Embedded Systemarchitekturen und Echtzeitbetriebssysteme

Inhaltsverzeichnis

Document History

Version, Date	Author(s) email address	Changes and other notes
22.5.2007	ludwig.eckert@fh-sw.de	Adressraumberechungen

10. AUFBAUSTRUKTUR VON EMBEDDED SYSTEMEN - ÜBERBLICK

10.1.1. Aufbau eines Embedded Systems

- CPU
- MCU (MCU = CPU mit On-Chip Peripherie)
- On-Board Peripherie
- Schnittstellen
- Watchdog
- Diagnosemöglichkeiten mit JTAG
- Memory Map eines Embedded Systems

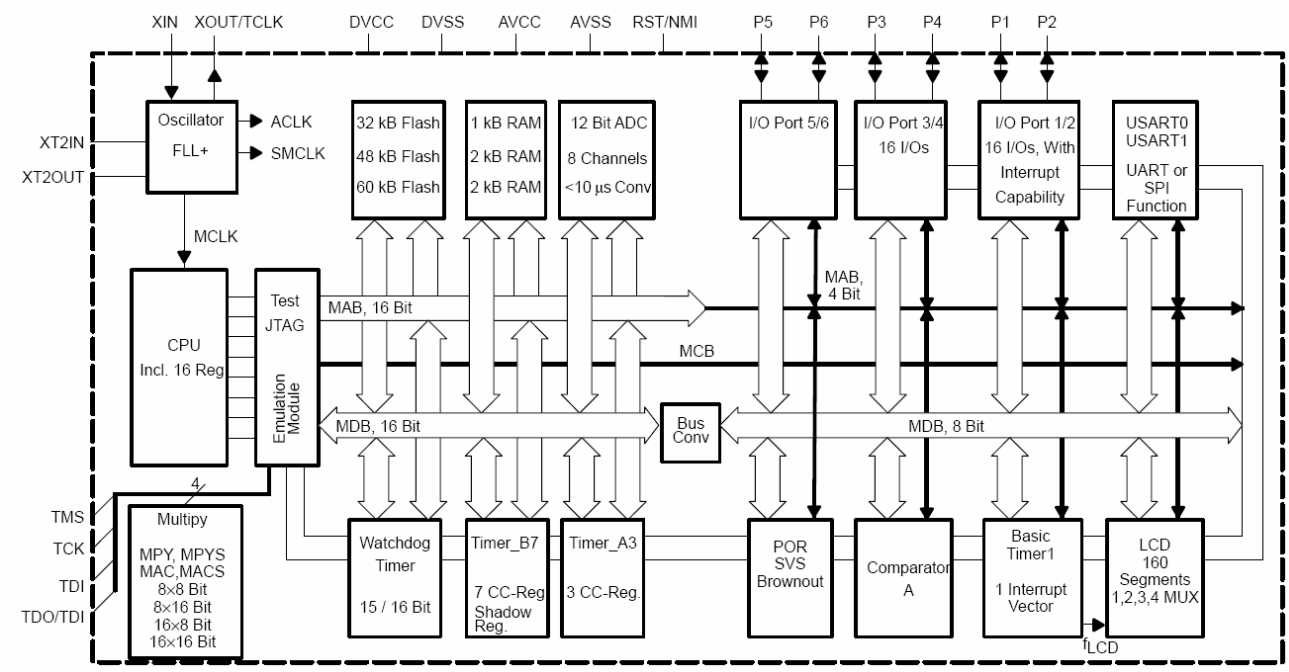


Bild: MSP430x44x functional block diagrams

Zentraleinheit

Die Zentraleinheit (Central Processing Unit CPU) stellt den Rechnerkern dar, um den sich alles dreht. Sie realisiert die Ablaufsteuerung des Rechners selbst, die Durchführung arithmetisch-logischer Operationen, den Transfer von Daten und den Ablauf des Programms.

Bei der Familie MSP430 handelt es sich um eine 16-Bit-RISC-CPU mit verschiedenen universellen Registern, einem breiten Spektrum an Adressierarten, einem kleinen, aber flexiblen Befehlssatz, alles ausgelegt auf hohe Leistung bei besonders kleinem Stromverbrauch.

Zum Test des gesamten Systems und zur Programmierung ist die CPU von einer JTAG-Schnittstelle umgeben, die es erlaubt, auf den gesamten Mikrocontroller mit nur vier Leitungen effektiv zuzugreifen.

Peripherieeinheiten

Die Familie MSP430 zeichnet sich durch eine breite Palette intelligenter und voneinander und von der CPU weitestgehend unabhängiger Peripherie aus, die die Realisierung komplexer Systeme mit nur geringer externer Beschaltung erlaubt.

Die Peripherie ist in Modulen realisiert, die mit der CPU durch den Speicheradressbus MAB, den Speicherdatenbus MDB und Unterbrechungsdienst- und -anforderungsleitungen verbunden sind.

Die Befehlsausführung bei 16-Bit-Modulen arbeitet ohne Einschränkungen (Watchdog, ADC, TIMER A, Hardware-Multiplier). Die 8 bit-Peripheriemodule sind byteorientiert. Ihre SFR's werden ausnahmslos mit einem Datenbereich von 8 Bit betrieben.

11. KOMPONENTEN EINES EMBEDDED SYSTEMS - DIE CPU

Ziele:

Funktionsweise CPU

Bussystem (Adress,- Daten-,

Bustimings (Lesezugriff, Schreibzugriff)

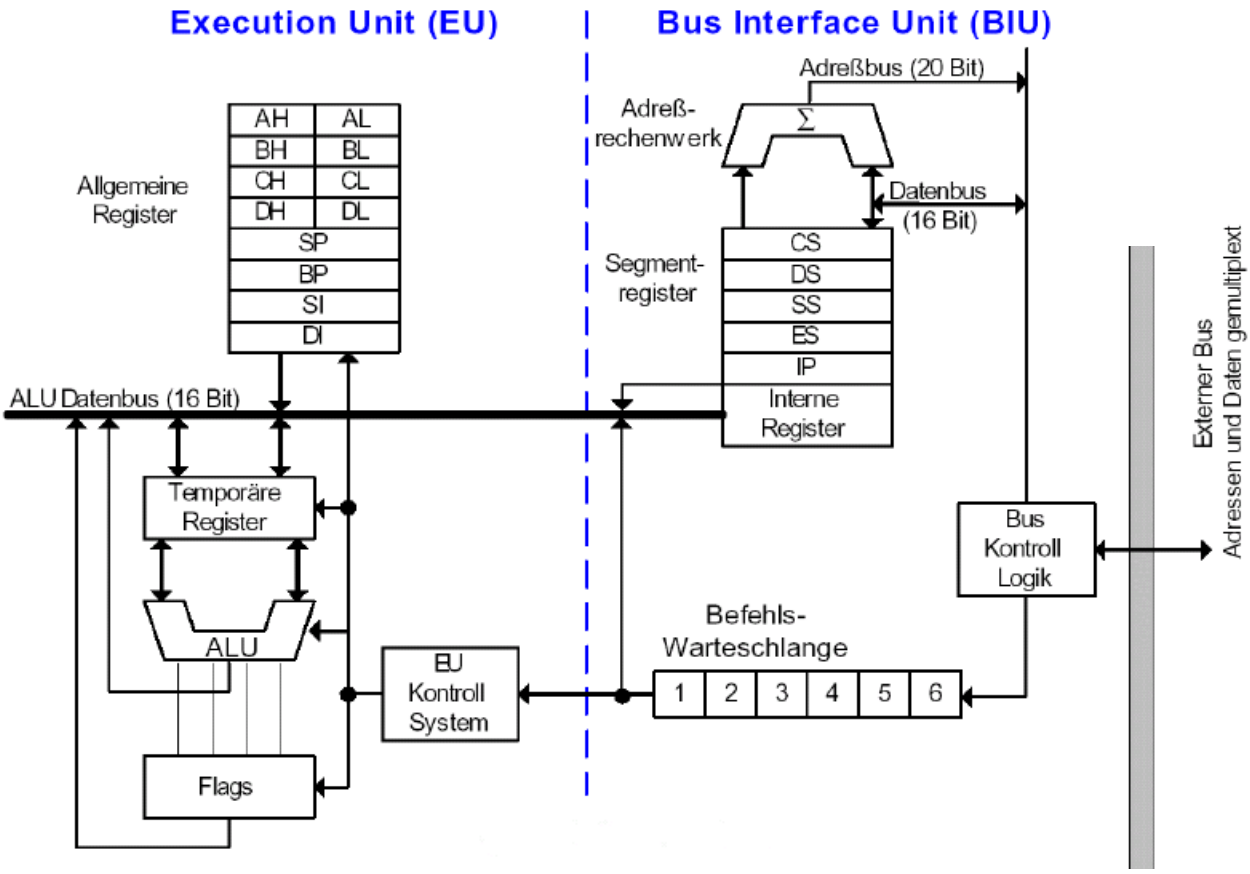


Bild: Aufbaustruktur der CPU i8086 (Hersteller Intel)

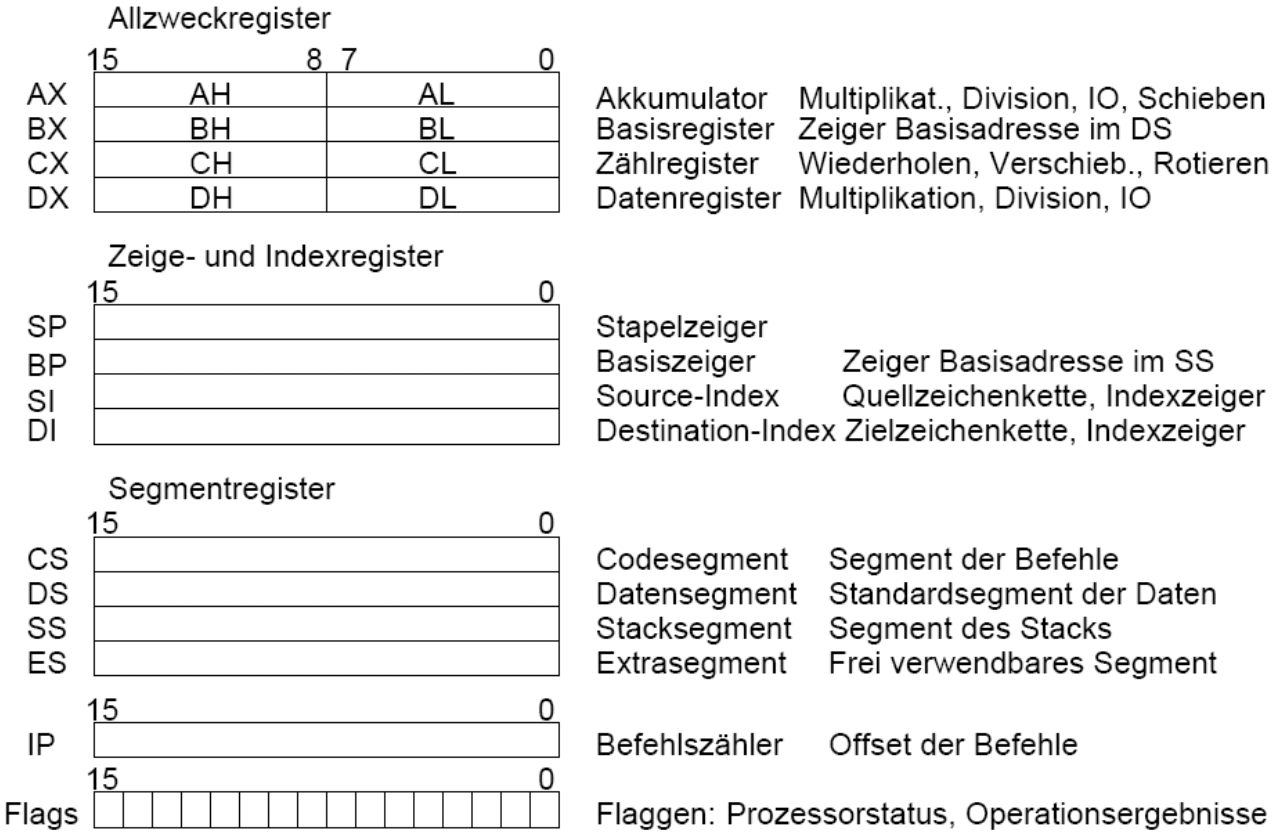
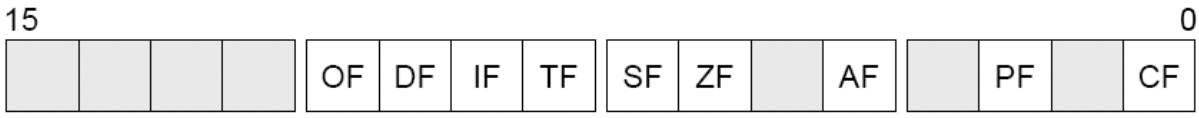


Bild: Registerstruktur der i8086 CPU

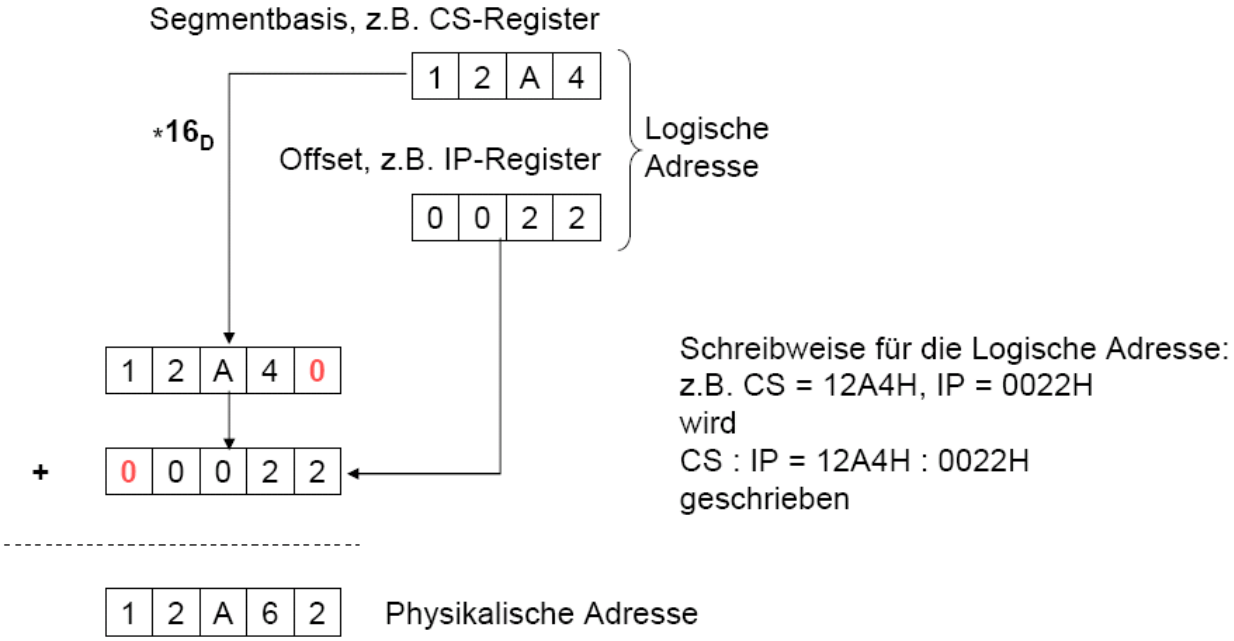
- CF - Carry Flag (Übertrags-Flagge):
  Zeigt an, ob bei arithmetischen Operationen der Rechenbereich des Registers überschritten wurde. Carry kann per Befehl gesetzt, gelöscht, komplementiert werden.
  → Sequentielle Addition und Subtraktion mit größerer Wortbreite als 16 Bit.
- PF - Parity Flag (Paritäts-Flagge):
  Gesetzt, wenn niederwertiges Byte des Ergebnisses eine gerade Anzahl Einsen enthält
  → Prüfung mit Quersumme z.B. bei serieller Datenübertragung.
- AF - Auxiliary Carry Flag (Hilfsübertrags-Flagge):
  Zeigt Übertrag von Bit 3 nach Bit 4 an.
  → Halbbyteüberlauf, für Arbeit mit BCD-Zahlen
- ZF - Zero Flag (Null-Flagge):
  Gesetzt, wenn das Ergebnis der letzten Operation Null war (Arithmetik, Logikbefehle)
  → Bedingte Programmverzweigungen, Zählschleifen
- SF - Sign Flag (Vorzeichen-Flagge):
  Kopie des Vorzeichens des Ergebnisses; gibt an, ob zuletzt durchgeführte arithmetische oder logische Operation ein positives (0) oder negatives (1) Ergebnis hatte
  → Bedingte Programmverzweigungen, 2er-Komplement Arithmetik

Bild: Das Flag- Register und die Bedeutung der Flags (1)



- TF - Trap Flag (Einzelschritt-Flagge):
  Versetzt den Prozessor in den Single-Step-Mode (Einzelschrittmodus). Prozessor führt nach jedem Befehl einen Software-Interrupt aus (INT 01H).
  → Zum Debuggen
- IF - Interrupt Flag (Unterbrechungs-Flagge):
  Kann per Befehl (STI, CLI) gesetzt beziehungsweise zurückgesetzt werden.
  1: Prozessor lässt keine von außen kommenden Unterbrechungsanforderungen zu.
  → Systemprogrammierung
- DF - Direction Flag (Richtungs-Flagge):
  Verarbeitungsrichtung bei Zeichenketten-Operationen (Strings)
  0: in aufsteigender Richtung der Speicheradressen (Setzen/Löschen mit den Befehlen STD und CLD)
- OF - Overflow Flag (Überlauf-Flagge):
  Gesetzt, falls Ergebnis einer Operation für Zieloperand zu groß oder zu klein, also Bereichsüberschreitung bei Arithmetik im 2er-Komplement, d.h. Überlauf in Vorzeichenstelle.
  → Arithmetik im 2er-Komplement

Bild: Das Flag-Register und die Bedeutung der Flags (2)



Physikalische Adresse = Offsetadresse + Segmentadresse * 16

Bild: Segmentierung und Adressbildung beim i8086

Art des Speicherzugriffs	Verwendete Segmentregister		Offset
	Standard-Zuweisung (feste Zuweisung)	Zuweisung durch Überschreiben	
Instruction Fetch	CS	./.	IP (feste Zuweisung)
Stack-Operation	SS	./.	SP (feste Zuweisung)
Variablen-Zugriff Ausnahmen: Stringoperationen Quelloperand Zieloperand	DS DS ES	CS, ES, SS CS, ES, SS ./.	Effektive Adresse (Offset) SI DI (feste Zuweisung)
BP-Register bei indizierter Basis-Adresse	SS	CS, DS, ES	Effektive Adresse = [BP]+Displacement

Bild: Verwendung der Segmentregister für welche Operation Codes

Das Speichermodell bestimmt u.a., wie der Assembler die Segmentbefehle zu behandeln hat.
Sprünge: NEAR - Nur der IP wird neu gesetzt
FAR - IP und CS werden neu gesetzt
CALL: NEAR - Nur der IP wird auf den Stack geschoben
FAR - IP und CS werden auf den Stack geschoben
RET Wie CALL, nur umgekehrt

- TINY:
  Daten (und Stack) und Code zusammen maximal 64 Kbyte; Daten NEAR, Code NEAR;
- SMALL:
  Daten (und Stack) und Code jeweils maximal 64 Kbyte; Daten NEAR, Code NEAR;
- MEDIUM:
  Daten maximal 64 Kbyte, Code mehr als 64 KByte; Daten NEAR, Code FAR;
- COMPACT:
  Daten mehr als 64 KByte, Code maximal 64 Kbyte; Daten NEAR, Code NEAR;
  Compiler: Datenfeld (Arrays) <= 64 KByte, innerhalb des Datenfeldes wird NEAR-Adressierung benutzt;
- LARGE:
  Daten und Code jeweils mehr als 64 Kbyte; Daten FAR, Code FAR;
  Compiler: Datenfeld (Arrays) <= 64 KByte, innerhalb des Datenfeldes wird NEAR-Adressierung benutzt;
- HUGE: Daten und Code jeweils mehr als 64 Kbyte; Daten FAR, Code FAR.
  Compiler: Datenfeld (Arrays) >= 64 KByte, innerhalb des Datenfeldes wird FAR-Adressierung benutzt;

Bild: Speichermodelle beim i8086

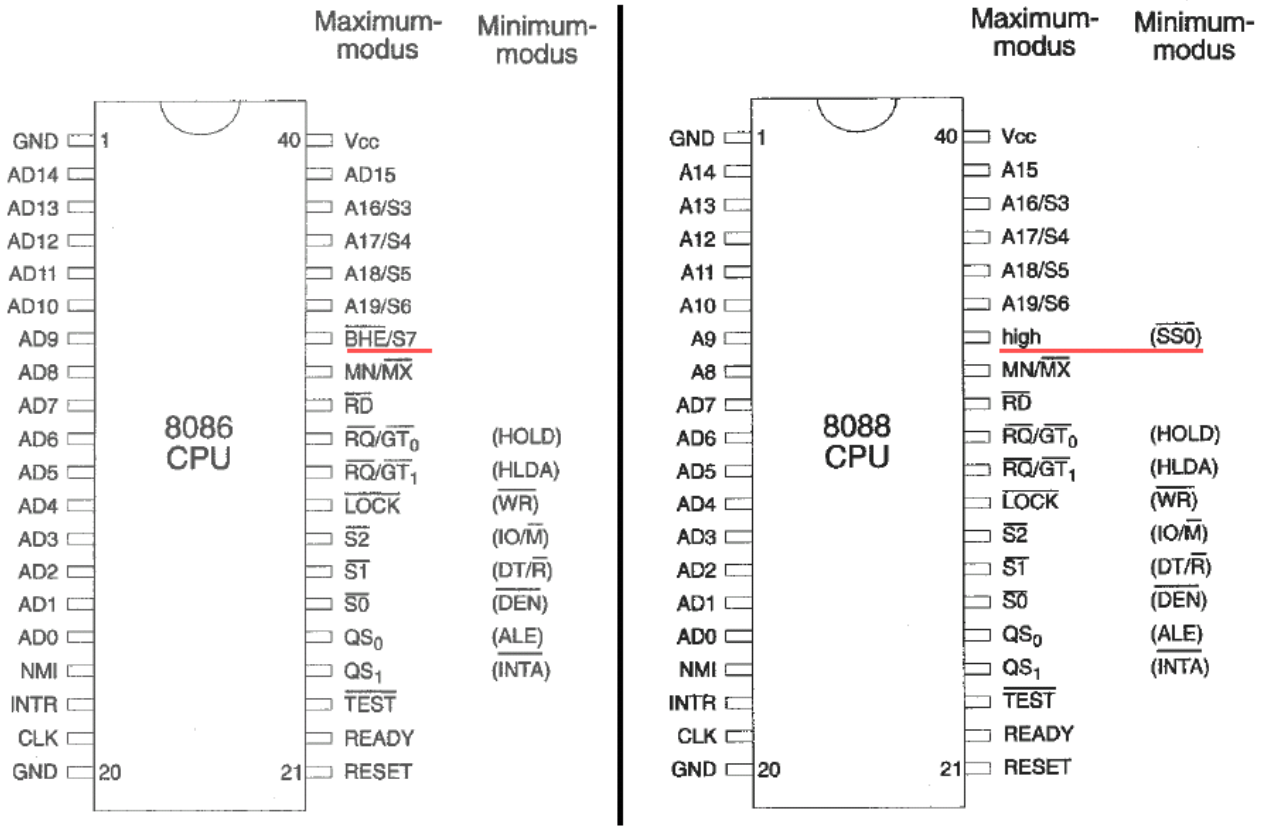
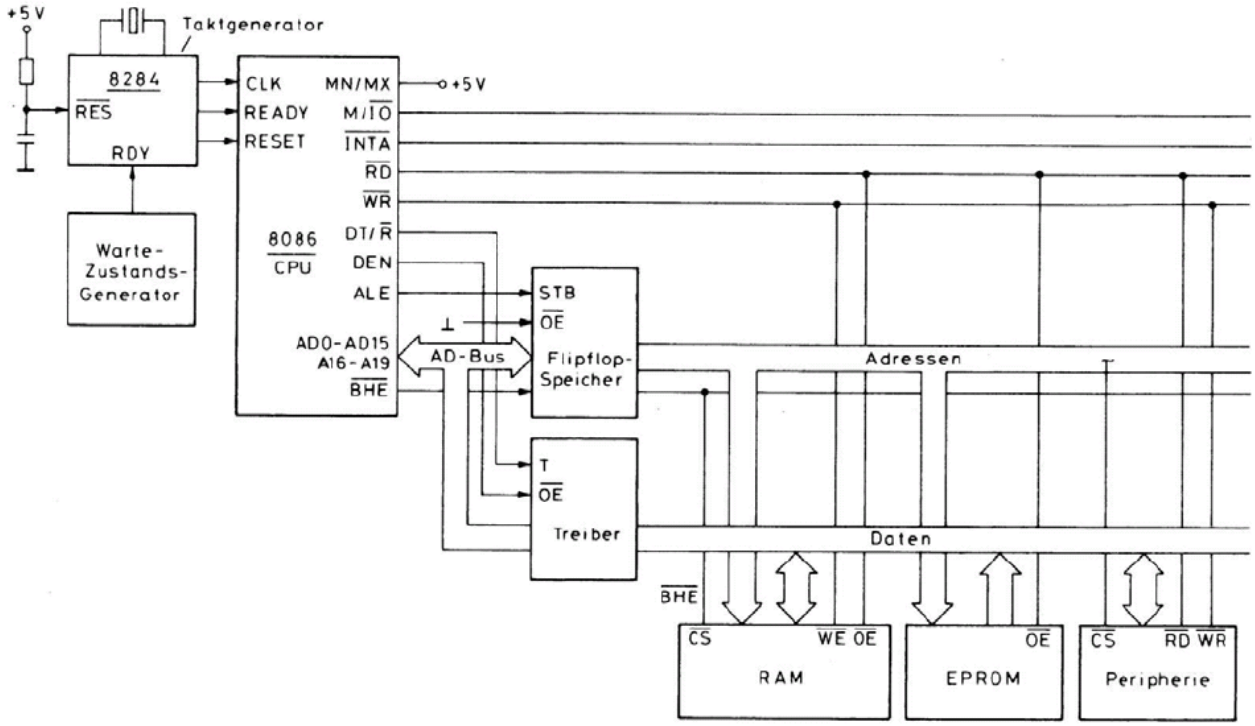


Bild: Pinbelegungen der CPU i8086 und i8088



/BHE - Byte High Enable
DEN - Data Enable
DT/-R - Data Transmitt -Receive

Bild: Minimalsystem mit RAM, EPROM und Peripherie

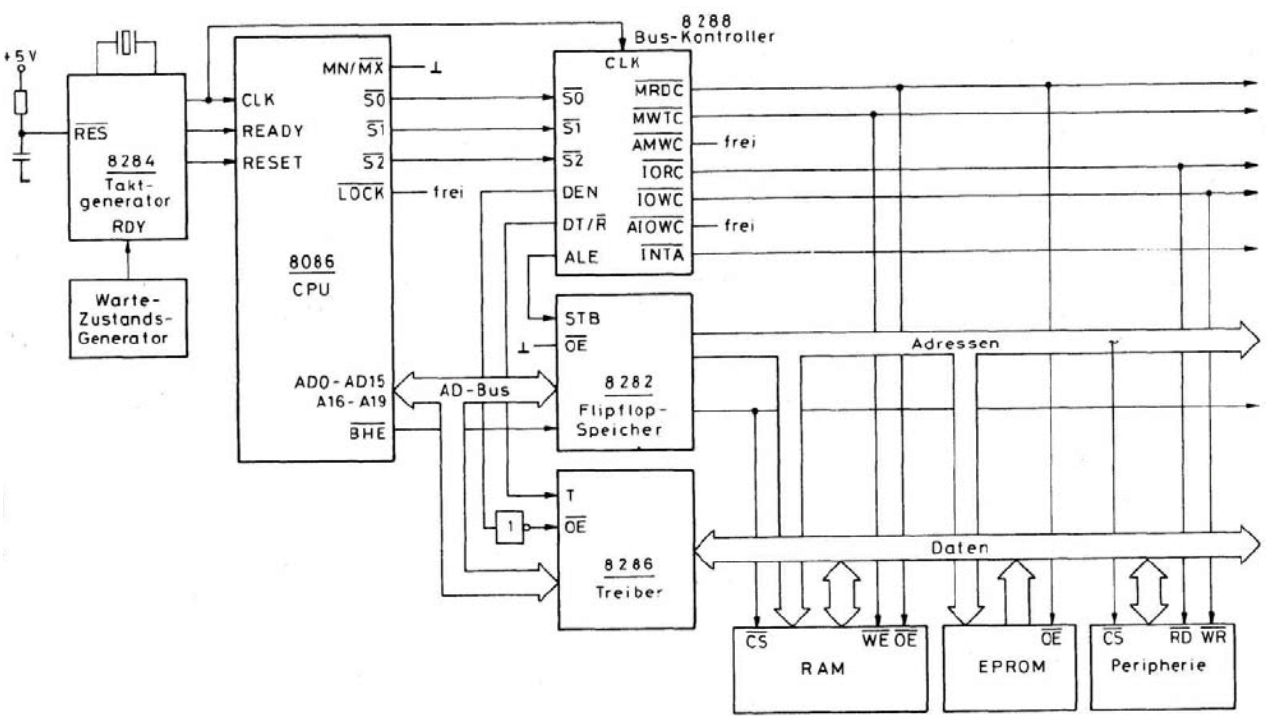
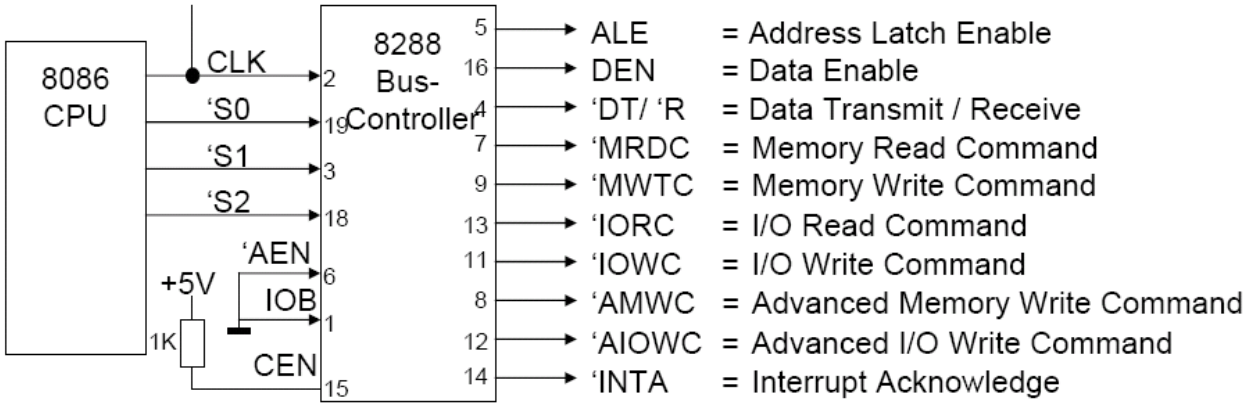


Bild: Maximalsystem RAM, EPROM, Peripherie und Buscontroller



$\overline{S2}$	$\overline{S1}$	$\overline{S0}$	
0	0	0	Interrupt-Bestätigung
0	0	1	Porteingabe
0	1	0	Portausgabe
0	1	1	Halt
1	0	0	Befehl holen
1	0	1	Speicher lesen
1	1	0	Speicher schreiben
1	1	1	Passiv, kein Buszyklus

IOB = Input-Output-Bus-Mode
L: Systembusmode
H: IO-Busmode

AEN = Address Enable (Systembus Mode)
H: Ausgabeleitungen in Tristate
H→L: IO-Busmode: Wirkungslos

CEN = Command Enable
H: Baustein-Freigabe
L: Ausgabeleitungen in Tristate

Bild: Der Buscontroller 8288

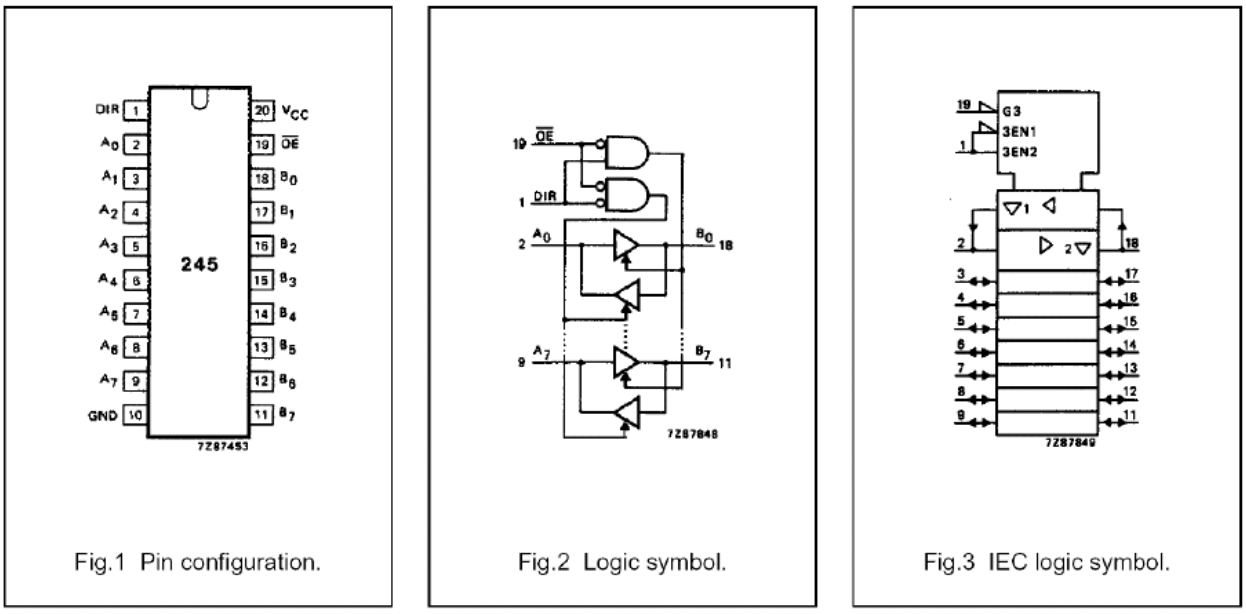


Fig.1 Pin configuration. Fig.2 Logic symbol. Fig.3 IEC logic symbol.

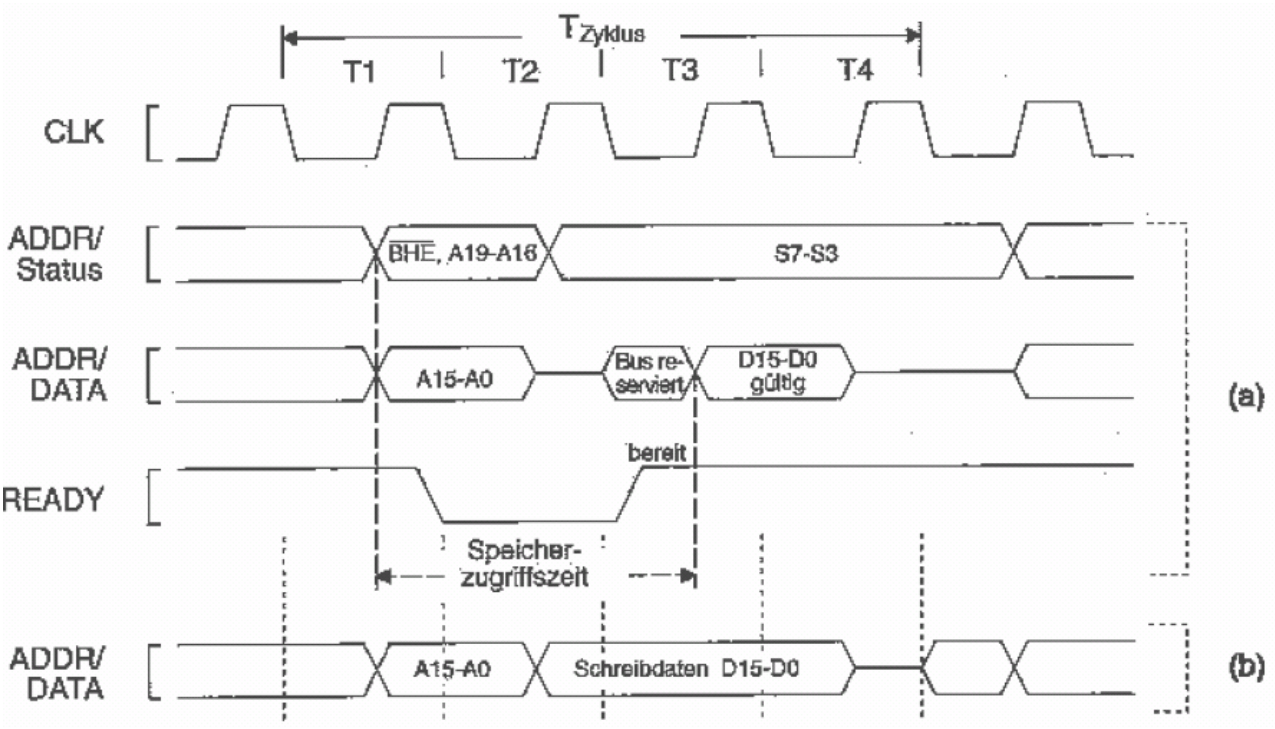
Funktionstabelle:

\OE	DIR	Operation
L	L	B → A
L	H	A → B
H	x	Isolation (Tri-State)

x = Don‘t Care

Quelle: Philips Semiconductors

Bild: Treiber 74245 (Octal Bus Transceiver)



(a) = Lesen, (b) = Schreiben

Bild: Lesen und Schreiben beim i8086 System

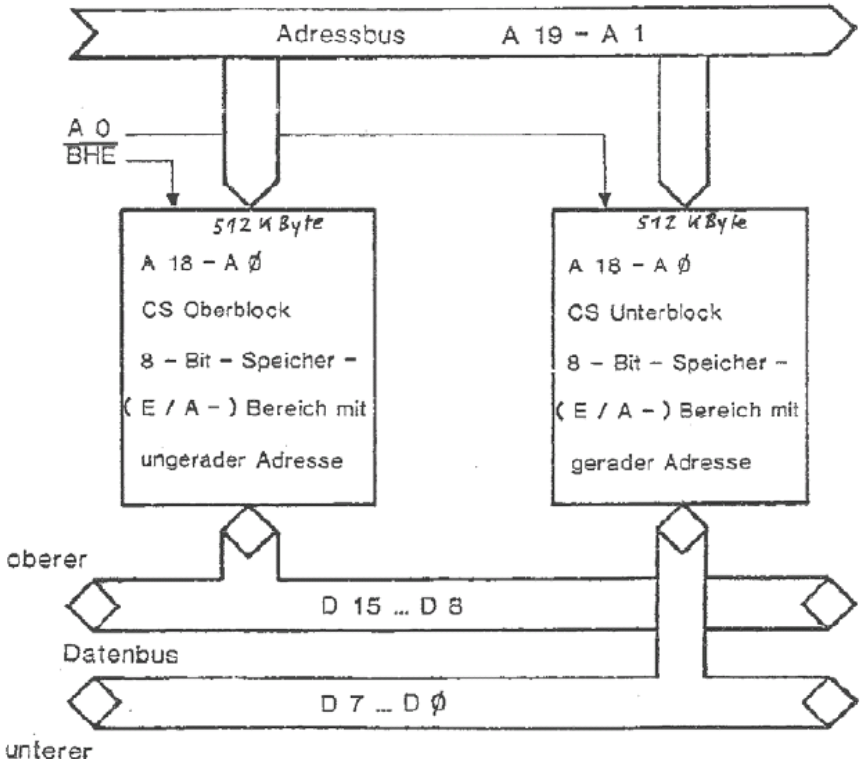


Bild: Das „Byte High Enable“-Signal (BHE)

12. MEMORY MAPS, ADRESSRÄUME U. ADRESSDEKODIERUNG

Ziele:

Memory Map (Speicher-Adressraum, I/O-Adressraum, Memory Maped I/O)

Speicherbausteine (Architekturen; seriell, parallel, bankorientiert (Flash))

Anschaltung von Speicher an die CPU

Adressdekodierung

Literatur und Weblinks

http://www.bjoern-koester.de/iogrundlagen/index.html

12.1. Der Adressraum der MSP430-Mikrokontrollerfamilie

Die MSP430-Familie nur über einen Adressraum, der auf eine Adressbreite von 16 Bit festgelegt ist. Dieser Adressraum umfasst

- alle Spezialregister der CPU und der weiteren im Chip integrierten peripheren Module,
- den Schreib-Lese-Speicher, der zur Aufnahme des Stapelspeichers (Stack), variabler Daten und Programme verwendet wird,
- den Programmspeicher
- leere Bereiche ohne Speicher
- Interruptvektoren

in seinem 16-Bit-Bereich, also den Adressen zwischen 0 und 0FFFFH bzw. 0 bis 65535 dezimal.

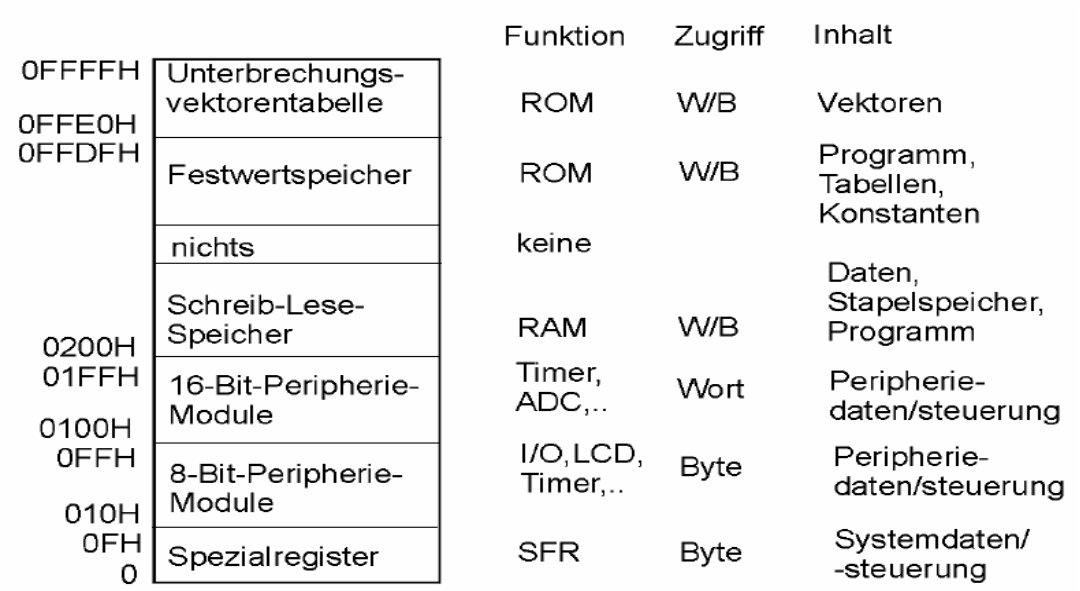


Bild: Speicherbelegungsplan der Mikrocontrollerfamilie TI MSP430x44x (memory map)

Damit sind natürlich dieselben Befehle zur Speicherbearbeitung, für Peripherieaktivitäten und für Ein-/Ausgabeoperationen zuständig, und das mit allen Adressierungsarten.

In Zukunft soll der Speicher durch segmentierte Speicherblöcke erweitert werden. Dazu werden im Statusregister die führenden Adressbits von Codesegmenten (Programmsegmenten) als Codesegmentzeiger (Code Segment Pointer CSP) und getrennt davon ein Teil der Datenadresse als Datenzeiger (Data Pointer DPP) gespeichert.

Memory Address	Description
0FFE0h–0FFFFh	Interrupt Vectors
0FFDFh	End of code space - All device
0F800h	Start of code space - 2K device
0F000h	Start of code space - 4K device
0E000h	Start of code space - 8K device
0D000h	Start of code space - 12K device
0C000h	Start of code space - 16K device
0A000h	Start of code space - 24K device
08000h	Start of code space - 32K device
04000h	Start of code space - 48K device
01100h	Start of code space - 60K device
010FFh	End of Information Memory: Flash devices except 'F110 and 'F1101
0107Fh	End of Information Memory: 'F110 and 'F1101
01000h	Start of Information Memory: Flash devices only
0FFFh	End of Boot Memory: Flash devices only
0C00h	Start of Boot Memory: Flash devices only
09FFh	End of RAM-2k devices
05FFh	End of RAM-1k devices
03FFh	End of RAM-512 byte devices
02FFh	End of RAM-256 byte devices
027Fh	End of RAM-128 byte devices
0200h	Start of RAM-All devices
01B0h – 01FFh	Unused (All devices)
01A0h – 01AFh	ADC Control ('1xx and '4xx devices) / Unused ('3xx devices)
0160h – 017Fh	Timer A (All devices)
0140h – 015Fh	ADC Conversion ('1xx and '4xx devices) / Unused ('3xx devices)
0130h – 013Fh	Multiplier (All devices)
0110h – 011Fh	ADC ('3xx devices) / Unused ('1xx and '4xx devices)
0100h – 010Fh	Unused (All devices)

00B0h – 00FFh	Unused (All devices)
0090h – 00AFh	LCD (Byte addressed, '4xx devices) / Unused ('1xx and '3xx devices)
0080h – 008Fh	ADC memory control (Byte addressed, '1xx and '4xx devices) / Unused ('3xx devices)
0070h – 007Fh	USART (Byte addressed, All devices)
0060h – 006Fh	Unused (All devices)
0050h – 005Fh	System Clock (Byte addressable, All devices) / Comparator ('1xx and '4xx devices) / Brownout ('4xx devices) / EPROM and crystal buffer ('3xx devices)
0040h – 004Fh	Basic Timer and 8-bit Counter (Byte addressable, '3xx and '4xx devices) / Unused ('1xx devices)
0030h – 003Fh	I/O ports 5 and 6 control (Byte addressable, '1xx and '4xx devices) / LCD (Byte addressable, '3xx devices)
0020h – 002Fh	I/O ports 1 and 2 control (Byte addressable, All devices)
0010h – 001Fh	I/O ports 3 and 4 control (Byte addressable, All devices), I/O port 0 (Byte addressable, '3xx devices)
0006h – 000Fh	Unused (All devices)
0005h	Module Enables 2 (Byte Addressable, all devices)
0004h	Module Enables 1(Byte Addressable, all devices)
0003h	Interrupt Flags 2 (Byte Addressable, all devices)
0002h	Interrupt Flags 1 (Byte Addressable, all devices)
0001h	Interrupt Enables 2 (Byte Addressable, all devices)
0000h	Interrupt Enables 1 (Byte Addressable, all devices)

Bild: Memory Map (detailliert)

All die physikalisch getrennten Speicherbereiche, sowohl die internen Bereiche mit Festwertspeicher (ROM,..), Schreib-/Lesespeicher (RAM,..), Spezialregistern (SFR) und Peripheriemodulen, als auch -wenn vorhanden- die externen Bereiche sind in einem einzigen Adressraum positioniert (mapped).

Für alle Prozessoren mit einer maximalen Größe des Adressraumes von 64 kByte (65.536 Bytes), gilt das “kleine Speichermodell” (small memory model).

Dieses Speichermodell benutzt einen eindimensionalen linearen Adressraum mit Adressen von Null (00000h) bis 65.535 (0FFFFh).

12.2. Speicher im Mikrocontroller

Der Speicher der MSP430-Familie ist vollständig auf dem Silizium-Chip enthalten.

Auf den gesamten Speicher wird über zwei 16-Bit-Busse zugegriffen:

- **Speicher-Adress-Bus (Memory Address Bus MAB)** zur Bereitstellung der Speicheradresse durch die CPU
- **Speicher-Daten-Bus (Memory Data Bus MDB)** zum Lesen (Speicher => CPU) und Schreiben (CPU => Speicher)

Der Speicher ist technologisch einzuteilen in die beiden Arten

- **Festwertspeicher:** ein Speicher, dessen Inhalt durch die CPU nicht beliebig geändert werden kann (ROM, OTPROM, EPROM, EEPROM,..)
- **Schreib-Lese-Speicher**: ein Speicher, dessen Inhalt jederzeit geändert werden kann und der bei Abschaltung der Versorgungsspannung seinen Wert verliert (Random Access Memory RAM)

Festwertspeicher kann bei der Definition eines Familienmitgliedes entweder benutzerprogrammierbar (One Time Programmable Read-Only Memory OTPROM, Ereasable Programmable Read-Only Memory EPROM) oder bei der Fertigung festgelegt (Read-Only Memory ROM) werden. Der Zugriff auf gewöhnlichen externen Speicher ist für spätere Familien mitglieder vorgesehen.Programmseitig ist der Speicher jedoch einzuteilen in Programmspeicher und Datenspeicher.

Programmspeicher

Der Zugriff auf den Programmspeicher ist beim Zugriff auf Anweisungen immer Word-orientiert, d. h. die CPU greift nur auf gerade Adressen wortweise mit jeweils 16 Bit zu. Daten können wortweise (16 Bit an gerader Adresse) oder byteweise (8 Bit an beliebiger Adresse) gelesen oder geschrieben werden.

Jeder Zugriff auf den Speicher benutzt den 16-Bit-Datenbus und soviele der niederwertigen Adressleitungen wie für die Speicherunterscheidung gerade benötigt werden. Speicherblöcke werden erst bei Zugriff automatisch selektiert und durch Modulaktivierungssignale (Module Enable Signals) aktiviert, um den Stromverbrauch des MSP430 möglichst klein zu halten.

Datenspeicher

Der gesamte Befehlssatz arbeitet wort- oder byteorientiert. Alle Zugriffe auf den Stapelspeicher und den Programmzähler haben wortorientiert mit geraden Adressen zu erfolgen.

Spezialregister

Der Betrieb der verschiedenen Module der MSP430-Familie wird im wesentlichen durch Informationen gesteuert, die in Spezialregistern (SFR) gespeichert sind. Die verschiedenen Bits in den SFR erlauben Unterbrechungen (Interrupts), liefern dem Programm Informationen über die Ursache einer Unterbrechung und steuern den Betrieb peripherer Module.

Angehaltene Peripheriegeräte stellen ihren Betrieb ein, um den Stromverbrauch zu minimieren. Dabei bleiben alle in den Registern des Moduls enthaltenen Daten erhalten. Die SFR’s sind in zwei Adressbereichen untergebracht, einem für Byte-Register und einem für Word-Register.

12.3. Ankopplung von Peripheriemodulen

12.3.1. Peripheriemodule - Übersicht

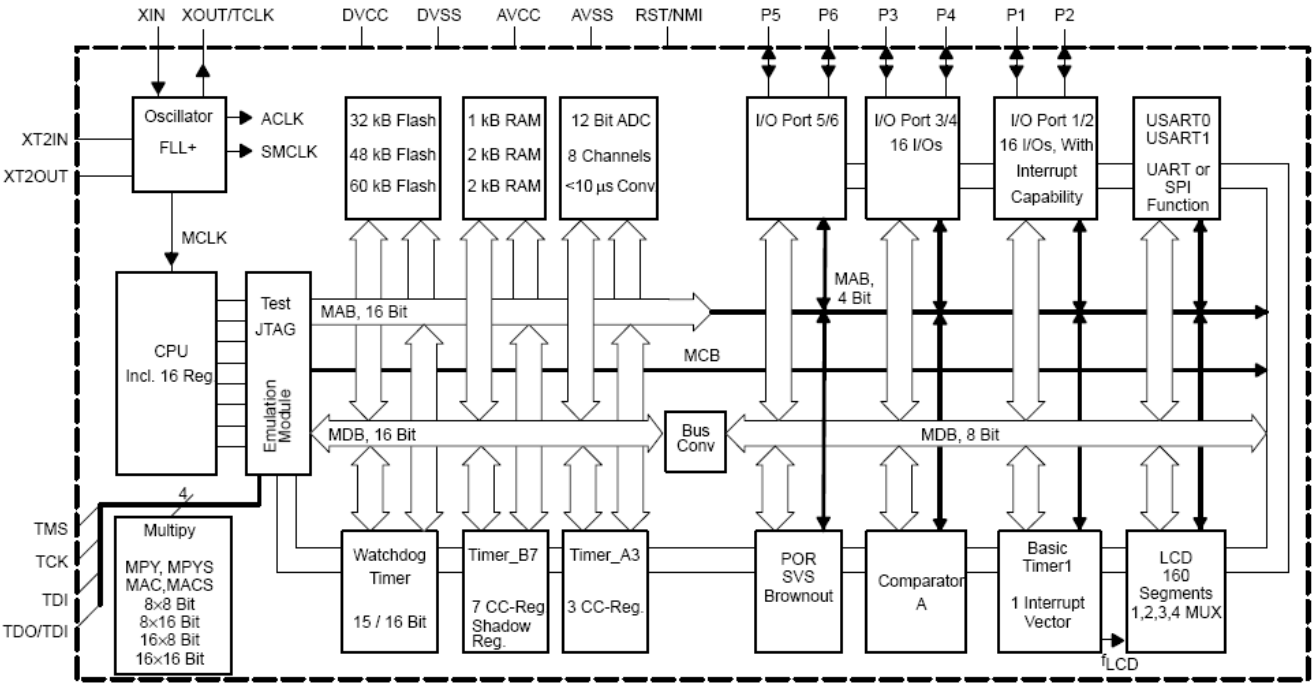


Bild: MSP430x44x functional block diagrams

430x33x	430x32x	430x31x	430x11x	
X	X	X	X	Power on Reset
X	X	X		Oscillator FLL System Clock
X	X	X		I/O Port P0.x
X			X	I/O Port P1.x & P2.x
X				I/O Port P3.x
X				I/O Port P4.x
X				Hardware Multiplier
	X			ADC 12+2-bit
X	X	X	X	Watchdog Timer (15/16 bit)
X			X	Timer_A
X				USART UART or SPI Function
X	X	X		8-bit Timer/Counter
X	X	X		Timer/Port Applications: ADC, Timer...
X	X	X		Basic Timer 1
X	X	X		LCD

Bild: MSP 430 Module Overview

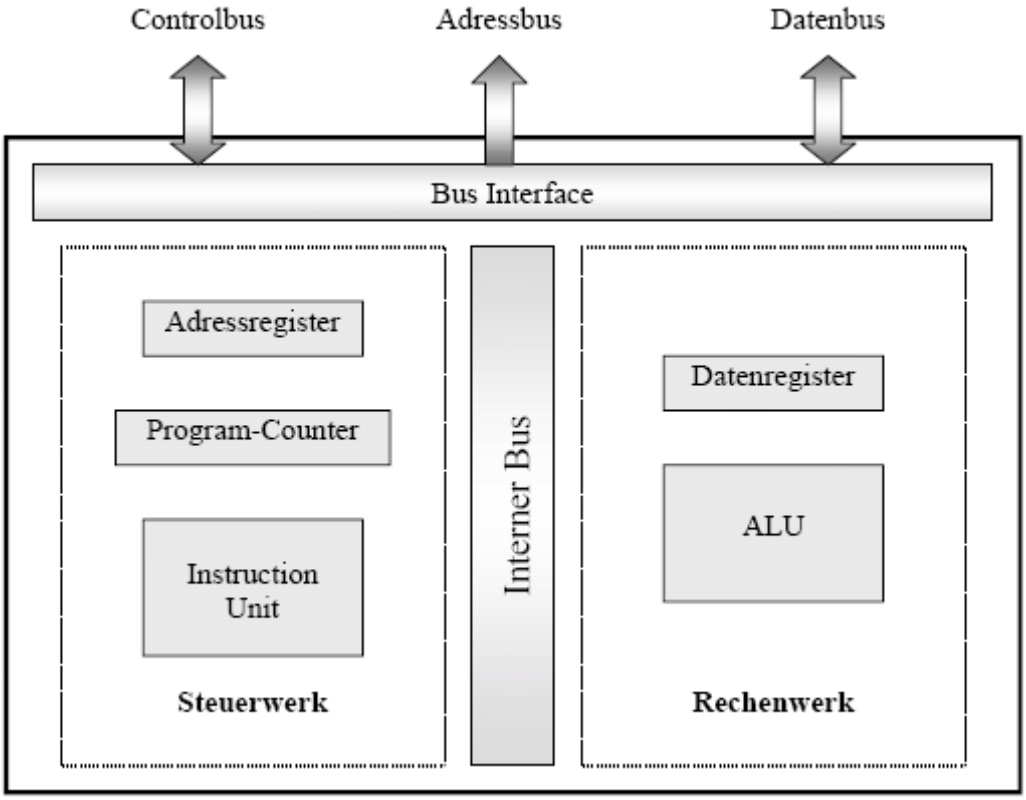


Bild: Aufbaustruktur einer CPU

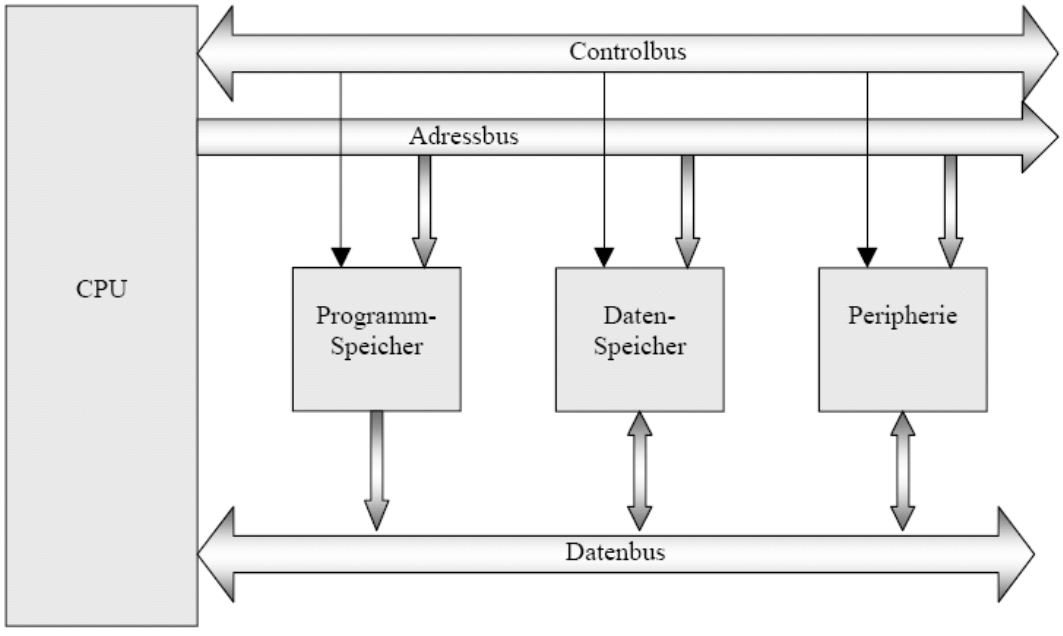


Bild: Adress-, Daten- und Controlbus

Man unterscheidet zwischen

- Memory-Adressraum
- I/O-Adressraum
- Memory Mapped-IO-Adressraum

12.3.2. Speicherbausteine

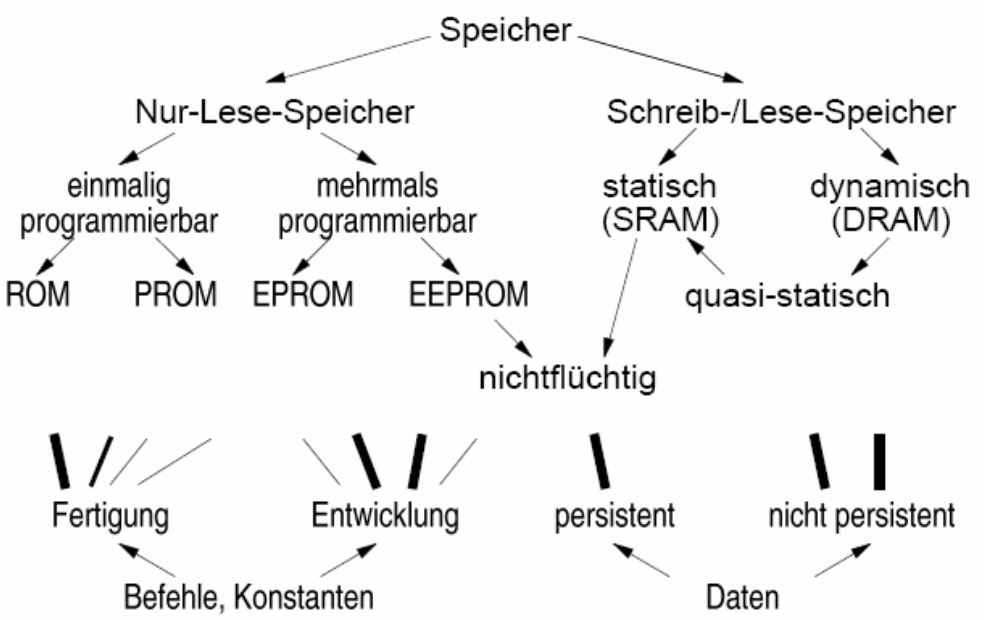


Bild: Speicherbausteine (Übersicht)

10.1.2.1 Terminologie

Speicherzelle:
kleinste Informationseinheit

physikalische Adresse:
Ortsangabe für eine Speicherzelle (im Unterschied zur rechnererzeugten logischen Adresse eines Operanden

Zugriffszeit:
Zeit um ein Datum aus einer Speicherzelle zu lesen

Random Access Memory (RAM):
Die Zugriffszeit für eine Speicherzelle ist unabhängig von ihrer Lage im Speicher

füchtiger Speicher:
bei Wegnahme der Energiequelle geht der Inhalt verloren

Read Only Memory (ROM):

- Speicherinhalt ist unter normalen Einsatzbedingungen nicht änderbar
  -> ROM ist nichtflüchtig

- Halbleitertechnologie ist sehr gut geeignet zur Produktion dieser Speicherchips;

- oft eingesetzt zur Speicherung von Systemsoftware bei Mikroprozessoren bzw. Applikationssoftware bei Mikrocontrollern

- Verschiedene Typen von ROMs

- maskenprogrammiert (seitens des Herstellers während der Produktion)
- PROM (Programmable Read Only Memory)–Kann einmalig programmiert werden
- EPROM (Eraseable Prgrammable ROM)–Kann mittels UV-Strahlen

gelöscht werden. EPROM (Erasable Programmable ROM): Kann vom Anwender mehrmals programmiert (spezielle Programmiergeräte) und mit UV-Licht wieder gelöscht werden.

- EEPROM (Elektrisch löschbarer PROM)•Flash-ROM (Elektrisch löschbarer ROM), d. h. alles gleichzeitig löschbar (bulk erase) oder sektorweise (sector erase). EEPROM (Elecrical Erasable Programmable ROM): Kann während dem Betrieb Zellenweise elektrisch gelöscht und neu programmiert werden. Kleine Speicherkapazität (einige kByte). EEPROM werden z.B. für Produktionsdaten wie Serienummer usw. verwendet.

- OTP (One Time Programmable): Aufgebaut wie ein EPROM, jedoch ohne Fenster. Kann vom Anwender genau 1 mal programmiert werden.

- ROM (Read Only Memory): Wird beim Chiphersteller maskenprogrammiert.

- FLASH: Kann während dem Betrieb page-weise elektrisch gelöscht werden. Sehr grosse Speicherkapazität (einige MByte). Flash-Bausteine werden vor allem für Programmcode verwendet.

Flüchtige Speicher

Der Inhalt dieser Speicher geht nach dem Ausschalten der Versorgungsspannung verloren. Diese Speicher werden allgemein als RAM (Random Access Memory) bezeichnet. Man unterscheidet folgende Kategorien:

Statischer Speicher (SRAM):

SRAM (Static RAM): Sind aufwändiger aufgebaut, benötigen aber keinen Auffrisch Zyklus. SRAM's sind schneller als DRAM's, brauchen weniger Strom, sind aber teurer.
Speicherinhalt bleibt bestehen, sofern
- er nicht überschrieben wird,
- bei flüchtigem Speicher die Energiequelle nicht entfernt wird.

Dynamischer Speicher (DRAM):

DRAM (Dynamic RAM): Diese halten die Information in einer Kapazität und müssen periodisch aufgefrischt werden.

- Speichert Informationen nur während kurzer Zeiten
- Speicherinhalt muss periodisch aufgefrischt werden

10.1.2.2 Statischer Speicher

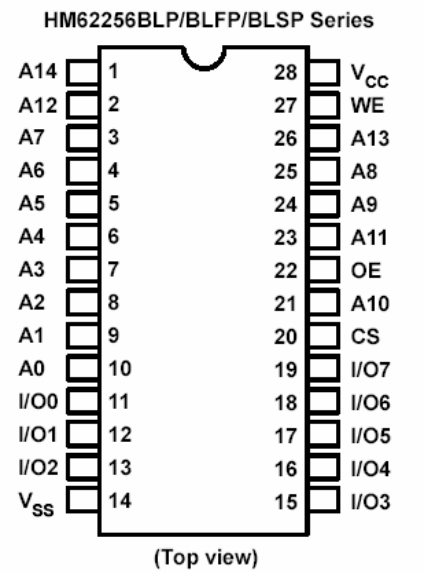
Eigenschaften:

- einfacherer Entwurf
- größere Zuverlässigkeit
- höhere Kosten

=> Einsatz als kleiner oder mittelgroßer Speicher

Beispiel:

statischer Speicher Typ 62256, Organisation: 32K x 8 (256 kBit), (z.B. Hitachi HM62256B)



Pin Description

Pin Name	Function
A0 to A14	Address input
I/O0 to I/O7	Data input/output
CS	Chip select
WE	Write enable
OE	Output enable
V_{cc}	Power supply
V_{ss}	Ground
NC	No connection

Bild: Kenngrößen eines Speicherbausteines

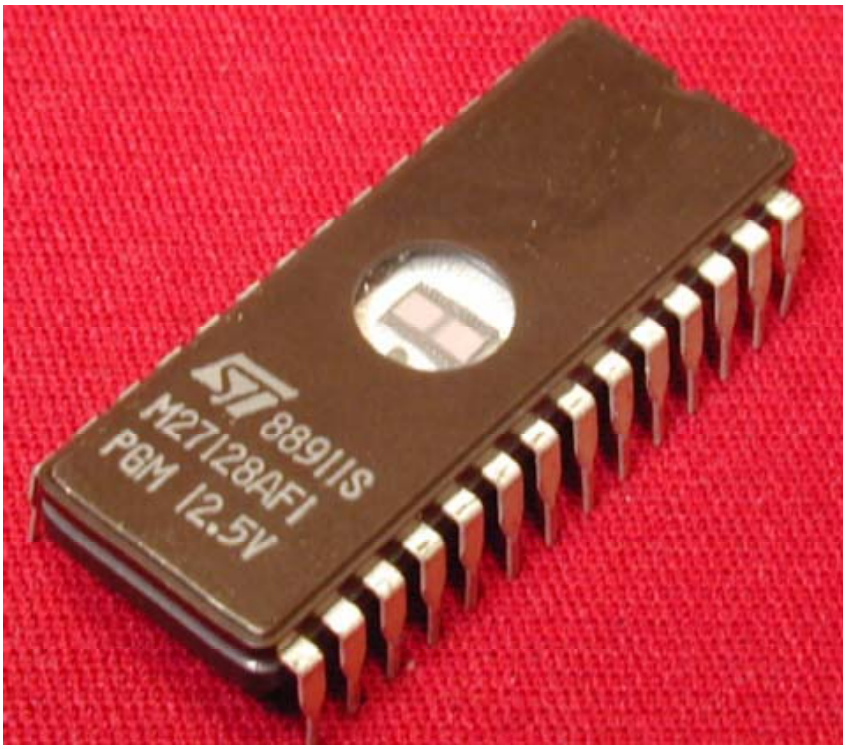
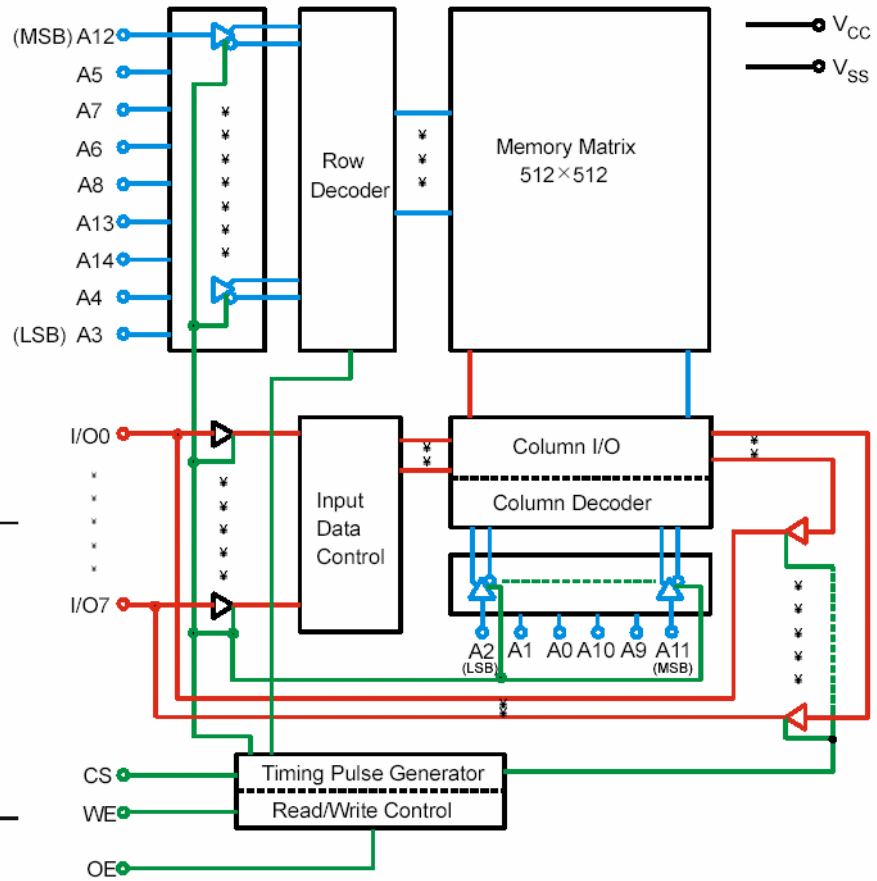


Bild: EEPROM, 128kBit (16k x 8 = 16kByte)

Interne Anordnung:



Signal	Function
A0 to A14	Address input
I/O0 to I/O7	Data input/output
CS	Chip select
WE	Write enable
OE	Output enable
V_{CC}	Power supply
V_{SS}	Ground
NC	No connection

Bild: Interne Aufbaustruktur des Speicherbausteines

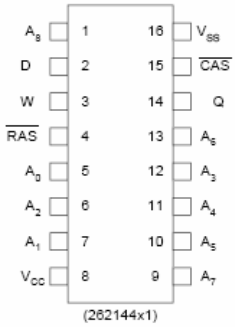
10.1.2.3 Dynamischer Speicher

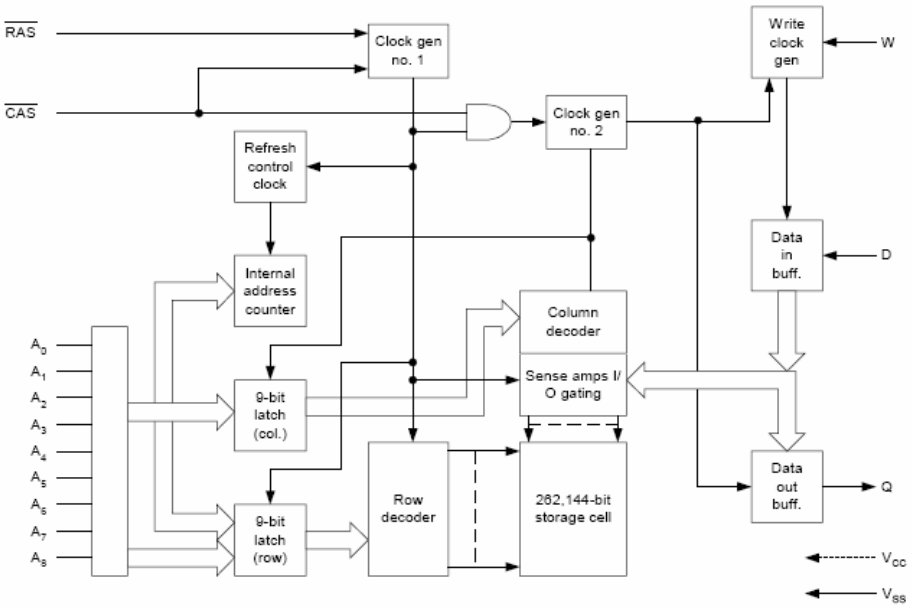
Eigenschaften:

- Höhere Dichte
- geringere Kosten
- komplexerer Entwurf

→ Einsatz als Massenspeicher

Interne Anordnung:





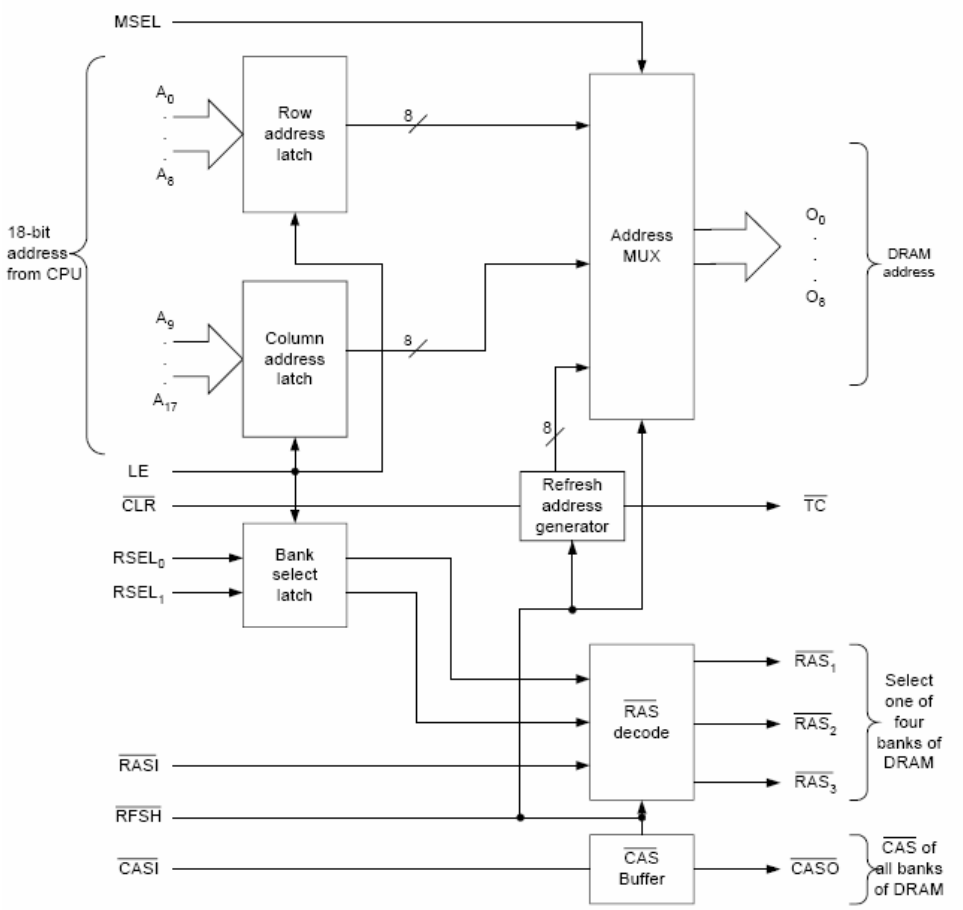


Bild: Kontrolllogik für einen 256Kdynamischen Speicher

Nachteile des dynamischen Speichers:

- benötigt mehr Strom durch zyklische Auffrischungen
- anfällig für Verunreinigung durch Alphapartikel (Heliumkerne); Abhilfe durch
  - sorgfältige Qualitätskontrolle-
  - Anwendung fehlerkorrigierender Codes (ECC RAM)

Anwendungsfelder verschiedener RAM-Typen:

dynamischer RAM: für
- Workstations, Mainframes, PCs

statischer RAM: für
- Single-board Mikrocontroller,
- High-speed Computer
- Cache Speicher

12.4. Adressraumaufteilung und -dekodierung

Ein vereinfachtes Schema mit Adress-Decoder, Speicher und Peripherie könnte wie folgt aussehen (Beispiel 32k ROM, 8k RAM, A/D-Wandler mit 1 Byte):

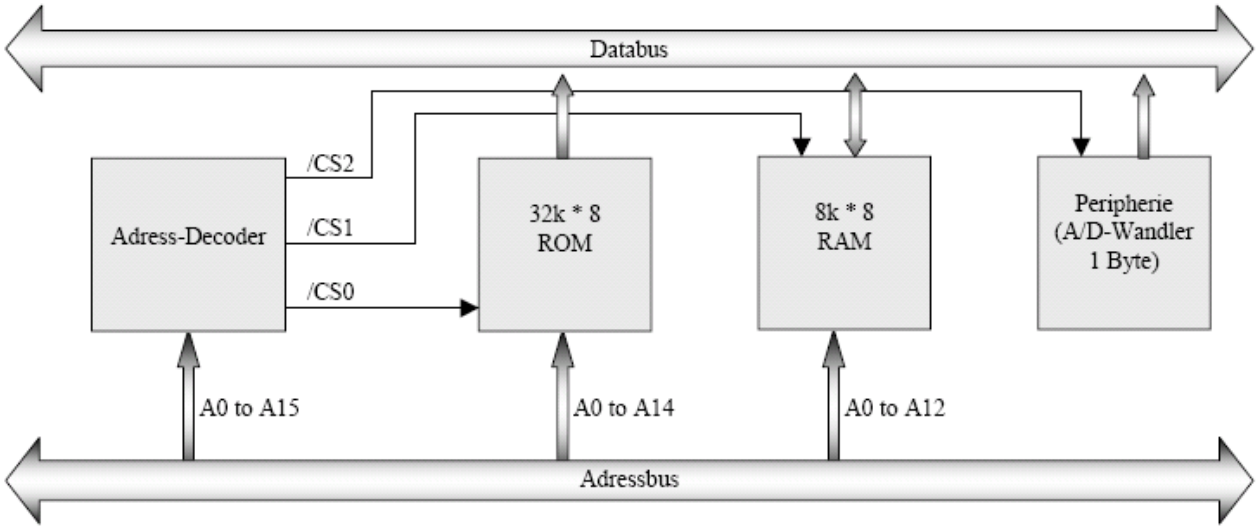


Bild: Schema mit Adress-Decoder, Speicher und Peripherie (Steuerleitungen /RD und /WR nicht aufgeführt)

Für programmierbare Adress-Decoder könnte ein Schema wie folgt aussehen:

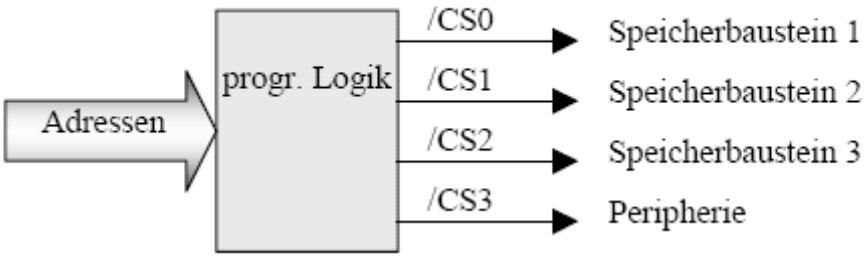


Bild:: Programmierbarer Adress-Decoder

Für die Programmierung werden dann logische Gleichungen verwendet.

Beispiel:
/CS0 = /A13 * A14 * A15

selektiert einen 8k-Block im Adressbereich 0xC000 bis 0xDFFF

12.4.1. Adressdekodierung bedeutet eindeutige Auswahl

Adress-Decoder

Die Aufgabe des Adress-Decoders ist es, einzelne Bausteine oder Baugruppen zu selektieren. Sollen z.B. mehrere RAM's am selben Adressbus betrieben werden, so stellt sich das Problem, dass die einzelnen Bausteine nicht nur durch die Adressleitungen selektiert werden können, sondern ein zusätzliches Selektierungs-Signal (Chip Select, CS) benötigen. Dieses CS-Signal wird vom Adress-Decoder geliefert, indem er die höchstwertigen Adressleitungen logisch verknüpft und daraus die verschiedenen CS-Signale generiert.

Einfache Adress-Decoder werden mit standard "1 of X Decodern" generiert. Für komplexere Decoder werden oft programmierbare Logiken eingesetzt. Beispiel für 1 of 8 Decoder ist der 74HCT138:

http://www.standardics.philips.com/products/hc/ Link to Semiconductors

INPUTS						OUTPUTS							
$\overline{E}_1$	$\overline{E}_2$	E_3	A_0	A_1	A_2	$\overline{Y}_0$	$\overline{Y}_1$	$\overline{Y}_2$	$\overline{Y}_3$	$\overline{Y}_4$	$\overline{Y}_5$	$\overline{Y}_6$	$\overline{Y}_7$
H	X	X	X	X	X	H	H	H	H	H	H	H	H
X	H	X	X	X	X	H	H	H	H	H	H	H	H
X	X	L	X	X	X	H	H	H	H	H	H	H	H
L	L	H	L	L	L	L	H	H	H	H	H	H	H
L	L	H	H	L	L	H	L	H	H	H	H	H	H
L	L	H	L	H	L	H	H	L	H	H	H	H	H
L	L	H	H	H	L	H	H	H	L	H	H	H	H
L	L	H	L	L	H	H	H	H	H	L	H	H	H
L	L	H	H	L	H	H	H	H	H	H	L	H	H
L	L	H	L	H	H	H	H	H	H	H	H	L	H
L	L	H	H	H	H	H	H	H	H	H	H	H	L

Notes

1. H = HIGH voltage level
   L = LOW voltage level
   X = don't care

Bild: 74HCT138 Function Table

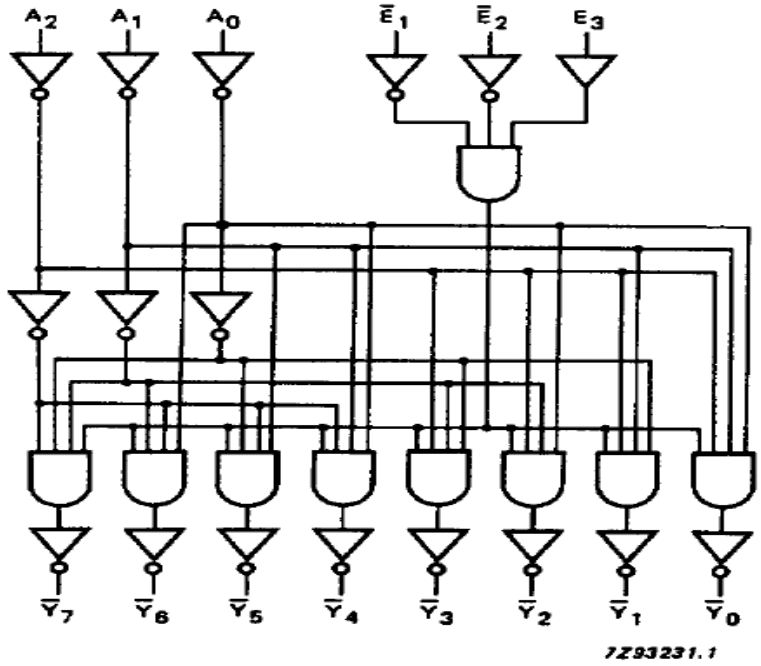


Bild: Logic diagram

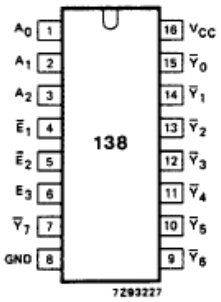


Fig.1 Pin configuration.

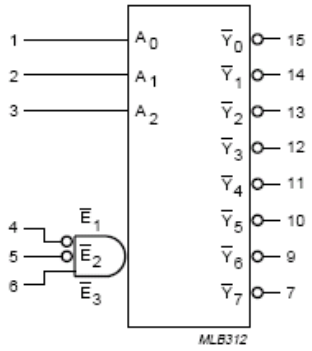


Fig.2 Logic symbol.

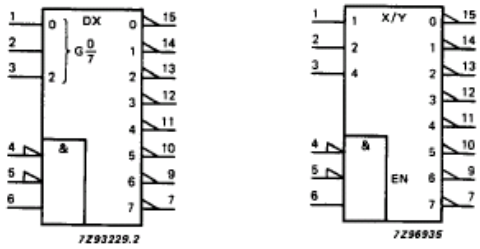


Fig.3 IEC logic symbol.

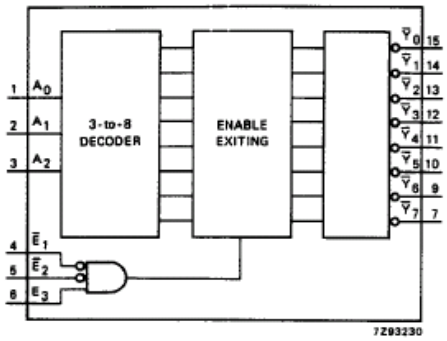
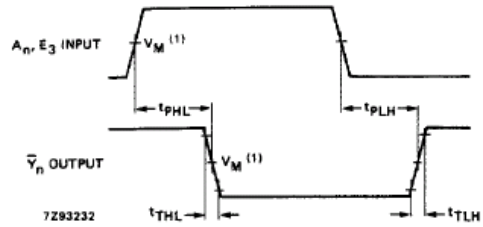
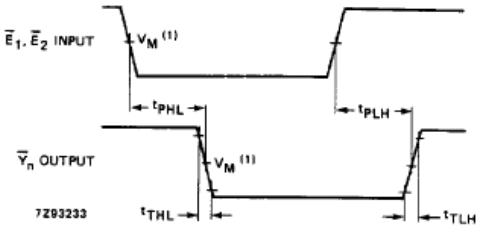


Fig.4 Functional diagram.



(1) HC : V_M = 50%; V_I = GND to V_{CC}.
HCT: V_M = 1.3 V; V_I = GND to 3 V.

Fig.6 Waveforms showing the address input (A_n) and enable input (E_3) to output ($\overline{Y}_n$) propagation delays and the output transition times.



(1) HC : V_M = 50%; V_I = GND to V_{CC}.
HCT: V_M = 1.3 V; V_I = GND to 3 V.

Fig.7 Waveforms showing the enable input ($\overline{E}_n$) to output ($\overline{Y}_n$) propagation delays and the output transition times.

Bild: AC Waveforms

GND = 0 V; $t_r = t_f$ = 6 ns; C_L = 50 pF

SYMBOL	PARAMETER	T_{amb} (°C)							UNIT	TEST CONDITIONS	
		74HC								V_{CC} (V)	WAVEFORMS
		+25			−40 to +85		−40 to +125				
		min.	typ.	max.	min.	max.	min.	max.			
t_{PHL}/t_{PLH}	propagation delay A_n to $\overline{Y}_n$		41 15 12	150 30 26		190 38 33		225 45 38	ns	2.0 4.5 6.0	Fig.6
t_{PHL}/t_{PLH}	propagation delay E_3 to $\overline{Y}_n$		47 17 14	150 30 26		190 38 33		225 45 38	ns	2.0 4.5 6.0	Fig.6
t_{PHL}/t_{PLH}	propagation delay $\overline{E}_n$ to $\overline{Y}_n$		47 17 14	150 30 26		190 38 33		225 45 38	ns	2.0 4.5 6.0	Fig.7
t_{THL}/t_{TLH}	output transition time		19 7 6	75 15 13		95 19 16		110 22 19	ns	2.0 4.5 6.0	Figs 6 and 7

Bild: AC CHARACTERISTICS FOR 74HC (High Speed CMOS)

GND = 0 V; $t_r = t_f$ = 6 ns; C_L = 50 pF

SYMBOL	PARAMETER	T_{amb} (°C)							UNIT	TEST CONDITIONS	
		74HCT								V_{CC} (V)	WAVEFORMS
		+25			–40 to +85		–40 to +125				
		min.	typ.	max.	min.	max.	min.	max.			
t_{PHL}/ t_{PLH}	propagation delay A_n to $\overline{Y}_n$		20	35		44		53	ns	4.5	Fig.6
t_{PHL}/ t_{PLH}	propagation delay E_3 to $\overline{Y}_n$		18	40		50		60	ns	4.5	Fig.6
t_{PHL}/ t_{PLH}	propagation delay $\overline{E}_n$ to $\overline{Y}_n$		19	40		50		60	ns	4.5	Fig.7
t_{THL}/ t_{TLH}	output transition time		7	15		19		22	ns	4.5	Figs 6 and 7

Bild: AC CHARACTERISTICS FOR 74HCT (High Speed CMOS TTL compatible)

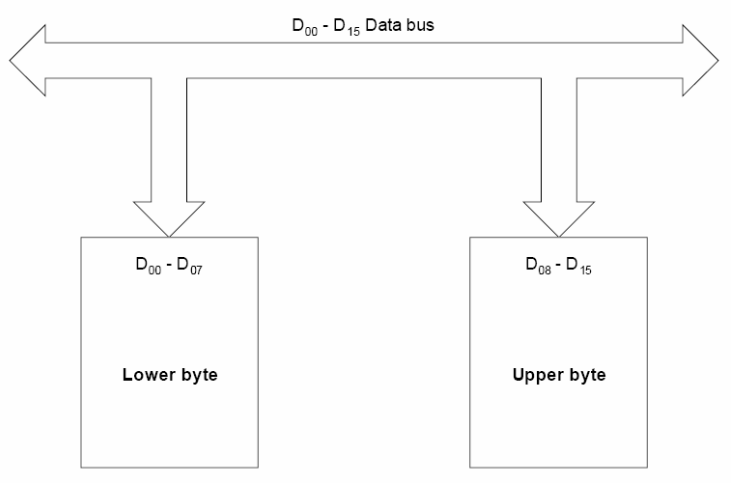


Bild: Beispiel eines byteweise organisierten Speichers

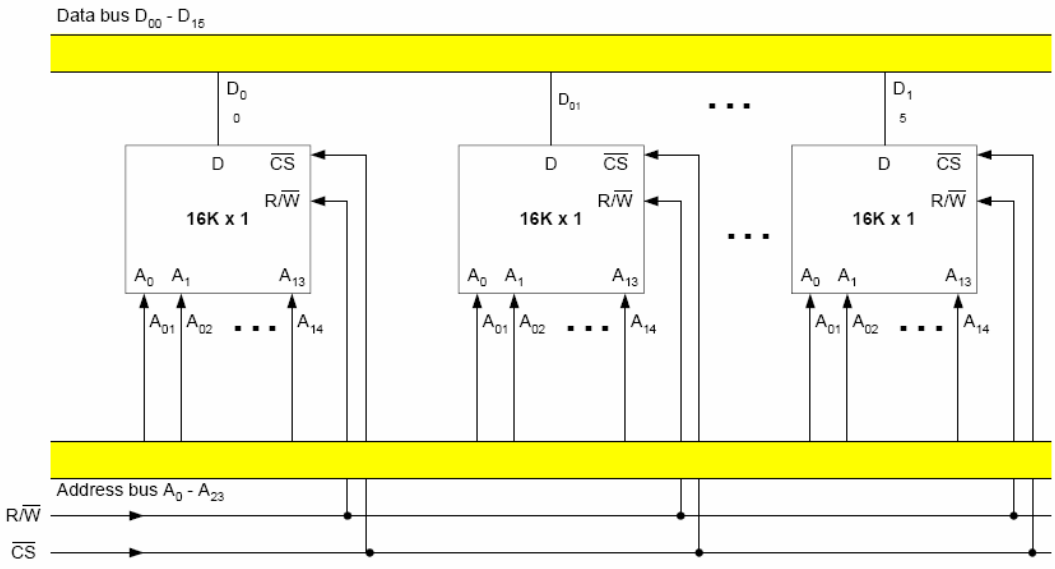


Bild: Beispiel eines bitweise organisierten Speichers

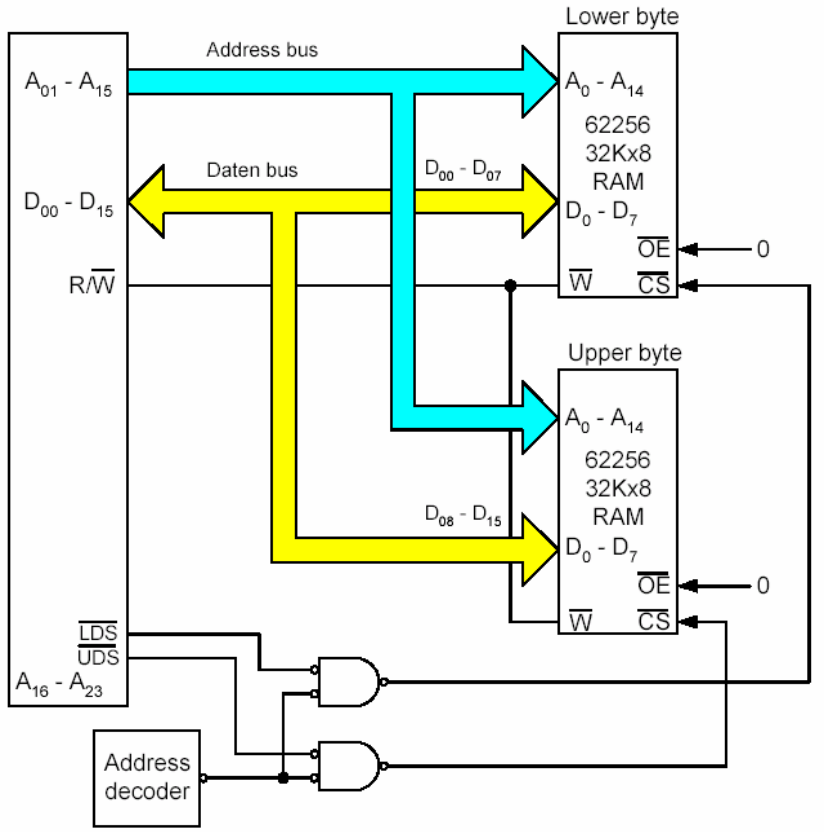


Bild: Verbindung eines SRAM Typ 62256 mit einer 68000 CPU

12.4.2. Datenaustausch der CPU mit Speicher u. der Peripherie

Jedes Speicherwort, jedes Register in einem Coprozessor oder einem peripheren Gerät besitzt eine eindeutige Adresse im System

Die Größe eines Speicherworts (Datenwortbreite) entspricht der Bitanzahl, die in einem Takt zum oder vom Prozessor übertragen werden kann (8,- 16-, 32 Bit-Datenbus)

Übertragungsablauf

- Schritt 1: Prozessor sendet Adresse, Gültigkeitssignal und Lese- oder Schreibsignal
- Schritt 2: Datenübertragung zum Prozessor (Lesen) oder vom Prozessor (Schreiben)

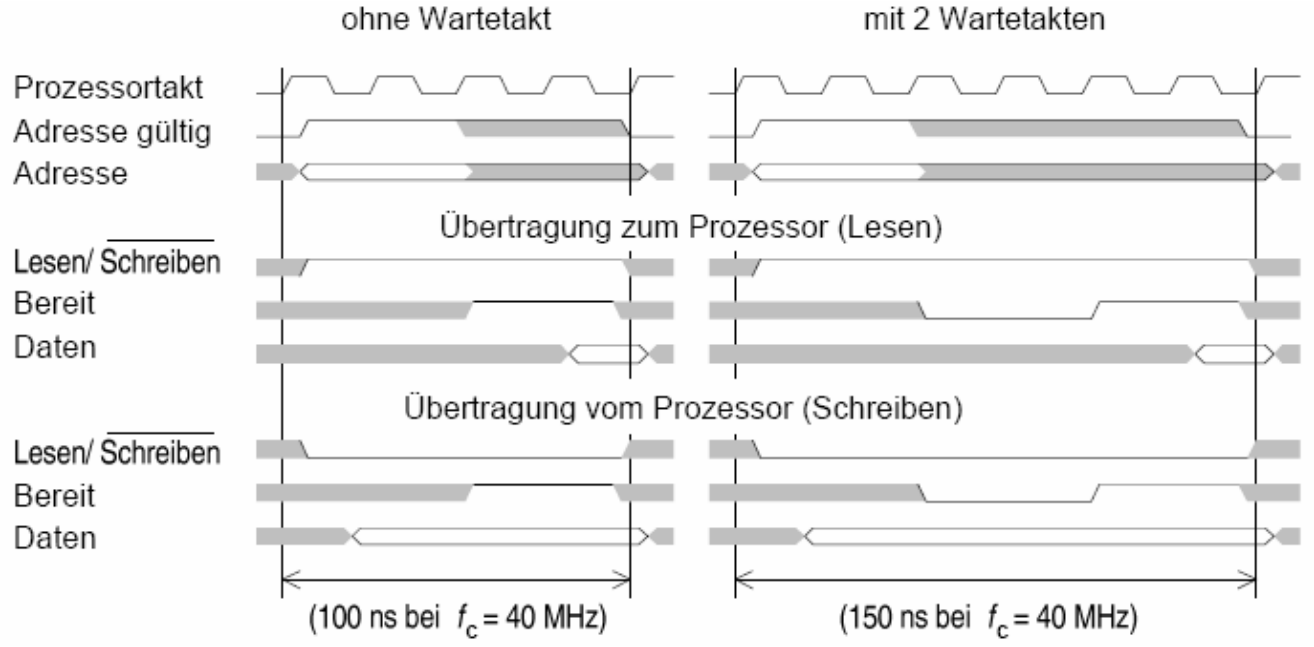


Bild: Kommunikationsablauf CPU und Peripherie

Spezielle Steuersignale für Daten- und Peripherie-Zugriff => versteckte zusätzliche Adressleitung (IORD/-WR, MEMRD/-WR)

Nomenklatur für die Bezeichnung der Steuersignale:

- IORD/-WR (Input Output Read, Write low activ)
- /CS (Chip Select low activ)
- $\overline{CS}$ (Chip Select low activ)

Problem unterschiedliche Geschwindigkeit

Prozessoren sind technologisch bedingt wesentlich schneller als Speicher (Unterschied bis 1:20, Tendenz steigend); periphere Einheiten sind noch langsamer als Speicher.

Lösungsmöglichkeiten:

- Synchronisation über Wartetakte: die Übertragung wird um eine Anzahl von Takten verlängert
  - starre Anzahl, z. B. bei 1 Wartetakt für alle Zugriffe auf die Peripherie
  - programmierbare Anzahl, Einstellung in einem Statusregister entsprechend der Geschwindigkeit der im System verwendeten Speicherschaltkreise
  - Speicher oder Peripherie signalisiert die Bereitschaft über eine spezielle Steuerleitung

- Asynchron (kein gemeinsamer Takt), Handshake (Quittungsbetrieb)
  - Aussenden der Adresse und eines Gültigkeitssignals (AS, address strobe)

<u>Schreiben</u>

- Aussenden der Daten, DS (data strobe) aktiv
- Quittierung der Datenübernahme durch Speicher/ Peripherie

Lesen

- Aussenden Leseanforderung
- Speicher/Peripherie legt Daten auf den Bus und signali siert, daß Daten bereit stehen

Multiplex-Bus

- Adressen und Daten werden zeitlich nacheinander übertragen; gemeinsamer Bus
- hierdurch Einsparung von Prozessoranschlüssen
- Realisierung durch Adresstreiber auf der Baugruppe mit integriertem Latch (LS244, LS245)

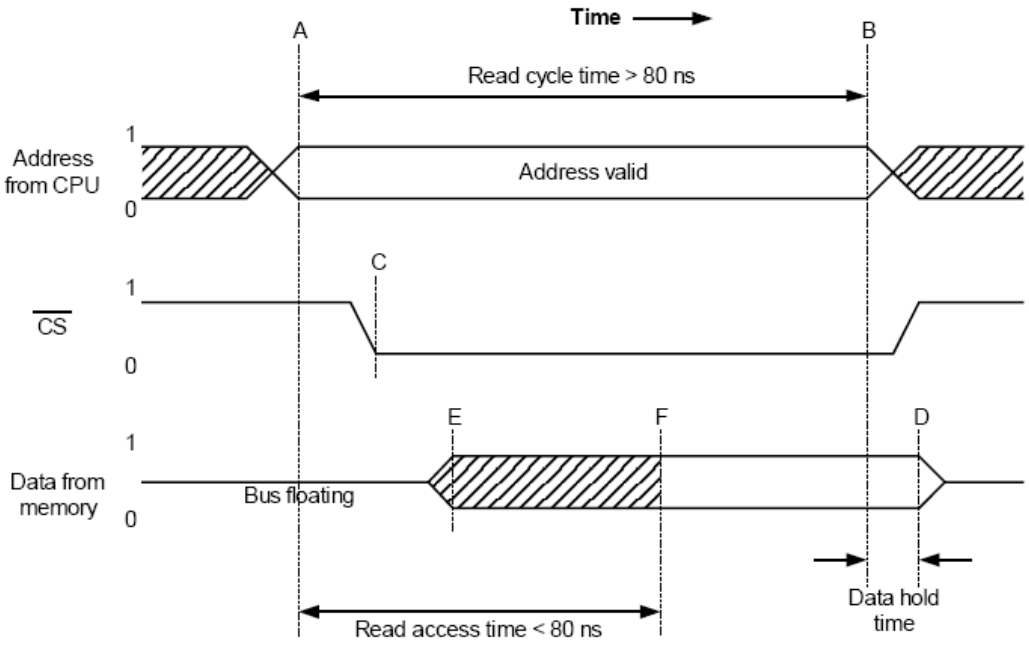


Bild: Zeitdiagramm des Speicher-Lesezyklus

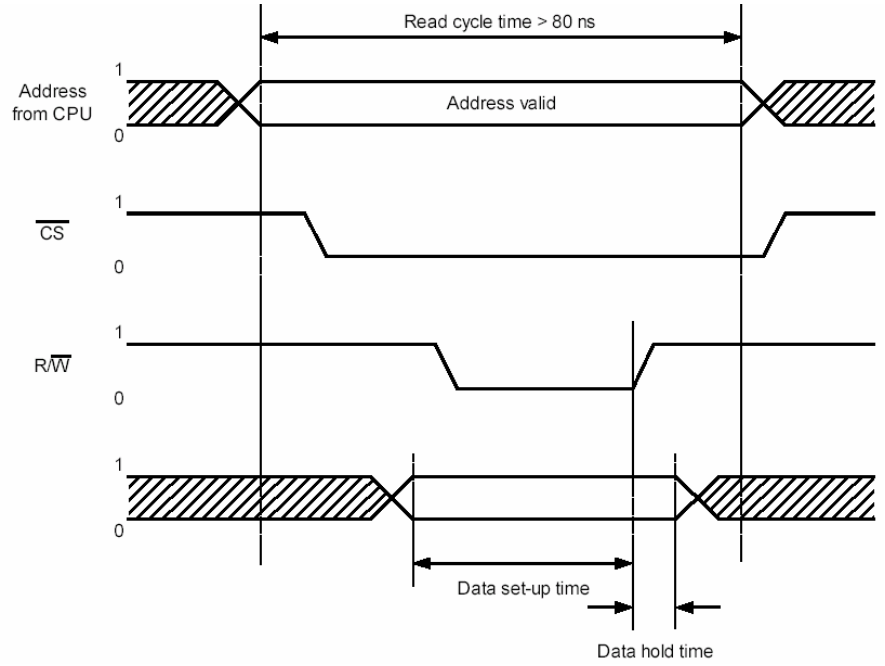


Bild: Zeitdiagramm des Speicher-Schreibzyklus

Der Lesevorgang

Zum Zeitpunkt 1 sendet der Prozessor die Speicheradresse auf den Adressleitungen A0–A15 aus. Zum Zeitpunkt 2 veranlasst der Prozessor den Speicher, den Inhalt der adressierten Speicherzeile auf den Datenbus zu schalten (Zeitpunkt 3). Mit dem L-H.Signalwechsel auf der Steuerleitung MEMR bzw. -IOR übernimmt der Prozessor den Signalzustand vom Datenbus und beendet gleichzeitig den Lesevorgang, so dass der Speicher die Daten vom Bus wegschaltet (Zeitpunkt 5).

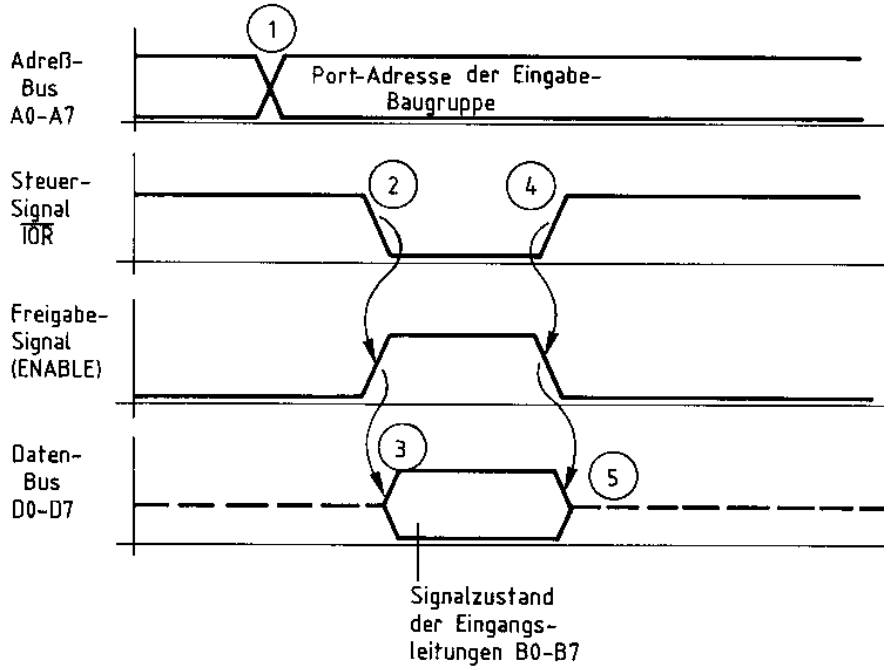


Bild: Signal-Zeitdigram zum Lesen des Speichers

Zum Zeitpunkt 1 sendet der Prozessor die Speicheradresse auf den Adressleitungen A0 - A15 aus. Zum Zeitpunkt 2 veranlasst der Prozessor den Speicher, den Inhalt der adressierten Speicherzeile auf den Datenbus zu schalten (Zeitpunkt 3). Mit dem L-H.Signalwechsel auf der Steuerleitung MEMR bzw. -IOR übernimmt der Prozessor den

Signalzustand vom Datenbus und beendet gleichzeitig den Lesevorgang, so dass der Speicher die Daten vom Bus wegschaltet (Zeitpunkt 5).

Der Schreibvorgang

Zum Zeitpunkt 1 sendet der Prozessor die Speicheradresse auf den Adressleitungen A0–A15aus. Zum Zeitpunkt 2 stellt er den neuen Speicherinhalt am Datenbus bereit.
Mit dem Steuersignal MEMW bzw. -IOW (Zeitpunkt 3) löst er die Datenübernahme in die adressierte Speicherzeile aus. Der Schreibvorgang wird mit H–Signal auf der Steuerleitung MEMW beendet (Zeitpunkt 4), so dass der Prozessor gleichzeitig die Daten vom Bus wegschalten kann (Zeitpunkt 5).

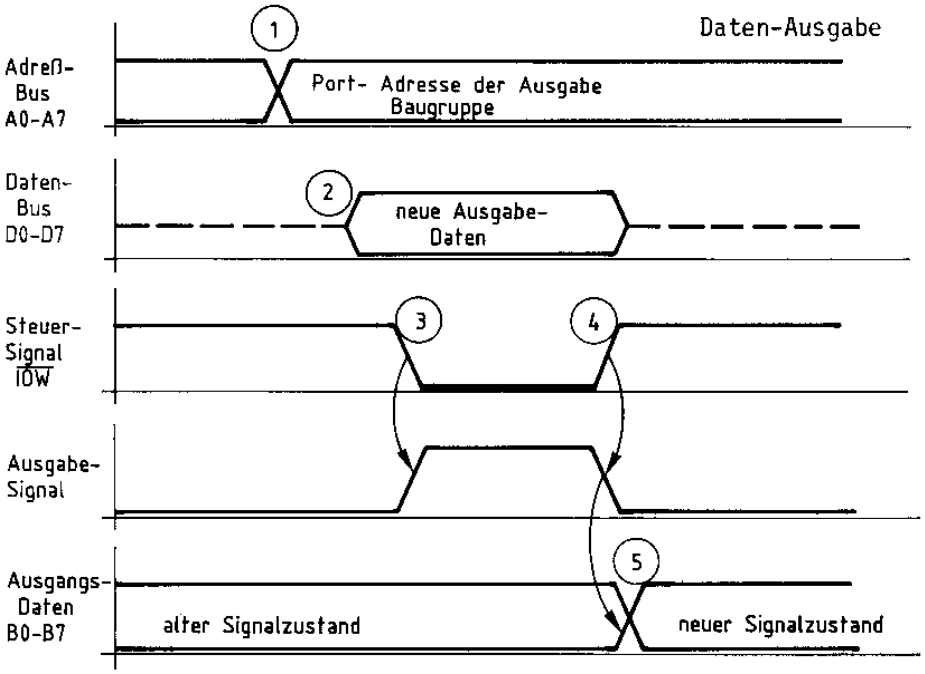


Bild: Signal-Zeitdiagramm zum Schreiben des Speichers

12.4.3. Übung 1 zum Interfacing von Peripheriemodulen

Der Programmspeicher soll auf 128 kB ausgebaut werden. Zur Verfügung stehen 32kB EPROMs, die geeignet verschaltet werden müssen.

Die CPU soll Daten mit dem EPROM austauschen, die Komponenten sind parallel an den Bussen (Adressbus, Datenbus, Steuerbus) angeschlossen.

Das nachstehende Signal-Zeitdiagramm veranschaulicht den Lesevorgang der CPU.

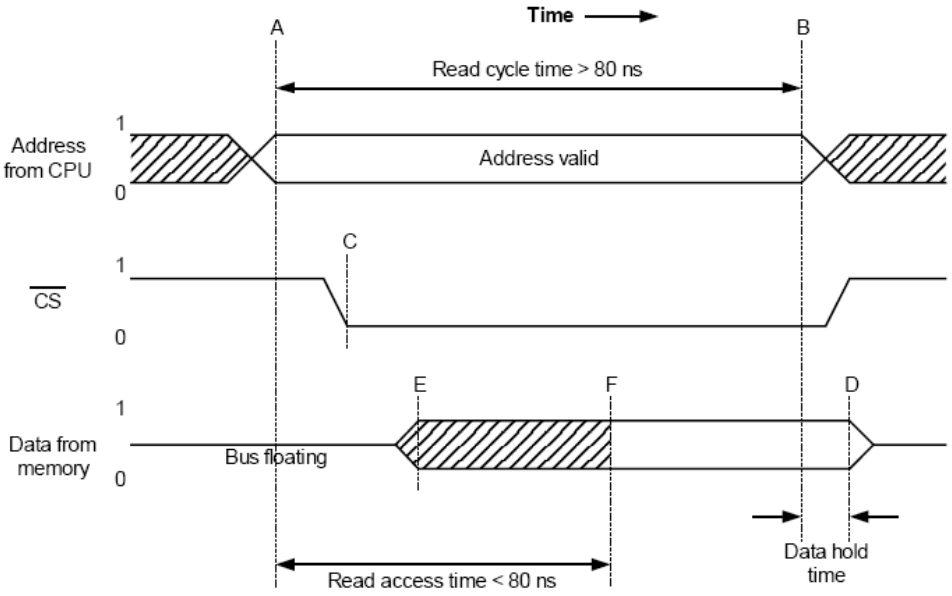


Bild: Zeitdiagramm des CPU-Lesezyklus

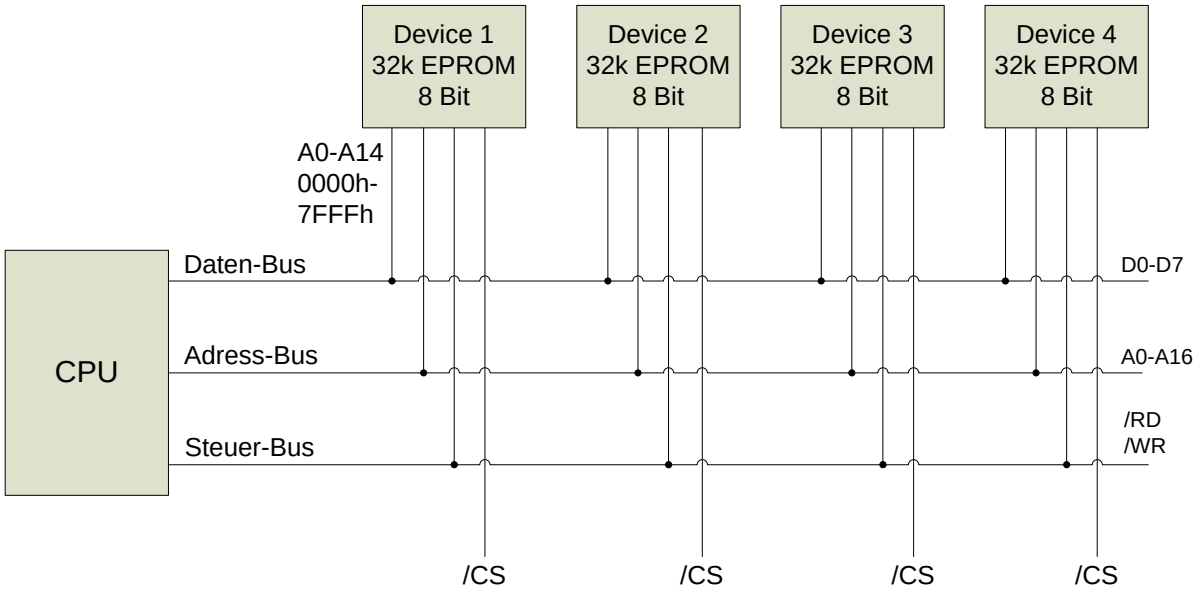


Bild: Blockschaltbild zur Anbindung externer Speicherbausteine

Aufgabenstellung:

a) Geben Sie den Adressraum des Programmspeichers in Hexadezimaldarstellung an.

Komponente	Adressraum
Device 1	
Device 2	
Device 3	
Device 4	

b) Geben Sie die algebraische logische Gleichung zur Dekodierung des Adressraums für die jeweiligen Chip Select (/CS)-Leitungen der Komponenten 1- 4 an.

Device 1:

Device 2:

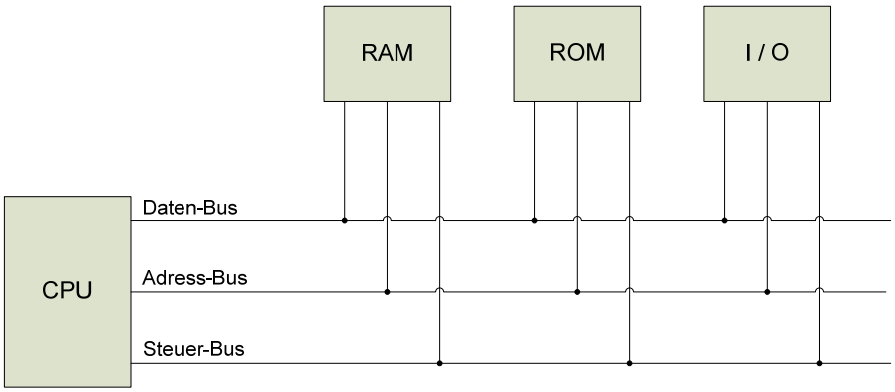
Device 3:

Device 4:

c) Entwerfen Sie eine Digitalschaltung, die sicherstellt, das jeweils nur eine Komponente angesprochen wird und vervollständigen Sie die obige Schaltung hinsichtlich der /ChipSelect-Signale.

12.4.4. Übung 2 zum Interfacing von Peripheriemodulen

Problem**:** Die CPU möchte Daten mit den einzelnen Komponenten (RAM, ROM, I/O) austauschen, die Komponenten sind jedoch parallel an den Bussen angeschlossen.



a) Durch welche beiden Maßnahmen kann gewährleistet werden, dass jeweils nur eine Komponente angesprochen wird?

1)

2)

b) Geben Sie ein Beispiel für die Ressourceneinstellungen des PC-BIOS für eine Komponente, wie z. B. Kommunikationsschnittstelle Serielle Schnittstelle (COM1) an.

Gehen Sie hierzu auf **Start** -> **Einstellungen** -> **Systemsteuerung** -> **System** -> **Hardware** -> **Geräte-**Manager, dann unter **Anschlüsse (COM und LPT)** auf **Kommunikationsanschluss (COM1)** gehen, unter **Ressourcen** finden Sie die **Ressourceneinstellungen** zur Komponente.

12.4.5. Übung 3: Adressraum-Aufteilung

Zwei Universal Asynchronous Receiver/Transmitter Controller (UART) A und B mit einem Adressraum von jeweils 7 I/O-Adressen sollen an eine CPU angeschlossen werden.

Die Registerstruktur des UART-Bausteines ist wie folgt:

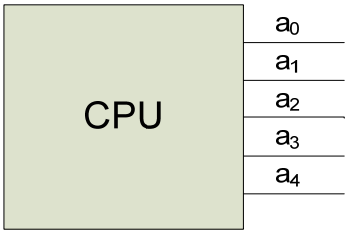
1. Adresse: Daten-Empfangsregister
2. Adresse: Daten-Senderegister.
3. Adresse: Baudrate-Register 1
4. Adresse: Baudrate-Register 2
5. Adresse: Modulations-Register 3
6. Adresse: Steuer-Register
7. Adresse: Interrupt-Register

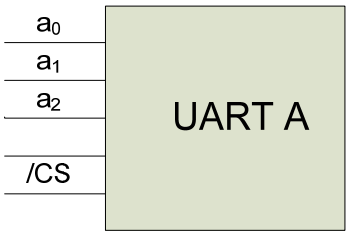
a) Wie groß ist der gesamte von der CPU adressierbare physikalische Adressraum?

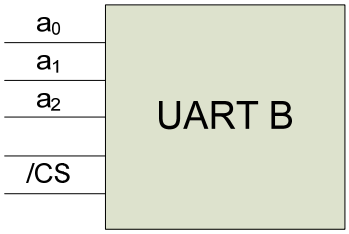
b) Nehmen Sie eine sinnvolle Aufteilung des Adressraums vor:

d) Wie groß ist der Adressraum beider UART Komponenten, wie viele Adressleitungen werden insgesamt benötigt?

e) Verbinden Sie die CPU mit den beiden Komponenten A und B unter Verwendung logischer Gatter so, dass der UART A im Adressraum vor UART B zu liegen kommt.







f) Welche Adresse (in HEX-Darstellung) muss die CPU auf den Adressbus legen, damit das Steuer-Register des UART B angesprochen wird?

12.4.6. Übung 5: Adressdekoder

Konstruieren Sie eine Baugruppe, die aus 2 Adressleitungen 4 /CS-Signale erzeugt.

12.4.7. Übung Speicherbelegungsplan erstellen

Erstellen Sie einen groben Speicherbelegungsplan für Ihren ArbeitsPC und erstellen eine Grafik und beschriften Sie diese.

Arbeitsspeicher-Adressraum, von - bis, in hexadezimaler Schreibweise

I/O-Speicher Adressraum, von - bis, in hexadezimaler Schreibweise

Memory Mapped I/Os (Welche Einheiten nutzen diese Technik)

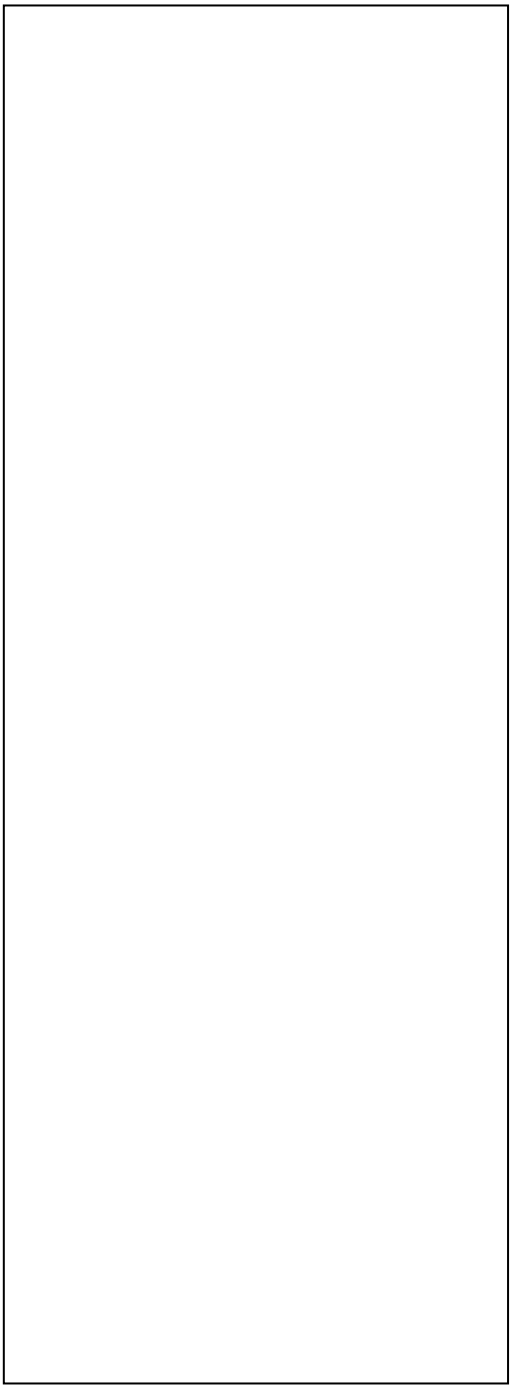


Bild: Hauptspeicher-Adressraum

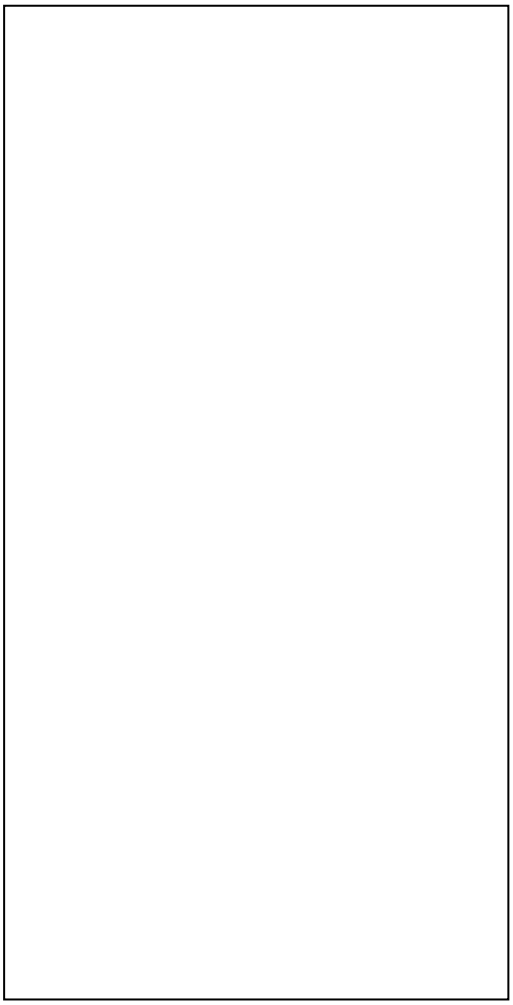


Bild: Eingabe/Ausgabe-Adressraum

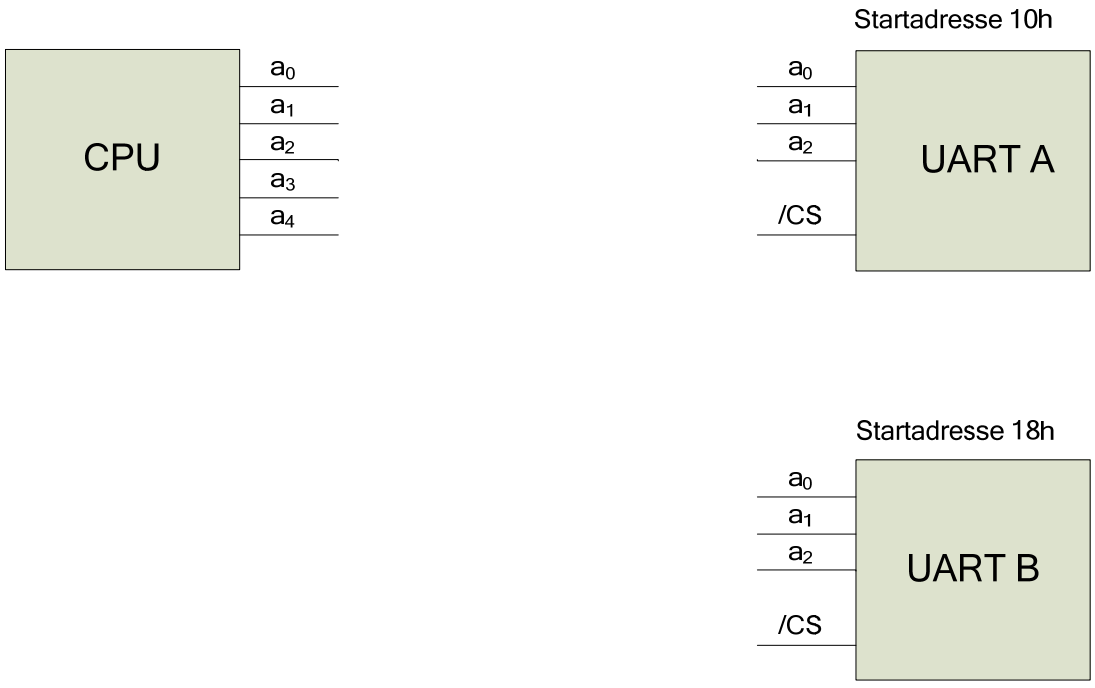
12.4.8. Übung 6: Adressraum-Aufteilung

Zwei Universal Asynchronous Receiver/Transmitter Controller (UART) A und B mit einem Adressraum von jeweils 7 I/O-Adressen sollen an eine CPU angeschlossen werden. Die CPU stellt über einen Adressbus die Leitungen a_0 bis a_4 zur Verfügung.

Die Registerstruktur des UART-Bausteines ist wie folgt:

1. Adresse: Daten-Empfangsregister
2. Adresse: Daten-Senderegister.
3. Adresse: Baudrate-Register 1
4. Adresse: Baudrate-Register 2
5. Adresse: Modulations-Register 3
6. Adresse: Steuer-Register
7. Adresse: Interrupt-Register

Erstellen Sie eine Dekodierschaltung für die Chip-Select Signale (/CS). Beachten Sie dabei, dass der UART-Baustein A ab Hex 10h im Adressraum zu liegen kommen soll und der UART-Baustein B mit der Startadresse 18h beginnt.



12.4.9. Übung 7: Einbettung eines CAN-Controllers

Das Host Controller Interface ist über zwei 8 Bit Datenleitungen und eine 6 Bit Adressleitung mit dem Memory Module verbunden. Dadurch ist ein einfacher Austausch dieser Schnittstelle bei Verwendung anderer Host Controller möglich.

Die Pin-Beschreibung ist in Abb. 2 und das Blockschaltbild in Abb. 3 dargestellt.

PIN-Name	I/O	Funktion
DATAIN	I	8-Bit Datenbus vom Host
DATAOUT	O	8-Bit Datenbus zum Host
ADR	I	6-Bit Adresse vom Host
RD_B	I	Read Signal vom Host
WR_B	I	Write Signal vom Host
CS_B	I	Chip Select (nur Intel)
INT_B	O	Interrupt zum Host
RST_B	I	Reset
TAKT	T	Systemtakt
RXD	I	CAN-Bus-Eingang
TXD	O	CAN-Bus-Ausgang
TST	I	Aktivierung der Testausgänge
SAMPLE	O	Testausgang
SYNCLK	O	Testausgang

Abb. 2: Pin-Beschreibung

EB8540 FEATURES	DESCRIPTION
Product Type	High performance PowerPC embedded computer core: • Processor: Motorola MPC8540 Integrated Processor (Power QUICC III) • Memory: DDR-SDRAM, SRAM, FLASH, EEPROM • Multiple system and communications I/Os • Form factor: E²Brain™ standard: 115 x 75 x 11.6 mm (minimum height) • Complies with the E²Brain™ specification
I/Os	System: • I²C • LPC • PCI-bus • Serial: terminal and console • CompactFlash Communications: • High speed serial UART • CAN • Ethernet (Fast and Gigabit)
Other	Test and Programming: • JTAG/TAP (test access port) Monitor and Control: • Reset • GPIOs (General Purpose IOs) • Switches • Temperature sensing

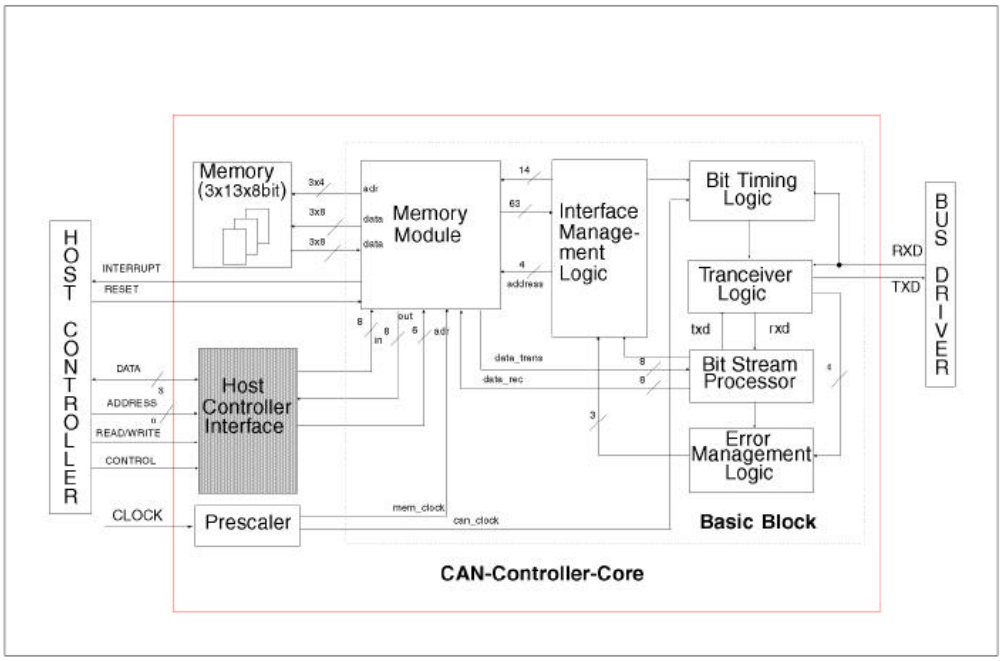
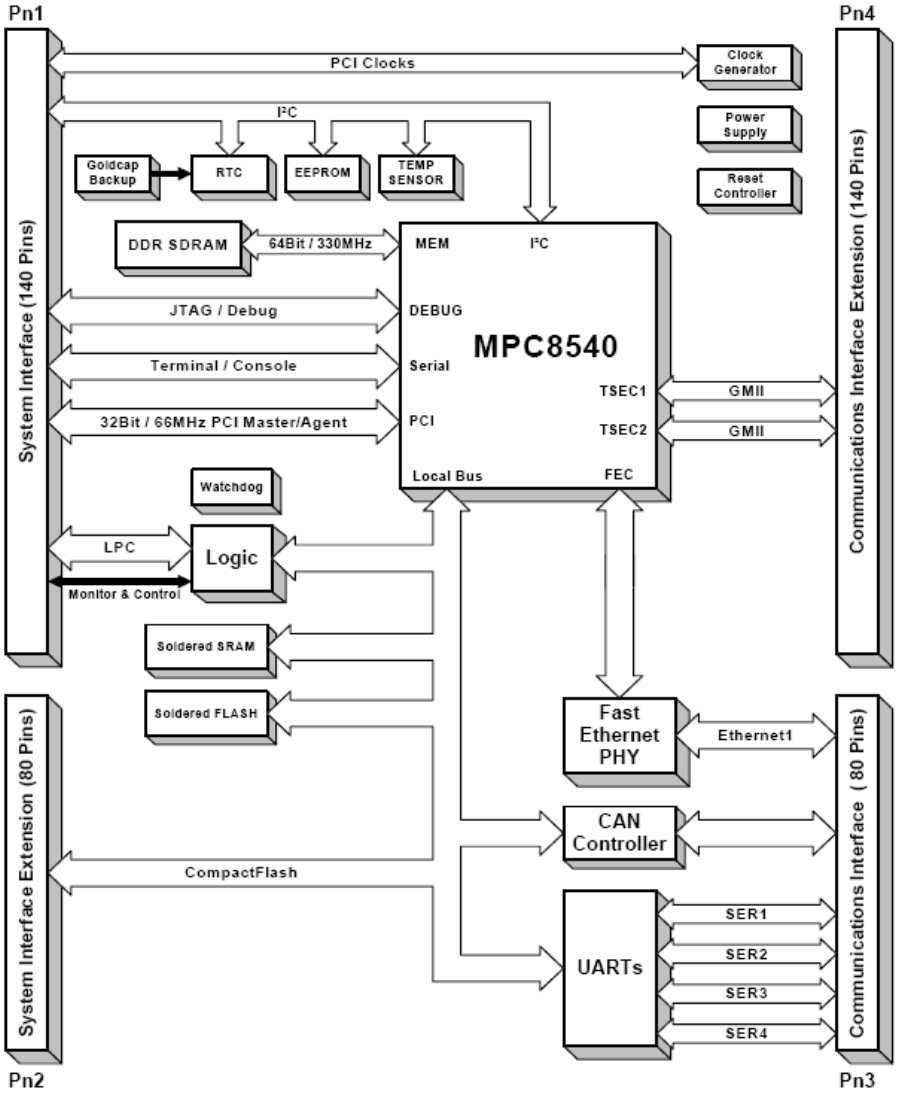


Abb. 3: Blockschaltbild



USART1 Control and Status Registers

12.4.10.Übung 8: Ankopplung eines Peripheriebausteins (1)

Das Host Controller Interface ist über zwei 8 Bit Datenleitungen und eine 6 Bit Adressleitung mit dem Memory Module verbunden. Dadurch ist ein einfacher Austausch dieser Schnittstelle bei Verwendung ander

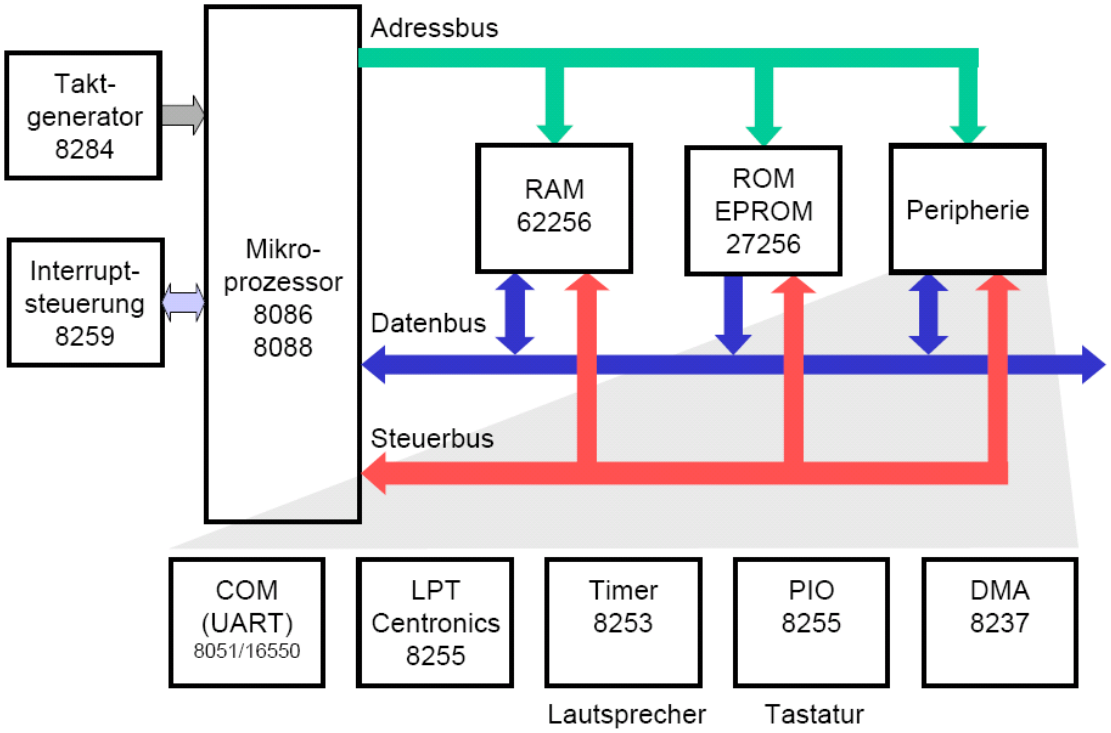


Bild: Ankopplung des Peripheriebausteines PIO 8255 (parallel input output)

Peripheriebausteine

- Timer-Baustein (8253)
- Paralleles Interface (8255)
- Serielles Interface (8251)
- DMA-Controller (8237)
- Programmierbare Interruptsteuerung (8259)

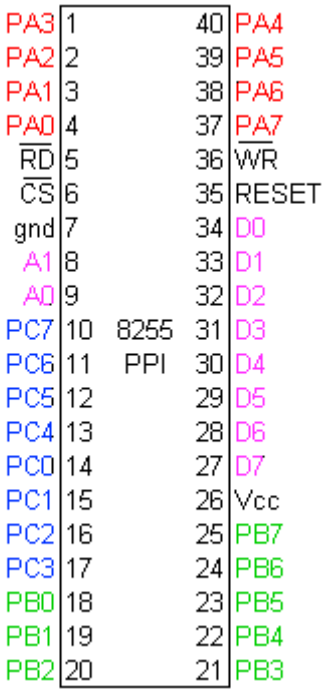


Bild: Pinbelegung des 40pol. 8255 ICs

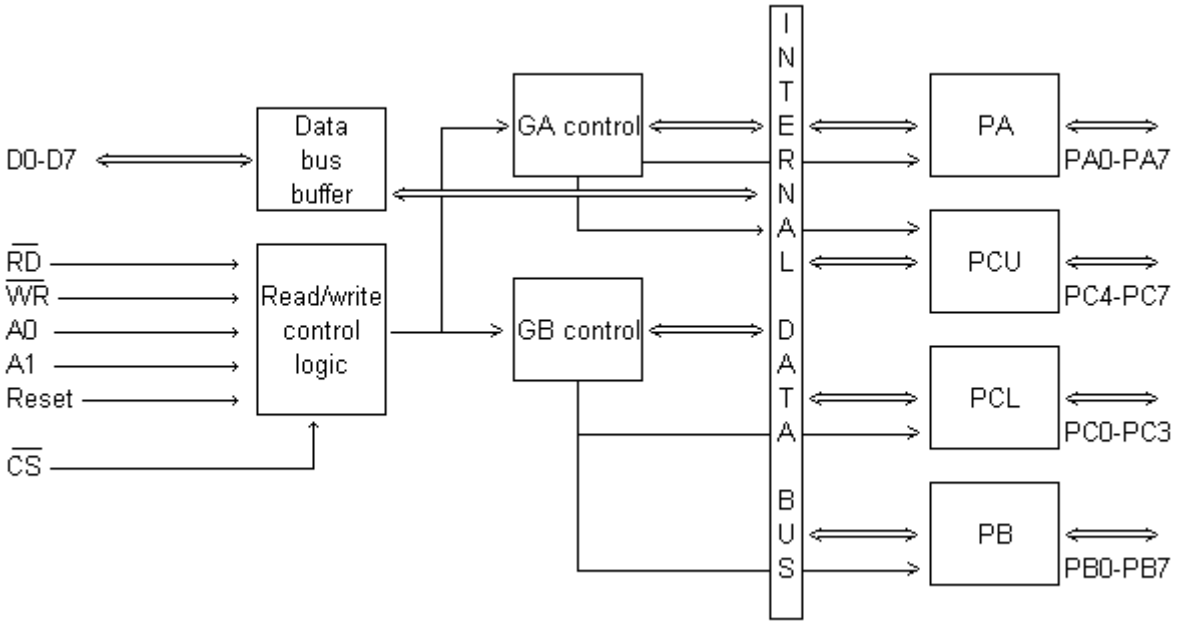


Bild: 8255 Functional Block Diagramm

PPI-Baustein mit drei 8bit TTL-Port

PIO (parallel input output) port-interface) bzw. PPI (peripherial-port-interface) und ist ein universeller I/O-Schaltkreis für die 8-bit parallele Ein-Ausgabe. Der 8255 bzw. uPD71055-C enthält drei Ein-Ausgabeports (bezeichnet mit Port A, B und C), über die der digitale Datenaustausch zwischen einem Prozessor und der Peripherie erfolgt. Die Konfiguration der drei Ein-Ausgabeports erfolgt über ein zusätzliches Register, dem Control-Register bzw. auch Komandoregister oder Status-Register genannt. Die drei Ports A, B und C werden durch das Einschreiben eines Bytes in das Control-Register entsprechend konfiguriert.

Beispiel
Mit dem C-Befehl: OUTb (Basisadresse + 2, Command) wird das Steuerregister des 8255-Portbausteins adressiert. Dieses Register gibt vor, welche der drei E/A-Ports (PA, PB und PC) auf Eingabe oder Ausgabe arbeiten soll. Alle Ports können sowohl als Ausgabe- oder Eingabe-Port fungieren.

Der Port C kann in zwei Hälften unterteilt werden, so dass 4 Bits als Ein- und 4 Bits als Ausgabe dienen können. Das Komandowort, dass zum Status-Port des 8255 gesendet wird, muss vor der jeweiligen Nutzung der E/As erfolgen. Das geschriebene Komandowort ist so lange gültig, bis es mit einem anderen Wert überschrieben wird und muss mindestens initial nach dem Einschalten des Mikrocomputersystems beschrieben werden. Die Initialisierung des Komandoregisters geht nach Wegnahme der Spannungsversorgung wieder verloren.

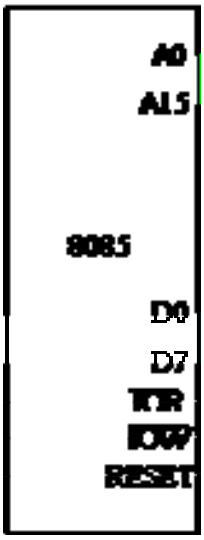
Registerbelegungstabelle für ISA-Karten:

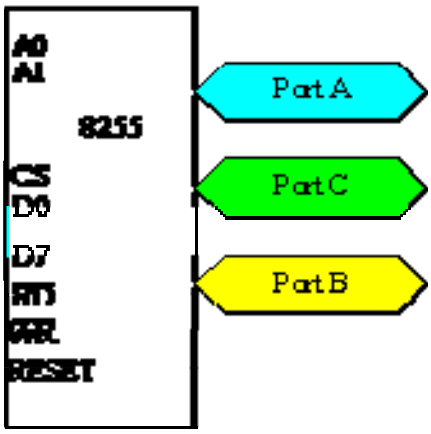
I/O-Adresse	Lesen	Schreiben
Basisadresse + 0	8255/Port A lesen	in 8255/Port A schreiben
Basisadresse + 1	8255/Port B lesen	in 8255/Port B schreiben
Basisadresse + 2	8255/Port C lesen	in 8255/Port C schreiben

Basisadresse + 3	-	Betriebsart des 8255 festlegen

Aufgabe:

Entwicklen Sie eine Schaltung zur eindeutigen Selektion des PIO 8255. Die Basisadresse besitzt die I/O-Adresse 0378h.





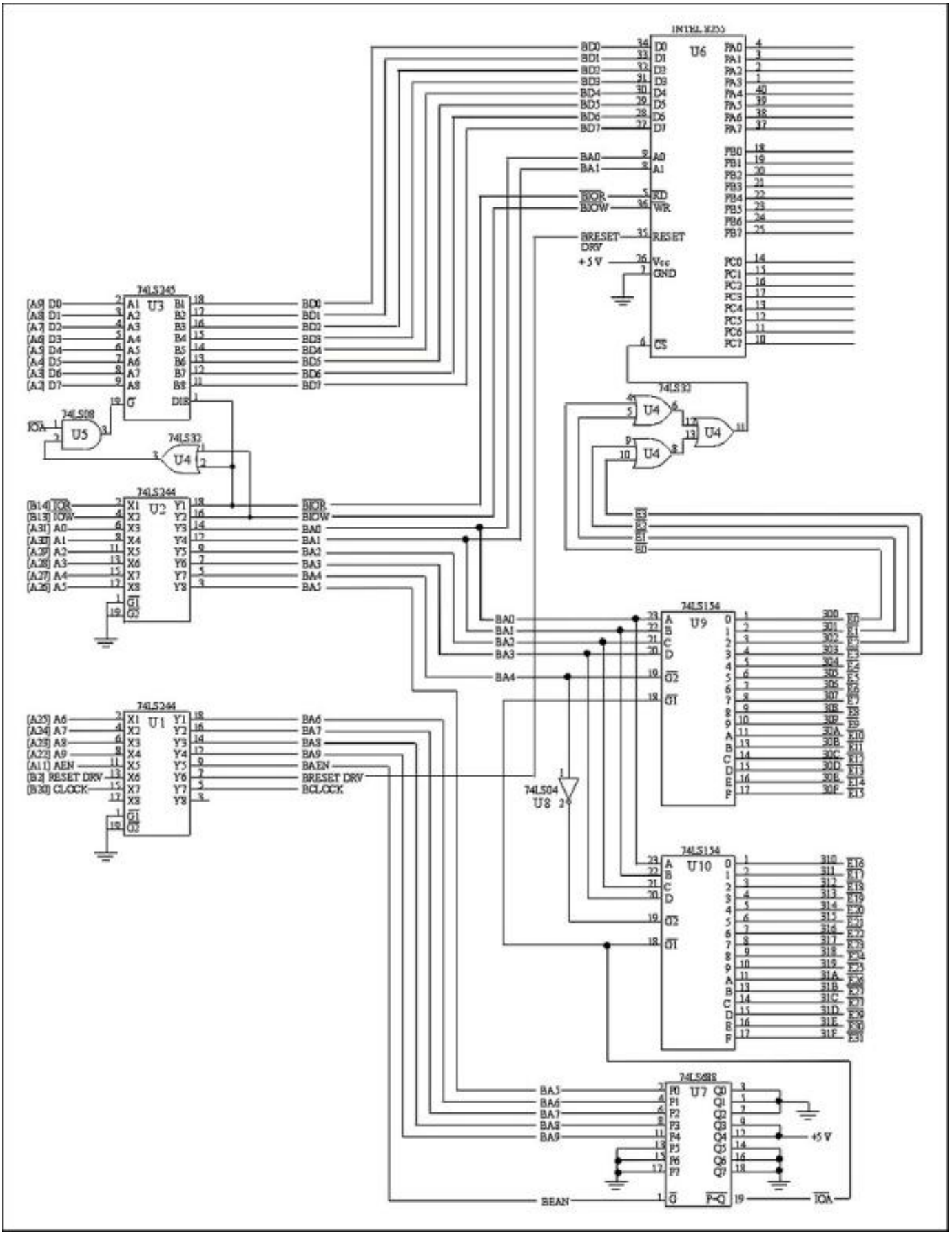


Bild: Beispiel einer Adressdekodierung für einen Portbaustein PIO8255 (U6)

12.4.11.Übung 9: Ankopplung eines Peripheriebausteins (2)

Geben Sie jeweils die Basisadressen zum Peripheriebaustein 8255 an.

Schaltung a)

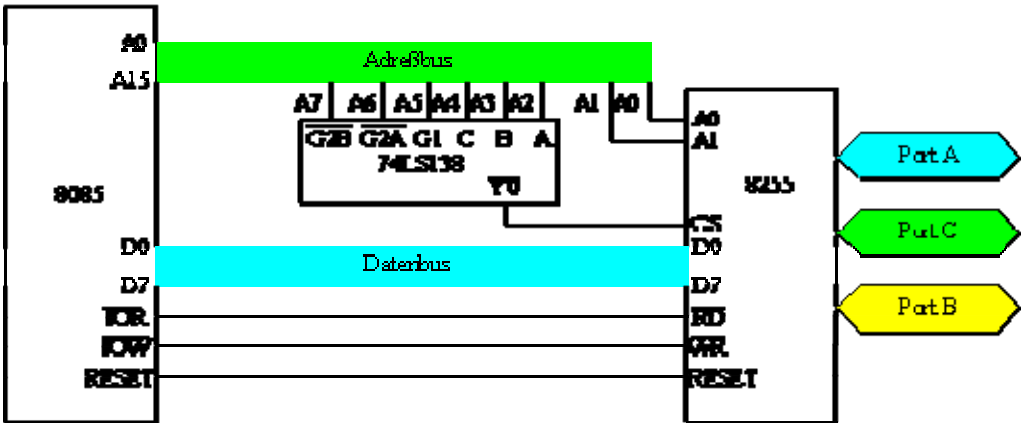


Bild: Schaltung a)

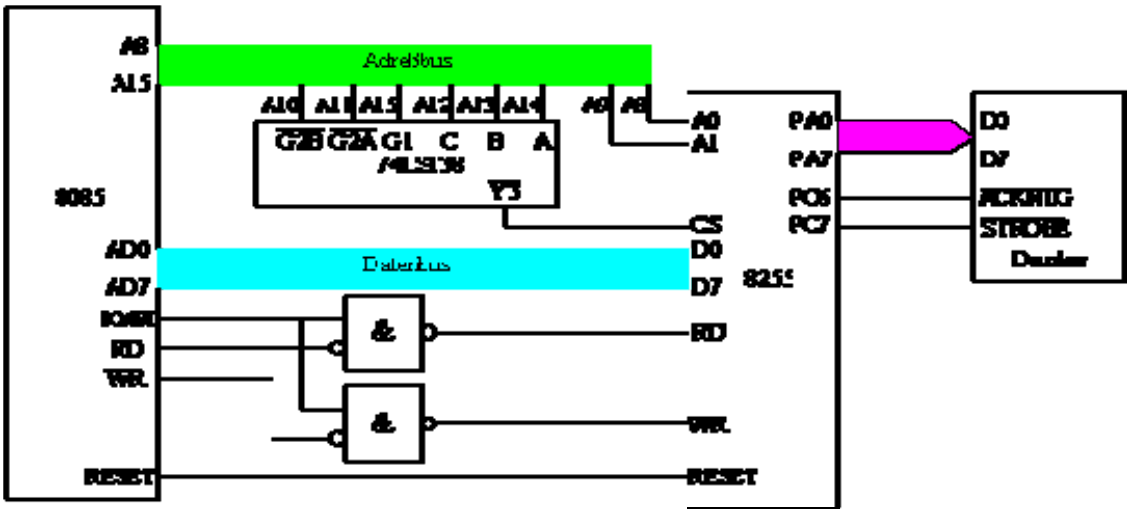


Bild: Schaltung b)