigt ist.

Eine Möglichkeit, den Controller zu entlasten, ist der Einbau einer Zeitschleife. Der Controller arbeitet andere Programme ab, während im Hintergrund einer der internen 16-Bit Zähler läuft. Dieser kann so programmiert werden, daß er mit 1/12 des Controllertaktes getaktet wird. Der Zähler zählt also automatisch hoch. Wenn der Zähler von 0xFFFF auf 0x0000 überläuft, löst er ein Interrupt Signal aus. Durch dieses Signal wird eine Interrupt Service Routine aufgerufen, welche auch dazu verwendet werden kann, die Tastatur einzulesen.

Mit dieser Methode ist es möglich, die Tastatur z.B. alle 0.1 Sekunden abzufragen. In der Zwischenzeit steht der Controller voll für alle anderen Aufgaben zur Verfügung. Die für die Programmierung der Interrupt Routinen nötigen Kenntnisse werden im nächsten Kapitel behandelt.

## 16.3 Die Programmierung von Interrupt Routinen

### 16.3.1 Allgemeines

In einem Mikrocontroller gibt es verschiedene Baugruppen, die einen Interrupt auslösen können. Dies sind beim 80C517 z. B. die Timer 0 und 1, der A/D Umsetzer, die seriellen Schnittstellen, usw. Außerdem besteht die Möglichkeit, über Portleitungen sieben externe Interrupts auszulösen, welche von Endschaltern, Signalgebern, usw. erzeugt werden können. Der Mikrocontroller besitzt einige special function register in seinem internen Speicher, über welche die nötigen Einstellungen für die Bedienung der Interrupts durchgeführt werden können. Die einzelnen Bits, welche als Schalter dienen, sind mit Namen versehen und in Abb. 3a und Abb. 3b dargestellt.

## 16.4 Vektoradressen

Interrupt-Quelle	Request-Flags	Vektoradresse
Ext. Interrupt 0	IE0	0003H
Timer 0-Interrupt	TF0	000BH
Ext. Interrupt 1	IE1	0013H
Timer 1-Interrupt	TF1	001BH
Interrupt der ser. Schnittst. 0	RI0/TI0	0023H
Timer 2-Interrupt	TF2/EXF2	002BH
A/D-Wandler-Interrupt	IADC	0043H
Ext. Interrupt 2	IEX2	004BH
Ext. Interrupt 3	IEX3	0053H
Ext. Interrupt 4	IEX4	005BH
Ext. Interrupt 5	IEX5	0063H
Ext. Interrupt 6	IEX6	006BH
Interrupt der ser. Schnittst. 1	RI1/TI1	0083H
Compare-Int. in CM0 - CM7 (nur 80C517A)	ICMP0 - ICMP7	0093H
Compare-Timer-Interrupt	CTF	009BH
Compare-Int. in COMSET (nur 80C517A)	ICS	00A3H
Compare-Int. in COMCLR (nur 80C517A)	ICR	00ABH

Abb 2: Vektoradressen der einzelnen Interrupts

### 16.5 Interruptsystem 80C517

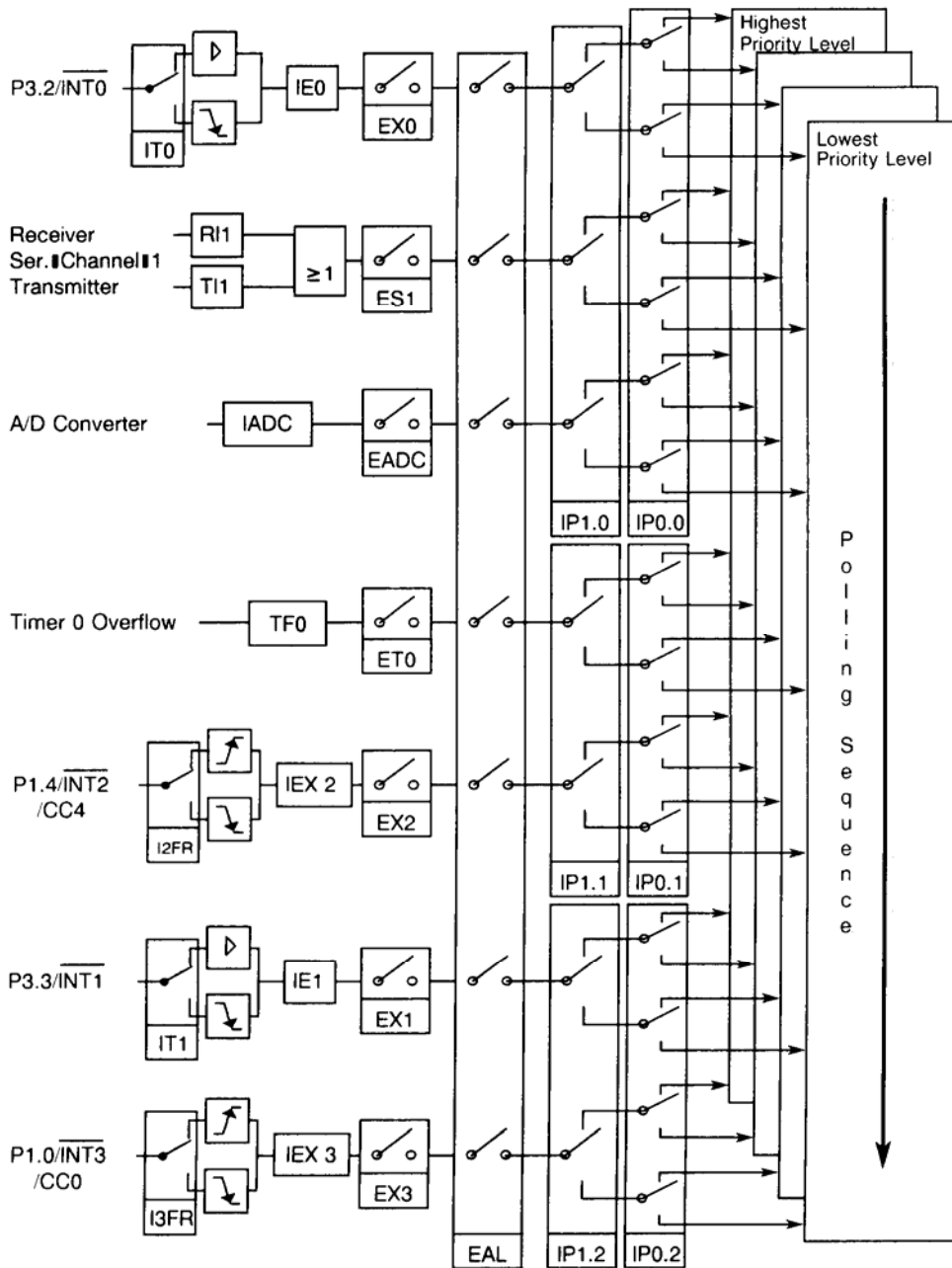


Abb. 3a: Interrupt Struktur des 80C517

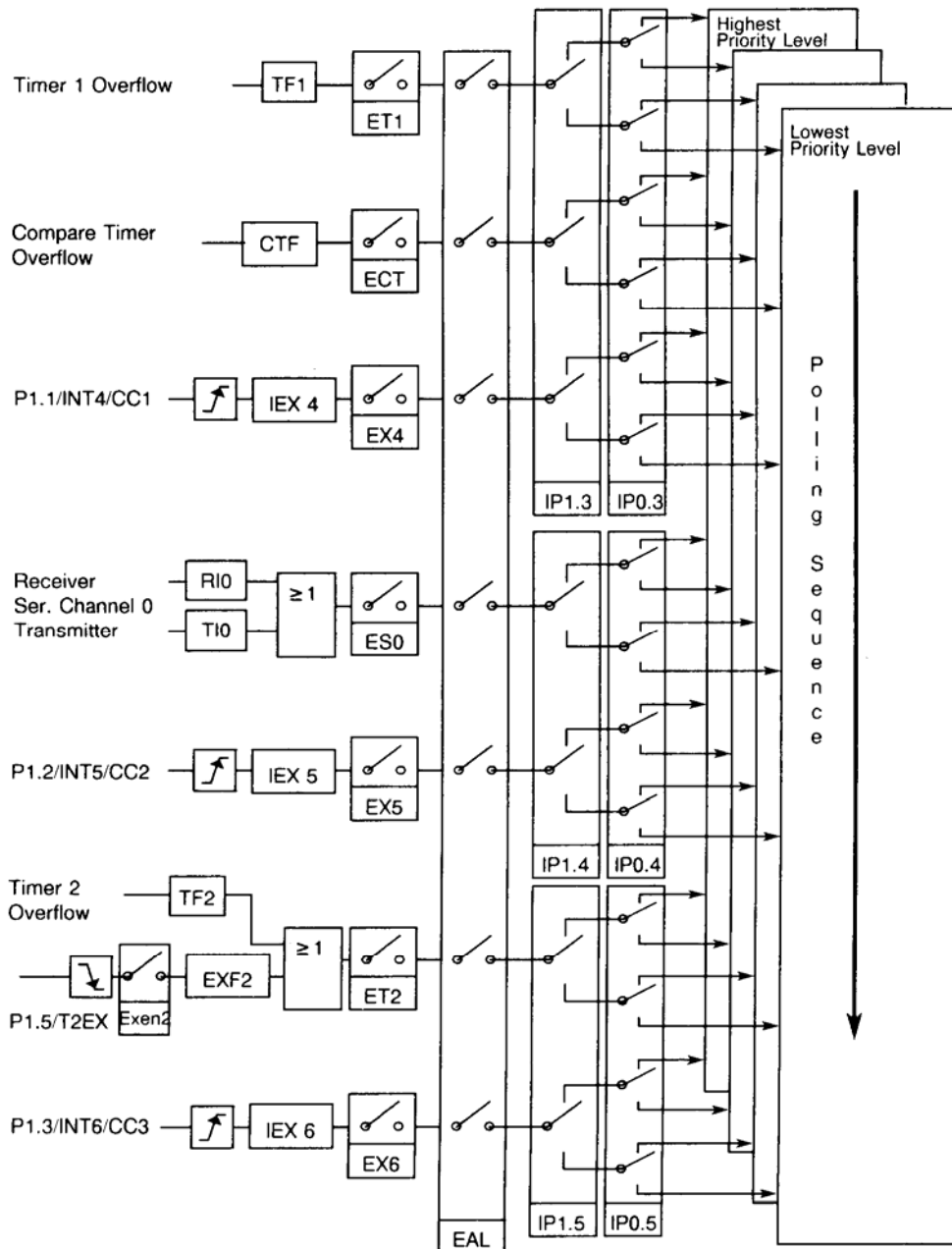


Abb. 2b: Interrupt Struktur des 80C517, Fortsetzung

Grundsätzlich sind zwei Schritte notwendig, um einen Interrupt überhaupt zu ermöglichen. Erstens muß die einzelne Interruptquelle eingeschaltet werden, anschließend müssen alle Interrupts allgemein eingeschaltet werden.

**Beispiel:** Der Timer 0 soll in einer Zeitschleife laufen und bei jedem Überlauf von 0xFFFF auf 0x0000 einen Interrupt auslösen.

Um den Interrupt zu erlauben, muß das Interrupt Enable Bit ET0 gesetzt werden ( Schalter ET0 wird geschlossen ). Außerdem muß das allgemeine Interrupt Enable Bit EAL gesetzt werden, das alle Interruptquellen gleichzeitig ein- und ausschaltet.

Die anschließenden Prioritätsregister IPX.X dienen dazu, festzulegen, welcher Interrupt wichtiger ist und zuerst bearbeitet wird, wenn mehrere Interruptsignale gleichzeitig auftreten. Sie werden im Rahmen dieser Übung auf ihrer Default Einstellung belassen.

## 16.6 Special Function Register für Interrupts

**IEN0 (A8H), bitadressierbar**

MSB						LSB	
<b>EAL</b>	<b>WDT</b>	<b>ET2</b>	<b>ES0</b>	<b>ET1</b>	<b>EX1</b>	<b>ET0</b>	<b>EX0</b>
AFH	AEH	ADH	ACH	ABH	AAH	A9H	A8H
Interrupt-Freigaberegister 0 (Interrupt Enable 0). Es enthält Freigabebits mehrerer Interrupts und ein Steuerbit des Watchdog-Timers. Reset-Wert: 00H							
<b>Bitsymbol</b>	<b>Funktion</b>						
EAL	Generelles Freigabebit aller Interrupts. Bei EAL = 0 wird kein angeforderter Interrupt bedient. Bei EAL = 1 gilt das individuelle Freigabebit des jeweiligen Interrupts.						
ET2	Freigabebit des Timer 2-Interrupts. Es gibt den Interrupt frei (ET2 = 1) bzw. sperrt ihn (ET2 = 0).						
ES0	Freigabebit des Interrupts der seriellen Schnittstelle 0. Es gibt den Interrupt frei (ES0 = 1) bzw. sperrt ihn (ES0 = 0).						
ET1	Freigabebit des Timer 1-Interrupts. Es gibt den Interrupt frei (ET1 = 1) bzw. sperrt ihn (ET1 = 0).						
EX1	Freigabebit des externen Interrupts 1. Es gibt den Interrupt frei (EX1 = 1) bzw. sperrt ihn (EX1 = 0).						
ET0	Freigabebit des Timer-0-Interrupts. Es gibt den Interrupt frei (ET0 = 1) bzw. sperrt ihn (ET0 = 0).						
EX0	Freigabebit des externen Interrupts 0. Es gibt den Interrupt frei (EX0 = 1) bzw. sperrt ihn (EX0 = 0).						

Bild 17-2: Special-Function-Register IEN0 (A8H)

**IEN1 (B8H), bitadressierbar**

MSB							LSB
EXEN2	SWDT	EX6	EX5	EX4	EX3	EX2	EADC
BFH	BEH	BDH	BCH	BBH	BAH	B9H	B8H
Interrupt-Freigaberegister 1 (Interrupt Enable 1). Es enthält Freigabebits mehrerer Interrupts und ein Steuerbit des Watchdog-Timers. Reset-Wert: 00H							
Bitsymbol	Funktion						
EXEN2	Freigabebit des extern gesteuerten Nachlade-Interrupts des Timers 2 (External Reload of Timer 2 Enable). Es gibt den Interrupt frei (EXEN2 = 1) bzw. sperrt ihn (EXEN2 = 0). Die Nachladefunktion selbst wird von EXEN2 nicht beeinflusst.						
EX6	Freigabebit des externen Interrupts 6 bzw. Compare/Capture-Interrupts 3 am Pin P1.3. Es gibt den Interrupt frei (EX6 = 1) bzw. sperrt ihn (EX6 = 0). Die Compare/Capture-Funktion wird von EX6 nicht beeinflusst.						
EX5	Freigabebit des externen Interrupts 5 bzw. Compare/Capture-Interrupts 2 am Pin P1.2. Es gibt den Interrupt frei (EX5 = 1) bzw. sperrt ihn (EX5 = 0). Die Compare/Capture-Funktion wird von EX5 nicht beeinflusst.						
EX4	Freigabebit des externen Interrupts 4 bzw. Compare/Capture-Interrupts 1 am Pin P1.1. Es gibt den Interrupt frei (EX4 = 1) bzw. sperrt ihn (EX4 = 0). Die Compare/Capture-Funktion wird von EX4 nicht beeinflusst.						
EX3	Freigabebit des externen Interrupts 3 bzw. Compare/Capture-Interrupts 0 am Pin P1.0. Es gibt den Interrupt frei (EX3 = 1) bzw. sperrt ihn (EX3 = 0). Die Compare/Capture-Funktion wird von EX3 nicht beeinflusst.						
EX2	Freigabebit des externen Interrupts 2 bzw. Compare/Capture-Interrupts 4 am Pin P1.4. Es gibt den Interrupt frei (EX2 = 1) bzw. sperrt ihn (EX2 = 0). Die Compare/Capture-Funktion wird von EX2 nicht beeinflusst.						
EADC	Freigabebit des A/D-Wandler-Interrupts. Es gibt den Interrupt frei (EADC = 1) bzw. sperrt ihn (EADC = 0).						

Bild 17-3: Special-Function-Register IEN1 (B8H)

**IEN2 (9AH), nicht bitadressierbar**

MSB						LSB	
-	-	ECR/-	ECS/-	ECT	ECMP/-	-	ES1
Interrupt-Freigaberegister 2 (Interrupt Enable Register 2). Es enthält mehrere Freigabebits für Interrupts. Reset-Wert: xxxx 0xx0B (80C517), xx00 00x0B (80C517A)							
Bitsymbol	Funktion						
ECR (80C517A)	Freigabebit des Compareregisters COMCLR (Set/Reset-Modus). Es gibt den Interrupt frei (ECR = 1) bzw. sperrt ihn (ECR = 0). Im 80C517 ist dieses Bit reserviert.						
ECS (80C517A)	Freigabebit des Compareregisters COMSET (Set/Reset-Modus). Es gibt den Interrupt frei (ECS = 1) bzw. sperrt ihn (ECS = 0). Im 80C517 ist dieses Bit reserviert.						
ECT	Freigabebit des Compare-Timer-Interrupts. Es gibt den Interrupt frei (ECT = 1) bzw. sperrt ihn (ECT = 0).						
ECMP (80C517A)	Freigabebit der Interrupts der Compareregister CM0 bis CM7 im Compare-Modus 1. Es gibt die Interrupts frei (ECMP = 1) bzw. sperrt sie (ECMP = 0). Im 80C517 ist dieses Bit reserviert.						
ES1	Freigabebit des Interrupts der seriellen Schnittstelle 1. Es gibt den Interrupt frei (ES1 = 1) bzw. sperrt ihn (ES1 = 0).						

Bild 17-4: Special-Function-Register IEN2 (9AH)



## TCON (88H), bitadressierbar

MSB				LSB			
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
8FH	8EH	8DH	8CH	8BH	8AH	89H	88H
Steuerregister für Timer 0 und 1 (Timer 0/1 Control Register). Neben Steuerbits für die Timer 0 und 1 enthält es auch Steuerbits für die Interrupts. Reset-Wert: 00H							
Bitsymbol	Funktion						
TF1	Interrupt-Request-Flag des Überlaufs von Timer 1. Es wird beim Einsprung in die Interrupt-Routine durch Hardware gelöscht.						
TF0	Interrupt-Request-Flag des Überlaufs von Timer 0. Es wird beim Einsprung in die Interrupt-Routine durch Hardware gelöscht.						
IE1	Interrupt-Request-Flag des externen Interrupts 1 (P3.3/INT1#). Es wird gesetzt, wenn eine fallende Flanke am Pin P3.3 entdeckt wird (Voraussetzung: mit IT1 wurde fallende Flanke selektiert). Im Fall der Low-Pegel-Aktivierung bleibt IE1 gesetzt, solange der Pegel anliegt, und kann nicht durch Software gelöscht werden.						
IT1	Selektionsbit für Interrupt 1 (Interrupt 1 Type Select Bit). Es legt fest, ob der Interrupt 1 durch eine fallende Flanke (IT1 = 1) oder durch einen Low-Pegel (IT1 = 0) ausgelöst wird.						
IE0	Interrupt-Request-Flag des externen Interrupts 0 (P3.2/INT0#). Es wird gesetzt, wenn eine fallende Flanke am Pin P3.2 entdeckt wird (Voraussetzung: mit IT0 wurde fallende Flanke selektiert). Im Fall der Low-Pegel-Aktivierung bleibt IE0 gesetzt, solange der Pegel anliegt, und kann nicht durch Software gelöscht werden.						
IT0	Selektionsbit für Interrupt 0 (Interrupt 0 Type Select Bit). Es legt fest, ob der Interrupt 0 durch eine fallende Flanke (IT0 = 1) oder durch einen Low-Pegel (IT0 = 0) ausgelöst wird.						

Bild 17-5: Special-Function-Register TCON (88H)

## T2CON (C8H), bitadressierbar

MSB						LSB	
T2PS	I3FR	I2FR	T2R1	T2R0	T2CM	T2I1	T2I0
CFH	CEH	CDH	CCH	CBH	CAH	C9H	C8H
Steuerregister des Timers 2 (Timer 2 Control Register). Es enthält neben Bits des Timers 2 auch Bits der Interrupt-Steuerung. Reset-Wert: 00H							
Bitsymbol	Funktion						
I3FR	Auswahlbit zur Einstellung der aktiven Flanke des externen Interrupts 3 (Interrupt 3 Falling/Rising Edge). Es legt fest, ob das Request Flag IEX3 gesetzt wird, wenn eine fallende (I3FR = 0) oder eine steigende (I3FR = 1) Flanke am Pin P1.0/INT3# entdeckt wird.						
I2FR	Auswahlbit zur Einstellung der aktiven Flanke des externen Interrupts 2 (Interrupt 2 Falling/Rising Edge). Es legt fest, ob das Request Flag IEX2 gesetzt wird, wenn eine fallende (I2FR = 0) oder eine steigende (I2FR = 1) Flanke am Pin P1.4/INT2# entdeckt wird.						

Bild 17-6: Special-Function-Register T2CON (C8H)

**IRCON/IRCON0 (C0H), bitadressierbar**

MSB						LSB	
<b>EXF2</b>	<b>TF2</b>	<b>IEX6</b>	<b>IEX5</b>	<b>IEX4</b>	<b>IEX3</b>	<b>IEX2</b>	<b>IADC</b>
C7H	C6H	C5H	C4H	C3H	C2H	C1H	C0H
Interrupt-Request-Register (Interrupt Request Control Register). Es enthält Flags zur Anforderung von verschiedenen Interrupts. Reset-Wert: 00H							
<b>Bitsymbol</b>	<b>Funktion</b>						
EXF2	Interrupt-Request-Flag des externen Timer 2-Nachlademodus (Timer 2 External Reload Flag). Es wird bei einer fallenden Flanke am Pin 1.5/T2EX gesetzt, wenn der externe Reload-Modus für den Timer 2 gewählt ist. Ist der Nachlade-Interrupt des Timers 2 freigegeben (EXEN2 = 1), wird in die Interrupt-Routine verzweigt. EXF2 muß durch Software gelöscht werden.						
TF2	Interrupt-Request-Flag des Überlaufs von Timer 2 (Timer 2 Overflow Flag). Bei einem Überlauf des Timers 2 wird es von der Hardware gesetzt. TF2 muß durch Software gelöscht werden.						
IEX6	Interrupt-Request-Flag des externen Interrupts 6. Es wird gesetzt, wenn eine steigende Flanke bzw. eine Compare-Flanke am Pin P1.3/INT6/CC3 auftritt. IEX6 wird beim Einsprung in die Interrupt-Routine durch Hardware gelöscht, es kann aber auch durch Software gelöscht werden.						
IEX5	Interrupt-Request-Flag des externen Interrupts 5. Es wird gesetzt, wenn eine steigende Flanke bzw. eine Compare-Flanke am Pin P1.2/INT5/CC2 auftritt. IEX5 wird beim Einsprung in die Interrupt-Routine durch Hardware gelöscht, es kann aber auch durch Software gelöscht werden.						
IEX4	Interrupt-Request-Flag des externen Interrupts 4. Es wird gesetzt, wenn eine steigende Flanke bzw. eine Compare-Flanke am Pin P1.1/INT4/CC1 auftritt. IEX4 wird beim Einsprung in die Interrupt-Routine durch Hardware gelöscht, es kann aber auch durch Software gelöscht werden.						
IEX3	Interrupt-Request-Flag des externen Interrupts 3. Es wird gesetzt, wenn eine steigende/fallende Flanke (abhängig von I3FR in T2CON) bzw. eine Compare-Flanke am Pin P1.0/INT3#/CC0 auftritt. IEX3 wird beim Einsprung in die Interrupt-Routine durch Hardware gelöscht, es kann aber auch durch Software gelöscht werden.						

Bild 17-7 a): Special-Function-Register IRCON/IRCON0 (C0H)

Bitsymbol	Funktion
IEX2	Interrupt-Request-Flag des externen Interrupts 2. Es wird gesetzt, wenn eine steigende/fallende Flanke (abhängig von I2FR in T2CON) bzw. eine Compare-Flanke am Pin P1.4/INT2#/CC4 auftritt. IEX2 wird beim Einsprung in die Interrupt-Routine durch Hardware gelöscht, es kann aber auch durch Software gelöscht werden.
IADC	Interrupt-Request-Flag des A/D-Wandlers. Es wird am Ende einer A/D-Wandlung von der Hardware gesetzt. Es muß durch Software gelöscht werden.

Bild 17-7 b): Special-Function-Register IRCON/IRCON0 (C0H), Fortsetzung

**CTCON (E1H), nicht bitadressierbar**

MSB				LSB			
T2PS1	-	ICR/-	ICS/-	CTF	CLK2	CLK1	CLK0
Steuerregister des Compare-Timers (Compare Timer Control Register). Es enthält neben Steuerbits für den Compare-Timer und den Timer 2 mehrere Interrupt-Request-Flags. Reset-Wert: 0xxx 0000B (80C517), 0x00 0000B (80C517A)							
Bitsymbol	Funktion						
ICR (80C517A)	Interrupt-Request-Flag des Compare-Interrupts mit COMCLR. Es wird im Set/Reset-Modus bei Übereinstimmung zwischen Timer 2 und COMCLR gesetzt. ICR wird beim Einsprung in die Interrupt-Routine durch Hardware gelöscht, kann aber auch durch Software gelöscht werden. Dieses Bit ist im 80C517 reserviert.						
ICS (80C517A)	Interrupt-Request-Flag des Compare-Interrupts mit COMSET. Es wird im Set/Reset-Modus bei Übereinstimmung zwischen Timer 2 und COMSET gesetzt. ICS wird beim Einsprung in die Interrupt-Routine durch Hardware gelöscht, kann aber auch durch Software gelöscht werden. Dieses Bit ist im 80C517 reserviert.						
CTF	Interrupt-Request-Flag des Compare-Timer-Überlaufs. Es wird bei Überlauf oder Nachladen des Compare-Timers gesetzt. CTF muß durch Software gelöscht werden.						

Bild 17-8: Special-Function-Register CTCON (E1H)

Der externe Interrupt 3 verhält sich völlig analog zum externen Interrupt 2. Das Request Flag ist IEX3 (in Register IRCON/IRCON0, Abbildung 17-7). Ebenso besteht die Wahlmöglichkeit, ob der Interrupt-Eingang P1.0/INT3#/CC0 auf positive oder negative Flanken reagiert; dies wird durch Bit I3FR gesteuert. Auch hier reagiert das Request Flag IEX3 auf Compare-Ereignisse im angeschlossenen Compare-Register CC0; dies geschieht bei allen Compare-Modi und unabhängig davon, ob und welcher Flankenwechsel durch diesen Compare am Pin erzeugt wird. Das Interrupt-Request-Flag IEX3 wird beim Sprung in die Interrupt-Routine automatisch zurückgesetzt.

## 16.7 Prioritätssteuerung

IP0 (A9H), nicht bitadressierbar

IP1 (B9H), nicht bitadressierbar

MSB				LSB			
OWDS	WDTS	IP0.5	IP0.4	IP0.3	IP0.2	IP0.1	IP0.0
-	-	IP1.5	IP1.4	IP1.3	IP1.2	IP1.1	IP1.0
Interrupt-Prioritätsregister 0 und 1. Neben den Statusflags für den Oszillator-Watchdog und den Watchdog-Timer enthalten sie Bits zur Einstellung von Interrupt-Prioritäten. Reset-Wert: IP0 = 00H, IP1 = xx00 0000B							
Bitsymbol	Funktion						
IP1.x IP0. x	Prioritätsprogrammierung der entsprechenden Interrupt-Gruppe (s. unten)						
0	0	setzt Prioritätsstufe 0 (niedrigste Priorität)					
0	1	setzt Prioritätsstufe 1					
1	0	setzt Prioritätsstufe 2					
1	1	setzt Prioritätsstufe 3 (höchste Priorität)					
		Bitpaar	Interrupt-Gruppe				
		IP1.0/IP0.0	Ext. Interrupt 0	Ser. Schnittst. 1	A/D-Wandler		
		IP1.1/IP0.1	Timer 0	-	Ext. Interrupt 2		
		IP1.2/IP0.2	Ext. Interrupt 1	Compare mit CMx *)	Ext. Interrupt 3		
		IP1.3/IP0.3	Timer 1	Compare-Timer	Ext. Interrupt 4		
		IP1.4/IP0.4	Ser. Schnittst. 0	Comp. mit COMSET *)	Ext. Interrupt 5		
		IP1.5/IP0.5	Timer 2	Comp. mit COMCLR *)	Ext. Interrupt 6		
		*) nur im 80C517A					

Bild 17-12: Special-Function-Register IP0 (A9H) und IP1 (B9H)

Hoch	->	Niedrig	Priorität
IE0	RI1/TI1	IADC	Hoch
TF0	-	IEX2	
IE1	ICMP0 - ICMP7 *)	IEX3	
TF1	CTF	IEX4	
RI0/TI0	ICS *)	IEX5	V
TF2/EXF2	ICR *)	IEX6	
			Niedrig

Bild 17-13: Rangfolge innerhalb einer Prioritätsstufe

### 16.8 Eine Interruptgesteuerte Zeitschleife

Alle Controller der Familie 8051 enthalten mehrere Timer/Zähler Bausteine. Diese Baugruppen können in verschiedenen Betriebsarten betrieben werden. Für die Zeitschleife soll der Timer/Zähler 0 als 16-Bit Zähler verwendet werden (Betriebsart 1).

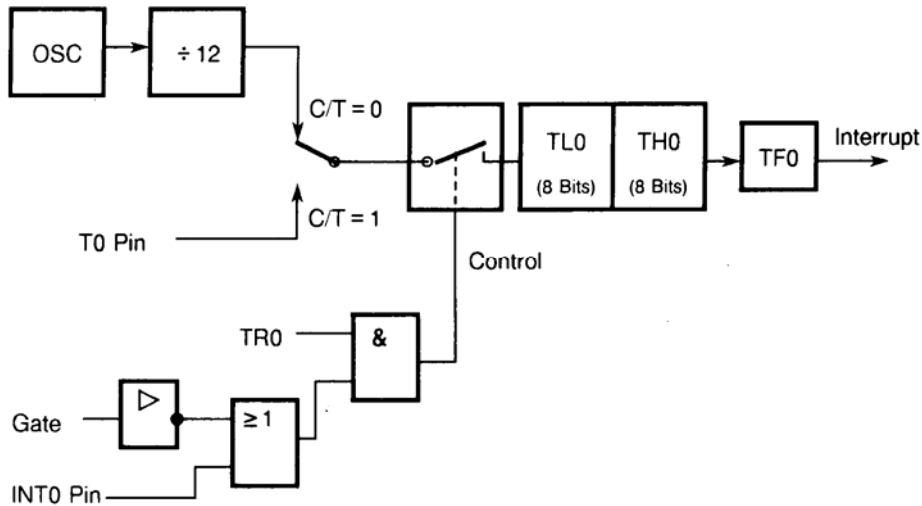


Abb. 3: Timer/Zähler 0, Betriebsart 1: 16-Bit Timer/Zähler  
Die Einstellungen werden über das special function register TMOD vorgenommen.

**TMOD (89H), nicht bitadressierbar**

MSB				LSB			
Timer 1				Timer 0			
Gate	C/T#	M1	M0	Gate	C/T#	M1	M0
Modusregister für Timer 0 und 1 (Timer Mode). Es enthält die Bits zur Einstellung der Betriebsarten für die Timer 0 und 1. In den Erläuterungen steht x für 0 oder 1, je nachdem, welcher Timer verwendet wird. Reset-Wert: 00H							
Bitsymbol	Funktion						
Gate	Wenn dieses Bit gesetzt ist, läuft der Timer nur dann, wenn er mit TRx freigegeben ist und gleichzeitig der Portpin P3.2/INT0# (Timer 0) bzw. P3.3/INT1# (Timer 1) auf High-Pegel ist.						
C/T#	Dieses Bit legt fest, ob Timer- oder Counter-Modus verwendet wird. C/T# = 0 wählt Timer- und C/T# = 1 wählt Counter-Modus.						
M0	M1	Arbeitsmodi					
0	0	THx dient als 8-bit-Timer; TLx bildet einen 5-bit-Vorteiler.					
0	1	THx und TLx bilden einen 16-bit-Timer.					
1	0	Auto-Reload-Timer (8 bit). Der Inhalt von THx wird beim Timerüberlauf nach TLx kopiert. THx selbst bleibt unverändert.					
1	1	Timer 0: Beide Register THx und TLx arbeiten als eigenständige 8-bit-Timer. TLx wird durch die Steuerbits von Timer 0, THx durch die Steuerbits von Timer 1 eingestellt. Timer 1: Timer 1 stoppt in dieser Betriebsart.					

Abb. 4: SFR TMOD

Die richtige Einstellung für die benötigte Betriebsart lautet: **TMOD = 0x01;**  
 Um eine Zeitschleife mit einer definierten Zeit aufzubauen, geht man folgendermaßen vor:  
 Der Timer wird mit 1/12 der Quarzfrequenz getaktet. In unserem Fall ist dies 1 MHz. Das Zählregister TL0, welches 8 Bit breit ist, muß von 0 bis 255 zählen, um einmal überzulaufen. Erst dann wird der Inhalt des Zählregister TH0 um Eins erhöht.

D. h., das Register TL0 teilt die Frequenz durch 256. Es ergibt sich eine Zählfrequenz von 3906.25 Hertz. Soll die Tastatur alle 1/100 Sekunden eingelesen werden, müssen die 3906.25 Hertz nochmals durch 39.0625, also ungefähr 39 geteilt werden. Dies erreicht man, wenn das Register TH0 mit dem Wert -39 ( = 217 ) vorgesetzt wird. Durch diese Einstellung wird alle 10 msec ein Interrupt ausgelöst.

Ein vollständiges Interruptprogramm:

```
void INTTIM0 (void) interrupt 1
{
  taste = P6;
  taste = taste >> 3;
  TH0 = -39;
}
```

### 16.9 Interruptquellen und -nummern

Interrupt-Quellen	Intr. Nr.	auslösendes Moment	Request -Flag	Rücksetzen	Einsprung-adressen
Externer Interrupt 0	0	neg. Flanke/Pegel	IE0	H	3H
Timer-0-Interrupt	1	Timer-0-Überlauf	TF0	H	BH
Externer Interrupt 1	2	neg. Flanke/Pegel	IE1	H	13H
Timer-1-Interrupt	3	Timer-1-Überlauf	TF1	H	1BH
Interrupt der seriellen Schnittstelle	4	Ende der Eingabe und Ausgabe	RI + TI	S	23H
Timer-2-Interrupt	5	Timer-2-Überlauf externer Reload	TF2 + EXF2	S	2BH
A/D-Wandler	8	Ende der Wandlung	IADC	S	43H
Externer Interrupt 2	9	neg./pos. Flanke	IEX2	H	4BH
Externer Interrupt 3/ Capture-0-Eingang/ Compare-0-Ausgang	10	neg./pos. Flanke neg./pos. Flanke neg./pos. Flanke	IEX3	H	53H
Externer Interrupt 4/ Capture-1-Eingang/ Compare-1-Ausgang	11	pos. Flanke pos. Flanke neg./pos. Flanke	IEX4	H	5BH
Externer Interrupt 5/ Capture-2-Eingang/ Compare-2-Ausgang	12	pos. Flanke pos. Flanke neg./pos. Flanke	IEX5	H	63H
Externer Interrupt 6/ Capture-3-Eingang/ Compare-3-Ausgang	13	pos. Flanke pos. Flanke neg./pos. Flanke	IEX6	H	6BH

Rücksetzen durch: H = Hardware; S = Software

Bild 9-1: Interrupt-Quellen und Einsprungadressen des 8051 - 80515

## 17 Literaturnachweis

1. KEIL-Elektronik GmbH, Bretonischer Ring 15, D-8011 Grasbrunn b. München „Der C-51 Compiler“ Bedienungsanleitung und Softwaremanual Version 3.40 (4.0).
2. MC-TOOLS Nr. 5 , Handbuch des 80C517 und 80C517A, R.Johannis/N. Papadopulos, Verlag: Feger + Reith, Hardware+Software Verlags OHG, Traunstein
3. Halbleiter-Schaltungstechnik, U.Tietze, Ch. Schenk, Springer-Verlag
4. VALVO, Die 8bit-Mikrocontroller-Familie 8051
5. Vogel Fachbuch, Elektronik 5, Mikroprozessortechnik, Müller/Walz
6. Fachzeitschrift elector, 10/94ff,
7. Datenbuch 80C517 der Firma Siemens
8. Fachzeitschrift elrad
9. Fachzeitschrift elektronik