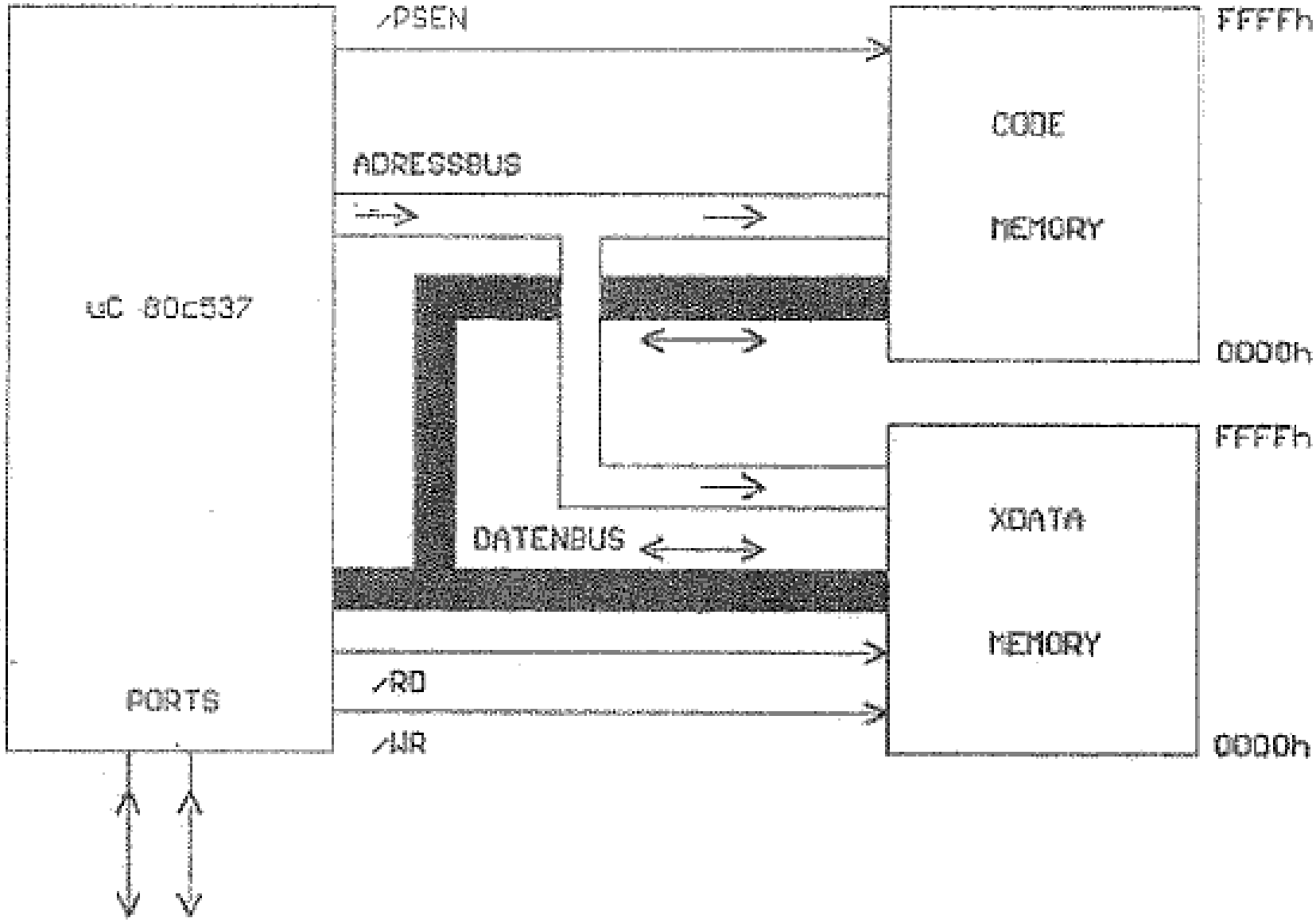
Kapitel 16

Externer Bus

Externer Systembus



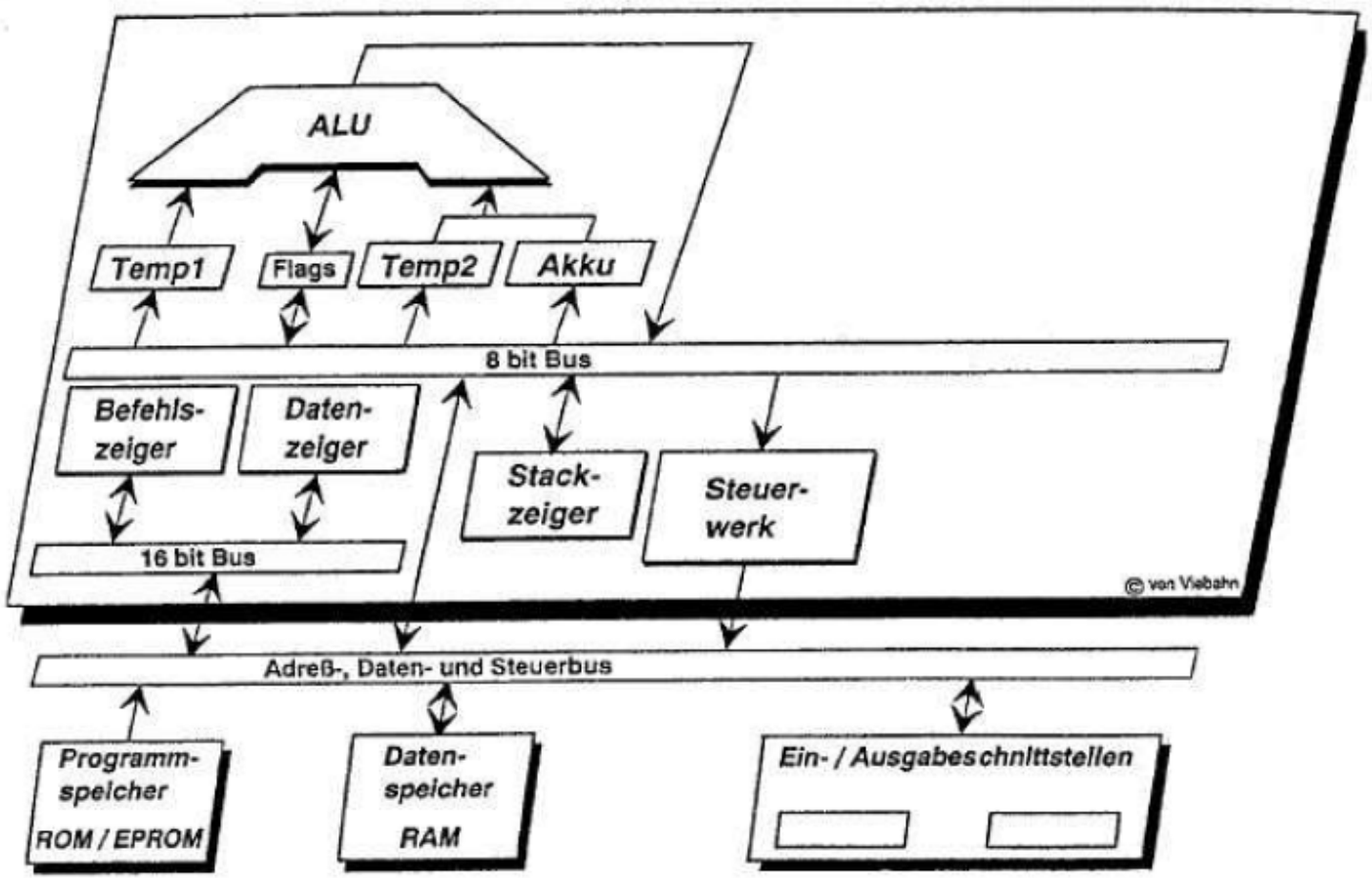
Skript Bild 22, S. 27 : Prinzip eines Mikrocontrollersystems (80c537) mit externem Speicher

80c537 Externer Systembus

- Externer Programmspeicher und externer Datenspeicher können über den **selben** Systembus angeschlossen werden.
  - von Neumann Architektur
- Zwei separate Adressräume zu je 64 kB (0x0000 ... 0xFFFF).
- Steuerleitungen
  - $\overline{RD}$: Read : steuert den Transfer vom Externen Datenspeicher zum Akkumulator.
  - $\overline{WR}$: Write : steuert den Transfer vom Akkumulator zum Externen Datenspeicher.
  - $\overline{PSEN}$: Program Storage ENable : steuert den Transfer vom Externen Programmspeicher zum Befehlsregister.
- Zugriff zum Programmspeicher mit der Steuerleitung $\overline{PSEN}$
- Zugriff zum Datenspeicher mit den Steuerleitungen $\overline{RD}$ und $\overline{WR}$.
- Adresse ist gleichzeitig am XRAM und am Programmspeicher sichtbar;
  - welche Einheit den Transfer tatsächlich ausführt, entscheidet die Steuerleitung $\overline{PSEN}$ bzw. $\overline{RD}$ oder $\overline{WR}$.

80c515c Externer Systembus

- **Prozessor-interner Systembus**
  - nur der Datenbus (8 Bit Bus) ist dargestellt.
  - Adressbus- und Steuerbusdetails sind nicht veröffentlicht.
- **Externer Systembus**
  - zum Anschluss von Speicher- und E/A-Bausteinen
  - 80c5151c ist eine von Neumann-Architektur-Implementierung
  - Adressbus : 16 Bits
  - Datenbus : 8 Bits
  - Steuerbus : 4 Bits



- **Adresse** auf dem externen Adressbus stammt vom
  - **Befehlszeiger** (16 Bits) : **PC**
    - wird für Instruction Fetches verwendet.
    - adressiert ein Byte im externen Programmspeicher.
    - Datenbyte wird nur gelesen und ins Befehlsregister (PS Puffer) übertragen.
    - Befehlszeiger ist nicht per Programm ansprechbar.
  - **Datenzeiger** (16 Bits) : SFR **DPTR**
    - wird für Transportbefehle verwendet.
    - adressiert ein Byte
      - im externen Datenspeicher mit dem Befehl MOVX,
      - in einem E/A-Baustein mit dem Befehl MOVX, oder
      - im externen Programmspeicher mit dem Befehl MOVC (nur lesen).
  - DPTR kann mit regulären Befehlen angesprochen werden (denn es liegt im SFR Bereich).
- **Datenbyte** wird vom/zum **Akkumulator** übertragen.
- **Steuerleitungen**
  - $\overline{RD}$: Read
  - $\overline{WR}$: Write
  - $\overline{PSEN}$: Program Storage ENable
  - ALE : Adress Latch Enable

8051 Prozessor Schnittstelle

Port 0 und Port 2 stehen für die Übergabe von Adresse und Daten zum externen Bus zur Verfügung.

- Im Prozessor : **24 Bits**
  - 16 Bit Adress Register
    - DPTR bzw PC
  - 8 Bit Datenregister
    - Akkumulator bzw. Befehlsregister
- Am Speicherbaustein : **24 Bits**
  - 16 Adressleitungen (A0 … A15)
  - 8 Datenleitungen (D0 … D7)
- An der Prozessorschnittstelle : **16 Bits**
  - Port 0 (8 Bits)
  - Port 2 (8 Bits)
- **Problem** : 24 Bits müssen über 16 Leitungen übertragen werden.
- **Lösung** : Multiplexing von Adressbits A7 ... A0 und Datenbits D7 ... D0 über Port 0.
  - Steuerleitung **ALE** gibt an, ob sich die Adressbits A0 … A7 oder die Datenbits auf Port 0 befinden.
  - Die Adressbits A0 … A7 werden in einem externen Registerbaustein zwischengespeichert.

8051 Prozessor Schnittstelle (Forts.)

- Adressbus Bits A15 ... A8 : **Port 2**
- Adressbus Bits A7 ... A0 : **Port 0** (Zeit-multiplexed)
- Datenbus Bits D7 ... D0 : **Port 0** (Zeit-multiplexed)
- $\overline{RD}$: **Port 3 Bit 7**
- $\overline{WR}$: **Port 3 Bit 6**
- $\overline{PSEN}$ und $\overline{ALE}$ sind Anschlüsse am Prozessorchip
- Auf der Platine :
  - Externen Systembusleitungen,
  - Prozessorbaustein,
  - Speicherbausteine,
  - E/A-Bausteine,
  - Latch-Register 74573.

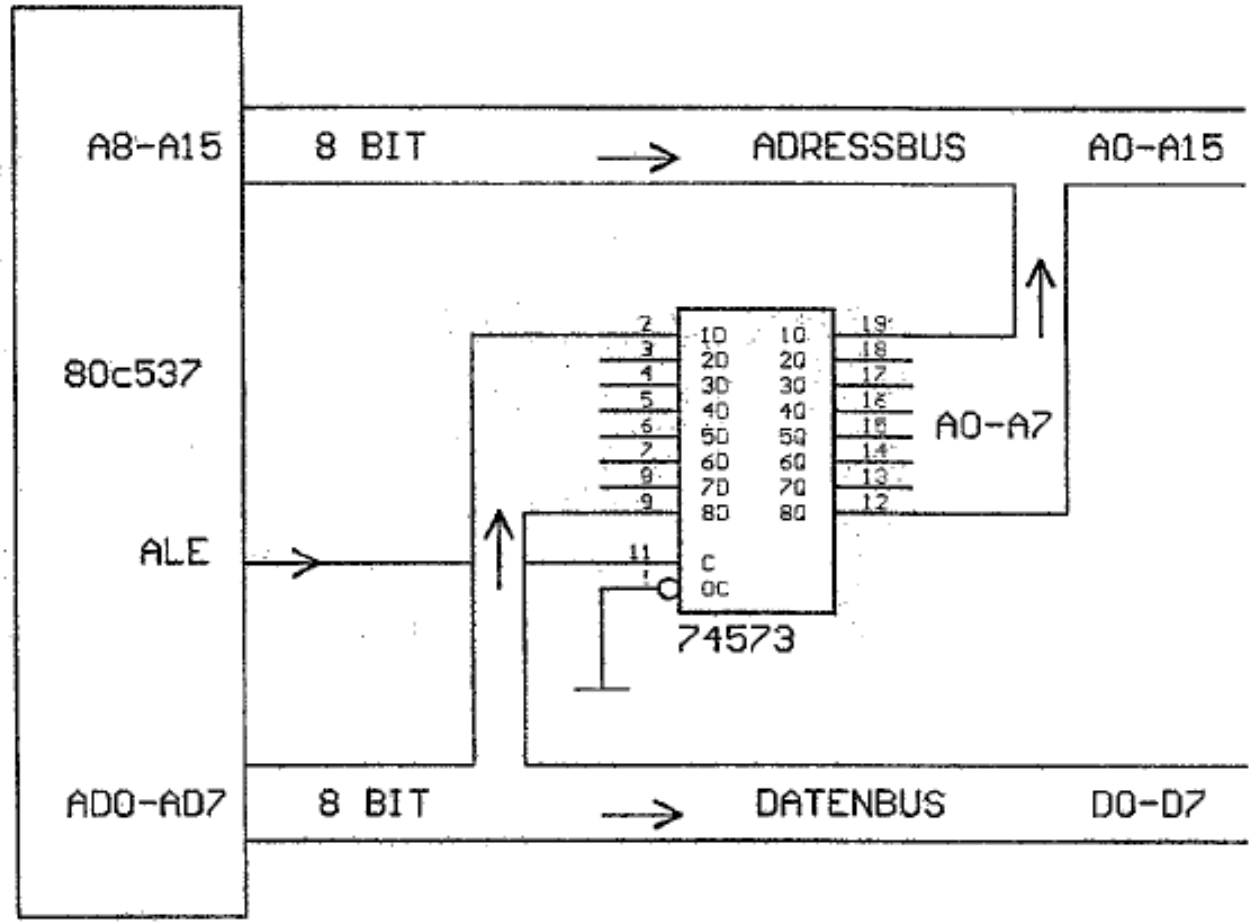
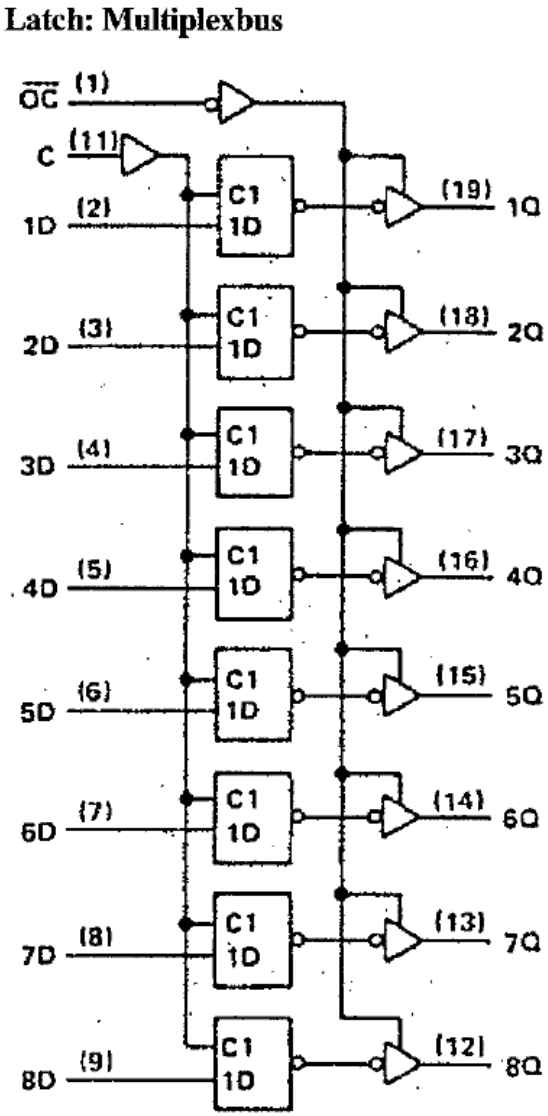


Bild 11: Erzeugung des vollständigen Adressbusses

Die Output Control (OC) Leitung an Masse aktiviert die Treiber für die Adressbusleitungen A7 ... A0 permanent.

Latch-Register Baustein



(EACH LATCH)

INPUTS			OUTPUT
ENABLE			
$\overline{OC}$	C	D	Q
L	H	H	H
L	H	L	L
L	L	X	Q_0
H	X	X	Z

Bild 19: Beispiel: 74 LS (HCT) 573

Multiplexing von Adress- und Datenbits

- Speicherzugriff in zwei Schritten (am Beispiel Programmspeicherzugriff)
  - **1. Schritt** : Addressbits A7 ... A0 werden im Baustein 74573 zwischengespeichert.
    - Steuerwerk schickt Adressbits A7 ... A0 auf Port 0.
    - Steuerwerk aktiviert ALE , PSEN bleibt deaktiviert
      - 8 Bits werden vom Port 0 in den Baustein 74573 übernommen und dort verlatched.
    - Steuerwerk deaktiviert ALE wieder.
    - 74573 Baustein ist ein 8 Bit Register. Es ist ein Zwischenspeicher für Adressbits A7 ... A0.
    - Der Ausgang des 74573 Bausteins bildet zusammen mit den Port 2 Leitungen die Adressbits A15 ... A0.
  - **2. Schritt** : Byte wird vom Programmspeicher in den Prozessor übertragen
    - Steuerwerk aktiviert $\overline{\text{PSEN}}$
    - Speicherbaustein übernimmt die Adressbits A15 ... A0 vom Adressbus und selektiert das gewünschte Byte.
    - Der Speicherbaustein stellt das adressierte Byte auf den externen Datenbus.
    - Das Steuerwerk leitet das Byte auf dem Datenbus in das Befehlsregister.
- Beim Zugriff auf den Datenspeicher werden die Steuerleitungen $\overline{\text{RD}}$ und $\overline{\text{WR}}$ eingesetzt.

External Bus Timing : Befehle auslesen

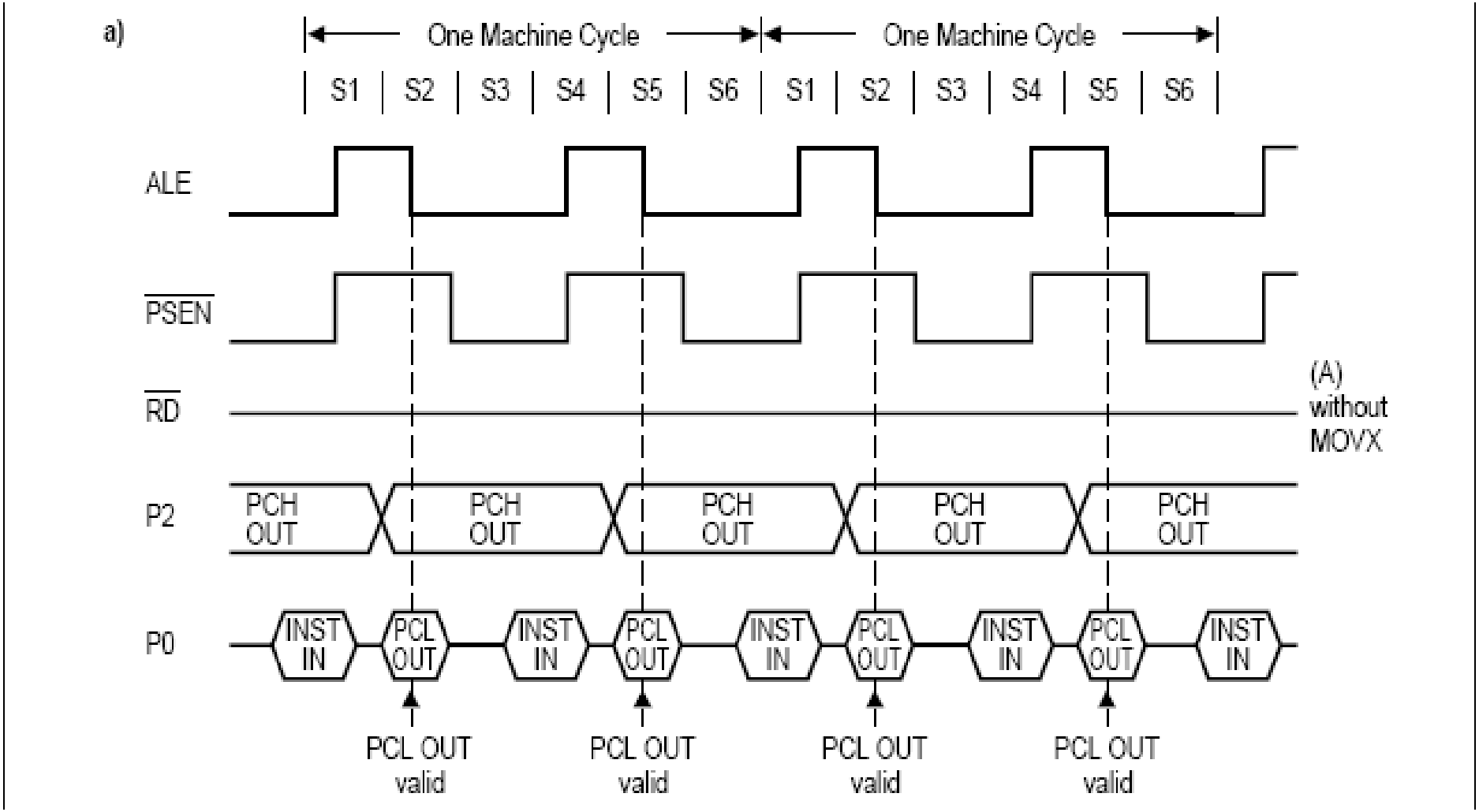


Figure 4.1 : External Program Memory Execution

External Bus Timing : Befehle auslesen und XRAM Zugriff

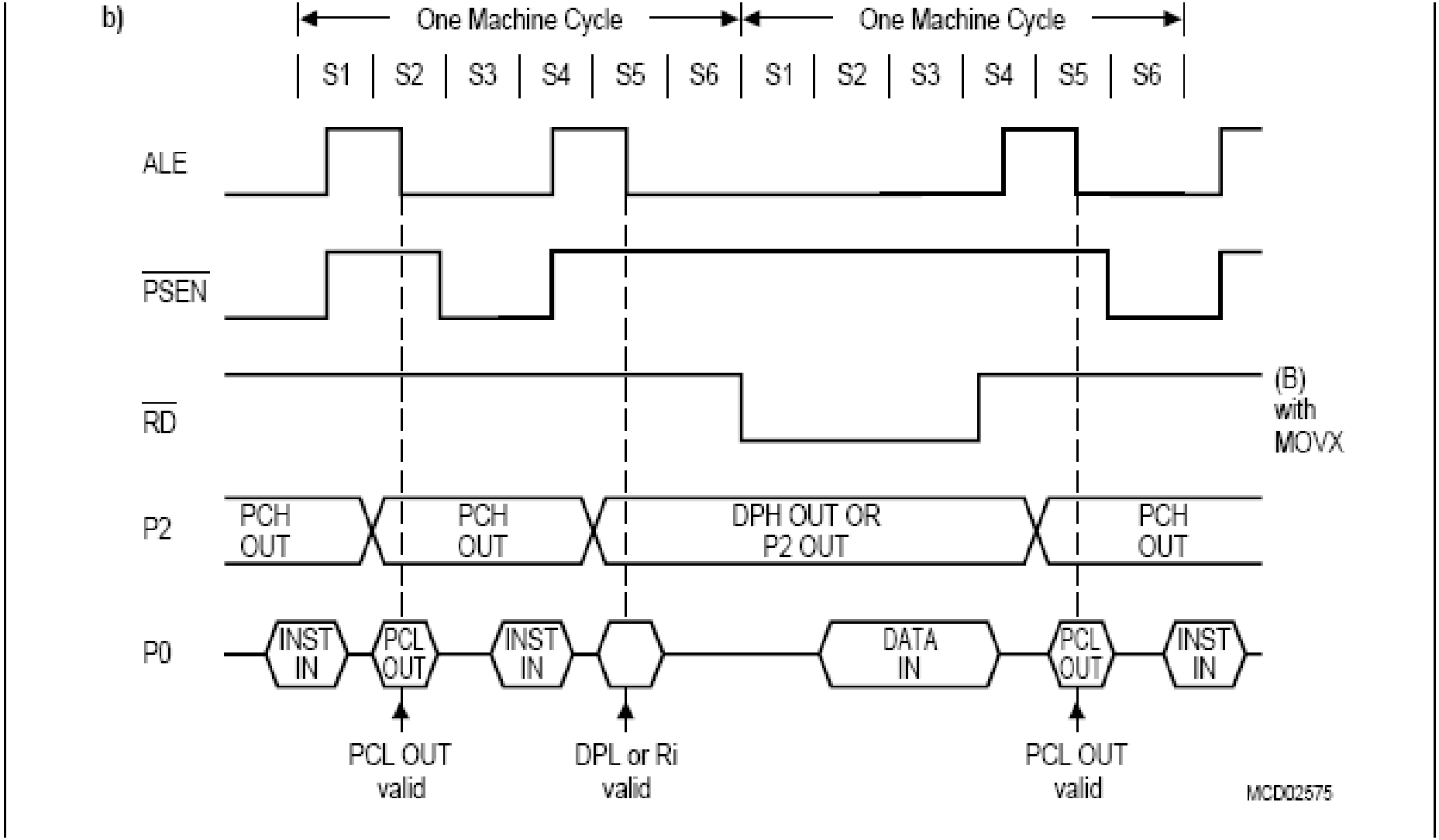


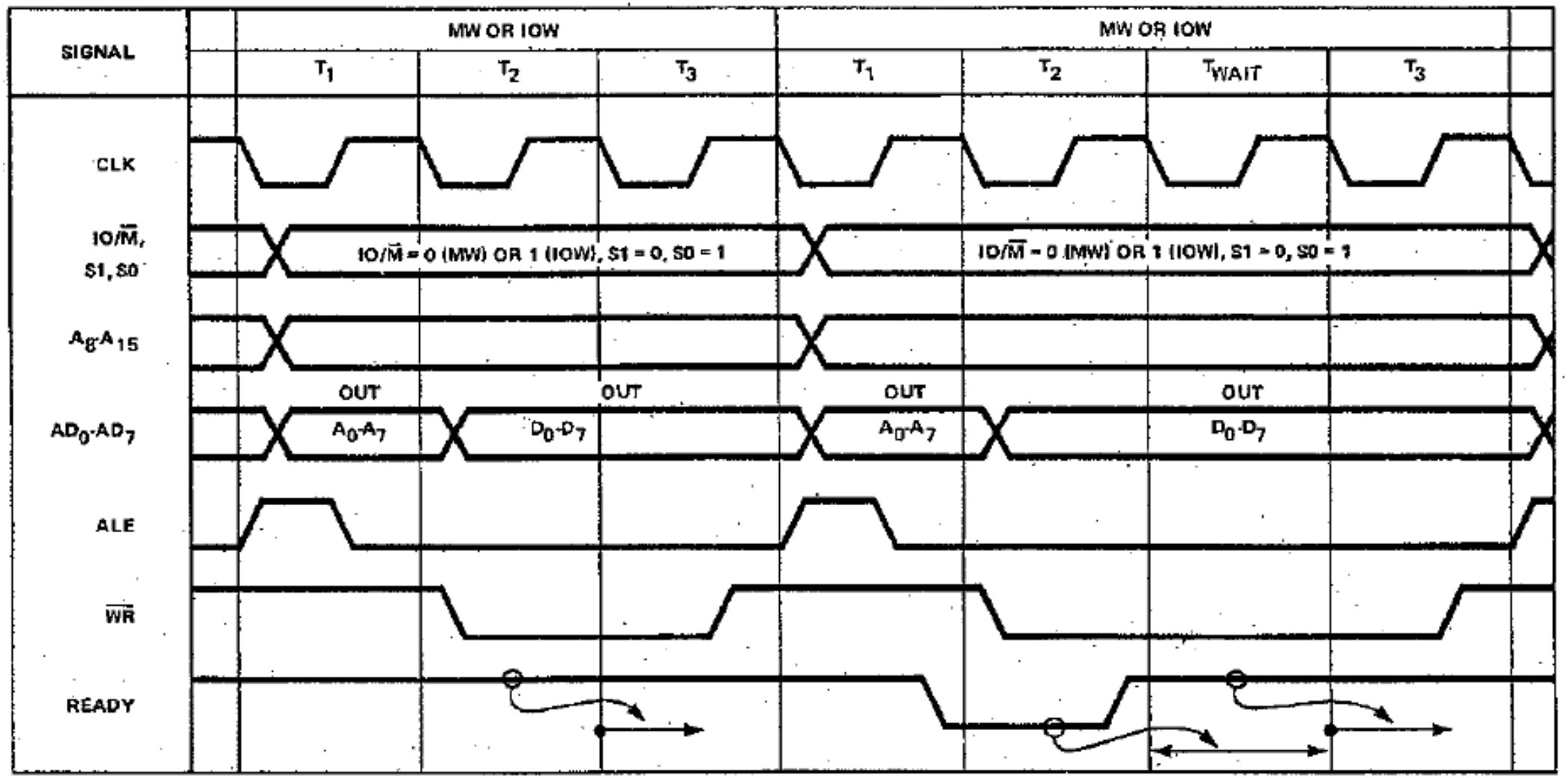
Figure 4.1 : External Program Memory Execution

External Bus Timing (Forts.)

- **Bild a)** : Timing bei Zugriffen zum Programmspeicher
  - Für Programmspeicherzugriffe werden im Step2 und im Step 5 des Maschinenzyklus die Bits 15 ... 8 des Befehlszählers (PCH OUT) am Port 2 und die Bits 7 ... 0 des Befehlszählers (PCL OUT) am Port 0 zur Verfügung gestellt.
  - Während ALE aktiv ist, wird der Wert des Port 0 Ausgangs im Adress-Latch 74LS374 gespeichert.
    - Die ALE Leitung ist mit den Clock-Eingangsleitungen der Flipflops verbunden.
    - Im Adress-Latch bleibt also der Wert gespeichert, der zum Zeitpunkt kurz vor der Deaktivierung von ALE auf den Dateneingangsleitungen des Adress-Latches anliegt.
  - Die Aktivierung der $\overline{\text{PSEN}}$ Leitung zu Beginn von Step 3 und Step 6 startet die Leseoperation im Speicherbaustein.
  - Kurz vor der Deaktivierung von $\overline{\text{PSEN}}$ in Step 4 und Step 1 übernimmt der Prozessor das Byte an den Eingangsleitungen des Ports 0.
    - Der Prozessor empfängt kein Quittungssignal vom Speicherbaustein.
    - Der Speicherbaustein muss schnell genug sein, um das vorgegebene Timing einhalten zu können.
- **Bild b)** : Timing bei eingeschobenem Zugriff zum Datenspeicher
  - Für Datenspeicherzugriffe werden die Bits 7 ... 0 des DPTR am Port 0 zur Verfügung gestellt.
  - Übernahme der niederwertigen Adressbits in das Adress-Latch wie beim Programmspeicherzugriff.
  - Die Aktivierung der $\overline{\text{RD}}$ bzw. $\overline{\text{WR}}$ Steuerleitung startet die Operation im Speicherbaustein.
  - Übernahme des Datenbytes wie beim Programmspeicherzugriff.
  - Der Zugriff zumm Datenspeicher dauert einen ganzen Maschinenzyklus (6 Steps).

Speicherzugriffe mit WAIT-Zyklen

Wenn der Speicherbaustein nicht schnell genug ist,kann er mit WAIT-Zyklen den Prozessor abbremsen.



Skript Bild 34, S. 38 : Write-Cycle mit und ohne WAIT State (8085 und 8086)

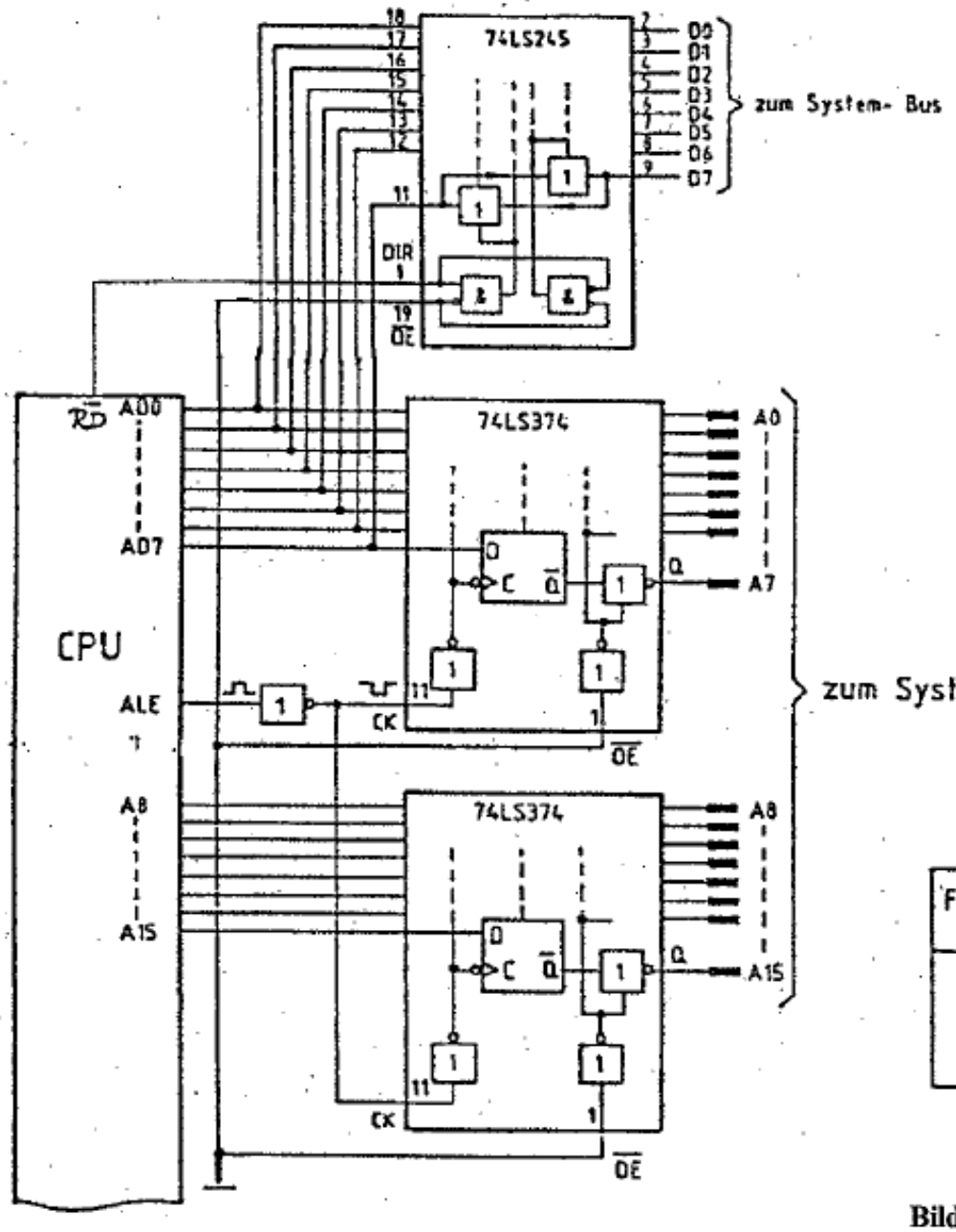
Speicherzugriffe mit WAIT-Zyklen (Forts.)

- Zusätzliche Steuerleitung **READY** vom Speicherbaustein zum Prozessor.
  - Die READY Steuerleitung signalisiert dem Prozessor, wann der Zugriff am Speicherbaustein beendet ist.
  - Wenn die READY Leitung bei der Aktivierung von $\overline{RD}$ bzw. $\overline{WR}$ aktiv ist, kann der Speicherzugriff mit optimaler Geschwindigkeit erfolgen.
  - Wenn die READY Leitung bei der Aktivierung von $\overline{RD}$ bzw. $\overline{WR}$ inaktiv ist, schiebt der Prozessor solange WAIT-Zyklen ein, bis die READY Leitung wieder aktiv wird.
- Dieses Verhalten trifft **nicht für 8051** Prozessoren zu.
  - Beim 8051 müssen die Speicherbausteine schnell genug sein.

Direct Memory Access - DMA

- Im Fall eines 8051-basierten Systems spielt der Prozessor immer die Rolle des Busmasters und kontrolliert die Operationen auf dem externen Systembus.
- Auf anderen Systemen ist auch DMA verfügbar.
- Die **Direct Memory Access (DMA)** Methode erlaubt einem externen Busteilnehmer, Datentransfers ohne Mitwirkung des Prozessors durchzuführen.
  - Zu diesem Zweck aktiviert dieser Busteilnehmer (z.B. ein Festplattencontroller) die **DMA-Request** Steuerleitung.
    - Das signalisiert dem Prozessor, dass er die Kontrolle über den Systembus wünscht.
  - Zu gegebener Zeit schaltet der Prozessor seine Bustreiber hochohmig und aktiviert die **DMA-Acknowledge** Steuerleitung.
    - Das zeigt der anfordernden Einheit an, dass der Prozessor die Kontrolle über den Bus abgibt.
  - Jetzt überträgt der Busteilnehmer Daten von/zum Speicher.
    - Dazu steuert der Busteilnehmer den Adressbus, den Datenbus, und die Steuerleitungen.
  - Zum Zeichen, dass der DMA-Datentransfer beendet ist, deaktiviert der Busteilnehmer die DMA-Request-Leitung.
  - Der Prozessor deaktiviert die DMA-Acknowledge Steuerleitung und nimmt die normale Programmausführung wieder auf.
- DMA wird von vielen Systemen, vor allem der PC Klasse, unterstützt.
- Die **8051 Prozessoren kennen DMA nicht**.

Beispielrealisierung einer Systembusschnittstelle



Funktionstabelle 74LS374

Freigabe $\overline{OE}$	Takt CK	Eingang D	Ausgang Q
L	⎍	H	H
L	⎍	L	L
L	L	x	Q_0
H	x	x	hochohmig

x: Pegel kann H oder L sein
⎍: ansteigende Flanke
Q_0: keine Änderung

Bild 20: CPU mit flexiblem System

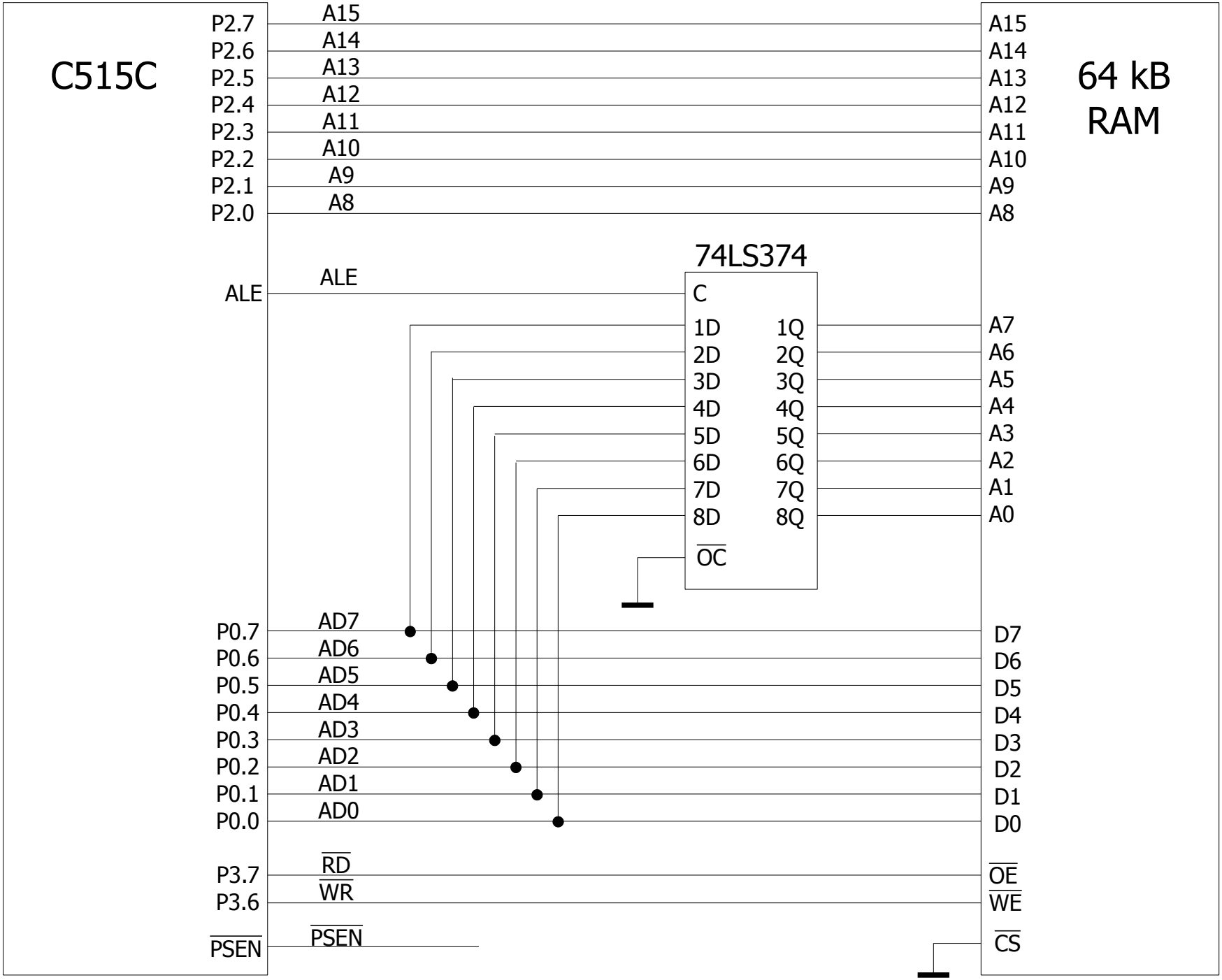
Beispielrealisierung einer Systembusschnittstelle (Forts.)

- **Bidirektionaler** Baustein 74LS245 zum Anschluss des Datenbusses
- **Latch Baustein** 74LS374 treibt sowohl Adressbus Bits A15 ... A8 als auch Adressbus Bits A7 ... A0.
  - Für Adressbus Bits A15 ... A8 würde auch ein einfacher Treiberbaustein genügen. Dann wäre allerdings ein Baustein mit zusätzlicher Teilenummer erforderlich.

Schlussbetrachtung

- Die Realisierung mit dem zusätzlichen Latch-Register ist nur erforderlich, weil im Fall des 8051 Prozessors nur 16 Anschlussleitungen (Port 0 und Port 2) für die Übergabe von 24 Bits (16 Bits Adresse und 8 Bits Daten) zu Verfügung stehen.
  - Wären 24 Leitungen vorhanden, würden einfache Treiber ausreichen.
- Andererseits ist die Realisierung des Systembusses aus der Sicht der angeschlossenen Bausteine transparent.
  - Die angeschlossenen Bausteine sehen lediglich 16 Adressbusleitungen, 8 Datenbusleitungen und 3 Steuerleitungen (+$\overline{CS}$).

Namen der Signale und Leitungen



Anschlüsse an den externen Bus des 8051

Anschlussleitungen eines **64 kB RAM** Bausteins

Anschlussleitungen des Bausteins	**Abkürzung**
16 Adressbitleitungen	A15 ... A0
8 Datenbitleitungen	D7 ... D0
Output Enable Steuerleitung	$\overline{OE}$
Write Enable Steuerleitung	$\overline{WE}$
Chip Select	$\overline{CS}$

- Ein Lesezugriff ist aus der Sicht des Speichers eine Output Operation, daher $\overline{OE}$.
- Wenn für einen Adressraum mehrere Bausteine vorhanden sind, dient die Chip Select Leitung dazu, den gewünschten Baustein auszuwählen.
  - Nur wenn seine Chip Select Leitung aktiviert ist, führt der Speicherbaustein einen Zugriff durch.
  - Mit $\overline{CS}$ an Masse ist ein Baustein permanent ausgewählt.

Anschluss eines **32 kB** Speicherbausteins

- Verbindung der Anschlussleitungen A14 ... A0 des Speicherbausteins mit den Adressbusleitungen A14 … A0.
- **Ansatz 1** : Adressbusleitung **A15** wird nicht verbunden.
  - **Konsequenz :** Die Adresse 0x0000 und 0x8000 spricht das selbe Byte im Speicher an.
  - Keine 100%ige Lösung. Aber es funktioniert, solange das Programm sicher stellt, dass nur ein Bereich (z.B. Adressen 0x0000 ... 0x7FFF) angesprochen wird.
  - Man spricht in einem solchen Fall von **Unvollständiger Adressdekodierung**.
- **Ansatz 2** : Adressbusleitung **A15** wird mit der $\overline{CS}$ Anschlussleitung des Bausteins verbunden.
  - Wenn der Prozessor die Adressleitung A15 auf 0 schaltet (für alle Adressen 0x0000 … 0x7FFF), wird der Baustein wegen $\overline{CS}$ = 0 ausgewählt.
  - Wenn der Prozessor die Adressleitung A15 auf 1 schaltet (für alle Adressen 0x8000 … 0xFFFF), wird der Baustein wegen $\overline{CS}$ = 1 nicht ausgewählt.
  - Man spricht in diesem Fall von **Vollständiger Adressdekodierung**.

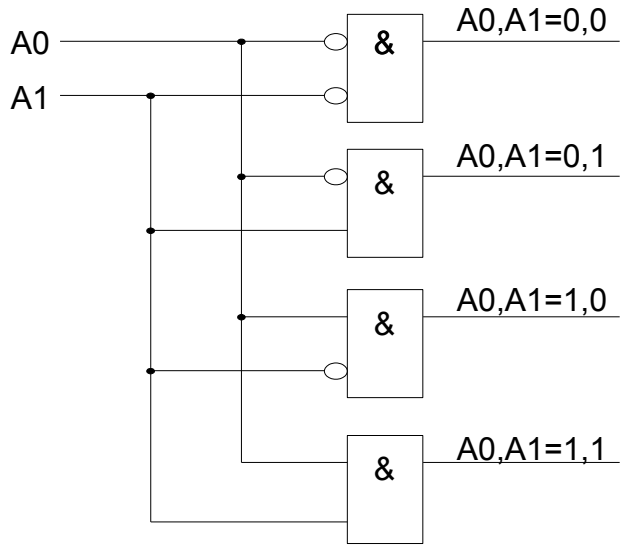
Vollständige Adressdekodierung : Alle Bits des Adressbusses werden zur Adressierung herangezogen. Damit kann jedes Byte des externen Bausteins über **genau eine Adresse** angesprochen werden.

Anschluss von zwei 16 kB Speicherbausteinen an den Adressen 0x0000 und 0x4000

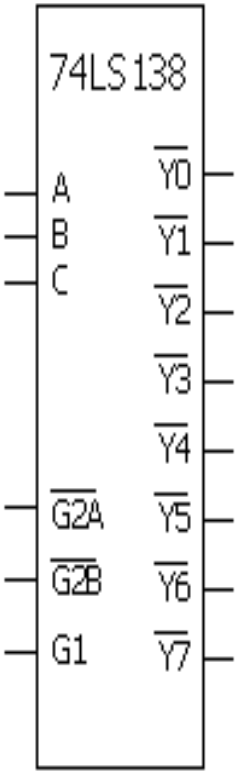
- Adressbusleitungen A13 ... A0 werden mit den Adressanschlüssen A13 ... A0 der beiden Speicherbausteine verbunden.
- Chip-Select des einen Bausteins aktivieren, wenn A15 ... A14 = 00B. Dann belegt dieser Baustein die Adressen 0x0000 ... 0x3FFF.
- Chip-Select des anderen Bausteins aktivieren, wenn A15 ... A14 = 01B.Dann belegt dieser Baustein die Adressen 0x4000 ... 0x7FFF.
- Benützung eines **Adressdekoders**.

Prinzip eines Dekoders

- Jede mögliche Kombination der Eingangsleitungen wird dekodiert und treibt eine individuelle Ausgangsleitung.
- n Eingangsleitungen → 2^n Ausgangsleitungen.



Dekoderbaustein 74LS138



Inputs						Outputs							
$\overline{G2A}$	$\overline{G2B}$	G1	C	B	A	$\overline{Y0}$	$\overline{Y1}$	$\overline{Y2}$	$\overline{Y3}$	$\overline{Y4}$	$\overline{Y5}$	$\overline{Y6}$	$\overline{Y7}$
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	L	L	H	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	L	H	H	H	H	H	L	H	H	H	H
L	L	H	H	L	L	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	H	H	L	H	H	H	H	H	H	L	H
L	L	H	H	H	H	H	H	H	H	H	H	H	L

- 3 Eingangsleitungen (A, B, C) → 2^3 = 8 Ausgangsleitungen ($\overline{Y0}$ ... $\overline{Y7}$)
- 3 Dekoder-Select Leitungen $\overline{G2A}$, $\overline{G2B}$, G1
  - Eventuell für mehrstufige Dekoderschaltungen (Schaltzeiten beachten !)
    - Wenn $\overline{G2A}$, $\overline{G2B}$, *G1* = 0, 0, 1, wird die durch A, B, C ausgewählte Ausgangsleitung $\overline{Yn}$ aktiviert (siehe Wahrheitstabelle).

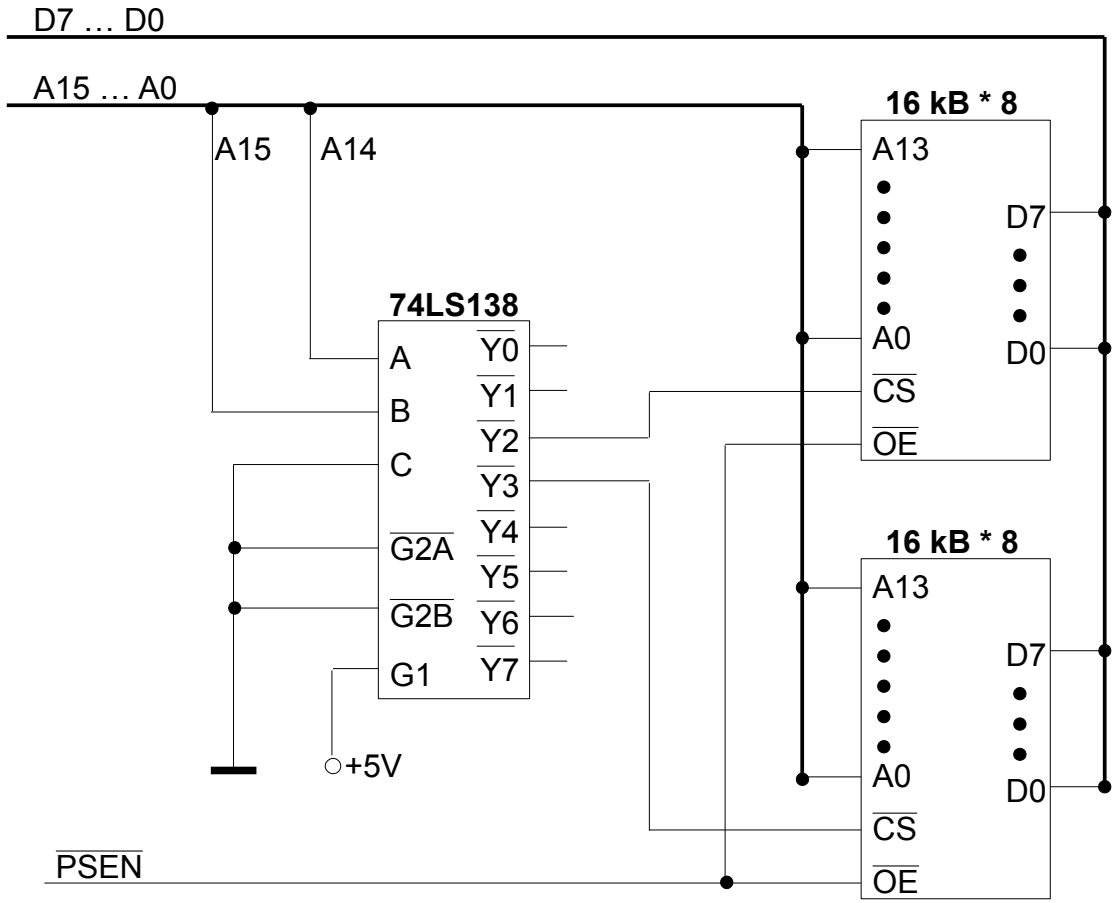
Beispiel : Zwei 16 kB ROM-Bausteine bei 0x8000 und 0xC000

Baustein 1 Adresse min. : 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
max. : 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1

Baustein 2 Adresse min. : 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
max. : 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

|----------Direkt-Adressteil----------|

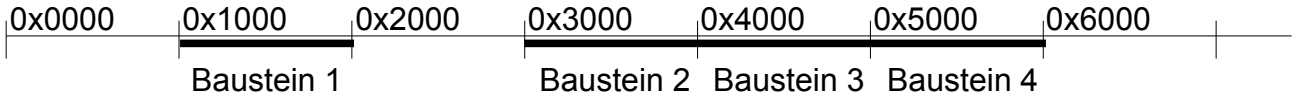
|-----| CS-Adressteil



Vorgehensweise

- Ermitteln der minimalen und der maximalen Adresse für jeden Baustein.
  - A_{min} = Startadresse des Bereichs; A_{max} = Startadresse + Größe - 1
- Ermitteln des direkten Adressteils für jeden Baustein.
  - Die Adressbits A0 ... An für 2^{n+1} adressierbare Elemente (Bytes, Bits, ...).
  - Jede Kombination der Adressbits A0 ... An wird gebraucht, um alle Speicherelemente im Speicherbaustein adressieren zu können.
- Ermitteln des CS-Adressteils für jeden Baustein.
  - Die anderen Bits An+1 ... A15 (beim 8051) sind der CS-Adressteil.
- Ermitteln der für alle Bausteine gleichen Bits des CS-Adressteils, der **konstante CS-Adressteil**.
  - Die Bits des CS-Adressteils, die für Bausteine unterschiedlich sind, **müssen** in die Dekodierschaltung aufgenommen werden, damit keine Mehrfachadressierung von Bausteinen auftreten kann.
  - Für eine vollständige Adressdekodierung müssen auch die Bits des konstanten CS-Adressteils in die Dekodierschaltung aufgenommen werden.
  - Einzelne Bits des konstanten CS-Adressteils können auch unverdrahtet bleiben, wenn eine unvollständige Dekodierung ausreichend ist. In diesem Fall können die Speicherelemente über mehrere unterschiedliche Adressen angesprochen werden. Das Programm muss dann dafür sorgen, dass nur Adressen aus dem gewünschten Bereich verwendet werden.
  - Die Bits des CS-Adressteils, die für Bausteine unterschiedlich sind, können nur an die Eingänge A, B oder C des Dekoderbausteins angeschlossen werden.
  - Für die Dekodierung der Bits des konstanten CS-Adressteils können auch die Leitungen $\overline{G2A}$, $\overline{G2B}$ und G1 verwendet werden.

Beispiel : Vier 4 kB Speicherbausteine bei 0x1000, 0x3000, 0x4000 und 0x5000



	Baustein 1	**Baustein 2**	**Baustein 3**	**Baustein 4**
Min.	0001 0000 0000 0000	0011 0000 0000 0000	0100 0000 0000 0000	0101 0000 0000 0000
Max.	0001 1111 1111 1111	0011 1111 1111 1111	0100 1111 1111 1111	0101 1111 1111 1111
Dir.	xxxx xxxx xxxx	xxxx xxxx xxxx	xxxx xxxx xxxx	xxxx xxxx xxxx
CS	0001	0011	0100	0101

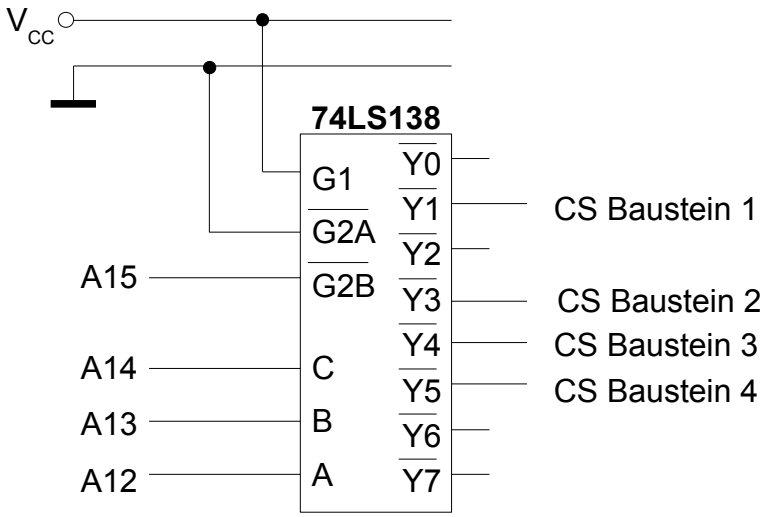
Direkter Adressteil : A0 ... A11

Nicht-konstanter CS-Adressteil : A12 ... A14

Konstanter CS-Adressteil : A15

Vollständige Dekodierung : A15 an $\overline{G2B}$.

	A14	A13	A12	
	C	B	A	
B1	0	0	1	$\overline{Y1}$
B2	0	1	1	$\overline{Y3}$
B3	1	0	0	$\overline{Y4}$
B4	1	0	1	$\overline{Y5}$



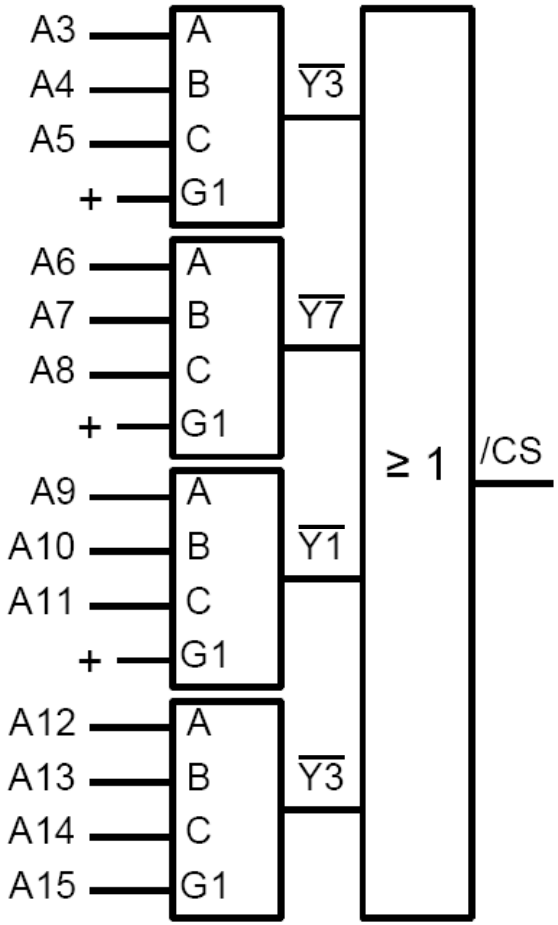
Unvollständige Dekodierung : A15 nicht angeschlossen → 1. Byte in Baustein 1 adressierbar über 1000H **und** 9000H.

Beispiel: vollständige Dekodierung eines Bausteins mit 8 internen Registern

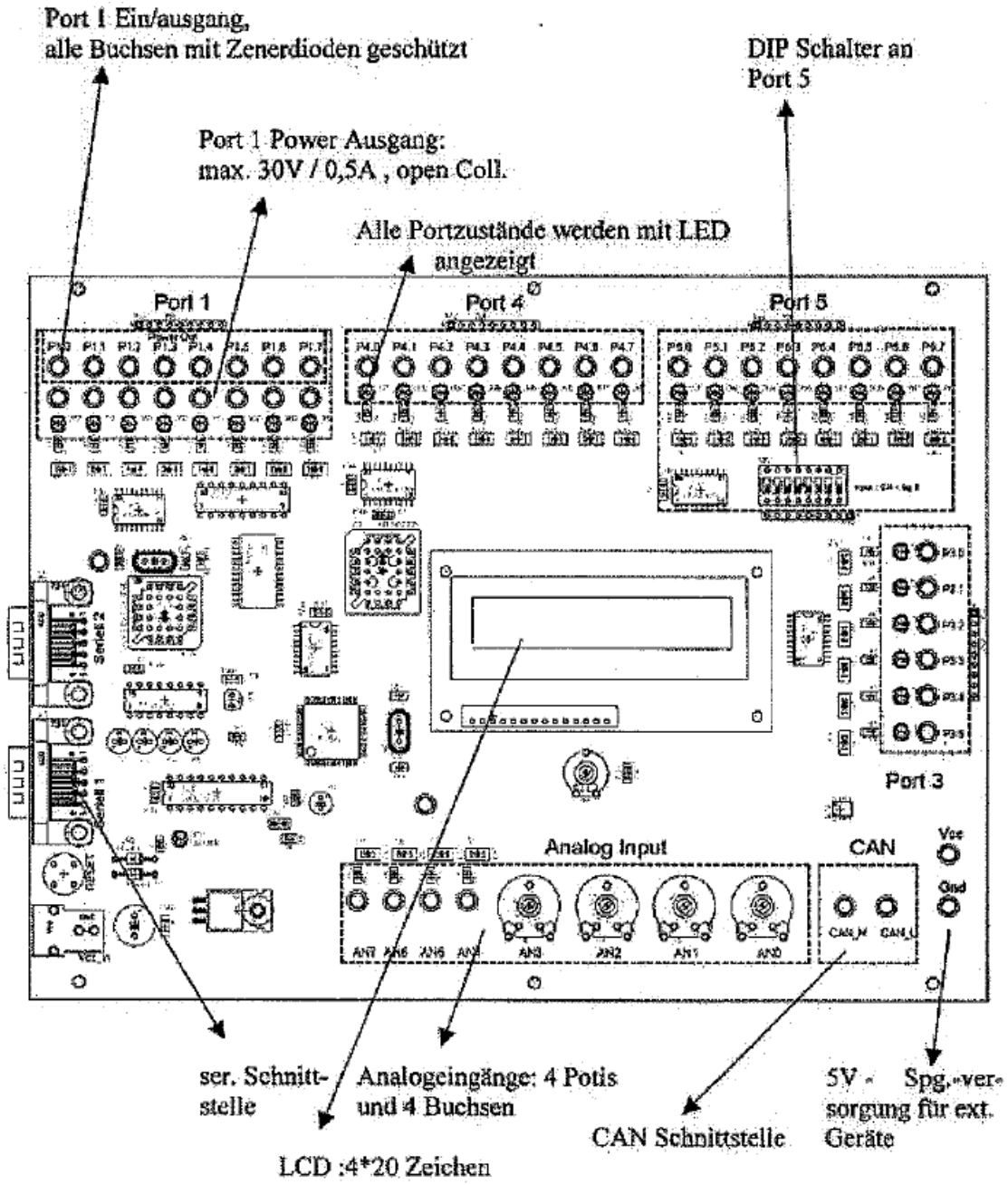
- 8 Register → 3 Bit interne Adresse des Bausteins
- Adressbereich sei 0xB3D8 – 0xB3DF
- 4 Dekoder 74LS138 + ein OR-Gatter
- externe Adresse:1011 0011 1101 1xxx

1	0	1	1	0	0	1	1	1	1	0	1	1	x	x	x
A15	**A14**	**...**	**A12**	**A11**	**...**	**A9**	**A8**	**...**	**A6**	**A5**	**...**	**A3**	**A2**	**...**	**A0**

- Signallaufzeiten beachten !!



MCT Laborkoffer



Laborkoffer (Forts.)

- Auf der Platine des Laborkoffers sind montiert :
  - 80c515c Mikrocontroller
  - 128 kB Flash ROM (nur 32 kB genutzt) für das Monitorprogramm
  - 32 kB SRAM
  - LDC mit 4 Zeilen zu je 20 Zeichen
  - 2 serielle Schnittstellen
  - Portanschlüsse
  - 4 Potentiometer zur Generierung von Analogsignalen
  - .....
- Da beliebige Programme ausgeführt werden sollen,
  - müssen diese Programme ladbar sein,
  - muss ein Bereich des Programmspeichers im RAM liegen.

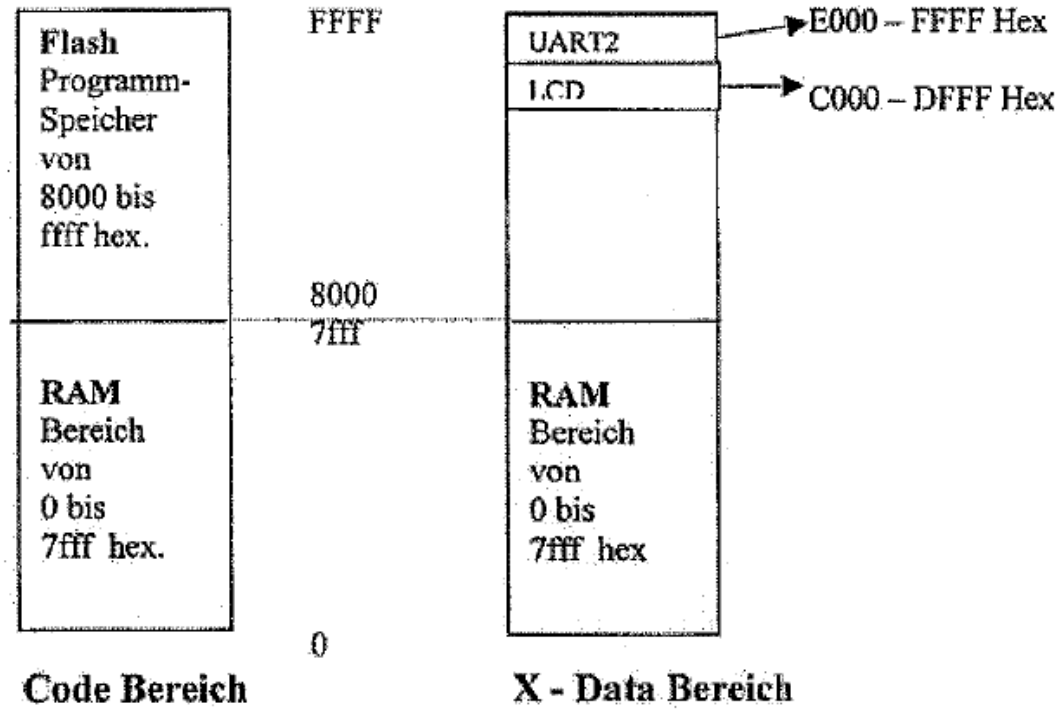
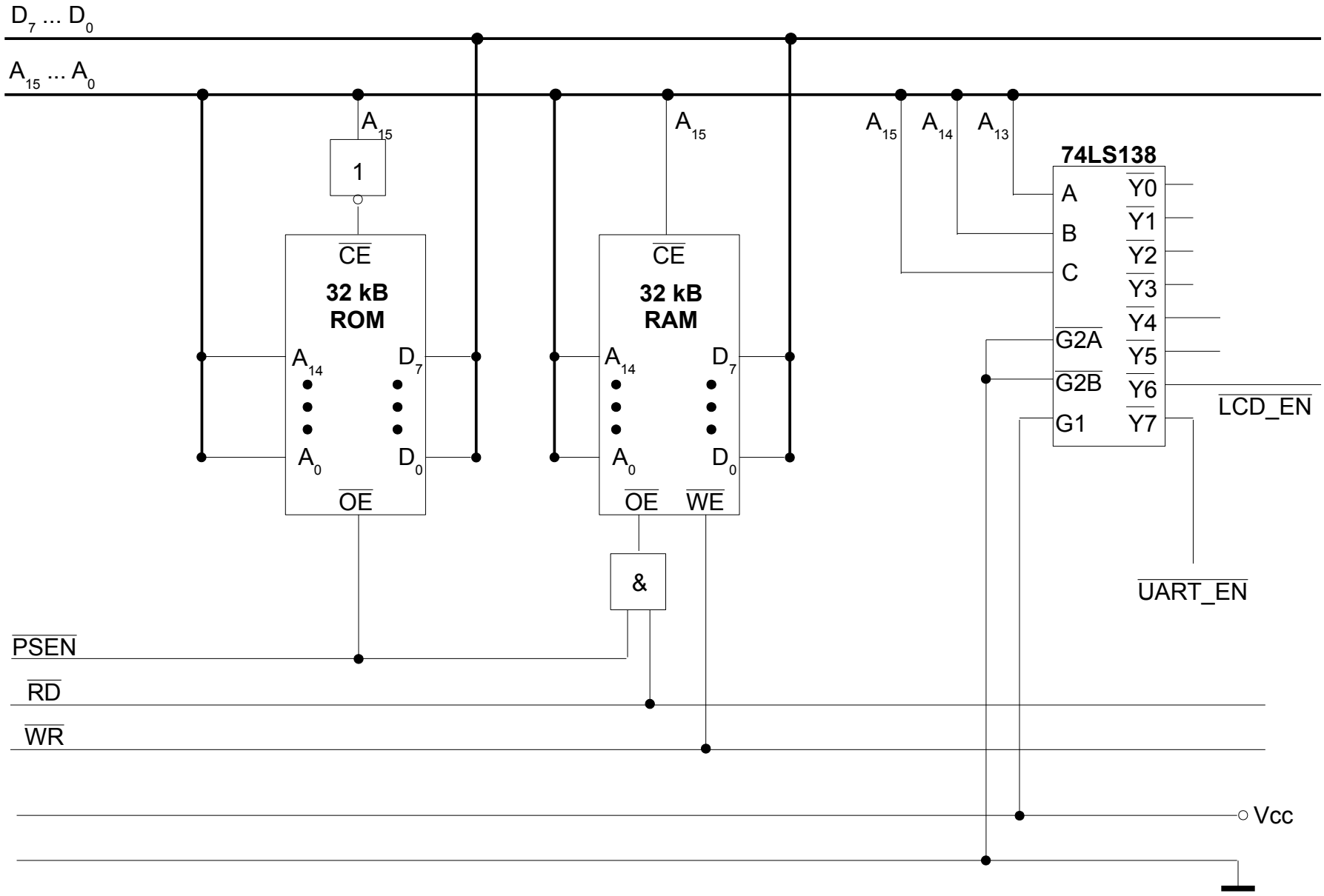


Bild 25: Externer Speicherplan 515

- Der RAM-Adressbereich 0x0000 ... 0x7FFF ist **gleichzeitig**
  - **Programmspeicher** für die Befehle bei der Ausführung des Testprogramms, und
  - **Datenspeicher**, damit das Monitorprogramm die Binärdaten, die das Testprogramm darstellen, dort hinein schreiben kann.

Systembusanschlüsse des Laborkoffers



Systembusanschlüsse des Laborkoffers (Forts.)

- Wenn A15 = 0 und ($\overline{\text{PSEN}}$ oder $\overline{\text{RD}}$) aktiviert sind, wird ein Byte aus dem RAM gelesen.
  - Der Prozessor aktiviert $\overline{\text{PSEN}}$ für einen Instruction Fetch oder einen MOVC Befehl.
  - Der Prozessor aktiviert $\overline{\text{RD}}$ für einen Lesebefehl MOVX A,@DPTR.
- Mit A15 = 0 und aktivierter $\overline{\text{WR}}$ Steuerleitung schreibt der Prozessor das Testprogramm byteweise in das RAM.
- Der Dekoder 74LS138 dient dazu die Chip Select Leitungen für die von dem LCD Baustein und der seriellen Schnittstelle reservierten Datenspeicherbereiche auszuwählen.
  - Dem LCD ist der Adressbereich 0xC000 ... 0xDFFF zugewiesen.
  - Der seriellen Schnittstelle ist der Adressbereich 0xE000 ... 0xFFFF zugewiesen.
  - Die Ausgänge $\overline{\text{Y4}}$ und $\overline{\text{Y5}}$ des Dekoders könnten dazu verwendet werden, die Chip Select Leitungen für die Adressbereiche 0x8000 ... 0x9FFF bzw. 0xA000 ... 0xBFFF auszuwählen.
  - Die Ausgänge $\overline{\text{Y0}}$ bis $\overline{\text{Y3}}$ können nicht verwendet werden, weil sonst eine Mehrfachadressierung mit dem RAM auftreten würde.
- Als alternative Lösung der obigen Schaltung könnte die Adressleitung A15 auch an die Chip Select Leitung des Dekoders G1 angeschlossen werden.

Komparatorbausteine

- Erlaubt die flexible Einstellung der Adressen, über die ein Speicherbaustein oder ein E/A-Baustein angesprochen werden kann.

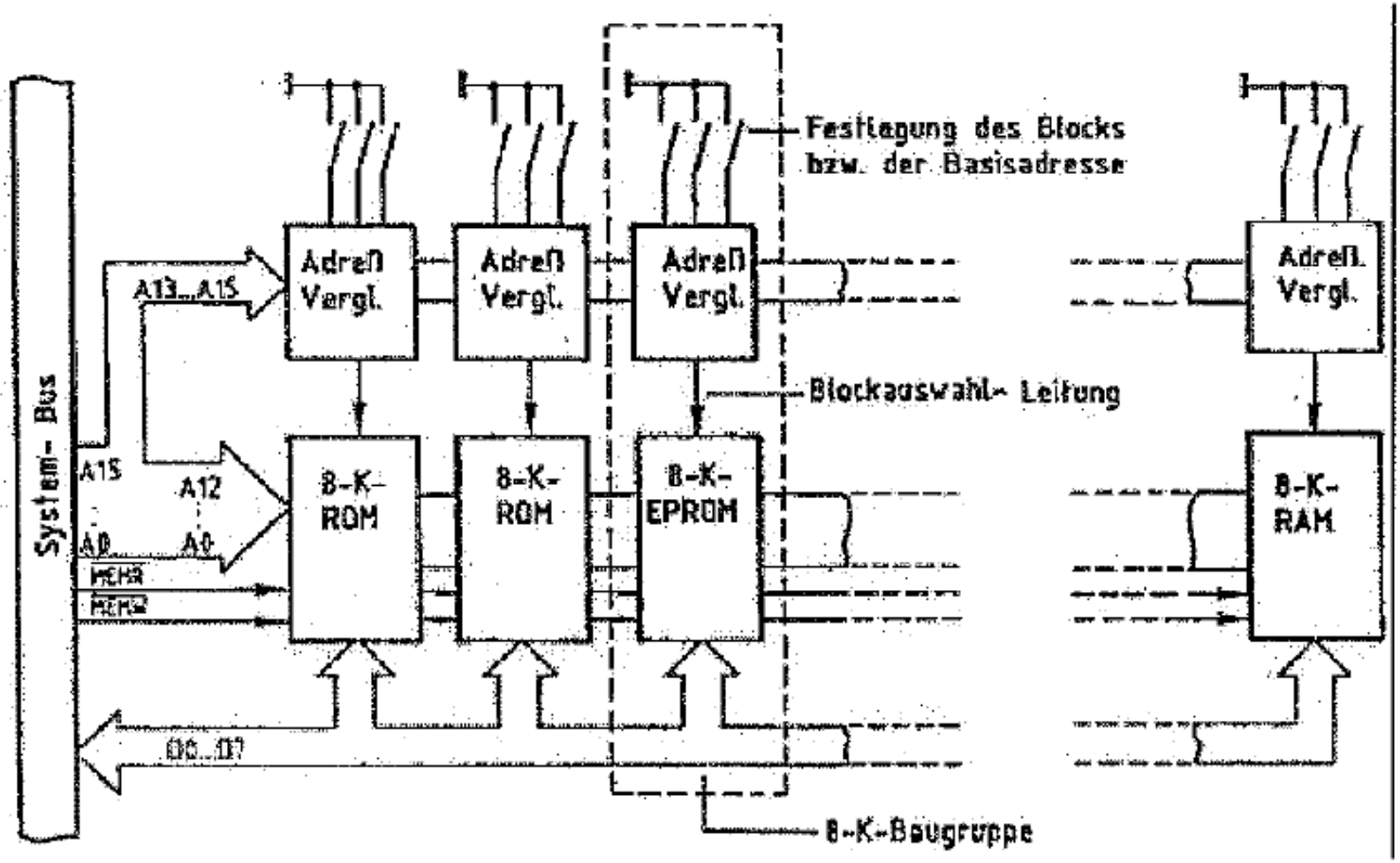
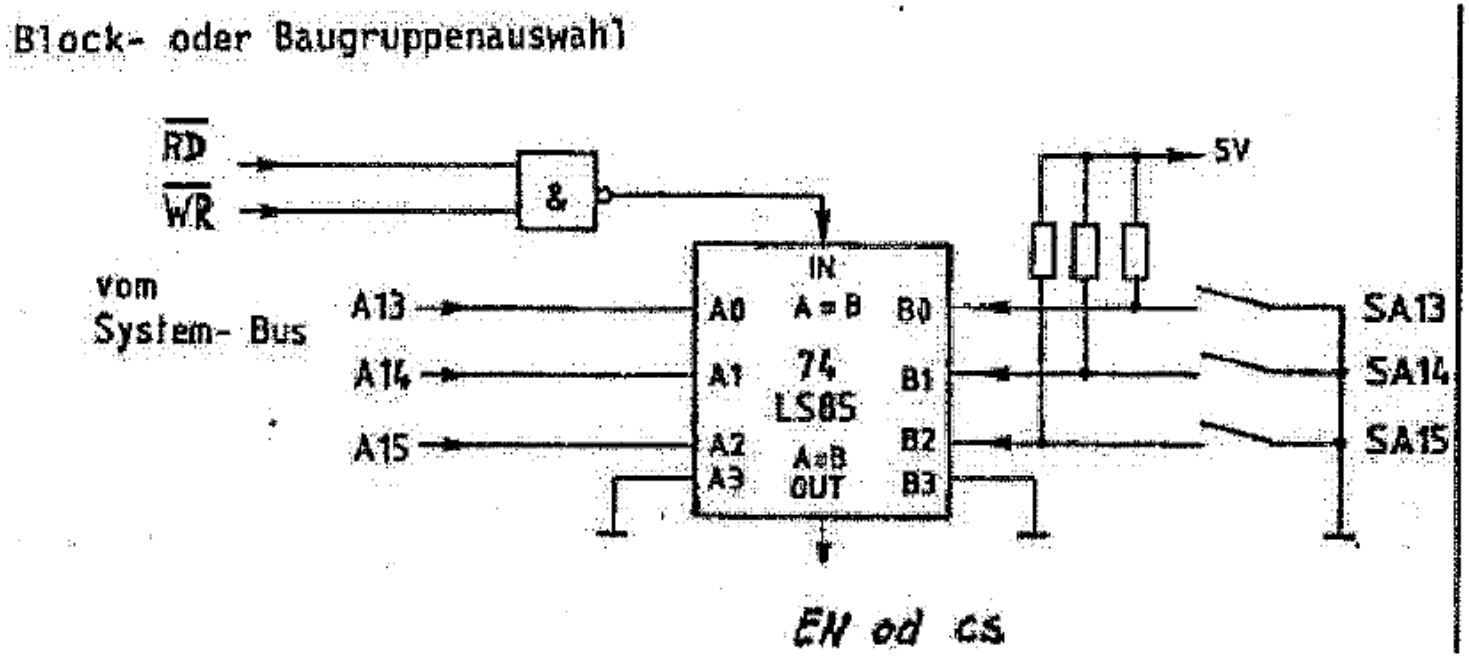


Bild 31: Speicher mit flexibler Adresse

Skript Bild 31, S.35 : Speicher mit flexibler Adresse

Komparatorbausteine (Forts.)

- Ein Komparatorbaustein vergleicht die angeschlossenen Adressbits (z.B. A0,A1, A2, A3) bitweise mit dem eingestellten Vergleichswert (z.B. B0, B1, B2, B3) und aktiviert eine ENABLE Leitung $\overline{OUT}$, falls die Bitpaare jeweils den gleichen Wert aufweisen, also falls (A0 = B0) & (A1 = B1) & (A2 = B2) & (A3 = B3) = 1 ist, **und** falls die eigene Enable-Leitung $\overline{IN}$ aktiviert ist.



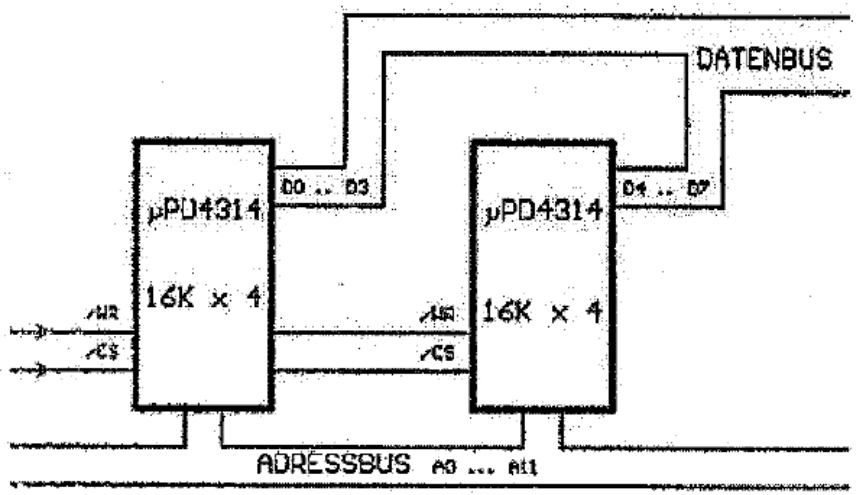
Skript Bild 32, S.36 :
Komparatorbaustein 74LS85

- Der Vergleichswert kann z.B. über Schalter eingestellt werden.
  - Ein über einen Pull-up-Widerstand auf '1' gesetzter Anschluss kann über einen an Masse angeschlossenen Schalter auf '0' gezogen werden.
- In heutigen Zeiten von Plug-and-Play werden Vergleichswerte häufig über Ausgangsportleitungen einstellbar gemacht.
- Komparatorbausteine und Dekoder können auch kombiniert werden
  - z.B. ein Dekoder wählt einen Adressbereich, in dem der Komparator ansprechbar ist ($\overline{Yn}$ verbunden mit $\overline{IN}$).

Speicherbausteine mit kürzeren Wortlängen

- Die Einheit der Datenübertragung bei einem Speicherzugriff ist das **Byte**.
- Die meisten produzierten Speicherchips besitzen nur **eine** Datenleitung.
- 8 Chips (z.B.: je 1 G * 1 Bit) werden auf eine kleine Platine montiert. So ergibt sich ein Speicherriegel, auf dem ganze Bytes adressiert werden können.
  (z.B.: 1 G * 8 Bits = 1 GB = 10^9 Bytes)
- Ein solcher Speicherriegel erscheint als Speicherbaustein mit einer Wortbreite von 8 Bits.
- Wenn ein Parity Bit pro Byte benötigt wird, werden einfach 9 Chips montiert.
- Neben regulären Speicherbausteinen mit einer Wortbreite von 8 Bits sind noch Bausteinen mit 4 Bits Wortbreite gebräuchlich.

- Um z. B. einen Speicher aus Bausteinen mit nur 4 Bits Wortbreitezu realisieren, müssen zwei Bausteine gleichzeitig angesprochen werden.
- Vom/zum ersten Baustein werden die Datenbits D7 ... D4 übertragen; entsprechend sind die vier Datenleitungen dieses Bausteins an die Datenbusleitungen D7 ... D4 angeschlossen.
- Vom/zum zweiten Baustein werden die Datenbits D3 ... D0 übertragen; entsprechend sind die vier Datenleitungen dieses Bausteins an die Datenbusleitungen D3 ... D0 angeschlossen.
- Die Adressbus-, Steuerbus- und Chip-Select-Anschlüsse sind für beide Speicherbausteine gleich.
- Siehe auch Skript Bild 30, S.35 : Adressbus A0 … A11
  (Die Größenangabe in den Speicherbausteinen 16k * 4 ist falsch. Wie muss sie richtig lauten ?)



Skript Bild 30, S.35 :
Erweiterung der Wortlänge auf 8 Bits