

P. E. Melsen & B. Thestrup

***micro  
computer  
hardware  
teori***

Simonsen & Weel's Eftf. A/S

**S&W**



# microcomputer hardware teori



## FORORD

Denne bog er udarbejdet som støttemateriale til et 40 timers kursus i  
MICROCOMPUTER HARDWARE

Bogen er skrevet af B. Thestrup og P.E. Melsen, Elektronikafdelingen, Århus tekniske Skole i samarbejde med C. Melgaard og P. Schmidt-Andersen, Simonsen & Weel's Eftf. A/S.


Det er bogens formål at beskrive en lang række af de komplicerede LSI komponenter, der indgår i et microcomputer system. Det er forfatterens opfattelse, at man først efter erhvervelsen af en tilbundsgående forståelse af hardware funktioner, vil være i stand til at programmere og fejlfinde microcomputer systemer effektivt.


Derfor er de første seks kapitler helliget generelle hardware begreber, mens de sidste kapitler beskriver de kredsløb, der specielt benyttes i MOTOROLA's microcomputer systemer. Yderligere findes en beskrivelse af det udstyr, der benyttes til udvikling og fejlfinding på microcomputer kredsløbet.


Bogen rundes af med et praktisk eksempel på sammenknytningen af hardware og software. Dette eksempel på analog til digital konvertering er hentet fra S&W's blodtryksmodul "PRESS 8041".


Illustrationerne til bogen er fortrinsvis hentet fra en række halvlederfirmaers datablade, application notes, etc.. De lånte tegninger er forsynet med de pågældende firmaers bomærke.

Vi ønsker at rette en tak til disse firmaer og især til:

Hewlett-Packard A/S, Birkerød 

Intel Danmark A/S, København 

Motorola Semiconductors, Lyngby 

Texas Instruments A/S, Herlev 

Århus, 1. November 1979

P.E. Melsen  
B. Thestrup

Albertslund, 1. November 1979

C. Melgaard  
P. Schmidt-Andersen



© 1979 - Simonsen & Weel's Eftf. A/S, Albertslund  
Printed in Denmark 1979.

Mangfoldiggørelse af denne bog - helt eller delvis - er ifølge loven om ophavsret forbudt uden ophavsmandens (Simonsen & Weel's Eftf. A/S, Albertslund) tilladelse. Dette gælder enhver form for mangfoldiggørelse, f.eks. fotokopiering, eftertryk, microfotografering etc.



## INDHOLDSFORTEGNELSE

Talsystemer .....	Kapitel 1
Micro-, Mini- og Macro-computere.....	Kapitel 2
Microcomputer lageret.....	Kapitel 3
Programmering.....	Kapitel 4
Programudførelse .....	Kapitel 5
Input/Output kontrolmetoder....	Kapitel 6
6800 familiens hardware.....	Kapitel 7
Udviklingsudstyr .....	Kapitel 8



## DECIMALTALLENE

Det mest kendte talsystem er talsystemet eller decimaltalsystemet. De ti taltegn, vi bruger i decimalsystemet, er:

0 – 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9

Hvis et decimaltal består af mere end et ciffer, f.eks. 11, betyder det egentlig  $10 + 1$

$$11 = 1 \times 10 + 1$$

På samme måde betyder 72 syv tiere plus to enere

$$72 = 7 \times 10 + 2$$

og tallet 3745 betyder tretusinder plus syv hundreder plus fire tiere plus fem enere

$$3745 = 3 \times 1000 + 7 \times 100 + 4 \times 10 + 5 \times 1$$

eller med potenser af 10

$$3745 = 3 \times 10^3 + 7 \times 10^2 + 4 \times 10^1 + 5 \times 10^0$$

Decimalsystemet siges at have Base = 10. Det vil sige, at værdien af det enkelte tal ikke blot bestemmes af cifferet selv, men også af den plads, det indtager. Dette er ikke noget enestående for talsystemet, men gælder for alle positionstalssystemer.

## BINÆRE TAL

I logiske kredsløb arbejder vi med to niveauer: High (HI) og Low (LO). Hvis vi kalder HI = 1 og LO = 0, vil det falde naturligt at benytte et talsystem med kun to taltegn, nemlig 0 og 1. Totalsystemet eller det binære talsystem er opbygget efter samme princip som talsystemet, blot er systemets basis nu 2.

Eks.

$$1011 = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$$

Udtrykt generelt: antag at et hvilket som helst talsystems cifre repræsenteres symbolsk af  $C_d, C_c, C_b, C_a$  og hvis B repræsenterer talsystemets base, kan et hvilket som helst tal udtrykkes med ligningen:

$$C_d C_c C_b C_a = C_d \times B^3 + C_c \times B^2 + C_b \times B^1 + C_a \times B^0$$

som f.eks. med et decimaltal (Base = 10)

$$2139 = 2 \times 10^3 + 1 \times 10^2 + 3 \times 10^1 + 9 \times 10^0$$

eller et binært tal (Base = 2)

$$1101 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$$



**Omregning mellem binærtal og decimaltal**

Haves et binært tal og ønskes den tilsvarende decimale værdi, kan denne findes ved at multiplicere de enkelte cifre med deres vægt og derefter addere disse produkter som vist:

$$\begin{array}{cc}
 \text{MSD} & \text{LSD} \\
 \downarrow & \downarrow \\
 101001_{(2)} & = 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 \\
 & = 32 + 0 + 8 + 0 + 0 + 1 = 41_{(10)}
 \end{array}$$

Nogle forkortelser, der er meget anvendt ved talsystemer, er:

MSD = Most Significant Digit  
 = Mest betydende ciffer  
 (kaldes også MSB = Most Significant Bit)

LSD = Least Significant Digit  
 = Mindst betydende ciffer  
 (kaldes også LSB = Least Significant Bit).

**Omregning fra decimaltal til binærtal**

Skal et decimaltal (base = 10) omregnes til et binært tal, sker dette ved at dividere det decimale tal med 2, resultatet af denne division divideres med 2, o.s.v., indtil tallet er reduceret til 0. Ved hvert trin ses der bort fra resten, som giver værdien af det binære digit:

$$\begin{array}{rcl}
 29_{(10)} \Rightarrow & \begin{array}{l} 29 : 2 = 14 \\ 14 : 2 = 7 \\ 7 : 2 = 3 \\ 3 : 2 = 1 \\ 1 : 2 = 0 \end{array} & \begin{array}{cc} \text{Rest} & \\ 1 & \text{LSD} \\ 0 & \\ 1 & \\ 1 & \\ 1 & \text{MSD} \end{array} \Bigg\} 11101_{(2)} \\
 & 29_{(10)} = 11101_{(2)} &
 \end{array}$$

Index (10) og (2) benyttes til at indikere, at tallenes basis er henholdsvis 10 og 2. Dette kan være nødvendigt, når der arbejdes med forskellige talsystemer.

**HEXADECIMALE TAL**

Ved skrivning af binære tal er det tit nødvendigt at anvende en hel del cifre, f.eks.  $35_{(10)} = 100011_2$ . Det er derfor let at lave fejl, når man skal skrive lange binære tal. For at forenkle skrivning og læsning af binære tal kan man anvende det hexadecimale talsystem. Basen for dette er 16 og de cifre, som anvendes, er:

0 – 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 – A – B – C – D – E – F

Da man skal have 16 ciffersymboler, må man tage bogstaverne A, B, C, D, E og F i brug. I tabellen herunder vises de hexadecimale tal sammenlignet med de øvrige talsystemer:



Decimal	Binary	Hexadecimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Decimal	Binary	Hexadecimal
16	10000	10
17	10001	11
18	10010	12
19	10011	13
20	10100	14
...	...	...
...	...	...
...	...	...
...	...	...
...	...	...
32	100000	20
50	110010	32
60	111100	3C
64	1000000	40
100	1100100	64
255	11111111	FF

Fig. 1.1

Fra binær til hexadecimal:

For at omskrive et binært tal til et hexadecimalt deler man det binære tal ind i 4-bitsgrupper fra højre mod venstre. Hver 4-bitgruppe svarer til et hexadecimalt ciffer, som kan ses i ovenstående tabel.

Eks.

10011011      Binært tal  
 1001 1011      4-bitsgrupper  
 9      B      tilsvarende hexadecimalt ciffer

d.v.s.  $1001\ 1011_{(2)} = 9\ B_{(16)}$

Fra hex. til binær:

Det hexadecimal tal omskrives til et binært tal ved at tage hvert ciffer i hex-tallet og skrive dette som et fire-bit binært tal:

$$A \ 2 \ C \ 7_{(16)} = \begin{array}{cccc} 1010 & 0010 & 1100 & 0111_{(2)} \\ A & 2 & C & 7 \end{array}$$

d.v.s.

$$A \ 2 \ C \ 7_{(16)} = 1010 \ 0010 \ 1100 \ 0111_{(2)}$$

## ADDITION

Addition af binære tal sker på samme måde som med decimaltal. Mente ved decimaltal fås, når summen er ti eller mere, medens mente ved binæraddition fås allerede, når summen bliver to ( $= 1 + 1$ )

Der gælder følgende additionsregler:

	SUM	MENTE
$0 + 0$	$= 0$	
$0 + 1$	$= 1$	
$1 + 0$	$= 1$	
$1 + 1$	$= 0$	1

Eks.

Binært	Decimalt
$\begin{array}{r} 1 \\ + \ 1 \\ \hline 10 \end{array}$	$\begin{array}{r} 1 \\ + \ 1 \\ \hline 2 \end{array}$
$\begin{array}{r} 111 \\ + 101 \\ \hline 1100 \end{array}$	$\begin{array}{r} 1 \\ 7 \\ + \ 5 \\ \hline 12 \end{array}$

## SUBTRAKTION

Binære tal kan subtraheres på samme måde som de decimale tal. To binære cifre subtraheres efter følgende regler:

$0 - 0 = 0$	
$0 - 1 = 1$	idet der „lånes” for at subtrahere
$1 - 0 = 1$	
$1 - 1 = 0$	



Eks.

$$101100 - 11000 = 10100$$

$\begin{array}{r} \overset{\{1\}}{\underset{\{1\}}{01100}} \\ - 011000 \\ \hline 10100 \end{array}$	<p>Decimalt</p> $\begin{array}{r} 44 \\ - 24 \\ \hline 20 \end{array}$
---	--

Her ses, at „låne” 1 giver to 1-taller i det næste mindre betydende bit. At „låne” kaldes på engelsk BORROW.

Sædvanligvis er de aritmetiske enheder konstrueret således, at de kan addere, men ikke subtrahere. Man må derfor ændre fortegnet på det tal, som skal subtraheres og derefter udføre subtraktionen som en addition af et negativt tal. Det er da vigtigt at kunne skrive negative binære tal.

Den måde, man normalt anvender ved repræsentationen af negative tal, kan sammenlignes med kilometertælleren i en bil. Hvis bilen køres fremad, udfører kilometertælleren en addition, medens der udføres en subtraktion, hvis bilen bakker. Lad os antage, at kilometertælleren står på nul, når bilen begynder at bakke. Den kommer da efterhånden til at vise følgende:

00000  
99999  
99998  
99997

99997 svarer til, at bilen har bakket 3 kilometer (d.v.s.  $-3$  km). Hvis bilen derefter kører 5 km fremad, så adderes disse til 99997:

00005  
+ 99997  
-----  
100002

hvis det venstre 1-tal udelades (det ses jo heller ikke på kilometertælleren), bliver resultatet 2,

d.v.s.

$$5 + (-3) = 2$$

Tallet 99997 kaldes for 10-komplementet til tallet 3. Decimalt kan man altså repræsentere negative tal ved hjælp af 10-komplementet.

I det binære talsystem kan man gøre på tilsvarende måde, men i stedet for 10-komplementet anvender man 2-komplementet. 2-komplementtallene kan fås på samme måde som for de decimale tal. Man skal blot forestille sig en binær „kilometertæller”.

00000000  
11111111  
11111110  
11111101

11111101 svarer da til  $-3_{(10)}$ , så hvis vi laver samme regnstykke som før, får vi

$$\begin{array}{r}
 \begin{array}{r}
 11111 \\
 0000101 \\
 + 1111101 \\
 \hline
 \end{array}
 \begin{array}{r}
 \text{Dec.} \\
 5 \\
 + (-3) \\
 \hline
 2
 \end{array}
 \end{array}$$

falder bort  $\rightarrow$  10000010 2

En anden måde at danne 2-komplementtal på er følgende:

1. Først tages tallets 1-komplement, som fås ved at invertere alle bits

Eks.  $\begin{array}{r} 0000011 \\ 1111100 \end{array} \begin{array}{l} 3_{(10)} \\ \text{1-komplement til 3} \end{array}$

2. Derefter dannes 2-komplementet ved at addere 1 til 1-komplementet:

$$\begin{array}{r}
 1111100 \quad \text{1-komplement til 3} \\
 + \quad 1 \\
 \hline
 1111101 \quad \text{2-komplement til 3}
 \end{array}$$

I microprocessorer arbejder man normalt med 8-cifrede binære tal. 2-komplementtallene kommer da til at se ud som vist:

8-BIT 2's COMPLEMENT		
Decimal	Binary	Hexadecimal
+127	01111111	7F
⋮	⋮	⋮
+ 64	01000000	40
⋮	⋮	⋮
+ 2	00000010	02
+ 1	00000001	01
0	00000000	00
- 1	11111111	FF
- 2	11111110	FE
⋮	⋮	⋮
- 64	11000000	C0
⋮	⋮	⋮
-127	10000001	81
-128	10000000	80

Fig. 1.2

Læg mærke til, at det mest betydende bit i 2-komplementtallene er „1” for negative tal og „0” for positive tal (fortegns bit).

Ved 2-komplement tal er de positive tal de samme som de tilsvarende binære tal. Det er kun de negative tal, der skal omskrives.

Bemærk endvidere, at når der arbejdes med 8-bit 2-komplementtal, kan man angive værdier i intervallet  $-128_{10} \rightarrow +127_{10}$ .

Hvis der bruges 8-bit binære tal (uden fortegn), er intervallet  $0 \rightarrow 255_{(10)}$ .

De enkelte cifre i et binært tal kaldes bits (binary – digits = binære cifre). En sammenstilling af bits til et binært tal kaldes ofte et ORD.

For eksempel kaldes det binære tal:

01100100<sub>(2)</sub>

et 8-BIT-ORD.

Da logiske kredsløb til behandling af binære tal kun kan arbejde med ORD af en bestemt størrelse, må der ofte tilføjes indledende nuller.

I computere og andre kredsløb anvendes ofte ord med 4, 8 eller 16 bits. Disse ordstørrelser har derfor fået specielle navne:

4-BIT-ORD = NIBBLE

8-BIT-ORD = BYTE

16-BIT-ORD = WORD

De første microcomputere arbejdede med DATAORD på 4 bit og kaldes af denne grund for 4-bit systemer. Disse blev anvendt omkring 1970–73 f.eks. INTEL's typer 4004 og 4040.

Udviklingen har medført, at DATAORDENE skulle være større for at kunne indeholde flere oplysninger og for at gøre systemerne bedre egnede til at løse mere indviklede problemer. Det mest udbredte system idag arbejder med DATAORD på 8 bits. Microprocessorer af denne størrelse findes i et stort antal forskellige fabrikater og typer med hver deres specielle egenskaber. Som eksempel kan der her nævnes nogle af de mest anvendte: INTEL's 8080, 8085, 8048 – MOTOROLA's 6800 – ZILOG's Z80 o.s.v.

Kravene til microcomputersystemerne og anvendelsesmulighederne har medført, at 16-bits systemer er begyndt at dukke op, og af disse er den mest kendte idag TEXAS 9900-serie og INTEL's 8086.



## **MICRO-, MINI-, MACRO-COMPUTERE**

### **MACRO**

Man deler ofte computerne op i tre grupper efter størrelse og hastighed.

De største computere er dem, der benyttes til bogholderiopgaver og lignende store administrative opgaver; de kaldes macrocomputere. Deres lager består af et „mindre” arbejdslager RAM eller kernelager og et båndlager, som kan være fordelt på flere båndstationer, og fylder derfor en del. Macrocomputere har en meget hurtig CPU, hvilket er nødvendigt, da der er mange data, som skal behandles.

### **MINI**

Den lidt mindre computer, Minicomputeren, fylder ikke nær så meget som en macrocomputer. Minicomputerens lager er normalt RAM lager, som indeholder både program og data. Minicomputere bruges både til mindre administrative opgaver og til styringsopgaver.

### **MICRO**

I modsætning til de to andre grupper fylder en microcomputer ikke ret meget (nogle få DIPs). Dens anvendelsesområde er normalt „mindre” styringsopgaver. De hurtigste og største microcomputere er ved at nærme sig minicomputerens formåen.

Microcomputerens program er lagret i ROM og er beregnet til en bestemt styringsopgave. Hvis man tænker sig, at en microcomputer anvendes til styring af en vaskemaskine, vil det være meget uheldigt at anbringe programmet i et RAM lager, da det så vil forsvinde, når der slukkes for maskinen.

Microcomputeren erstatter idag en stor del af den traditionelle elektronik, ligesåvel som den finder anvendelse på næsten alle områder, som involverer programmering eller automatisk kontrol. Den har to enestående fordele:

#### **1. Færre komponenter**

Det forholdsvis lille antal komponenter, som kræves af et microcomputersystem, medfører en del fordele

- mindre størrelse
- mindre effektforbrug
- mindre effekttab
- det mindre antal forbindelser giver større pålidelighed
- lavere omkostninger
- lettere at servicere og udføre ændringer på.

#### **2. Programmerbar**

Den største fordel ved programmeringen er, at den medfører et simplere design og reducerer udviklingstiden.

Endvidere resulterer programmering i standardiserede hardware moduler. Et standard microcomputersystem kan således programmeres til at udføre et stort antal forskellige opgaver. At erstatte et program med et andet kræver måske ingen hardware-mæssige forandringer, men simpelthen blot, at en program memory kredsløb erstattes med en anden. Et produkt kan således udvikles hurtigt, afprøves i „marken” og afpudses efterhånden uden behov for konstruktionsændringer i hardwaren.

## MICROCOMPUTER

En microcomputer består af følgende enheder, når der er tale om et minimumssystem:

1. CENTRAL PROCESSING UNIT (CPU) (også benævnt micro-processor eller MPU).
2. LAGER (hukommelse, memory)
3. INPUT-OUTPUT-ENHED

for dette kan følgende blokdiagram opstilles:

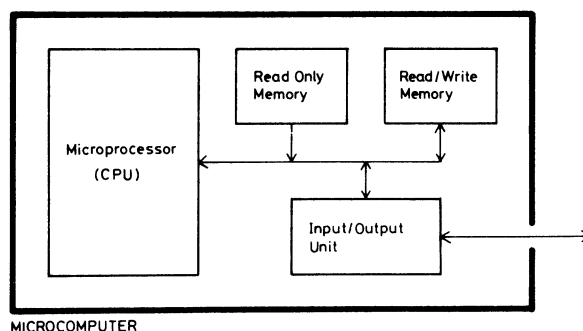


Fig. 2.1

1. CPU'en er den centrale enhed eller „hjernen” i systemet. Denne enhed STYRER og BEHANDLER det, der skal foretages i microcomputer.
2. LAGERET benyttes til opbevaring af de binære informationer, som CPU'en skal benytte sig af. Lageret er opdelt i to enheder, et læse-lager (ROM), hvorfra der kun kan hentes informationer (her findes programmet) og et læse-skrive-lager (RAM), hvori der kan gemmes informationer, som senere skal bruges igen (normalt de data, som skal behandles).
3. INPUT-OUTPUT ENHEDEN benyttes til at skabe kontakt med „den ydre verden” udenfor microcomputeren.

Over denne enhed tilføres computeren de DATA, der ønskes behandlet. Efter endt behandling afleverer computeren RESULTATERNE via input-output-enheden.

Indeni microcomputeren er de enkelte enheder forbundet ved hjælp af et BUS system.

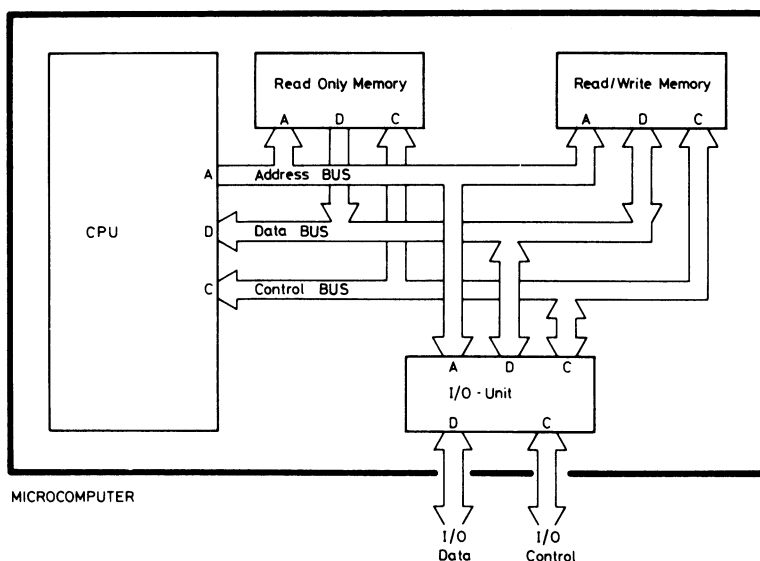


Fig. 2.2

Ordet BUS angiver, at der er tale om signalledninger, der benyttes i fællesskab af forskellige enheder. De tre BUSSE, som findes i ethvert microcomputersystem, er:

1. **DATABUSSEN**, som sender data frem og tilbage mellem systemets forskellige enheder. For eksempel bringer den data fra input-output enheden til microprocessoren og bringer data tilbage fra microprocessoren til hukommelsen (memory). I standard microcomputeren er det en 8-bit bi-directional bus (d.v.s. den kan bruges i begge retninger). Det er næsten altid en tri-state bus, så den kan bruges sammen med DMA (Direct-Memory-Access).
2. **ADRESSEBUSSEN** udgår fra microprocessoren og har forbindelse med alle de øvrige kredse i minimumsystemet. Adressebussen er normalt en 16-bit bus, som giver mulighed for  $2^{16} = 65.536$  (64k) adresser. En adresse på adressebussen udvælger en enhed (en chip) og udpeger et register i enheden. Adressebussen bliver altid brugt samtidig med databussen til at udpege kilden eller destinationen for de data, som sendes på databussen.
3. **CONTROLBUSSEN** fører synkroniseringssignaler fra de andre enheder til microprocessoren eller fra microprocessoren til de andre systemkomponenter. Af typiske signaler på controlbussen kan nævnes: read, write, interrupt og reset.

Hver af de tre busser kan normalt drive 5–8 systemkredse. I de fleste systemer, undtagen de mindste, er det derfor nødvendigt at indskyde buffere (busdrivere) i busserne for at kunne forbinde et tilstrækkeligt antal ydre kredse.

## 2.7

I de fleste microcomputere af idag findes de enkelte enheder (ROM, RAM og I/O) som separate DIP's (Dual Inline Package). Dette kan f.eks. ses af nedenstående diagram af et Motorola minimum-system.



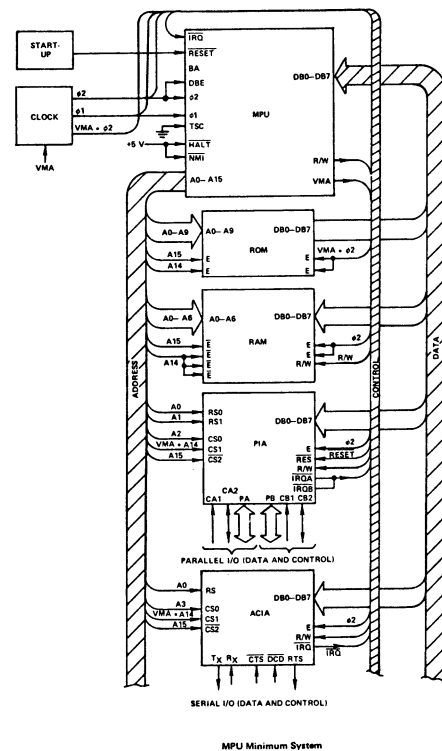


Fig. 2.3

Som det ses af diagrammet skal der anvendes et vist minimum antal kredse for at danne et helt system. Udviklingen har heldigvis medført, at flere og flere af de nødvendige funktioner er blevet samlet på en og samme chip. Således har de forskellige fabrikater fået hver deres ONE CHIP MICROCOMPUTER.

Her ses Motorolas MC 6801

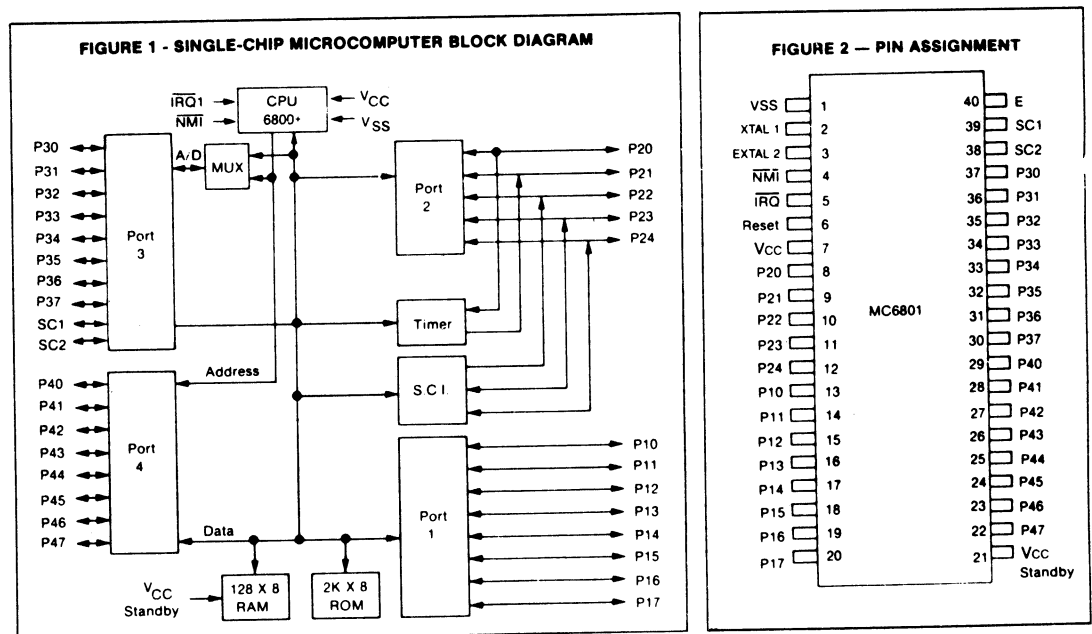


Fig. 2.4

Kan man nøjes med 2k bytes ROM og 128 bytes RAM lager, så består microcomputeren kun af denne ene DIP.

De benforbindelser, som før blev anvendt til adresse- og databus (16 + 8), kan nu benyttes til input-output porte.

I fremtiden vil man, eftersom udviklingen skrider hurtigt fremad, kunne få større og større computere på kun een DIP.

## LAGERET (MEMORY)

Microprocessor lageret kan opdeles i to hovedtyper af hukommelses-elementer. Et læse-skrivelager (RAM), hvis indhold både kan skrives og læses. Den største ulempe ved RAM-lagre er, at de er volatile, d.v.s. hvis forsyningsspændingen til RAM-lageret forsvinder, forsvinder det binære indhold også. Derfor bruges RAM også sjældent som program memory. Hvis forsyningsspændingen skulle svigte, ville det være nødvendigt at læse programmet ind igen, før computeren igen kunne sættes igang. Derfor bruges RAM til at opbevare data, så som måleresultater eller mellemresultater, hvis tab vil være uden betydning ved strømsvigt.

Den anden type af hukommelselementer er ROM, hvis indhold kun kan læses. Når engang indholdet af en ROM er fastlagt i den sidste del af fabrikationsprocessen, kan det ikke længere ændres. En ROM er non-volatile, d.v.s. at den beholder sit indhold også efter, at spændingen har været afbrudt.

Et memory-system kan, uanset type (RAM, ROM), tegnes som en MEMORY og nogle ydre styreenheder.

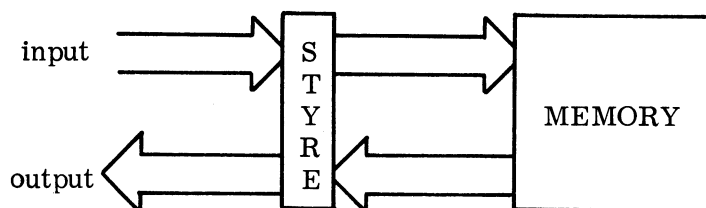


Fig. 3.1

Selve MEMORY'en er opbygget som et matrix-system, hvor hvert „krydspunkt” er i stand til at opbevare en binær information, f.eks. som vist nedenfor:

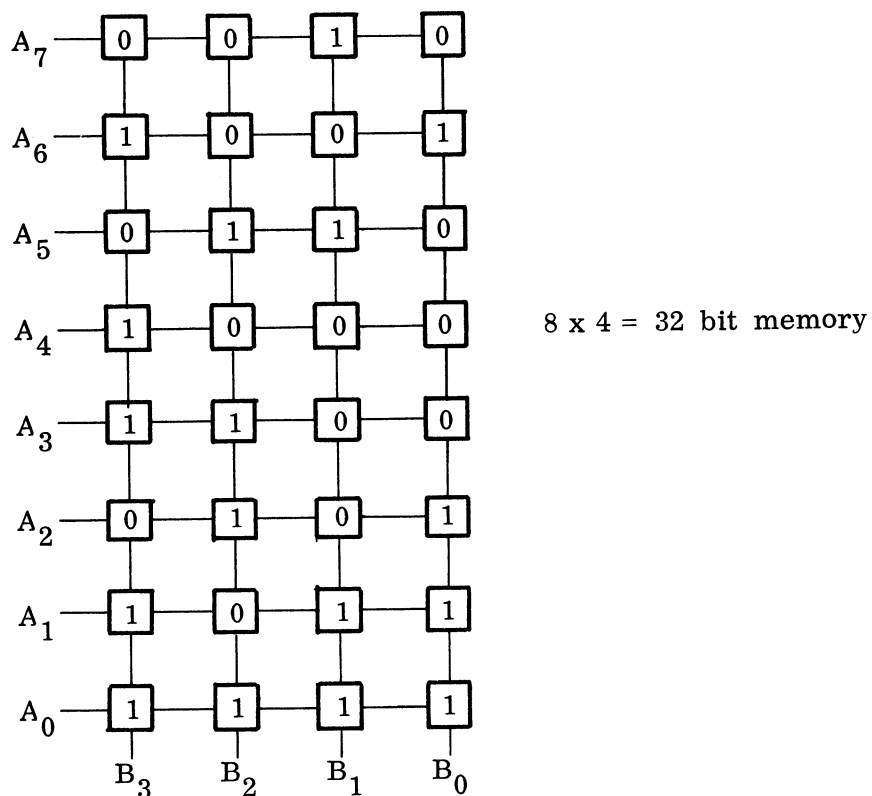


Fig. 3.2

I en ROM består styrekredsen af en adresse-dekoder, og output kommer ofte direkte fra memory (evt. via buffer/drivere).

Som eksempel kan anvendes den på foregående side viste memory, hvor A-terminalerne er de dekodede adresse-indgange, og B-terminalerne er output.



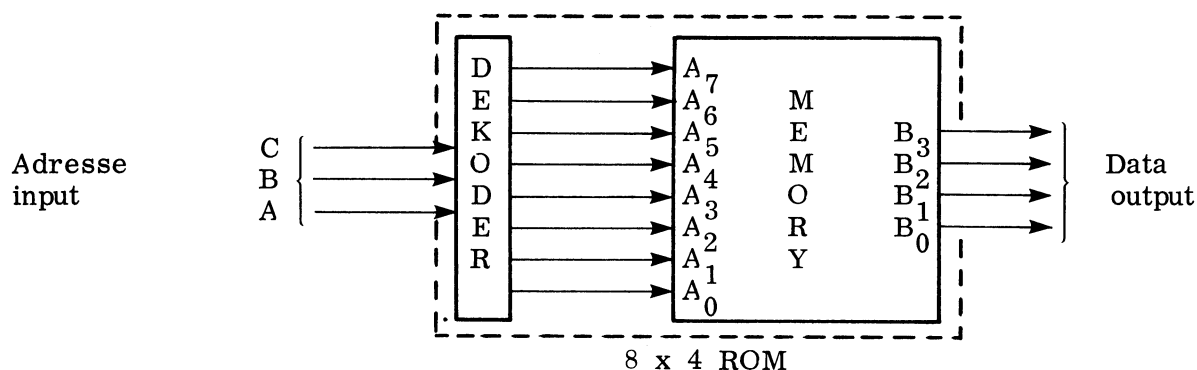


Fig. 3.3

Er f.eks. CBA = 101, kan dette dekodes og aktivere  $A_5$ . Herefter udlæses memory-indholdet på denne adresse på B-terminalerne, d.v.s.  $B_3 = 0$ ,  $B_2 = 1$  og  $B_0 = 0$  eller blot B = 0110.

Ændres nu adresse-input til CBA = 001, aktiveres  $A_1$ , og herved fås B = 1011.

En ROM betegnes ved antal adresse-input EFTER DEKODNING og antal outputs, hvilket også giver memorys matrix-dimension, f.eks. som ovenfor ved

$$8 \times 4 = 32 \text{ bits}$$

Selve memory-matricen kan være opbygget som en modstands diode matrix eller som en transistor matrix (MOS eller bipolar).

Eksempler på kredsløb i det følgende vises med diodematrix.

Et eksempel er en  $4 \times 4 = 16$  bits ROM ("hjemmestrikket" til formålet), der kan danne  $A \oplus B$ ,  $\overline{A \oplus B}$ ,  $\overline{AB}$  og  $\overline{A + B}$ .

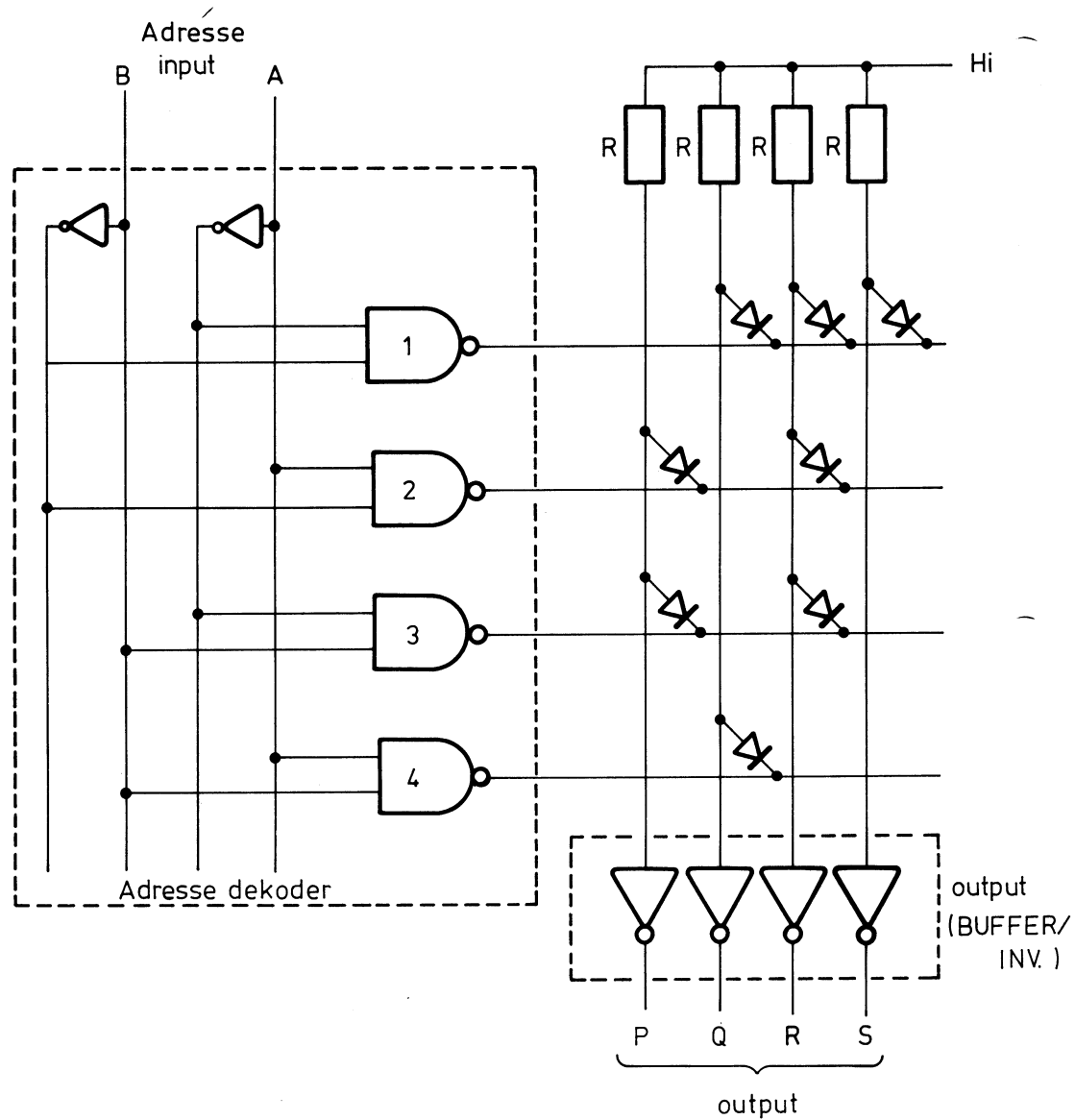


Fig. 3.4

Med den viste diodematrix (ROM-memory) fås:

Input		Adresse decoder NAND=LOW	output			
B	A		P	Q	R	S
0	0	1	0	1	1	1
0	1	2	1	0	1	0
1	0	3	1	0	1	0
1	1	4	0	1	0	0

Af ovenstående S-tabeller ses:

$$P = A \oplus B$$

$$Q = \overline{A \oplus B}$$

$$R = \overline{A \cdot B}$$

$$S = \overline{A + B}$$

Kredsløbets virkemåde kan kort beskrives ved:

Til en bestemt kombination af A og B fås LOW på EEN af adresse-dekoderens udgange.

Dette LOW vil få dioderne, der er tilsluttet denne NAND udgang, til at lede.

Herved bliver der LOW på de „lodrette” ledere, der fører til output buffer/inverterne. Output bliver derfor HI (1) for „aktive” dioder, og LO (0) for ledere, der „mangler” dioder.  
(NB: der er INGEN forbindelse i Matricens krydspunkter).

Der eksisterer idag 3 hovedtyper af ROMs: ROM, PROM og EPROM.

## ROM

En ROM er en maskeprogrammeret hukommelse, som kun kan fremstilles af fabrikanten. Det bitmønster, som ønskes, skrives op i en sandhedstabel. Når fabrikanten modtager denne, kan han fremstille den maske, der skal anvendes i det sidste trin i fabrikationen af en ROM. Ved det sidste trin etableres nemlig forbindelsen mellem række og søjle i nogle af memory matrixens krydspunkter således, at det ønskede bit-mønster opnås. Da der er store omkostninger forbundet med at fremstille en maske, vil ROMs kun blive brugt, når der er tale om store styktal (typ. > 1000).

## PROM (Programmable Read Only Memory)

Dette er en ROM, som kan programmeres af brugeren med en speciel PROM-programmer. Det er en såkaldt fusable-link. Hver hukommelsescelle er fremstillet med en „sikring”. Under programmeringen vil PROM-programmeren generere nogle pulser, som brænder nogle af disse „sikringer” af. Det tager kun nogle få minutter at „brænde” en PROM og sætte den i udstyret.

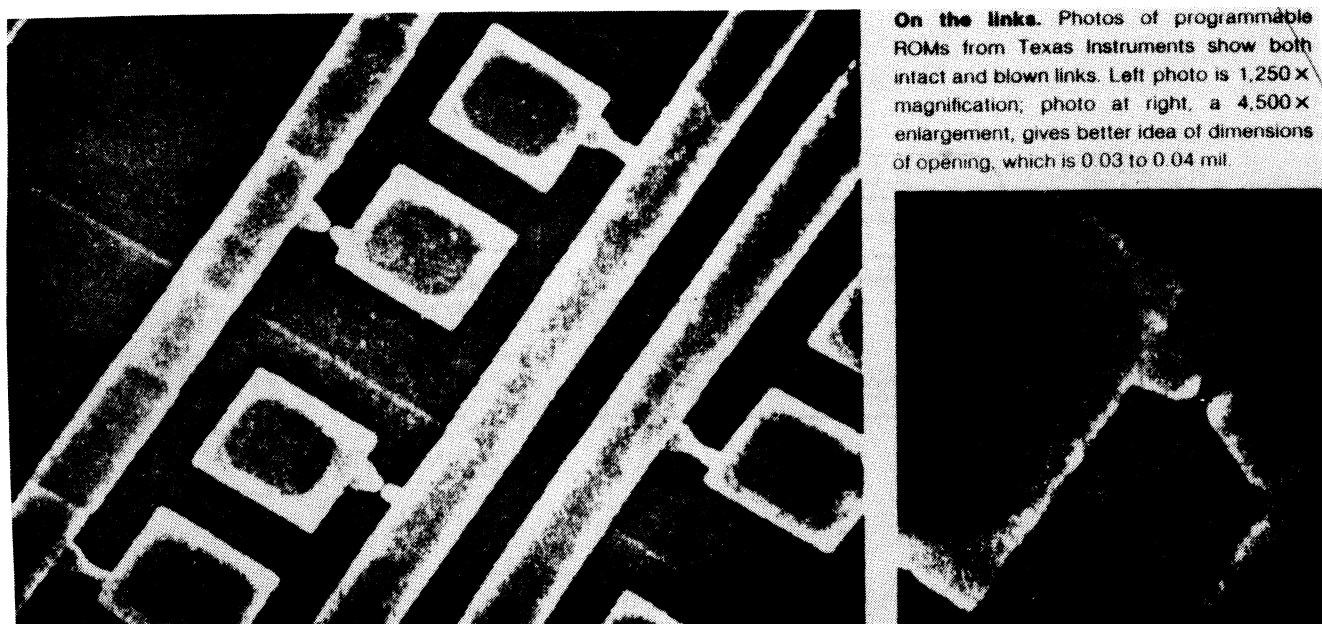
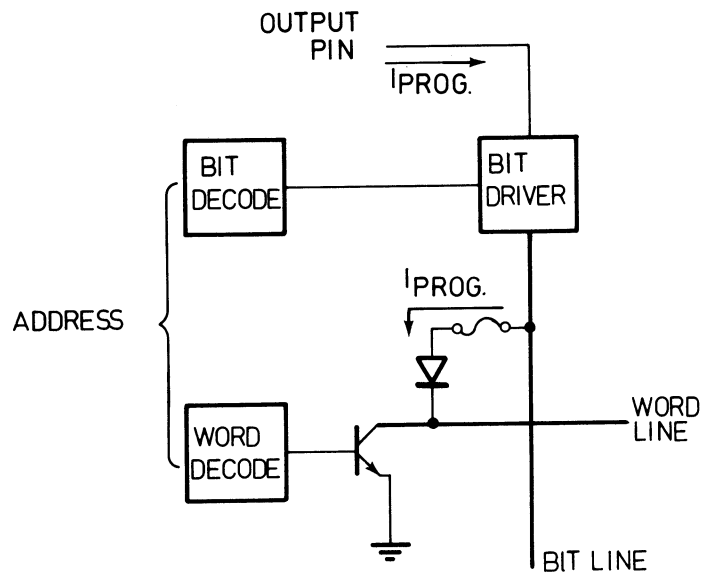


Fig. 3.5

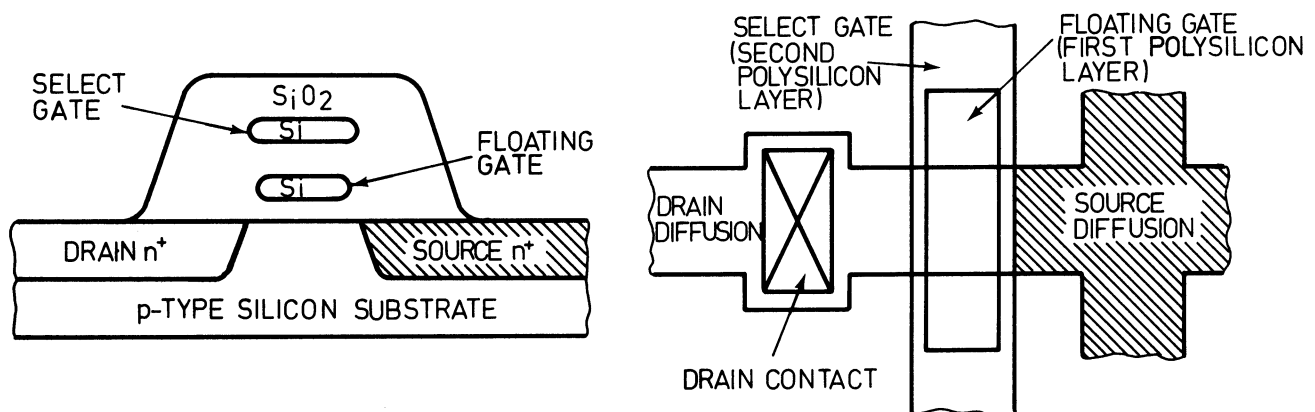


Strømvej ved programmering af PROM.

Fig. 3.6

**EPROM (Erasable PROM)**

Dette er læse-hukommelser, som kan programmeres og slettes igen mange gange. En EPROM hukommelsescelle består ofte af en „flydende” silicium gate (kapacitet), som lades op af et pulstog. Engang programmeret kan en EPROM holde sin ladning i flere år. Sletning foretages let med en kraftig ultra-violet lampe. Når lyset rammer de „flydende” silicium gates, bliver disse afladet og EPROM'en er slettet.



EPROM celle.

Fig. 3.7





Fordele og ulemper ved STATISK og DYNAMISK RAM fremgår af nedenfor viste skema, hvor de væsentligste ting i sammenligningen mellem de to typer er nævnt.

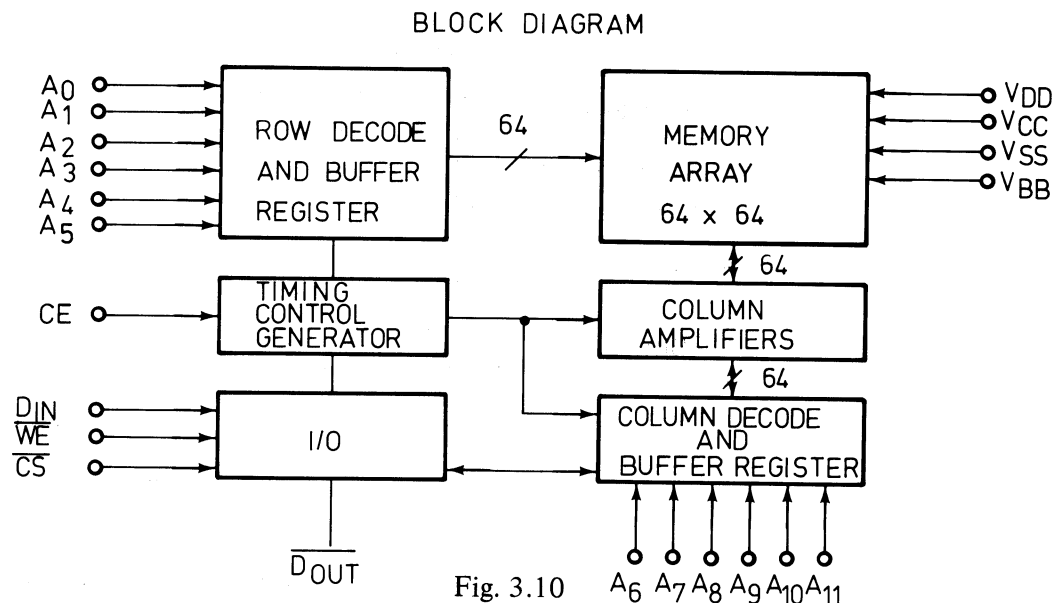
	Fordele	Ulemper
<b>S T A T I S K</b>	INGEN REFRESH. OFTE KUN EN FORSYNING NØDVENDIG (+ 5V).	DYRERE pr. BIT MINDRE BITANTAL pr. areal- enhed på chip'en (i dag nået til 16 K bits). Kræver større DC strømforsyning pr. BIT
<b>D Y N A M I S K</b>	BILLIGERE pr. BIT: STØRRE BITANTAL pr. arealenhed på chip'en (i dag nået til 64 K bits).  Kræver MINDRE DC strøm end statisk RAM.	Kræver REFRESH kredsløb. OFTE FLERE FORSYNINGS- SPÆNDINGER (+ 5V, - 5V og + 12V).  1978

## REFRESH

De kapaciteter, der bruges i en dynamisk RAM, lækker som alle andre kapaciteter. Efter få millisek. er det meste af ladningen forsvundet. For at bevare den information, der er i en dynamisk RAM, er det nødvendigt at genopfriske (refreshe) hukommelsen for hver 1–2 millisek.

Refresh består af at læse informationen ud af hukommelsen for derefter at skrive den ind igen. For at spare tid vil refresh processen læse en hel række eller søjle ad gangen. Det skal lige nævnes, at f.eks. en 4k dyn. RAM (2107C) kan være opbygget som en 64 x 64 matrix.

Der skal derfor kun bruges 64 operationer for at genopfriske hukommelsen.



Til styring af refresh pulser til den dyn. RAM kræves der extra ydre styrelogik. Endvidere vil computerens arbejdhastighed blive nedsat, medens der foretages refresh (ca. 1–5%). I nyere dyn. RAM kredsløb udføres genopfriskningen som „hidden refresh” (skjult genopfriskning). Her foretages genopfriskningen i den clock-halvperiode, som ikke bruges af processoren. Processoren kan derfor ikke „se”, at der anvendes dyn. RAM.

De allernyeste dyn. RAM kredse har internt refresh kredsløb og kan udføre en refresh cycle ved blot at blive aktiveret på et ben.

## TIMING

For at kunne skrive i eller læse fra en memory kreds (chip) er det nødvendigt at:

1. fortælle kredsen, at den bliver adresseret. Dette er chip select (CS) eller chip enable (CE) signalet, som udvælger een memory chip blandt alle de kredse, der er tilsluttet adressebussen.
2. tilføre den adresse, som udvælger det ønskede data ord. Denne adresse vil befinde sig på kredsens adresse indgange. Hvis der f.eks. anvendes et 4k RAM lager som vist i fig. 3.10, vil det være nødvendigt at tilføre et CS signal til at udvælge kredsen og 12 bits på  $A_0 - A_{11}$  til at give den 12-bit adresse, som vil udvælge et ord ud af 4k. ( $2^{12} = 4096 = 4k$ ).

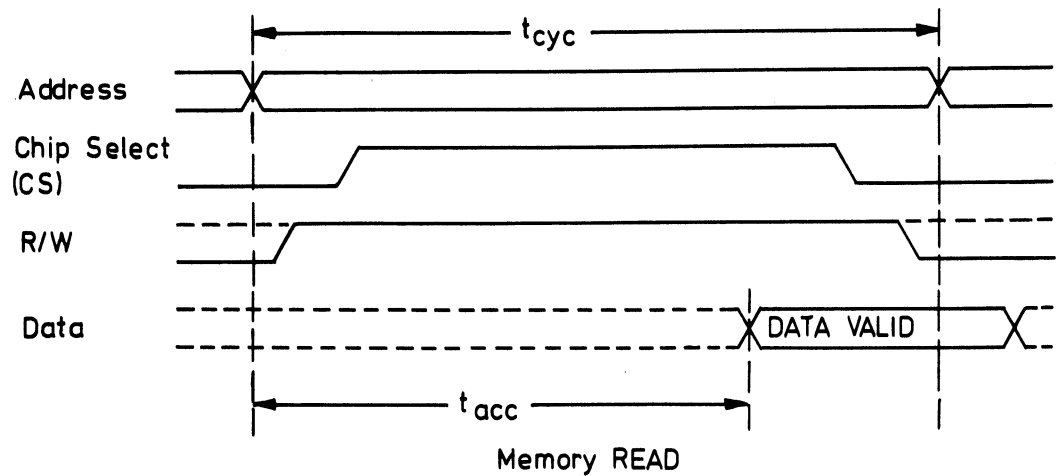


Fig. 3.11 Memory READ.

Efter en tid kaldt access time ( $t_{acc}$ ) vil data kunne læses på data-udgangene. De vil så blive ført til processoren via databussen.

Databussens indhold skal læses, medens der er data tilstede. Dette gøres ved hjælp af chip select, idet databussen læses, når chip select går fra high til low. Adressen skal holdes stabil mindst lige så lang tid som access-time og sædvanligvis længere. Chip select eller chip enable signalet bliver normalt tilført samtidig med adressen, men behøver ikke at blive det.

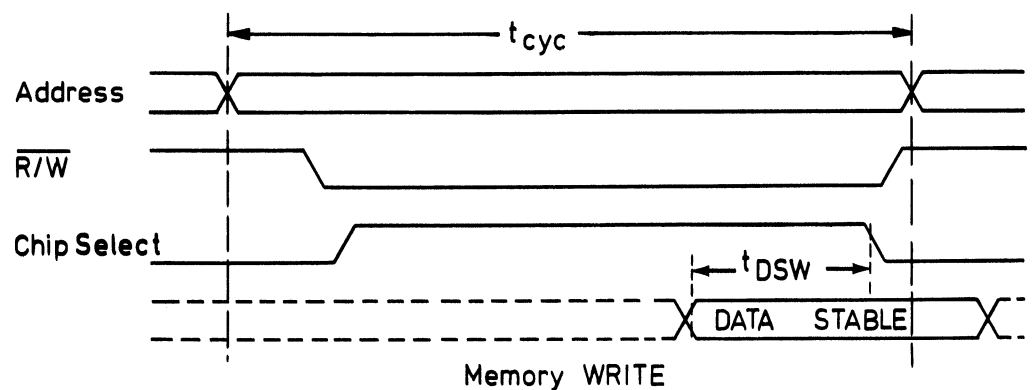


Fig. 3.12 Memory WRITE.

Skrivning i hukommelsen er meget lig læseoperationen. Adresse og chip select skal udpege en bestemt plads til data ordet. Samtidig skal de data, som skal skrives ind i lageret, være tilstede på databussen i en nærmere specificeret tid ( $t_{DSW}$ ) inden chip select igen går på LO. Efter en tid, som kaldes cycle-time ( $t_{cyc}$ ) vil data være skrevet ind på den valgte adresse.

### 3.3 I/O-KREDSLØB (INPUT-OUTPUT KREDSLØB)

For at kunne forbinde microprocessoren til de ydre enheder skal der foretages en tilpasning af signalerne. Det kan f.eks. være enheder som skærmterminaler, båndstationer eller en TTY (TeleTYpe writer: fjernskrivere), der kræver data på serieform. Hvis det drejer sig om læsning af data fra et tastatur, A/D-converter eller styring af displays anvendes data i parallel i kommunikationen.

#### 3.3.1 PIO (PROGRAMMABLE INPUT-OUTPUT)

PIO eller PIA (Peripheral Interface Adapter) er en parallel interface kreds, hvis virkemåde kan programmeres.

For at kunne forbinde et input- eller output signal til microprocessorens databus er det normalt nødvendigt, at der er en latch på input og en latch på output. Input-latch'en vil holde udefra kommende data længe nok til at processoren kan nå at læse dem. På samme måde er det nødvendigt at „fryse” output data i så lang tid, at de ydre enheder kan nå at gøre brug af dem. Ved hurtige input-output signaler som dem, der skal bruges af en puncher og en tape-reader er det ofte en fordel, hvis I/O-enheden har mulighed for „hand shaking” (se i det følgende).

**PIA (6821)**

Som et eksempel på en parallel input-output enhed kan vi se på Motorolas M6821 PIA. Denne PIA har 2 porte (hver på 8 bits) plus to control lines til hver port. Andre fabrikater kan have andre kombinationer. Hver port har her 3 registre (se fig. 3.13):

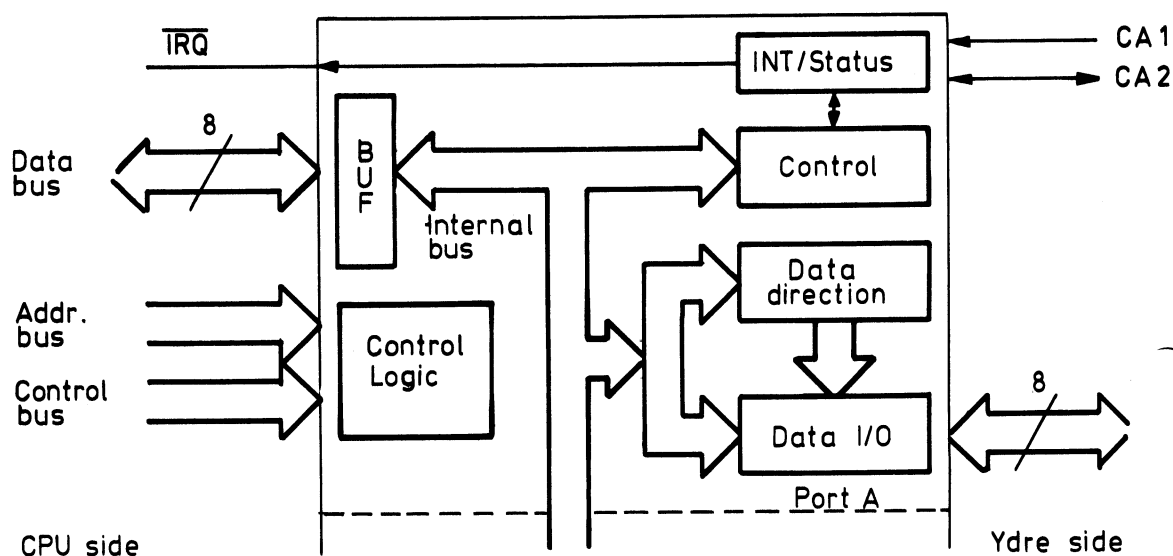


Fig. 3.13

1. Data-buffer registret (Data I/O), som indeholder input eller output data for hver af de 8 I/O-lines.
2. Data-direction registret. Et „Ø” eller et „1” skrevet i en bit position i dette register bestemmer, om den tilsvarende dataledning til den ydre verden skal være input eller output. „Ø” betyder input og „1” output.

Det kunne godt have været omvendt, men er valgt således af sikkerhedsmæssige grunde. Under opstart af et computersystem vil PIA'en blive tilført et reset signal, som sætter alle registre på "Ø". Dette medfører, at alle de perifere linier bliver inputs og ikke outputs med eventuelle signaler, som kunne ødelægge noget i den ydre verden.

3. Control-registret gemmer de kommando-bits, som microprocessoren sender til porten. For hver port vil microprocessoren specificere, om der skal genereres interrupt eller ej, og hvilke kontrolsignaler, der skal have indflydelse på porten. Dermed menes, om der skal afgives signal, når data-registret er fuldt eller tomt (hand shake). Endvidere indeholder controlregistret to status bits, som kan fortælle, om der er modtaget et signal på control ledningerne.

**HAND SHAKE**

Som et eksempel på, hvorledes hand shake signalerne benyttes, vil vi se på, hvorledes signalerne kan bruges i forbindelse med en paper tape-reader (hulstrimmellæser).

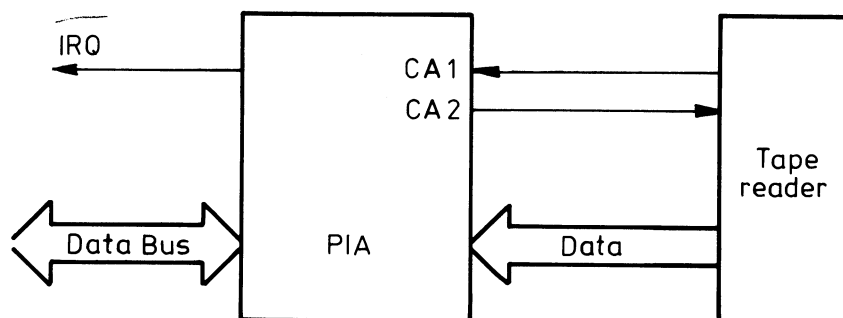


Fig. 3.14

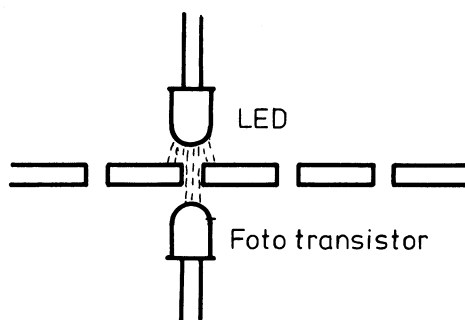


Fig. 3.15

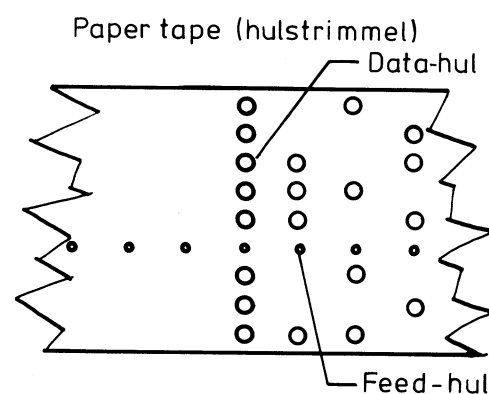


Fig. 3.16

Hvis det antages, at alle hullerne i hulstrimlen aftastes af foto-transistorer som vist i fig. 3.15, vil data- og feed-hulsignalerne blive som vist herunder:

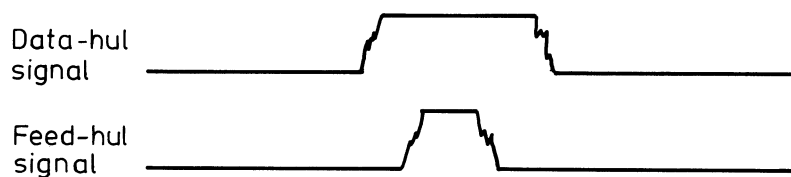


Fig. 3.17

Feed-hul signalet tilføres CA1 og data-hul signaler kobles til PIA'ens input ledninger. CA2 control ledningen styrer fremføringen af strimlen.



Arbejdsmåden er da således:

1. CA2 ledningen aktiverer motoren, så hulstrimlen føres frem til næste række huller.
2. Feed-hul signalet (CA1) fortæller microprocessoren, at nye data er klar på PIA'ens input ledninger.
3. PIA'ens dataregister læses af microprocessoren.
4. Idet microprocessoren udfører denne læsning, giver PIA'en automatisk signal via CA2 til tape-readeren og denne fører strimlen frem til næste sæt huller.

O.S.V.

Ved at lade PIA'en styre tape-readeren via control ledningerne får microprocessoren tid til andre opgaver, medens strimlen køres frem til de næste huller.

## SERIEL INTERFACE

Overførsel af data på serieform bruges, hvor der ikke er mulighed for at have mange parallelle ledere, f.eks. hvor der er tale om store afstande, eller hvor overførslen sker trådløst.

De første forbindelser over længere afstande blev foretaget ved hjælp af serietransmission, telegraf. Det var amerikaneren Samuel Morse (1837), som fastlagde et alfabet bestående af prikker og streger, der først lavede et brugbart telegrafsystem.

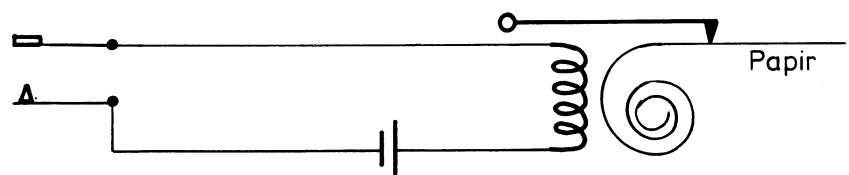


Fig. 3.18

Dette system består af en kontakt på sendersiden og på modtagersiden en elektromagnet med en skrivestift, som registrerer prikkerne og streger på en løbende papirstrimmel.

Senere (1925) fremkom det nogenlunde ideelle telegrafapparat, fjernskriveren (TTY), der betjenes som en almindelig skrivemaskine. Det er nu ikke en håndbetjent kontakt på senderen, men en motor med et bestemt omdrejningstal, som taster senderkontakten. På modtagersiden er der ligeledes en motordrevet aftastning af signalerne.

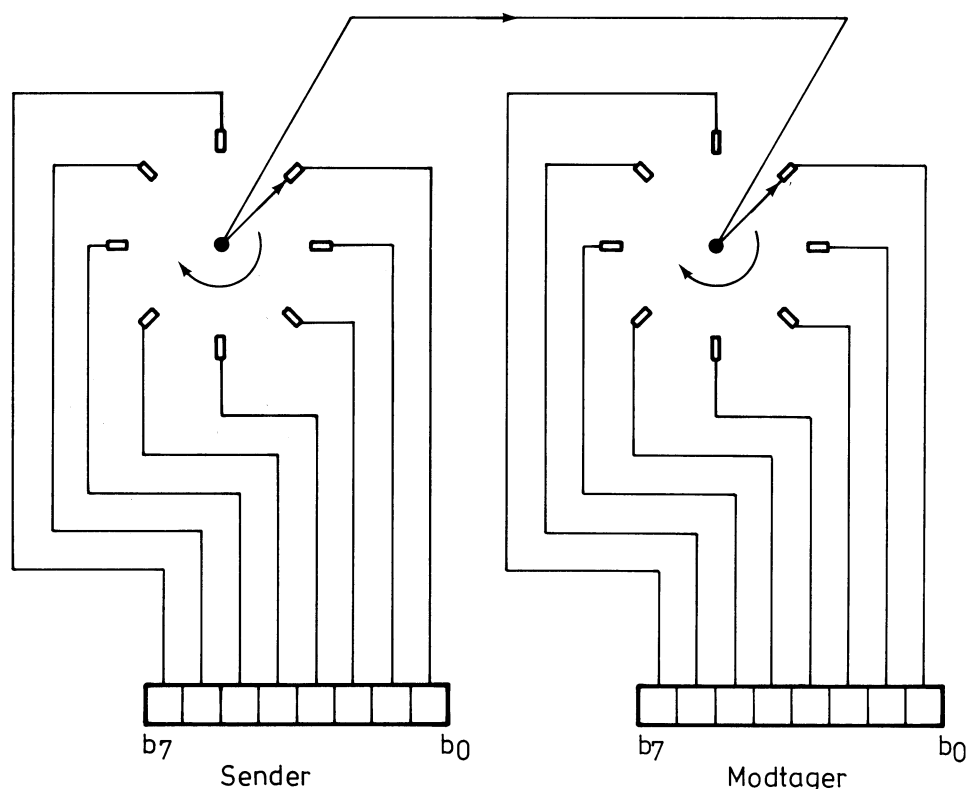


Fig. 3.19

Da to motorers omdrejningstal ikke kan blive helt ens, kræves der en form for synkronisering af sender og modtager. Derfor arbejder fjernskriveren efter det såkaldte start-stop princip, idet sender og modtager startes ved begyndelsen af hvert tegn og stoppes ved tegnets slutning. I normale fjernskrivere består tegnet af 5 impulser, som betegner  $2^5 = 32$  forskellige tegn.

I forbindelse med microcomputere vil hvert tegn normalt bestå af 8 impulser, som sendes med det mindst betydende bit først, svarende til databussens 8 ledninger. Et seriesignal kommer da til at se således ud:

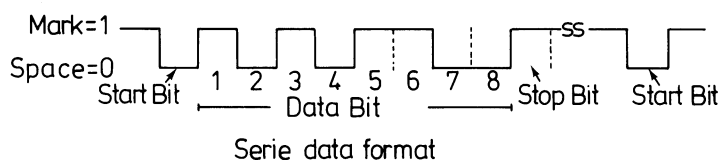


Fig. 3.20

I forbindelse med seriesignaler benævnes High = logisk 1 for MARK og Low = logisk 0 for SPACE. Når der ikke skrives på fjernskriveren, vil senderen give et konstant mark signal. Til synkronisering af modtageren afsendes der i begyndelsen af hvert tegn en start puls, som altid er et space signal. Herefter afsendes de otte data bits og til sidst et eller to stop bits, som altid er et mark signal. Adskillelsen mellem de enkelte data bits ligger i deres tidsmæssige rækkefølge. For at kunne modtage det afsendte seriesignal korrekt, må man kende de enkelte bits varighed. Sendehastigheden bestemmes udfra de enkelte bits varighed. Hvis et bit interval er f.eks. 9 msec, er sendehastigheden

$$\frac{1}{9 \text{ m}} = 110 \text{ Baud},$$

hvilket er normal sendehastighed for en TTY.

Den her omtalte måde at overføre seriedata på kaldes for asynkron-serietransmission.

Det asynkrone serietransmissionssystem anvendes også til andre medier end TTY, f.eks. CRT (skærmterminal).

Disse andre enheder kan køre med andre signalhastigheder end 110 Baud. Der findes nogle standard signalhastigheder, Baud-rates, som normalt anvendes:

50–75–110–150–300–600–1200–2400–4800–9600–19200 baud.

Når et seriesignal skal modtages, er der forskellige muligheder for fejl. Signalet vil under sin vej fra sender til modtager blive forvrænget og måske også tilført støjimpulser. Endvidere er det mest sandsynligt, at der også er en afvigelse med hensyn til sende- og modtager-aftastningsfrekvens. For at undgå disse fejl aftaster modtageren det indkommende signal ved at tage en „smagsprøve” = scanning på signalniveauet midt i de forskellige bit intervaller, som vist i fig. 3.21.

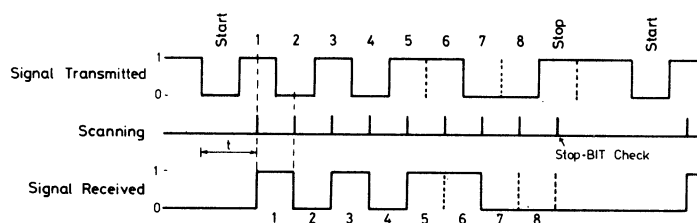


Fig. 3.21

På fig. 3.22 og 3.23 ses det, at selvom der er tale om ret store hastighedsafvigelser, modtages der stadig korrekt signal.

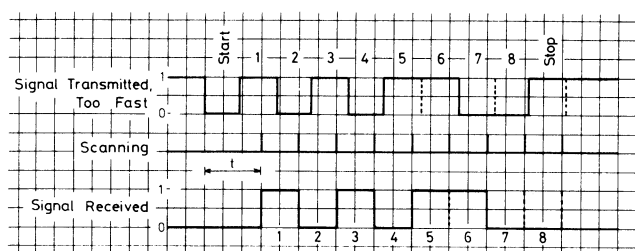


Fig. 3.22

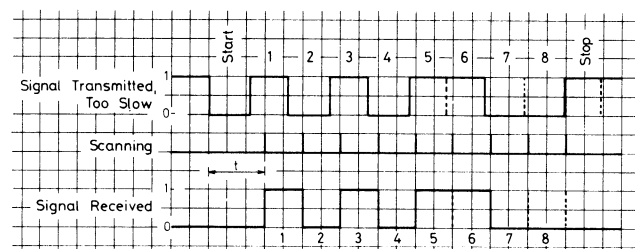


Fig. 3.23

Et støjsignal kan måske blive opfattet af modtageren som et start-signal. Derfor undersøges startbittets midte, hvis der her er et mark signal, resettes modtageren og er dermed klar til at modtage en ny startinformation. Når et startsignal er godkendt, aftastes de 8 databits og derefter undersøges det, om stopbittet er et mark signal. Er dette ikke tilfældet, må der være en synkroniseringsfejl, benævnt framing error.

For at kunne få større mulighed for at opdage fejl, sendes der ofte et ekstra bit med signalet, som det mest betydende bit. Dette ekstra bit betegnes PARITETS BIT.

**LIGE PARITET (even):** Ved lige paritet gives paritetsbittet en sådan værdi 1 el. 0, at summen af alle ettere incl. paritetsbittet er lige.

**ULIGE PARITET (odd):** Ved ulige paritet skal summen af alle bits incl. paritetsbittet være ulige.

Ved på modtagersiden at undersøge paritetsbittets værdi kan alle **ULIGE ANTAL FEJL** opdages, hvorimod et lige antal fejl ikke vil opdages. Der anvendes oftest ulige paritet, da denne vil give en fejlindikation, hvis et signal består af ene nuller.

De fleste skærmterminaler, printere, o.s.v., som bruges i forbindelse med microcomputeren, benytter sig af ASCII-koden (The American Standard Code for Information Interchange). Denne kode består af syv bits (128 mulige karakterer) og et ottende bit, paritets bit.



## ACIA

Ved seriel datakommunikation i forbindelse med microcomputere anvendes en særlig interface kreds, ACIA (Asynchronous Communications Interface Adapter). Denne interface kreds minder meget om de tidligere brugte UARTs (Universal Asynchronous Receiver Transmitter).

En standard seriel interface kreds (UART) kan opdeles i 3 sektioner: en modtager, en sender og en kontrolenhed.

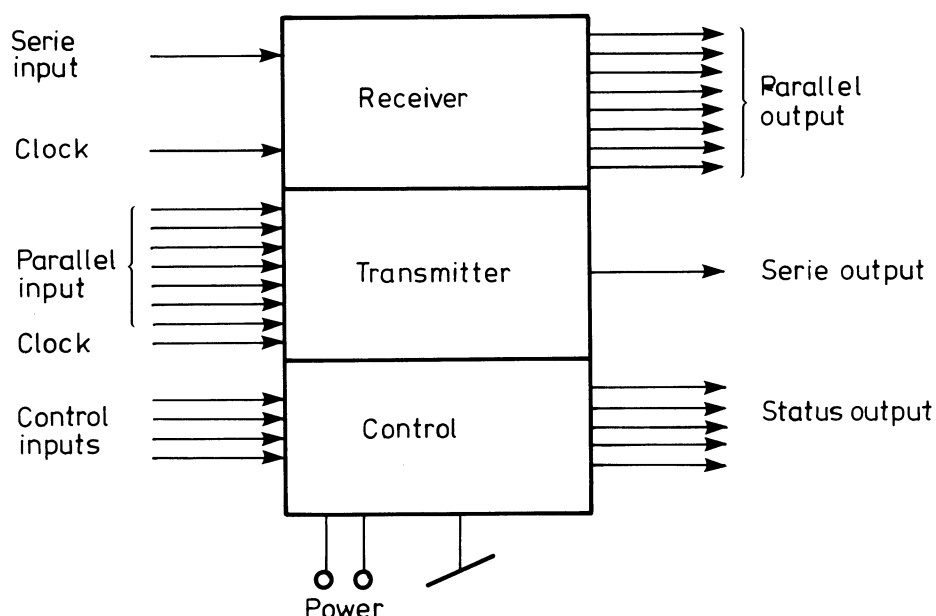


Fig. 3.25

Modtageren (Receiver) modtager data på serieform og omformer disse til 8 bit parallel data. De 8 bit data befries i modtageren for startbittet og stopbits.

Senderen (Transmitter) modtager data som 8 bits parallel (fra MPU'en) og omformer disse til seriedata. I senderen genereres automatisk startbit, stopbits og et eventuelt paritets bit, så de seriedata, som udsendes, får det ønskede udseende.

Sende- og modtagehastigheden bestemmes af de respektive clock-signaler.

Control sektionens inputsignaler bestemmer via controlsektionen, hvorledes sender og modtager skal virke. Endvidere fortæller controlsektionen via status outputs, hvorledes den øjeblikkelige situation er.

I ACIA'en (MC 6850) findes de samme sektioner som i UART'en. ACIA'en er designed, så den kan kobles direkte på microcomputerens bussystem, så microprocessoren kan styre ACIA'ens funktioner.

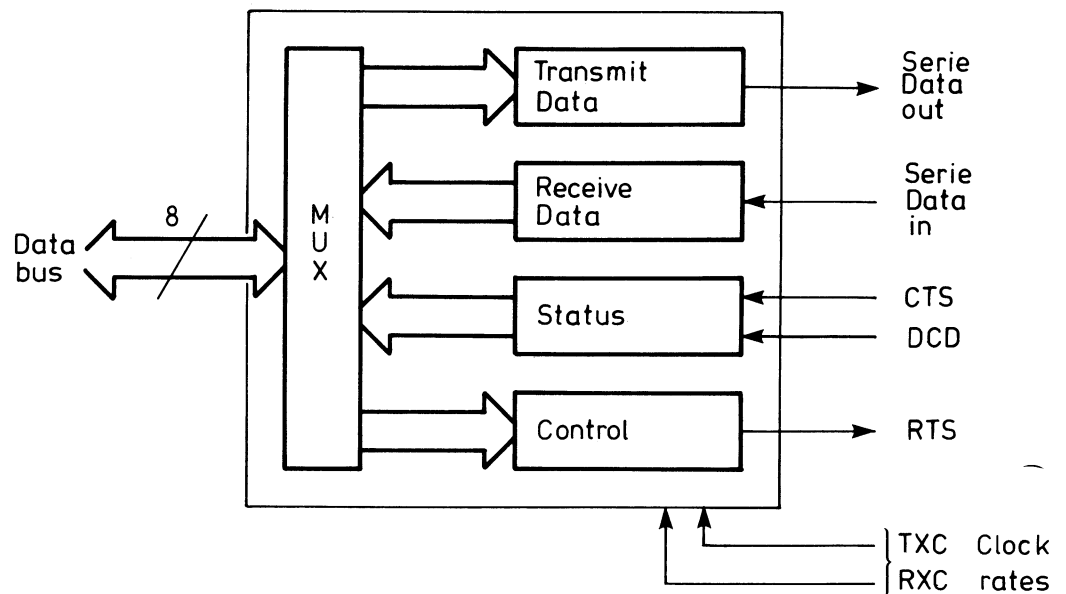


Fig. 3.26

Foruden omformning fra serie til parallel og fra parallel til serie har ACIA'en RS 232C standardens control funktioner, så den kan kobles direkte sammen med et modem (MC 6860), (ang. RS232C og modem se senere).

Det interne clock-signal til modtager og sender fås ved at neddele den frekvens, som tilføres TXC og RXC med 1, 16 eller 64 gange. Hvis der deles med 16 eller 64 gange, kan ACIA'en foretage synkronisering af sender og modtager automatisk, hvorimod denne synkronisering må foretages eksternt, når TXC og RXC ikke neddeles.



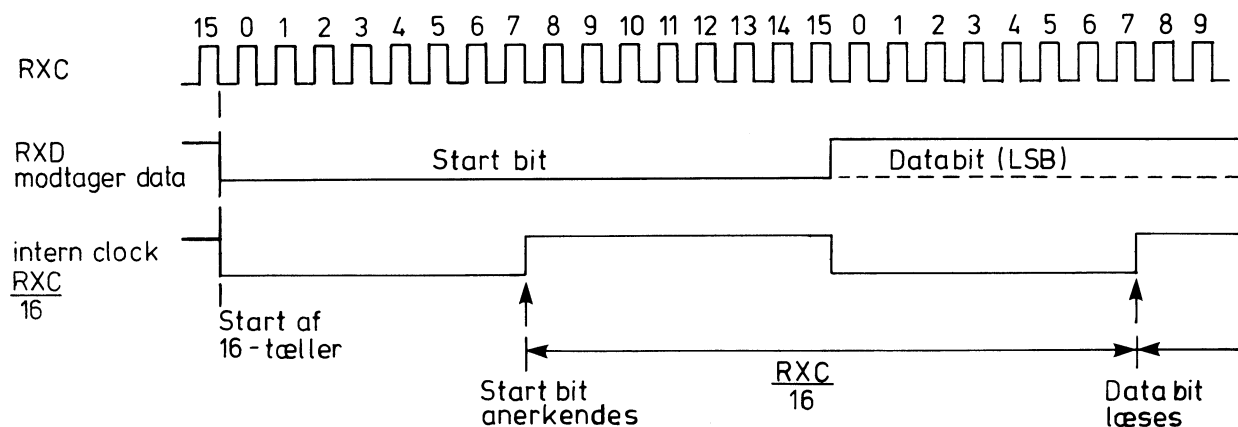


Fig. 3.27

Når der modtages en mark → space transision, startes 16-tælleren som vist på fig. 3.27 på hver forkant af det interne clocksignal aflæses seriedataledningens niveau. Første gang aflæses altid et space signal (startbit). Hvis det ikke er et space signal, må det være et støjsignal, som har startet tælleren, hvorfor denne standses og ACIA'en er klar til at modtage en ny mark → space transision. Er startbittet derimod i orden, aftastes databittene ved hver forkant af den interne clock. På denne måde undersøges midten af hvert databit og der er dermed mindst risiko for fejldetektering på grund af linieforvrængning eller hastighedsfejl.

## SIGNAL STANDARD

### Current Loop

I fjernskrivernet, hvor der sendes med lave baudrates, er et Mark signal defineret som en strøm på 20 mA og et space som ingen strøm. Ved at anvende strømmen som reference, får ledningsforbindelsens længde (modstand) ikke den store betydning. Der opnås endvidere en god støjimmunitet, da en linie, hvorpå der sendes 20 mA logiske signaler, udgør en lavimpedanset transmissionslinie.

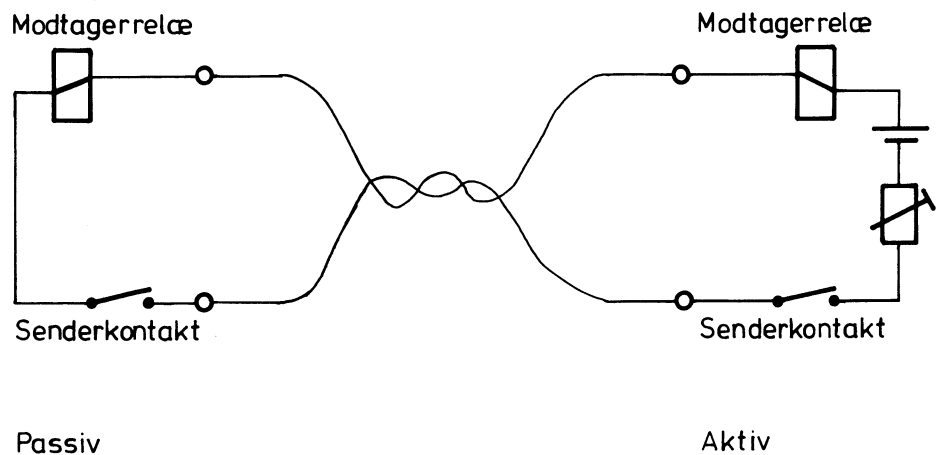


Fig. 3.28

Som det ses på fig. 3.28 sendes de digitale signaler blot ved at slutte og bryde strømmen. Modtageren består af et relæ, som føler, om der er strøm (mark) eller ikke-strøm (space).

Ved 20 mA current loop-forbindelsen tales der om:

Aktiv current loop:

Kredsløbet indeholder en strømforsyning, som leverer de 20 mA til strøm loop'en (fig. 3.28).

Passiv current loop:

Her bidrager kredsløbet ikke med nogen strøm (fig. 3.28).

## RS 232C

EIA (Electronics Industry Association) har lavet en standard for serie transmission, hvor mark og space udgøres af spændingsniveauer. Foruden selve datasignalet omfatter RS232C en del andre signaler, som primært skal bruges, når der skal kobles et MODEM (Modulator – Demodulator) til serieporten. Et MODEM bruges, hvor data skal transmitteres via telefonnettet el.lign. Foruden de elektriske data, som kan ses i tabellen, fig. 3.29, angiver RS 232C også de fysiske specifikationer (stik, benforbindelser), se tabel fig. 3.30

Electrical characteristics of RS232C/RS422/RS423

Driver configuration	RS232C Single-ended (unbalanced)	RS423 Single-ended (unbalanced)	RS422 Differential (balanced)
Max recommended cable length	50 ft	4000 ft (1200 m)	4000 ft (1200 m)
Max signaling (data) rate	20 kbits/s	100 kbits/s at 40 ft	10 Mbits/s at 40 ft
Driver output-voltage, open circuit ( $V_0$ )	$\pm 3$ V to $\pm 25$ V	$\pm 4$ V to $\pm 6$ V	$\leq 6$ V between A and B terminals
Driver output-voltage, loaded ( $V_1$ )	$\pm 5$ V to $\pm 15$ V 3 k $\Omega$ to 7 k $\Omega$ load	$ V_1  \geq 9  V_0 $ 450 $\Omega$ load	2 V between A and B 100 $\Omega$ balanced load (2 to 50 $\Omega$ loads)
Driver output short-circuit currents ( $I_s$ )	$ I_0  \leq \pm 500$ mA	$ I_0  \leq \pm 150$ mA	$ I_0  \leq \pm 150$ mA
Driver output slew rate (dV/dt)	30 V/ $\mu$ s max	Based on cable length and signal- ing rate	No control necessary
Receiver MARK(off=1)	$\leq -3$ V	A = negative wrt B'	A B
Receiver SPACE (on=0)	$\geq +3$ V	A = positive wrt B'	A B
Receiver-input threshold	$\pm 3$ V	200 mV differential	200 mV differential
Receiver-input impedance	3 to 7 k $\Omega$ 2500 pF	$\geq 4$ k $\Omega$	$\geq 4$ k $\Omega$
Receiver input-voltage range	- 25 V to + 25 V	- 12 V to + 12 V	- 12 V to + 12 V
Power-off measurement ( $I_x$ ) ( $V_0$ applied to unpowered device)	$2$ V $\leq V_0 \leq 25$ V	$ I_x  \leq \pm 100$ $\mu$ A $-6$ V $\leq V_0 \leq 6$ V	$ I_x  \leq 100$ $\mu$ A $0.25$ V $< V_0 \leq 6$ V

Fig. 3.29

EIA RS-232C circuits by category

Pin no. <sup>+</sup>	EIA RS232C Interchange Circuit	CCITT V.24 equivalent	Description	Data		Control		Timing	
				from DCE	to DCE	from DCE	to DCE	from DCE	to DCE
1	AA	101	Protective ground						
7	AB	102	Signal/ground/common return						
2	BA	103	Transmitted data		X				
3	BB	104	Received data	X					
14	SBA	118	Secondary transmitted data		X				
16	SBB	119	Secondary received data	X					
4	CA	105	Request To Send				X		
5	CB	106	Clear To Send			X			
6	CC	107	Data Set Ready			X			
20	CD	108.2	Data Terminal Ready				X		
22	CE	125	Ring indicator			X			
8	CF	109	Received-line signal indicator			X			
21	CG	110	Signal-quality detector			X			
19	SCA	120	Secondary Request To Send				X		
13	SCB	121	Secondary Clear To Send			X			
12	SCF	122	Secondary received line-signal detector						
23	CI	112	Data-signal rate selector (DCE)			X			
23	CH	111	Data-signal rate selector (DTE)				X		
24	DA	113	Transmitter-signal element timing (DTE)						X
15	DB	114	Transmitter-signal element timing (DCE)					X	
17	DD	115	Receiver-signal element timing (DCE)					X	

+ Refers to 25-pin Dataphone connector Pins 9, 10 reserved for data-set testing. Pins 11, 18 and 25 are unassigned.

DCE = Data Terminating Equipment

DTE = Data Terminal Equipment

Fig. 3.30

I fig. 3.29 er der foruden data for RS 232C vist data for RS 422 og RS 423. Disse standarder bruges ikke meget på grund af RS 232C's store udbredelse.

På fig. 3.31 ses de forskellige typer af drivers og receivers for de tre standarder.

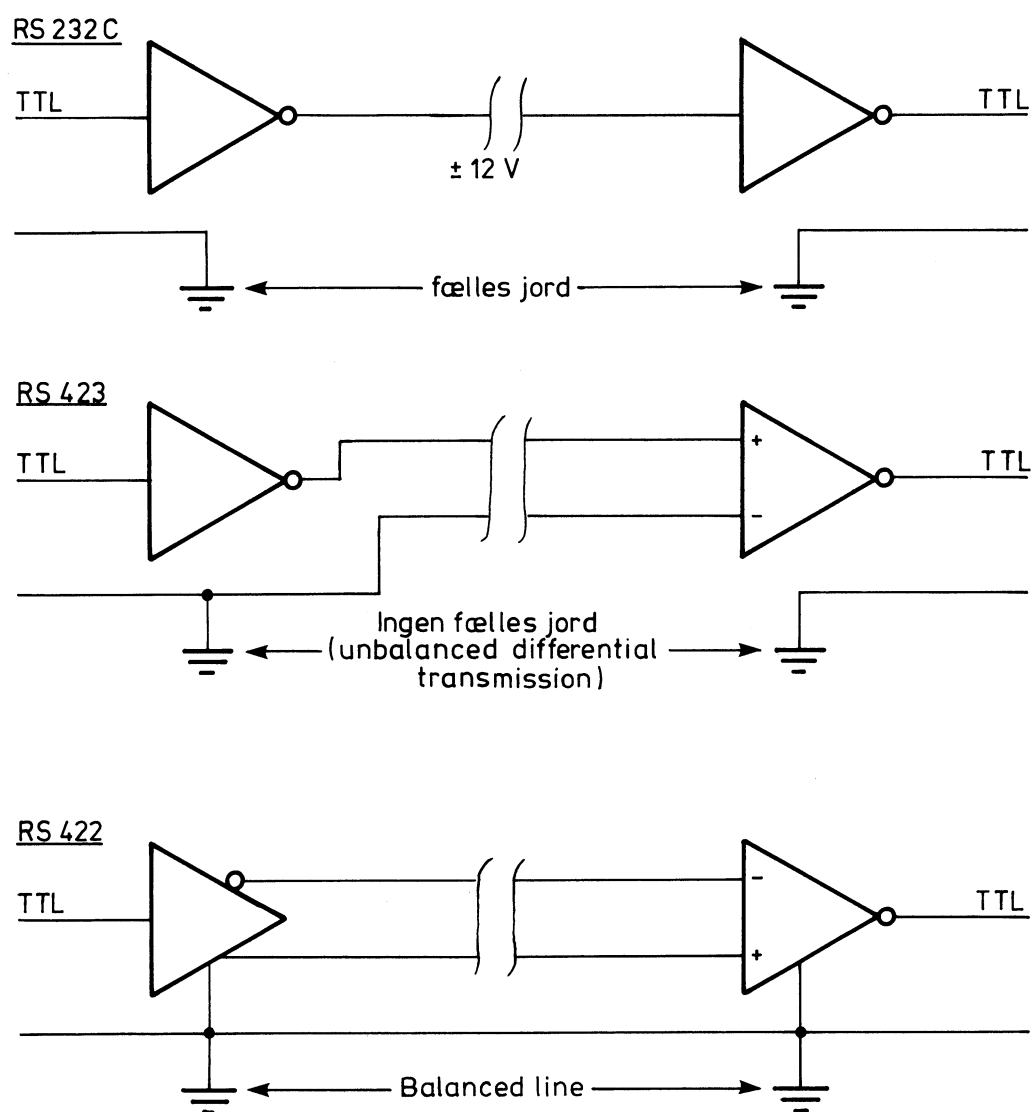


Fig. 3.31

Da RS 232C bruges i forbindelse med modems, vil vi se lidt nærmere på, hvad et modem er.

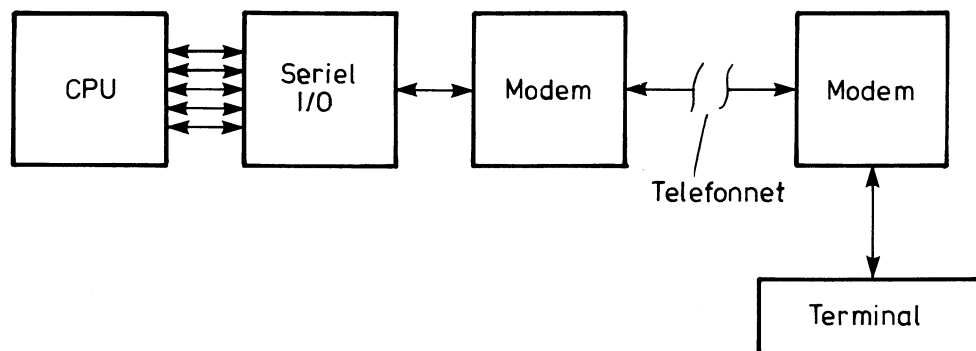


Fig. 3.32

Et modem er et kredsløb, som omformer de digitale signaler "0" og "1" til et tonesignal, som har en frekvens for et logisk 0 og en anden frekvens for et logisk 1. Da modem'et er forbundet til telefonnettet, skal tonerne være indenfor frekvensområdet 500–3000 Hz. Signalet på telefonlinien kan også opfattes som en bæreølge (CARRIER) som frekvensmoduleres, også kaldet Frequency Shift Keysing (FSK).

Hvis der benyttes to forskellige bæreølgefrekvenser, er det muligt at sende og modtage samtidigt. Dette kaldes FULD DUPLEX. Hvis der IKKE kan sendes og modtages samtidigt, opereres med HALV DUPLEX. Arbejdes der med HALV DUPLEX, kræves nogle signaler til at styre omskiftningen mellem sending og modtagning.

Som tidligere nævnt er RS 232C udformet med henblik på forbindelse til et modem, derfor de mange kontrolsignaler. Vi vil her prøve at se på nogle af disse signalers funktion:

#### DCD (Data Carrier Detect).

Hvis DCD er logisk 1, betyder det, at modem'et modtager en bæreølge med den rette frekvens og amplitude. Er DCD derimod logisk 0, betyder det, at signalet på telefonlinien ikke er OK.

RTS (Request To Send). Når RTS går fra "0" til "1", er det en besked til modem om at skifte til sendetilstand. Et "1" til "0" skift giver modem besked om at skifte til modtagning af data fra telefonnettet.

#### CTS (Clear To Send).

"0" til "1" er et svarsignal fra modem, som fortæller, at modem er klar til at sende. Hvis CTS går fra "1" til "0", er det et svar, som siger, at modem nu er klar til at modtage.

Disse kontrolsignalers funktion kan vises med følgende lille eksempel: De to modems er forbundet til telefonlinien og arbejder i halvduplex. Kontrolsignalerne bliver da som vist:

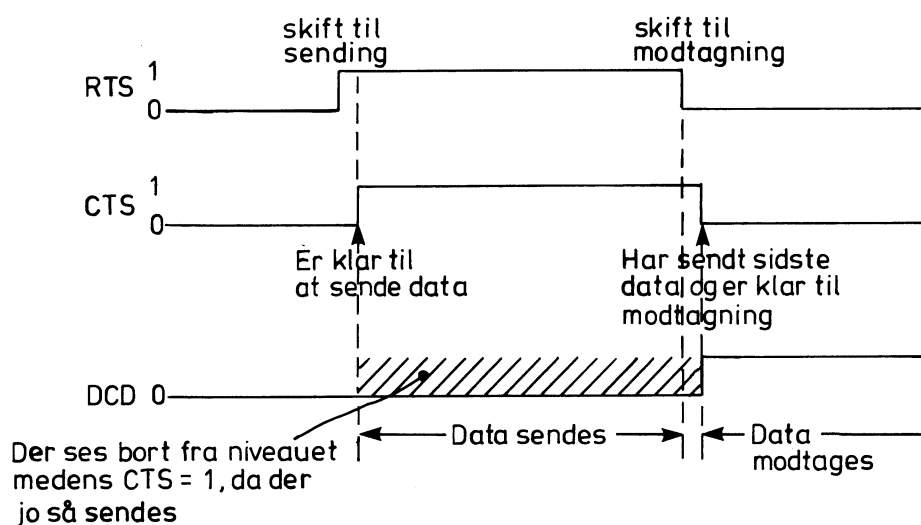


Fig. 3.33

De her viste hand shake signaler går ud fra, at telefonlinien er etableret.

RS 232C har mange andre kontrolsignaler, f.eks. til etablering af telefonforbindelse, o.s.v. Disse vil dog ikke blive behandlet her.

## MICROPROCESSOREN

Microprocessoren er microcomputerens centrale enhed og er derfor også det mest komplekse af de kredsløb, der indgår i microcomputeren.

Microprocessoren kaldes også:

<u>C</u> entral <u>P</u> rocessing <u>U</u> nit	(CPU)
<u>M</u> icro <u>P</u> rocessing <u>U</u> nit	(MPU)

Det er denne kreds, som udfører det „praktiske arbejde” (i microcomputeren), som består i at udføre beregninger samt kontrollere de øvrige kredsløb.

En typisk microprocessor kan opdeles i forskellige funktioner, der hver tjener et bestemt formål. Dette er skitseret i nedenstående „kassediagram”:

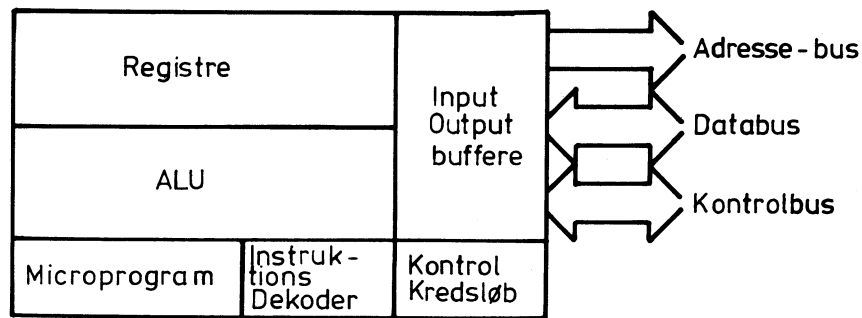


Fig. 3.34

**Arithmetic/Logic Unit (ALU)**

Alle processorer indeholder en Aritmetrisk/Logisk enhed eller blot ALU. ALU'en er, som navnet siger, den del af microprocessoren, der udfører de aritmetriske og logiske behandlinger af de binære tal.

ALU'en skal indeholde en Adder, som kan udføre en addition af to registres indhold. Dette betyder, at processoren kan udføre aritmetriske beregninger på data, som den modtager fra memory eller andre enheder.

Ved hjælp af adderen kan en programmør få processoren til at foretage de øvrige regningsarter: subtraktion, multiplikation og division. Normalt er der dog indbygget en subtraktionsrutine i ALU'en. Endvidere kan ALU'en udføre de almindelige logiske funktioner AND og OR samt nogle skiftefunktioner.

**Registre**

Registre er midlertidige lagerceller indeni microprocessoren. Nogle af registre har et bestemt formål (programtæller og instruktionsregister), medens andre har et mere alment formål (accumulatorer).

**Input-Output buffere**

Input-output bufferne er alm. driver kredsløb, som skal isolere MPU'ens interne bus fra den externe. Adressebusbufferen er en output buffer, hvorimod databus bufferen er bidirectional. Det vil sige, at databussen kan fungere både som ind- og udgang. De to bufferkredsløb er endvidere 3-state buffere, så forbindelsen til busserne kan „afbrydes”, hvis det ønskes.

**Instruktionsdekoder**

I 8-bit processorer, hvor databussens bredde er 8 bit, vil det være naturligt at alle data og instruktioner gemmes i lageret som 8-bit ord (BYTE).

Hver af de operationer, som processoren kan udføre, har sin egen 8-bit kode (byte), kendt som en INSTRUKTIONSKODE eller OPERATIONSKODE. Når der bruges 8-bit ord til at identificere hver instruktion, kan der skelnes mellem 256 forskellige instruktioner ( $2^8 = 256$ ). Dette er tilstrækkeligt for de fleste processorer.

Oversættelsen af instruktionskoden til handling foretages af INSTRUKTIONSDEKODEREN sammen med det tilhørende kontrolkredsløb.



## Microprogram

Når instruktionsdekoderen har genkendt en kode, udvælges en bestemt del af microprogrammet, som derefter styrer den videre behandling i processoren. Det er således microprogrammet, der bestemmer, hvad de enkelte instruktioner kan udføre.

## Kontroll kredsløbet

Kontroll kredsløbet modtager TIMING fra clock-generatoren. Kontroll kredsen aktiverer de forskellige kredsløb indeni processoren i den rigtige rækkefølge. Desuden modtages signaler ude fra microcomputerens andre enheder til styring af processoren. Der udsendes også styresignaler fra processoren via kontroll kredsløbet til de øvrige af microcomputerens kredse.

De kontrolsignaler, der sendes ud og ind af processoren, har sin egen bus kaldet kontroll bussen. Et par af de vigtigste controlsignaler, der sendes ud fra processoren, er:

### VMA (Valid Memory Address)

er et signal, som angiver, hvornår der står en brugbar adresse på adressebussen.

### R/ $\overline{W}$ (Read/Write)

er et signal, som fortæller, hvilken vej data transmitteres på databussen. READ: data føres til processoren og WRITE: data går fra processoren og ud til de øvrige kredse.

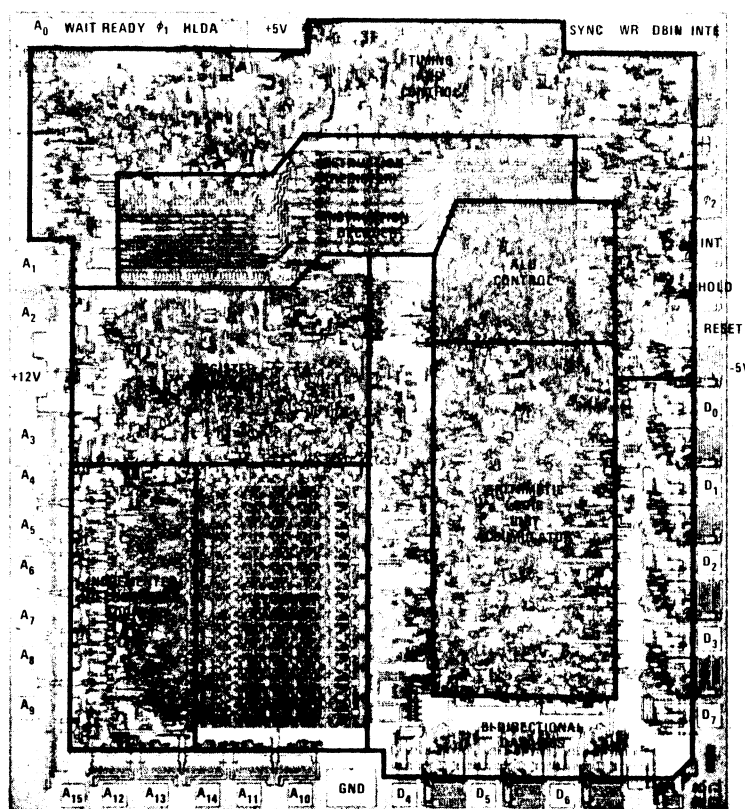


Fig. 3.35 intel

På microfotoet af en CPU (8080) kan man tydeligt se, at denne er opdelt i forskellige områder som ALU, Registerne o.s.v.

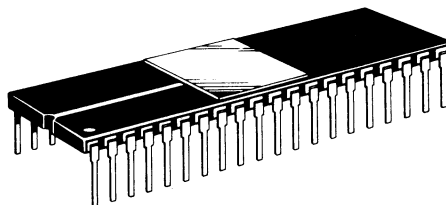


Fig. 3.36

### **μP Hardware**

Den chip (rektangulær stykke silicium), hvorpå microprocessoren er lavet (fig. 3.35), bliver normalt placeret i en DIP (Dual Inline Package) med 40 ben. Hvis dette benantal ikke er tilstrækkeligt, benyttes databussens ben, f.eks. også til nogle af adressebussens forbindelser ved hjælp af multiplexing.

Microprocessorer består af flere tusinde transistorfunktioner. Disse transistorer er normalt af NMOS-typen, da disse er de hurtigste. Tidligere skulle NMOS kredsløb forsynes fra forskellige spændingskilder (+12, +5 og -5 V), men idag skal de fleste processorer kun have en forsyningspænding, +5 V. Effektforbruget er fra 0,5–2 W. Der er begyndt at komme CMOS processorer frem, RCA COSMAC, som har et lavt strømforbrug samt den fordel, at den kan arbejde ved forsyningspændinger på 2–12 V.

De mest brugte processorer (NMOS-type) har som regel TTL-kompatible inputs og outputs med et fan out på 5–8 af de øvrige kredsløbs komponenter (RAM, PIA o.s.v.). Hvis der skal kobles flere kredsløb på busserne, må der indsættes specielle buffere (drivers).

### **Clock-signalet**

Clockfrekvensen kan ikke bruges som sammenligningsgrundlag for en vurdering af de forskellige fabrikaters arbejdhastighed. Nogle typer har en indbygget neddeling af clockfrekvensen, som ikke findes i andre typer. Så vil man sammenligne processorers arbejdhastighed, må man give de forskellige typer den samme beregningsopgave og på denne måde finde den hurtigste. Nogle typer (6800) har både en max. og en min. clockfrekvens, da de interne registre i processoren ofte er dynamiske.

### **PROGRAMMERINGS- MODELLER**

Set fra programmørens side vil microprocessoren tage sig anderledes ud end tidligere beskrevet. For programmøren vil alle de hardware-mæssige problemer være løst, så han kan koncentrere sig om de ting, som han kan ændre på ved hjælp af data og program.

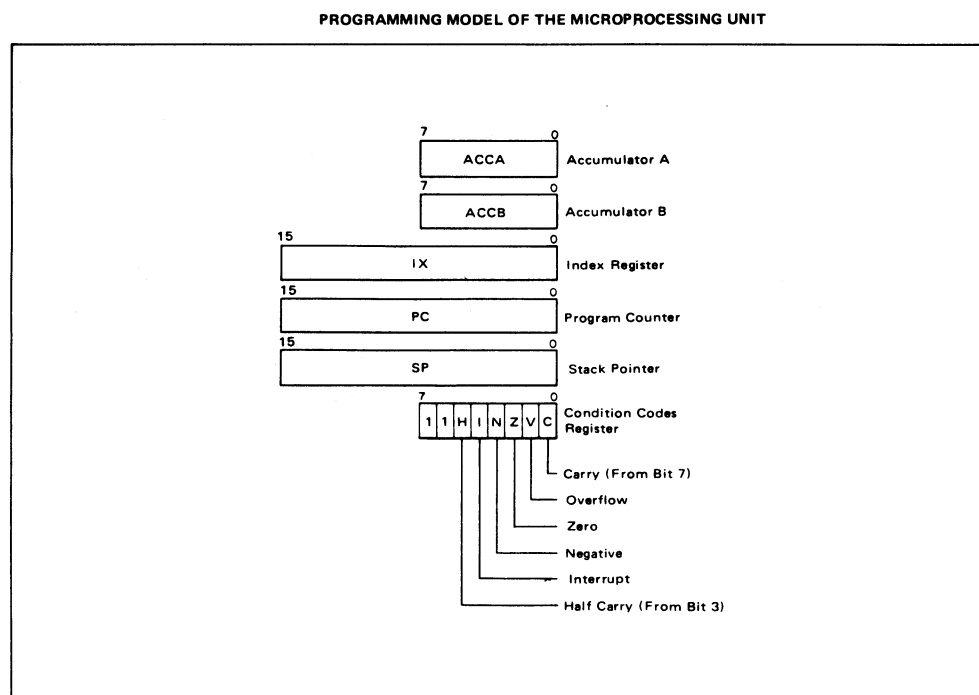


Fig. 3.37

Som det ses af fig. 3.37, ser programmøren på, hvor mange og hvor store registre, der findes i processoren, samt på hvilken måde de kan anvendes.

En kort gennemgang af de enkelte registres navne og anvendelse (6800):

#### Accumulator (Acc) 8 bits

I MC 6800 er der to accumulatorer (A og B). Accumulatorerne bruges sædvanligvis til at opbevare en af de operander, som skal behandles af ALU'en. En typisk instruktion (ADD) vil fortælle ALU'en, at den skal addere indholdet af en memory celle med indholdet af accumulatoren og gemme resultatet i accumulatoren igen. Generelt er accumulatoren både kilde (operand) og destination (resultat) register.

#### Program Counter (PC) 16 bits

De instruktioner, som danner et program, er gemt i lageret (ROM). Microprocessoren refererer til indholdet af lageret for at få at vide, hvad der nu skal foretages. Dette betyder, at processoren må vide, hvilken lagerceller der indeholder den næste instruktion.

Processoren har en tæller, som indeholder adressen på den næste instruktion i programmet. Dette register kaldes Program Counter (PC). Processoren opdaterer PC ved at addere en til PC, hver gang den har hentet en instruktion. Programmøren skal således blot placere sit program med instruktionerne på stadigt stigende adresser.

#### Index Register (IX) 16 bits

En microprocessor bruger ofte et register til at indeholde en adresse på en lagercelle, hvorfra der skal hentes data. Dette register, hvis indhold kan ændres via programmet, kaldes Index Registret.

**Stack Pointer (SP) 16 bits**

Stack'en er et område af lageret, som bruges af microprocessoren, som en notesblok (scratch pad). Stack'en fungerer som et „Last-in-first-out” register (LIFO). Ved visse instruktioner bruger microprocessoren automatisk stack'en. For at kunne holde styr på, hvor i stack'en det sidste dataord står, findes der i processoren et register til dette formål: Stack Pointeren (SP), som hele tiden udpeger den næste tomme celle i stack'en.

**Condition Code Registret (CCR) 8 bits**

I forbindelse med ALU'en sidder der et register (CCR), som hele tiden skal indikere noget om den sidste operation ALU'en foretog: Negativ (N), Zero (Z), Overflow (V), Carry fra bit 7 (C) og Half Carry (H), som indikerer half carry fra bit 3 til 4. Disse bits i CCR bliver brugt til testbare resultater i forbindelse med betingede hop-funktioner.

## PROGRAMMERING

Løsningen af en kontrolopgave, som f.eks. styringen af en vaskemaskine, kan opstilles som en form for „trin for trin“-løsning, der kaldes en ALGORITHM. En algorithm er en „trin for trin“-løsning af en specificeret styringsopgave. En algorithm kan udtrykkes på forskellig måde. Men da en computer ikke kan forstå det normale sprog, skal enhver algorithm oversættes til et simpelt sprog, som en computer kan forstå og udføre. Dette sprog kaldes et PROGRAMMERINGSSPROG. Sekvensen af instruktioner, som svarer til algorithmen og udtrykt på et programmeringssprog kaldes et PROGRAM. Det sæt af instruktioner, som computeren kan forstå direkte (normalt som binære dataord), kaldes MASKINSPROGET. Det at oversætte en algorithm til et programmeringssprog kaldes at PROGRAMMERE.

## ASSEMBLERSPROGET

Det sprogniveau, som ligger lige over maskinkoden (binære ord), er ASSEMBLERKODEN (eller assemblersproget). Assemblersprogets instruktioner svarer nøje til maskinsproget. Assemblersprogets instruktioner er nogle forkortelser af det, som instruktionen (på engelsk) befaler processoren at gøre. Disse forkortelser er udformet som MNEMONICS hver på 3 bogstaver. Mnemonics er mnemotekniske koder (mnemo = hukommelse), altså hukommelsestekniske koder. For eksempel betyder koden LDA: Load Accumulator.

## MASKINSPROGET

Maskinsproget er de binære ord (8 bit ord = BYTE), som fra computerens lager (normalt ROM) tilføres processoren via databussen. Når en sådan databyte (instruktion) når frem til processorens instruktionsdekoder, vil det medføre en aktivitet i processoren, og den vil udføre det, den fik besked om via instruktionen, der er den tilførte databyte.

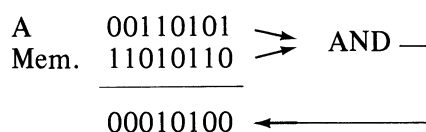
Da instruktionskoden består af en byte = 8 bits, må der være  $2^8 = 256$  mulige instruktionskoder. Af disse 256 udnyttes i M 6800 „kun“ de 197 (se side 4.3 og 4.4).

Disse er delt op i 72 grundinstruktioner fordelt på 6 forskellige adresseringsformer. Disse grundinstruktioner kan igen deles op i nogle instruktionsgrupper:

### Aritmetiske/logiske

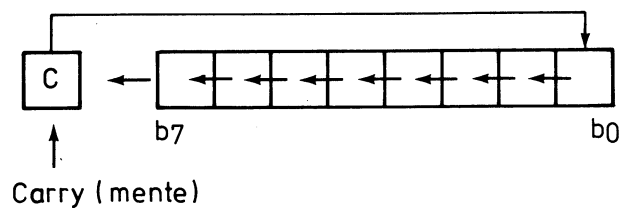
Det er instruktioner som ADD (Addition), SUB (Subtraktion), AND (Logisk AND), OR (Logisk OR), XOR o.s.v. Disse instruktioner har det fælles træk, at de gør brug af ALU'en. F.eks. ved en addition vil det ene af de to tal, som skal adderes, befinde sig i en af accumulatorerne. Det andet tal befinder sig i lageret (memory) og bringes sammen med tallet fra acc. til ALU'en, som foretager additionen, hvorefter resultatet føres til accumulatoren.

Ved en logisk funktion f.eks. AND udføres den logiske behandling bit for bit. Eks.



### Flytte data

Af disse instruktioner kan nævnes: LDA, STA (STORE = gemme), ROL (ROtate Left), o.s.v. Ved en instruktion som ROL, kan accumulatoren opfattes som et skifteregister, der skifter en gang, når instruktionen modtages.



### Sammenligne data

Her er det instruktioner: BIT (bit test), TST (test), CMP (compare = sammenligne), o.s.v. Det karakteristiske ved disse instruktioner er, at de ikke ændrer på indholdet af de registre, hvorpå sammenligningen sker. Instruktionen CMP er egentlig en subtraktion, men resultatet af subtraktionen vil ikke blive gemt noget sted.

Eks.	CMPA	01100000	← Acc. A
		00100000	← Memory
		<hr/>	
		01100000	← Acc. A (uændret)
		HINZVC	
		<u>11XX0000</u>	← CCR

Det der er sket ved udførelsen af instruktionen CMPA er, at CCR (Condition Code Register) blev indstillet, som om der blev foretaget en SUBA. Ved nu at undersøge bittene N og Z, kan man se om Acc A er større end, mindre end eller lig med memory.

### Styre programmet

Det er disse instruktioner, som gør computeren „intelligent“. Disse instruktioner adskiller computeren fra calculatoren. De giver computeren evnen til at foretage beslutninger (forskellige programmer) afhængigt af de målte parametres værdi.

Det er instruktionerne JMP (JuMP) og branch, f.eks. BPL (Branch if Plus).

Disse forgrenings- eller hop-instruktioner kan opdeles i ubetingede og betingede hop. De ubetingede hop foretages altid, hvorimod de betingede undersøger, om betingelserne er opfyldt ved at undersøge nogle af status bittene i CCR.

## ADRESSERINGER

Microcomputeren (M 6800) har forskellige måder, hvorpå den kan udpege de data, som skal behandles. At udpege et dataord kaldes at adressere, da det sted i lageret, hvor det pågældende dataord er gemt, kaldes en adresse. En adresse er en bestemt kombination af bits på adressebussen. Med 16 bits i adressebussen er det muligt at udpege  $2^{16} = 65.536$  forskellige adresser.

De forskellige adresseringsmåder er:

- Inherent adressering
- Immediate –
- Direct –
- Extended –
- Indexed –
- Relativ –

### Inherent

Denne adresseringsmåde har en-byte instruktioner og kræver ikke adressering af en memory lokation, eller adresseringen er indeholdt i instruktionen. Et eksempel på en inherent instruktion vil være at udføre en „forøg akkumulator B” instruktion, som vil se således ud:

Memory adresse	Memory indhold
5001	5C (INC B opcode)

5C (hex) er INC B inherent opcode, som resulterer i, at acc. B bliver en større.

Source input koden vil være:

INC B

### ACCUMULATOR/INHERENT ADDRESSING

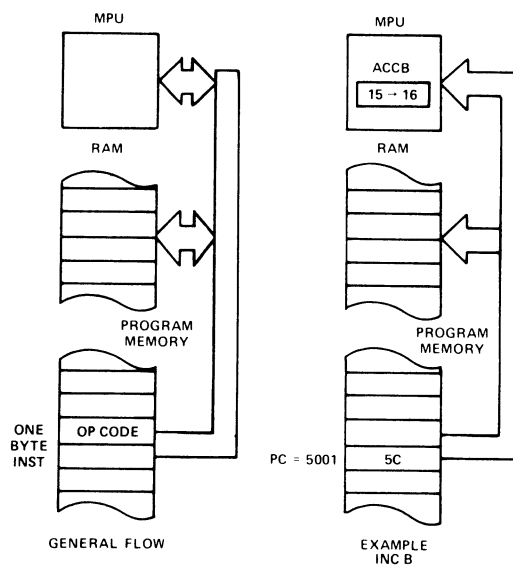


Fig. 4.1



**Immediate**

Ved denne adresseringsmetode findes OPERAND'en på den memory adresse, som følger umiddelbart efter operationskoden (opcode). For eksempel vil „hent til acc. A” (load acc. A) med hex-tallet 25 se således ud:

Memory adresse	Memory indhold
5002	86 (LDA A opcode)
5003	25 (Data)

86 (i hex) er LDA A immediate opcode. 25 (i hex) er data. Resultatet af den ovenfor nævnte instruktion er, at hex-tallet 25 bliver det nye indhold af accumulator A.

Source input koden vil være:

LDA A # \$ 25

# betyder immediate addressing og \$ betyder, at det er et hex-tal.

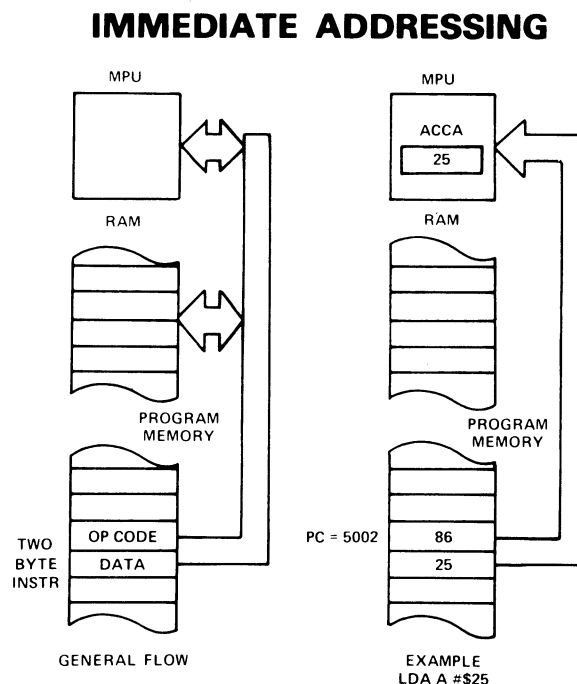


Fig. 4.2

**Direct**

Ved denne adresseringsmåde findes ADDRESSSEN på den memory adresse, som følger efter operationskoden. Dette gør, at man kan adressere de første 256 bytes af memory direkte (direct) (0000 → 00FF i hex). F.eks. vil en „hent til acc. A” (load acc. A) fra adresse F2 (hex) se således ud:

Memory adresse	Memory indhold
5004	96 (LDA A opcode)
5005	F2 (memory adresse, som indeholder data)

96 (i hex) er LDA A direct opcode. F2 (i hex) er den adresse, hvor-



fra data skal hentes, men da F2 kun svarer til en byte (8 bits), og en adresse skal være på 16 bits (2 bytes), må processoren selv forsyne adressen med indledende nulle. Den adresse, som processoren henter data fra, bliver da 00F2 (i hex).

Source input koden vil være:

```
LDA A $F2
```

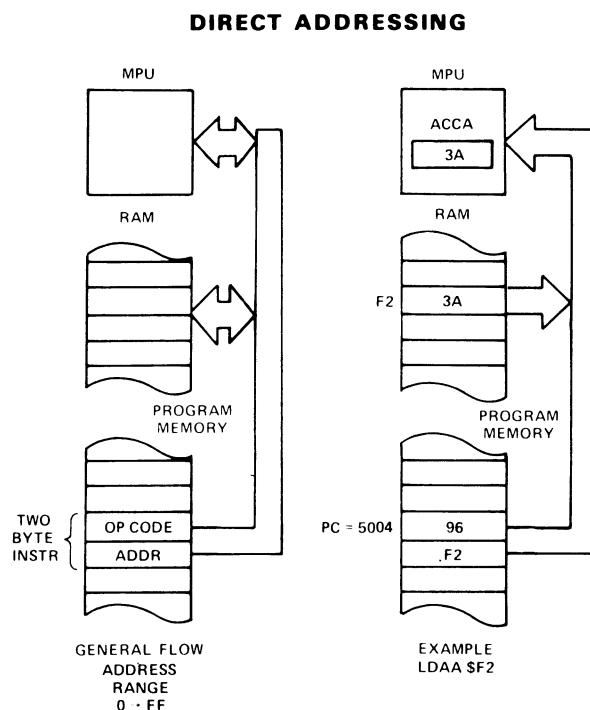


Fig. 4.3

## Extended

Denne adresseringsmetode bruges til at adressere sig frem til adresser over 00FF. Instruktionens anden memory celle indeholder de 8 mest betydende bits af adressen, og den tredje memory celle indeholder de 8 mindst betydende bits af adressen. F.eks. at hente til acc. A indholdet af adresse 2A84 (hex) vil se således ud:

Memory adresse	Memory indhold
5006	B6 (LDA A opcode)
5007	2A (adresse high byte)
5008	84 (adresse low byte)

B6 (hex) er LDA A extended opcode. 2A (hex) er den mest betydende halvdel af adressen og 84 (hex) er den mindst betydende halvdel af adressen, hvorfra data skal hentes. Efter udførelsen af ovennævnte instruktion vil indholdet af acc. A være det samme som indholdet af adresse 2A84.

Source input koden er:

```
LDA A $2A84
```

## EXTENDED ADDRESSING

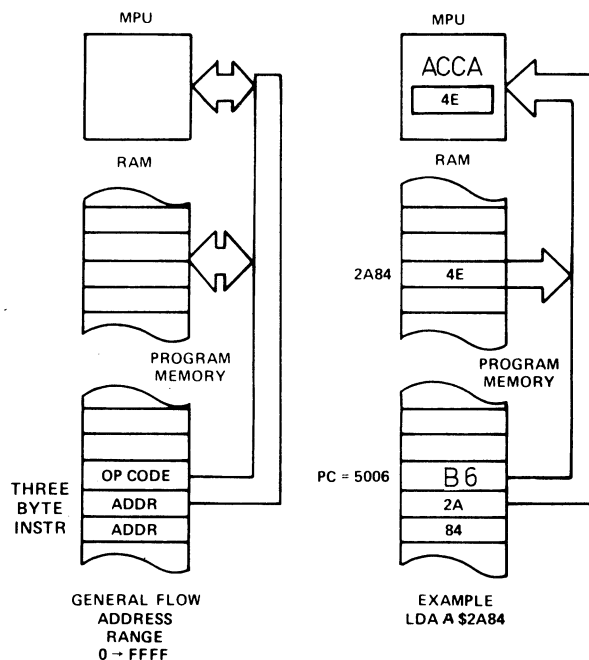


Fig. 4.4

## Indexed

Ved denne adresseringsmåde bliver det tal, som findes på instruktionens anden memory adresse adderet til indholdet af index registret og frembringer dermed en ny „effektiv adresse”. Den nye „effektive adresse” er den adresse, hvis data der skal arbejdes med.

Den effektive adresse opbevares i et midlertidigt adresseregister, så indholdet af index registret ikke bliver ødelagt eller ændret.

F.eks. hvis index registret indeholder værdien 1A00 (hex) og der udføres en „hent til acc. A” (load acc. A) instruktion med offset værdien 05 (hex), vil offset værdien 05 (hex) blive adderet til indholdet af index registret (1A00) og give en ny „effektiv adresse” på 1A05 (hex).

Memory adresse	Memory indhold
5009	A6 (LDA A opcode)
500A	05 (offset)

A6 (hex) er LDA A indexed opcode. 05 (hex) er offset værdien. For at danne den nye „effektive adresse” adderes offset værdien til index registrrets indhold  $1A00 + 05 = 1A05$ . Efter udførelsen af ovennævnte instruktion vil indholdet af acc. A være det samme som indholdet af adresse 1A05.

Source input koden vil være:

LDA A \$0,5X

↑                      ↙

offset                  indexeret adressering

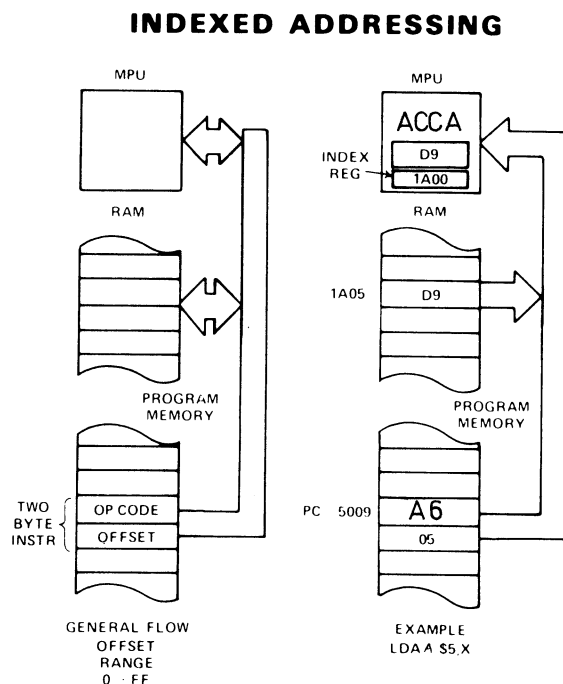


Fig. 4.5

### Relative

Denne adresseringsmåde bruges kun i forbindelse med **BRANCH** instruktioner. Branch benyttes i programmerne til at udføre **HOP** (afgrening). Hvis betingelsen for hoppet er opfyldt, vil processoren udføre hoppet, men hvis det ikke er opfyldt, vil programmet fortsætte med den instruktion, som kommer lige efter branch instruktionen. Hoppet som branch instruktionerne kan udføre er begrænset til 126 programbytes baglæns og 129 programbytes fremad. Da det er en 2-byte instruktion, vil hoppet altid referere til den næste instruktion, som processoren ville have udført, hvis den ikke hoppede, eller den nuværende lokation + 2. Den anden byte i instruktionen er offset værdien, det antal bytes, der skal springes, udtrykt som et 2-komplementtal.

Alle spring udføres ved at addere offset-værdien til programtællers indhold, efter at branch instruktionen og offset-værdien er læst.

For eksempel vil „hop hvis lig med nul” (Branch Equal to Zero = **BEQ**) kunne se således ud:

Memory adresse	Memory indhold
500B	27 ( <b>BEQ</b> opcode)
500C	15 (offset)
500D	

Hvis betingelsen ( $Z = 1$ ) er opfyldt, vil programtællerens nye indhold blive:

$$500D + 15 = 5022 \text{ (hex)}$$

Source input koden vil være:

Label	opcode	operand
	BEQ	LOOP
	...	
	...	
LOOP	LDA A	#\$03

↑  
næste opcode, hvis betingelsen er opfyldt.

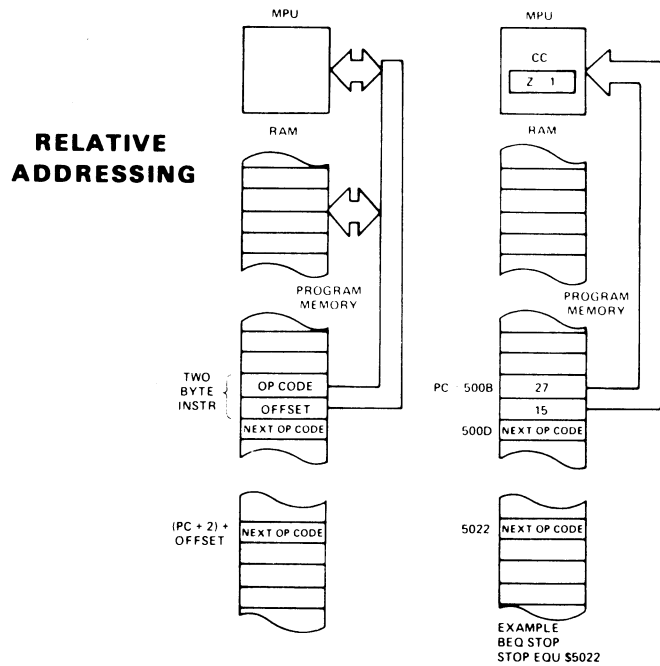


Fig. 4.6

- M
1. Læs det tal, der er lagret på adresse 0001. Addér 3 til tallet, og placér resultatet på adresse 0002. D
  - M 2. Læs det tal, der er lagret på adresse A001. Addér indholdet af adresse A002 til tallet, og placér resultatet på adresse A003. E×ī
  3. Skriv et program, som multiplicerer indholdet af adresse 0002 med 2,5. Placér resultatet på adresse 0003.  
Det antages, at tallet på adresse 0002 er lige.  
Hvor stor bliver fejlen, hvis tallet på adresse 0002 er ulige?
  - M 4. Læs det tal, som er lagret på adresse 001F. Træk \$17 fra tallet, og placér resultatet på adresse 002F.  
\$ betyder, at tallet er angivet på Hexadecimal form.
  - Ti 5. Læs det tal, der er lagret på adresse 0000. Rotér tallet 3 pladser til venstre, og placér resultatet på adresse 000F.
  - Ti 6. Skriv et program, som gennemfører en logisk AND operation imellem indholdet af adresse 0001 og adresse 0002. Resultatet placeres på adresse 0003.
  - Ti 7. Skriv et program, som gennemfører en logisk OR operation imellem indholdet af adresse 0001 og adresse 0002. Resultatet placeres på adresse 0003. (Logisk OR kaldes også Inclusive OR)
  - Ti 8. Skriv et program, som gennemfører en Exclusive OR operation imellem indholdet på adresse 0001 og adresse 0002. resultatet placeres på adresse 0003.
  - Ti 9. Læs det tal, som er lagret på adresse 00AC. 0-stil bit 0, 1, 4 og 5 i tallet, og placér resultatet på adresse 00AD.
  - Ti 10. Læs det tal, som er lagret på adresse 00AC. 1-stil bit 0, 1, 4 og 5 i tallet, og placér resultatet på adresse 00AD.



ADR	Code	LABEL	OPERATION	Mode	OPERAND	TITLE
0						
1		DATA 05				
2		RES				
3	86	Load Data	L D A A I			
4	03	03				
5	9B		A D O A D			
6	01					
7	97		S T A D			
8	02					
9	3F		S W I			
A						
B						
C			OK			
D						
E						
F						
20	B6	L	L D A A E		A001 3 DATA	
1	A0	<del>DATA</del>			A002 2 DATA	
2	01				A003 res	
3	BB		A D O A E			
4	A0	<del>DATA</del>				
5	0B					
6	B7		S T A A E			
7	A0	Res				
8	03					
9	3F		S W I			
A						
B						
C						
D			OK			
E						
F						
30		DATA				
1		Resultat				
2	96		L D A A D			
3	30					
4	16		A → B			
5	0C		C L C			
6	44		L S R A		x 0,5	
7	0C		C L C			
8	58		A S L B		x 2	
9	1B		A B A			
A	97		S T A A D			
B	31					
C	3F					
D						
E		Fejl ved	ulige		er lig K2	
F						
0	96		L D A A D		001F 20	
1	1F				002F res 9	
2	80		S U B A I			
3	17					
4	97		S T A A D			
5	2F					
6	3F		S W I			
7						
8						
9						
A			OK			
B						
C						
D						
E						
F						

ADR	Code	LABEL	OPERATION	OPERAND	TITLE
0	0C				0010 0101
1	0B		LDABD		2 5
2	0A				4 A
3	59		ROLB		R 01001010
4	59		ROLB		9 4
5	59		ROLB		R 10010100
6	D7		STABD		2 8
7	0F				R 00101000
8	3F				
9					
A					
B					
C					
D					
E					
000 F		RES			
0					
1		Data 1	43		
2		Data 2	52		
3		res	42		
4	96		LDABD		
5	01				
6	94		ANDAD		
7	02				
8	97		STAD		
9	03				
A	3F				
B					
C					
D					
E					
F					
0					
1		Data 1	2F		
2		Data 2	F2		
3		res	FF		
4	96		LDABD		
5	01				
6	9A		ORABD		
7	02				
8	97		STAD		
9	03				
A	3F				
B					
C					
D					
E					
F					
0					
1		Data 1			
2		Data 2			
3					
4	96				
5	01				
6	98				
7	02				
8	97				
9	03				
A	3F				
B					
C					
D					
E					
F					



[illegible]

Blad af



# opgave 2

ADR		OBJEKT KODE								PROGRAM																																		
HEX		BINÆR								HEX	LABEL						MNE					OPERAND										COMMENT												
		7	6	5	4	3	2	1	0		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32		
0010									B6										LDA	A																							4	
11									A0																																			
12									01																																			
13									BB										ADD	A																							4	
14									A0																																			
15									02																																			
16									B7										STA	A																							5	
17									A0																																			
18									03																																			
19									3F										SWI																								12	
Program fylder 9 lokationer, og bruger 13																																CS												
0020									CE										LDX																								3	
									A0																																			
									01																																			
									A6										LDA	A																							5	
									00																																			
									AB										ADD	A																							5	
									01																																			
									A7										STA	A																							6	
									02																																			
									3F										SWI																									
																																, og bruger 19	CS											

Dato  
Kons./Tegn. af  
Tegn.nr.

Blad af





## Program 4

[illegible]

Dato	
Kons./Tegn. af	
Tegn.nr.	

Blad af

opgave 5

ADR		OBJEKT KODE								PROGRAM																																														
HEX		BINÆR								HEX	LABEL							MNE							OPERAND																COMMENT															
		7	6	5	4	3	2	1	0		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32														
										96																																														
										00																																														
										49																																														
										49																																														
										49																																														
										97																																														
										0F																																														
										3F																																														

Opri ndel igt bitmønster : FO = 11110000

Resultat : 87 - 10000111

OBS: tolkning af tekst på side A-54  
: og hvad forstås ved ROTATE

: og hvad forstås ved ROTATE

Dato  
Kons./Tegn. af  
Tegn.nr.

Blad af



# opgave 6, 7 og 8

ADR		OBJEKT KODE								PROGRAM																																												
HEX		BINÆR								HEX	LABEL						MNE						OPERAND																COMMENT															
		7	6	5	4	3	2	1	0		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32												
0060										96									LDA	A																																		
61										01																																												
62										94									AND	A																																		
63										02																																												
64										97									STA	A																																		
65										03																																												
66										3F									SWI																																			

Opgave 7 : Lokation 62 → 9A

Opgave 8 : Lokation 62 → 98

Dato  
Kons./Tegn. af  
Tegn.nr.

Blad af

Opgave 9 og 10

ADR		OBJEKT KODE								PROGRAM																																											
HEX	BINÆR								HEX	LABEL								MNE				OPERAND																COMMENT															
	7	6	5	4	3	2	1	0		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32												
0090									96																																												
91									AC																																												
92									84																																												
93									CC																																												
94									97																																												
95									AD																																												
96									3F																																												

① pgave 10: Lokation 92 → 8A  
Lokation 93 → 33

Dato	Kons./Tegn. af	Tegn.nr.
------	----------------	----------

Blad af

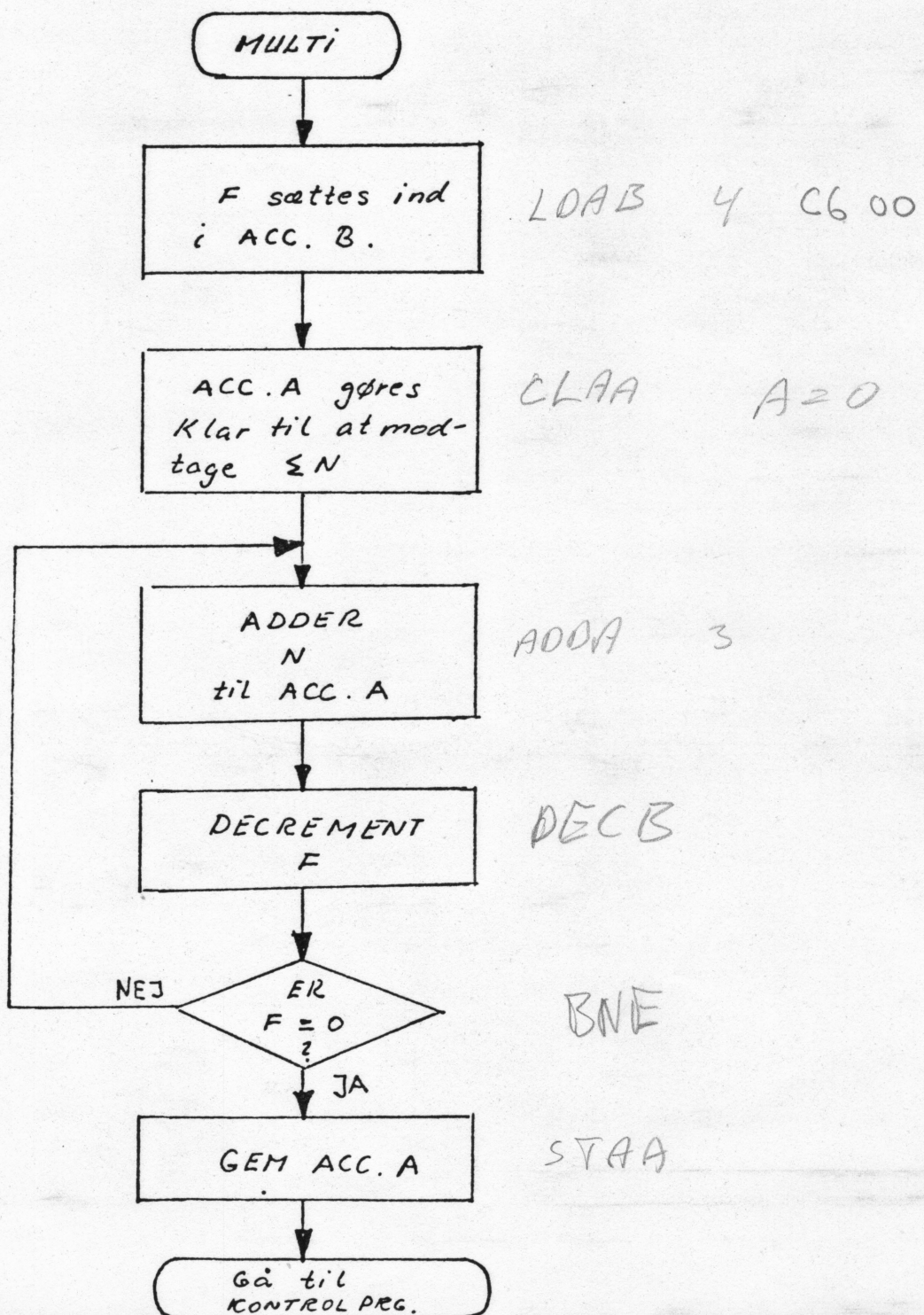


# MULTIPLICATIONS - PROGRAM.

„Algoritme“ :

$$N \times F = R \quad \Rightarrow \quad \underbrace{N + N + N \dots + N}_{F \text{ gange.}} = R$$

## RUTEDIAGRAM:



# ASSEMBLER LISTING:

LABEL	OPCODE	OPERAND	COMMENTS
MULTI	LDA B	\$ 01	Hent faktor F på \$01
	CLR A		Rens ACC. A
LOOP	ADD A	\$ 02	Adder N til ACC. A
	DEC B		$F = F - 1$
	BNE	LOOP	$F \neq 0$ ? JA $\Rightarrow$ GÅ TIL LOOP NEJ $\Rightarrow$ FORTSÆT.
	STA A	\$ 00	Gem Resultat på \$00
	JMP	\$ E08D	Gå til Kontrol program.

Følgende ADR. er reserveret:

- \$ 0000 : Resultatet af multiplikationen. (R)
- \$ 0001 : Den indlæste multiplikationsfaktor. (F)
- \$ 0002 : Den størrelse, der skal multipliceres. (N)

\_\_\_\_\_ o \_\_\_\_\_

„Håndassembler“ ovenstående program og afprøv det på kit 2.



ADR	Code	LABEL	OPERATION	OPERAND	TITLE
0		1200			
1		1. data			
2		2. data			
3	D6		LDABD		
4	01				
5	4F		CLRA		
6	9B		ADDAD		
7	02				
8	5A		DECB		
9	26		BNE		
A	F8	- 05			
B	97		STAA		
C	00				
D	3F				
E					
F					
0					
1					
2	FE	jump restant			
3					
4					
5					
6					
7					
8					
9					
A					
B					
C					
D					
E					
F					
0					
1					
2					
3					
4					
5					
6					
7					
8					
9					
A					
B					
C					
D					
E					
F					
0					
1					
2					
3					
4					
5					
6					
7					
8					
9					
A					
B					
C					
D					
E					
F					

0111 1001

0111 1101

1111 0110

~~1111~~

0000 1001

1000 0011

0111 1101

83

<sup>4</sup>  
-16+8

128+

-125

7 3

112

13

-125

L DAA # 30 H/V

L

1010 1101

0101 0011

12x16

14x16+

224

131

5

3

4

9



ADR	Code	LABEL	OPERATION	Mode	OPERAND	TITLE
0	96		L D A A	D		
1	AC					
2	84		A N D A	I		
3	CC					
4	97		S T A A	D		
5	AD					
6	3F					
7						
8						
9						
A						
B						
A C	55	Data				
D	44	resultat				
E						
F						
0	96		L D A A	D		
1	AC					
2	8A		O R A A	I		
3	33					
4	97		S T A A	D		
5	AD					
6	3F					
7						
8						
9						
A						
B						
C						
D						
E						
F						
0						
1						
2						
3						
4						
5						
6						
7						
8						
9						
A						
B						
C						
D						
E						
F						
0	86		L D A A	I		
1	0A					
2	C6		L D A B	I		
3	0B					
4	37		P S H B			
5	16		T A B			
6	32		P U L A			
7	3F					
8						
9						
A						
B						
C						
D						
E						
F						

✓

✓

Register i stack pointer  
 Register i B register  
 stack pointer i Register

~~1000~~

ST B 07

TBA  
JSR \$E29A

STABE \$A011

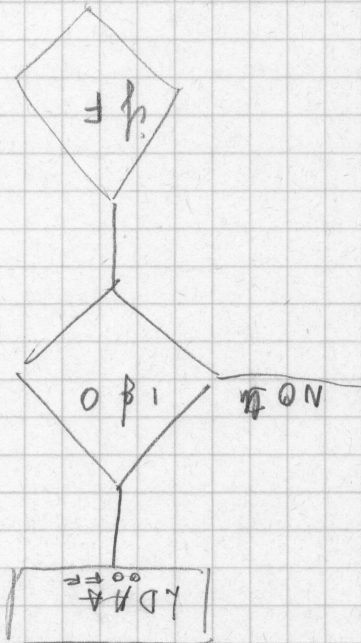
STAAE \$A010

JMP E \$E0FF

///

8020  
8022

HELP US



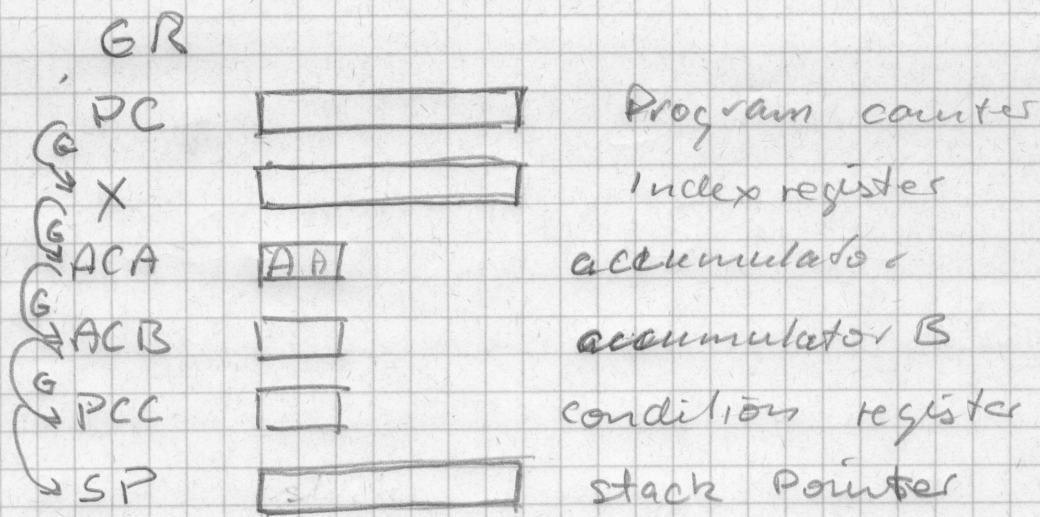
00FF



ADR	Code	LABEL	OPERATION	Mode	OPERAND	TITLE
0	96		LDAA	D		
1	FF					
2	26		BNE	R	BNE	Branch T zero
3	06	offset				offset
4	C6		LDAB	I		
5	FF					
6	D7		STAB	D		
7	FE					
8	20		BRA	R		Branch
9	0D	offset				offset
A	42		COMA			Complement A
B	26		BNE		BNE	
C	06	offset				
D	C6		LDAB	I		
E	00					
F	D7		STAB	D		
0	FE					
1	20		BRA	R		
2	04	offset				
3	C6		LDAB	I		
4	07					FF = 00 OK
5	D7		STAB	D		00 = FF
6	FE					36 = 07
7	3F					
8						
9						
A						
B						
C						
D						
E		00 FF 07				
F		Hex tail mal				
0						
1						
2						
3						
4						
5						
6						
7	F7		TBR			
8	BD		JSR			
9	F2					
A	9A					
B	F7		STAB	E		
C	A0					
D	11					
E	B7		STAB	E		
F	A0					
0	10					
1	7E		JMP			
2	E0					
3	FE					
4						
5						
6						
7						
8						
9						
A						
B						
C						
D						
E						
F						


Display routine  
over



12





3F SWI soft ware interrupt

Singel step

Adresse start  breakpoint

 escape start adresse 

program afvikling i singel step

ved  aktivering med  kan

man se alle registre i rækkefølge



LDA Extended  
MSB LSB

LDA \$ 1F00

24

1 + at  
resultant

2/b 0000 1000

1' 1

Load A

copier A + B

Roter A til Højre  $\times 5$

Roter B til venstre  $\times 2$

AD A + B = res

8  $\times$  2.5

16

4 = 11

10

0000 1000  $\times 12$   
 + 0001 0000  
 + 0000 0100  $\times 0.5$   
 -----  
 0001 0100

9

1 46  
 16  
 -----  
 20

16  $\frac{9 \times}{18}$

4.5

16  $\frac{1}{2}$   $\frac{22.5}{16}$

ADRESSERING	INHERENT (IMPLIED)	IMMEDIATE	DIRECT	EXTENDED	INDEXED	RELATIVE
Betyder oversat til dansk	Indeholdt adr.	Nærmeste adr.	Direkte adr.	Fuld adresse	Indexeret adr. (Indirekte adr.)	Relativ adr.
INSTRUK- TIONENS STØRRELSE	<u>OPCODE</u>  (1 BYTE)	<u>OPCODE</u> <u>DATA</u>  (2 BYTES) <sub>24, 16, 8</sub> (enkelte 3 BYT.)	<u>OPCODE</u> <u>ADDRESS (Lo)</u>  (2 BYTES)	<u>OPCODE</u> <u>ADDRESS (Hi)</u> <u>ADDRESS (Lo)</u>  (3 BYTES)	<u>OPCODE</u> <u>ADR.OFFSET</u>  (2 BYTES)	PC: <u>OPCODE</u> (BRANCH) OFFSET OPCODE (1) OPCODE (2) (2 BYTES) 
ANVENDELSE	INSTRUKTION- EN INDEHOL- DER DE NØD- VENDIGE OPLYSNINGER	DATA ligger umiddel- bart efter OPCODE i PROGRAM MEMORY	Benyttes til at ADRESSERE side 0, d.v.s. ADRESSE fra (00)00 - (00)FF	Benyttes til at ADRESSERE HELE ADR. OMRÅDET fra 0000-FFFF	Benytter INDEX REGISTER SOM GRUNDTAL og adresser OFFSET til IX for at få ADR	Benyttes kun til BRANCH Hvis BRANCH betingelse er OPFYLDT hoppes til PC+ 2+OFFSET i programmet (2) Ellers fortsættes med næste OPCODE (1)
DATA HENTES FRA	CPU Manipulering d.v.s. der hentes ikke data fra ydre lager	Programmet lige efter OPCODEN	Den angivne ADR. på side 0	Den angivne fulde ADR.	ADR=IX+OFFSET NB: OFFSET fra 0 til 255(10) (00 → FF)	NB: OFFSET fra -128 til +127(10) (80 → 7F) angives som 2 komplement



# OPGAVER TIL KAP. 4 PROGRAMMERING:

OPG. 4.1) Hvilke forskellige adresseringsmetoder er der brugt i nedenstående rutine for

BINÆR til BCD konvertering ?

*Extend Direct Immediate Relative Implied*

OPG. 4.2) Hvilke RAM adresser benyttes til DATA i

dette program ?

*IB data IC res Ent Huns BCD  
ID Hundrede BCD*

OPG. 4.3) Indtast programmet i KIT II og prøv om

det vil virke med det binære tal: 1000 0111<sup>(2)</sup>

(Prøv evt. selv med andre tal).

```

00009      ;
00010      ;
00011      ; *****
00012      ; * SUBROUTINE FOR BINARY TO BCD CONVERSION *
00013      ; *   SUBTRACT 100 & 10 ECT.ALGORITHM   *
00014      ; *****
00015      ;
00016      ; BINARY--> ADDRESS : DPLDTA
00017      ; HUNDREDS-> - : HUNBCD
00018      ; UNITS,TENS - : UNTBCD
00019      ;
00020      ;
00021 0100 7F001D BTBCD CLR HUNBCD E 0010
00022 0103 961B LDA A DPLDTA ;BINARY TO ACC.A D IB
00023      ;
00024      ;
00025 0105 8164 HUND CMP A #100 ;ACC.A < BINARY IMI 64 data
00026 0107 2507 BCS TENS ;YES! GO TO TENS R 07 data
00027 0109 8064 SUB A #100 ;SUBTRACT 100 IM 0100
00028 010B 7C001D INC HUNBCD ;INCREMENT HUNDREDS E 101D
00029 010E 20F5 BRA HUND R 103
00030 0F      ;
00031      ;
00032 0110 971C TENS STA A UNTBCD D IC
00033 0112 16 TAB ;ACC.A -> B IMP
00034 0113 800A IGEN SUB A #10 ;"BINARY" < 10 ? IMI
00035 0115 2504 BCS DONE ;YES! GO TO DONE! 041B
00036 0117 CB06 ADD B #6 ;NO! ADJUST BY IMI 06B
00037 0119 20F8 BRA IGEN ; ADDING 6
00038 011B D71C DONE STA B UNTBCD ;
00039 011D 7EE08D JMP BACK
00040      ;
00041      ;
00042      END

```

*Handwritten notes and arrows:*

- Arrows pointing to lines 00025, 00026, 00027, 00028, 00029, 00032, 00033, 00034, 00035, 00036, 00037, 00038, 00039.
- Handwritten "3F" next to line 00039.
- Handwritten "101D" next to line 00028.
- Handwritten "041B" next to line 00035.
- Handwritten "06B" next to line 00036.

## OPGAVELØSNINGER TIL KAP. 4 PROGRAMMERING:

4.1) Der er brugt følgende adresseringsmetoder:

EXTENDED ADRESSERING		f.eks. på Adr.	Ø100 (7F 001D )
			CLR HUNBCD
DIRECT	- " -	f.eks. på Adr.	Ø103 (96 1B )
			LDA A DPLDTA
IMMEDIAT	- " -	f.eks: på Adr.	Ø105 ( 81 64 )
			CMP A 100
RELATIVE	- " -	f.eks. på Adr.	Ø107 (25 07 )
			BCS TENS
INHERENT	- " -	f.eks. på Adr.	Ø112 ( 16 )
			TAB

4.2) Følgende RAM adresser benyttes af programmet:

ADR.	LABEL	INDHOLD
001B	DPLDTA	Binært input
001C	UNTBCD	Enere & tiere BCD
001D	HUNBCD	Hundreder BCD

4.3) Når programmet er indlæst i KIT II, indlæses det binære tal: 1000 0111<sub>(2)</sub> som HEX tal: 87 på adresse 001B .

Derefter startes programmet ved at indtaste programmets start adresse: 0100 og derefter den blå tast "G"

Når der fremkommer en vandret streg længst til venstre på display'et, er programmet færdigt.

Resultatet af konverteringen kan læses ved at kalde følgende adresser:

ADR.	INDHOLD	
001D	01	antal hundreder
001C	35	" tiere & enere

ADR	Code	LABEL	OPERATION	Mode	OPERAND	FILE
0	86		LDA A	1		
1	01		01			
2	CE		LDX			
3	02		02			
4	00		00			
5	09		DEX			
6	A7		STAA	IND		
7	00		00			
8	09		DEX			
9	26		BNE			
A	FB		FB			
B	A7		STAA			
C	00		00			
D	A1		CMP			
E	00		00			
F	26		BNE			
0	0B		0B			
1	63		COM			
2	00		00			
3	08		INX			
4	8C		GRX			
5	03	01	01			
6	00	FF	FF			
7	26		BNE			
8	F4		F4			
9	48		ASL			
A	24		BCC			
B	E9		E9			
C	3F		SWI			
D						
E						
F						
0						
1						
2						
3						
4						
5						
6						
7						
8						
9						
A						
B						
C						
D						
E						
F						
0						
1						
2						
3						
4						
5						
6						
7						
8						
9						
A						
B						
C						
D						
E						
F						

} highest addr + 1

} highest addr

A QUICK MARCH CHECKS MEMORY



## A quick march checks memory

by A. James Laurino  
Wilmington, Del.

While forms of what is known as the march test are commonly used by chip manufacturers to check whether a random-access memory has its full complement of independent storage cells, users have no simple way of detecting RAM faults with a microprocessor. The 14-line version of the march test presented here provides an easy way to march memory to check for problems using the 6800 microprocessor. It can also be easily adapted for any other computer system.

This test will show whether each bit location in memory is capable of storing both 1s and 0s and whether storing any bit in a given location causes a change in the other locations. The test may not detect marginal failures, but there is no single, simple worst-case test for all systems.

The program makes two passes through the address range for each bit in the memory word (for example, 16 passes for an 8-bit machine). Test patterns are produced by shifting a single logic-1 bit through the accumulator to see if the bits within each word are independent, and that each bit location can store both 1s and 0s.

The first pass for each pattern stores that pattern in every location. On the second pass each location is read to see if there has been any change. If there has been no change, the program marks that location to show that it has been tested, by performing a logic complement operation. If one location responds to two addresses, then the complement pattern will be found when the test reaches the second address affected.

The program is entered at START, which must be an address above the range to be tested. The address range extends from 0 to the value of Top - 1 (e.g. Top must be 1000 hex to test addresses 0 through 0FFF hex). Upon entry, accumulator A should contain 1, and the index register (X) should contain the value of Top.

The locations in the address range to be tested are then filled with the first test pattern, generated by program lines 1 through 5. Each location is checked for a proper pattern and appropriately marked during lines 6 through 11. The next test pattern is generated and the test repeated until all patterns have been used (lines 12 and 13). At line 14, the test is concluded and the results reported.

If the memory is good, then register X will contain the value of Top and A will be zero. If a failure occurs, A will contain the failing pattern and X will have the failing address.

For example, consider the case where address line 5 (i.e.,  $2^5$  bit) is stuck. The first failure would occur at address 20 hex, which would contain FE hex instead of the expected value of 1. □

6800 MEMORY CHECK PROGRAM

Label	Source statement			Comments
Start	DEX		09	Get the highest address to be tested
Fill	STA	A X	A7 00	Put the pattern in the current location
	DEX		09	Go to the next location
	BNE	Fill	26 FB	Continue through the address range
Test	STA	A X	A7 00	Put the pattern in the last location
	CMP	A X	A1 00	Check pattern in the current location
	BNE	End	26 0B	Stop if the pattern is not correct
	COM	X	63 00	Mark the location as tested
	INX		08	Go to the next location
	CPX	#Top	8C -- --	Fill in the upper address limit here
	BNE	Test	26 F4	Continue through the address range
	ASL	A	48	Shift the one left to the next bit
	BCC	Start	24 E9	Repeat the test until all bits are used
End	SWI		3F	Return to monitor and display registers

ID

A

B

X

PC

SP

H I N Z V C

CC

11

0000

1

2

3

4

5

6

7

8

9

A

B

C

D

E

F

001 0 4F

1

2

3

4

5

6

7

8

9

A

B

C

D

E

F

002 0 02

1

2

3

4

5

6

7

8

9

A

B

C

D

E

F

003 0

1

2

3

4

5

6

7

8

9

A

B

C

D

E

F

CLRA

LDAB I

05

STABD

05

LDX I

00

28

ADDA IND

05

LDS IM

00

3D

BSR R

05

WAI

PSHA

LDAA I

05

DECA

BNE R

FD

PULA

RTS

03

02

07

MSB

LSB

offset

H MSB

L LSB

offset

interrupt

Navn:

B Hingeberg Pedersen

Brønche:

Klasse- og elev nr.:

MODE: X Indexed E Extended H Shift I Immediate

## MC6800

TABLE 4 – INDEX REGISTER AND STACK MANIPULATION INSTRUCTIONS

POINTER OPERATIONS	MNEMONIC	IMMED			DIRECT			INDEX			EXTND			IMPLIED			BOOLEAN/ARITHMETIC OPERATION	COND. CODE REG.					
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#		5	4	3	2	1	0
																		H	I	N	Z	V	C
Compare Index Reg	CPX	8C	3	3	9C	4	2	AC	6	2	BC	5	3				$X_H - M, X_L - (M + 1)$	•	•	⑦	↑	⑧	•
Decrement Index Reg	DEX													09	4	1	$X - 1 \rightarrow X$	•	•	•	↑	•	•
Decrement Stack Pntr	DES													34	4	1	$SP - 1 \rightarrow SP$	•	•	•	•	•	•
Increment Index Reg	INX													08	4	1	$X + 1 \rightarrow X$	•	•	•	↑	•	•
Increment Stack Pntr	INS													31	4	1	$SP + 1 \rightarrow SP$	•	•	•	•	•	•
Load Index Reg	LDX	CE	3	3	DE	4	2	EE	6	2	FE	5	3				$M \rightarrow X_H, (M + 1) \rightarrow X_L$	•	•	⑨	↑	R	•
Load Stack Pntr	LDS	8E	3	3	9E	4	2	AE	6	2	BE	5	3				$M \rightarrow SP_H, (M + 1) \rightarrow SP_L$	•	•	⑨	↑	R	•
Store Index Reg	STX				DF	5	2	EF	7	2	FF	6	3				$X_H \rightarrow M, X_L \rightarrow (M + 1)$	•	•	⑨	↑	R	•
Store Stack Pntr	STS				9F	5	2	AF	7	2	BF	6	3				$SP_H \rightarrow M, SP_L \rightarrow (M + 1)$	•	•	⑨	↑	R	•
Indx Reg → Stack Pntr	TXS													35	4	1	$X - 1 \rightarrow SP$	•	•	•	•	•	•
Stack Pntr → Indx Reg	TSX													30	4	1	$SP + 1 \rightarrow X$	•	•	•	•	•	•

TABLE 5 – JUMP AND BRANCH INSTRUCTIONS

OPERATIONS	MNEMONIC	RELATIVE			INDEX			EXTND			IMPLIED			BRANCH TEST	COND. CODE REG.					
		OP	~	#	OP	~	#	OP	~	#	OP	~	#		5	4	3	2	1	0
															H	I	N	Z	V	C
Branch Always	BRA	20	4	2										None	•	•	•	•	•	•
Branch If Carry Clear	BCC	24	4	2										$C = 0$	•	•	•	•	•	•
Branch If Carry Set	BCS	25	4	2										$C = 1$	•	•	•	•	•	•
Branch If = Zero	BEQ	27	4	2										$Z = 1$	•	•	•	•	•	•
Branch If ≥ Zero	BGE	2C	4	2										$N \oplus V = 0$	•	•	•	•	•	•
Branch If > Zero	BGT	2E	4	2										$Z + (N \oplus V) = 0$	•	•	•	•	•	•
Branch If Higher	BHI	22	4	2										$C + Z = 0$	•	•	•	•	•	•
Branch If ≤ Zero	BLE	2F	4	2										$Z + (N \oplus V) = 1$	•	•	•	•	•	•
Branch If Lower Or Same	BLS	23	4	2										$C + Z = 1$	•	•	•	•	•	•
Branch If < Zero	BLT	2D	4	2										$N \oplus V = 1$	•	•	•	•	•	•
Branch If Minus	BMI	28	4	2										$N = 1$	•	•	•	•	•	•
Branch If Not Equal Zero	BNE	28	4	2										$Z = 0$	•	•	•	•	•	•
Branch If Overflow Clear	BVC	26	4	2										$V = 0$	•	•	•	•	•	•
Branch If Overflow Set	BVS	29	4	2										$V = 1$	•	•	•	•	•	•
Branch If Plus	BPL	2A	4	2										$N = 0$	•	•	•	•	•	•
Branch To Subroutine	BSR	8D	8	2											•	•	•	•	•	•
Jump	JMP				6E	4	2	7E	3	3				See Special Operations	•	•	•	•	•	•
Jump To Subroutine	JSR				AD	8	2	BD	9	3					•	•	•	•	•	•
No Operation	NOP										01	2	1	Advances Prog. Cntr. Only	•	•	•	•	•	•
Return From Interrupt	RTI										38	10	1		•	•	•	•	•	•
Return From Subroutine	RTS										39	5	1	See Special Operations	•	•	•	•	•	•
Software Interrupt	SWI										3F	12	1		•	•	•	•	•	•
Wait for Interrupt*	WAI										3E	9	1		•	•	•	•	•	•

\*WAI puts Address Bus, R/W, and Data Bus in the three-state mode while VMA is held low.

- |  |   |
|--|---|
| 1 (Bit V) Test: Result = 10000000?   | 7 (Bit N) Test: Sign bit of most significant (MS) byte = 1?   |
| 2 (Bit C) Test: Result = 00000000?   | 8 (Bit V) Test: 2's complement overflow from subtraction of MS bytes?   |
| 3 (Bit C) Test: Decimal value of most significant BCD Character greater than nine?<br>(Not cleared if previously set.) | 9 (Bit N) Test: Result less than zero? (Bit 15 = 1)   |
| 4 (Bit V) Test: Operand = 10000000 prior to execution?   | 10 (All) Load Condition Code Register from Stack. (See Special Operations)  |
| 5 (Bit V) Test: Operand = 01111111 prior to execution?   | 11 (Bit I) Set when interrupt occurs. If previously set, a Non-Maskable Interrupt is required to exit the wait state. |
| 6 (Bit V) Test: Set equal to result of $N \oplus C$ after shift has occurred.  | 12 (All) Set according to the contents of Accumulator A.  |



## MC6800

TABLE 3 — ACCUMULATOR AND MEMORY INSTRUCTIONS

		ADDRESSING MODES										BOOLEAN/ARITHMETIC OPERATION	COND. CODE REG.																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																				
		IMMED			DIRECT			INDEX			EXTND			IMPLIED			(All register labels refer to contents)	5	4	3	2	1	0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																										
OPERATIONS		MNEMONIC	OP		~		=		OP		~		=		OP		~		=		OP		~		=		H	I	N	Z	V	C																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																	
Add		ADDA	38	2	2	9B	3	2	AB	5	2	BB	4	3																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																			

## LEGEND:

OP Operation Code (Hexadecimal);  
 ~ Number of MPU Cycles;  
 = Number of Program Bytes;  
 + Arithmetic Plus;  
 - Arithmetic Minus;  
 • Boolean AND;  
 Msp Contents of memory location pointed to by Stack Pointer;

+ Boolean Inclusive OR;  
 ⊕ Boolean Exclusive OR;  
 M Complement of M;  
 → Transfer Into;  
 0 Bit = Zero;  
 00 Byte = Zero;

## CONDITION CODE SYMBOLS:

H Half-carry from bit 3;  
 I Interrupt mask;  
 N Negative (sign bit);  
 Z Zero (byte);  
 V Overflow, 2's complement;  
 C Carry from bit 7;  
 R Reset Always;  
 S Set Always;  
 ↑ Test and set if true, cleared otherwise;  
 • Not Affected

Note - Accumulator addressing mode instructions are included in the column for IMPLIED addressing



00	*		40	NEG	A	80	SUB	A	IMM	C0	SUB	B	IMM
01	NOP		41	*		81	CMP	A	IMM	C1	CMP	B	IMM
02	*		42	*		82	SBC	A	IMM	C2	SBC	B	IMM
03	*		43	COM	A	83	*			C3	*		
04	*		44	LSR	A	84	AND	A	IMM	C4	AND	B	IMM
05	*		45	*		85	BIT	A	IMM	C5	BIT	B	IMM
06	TAP		46	ROR	A	86	LDA	A	IMM	C6	LDA	B	IMM
07	TPA		47	ASR	A	87	*			C7	*		
08	INX		48	ASL	A	88	EOR	A	IMM	C8	EOR	B	IMM
09	DEX		49	ROL	A	89	ADC	A	IMM	C9	ADC	B	IMM
0A	CLV		4A	DEC	A	8A	ORA	A	IMM	CA	ORA	B	IMM
0B	SEV		4B	*		8B	ADD	A	IMM	CB	ADD	B	IMM
0C	CLC		4C	INC	A	8C	CPX	A	IMM	CC	*		
0D	SEC		4D	TST	A	8D	BSR		REL	CD	*		
0E	CLI		4E	*		8E	LDS		IMM	CE	LDX		IMM
0F	SEI		4F	CLR	A	8F	*			CF	*		
10	SBA		50	NEG	B	90	SUB	A	DIR	D0	SUB	B	DIR
11	CBA		51	*		91	CMP	A	DIR	D1	CMP	B	DIR
12	*		52	*		92	SBC	A	DIR	D2	SBC	B	DIR
13	*		53	COM	B	93	*			D3	*		
14	*		54	LSR	B	94	AND	A	DIR	D4	AND	B	DIR
15	*		55	*		95	BIT	A	DIR	D5	BIT	B	DIR
16	TAB		56	ROR	B	96	LDA	A	DIR	D6	LDA	B	DIR
17	TBA		57	ASR	B	97	STA	A	DIR	D7	STA	B	DIR
18	*		58	ASL	B	98	EOR	A	DIR	D8	EOR	B	DIR
19	DAA		59	ROL	B	99	ADC	A	DIR	D9	ADC	B	DIR
1A	*		5A	DEC	B	9A	ORA	A	DIR	DA	ORA	B	DIR
1B	ABA		5B	*		9B	ADD	A	DIR	DB	ADD	B	DIR
1C	*		5C	INC	B	9C	CPX		DIR	DC	*		
1D	*		5D	TST	B	9D	*			DD	*		
1E	*		5E	*		9E	LDS		DIR	DE	LDX		DIR
1F	*		5F	CLR	B	9F	STS		DIR	DF	STX		DIR
20	BRA	REL	60	NEG		A0	SUB	A	IND	E0	SUB	B	IND
21	*		61	*		A1	CMP	A	IND	E1	CMP	B	IND
22	BHI	REL	62	*		A2	SBC	A	IND	E2	SBC	B	IND
23	BLS	REL	63	COM	IND	A3	*			E3	*		
24	BCC	REL	64	LSR	IND	A4	AND	A	IND	E4	AND	B	IND
25	BCS	REL	65	*		A5	BIT	A	IND	E5	BIT	B	IND
26	BNE	REL	66	ROR	IND	A6	LDA	A	IND	E6	LDA	B	IND
27	BEQ	REL	67	ASR	IND	A7	STA	A	IND	E7	STA	B	IND
28	BVC	REL	68	ASL	IND	A8	EOR	A	IND	E8	EOR	B	IND
29	BVS	REL	69	ROL	IND	A9	ADC	A	IND	E9	ADC	B	IND
2A	BPL	REL	6A	DEC	IND	AA	ORA	A	IND	EA	ORA	B	IND
2B	BMI	REL	6B	*		AB	ADD	A	IND	EB	ADD	B	IND
2C	BGE	REL	6C	INC	IND	AC	CPX		IND	EC	*		
2D	BLT	REL	6D	TST	IND	AD	JSR		IND	ED	*		
2E	BGT	REL	6E	JMP	IND	AE	LDS		IND	EE	LDX		IND
2F	BLE	REL	6F	CLR	IND	AF	STS		IND	EF	STX		IND
30	TSX		70	NEG	EXT	B0	SUB	A	EXT	F0	SUB	B	EXT
31	INS		71	*		B1	CMP	A	EXT	F1	CMP	B	EXT
32	PUL	A	72	*		B2	SBC	A	EXT	F2	SBC	B	EXT
33	PUL	B	73	COM	EXT	B3	*			F3	*		
34	DES		74	LSR	EXT	B4	AND	A	EXT	F4	AND	B	EXT
35	TXS		75	*		B5	BIT	A	EXT	F5	BIT	B	EXT
36	PSH	A	76	ROR	EXT	B6	LDA	A	EXT	F6	LDA	B	EXT
37	PSH	B*	77	ASR	EXT	B7	STA	A	EXT	F7	STA	B	EXT
38	*		78	ASL	EXT	B8	EOR	A	EXT	F8	ADC	B	EXT
39	RTS		79	ROL	EXT	B9	ADC	A	EXT	F9	ADC	B	EXT
3A	*		7A	DEC	EXT	BA	ORA	A	EXT	FA	ORA	B	EXT
3B	RTI		7B	*		BB	ADD	A	EXT	FB	ADD	B	EXT
3C	*		7C	INC	EXT	BC	CPX		EXT	FC	*		
3D	*		7D	TST	EXT	BD	JSR		EXT	FD	*		
3E	WAI		7E	JMP	EXT	BE	LDS		EXT	FE	LDX		EXT
3F	SWI		7F	CLR	EXT	BF	STS		EXT	FF	STX		EXT

Notes: 1. Addressing Modes: A = Accumulator A IMM = Immediate  
B = Accumulator B DIR = Direct  
REL = Relative  
IND = Indexed

2. Unassigned code indicated by "\*\*\*\*".

TABLE 3-1. Hexadecimal Values of Machine Codes



LDA # \$20

Hex

immediate adr

86 20

LDA 20

direct page 0 adr 96 20

LDA 0020

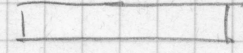
extended 16 bit adr B6 00 20

LDA X 00

indexed adr

A6 00

index reg



first shal X Loader

LDX 02F4

CE 02 F4

offset via index reg

Branch relative

LDA # \$XX

ROL B

BCS Branch if carry set. ZS + offset

STB # \$00

TBA

F  
5

C6

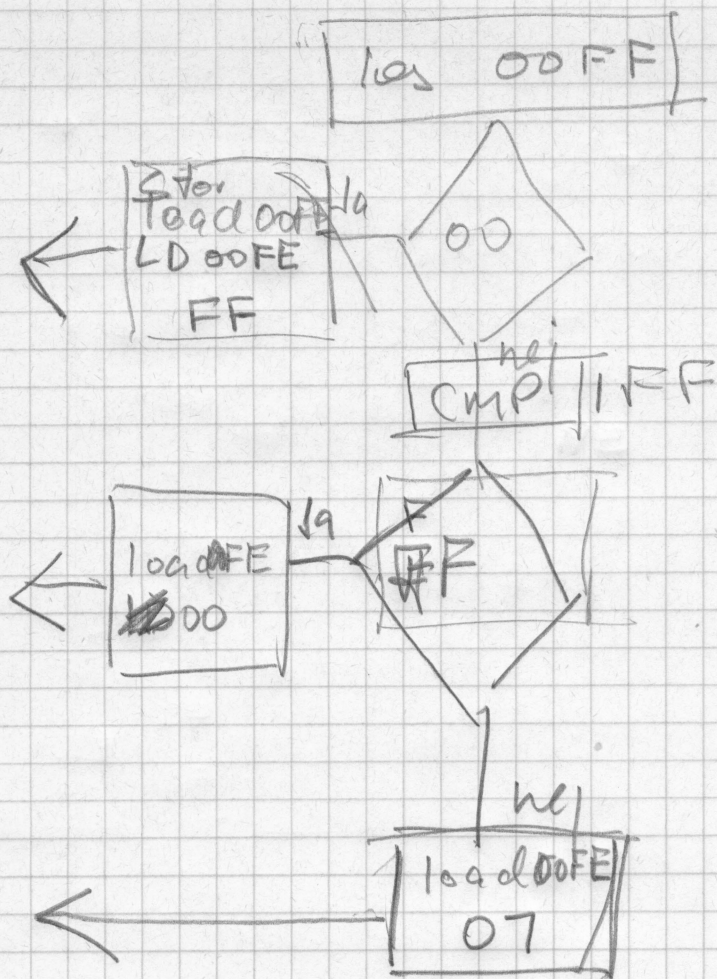
- 5

0000 0101

0111 1011

F

B



ikke gå fra decision  
til decision uden  
at opdatere status  
register gøres ved  
at compare med  
indgang til decision

input

lager adr	FF
" "	00
" "	07

FFFF 8

E014

FFFA

E032

FFF 6

E019

FFF E

E08d

## PROGRAMUDFØRELSE

Vi har tidligere set på, hvorledes et program kan udvikles (skrives) ved hjælp af en assembler. Det maskinprogram, som er kommet ud af assembleren (object program), indkodes nu i en ROM og er klar til at blive udført. Udførelsen af programmet sker på følgende måde:

CPU'en henter en instruktion fra lageret (ROM) ad gangen, efterfulgt af en undersøgelse (dekodning) af instruktionen. Resultatet af dekodningen angiver CPU'ens næste skridt, f.eks. overfører en data-værdi fra lageret via databussen til et af CPU'ens registre (Akkumulator). Når instruktionen er udført, henter CPU'en den næste instruktion i lageret o.s.v.

Programmet er normalt indlæst i en ROM, derved sikrer man sig, at dette ikke forsvinder, når der slukkes for forsyningsspændingen.

Microprocessorens arbejdsmåde kan anskueliggøres ved hjælp af en lille programstump, hvor to tal adderes.

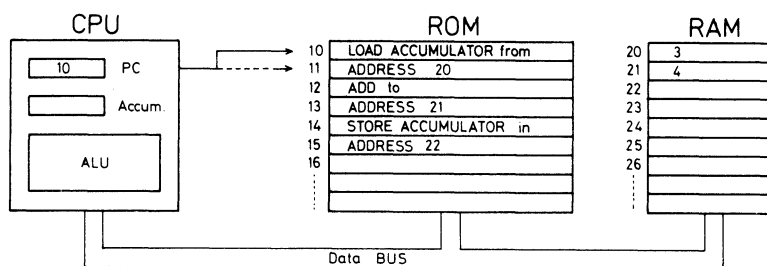



Fig. 5.1 

Her ses programmet indlæst i ROM-lageret og data i RAM. Først indstilles programtælleren (PC) på den første instruktion. Denne dekoderes til operationen "Load accumulator". Processoren skal nu vide, hvor data skal hentes, hvorfor PC øges (11), således at dataadressen kan læses af CPU'en.

CPU'en udfører nu instruktionen ved at udpege data (3) i RAM-lageret (adr. 20) og indlæse denne værdi i akkumulatoren.

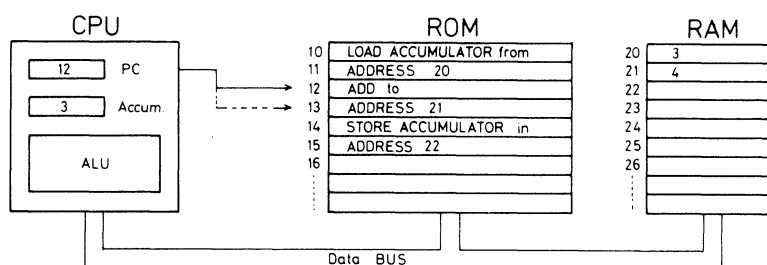



Fig. 5.2 

Programtælleren øges (12), og den næste instruktion (Adder til) læses og dekoderes af CPU'en. PC øges igen (13) og adressekoden (21)

læses. Med denne adresse udpeges data (4), som via databussen tilføres ALU'en samtidig med, at akkumulatorindholdet (3) tilføres. ALU'en udfører additionen, og resultatet gemmes i akkumulatoren.

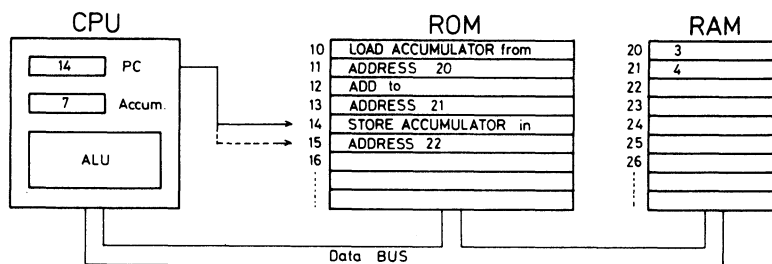


Fig. 5.3

PC øges (14), næste instruktion hentes til CPU'en og dekodes. PC øges igen og adresse (22) hentes. Akkumulatorindholdet føres nu via databussen til RAM og indlæses på adresse 22. PC øges (16) o.s.v., o.s.v.

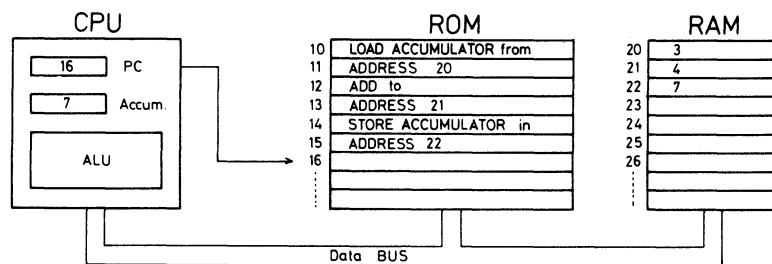


Fig. 5.4

Ovenstående program vil i ROM'en bestå af 0'ere og 1'ere.

Maskinkode	Hex-tal	Assemblerkode
ROM		
10 1 0 0 1 0 1 1 0	96	LDA \$ 20
11 0 0 1 0 0 0 0 0	20	
12 1 0 0 1 1 0 1 1	9B	ADD \$ 21
13 0 0 1 0 0 0 0 1	21	
14 1 0 0 1 0 1 1 1	97	STO \$ 22
15 0 0 1 0 0 0 1 0	22	
16		

Fig. 5.5

## INSTRUKTIONSEKVENNS

En INSTRUKTIONSEKVENNS (INSTRUCTION CYCLE) er det, der skal foregå for at HENTE (FETCH) og UDFØRE (EXECUTE) en enkelt instruktion.

Instruktions-sekvensen kan måles i et antal PROGRAM BYTES (#) og vil for M6800 være fra 1 til 3 bytes, afhængig af instruktionstype.

Dette fortæller, hvor mange bytes instruktionen fylder i program-memory.

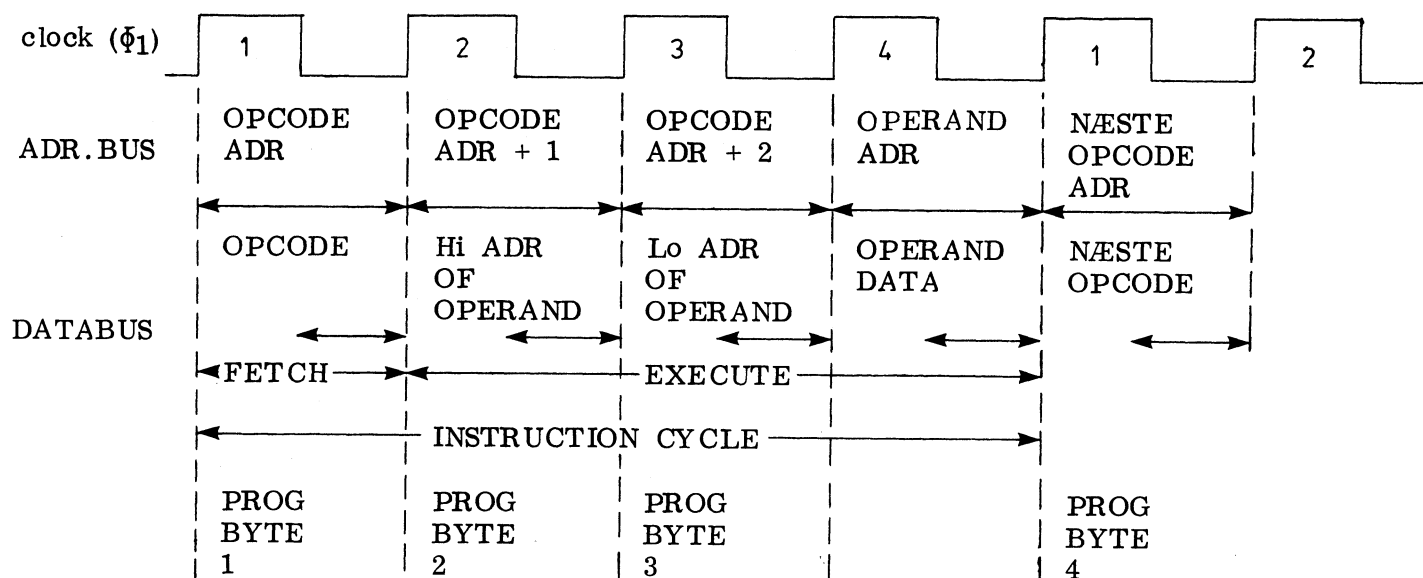
En anden måde at måle instruktions-sekvensen på, er at angive, hvor lang tid det tager at udføre den. Dette angives ved M6800 som et antal

MPU cycles = clock cycles = machine cycles (~)

og kan være fra 2 til 12, afhængig af instruktionstype. (Bemærk, at clock cycle er det samme som machine cycle ved M6800 – dette er ikke altid tilfældet ved andre microprocessor-fabrikater, hvilket ofte gør det besværligt at sammenligne de forskellige processors hastigheder).

Clock-signalet er microprocessorens taktgiver. Ofte er det en krystalstyret oscillator med en periodetid på ca. 1  $\mu$ sek. For M6800 gælder  $f_{\min} = 100 \text{ kHz}$  og  $f_{\max} = 1 \text{ MHz}$ .

Nedenfor er vist sammenhængen mellem de forskellige begreber ved en INSTRUCTION CYCLE, og eksemplet viser instruktionen LOAD ACCUMULATOR A ved hjælp af EXTENDED ADRESSERING (LDA A m. opcode = \$ B6, som har # = 3 og ~ = 4).



Yderlig oplysning om ADR. og DATABUS TIMING kan findes i "SYSTEM DESIGN DATA FOR M6800".

Fig. 5.6

## PROGRAMOPBYGNING

Et program opbygges for overskuelighedens skyld som et hovedprogram og nogle underprogrammer (subrutiner).

En anden grund til at anvende subrutiner er, at der ofte er nogle rutiner (beregninger o.s.v.), som bliver gentaget flere gange i løbet af et programgennemløb. Hvis der f.eks. er brug for en multiplikation flere gange i et program, kan selve multiplikationen laves som et underprogram, der kaldes hver gang, der er brug for en multiplikation.

Hovedprogram

Start

Slut

Subprogram  
(multiplikation)

Fig. 5.7

Hvis en af de rutiner, som foretages i multiplikationsrutiner, f.eks. en addition, laves som et underprogram i forvejen, kan man lade multiplikationsrutinen kalde additionsrutinen. Når en subrutine kalder en anden subrutine, kaldes det NESTING (redebygning).

Hovedprogram

1. Subrutine

2. Subrutine

Start

(multiplikation)

(addition)

osv.

Nesting

Fig. 5.8



## INTERRUPT ROUTINE

De ovenfor nævnte underprogrammer blev kaldt af hovedprogrammet eller et andet underprogram. En interrupt-rutine er et underprogram, som kaldes ved at påvirke microprocessoren med et elektrisk signal. Denne form for underprogrammer anvendes ofte i forbindelse med input/output enhedernes betjening. Hvis interrupt-rutinen netop er en input betjeningsrutine, kan et programs forløb se således ud:

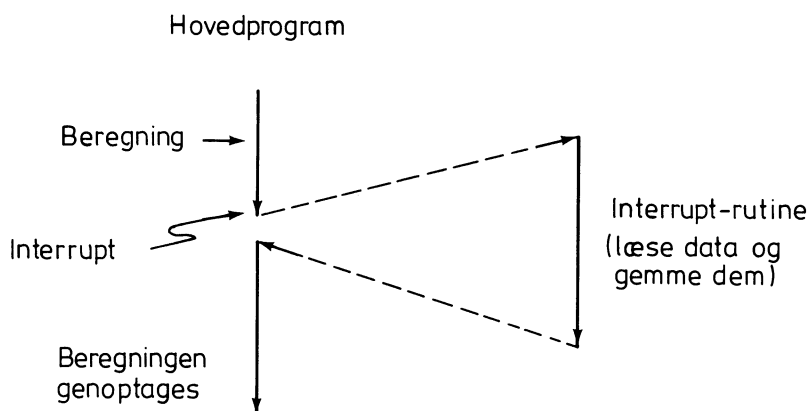


Fig. 5.9

På fig. 5.9 ses det, at interrupt signalet (interrupt request (IRQ)) afbryder en beregning og tvinger processoren til at køre interrupt-programmet. Når dette er færdigt, hopper processoren automatisk tilbage og genoptager den afbrudte beregning.

## I/O KONTROL METODER

Der findes 3 grundprincipper for styring af input-output kredsløbene, når data skal transporteres. De 3 metoder benævnes:

1. POLLING
2. INTERRUPT (IRQ)
3. DIRECT MEMORY ACCESS (DMA)

Af disse benyttes 1 ved mindre systemer, 2 ved mindre og mellemstore systemer og den 3. næsten udelukkende ved store systemer.

System 1 og 2 vil blive omtalt her med vægten lagt på 2, medens der ved system 3 må henvises til den litteratur, der behandler emnet, hvis yderligere oplysninger ønskes.

### POLLING

POLLING kan også benævnes PROGRAMSTYRET INPUT-OUTPUT, idet al trafik til og fra microcomputeren over I/O kredsløbene styres fra CPU'ens program. Processoren udsender og henter data og alle input og output operationer foretages under kontrol af det program, der bliver udført.

Datatransporten skal koordineres med en "HANDSHAKE" procedure. Ved hjælp af FLAG i I/O enhederne kan processoren se, om en enhed kræver opmærksomhed. Et FLAG består af et enkelt register-bit eller en speciel flip flop. Flaget kan SÆTTES, når en input-enhed har et fyldt register, som det gerne vil aflevere til CPU'en eller når et output register har afleveret sit indhold til den ydre verden.

Selve POLLING processen består i, at microcomputerens PROGRAM i „tur og orden“ spørger de enkelte enheder, om de har flaget SAT. Hvis CPU'en møder et SAT FLAG, vil den betjene denne I/O enhed, SLETTE FLAGET (reset) og gå videre til den næste enhed. Er et flag reset, går processoren straks videre til næste enhed.

Undersøgelsen er udelukkende PROGRAMSTYRET d.v.s. SOFTWARE-BESTEMT og afvikles SYNKRONT med programudførelsen. Processen kan foregå som vist i fig. 6.1.



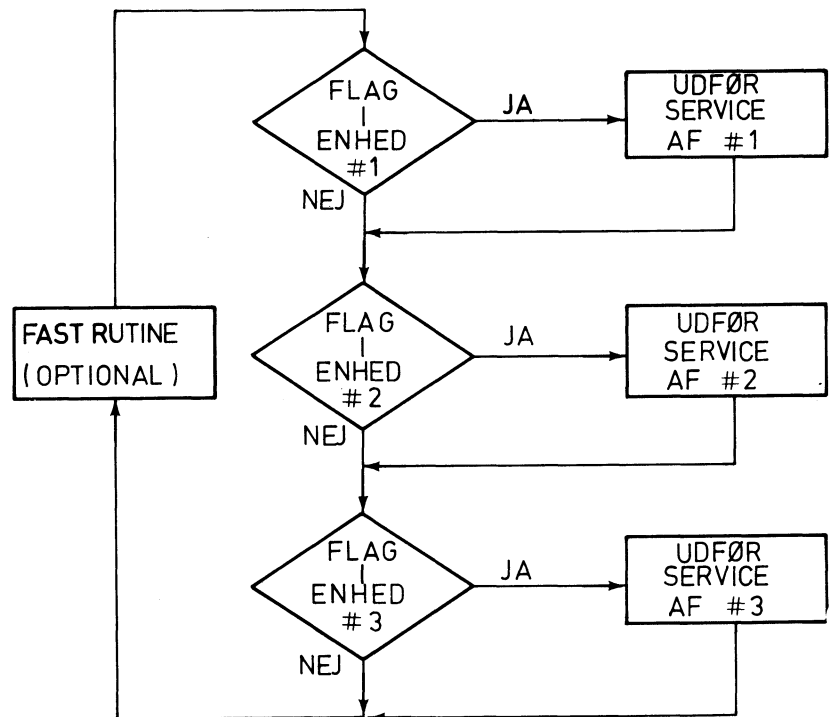


Fig. 6.1

At udføre input-output ved hjælp af POLLING kræver meget program (software), men til gengæld kun lidt hardware. Processoren er i en stor del af dens tid beskæftiget med at gennemløbe POLLING programmet. Der kan kun udføres mindre programsekvenser udenfor polling-rutinen, hvilket bevirker, at denne metode normalt kun anvendes ved mindre systemer. Desuden bevirker det relativt langsomme gennemløb, at der kun kan arbejdes med LANGSOMME YDRE ENHEDER som f.eks. KEYBOARD og SERIELLE SIGNALER.

## INTERRUPT

De fleste microprocessorer er forsynet med terminaler og internt kredsløb, der giver mulighed for I/O styring ved hjælp af INTERRUPT.

Et INTERRUPT er et ASYNKRONT krav om CPU'ens OPMÆRKSOMHED til det kredsløb, der har afgivet INTERRUPT REQUEST (IRQ). INTERRUPT omtales også ofte som „TVUNGEN AFBRYDELSE”.

Microcomputeren kører normalt i sit hovedprogram og først når en I/O enhed afgiver et IRQ-signal til CPU'ens IRQ-input, vil processoren reagere. Reaktionen kan være en af følgende to:

1. Fortsætter i hovedprogram, d.v.s. IRQ afvises (MASK ON).
2. Går til IRQ-rutinen, når igangværende statement er udført (MASK OFF).

Programudførelse med mulighed for INTERRUPT kan ses af fig. 6.2 og fig. 6.3.

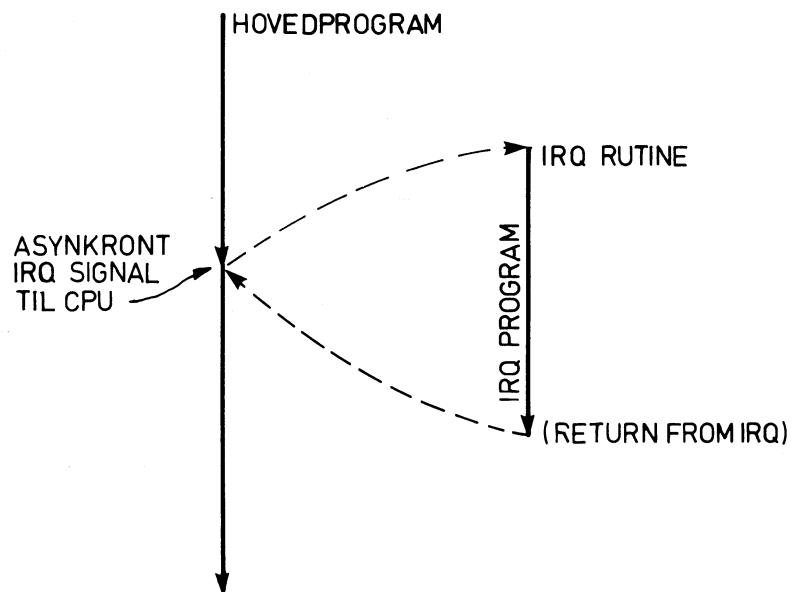
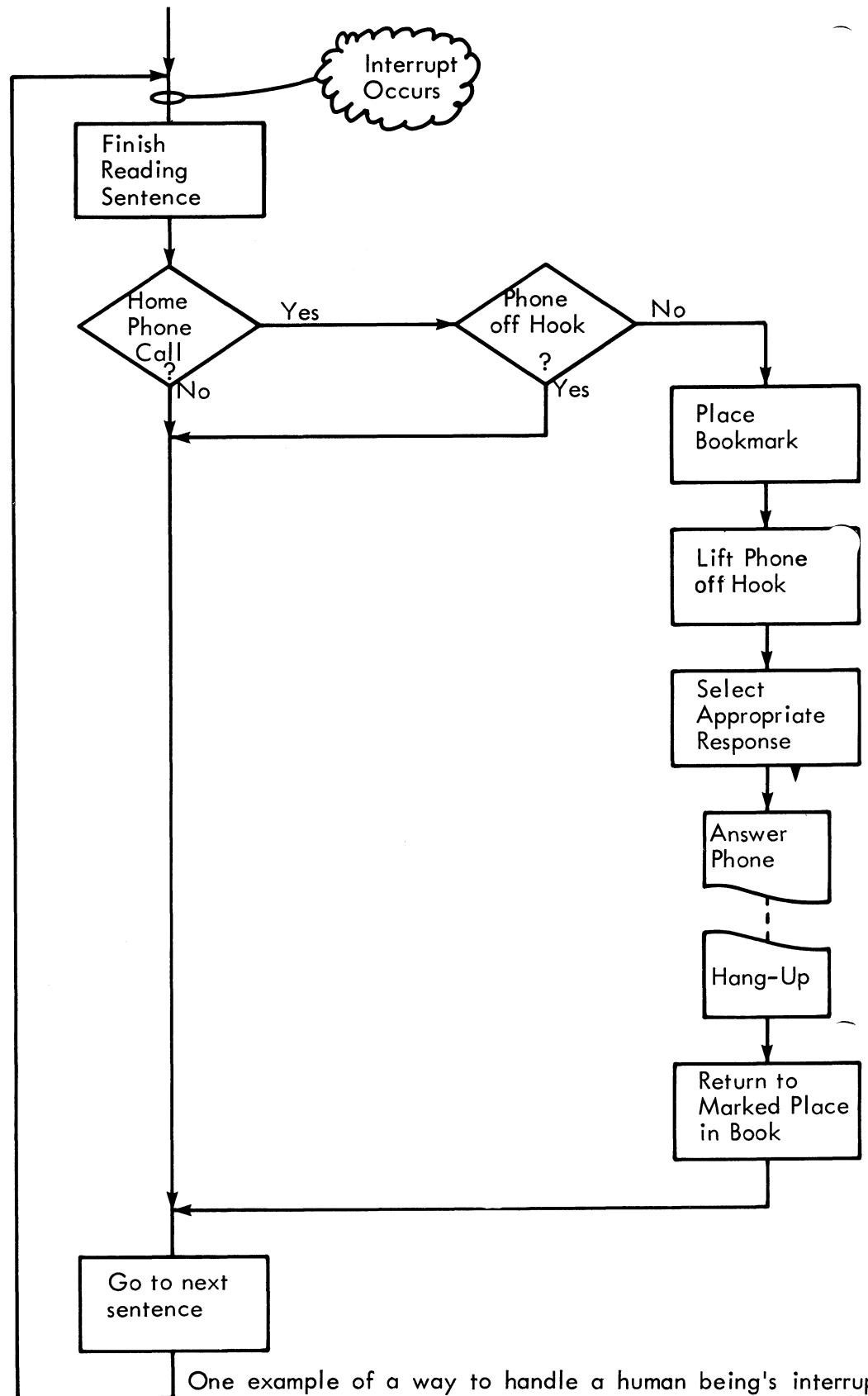
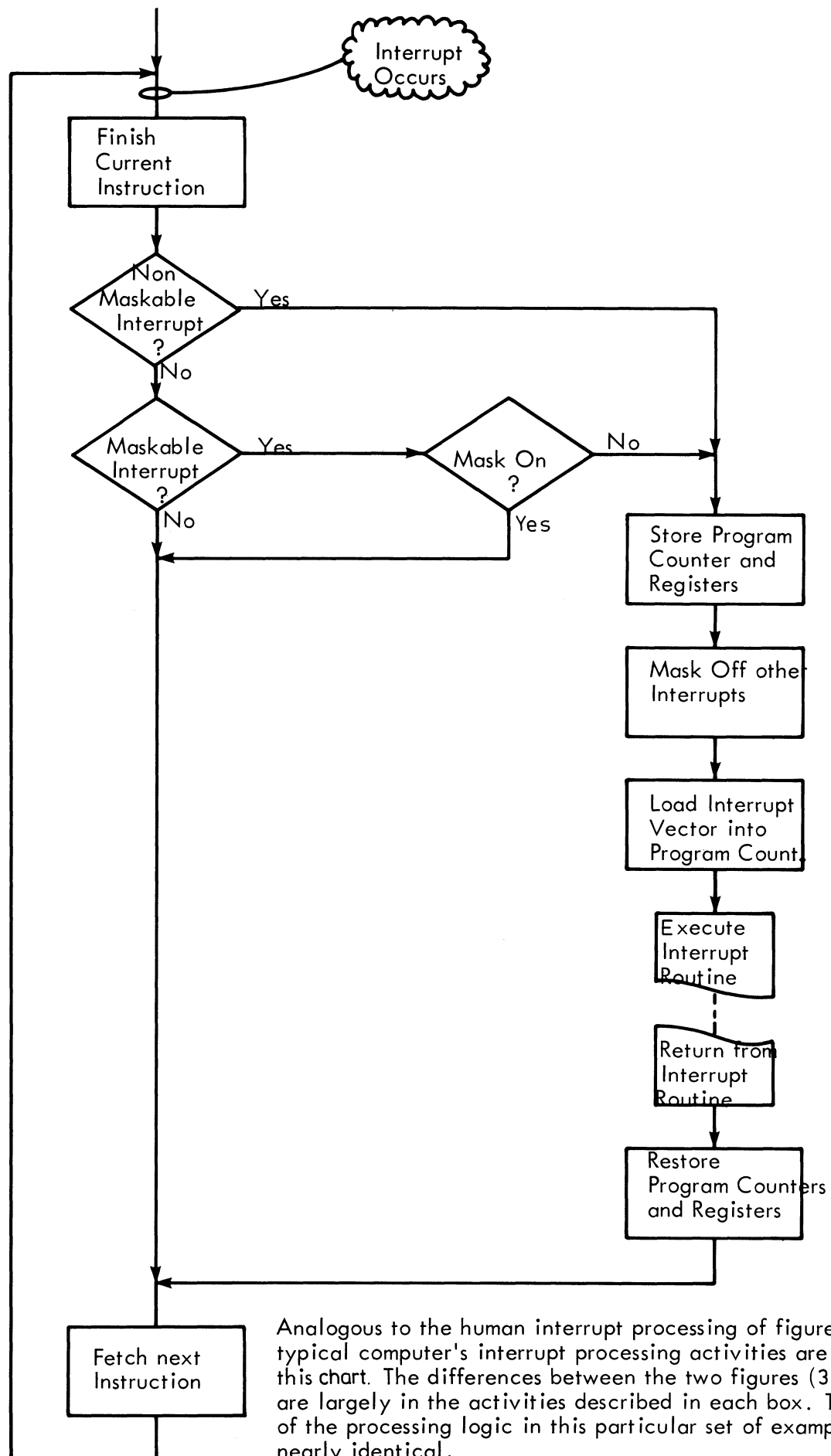


Fig. 6.2



One example of a way to handle a human being's interrupt processing in response to a ringing bell. The ringing bell is like a signal on a multiple source interrupt line of a computer. The first object is to identify the source of the interrupt. If processing is done, the state of the interrupted process (reading a book here) is saved (with a bookmark) while the phone is answered. After the phone call, the reading of the book may be resumed at the place of the bookmark, restoring the original process.

Selve INTERRUPT RUTINEN udføres forskelligt hos de forskellige fabrikater, men hos de fleste har det et forløb som vist i fig. 6.4.

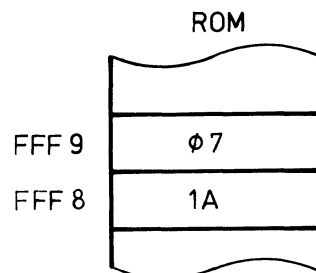


Analogous to the human interrupt processing of figure 3, the typical computer's interrupt processing activities are shown by this chart. The differences between the two figures (3 and 4) are largely in the activities described in each box. The form of the processing logic in this particular set of examples is nearly identical.

Fig. 6.4

Når CPU'en har forhindret yderlig IRQ (MASK ON) og har lagt de „gamle” registre ud i STACK, skal den til at finde den enhed, der afgav IRQ.

Dette opnås ved, at CPU'en AUTOMATISK UDSENDER EN BESTEMT ADRESSE på adressebussen. F.eks. vil MC 6800 udsende først FFF8 og derefter FFF9 på adressebussen. Herved udpeges to bytes i ROM, som indeholder IRQ-VEKTOREN, d.v.s. den adresse, hvor IRQ-PROGRAMMET STARTER. Det kan f.eks. have følgende udseende:



Dette vil bevirke, at PROGRAMTÆLLEREN får indholdet 1A07 ved start af IRQ. På denne adresse (1A07) findes den første instruktion i IRQ-PROGRAMMET.

Findes der kun EN ENHED, der kan afgive IRQ, vil betjeningsprogrammet til denne enhed begynde ved IRQ VEKTORENS værdi.

## INTERRUPT med POLLING

Findes der FLERE ENHEDER, der kan gøre krav på CPU'ens opmærksomhed via IRQ, som det f.eks. er vist i fig. 6.5, så må IRQ-PROGRAMMET f.eks. starte med at spørge de enkelte enheder, hvem der afgav IRQ, d.v.s. gennemløbe et POLLING PROGRAM som skitseret i fig. 6.6.

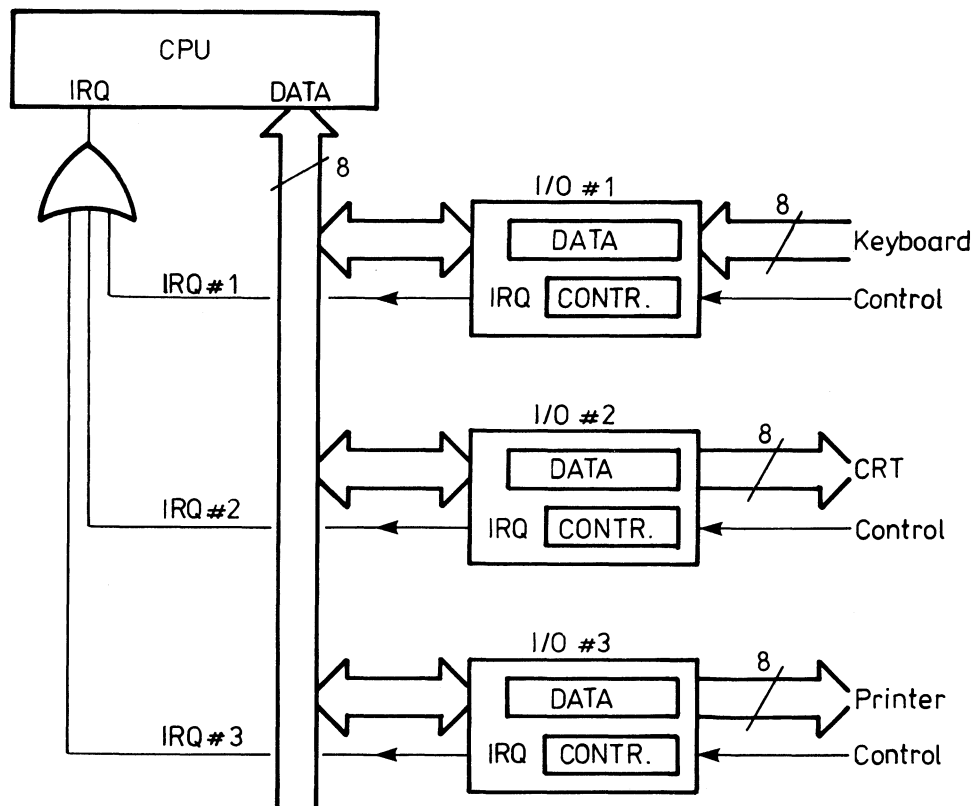


Fig. 6.5

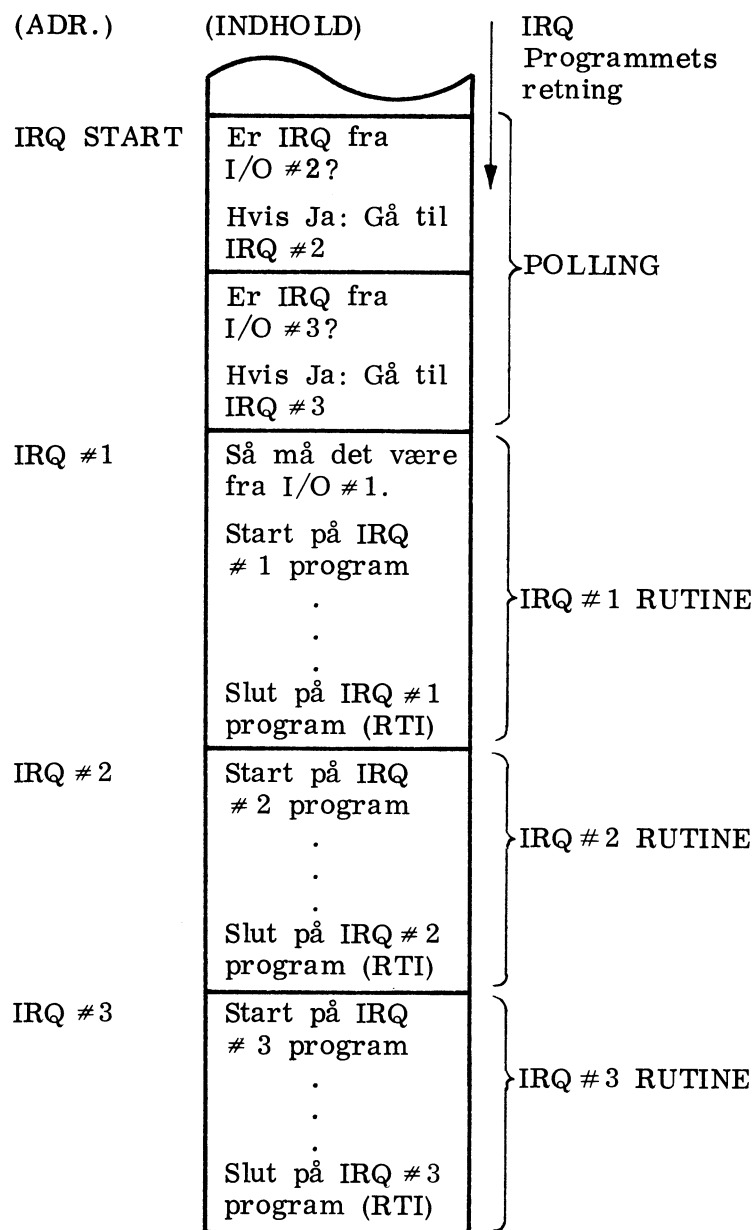


Fig. 6.6

Af dette ses, at det er softwaren i microcomputeren, der finder enheden og betjener den. Af fig. 6.6 ses, at der er en prioriteret rækkefølge, der sædvanligvis hænger sammen med hastigheden på de ydre enheder, således at hurtigste enhed har den højeste prioritet, d.v.s. undersøges FØRST i polling-processen.

### „DAISY CHAIN”

En anden og hurtigere metode til at finde kredsen, der afgav IRQ, er ved at anvende en blanding af software og hardware. Dette kan f.eks. udføres som den såkaldte „DAISY CHAIN”, der er vist i fig. 6.7.

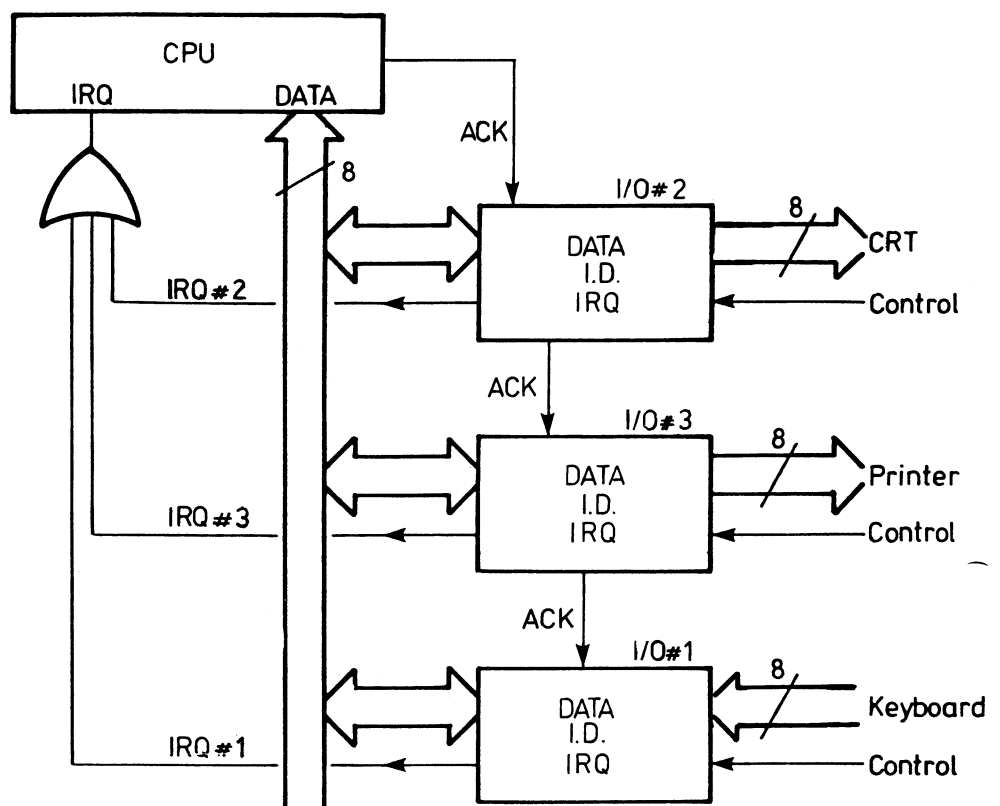


Fig. 6.7

Forskellen i forhold til den tidligere udgave (fig. 6.5) ligger i, at CPU'en udsender et ACKNOWLEDGE SIGNAL (ACK), som tilføres enheden med den højeste prioritet. Hvis det er denne, der har afgivet IRQ, svarer den ved at sende et I.D. ud på databussen. Dette I.D. fortæller CPU'en, hvor IRQ-programmet til enheden skal findes og CPU'en går så til dette program og udfører det. Hvis første enhed IKKE har udsendt IRQ, vil denne enhed i stedet generere et ACK signal, som sendes til næste enhed i prioritetsforløbet og så fremdeles, indtil IRQ giveren findes.

## VEKTORISERET INTERRUPT

Den hurtigste metode betegnes vektoriseret interrupt. Her afgiver den enkelte I/O enhed både IRQ signal og IDENTIFIKATION samtidig til CPU'en og til en PRIORITETS INTERRUPT CONTROLLER.

CONTROLLER KREDSLØBET genererer ud fra identifikationen den aktuelle 16 bit ADRESSE, som peger på IRQ-PROGRAMMET til betjening af den enhed, der afgav IRQ.

Ofte mødes en blanding af de forskellige udpegningsrutiner. F.eks. kan man ved MC 6800 (og MC 6802) tale om VEKTORISERET INTERRUPT, idet der skelnes mellem:

Signal type	CPU ben nr	CPU udsender på ADR. BUS (MS) (LS)		IRQ type.
$\overline{\text{RESTART}}$	40	FFFE	FFFF	H. W.
$\overline{\text{NMI}}$	6	FFFC	FFFD	H. W.
$\overline{\text{IRQ}}$	4	FFF8	FFF9	H. W.
SWI		FFFA	FFFB	S. W.

Fig. 6.8

Af fig. 6.8 ses det, at der findes forskellige INTERRUPT-NIVEAUER (prioriteter) udført i hardware ved forskellige ben på CPU'en. På de enkelte ben kan der være koblet flere enheder, der kan afgive interrupt og problemet med identifikation af disse enheder løses oftest med en POLLING procedure.

Et eksempel på, hvordan interrupt gennemføres i MC 6800 (eller 6802) kan ses af rutediagrammet i fig. 6.9.



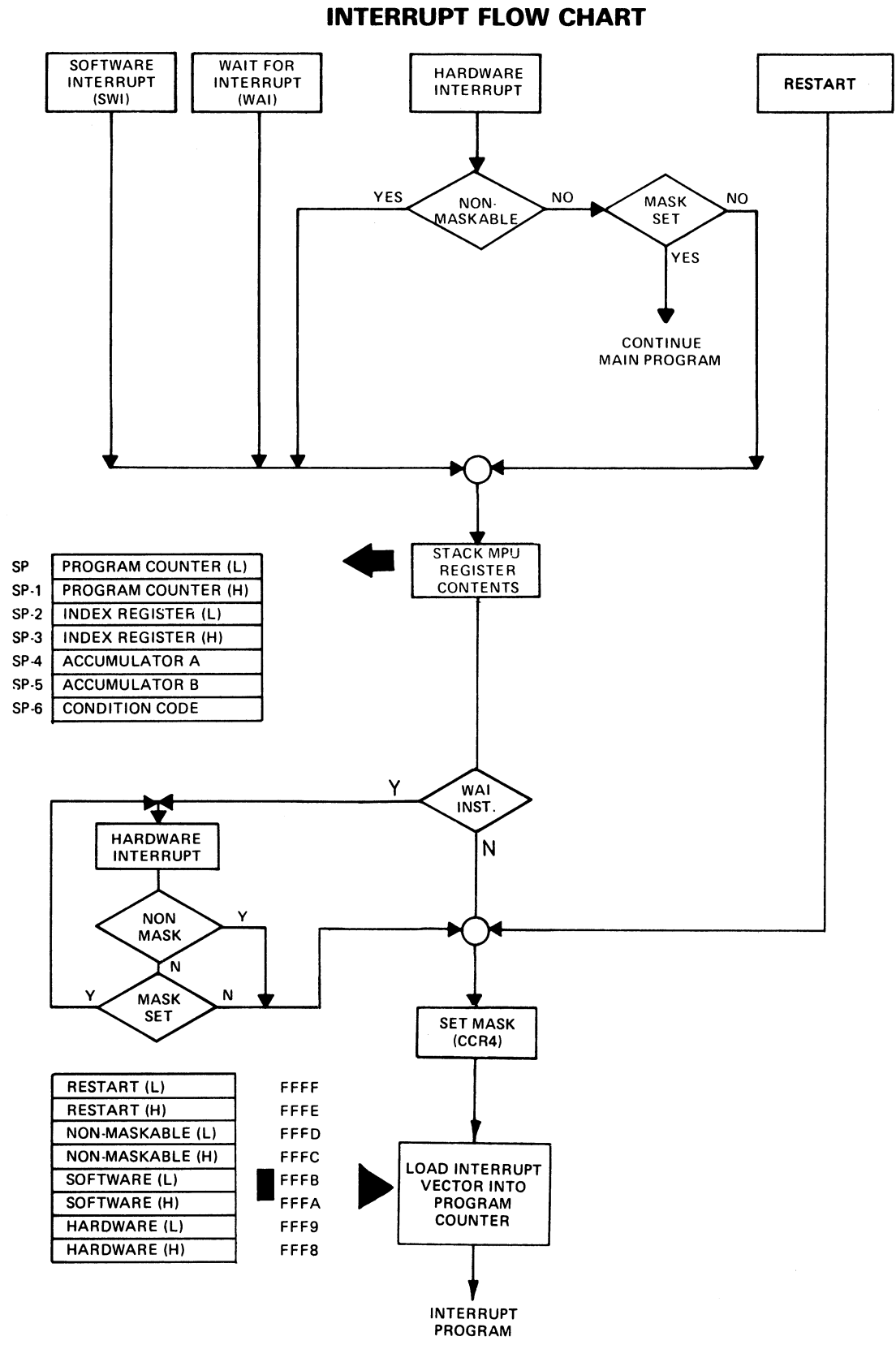


Fig. 6.9

## **INTERRUPT NESTING**

Hvis der er forskellig prioritet på INTERRUPT, eller hvis IRQ MASK resettes i løbet af en interrupt rutine, kan der skabes INTERRUPT INDEN ET INTERRUPT PROGRAM — dette betegnes som INTERRUPT NESTING.

Hvor mange NIVEAUER, der kan foretages NESTING i, afhænger først af størrelsen på STACK'en, men nok i højere grad af PROGRAMMØRENS OVERBLIK. Det er unormalt i tilsigtede programforløb, at nå ud over 5–7 niveauer. Som eksempel på INTERRUPT i flere niveauer kan fig. 6.10 benyttes. Her viser en streg (—), hvilket program, der kører i øjeblikket og cirkler (0000), hvilket INTERRUPT NIVEAU, der har afgivet IRQ. Niveau Ø er pr. definition det, der har HØJEST PRIORITET:

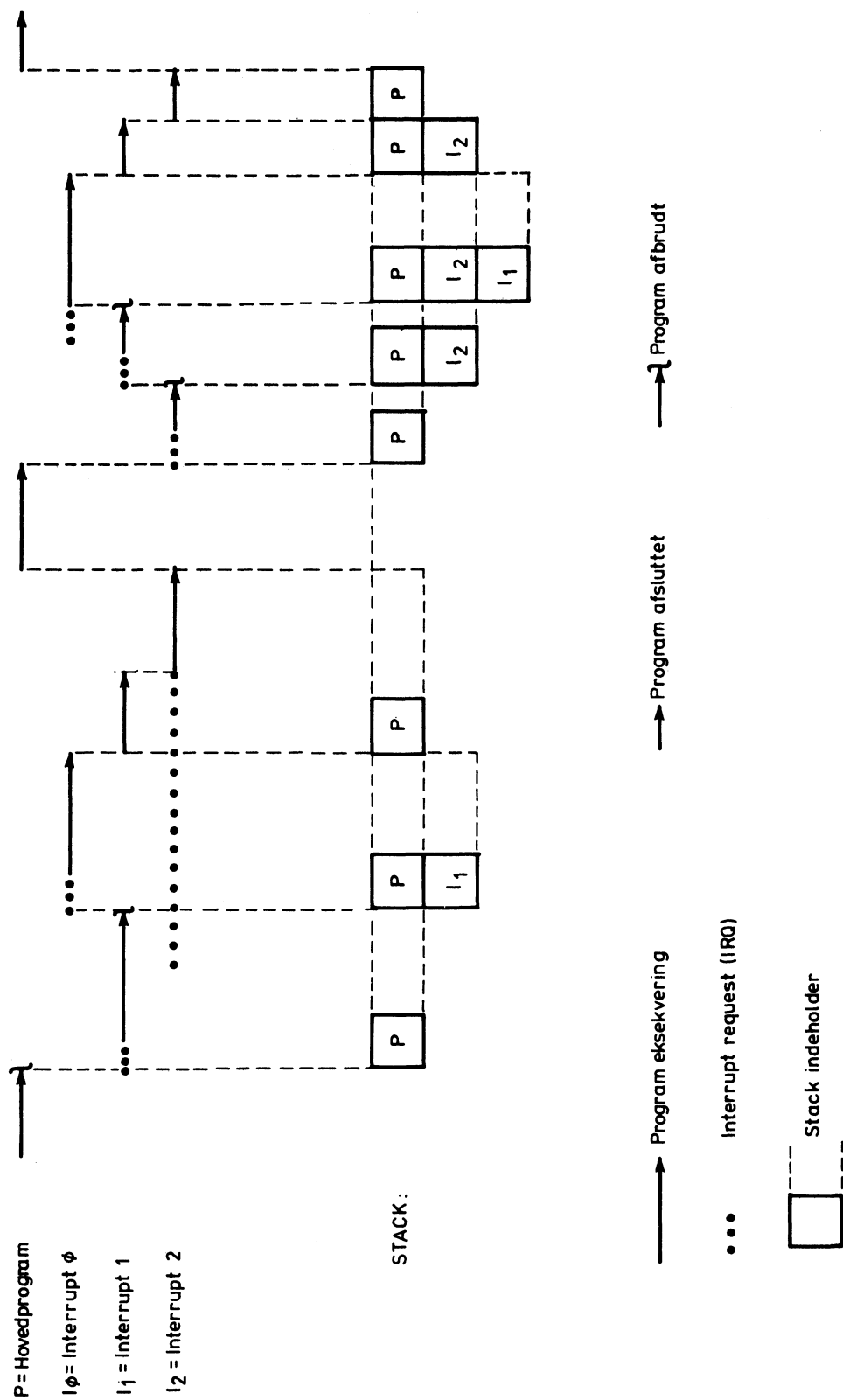


Fig. 6.10

6800 FAMILIENS HARDWARE

Indholdsfortegnelse	Side
Indledning.....	7.3
6800 familiens medlemmer.....	7.3
MC 6801 one chip microcomputer .....	7.3
MC 6802 microcomputer .....	7.4
Kompatibilitet mellem MC 6800 og 6802 .....	7.5
MC 6802 blokdiagram .....	7.6
6802 CPU'ens registre og programmeringsmodel .....	7.7
MC 6802 CPU'en .....	7.8
Clock .....	7.9
On chip RAM .....	7.9
RESET .....	7.10
HALT .....	7.10
READ/WRITE (R/W) .....	7.11
Bus Available (BA).....	7.11
Valid Memory Address (VMA) .....	7.11
Non Maskable Interrupt (NMI) .....	7.11
Interrupt Request (IRQ).....	7.11
MC 6802 Elektriske data.....	7.14
MC 6802 Timing for READ og WRITE .....	7.14
MCM 6810 Random Access Memory (RAM).....	7.16
MCM 6810 A Datablad .....	7.18
MC 6821 Peripheral Interface Adapter (PIA) .....	7.19
Interface PIA - CPU .....	7.19
Interface PIA - Ydre verden.....	7.20
PIA Interface (Pin Configuration).....	7.20
PIA Registre .....	7.23
PIA Adresse .....	7.24
PIA Initialisering .....	7.26
Controlregistre i PIA .....	7.27
MC 6821 Datablad .....	7.33
MC 6850 Asynchronous Communication Interface	
Adapter (ACIA) .....	7.34
ACIA blokdiagram .....	7.35
ACIA adressering .....	7.36
ACIA registre.....	7.37
TDR-register .....	7.37
RDR-register .....	7.37
CONTROL-register.....	7.38
STATUS-register.....	7.42
ACIA signaler til og fra den ydre verden.....	7.47
Programmering af ACIA .....	7.48
S 6850 Datablad .....	7.53
Programable Read Only Memory (EPROM) .....	7.54
Sletning af EPROM.....	7.54
Programmering af EPROM .....	7.54
2716 Datablad .....	7.56
Kompatible kredsløb.....	7.57
Intel 2716 anvendt i PRESS 8041 .....	7.58
Adressedekoder.....	7.59
74LS138 dekoder .....	7.60
Memory map .....	7.62
Analog/digital og digital/analog omformning.....	7.67
D/A omformer .....	7.67
MC 1408 .....	7.69
A/D omformer.....	7.71
Successive Approximation.....	7.71



## INDLEDNING

I dette afsnit behandles en bestemt microcomputer families hardware for at belyse de virkelige fysiske forhold ved microcomputerens del-elementer. Den generelle teori er omtalt i afsnit 3.

Den valgte familie er MOTOROLA's M6800 familie, som bl.a. second sources fra American Microsystems Inc. (AMI), hvor familien benævnes S6800, men er fuldt ud kompatibel med Motorola's. Fairchild er også second source med deres F6800.

## 6800 FAMILIENS MEDLEMMER

### MC 6800 microcomputer

De tidligste udgaver af microcomputere opbygget med Motorola's komponenter indeholdt:

	CPU	= MC 6800 [processor]
128 bytes	RAM	= MC 6810 [hukommelse]
	PIA	= MC 6820 [parallel I/O]
1 k bytes	ROM	= MC 6830 [hukommelse]
	ACIA	= MC 6850 [seriel I/O]
1 MHz	CLOCK	= MC 6871 [oscillator]

Et microcomputer system opbygget af disse komponenter ses i fig. 7.1.

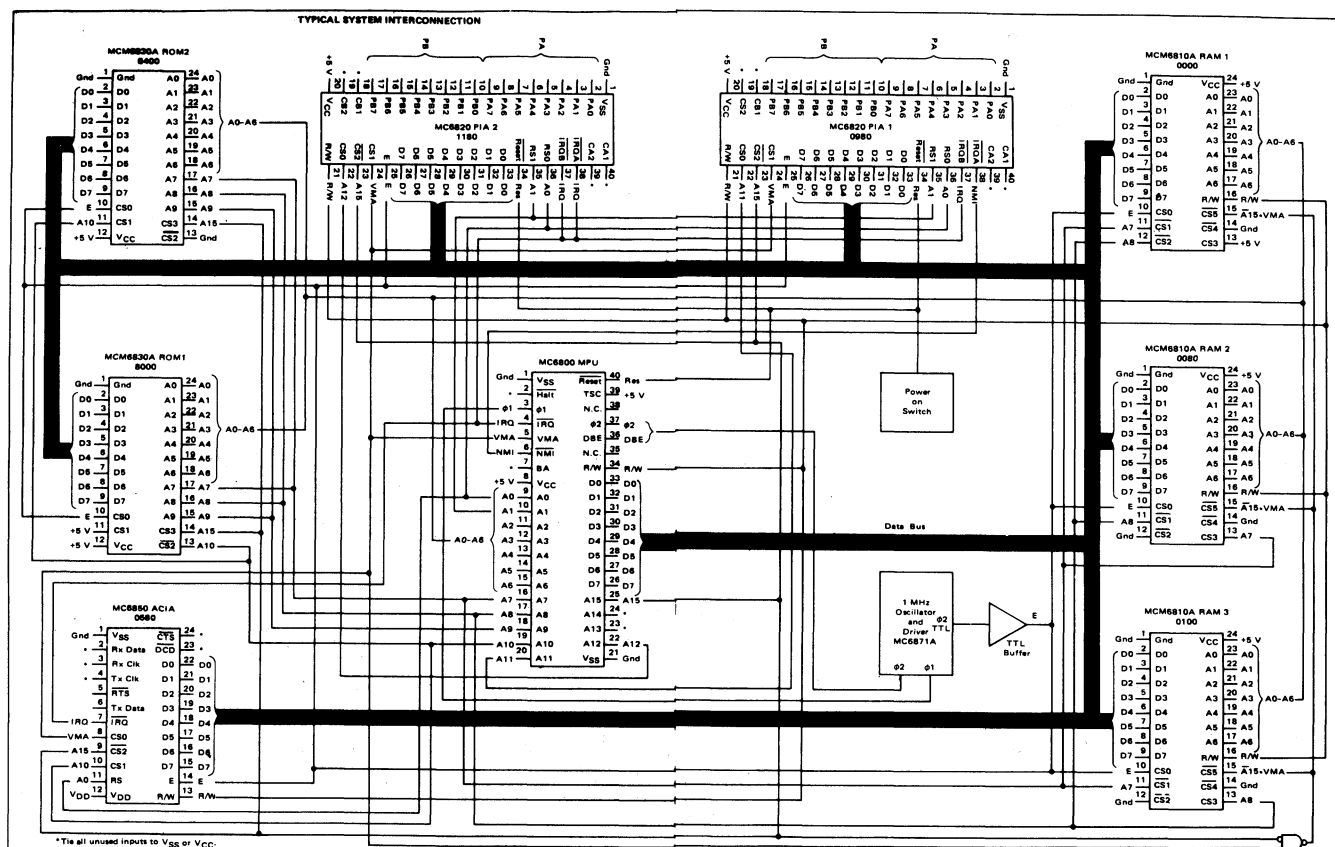


Fig. 7.1. Ⓐ

MC 6801 one chip micro-computer

Udviklingen hen mod mere integrerede systemer har til idag ført frem til

ONE CHIP COMPUTEREN

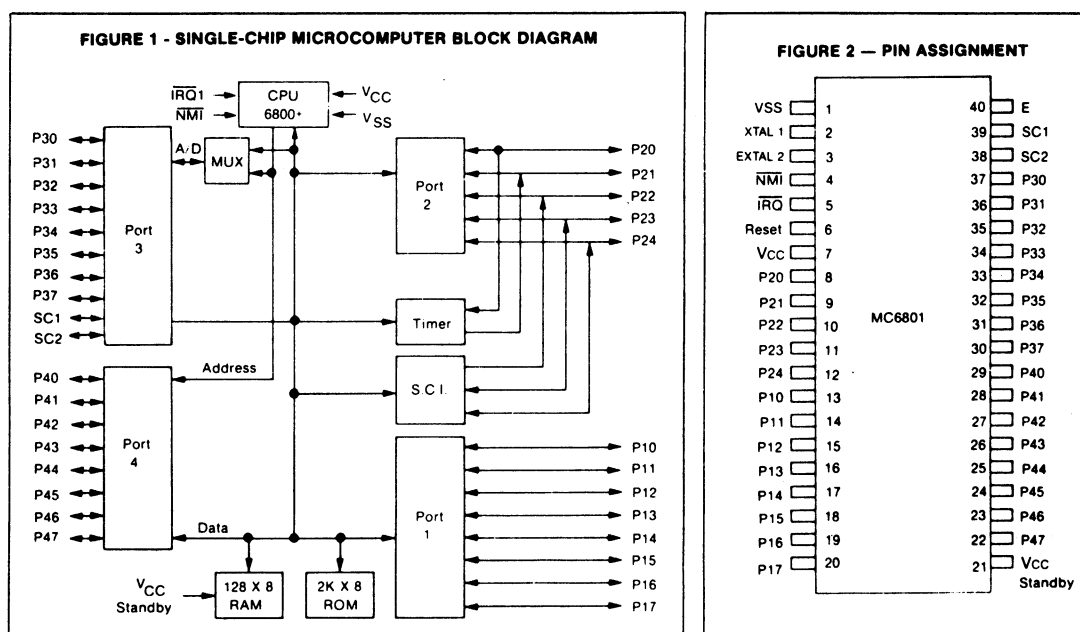



hvor alle funktionerne, der ses i fig. 7.1, er samlet på et integreret areal og kapslet i EN kreds, som f.eks.

## MC 6801

der foruden CPU funktionen består af 128 bytes RAM, 2 k bytes ROM, serial I/O, parallel I/O og intern timer.

I fig. 7.2 vises blokdiagrammet for 6801.



**Fig. 7.2.** 

Er den på chipen værende lagerkapacitet tilstrækkelig til programmet (2 k bytes ROM i 6801 eller 2 k bytes EPROM i 68701 samt 128 bytes SCRATCH PAD i RAM), så kan alle 4 I/O porte benyttes til perifere enheder.

Skal lagerkapaciteten udvides, benyttes port 3 og 4 til ADRESSE-BUS og DATABUS fra ONE CHIP COMPUTEREN til de ydre ROM- og RAM-enheder samt eventuelle andre I/O enheder.

## MC 6802 microcomputer

En anden konfiguration, der giver mulighed for at benytte en mellemting mellem de tidligste systemer og ONE CHIP systemerne, fås ved at benytte CPU'en:

## MC 6802

Denne består af en alm. MC 6800 CPU, on chip CLOCK kredsløb samt 128 bytes on chip RAM. Dette medfører, at tidligere tiders ydre clock kredsløb (MC 6871) ikke skal benyttes her. Samtidig fås RAM lager svarende til 1 stk. 6810 sammen med CPU funktionen. Sammenlignes blokdiagrammet for et 6800 system (fig. 7.3) og et 6802 system (fig. 7.4) fås:

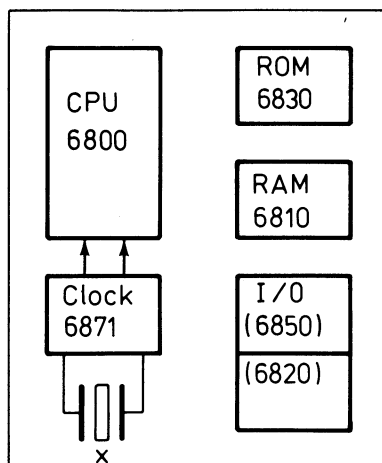


Fig. 7.3.

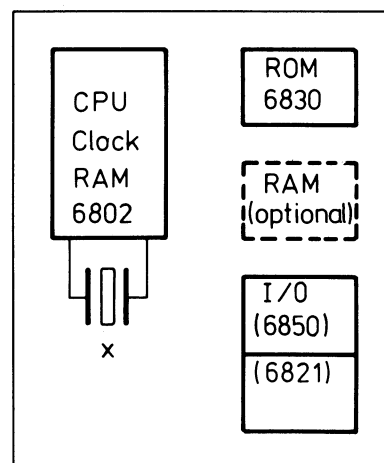


Fig. 7.4.

## KOMPATIBILITET

6802 CPU'en er total SOFTWARE KOMPATIBEL med 6800 CPU'en. De to kredsløb er tillige næsten HARDWARE KOMPATIBLE, hvilket ses af fig. 7.5 (6800 BEN NR/NAVNE) og fig. 7.6 (6802 BEN NR/NAVNE). Dette bevirker, at 6800 i ældre systemer let lader sig erstatte af 6802 og ved ny udvikling bør 6802 benyttes frem for 6800.

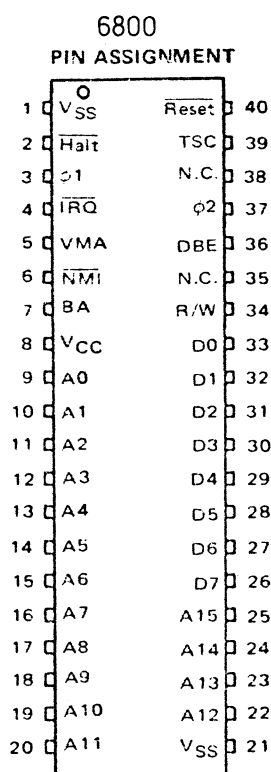


Fig. 7.5.

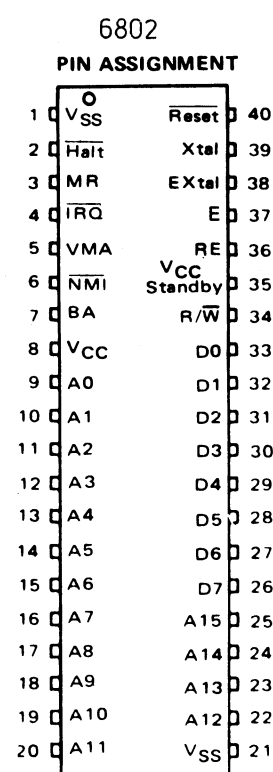


Fig. 7.6.

Der træffes samme ben numre for:

V<sub>CC</sub> = +5 V på ben 8

V<sub>SS</sub> = 0 V på ben 1 og 21

Adressebussen (A<sub>0</sub> → A<sub>15</sub>) på benene fra 9 til 25.

Databussen (D<sub>7</sub> → D<sub>0</sub>) på benene fra 26 til 33.

Controlsignalerne HALT, IRQ, VMA, NMI, BA, R/W og RESET

Afvigende funktioner findes på følgende ben numre:

BEN	6800	6802
3	$\phi_1$	MR
35	N.C.	V <sub>CC</sub> standby
36	DBE	RE
37	$\phi_2$	E
38	N.C.	EXtal
39	TSC	Xtal

Fig. 7.7.

## MC 6802 BLOKDIAGRAM

Blokdiagrammet for processoren MC 6802 ses i fig. 7.8.

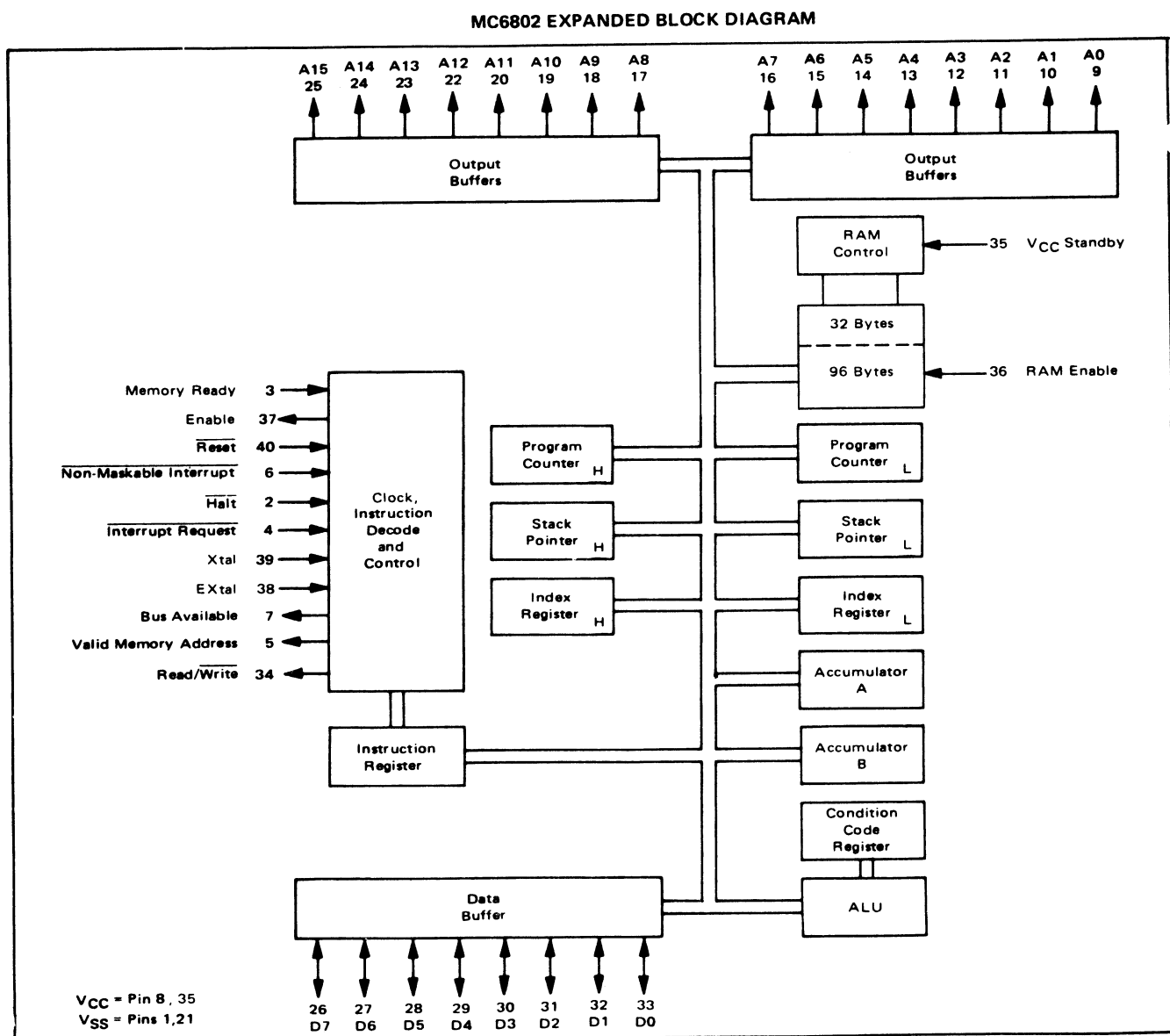


Fig. 7.8. (AA)

En nærmere inspektion af diagrammet vil kun vise få afvigelser fra et standard 8-bit CPU diagram som det i kap. 3 omtalte.

Disse afvigelser består af ON CHIP CLOCK og ON CHIP RAM (32 + 96 bytes), samt RAM CONTROL og kontrolterminalerne RAM Enable (RE) og  $V_{CC}$  standby.

## 6802 CPU'ens REGISTRE OG PROGRAMMERINGSMODEL

CPU'en i 6802 har 3 stk. 16 bit registre og 3 stk. 8 bit registre, der er tilgængelige for brugeren (programmøren), se fig. 7.9.

PROGRAMMING MODEL OF THE MICROPROCESSING UNIT

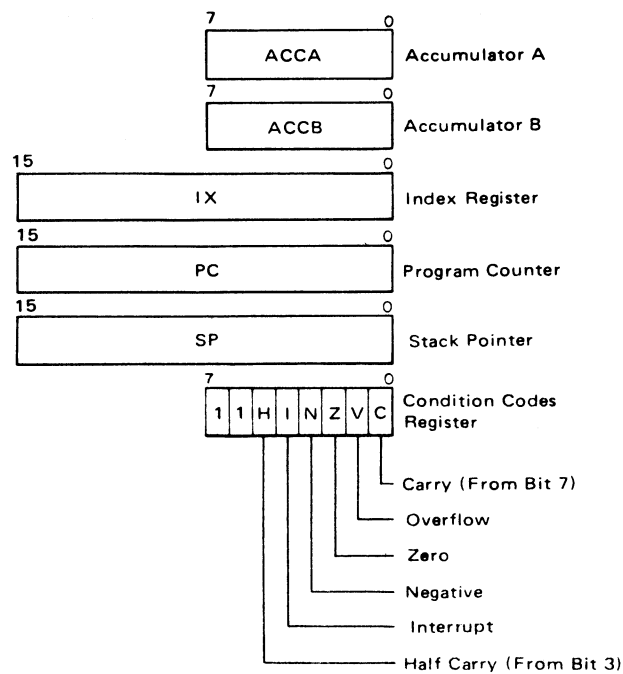


Fig. 7.9. (M)

### PROGRAM COUNTER (PC)

Programtælleren (PC) er et 16 bit register, der indeholder den aktuelle PROGRAMADRESSE og indholdet benyttes til at udpege den byte, der nu skal hentes fra PROGRAM-LAGERET.

### STACK POINTER (SP)

Stack Pegepinden (SP) er et 16 bit register, der indeholder adressen på den næste ledige byte i STACK'en.

Stacken er et eksternt PUSH DOWN/POP-UP RAM hukommelsesområde, der er reserveret til formålet. Det virker som et SIDST IND/FØRST UD LAGER, hvor (SP) automatisk FORMINDSKES/FØRØGES ved STACKINSTRUKTIONENS udførelse.

SP kan påvirkes af forskellige instruktioner som f.eks.:

\* **LOAD STACKPOINTER** med hex nr. FFF0  
**LDS \$FFF0** i assemblerprogram.

**INDEX REGISTER (IX)**

Index registret (IX) består af 16 bit, som er referenceadresse (grundadresse) i den adresseringsform, der i computer-programmet benævnes INDEX ADDRESSING.

**ACCUMULATORER**

(ACC A)

(ACC B)

CPU'en indeholder 2 stk. 8 bit accumulatorer (ACC), som benyttes til at opbevare de OPERANDER (TALVÆRDIER), der skal anvendes som input til ALU'en og ligeledes opbevare de resultater, ALU'en afgiver.

**CONDITION CODE REGISTER (CCR)**

Condition Code Registret (CCR) er et 8 bits register, hvor kun 6 af bittene benyttes, idet de to MSB = 1. Registret er en STATUS af de aritmetiske/logiske resultater, der kommer fra ALU'en. Der er bit for følgende resultater:

C = Carry fra bit nr. 7

V = Overflow i to-komplement

Z = Zero

N = Negativ

H = Half Carry fra bit nr. 3

I = Interrupt masken

**MC 6802 CPU'en**

Benforbindelserne til 6802 ses af fig. 7.6, og en typisk anvendelse ses i fig. 7.10.

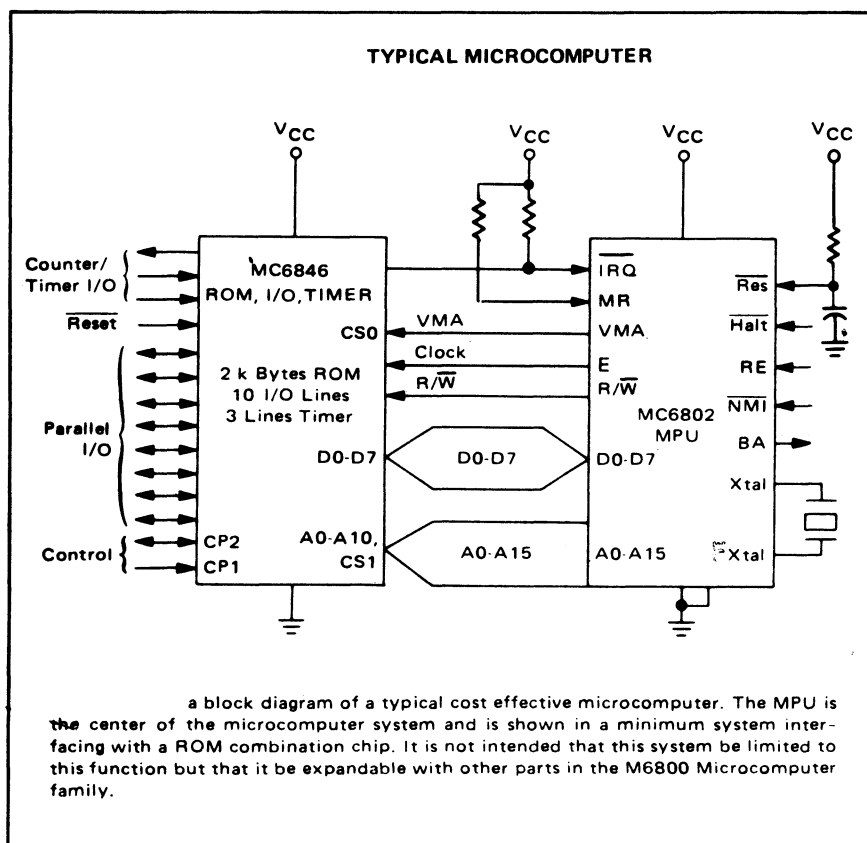


Fig. 7.10. Ⓐ

## **V<sub>CC</sub> ADRESSEBUSSEN OG DATABUSSEN**

Forsyningsspændingen (+5 V), adressebussen (16 ben) og databussen (8 ben) anvendes som angivet i teorien behandlet i kapitel 3. De to bussystemer kan som output drive en standard TTL Load (fan out = 1). Databussen kan placeres i three-state (f.eks. ved  $\overline{\text{H\AA LT}}$ ).

## **CLOCK (E, EXtal, Xtal)**

6802 har intern oscillator (on chip osc.), som kan krystalstyres. Krystallet skal være af SERIERESONANSTYPEN og med  $f_{\text{Xtal}}$  4 gange så høj som clockfrekvensen der ønskes, idet 6802 automatisk dividerer  $f_{\text{Xtal}}$  med 4. ( $\text{Xtal} = 4 \text{ MHz} \Rightarrow \text{clock} = 1 \text{ MHz}$ ). Krystallet forbindes mellem Xtal (ben 39) og EXtal (ben 38).

## **ENABLE (E)**

Enable (E) er TTL kompatibelt output fra 6802 og forsyner det øvrige system uden for 6802 med clocksignal. (E) i 6802 svarer til  $\phi_2$  ved 6800.

(E) kan styres af (MR) signalet.

## **MEMORY READY (MR)**

Memory ready (MR) er et 6802 input kontrolsignal, der tillader „strækning” af Enable-signalet (E). For (MR = HI) er (E) signalet det alm. clocksignal. Når (MR = LO) bliver (E) signalet holdt på sit niveau et helt antal gange halvperioden for den interne clock. Dette benyttes, når der skal udføres INTERFACE til langsomme ydre hukommelser.

## **EXTERN CLOCK**

6802 kan clockes eksternt ved at tilføre clockpulserne til EXtal (TTL kompatibelt input)

## **ON CHIP RAM**

6802 har 128 bytes ON CHIP RAM, hvilket betyder, at der på samme chip som CPU'en befinder sig 128 x 8 bits Random Access Memory. Dette RAM lager er placeret på HEX ADDR. 0000 → 007F.

## **POWER BACK UP ved POWER DOWN**

Området på ADR. 0000 → 001F (32 bytes) kan bevares ved anvendelse af en hjælpeforsyningsspænding (power back up) på 5 V der tilsluttes V<sub>CC</sub> standby terminalen (ben 35). Som back up benyttes ofte en NiCa-celle eller lignende, der placeres på printkortet og holdes opladet ved hjælp af den normale V<sub>CC</sub>. Når forsyningen V<sub>CC</sub> forsvinder (POWER DOWN) og RAM Enable (RE) er LO vil de første 32 bytes on chip RAM forblive intakte (non volatile).

## **RAM ENABLE (RE)**

6802's interne RAM-lager kan kontrolleres af den TTL compatible inputterminal RAM Enable (RE).

RE = HI giver 6802 CPU'en kontrol over on chip RAM, d.v.s. RAM er ENABLE. RE = LO disables on chip RAM.

RE terminalen benyttes også til at disable READING og WRITING på on chip RAM i POWER DOWN situationer. RE skal være LO mindst 3  $\mu\text{s}$ , før V<sub>CC</sub> ≤ 4.75 V indtræffer (se fig. 7.11).

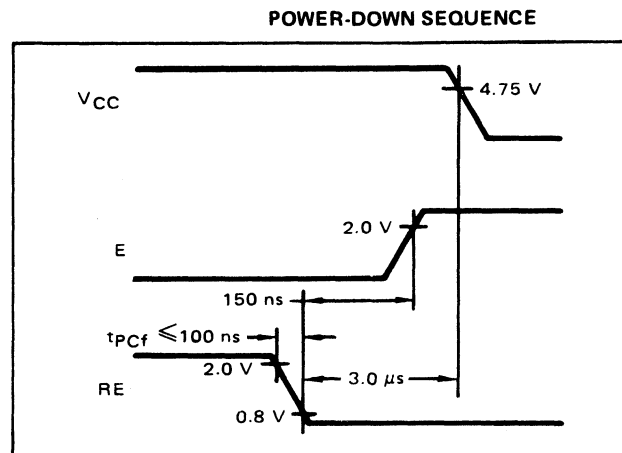


Fig. 7.11. Ⓜ

**RESET**

Dette input til 6802 benyttes til at „RENSE” og starte CPU'en igen enten efter en POWER DOWN forårsaget af forsyningsspændingsfejl ( $V_{CC} \leq 4.75 \text{ V}$ ) eller ved opstart af processoren.

Når **RESET** er LO, befinder CPU'en sig i en INAKTIV tilstand og indholdet i registrene mistes.

Når **RESET** går på HI, starter CPU'en automatisk på reset-sekvensen. Denne indledes med, at ADR. FFFE udsendes på ADR. BUS. Memory-byten på ADR FFFE føres til CPU'en via DATA BUS.

Herefter udsendes ADR. FFFF og den memory byte, der findes på denne adr. føres ligeledes til CPU'en via databus.

De to bytes, der er ført til CPU'en sammensættes til en 16 bit adresse, og denne er STARTADRESSEN PÅ RESET PROGRAMMET (se fig. 7.12).

**MEMORY MAP**

ADR. (HEX)			
FFFF	RESTART	(LS)	} INTERRUPT VEKTORER (ROM MEMORY)
FFFE	—	(MS)	
FFFD	NMI	(LS)	
FFFC	—	(MS)	
FFFB	SWI	(LS)	
FFFA	—	(MS)	
FFF9	IRQ	(LS)	
FFF8	—	(MS)	
FFF7			

Fig. 7.12

RESTART (MS), RESTART (LS) = 16 bit ADRESSE, hvor Restart programmet begynder.

**HALT**

Når **HALT** input til processoren er i Low tilstanden, er maskinens aktiviteter suspenderet.

I **HALT** mode stoppes aktiviteterne med afslutningen af den instruktion, der er under udførelse, når **HALT** går fra HI til LO.



De processorterminaler, der påvirkes af  $\overline{\text{HALT}} = \text{LO}$ , er:

$\text{R}/\overline{\text{W}}$  (Read/ $\overline{\text{Write}}$ ) går HI  
 BA (Bus Available) går HI  
 VMA (Valid Memory Addr.) går LO  
 Three-State outputs (DATABUS) går i Three-State  
 ADDRESS BUS viser CPU PC

Adressebussen vil vise programtællerens indhold (PC), idet denne står på næste instruktionsadresse i programhukommelsen.

## READ/ $\overline{\text{WRITE}}$ (R/ $\overline{\text{W}}$ )

Dette TTL compatible CPU output fortæller memory- og I/O-enhederne, om processoren er i READ (high) eller WRITE (low) tilstanden.

Normal standby tilstand er high (READ), hvilket også gælder ved  $\overline{\text{HALT}}$ .

## BUS AVAILABLE (BA)

Dette signal er normalt i Low. Når det aktiveres og går på HIGH, indikerer det, at processoren er stoppet og at adressebussen er tilgængelig for den enhed, der har udløst (BA).

Processoren kan stoppes ved hjælp af enten  $\overline{\text{HALT}}$  (Hardware) eller WAIT instruktionen (Software).

## VALID MEMORY ADDRESS (VMA)

Dette processor output signal indikerer over for ydre enheder, at der befinder sig en stabil adresse på adressebussen. Det benyttes normalt i forbindelse med chip select til at ENABLE de ydre enheder, specielt PIA og ACIA (se fig. 7.1). VMA kan belastes med en standard TTL indgang.

## NON MASKABLE INTERRUPT ( $\overline{\text{NMI}}$ )

$\overline{\text{NMI}}$  aktiveres ved at tilføre et Low niveau til NON MASKABLE INTERRUPT terminalen på processoren. Denne afslutter herefter den igangværende instruktionssekvens og gemmer indhold af PC, X, ACC A, ACC B og CCR i 7 bytes ude i STACK.

På ADDR. BUSSEN ses derefter først ADDR. FFFC og så ADDR. FFFD. Disse henter 2 bytes i VEKTOR-MEMORY (se fig. 7.12 side 10) og disse bytes danner  $\overline{\text{NMI}}$  programmets startadresse. CPU'en hopper derefter til denne adresse og påbegynder eksekveringen af  $\overline{\text{NMI}}$  programmet.

$\overline{\text{NMI}}$ -terminalen har internt på chip'en en PULL-UP modstand, men trods dette bør der findes en 3 k $\Omega$  modstand mellem  $\overline{\text{NMI}}$  terminalen (ben 6) og  $V_{CC}$  for at optimere kontrollen af  $\overline{\text{NMI}}$  interrupts.

## INTERRUPT REQUEST ( $\overline{\text{IRQ}}$ )

$\overline{\text{IRQ}}$  aktiveres ved at tilføre Low-signal til processorens Interrupt Request indgang (ben 4). Processoren venter til den igangværende instruktion er afsluttet, før  $\overline{\text{IRQ}}$  observeres. Hvis tilstandsregistrets (CCR) INTERRUPT-BIT (I) står på LO, accepteres  $\overline{\text{IRQ}}$  og Interrupt Request rutinen påbegyndes. PC, X, ACC A, ACC B og CCR flyttes ud i STACK. Herefter sættes INTERRUPT-BIT (I) i CCR på HI for at forhindre yderlig  $\overline{\text{IRQ}}$  i at blive accepteret.

På ADR. BUSSEN ses herefter først ADR. FFF8 og senere ADR. FFF9, som henter 2 bytes i VEKTOR-MEMORY. De to bytes danner begyndelsesadressen på IRQ-programmet, og processoren går nu videre med programmet fra denne adresse.

Selvom  $\overline{\text{IRQ}}$  terminalen er forsynet med intern PULL-UP modstand, bør denne terminal (ben 4) forsynes med en extern PULL-UP modstand på 3 k $\Omega$  til V<sub>CC</sub>.

### $\overline{\text{RESET}}$ og $\overline{\text{IRQ}}$

Det skal bemærkes, at  $\overline{\text{Reset}}$  påvirker INTERRUPT-BIT (I) til at gå HIGH, således at dette bit først skal resettes, før  $\overline{\text{IRQ}}$  kan accepteres af processoren.

### $\overline{\text{HALT}}$ og $\overline{\text{IRQ}}$

$\overline{\text{HALT}}$  skal være på High, for at  $\overline{\text{IRQ}}$  kan udføres af processoren.

I fig. 7.13a nedenfor vises CPU'ens arbejdsgang i et FLOW CHART.

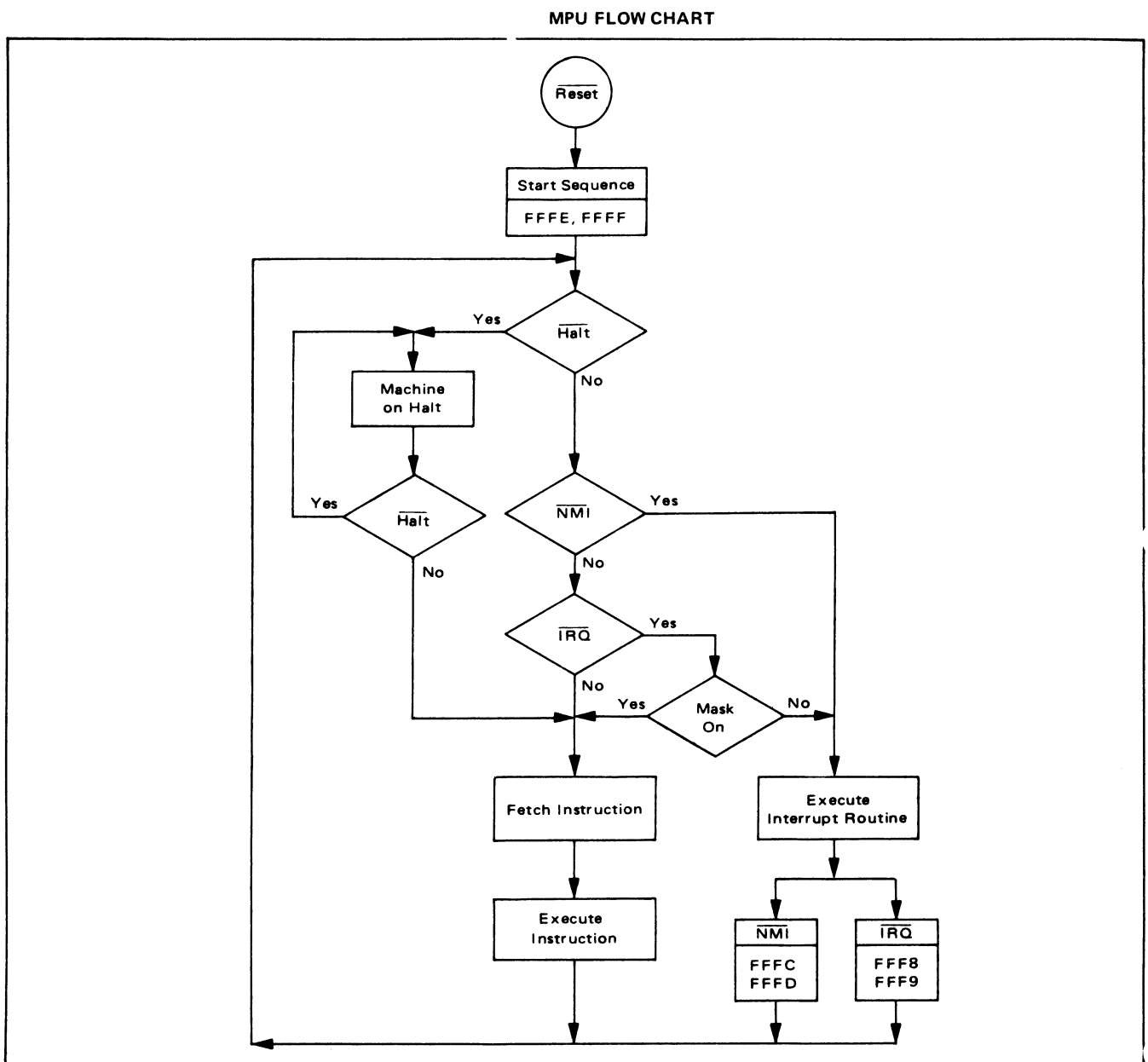


Fig. 7.13.a (M)

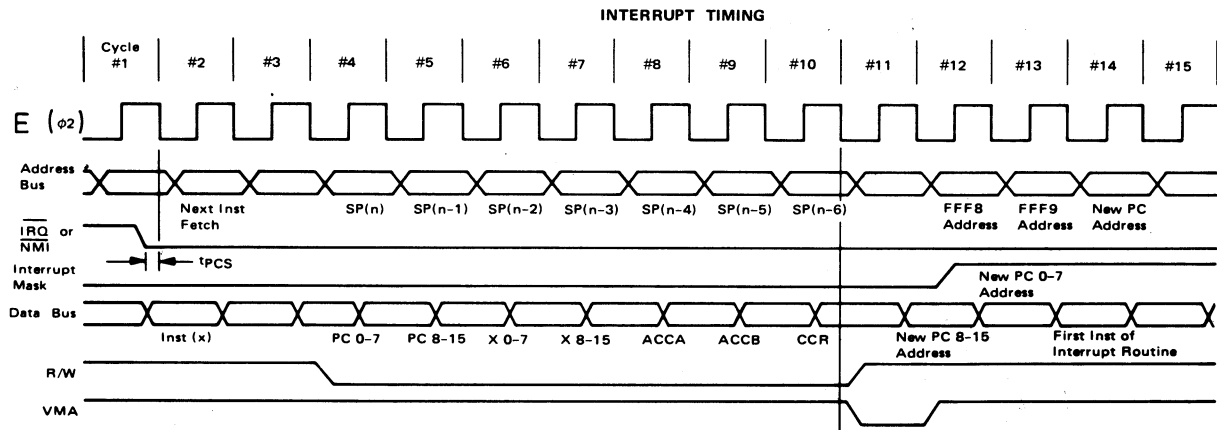
Fig. 7.13b viser pulspanen for  $\overline{\text{IRQ}}$  og  $\overline{\text{NMI}}$  opstart.

Fig. 7.13.b

I Fig. 7.14 ses de maksimale påvirkninger, 6802 kan tåle med hensyn til spænding og temperatur.

**MAXIMUM RATINGS**

Rating	Symbol	Value	Unit
Supply Voltage	$V_{CC}$	-0.3 to +7.0	Vdc
Input Voltage	$V_{in}$	-0.3 to +7.0	Vdc
Operating Temperature Range	$T_A$	0 to +70	°C
Storage Temperature Range	$T_{stg}$	-55 to +150	°C
Thermal Resistance	$\theta_{JA}$	70	°C/W

This device contains circuitry to protect the inputs against damage due to high static voltages or electric fields; however, it is advised that normal precautions be taken to avoid application of any voltage higher than maximum rated voltages to this high impedance circuit.

Fig. 7.14. Ⓐ

De elektriske karakteristikker ved normal drift incl. tolerancer i værste tilfælde kan aflæses i fig. 7.15.


**ELECTRICAL CHARACTERISTICS** ( $V_{CC} = 5.0 \text{ V} \pm 5\%$ ,  $V_{SS} = 0$ ,  $T_A = 0$  to  $70^\circ\text{C}$  unless otherwise noted.)

Characteristic		Symbol	Min	Typ	Max	Unit
Input High Voltage	Logic, EXtal, $\overline{\text{Reset}}$	$V_{IH}$	$V_{SS} + 2.0$ $V_{SS} + 4.0$	— —	$V_{CC}$ $V_{CC}$	Vdc
Input Low Voltage	Logic, EXtal, $\overline{\text{Reset}}$	$V_{IL}$	$V_{SS} - 0.3$	—	$V_{SS} + 0.8$	Vdc
Input Leakage Current ( $V_{in} = 0$ to $5.25 \text{ V}$ , $V_{CC} = \text{max}$ )	Logic*	$I_{in}$	—	1.0	2.5	$\mu\text{Adc}$
Output High Voltage ( $I_{Load} = -205 \mu\text{Adc}$ , $V_{CC} = \text{min}$ ) ( $I_{Load} = -145 \mu\text{Adc}$ , $V_{CC} = \text{min}$ ) ( $I_{Load} = -100 \mu\text{Adc}$ , $V_{CC} = \text{min}$ )	D0-D7 A0-A15, $R/\overline{W}$ , VMA, E BA	$V_{OH}$	$V_{SS} + 2.4$ $V_{SS} + 2.4$ $V_{SS} + 2.4$	— — —	— — —	Vdc
Output Low Voltage ( $I_{Load} = 1.6 \text{ mAdc}$ , $V_{CC} = \text{min}$ )		$V_{OL}$	—	—	$V_{SS} + 0.4$	Vdc
Power Dissipation		$P_D^{**}$	—	0.600	1.2	W
Capacitance # ( $V_{in} = 0$ , $T_A = 25^\circ\text{C}$ , $f = 1.0 \text{ MHz}$ )	D0-D7 Logic Inputs, EXtal A0-A15, $R/\overline{W}$ , VMA	$C_{in}$	— —	10 6.5	12.5 10	pF
		$C_{out}$	—	—	12	pF
Frequency of Operation (Input Clock $\div 4$ ) (Crystal Frequency)		$f$ $f_{Xtal}$	0.1 1.0	— —	1.0 4.0	MHz
Clock Timing						
Cycle Time		$t_{cyc}$	1.0	—	10	$\mu\text{s}$
Clock Pulse Width (Measured at 2.4 V)		$PW_{\phi Hs}$ $PW_{\phi L}$	450	—	4500	ns
Fall Time (Measured between $V_{SS} + 0.4 \text{ V}$ and $V_{SS} + 2.4 \text{ V}$ )		$t_{\phi}$	—	—	25	ns

\*Except  $\overline{IRQ}$  and  $\overline{NMI}$ , which require  $3 \text{ k}\Omega$  pullup load resistors for wire-OR capability at optimum operation. Does not include EXtal and Xtal, which are crystal inputs.

\*\*In power-down mode, maximum power dissipation is less than 40 mW.

#Capacitances are periodically sampled rather than 100% tested.

Fig. 7.15. 

## MC 6802 TIMING

Tidsreferencen for microcomputeren er CLOCK CYCLE TIME, der skal ligge fra  $1 \mu\text{s}$  til  $10 \mu\text{s}$  (se fig. 7.15).

Clockpulsen benævnes ved MC 6802 ENABLE (E), og denne benyttes derfor som reference ved angivelse af max/min tiden for LÆSE- og SKRIVEOPERATIONER (READ/WRITE), som computeren udfører.

## READ

Når processoren skal LÆSE en byte fra memory eller I/O enhed, skal  $R/\overline{W}$  gå HI, Adressen indstilles og VMA være på HI. Dette indstiller processoren automatisk, når en LÆSEINSTRUKTION er DEKODET og den tilhørende ADRESSE er hentet fra programmet (se evt. kap. 5).

Fra bagkanten af clockpulsen (E) skal der regnes med 270 ns ( $t_{AD}$ ) før  $R/\overline{W}$ , Adresse og VMA er etableret. Herefter vil der gå op til 530 ns ( $t_{acc}$ ), før den databyte, der skal læses, befinder sig på DATABUS. Fra bagkanten af (E) kan der altså gå op til 800 ns, før Data er gyldige og kan bruges på databussen (Data Valid).

Disse data skal herefter HOLDES mindst 100 ns ( $t_{DSR}$ ) på databussen for at give CPU'en mulighed for at læse dem.

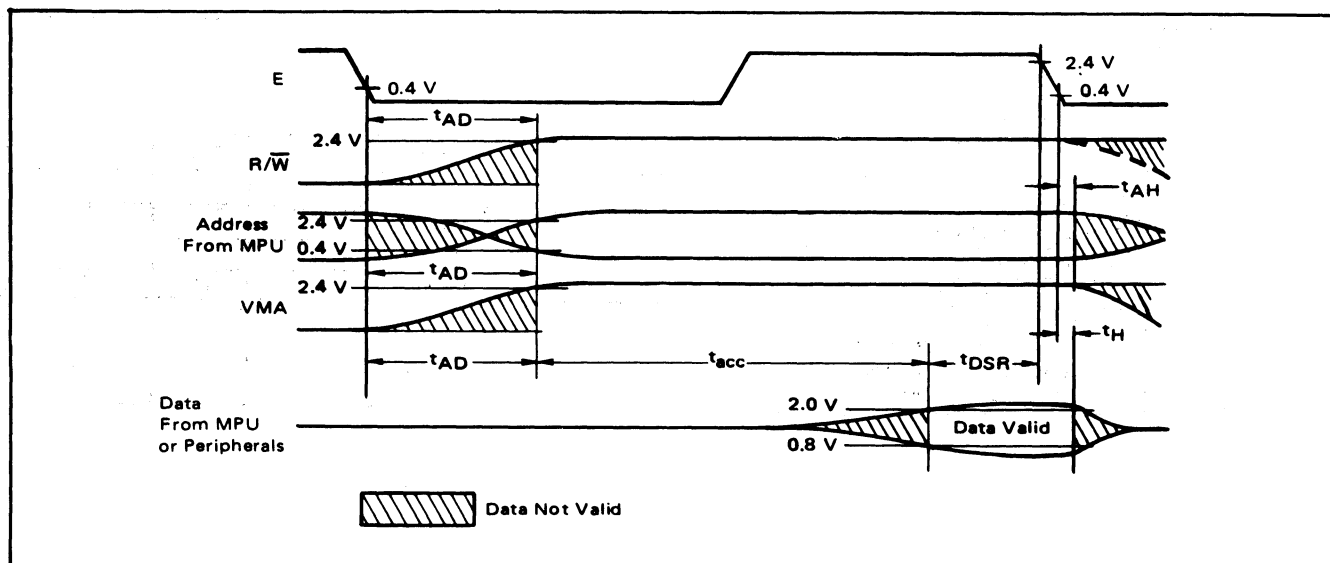
Faldetid på clock ( $t_{\phi} \leq 25$  nsek) og holdetider på data og adresse ( $t_H$  ca. 10–20 nsek.) bevirker, at korteste clockperiode bliver

$$t_{min} = t_{AD} + t_{acc} + t_{DSR} + t_{\phi} + t_H$$

$$t_{min} = 270 + 530 + 100 + 25 + 20 = 945 \text{ nsek.}$$

og heraf ses, hvorfor hurtigste tid er omkring 1  $\mu$ sek. (se fig. 7.16).

#### READ DATA FROM MEMORY OR PERIPHERALS



#### READ TIMING

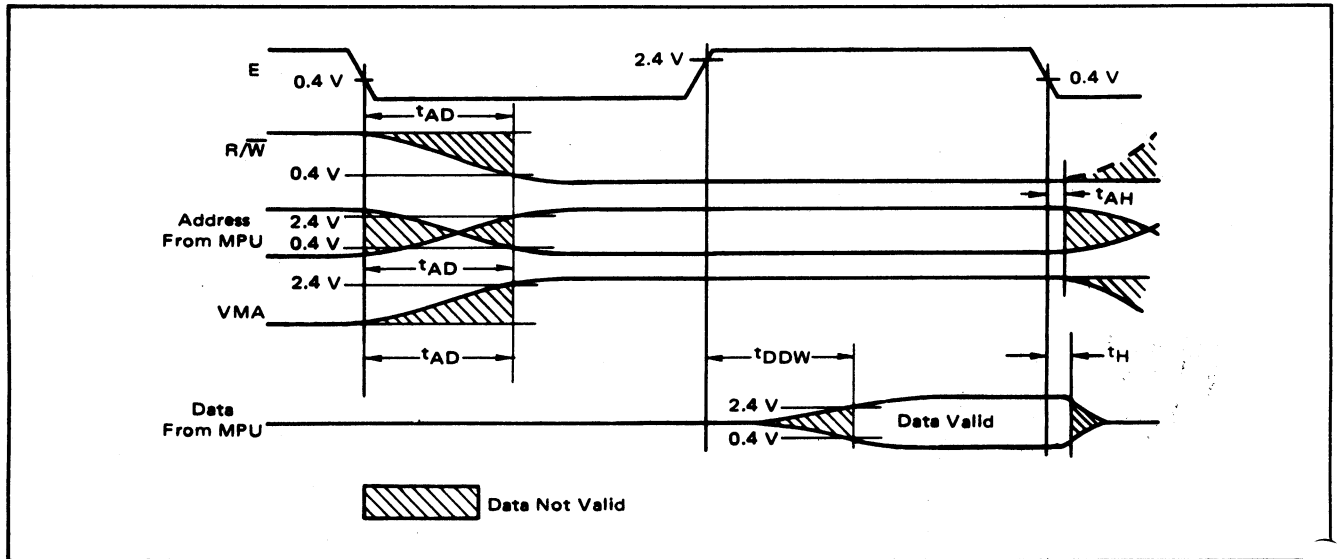
Characteristic	Symbol	Min	Typ	Max	Unit
Address Delay	$t_{AD}$	—	—	270	ns
Peripheral Read Access Time $t_{acc} = t_{ut} - (t_{AD} + t_{DSR})$	$t_{acc}$	—	—	530	ns
Data Setup Time (Read)	$t_{DSR}$	100	—	—	ns
Input Data Hold Time	$t_H$	10	—	—	ns
Processor Controls					
Processor Control Setup Time	$t_{PCS}$	200	—	—	ns
Processor Control Rise and Fall Time (Measured between 0.8 V and 2.0 V)	$t_{PCr}, t_{PCf}$	—	—	100	ns

Fig. 7.16. Ⓐ

#### WRITE

Når processoren skal SKRIVE til sit lager eller til I/O enhederne, skal  $R/\overline{W}$  være LO, Adressen indstillet og VMA gå HI, hvilket sker efter (E) bagkanten som ved READ (se fig. 7.17).

## WRITE DATA IN MEMORY OR PERIPHERALS



## WRITE TIMING

Characteristic	Symbol	Min	Typ	Max	Unit
Address Delay	$t_{AD}$	—	—	270	ns
Output Data Hold Time	$t_H$	20	—	—	ns
Address Hold Time (Address, R/W, VMA)	$t_{AH}$	20	—	—	ns
Data Delay Time (Write)	$t_{DDW}$	—	165	225	ns
Processor Controls					
Processor Control Setup Time	$t_{PCS}$	200	—	—	ns
Processor Control Rise and Fall Time (Measured between 0.8 V and 2.0 V)	$t_{PCr}, t_{PCf}$	—	—	100	ns

Fig. 7.17. Ⓐ

I SKRIVE-mode er det CLOCK FORKANT, der aktiverer DATA-UDLÆSNING fra CPU'en. Fra denne forkant kan der gå op til 225 nsek. ( $t_{DDW}$ ), før data er gyldige på DATABUS

## 6810 RAM

MCM 6810 er 128 bytes static Random Access Memory, der kan interfaces direkte til 6800 familiens CPU'er.

Blokdiagrammet for RAM'en ses i fig. 7.18 (forbindelser til BUS og benforbindelser) og i fig. 7.19 (internt blokdiagram).





Hukommelsens 128 bytes Adresseres med  $A_0 \rightarrow A_6$  (7 laveste ADR. ledninger).

## CHIP SELECT

Anvendes flere RAM kredsløb, vil placeringen af de enkelte i MEMORY MAP foregå ved forskellige adressers forbindelse til kredsløbenes CHIP SELECT input, hvoraf der findes 6 på MCM 6810. CHIP SELECT og TIMING foregår ofte „samlet”, hvilket bl.a. ses af fig. 7.1, hvor der anvendes 3 stk. MCM 6810 RAM med følgende forbindelser til ADR. og TIMING.

RAM	$\overline{CS}_5$	$\overline{CS}_4$	$CS_3$	$\overline{CS}_2$	$\overline{CS}_1$	$CS_0$
#1	$(\overline{A_{15}} \cdot VMA)$	0	1	$A_8$	$A_7$	(E)
#2	$(\overline{A_{15}} \cdot VMA)$	0	$A_7$	0	$A_8$	(E)
#3	$(\overline{A_{15}} \cdot VMA)$	0	$A_8$	0	$A_7$	(E)

Fig. 7.20.

Af dette ses RAM blive SELECTED, når

(E) = clock = HIGH  
VMA = HIGH  
 $A_{15}$  = LOW

og desuden:

RAM # 1:  $\overline{A_{15}} \cdot \overline{A_8} \cdot \overline{A_7} \Rightarrow 0\text{---} \text{---} 0 \text{ 0xxx xxxx}$   
RAM # 2:  $\overline{A_{15}} \cdot \overline{A_8} \cdot \overline{A_7} \Rightarrow 0\text{---} \text{---} 0 \text{ 1xxx xxxx}$   
RAM # 3:  $\overline{A_{15}} \cdot \overline{A_8} \cdot \overline{A_7} \Rightarrow 0\text{---} \text{---} 1 \text{ 0xxx xxxx}$

det vil sige, når:

RAM # 1 har ADR. HEX  $0000 \rightarrow 007F$   
RAM # 2 har ADR. HEX  $0080 \rightarrow 00FF$   
RAM # 3 har ADR. HEX  $0100 \rightarrow 017F$

Desuden er deres „SPEJL” beliggende på andre adresser helt op til ADR. 7F7F.



**MOTOROLA**  
**Semiconductors**

BOX 20912, PHOENIX, ARIZONA 85036

**MCM6810A**

(0 to 70°C; L or P Suffix)

**MCM6810AC**

(-40 to 85°C; L Suffix only)

## 128 X 8-BIT STATIC RANDOM ACCESS MEMORY

The MCM6810 is a byte-organized memory designed for use in bus-organized systems. It is fabricated with N-channel silicon-gate technology. For ease of use, the device operates from a single power supply, has compatibility with TTL and DTL, and needs no clocks or refreshing because of static operation.

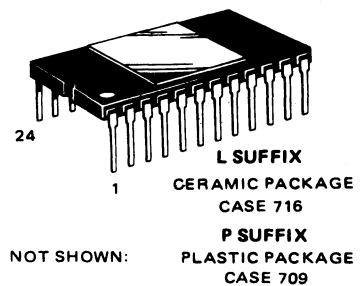
The memory is compatible with the M6800 Microcomputer Family, providing random storage in byte increments. Memory expansion is provided through multiple Chip Select inputs.

- Organized as 128 Bytes of 8 Bits
- Static Operation
- Bi-Directional Three-State Data Input/Output
- Six Chip Select Inputs (Four Active Low, Two Active High)
- Single 5-Volt Power Supply
- TTL Compatible
- Maximum Access Time = 350 ns — MCM6810AL1  
450 ns — MCM6810AL

**MOS**

(N-CHANNEL, SILICON-GATE)

**128 X 8-BIT STATIC  
RANDOM ACCESS MEMORY**



## PIN ASSIGNMENT

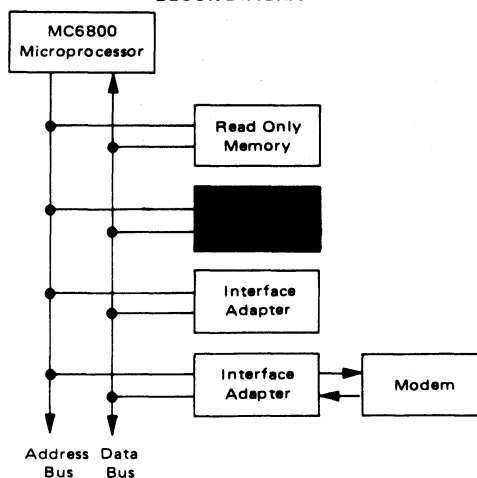
1	Gnd	0	V <sub>CC</sub>	24
2	D0		A0	23
3	D1		A1	22
4	D2		A2	21
5	D3		A3	20
6	D4		A4	19
7	D5		A5	18
8	D6		A6	17
9	D7		R/W	16
10	CS0		CS5	15
11	CS1		CS4	14
12	CS2		CS3	13

## ABSOLUTE MAXIMUM RATINGS (See Note 1)

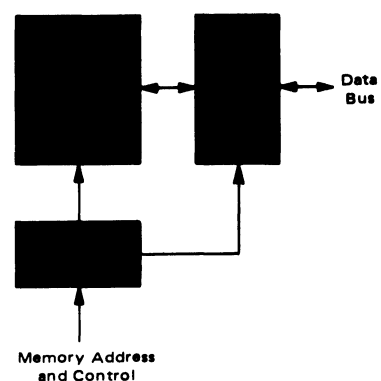
Rating	Symbol	Value	Unit
Supply Voltage	V <sub>CC</sub>	-0.3 to +7.0	Vdc
Input Voltage	V <sub>in</sub>	-0.3 to +7.0	Vdc
Operating Temperature Range	T <sub>A</sub>	0 to +70	°C
Storage Temperature Range	T <sub>stg</sub>	-65 to +150	°C

NOTE 1: Permanent device damage may occur if ABSOLUTE MAXIMUM RATINGS are exceeded. Functional operation should be restricted to RECOMMENDED OPERATING CONDITIONS. Exposure to higher than recommended voltages for extended periods of time could affect device reliability.

## M6800 MICROCOMPUTER FAMILY BLOCK DIAGRAM



## MCM6810A RANDOM ACCESS MEMORY BLOCK DIAGRAM



**MCM6810A**

(Full operating voltage and temperature range unless otherwise noted.)

## RECOMMENDED DC OPERATING CONDITIONS

Parameter	Symbol	Min	Nom	Max	Unit
Supply Voltage	V <sub>CC</sub>	4.75	5.0	5.25	Vdc
Input High Voltage	V <sub>IH</sub>	2.0	—	5.25	Vdc
Input Low Voltage	V <sub>IL</sub>	−0.3	—	0.8	Vdc

## DC CHARACTERISTICS

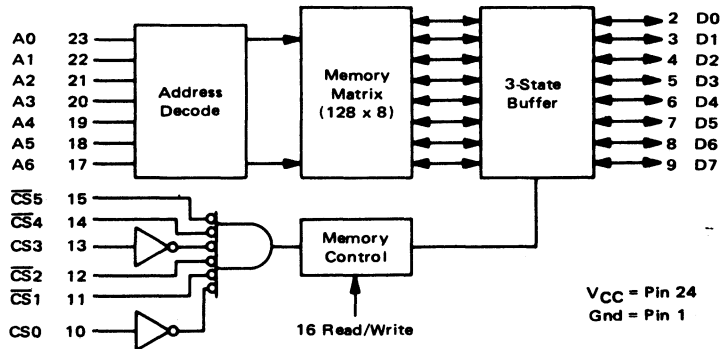
Characteristic	Symbol	Min	Typ	Max	Unit
Input Current ( $A_n$ , R/W, $CS_n$ , $\overline{CS}_n$ ) ( $V_{in} = 0$ to $5.25$ V)	$I_{in}$	—	—	2.5	$\mu A_{dc}$
Output High Voltage ( $I_{OH} = -205 \mu A$ )	$V_{OH}$	2.4	—	—	V <sub>dc</sub>
Output Low Voltage ( $I_{OL} = 1.6$ mA)	$V_{OL}$	—	—	0.4	V <sub>dc</sub>
Output Leakage Current (Three-State) ( $CS = 0.8$ V or $\overline{CS} = 2.0$ V, $V_{out} = 0.4$ V to $2.4$ V)	$I_{LO}$	—	—	10	$\mu A_{dc}$
Supply Current ( $V_{CC} = 5.25$ V, all other pins grounded, $T_A = 0^\circ C$ )	$I_{CC}$	—	—	70	mA <sub>dc</sub>
MCM6810AL MCM6810AL1		—	—	80	

**CAPACITANCE** (f = 1.0 MHz, T<sub>A</sub> = 25°C, periodically sampled rather than 100% tested.)

Characteristic	Symbol	Max	Unit
Input Capacitance	C <sub>in</sub>	7.5	pF
Output Capacitance	C <sub>out</sub>	12.5	pF

This device contains circuitry to protect the inputs against damage due to high static voltages or electric fields; however, it is advised that normal precautions be taken to avoid application of any voltage higher than maximum rated voltages to this high-impedance circuit.

### BLOCK DIAGRAM



# MCM6810A

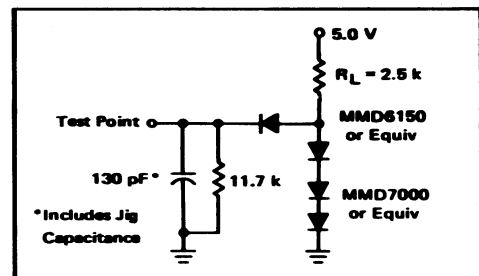
## AC OPERATING CONDITIONS AND CHARACTERISTICS

(Full operating voltage and temperature unless otherwise noted.)

### AC TEST CONDITIONS

Condition	Value
Input Pulse Levels	0.8 V to 2.0 V
Input Rise and Fall Times	20 ns
Output Load	See Figure 1

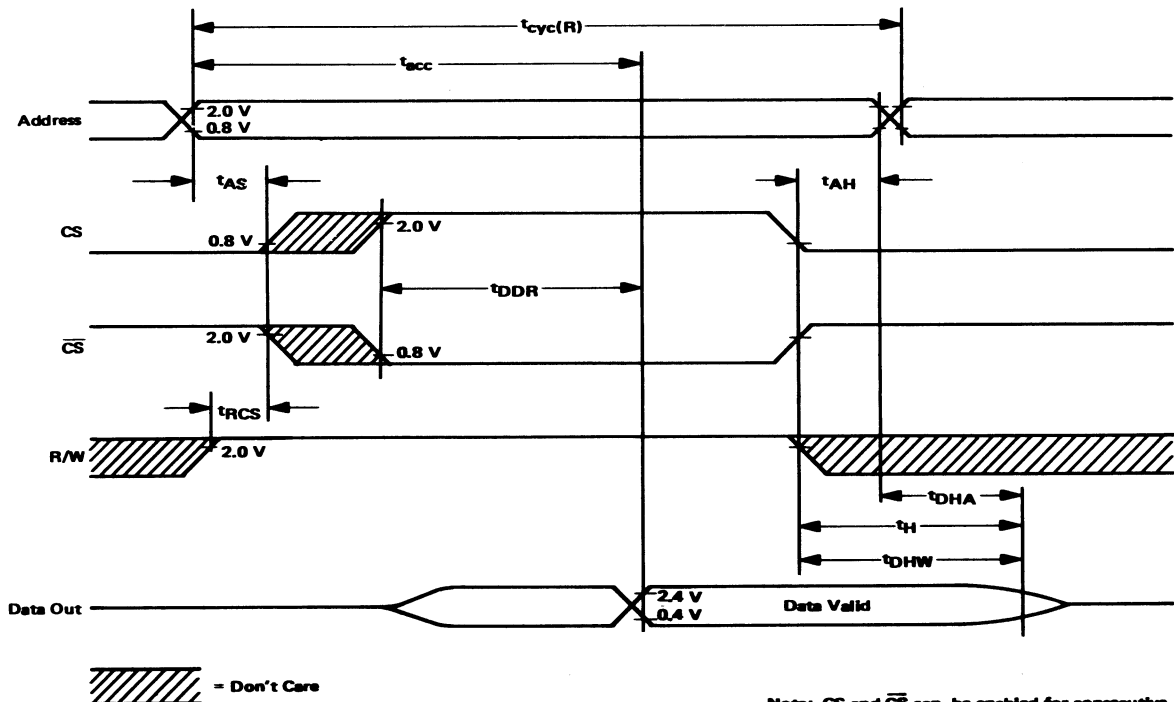
FIGURE 1 – AC TEST LOAD



### READ CYCLE

Characteristic	Symbol	MCM6810AL		MCM6810AL1		Unit
		Min	Max	Min	Max	
Read Cycle Time	$t_{cyc(R)}$	450	—	350	—	ns
Access Time	$t_{acc}$	—	450	—	350	ns
Address Setup Time	$t_{AS}$	20	—	20	—	ns
Address Hold Time	$t_{AH}$	0	—	0	—	ns
Data Delay Time (Read)	$t_{DDR}$	—	230	—	180	ns
Read to Select Delay Time	$t_{RCS}$	0	—	0	—	ns
Data Hold from Address	$t_{DHA}$	10	—	10	—	ns
Output Hold Time	$t_H$	10	—	10	—	ns
Data Hold from Write	$t_{DHW}$	10	80	10	60	ns

### READ CYCLE TIMING



Note: CS and  $\overline{CS}$  can be enabled for consecutive read cycles provided R/W remains at  $V_{IH}$ .



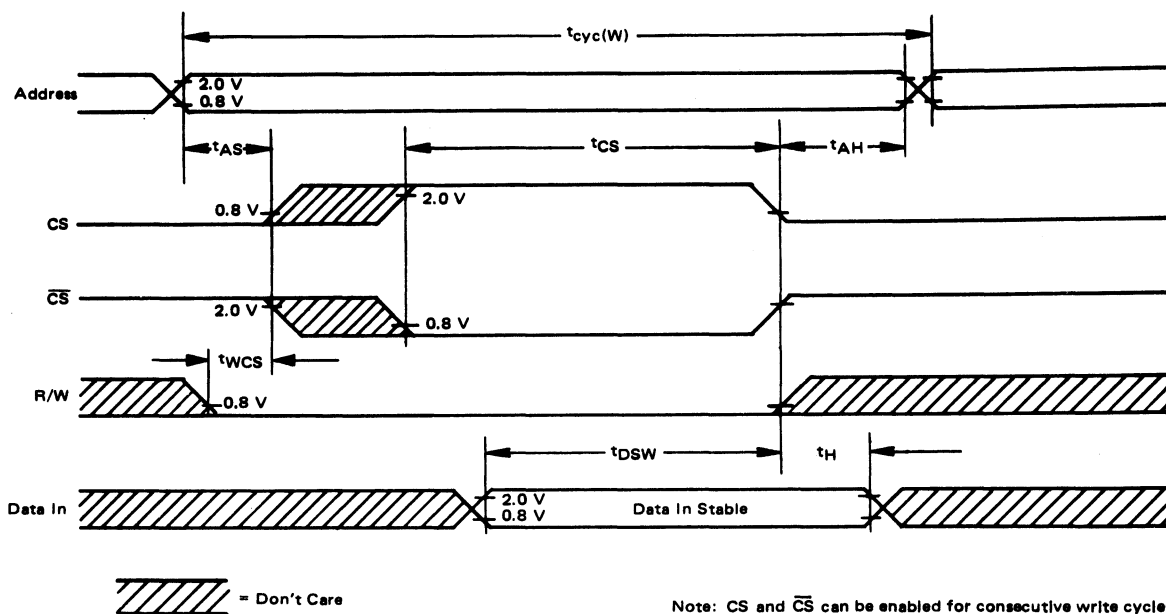
MOTOROLA Semiconductor Products Inc.

# MCM6810A

## WRITE CYCLE

Characteristic	Symbol	MCM6810AL		MCM6810AL1		Unit
		Min	Max	Min	Max	
Write Cycle Time	$t_{cyc(W)}$	450	—	350	—	ns
Address Setup Time	$t_{AS}$	20	—	20	—	ns
Address Hold Time	$t_{AH}$	0	—	0	—	ns
Chip Select Pulse Width	$t_{CS}$	300	—	250	—	ns
Write to Chip Select Delay Time	$t_{WCS}$	0	—	0	—	ns
Data Setup Time (Write)	$t_{DSW}$	190	—	150	—	ns
Input Hold Time	$t_H$	10	—	10	—	ns

## WRITE CYCLE TIMING



Note: CS and  $\overline{CS}$  can be enabled for consecutive write cycles provided R/W is strobed to  $V_{IH}$  before or coincident with the Address change, and remains high for time  $t_{AS}$ .

## MC 6821 PIA

MC 6821 Peripheral Interface Adapter (PIA) er et kredsløb til parallel-interface mellem CPU'ens 8 bits DATABUS og YDRE 8 bits parallelle DATABUSSER. Et eksempel på kobling mellem CPU og PIA kan ses i fig. 7.1, hvor der er anvendt MC 6820, d.v.s. en ældre udgave af PIA.

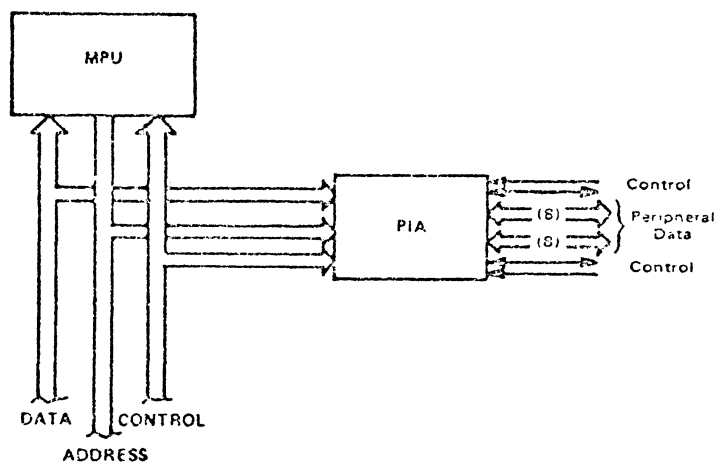
Den teoretiske anvendelse af parallel-interface er beskrevet i kap. 3.

## PIA ↔ CPU

Interface mellem CPU'en og PIA'en består af følgende:

DATA	1 stk.	8 bit bidirectional DATABUS
ADR.	3 stk.	CHIP SELECT (CS) ledere
	2 stk.	REGISTER SELECT (RS) ledere
	1 stk.	ENABLE ( $E = \text{clock } \phi_2$ ) leder
	1 stk.	RESET ( $\overline{\text{Reset}}$ ) leder
CONTROL	1 stk.	Read/write ( $R/\overline{w}$ ) leder
	2 stk.	INTERRUPT REQUEST ( $\overline{\text{IRQ}}$ ) ledere

se fig. 7.21.



MPU Parallel I/O Interface (AA)

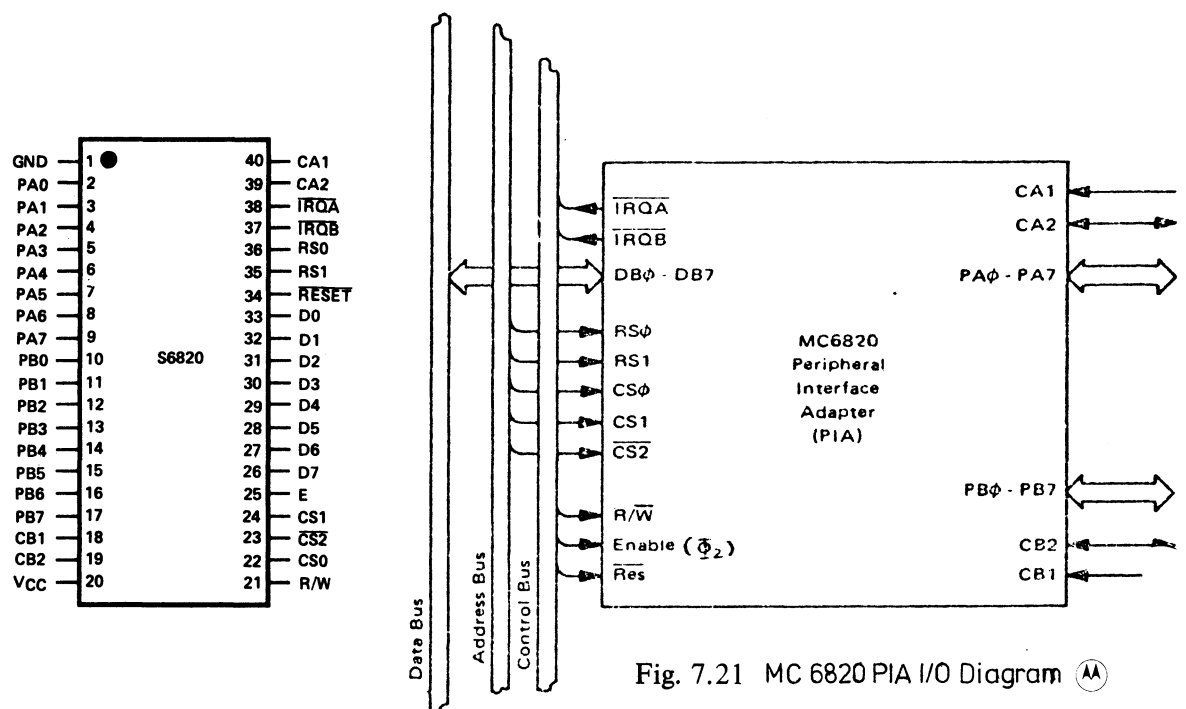


Fig. 7.21 MC 6820 PIA I/O Diagram

## PIA ↔ extern

Hver PIA har 2 stk. 8 bit bidirectionale perifere databusser (PA og PB) samt 2 x 2 kontrolledninger (CA og CB) til interface med ydre kredsløb. Forbindelsen kan f.eks. være etableret som vist i fig. 7.22.

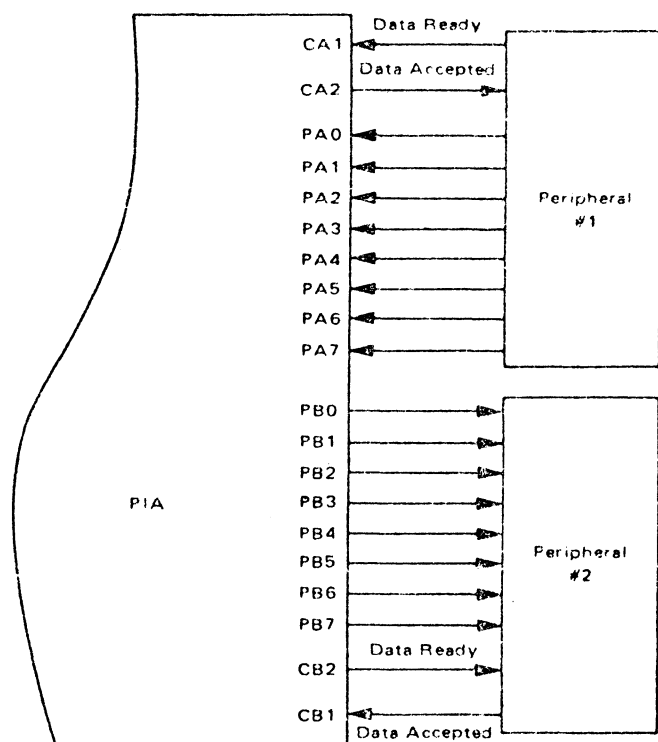


Fig. 7.22. Typical I/O Configuration

**PIA INTERFACE**

(Pin Configuration ses af fig. 7.21 side 7.20).

**BIDIRECTIONAL DATA**

D0  
D1  
D2  
D3  
D4  
D5  
D6  
D7

De bidirectionale dataledninger (D0–D7) benyttes til transporten af data mellem CPU'en og PIA'en. Databus OUTPUT DRIVERE er THREE STATE kredsløb, som bliver i den højimpedansede (OFF) tilstand på nær når CPU'en udfører en PIA LÆSE OPERATION. READ/write ledningen er i READ (HIGH) tilstanden, når PIA'en udpeges til en LÆSE-OPERATION.

E

ENABLE: Enablepulsen (E) er det eneste tidssignal, der tilføres PIA'en. Tidsbestemmelser for alle de andre signaler bestemmes ud fra forkant og bagkant på (E) pulsen.

(E) signalet vil normalt være afledt af MC 6800 fase 2 clocksignalet  $\phi_2$ . (E) pulsen anvendes til at sætte betingelser for input interrupt kontrolledningerne CA1, CA2, CB1 og CB2. Der skal forløbe MINDST en (E)-puls fra den inaktive kant og til den aktive kant på input signalet for at „hejse” interrupt flaget i PIA'ens kontrolregister.

R/ $\overline{W}$

READ/ $\overline{WRITE}$ : Dette signal genereres af CPU'en for at kontrollere retningen af DATA-flow på DATABUSSEN.

LO på PIA'ens R/ $\overline{W}$  ledning (WRITE) aktiverer input bufferne og data transmitteres fra CPU'en til PIA'en på (E) signalet, hvis PIA'en er selected.

HI på R/ $\overline{W}$  ledningen (READ) aktiverer PIA'ens output buffere for datatransport til DATABUSSEN. For at få data FRA PIA TIL DATABUS, skal desuden passende ADRESSE og ENABLE PULSEN være til stede.

$\overline{RESET}$

$\overline{RESET}$ : Den aktivt LOW  $\overline{RESET}$  ledning benyttes til at „rense” alle registerbits i PIA'en til logisk 0. Ledningen kan benyttes til RESET ved POWER-ON og som MASTER RESET under DRIFT af systemet.

CS0  
CS1  
 $\overline{CS2}$

CHIP SELECT: Disse tre input signaler benyttes til at udvælge (udpege) PIA'en. CS0 og CS1 skal være HI og  $\overline{CS2}$  skal være LO for at udvælgelsen sker. Data transport foregår derefter under kontrol af (E) og R/ $\overline{W}$  signalerne.

CHIP SELECT ledningerne skal være STABILE, medens (E) pulsen er til stede.

RS0  
RS1

PIA REGISTER SELECT: De to register select ledninger benyttes til at udvælge de forskellige registre inde i PIA'en.

Disse to ledninger benyttes i forbindelse med det interne kontrol register til at udvælge et enkelt register, hvor der skal LÆSES eller SKRIVES i.

Register select ledninger skal være stabile, når (E) pulsen er til stede, medens man befinder sig i READ- eller WRITE CYCLE.



**IRQA**  
**IRQB**

**INTERRUPT-REQUEST:** De aktivt LO interrupt request ledninger giver INTERRUPT til CPU'en enten direkte eller gennem interrupt prioritetskredsløb.

Interruptledningerne er med „open source” (ingen belastningsmodstand på Chip'en) og er i stand til at „SINK” en strøm på 1,6 mA fra en extern kilde. Dette tillader alle interrupt request ledninger at blive forbundet sammen som WIRED-AND.

Hver interrupt request ledning har to interne interrupt flag bits, der kan forårsage  $\overline{IRQ}$ -ledningen til at gå LO. Hvert af flagene er „forbundet” med en bestemt perifer interrupt ledning (CA eller CB).

Der findes også 4 interrupt aktiveringsbit inde i PIA'ens kontrolregistre. Disse bits kan benyttes til at forhindre interrupt fra en bestemt perifer enhed.

CPU'ens måde at betjene et interrupt er ved hjælp af software (interrupt programmet) at foretage prioriteret afsøgning af PIA'ernes kontrolregistre og teste disse for INTERRUPT FLAG BITS, som er SET.

Interrupt flaget sættes på LO (O), når CPU'en udfører en instruktion med at LÆSE DATAREGISTRET i PIA'en.

**PA0**  
**PA1**  
**PA2**  
**PA3**  
**PA4**  
**PA5**  
**PA6**  
**PA7**

**AFSNIT A YDRE DATA:** Hver af de ydre dataledninger kan programmeres til at virke som input eller som output. Output etableres ved at sætte et 1-tal i de tilhørende DATA DIRECTION REGISTER bit. Et 0 i et bit i DDR bevirker, at den tilsvarende ydre ledning bliver et input.

Når data på de ydre dataledninger læses ind i CPU'en af en LOAD instruktion, vil de ledninger, der er bestemt som input ledninger, føre data direkte til den interne databus og ind i det register, der er udpeget i CPU'en. Ydre inputledninger belaster højst som 2 stk. standard TTL (fan in = 2).

Som OUTPUT er ledningerne i stand til at drive 2 stk. standard TTL belastninger (fan out = 2). Med en OUTPUT DATA INSTRUCTION (STORE) føres data fra CPU via databus til PIA'ens DATA REGISTER.

Et „1-tal” som output vil forårsage et HIGH på den tilsvarende ydre dataledning og et „0” vil give LOW.

De data, der ligger i DATA REGISTRET og hvor ydre ledninger er programmeret som output, kan godt læses af CPU'en (LOAD from PIA DATA REG). De ydre dataledninger kan dog være belastet på en sådan måde, at der læses andre logiske værdier end de, der blev skrevet i registret (hvis  $V_{out}$  ligger mellem 2,0 V og 0,8 V).

PB0  
PB1  
PB2  
PB3  
PB4  
PB5  
PB6  
PB7

**AFSNIT B YDRE DATA:** De ydre dataledninger i afsnit B af PIA'en, kan programmeres til at virke enten som input eller som output på samme måde som ved A-siden.

Udgangsførstærkerne, der driver B-siden, afviger fra de udgangstrin, der anvendes i A-siden. B-siden har THREE-STATE output, som går i OFF (HIGH-Z) tilstanden, når de ydre dataledninger benyttes som input. Som output er B-siden TTL kompatibel eller kan belastes med 1 mA ved 1,5 V<sub>out</sub>, hvilket bevirker, at B-siden kan drive switch-transistorer direkte.

## YDRE KONTROLLEDNINGER

CA1  
CB1

**INTERRUPT INPUT:** De ydre ledninger CA1 og CB1 kan kun benyttes som input ledninger. De kan SÆTTE interrupt flagene (bit 7) i PIA'ens kontrolregistre. Den aktive signalkant på input signalerne kan programmeres af de respektive kontrolregistre.

CA2

**YDRE KONTROL:** Den ydre kontrolledning CA2 kan programmeres til at virke som interrupt input eller som perifer kontrol output. Funktionen af CA2 programmeres med kontrolregister A (CRA).

CB2

**YDRE KONTROL:** Den ydre kontrolledning CB2 kan programmeres til at virke som interrupt input eller som perifer kontrol output. Funktionen af CB2 programmeres fra kontrolregister B (CRB).

## FORSYNINGSSPÆNDING

PIA MC 6821 skal tilsluttes

$$V_{CC} = +5 \text{ V}$$

$$V_{SS} = 0 \text{ V (GROUND)}$$

## PIA REGISTRE

Inde i PIA'en findes ialt 6 stk. 8 bit registre med 3 registre til hver af halvdelene benævnt A-siden og B-siden (se fig. 7.23).

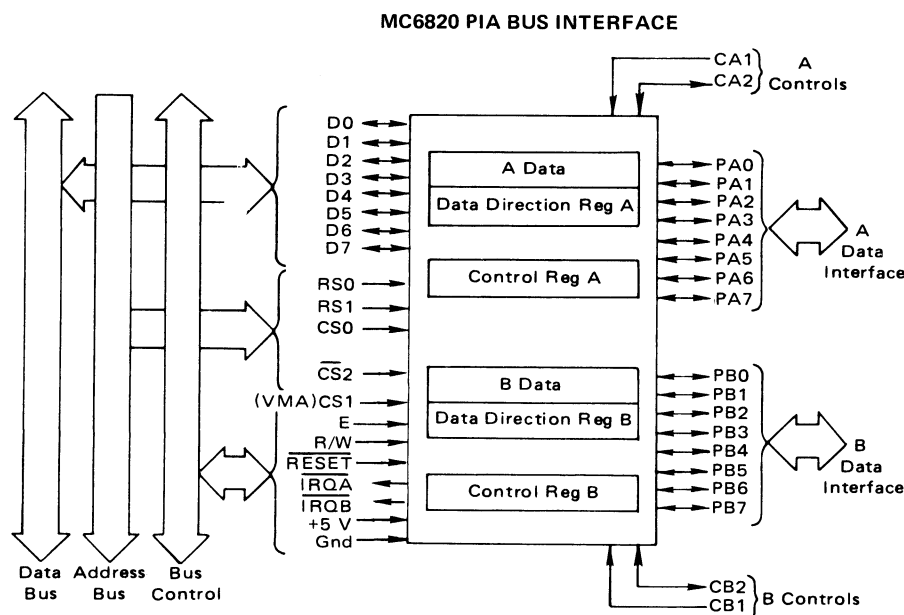


Fig. 7.23. AA

**PERIPHERAL DATA  
REGISTRET (DR)**

DATA REGISTRET (DR) benyttes som kommunikationsregister mellem CPU'ens DATABUS og den YDRE DATABUS.

**DATA DIRECTION  
REGISTRET (DDR)**

DATA RETNINGSREGISTRET (DDR) benyttes til at fastlægge retningen på de enkelte ledninger på den YDRE DATAPORT, idet disse ledninger kan programmeres til at være enten INPUT eller OUTPUT. Programmeringen foregår sædvanligvis i programmets start, hvor enhederne initialiseres til den ønskede virkemåde.

Hvert bit i DDR svarer til en port-ledning med samme nummer. Er et bit = 0 i DDR, er den tilsvarende port-ledning et INPUT. Hvis registrets bit = 1, er den tilsvarende ledning et OUTPUT. Fordelingen mellem IN- og OUTPUT i hver af portene er vilkårlig og kun bestemt af bit-mønstret i DDR.

**CONTROL REGISTRET (CR)**

CONTROL REGISTRET (CR) indeholder et bitmønster, som micro-computerprogrammet skriver i dette register i initialiseringsfasen. Controlregistret bestemmer virkningen af de eksterne controlled-ninger C1 og C2 og CPU DATABUSSEN's forbindelse til enten DR eller DDR. Desuden er to af controlregistrets bit interrupt flag, hvor CPU'en kan se, om der er IRQ fra den port, som kontrolregistret tilhører (se fig. 7.24).

		CONTROL WORD FORMAT							
		7	6	5	4	3	2	1	0
CRA	IRQA1	IRQA2	CA 2 Control			DDRA Access	CA 1 Control		
CRB	IRQB1	IRQB2	CB 2 Control			DDRB Access	CB 1 Control		

Fig. 7.24. (AA)

**PIA ADRESSE**

Register Select (RS1 og RS0) benyttes til at adressere PIA'ens interne registre. Der benyttes sædvanligvis forbindelserne RS1 = A<sub>1</sub> og RS0 = A<sub>0</sub>. Herved opnås 4 forskellige adresser på den enkelte PIA. DATA REGISTER (DR) og DATA DIRECTION REGISTER (DDR) ligger set fra CPU'en på samme adresse (se fig. 7.23), men controlregistrets bit 2 (CR bit 2) bestemmer, om CPU'ens DATABUS er koblet til DATA REGISTRET (CR bit<sub>2</sub> = 1) eller er koblet til DATA DIRECTION REGISTRET (CR bit<sub>2</sub> = 0).

Opstillingen fig. 7.1 kan benyttes som eksempel på adresseringen til PIA.

						MIN. ADRESSE			
	$\overline{CS_2}$	CS1	CS0	RS1	RS0	A <sub>15</sub>	A <sub>11</sub>	A <sub>7</sub>	A <sub>3</sub>
PIA # 1	A <sub>15</sub>	VMA	A <sub>11</sub>	A <sub>1</sub>	A <sub>0</sub>	0XXX	1XXX	XXXX	XX--
PIA # 2	A <sub>15</sub>	VMA	A <sub>12</sub>	A <sub>1</sub>	A <sub>0</sub>	0XX1	XXXX	XXXX	XX--

Fig. 7.25.

PIA # 1      0800 → 0803      (HEX)  
 PIA # 2      1000 → 1003      (HEX)

Dette giver følgende registeradresser:

PIA # 1:

DATA REGISTER A (DRA)    – \$ 0800      (a)  
 DATA RETN.REG. A (DDRA) – \$ 0800      (b)  
 CONTROL REG.    A (CRA)    – \$ 0801  
  
 DATA REGISTER B (DRB)    – \$ 0802      (c)  
 DATA RETN. REG.B (DDRB) – \$ 0802      (d)  
 CONTROL REG.    B (CRB)    – \$ 0803

idet register select (RS1 og RS0) har følgende funktion:

RS1	udpeger	RS0	CR-bit 2	udpeger
0	A-siden	0	0	Data Direction Reg. (DDR)
1	B-siden	0	1	Data Register (DR)
		1	0	} Control Reg. (CR)
		1	1	

(a): CRA bit 2 = 1                      (c): CRB bit 2 = 1  
 (b): CRA bit 2 = 0                      (d): CRB bit 2 = 0

og tilsvarende for PIA # 2:

DRA    – \$ 1000    når CRA bit 2 = 1  
 DDRA – \$ 1000    når CRA bit 2 = 0  
 CRA    – \$ 1001  
 DRB    – \$ 1002    når CRB bit 2 = 1  
 DDRB – \$ 1002    når CRB bit 2 = 0  
 CRB    – \$ 1003

#### OMBYTNING AF A<sub>1</sub> og A<sub>0</sub> PÅ REGISTER SELECT

I visse tilfælde kan det være ønskeligt, at dataportene (PA og PB) og dermed DATA REGISTRENE (DRA og DRB), ligger på to adresser umiddelbart efter hinanden. Dette er f.eks. tilfældet, når der arbejdes med 16 bit dataord, der kommunikeres mellem PIA dataportene og indexregistret i CPU'en.

Dataregistrenes placering ved siden af hinanden adresse-mæssigt opnås ved at ombytte RS1 og RS0 på adresseforbindelserne, således at

RS1 tilsluttes A<sub>0</sub>  
 RS0 tilsluttes A<sub>1</sub>

Dette giver følgende rækkefølge for registre, idet eksemplet fra foregående sider anvendes til ombytning af register select ledningernes adresseforbindelser.

Herved opnås følgende adresser for kredsløbenes registre:

PIA # 1	DRA	– \$ 0800	}	REG.A
	DRB	– \$ 0801		DATA REG.B
	CRA	– \$ 0802	}	REG.A
	CRB	– \$ 0803		CONTR.REG.B

```

PIA # 2  DRA  - $ 1000 } DATA
          DRB  - $ 1001 } REG.
          CRA  - $ 1002 } CONTR.
          CRB  - $ 1003 } REG.

```

Data på 16 bit kan herefter indlæses i index registret (evt. i SP) fra dataportene, f.eks. som

```

LDX $ 0800 =>
index reg. MSB= DATA fra $ 0800
index reg. LSB= DATA fra $ 0801

```

## PIA INITIALISERING

Initialisering af PIA'en kan belyses ved at benytte fig. 7.1 og antage følgende virkemåde:

```

PIA # 1  (Adr. $ 0800 -> 0803)
PA0 -> PA3 = input
PA4 -> PA7 = output
PB0 -> PB7 = output
CA1, CA2, CB1, CB2 = DISABLED

```

```

PIA # 2  (Adr. $ 1000 - 1003)

PA0 -> PA7 = input
PB0 -> PB7 = output
CA1, CA2, CB1, CB2 = DISABLED

```

## PROGRAM TIL INITIALISERING

PROGRAM VERSION 1 (anvender ACC A)

LABEL	OPCODE	OPERAND	COMMENTS
INPIA1	CLR	\$ 0801	clear CRA PIA1
	CLR	\$ 0803	clear CRB PIA1
	LDA A	#\$ F0	{ 4 OUTPUT, 4 INPUT
	STA A	\$ 0800	
	LDA A	#\$ 04	{ CRA bit 2 = 1 =>
	STA A	\$ 0801	
	LDA A	#\$ FF	{ 8 OUTPUT på
	STA A	\$ 0802	
	LDA A	#\$ 04	{ CRB bit 2 = 1 =>
	STA A	\$ 0803	
INPIA2	CLR	\$ 1000	clear CRA PIA2
	CLR	\$ 1002	clear CRB PIA2
	LDAA	#\$ 00	{ 8 INPUT
	STA A	\$ 1000	
	LDA A	#\$ 04	{ CRA bit 2 = 1 =>
	STA A	\$ 1001	
	LDA A	#\$ FF	{ 8 OUTPUT
	STA A	\$ 1002	
	LDA A	#\$ 04	{ CRB bit 2 = 1 =>
	STA A	\$ 1003	

Fig. 7.26.

## PROGRAM VERSION 2 (anvender index registret)

LABEL	OPCODE	OPERAND	COMMENTS
INPIA1	CLR	\$ 0801	
	CLR	\$ 0803	
	LDX	#\$F004	
	STX	\$ 0800	
	LDX	#\$FF04	
	STX	\$ 0802	
INPIA2	CLR	\$ 1001	
	CLR	\$ 1003	
	LDX	#\$0004	
	STX	\$ 1000	
	LDX	#\$FF04	
	STX	\$ 1002	

Fig. 7.27.

### CONTROL TERMINALERNE CA1, CA2, CB1 og CB2's PROGRAMMERING

For at kunne anvende KONTROLLEDNINGERNE (CA1, CA2, CB1, CB2) på PIA'ens udvendige interface-side er kendskab til kontrolregistre og de enkelte bit i dette nødvendigt.

CONTROL REGISTER A (CRA)

7	6	5	4	3	2	1	0
IRQA1	IRQA2	CA2 Control			DDRA	CA1 Control	

Fig. 7.28. (AA)

### CONTROL REGISTER A (CRA)

#### CA1 CONTROL (bit 0 og 1):

Den ydre kontrolledning CA1 kan kun virke som input ledning og kan benyttes til at give interrupt til CPU'en afhængig af stillingen på kontrolregistrets bit 0 og 1. CA1 påvirker også interrupt flag bit (bit 7 i CRA) ved at „SÆTTE” dette flag, når der sker logisk skift på CA1.

Når CPU'en læser DATA REGISTER A i PIA'en, cleares samtidig IRQA FLAG (CRA bit 7 og 6).

Fig. 7.29.

Pulsform på INTERRUPT ledning CA1	CRA bit 1 (CA1 kant)	CRA bit 0 ( $\overline{\text{IRQA}}$ ON/OFF)	CRA bit 7 (IRQA FLAG)	$\overline{\text{IRQA}}$ til CPU'en
	0	0	1	NEJ: (Forbliver HI)
	0	1	1	JA: (Går LO)
	1	0	1	NEJ: (Forbliver HI)
	1	1	1	JA: (Går LO)

Alle andre kombinationer af CA1 pulsskift og status af bit 0 og 1 vil blive ignoreret.

**BIT NR. 1**

Som vist i ovenstående tabel er CRA bit 1 KANT PROGRAM BIT.

„0” i CRA bit 1 programmerer CRA bit 7 (IRQA FLAG) til at gå HI på NEGATIV KANT af CA1 signalet.

„1” i CRA bit 1 programmerer CRA bit 7 (IRQA FLAG) til at gå HI på POSITIV KANT af CA1 signalet.

**BIT NR. 0**

CRA bit 0 er maskeringsbit for interrupt til CPU'en.

„0” i CRA bit 0 bevirker, at  $\overline{\text{IRQA}}$  (interrupt til CPU'en) IKKE påvirkes, når CRA bit 7 (IRQA FLAG) går HI.

„1” i CRA bit 0 bevirker, at  $\overline{\text{IRQA}}$  går på LOW (og sender  $\overline{\text{IRQ}}$  til CPU), når CRA bit 7 går HI ved påvirkning af CA1 med en kant.

Data Direction Access Control (DDRA) (bit 2)**BIT NR. 2**

Dette bit bruges i forbindelse med Register select (RS1 og RS0) til at vælge enten TRANSMISSIONS DATA REGISTER A (Peripheral Data Reg.A = DRA) eller DATA RETNINGS REGISTER A (Data Direction Reg. A = DDRA). Udvalgelsen kan ses af nedenstående tabel:

RS1	RS0	CRA bit 2	Udvalgt Register
0	0	0	Data Retnings Register A
0	0	1	DATA REGISTER A
0	1	X	CONTROL REG. A

Fig. 7.30.

### CA2 kontrol bits (bit 3, 4 og 5) med CA2 som Interrupt Input (bit 5 = 0)

Bit nr. 3, 4 og 5 i kontrolregistret CRA bestemmer funktionen af CA2 linien:

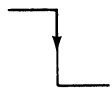
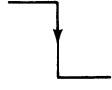


Pulsform på input CA2	CRA bit 5 (OUT/ $\overline{\text{IN}}$ )	CRA bit 4 (KANT)	CRA bit 3 ( $\overline{\text{IRQA}}$ = ON/ $\overline{\text{OFF}}$ )	CRA bit 6 (IRQA FLAG)	$\overline{\text{IRQA}}$ til CPU'en
	0	0	0	1	NEJ: (forbliver HI)
	0	0	1	1	JA: (går LO)
	0	1	0	1	NEJ: (forbliver HI)
	0	1	1	1	JA: (går LO)

Fig. 7.31.

Alle andre kombinationer af CA2 skift og stillinger af bit 3 og bit 4 vil blive ignoreret.

#### BIT NR. 5

Bit 5 i kontrolregister A er  $\text{OUTPUT}/\overline{\text{INPUT}}$  programmeringsbit for CA2. Hvis bit 5 indeholder logisk 0, er CA2 programmeret som INTERRUPT INPUT ledning.

#### BIT NR. 3 og 4

Når CA2 er programmeret som input ledning, er programmeringen af bit 4 og bit 3 svarende til den, der blev anvendt ved bit 1 og bit 0 programmering af CA1.

#### CA2 benyttet som OUTPUT (bit 5 = 1)

#### BIT NR. 5

Når bit 5 er på logisk 1, er CA2 programmeret som OUTPUT og bit 4 og 3 benyttes til at programmere til en af følgende 3 virkemåder:

HANDSHAKE MODE	[CRA = 100]	} [bit nr. = 5,4,3]
PULSE MODE	[CRA = 101]	
BIT 3 FOLLOWING MODE	[CRA = 11X]	

De fire opstillinger, der benytter CA2 som output er beskrevet nedenfor. I alle fire opstillinger forbliver IRQA2 FLAG (CRA bit 6) påLO.



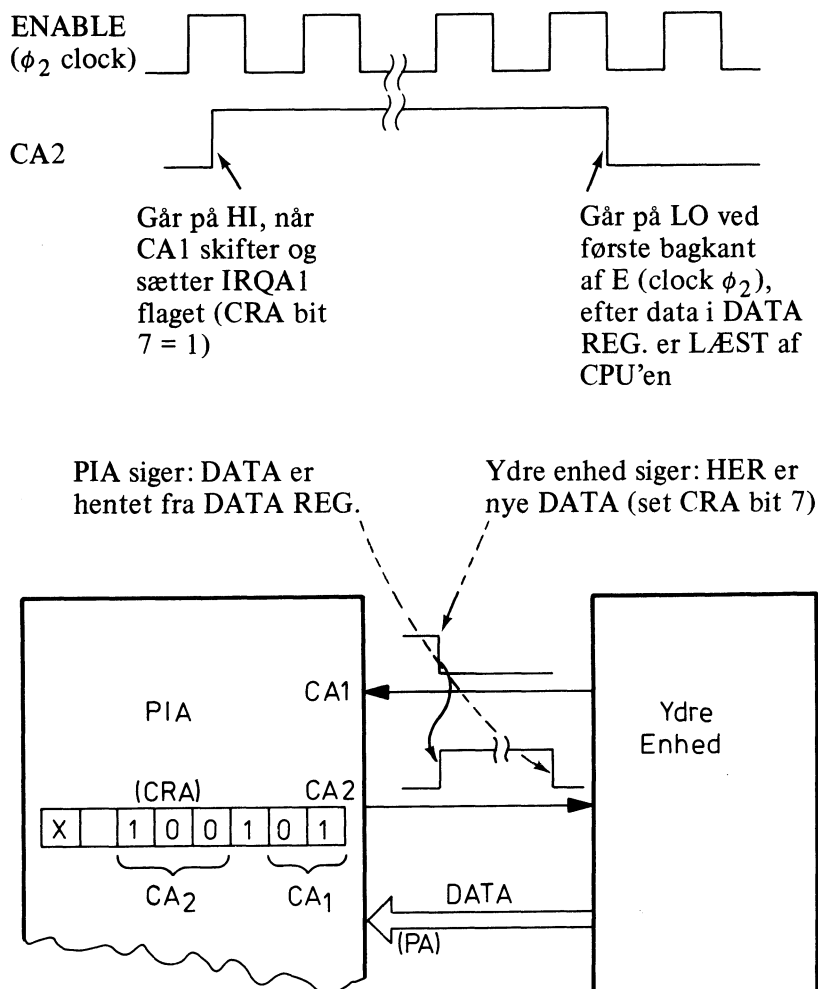
**HANDSHAKE (CRA)****HANDSHAKE (CRA bit 5, 4, 3 = 100):**

Fig. 7.32.

Handshake mode benyttes f.eks. mellem et keyboard og en PIA. Keyboard sender LO til CA1, når DATA BYTE afleveres til DATA REG. A (DRA) i PIA'en. Herved sættes IRQA FLAG (CRA bit 7) HI og dette bevirker, at CA2 OUTPUT også går HI. Det fortæller den perifere enhed (keyboard), at PIA'en IKKE kan modtage flere oplysninger.

Når CPU'en er ude at læse PIA'ens DATA REG. A (på grund af  $\overline{\text{IRQA}} \rightarrow \text{LO}$ , fordi CRA bit 1 og 0 = 01), RESETTES IRQA FLAG (CRA bit 7  $\rightarrow \text{LO}$ ), hvilket bevirker, at CA2 også går LO og fortæller keyboard, at nu er PIA'en klar til at modtage nye DATA.

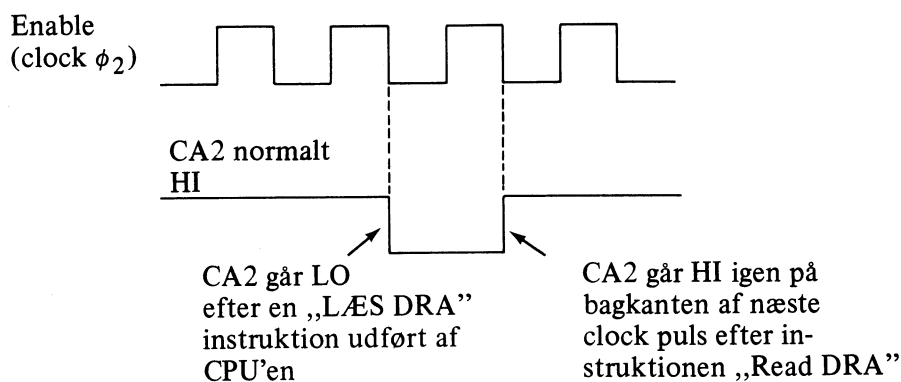
**PULSE MODE (CRA)****PULSE MODE (CRA bit 5, 4, 3 = 101):**

Fig. 7.33.

Når CPU'en læser PIA'ens DATA REGISTER, går CA2 Low for at fortælle den perifere enhed, at der nu er klar i PIA'en til nye data. LO-pulsen, der genereres af PIA'en kan herved benyttes til at udløse afsendelse af næste DATA fra den perifere enhed.

**FOLLOW MODE****FOLLOW MODE (CRA bit 5, 4, 3 = 110 eller 111)**

I denne mode vil CA2 følge stillingen af bit 3 og skift på CA2 OUTPUT opnås ved, at CPU'en skriver til controlregistret.

CRA bit nr.	5	4	3	CA2
	1	1	0	0
	1	1	1	1

**CONTROL REGISTER B (CRB)**

CONTROL REGISTER B (CRB)						
7	6	5	4	3	2	1 0
IRQB1	IRQB2	CB2 Control			DDRB	CB1 Control

Fig. 7.34. Ⓐ

For programmering af dette register og for virkemåden på CB1 og CB2 gælder NØJAGTIGT DE SAMME FORHOLD som beskrevet for kontrolregister A på de foregående sider.

Det vil sige, at i den foregående behandling af Register A skal bogstavet A blot erstattes med bogstav B for at give beskrivelsen for B-sidens kontrolregister.

Dette gælder i alle forhold med undtagelse af CB2 som OUTPUT i HANDSHAKE MODE og i PULS MODE.

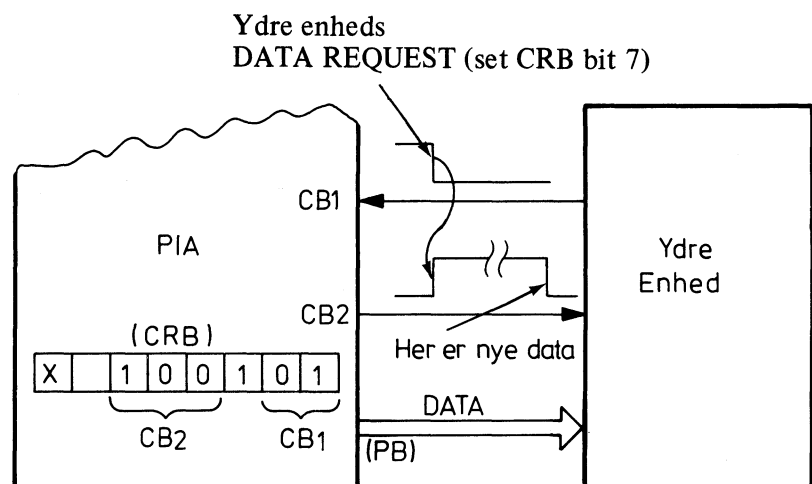
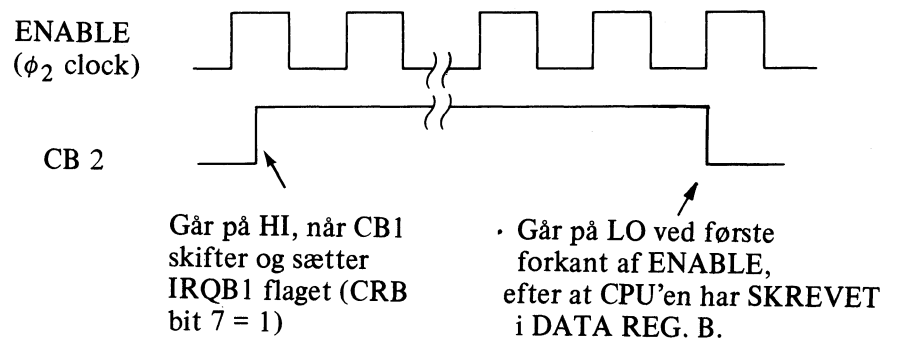
**HANDSHAKE (CRB)****HANDSHAKE (CRB bit 5, 4, 3 = 100):**

Fig. 7.35.

(sammenlign evt. med fig. 7.32. for A-siden)

Af dette ses det, at B-siden benyttes som output fra microcomputeren til ydre enheder.

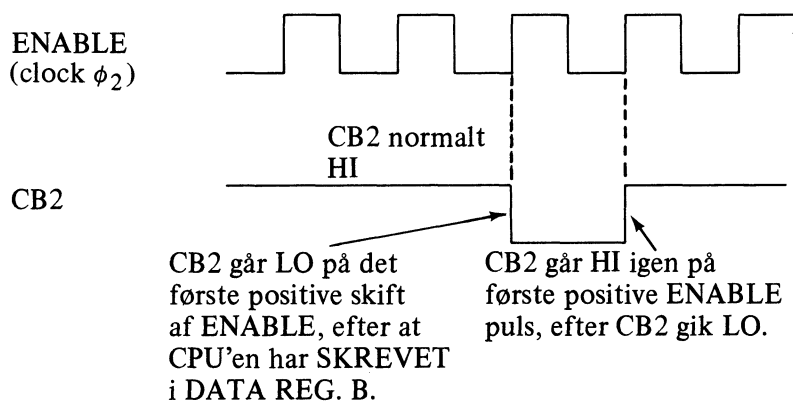
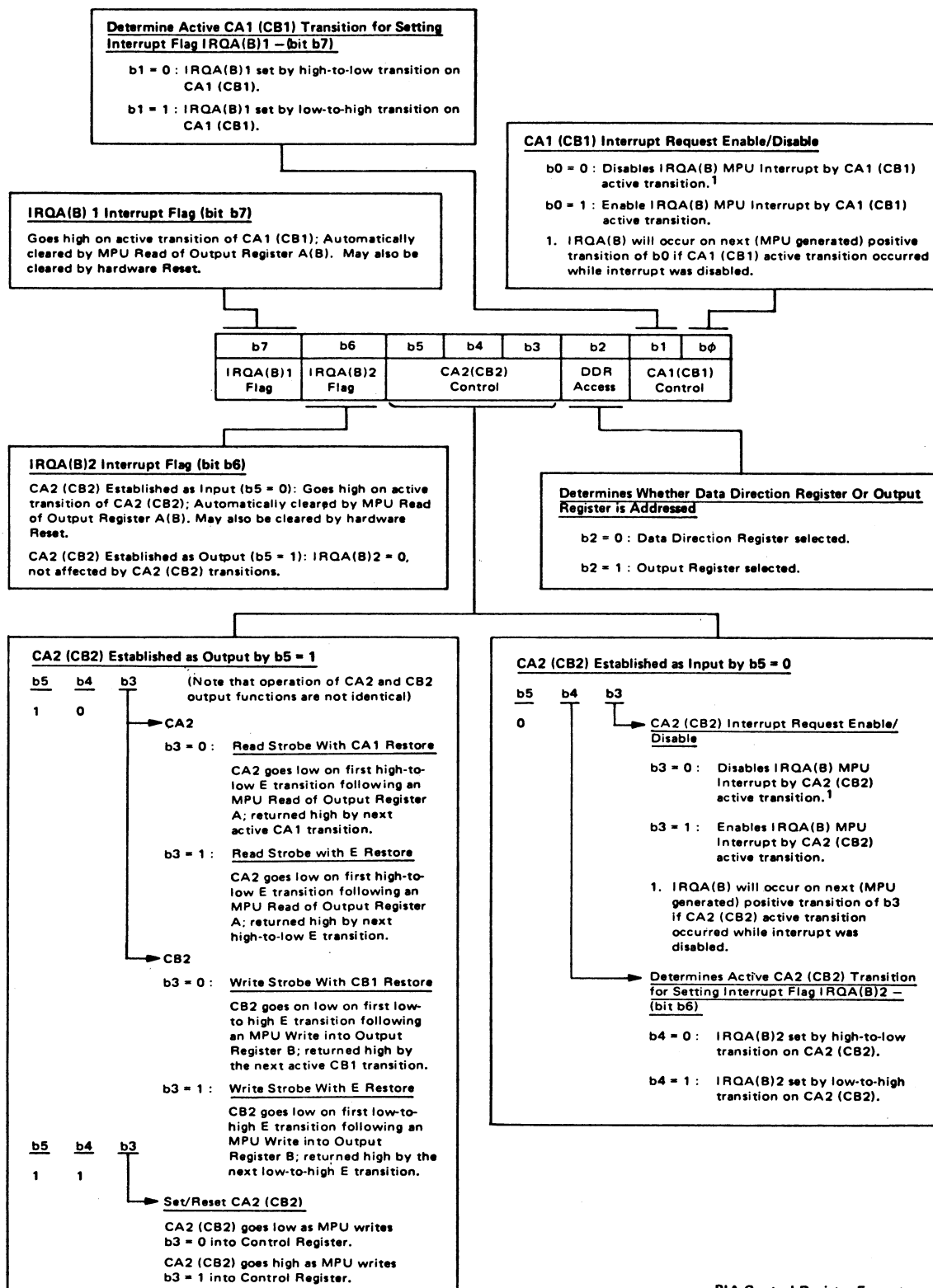
**PULS MODE (CRB)****PULS MODE (CRB bit 7, 4, 3 = 101):**

Fig. 7.36.


(sammenlign evt. med fig. 7.33 for A-siden)

CB2 pulsen tilføres den ydre enhed for at fortælle, at CPU'en har skrevet DATA i PIA'ens DATA REG. B og pulsen kan benyttes af perifere enheder til at aflæse de data, der står på PIA'ens port (= data i PIA DRB).

### OVERSIGT OVER PROGRAMMERING AF PIA CONTROL REGISTRE



PIA Control Register Format

Fig. 7.37. 





**MOTOROLA**  
**Semiconductors**

### PERIPHERAL INTERFACE ADAPTER (PIA)

The MC6821 Peripheral Interface Adapter provides the universal means of interfacing peripheral equipment to the MC6800 Micro-processing Unit (MPU). This device is capable of interfacing the MPU to peripherals through two 8-bit bidirectional peripheral data buses and four control lines. No external logic is required for interfacing to most peripheral devices.

The functional configuration of the PIA is programmed by the MPU during system initialization. Each of the peripheral data lines can be programmed to act as an input or output, and each of the four control/interrupt lines may be programmed for one of several control modes. This allows a high degree of flexibility in the over-all operation of the interface.

- 8-Bit Bidirectional Data Bus for Communication with the MPU
- Two Bidirectional 8-Bit Buses for Interface to Peripherals
- Two Programmable Control Registers
- Two Programmable Data Direction Registers
- Four Individually-Controlled Interrupt Input Lines; Two Usable as Peripheral Control Outputs
- Handshake Control Logic for Input and Output Peripheral Operation
- High-Impedance 3-State and Direct Transistor Drive Peripheral Lines
- Program Controlled Interrupt and Interrupt Disable Capability
- CMOS Drive Capability on Side A Peripheral Lines
- Two TTL Drive Capability on All A and B Side Buffers
- TTL-Compatible
- Static Operation

### ORDERING INFORMATION

Speed	Device	Temperature Range
1.0 MHz MIL-STD-883B MIL-STD-883C	MC6821P, L	0 to +70°C
	MC6821CP, CL	-40 to +85°C
	MC6821BQCS	-55 to +125°C
	MC6821CQCS	
1.5 MHz	MC68A21P, L	0 to +70°C
	MC68A21CP, CL	-40 to +85°C
2.0 MHz	MC68B21P, L	0 to +70°C

**MC6821**  
(1.0 MHz)

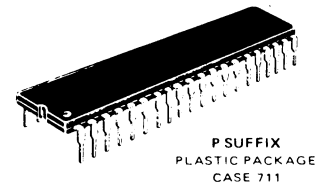
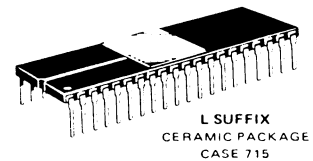
**MC68A21**  
(1.5 MHz)

**MC68B21**  
(2.0 MHz)

**MOS**

(N-CHANNEL, SILICON-GATE,  
DEPLETION LOAD)

**PERIPHERAL INTERFACE  
ADAPTER**



### PIN ASSIGNMENT

1	V <sub>SS</sub>	CA1	40
2	PA0	CA2	39
3	PA1	IRQA	38
4	PA2	IRQB	37
5	PA3	RS0	36
6	PA4	RS1	35
7	PA5	Reset	34
8	PA6	D0	33
9	PA7	D1	32
10	PB0	D2	31
11	PB1	D3	30
12	PB2	D4	29
13	PB3	D5	28
14	PB4	D6	27
15	PB5	D7	26
16	PB6	E	25
17	PB7	CS1	24
18	CB1	CS2	23
19	CB2	CS0	22
20	V <sub>CC</sub>	R/W	21

## MAXIMUM RATINGS

Rating	Symbol	Value	Unit
Supply Voltage	$V_{CC}$	-0.3 to +7.0	Vdc
Input Voltage	$V_{in}$	-0.3 to +7.0	Vdc
Operating Temperature Range MC6821, MC68A21, MC68B21 MC6821C, MC68A21C MC6821CQCS, MC6821BQCS	$T_A$	$T_L$ to $T_H$ 0 to 70 -40 to 85 -55 to 125	$^{\circ}C$ $^{\circ}C$ $^{\circ}C$
Storage Temperature Range	$T_{stg}$	-55 to +150	$^{\circ}C$
Thermal Resistance	$\theta_{JA}$	82.5	$^{\circ}C/W$

This device contains circuitry to protect the inputs against damage due to high static voltages or electric fields; however, it is advised that normal precautions be taken to avoid application of any voltage higher than maximum rated voltages to this high impedance.

ELECTRICAL CHARACTERISTICS ( $V_{CC} = 5.0\text{ V} \pm 5\%$ ,  $V_{SS} = 0$ ,  $T_A = T_L$  to  $T_H$  unless otherwise noted)

Characteristic	Symbol	Min	Typ	Max	Unit
----------------	--------	-----	-----	-----	------

BUS CONTROL INPUTS ( $R/\bar{W}$ , Enable,  $\bar{\text{Reset}}$ , RS0, RS1, CS0, CS1,  $\bar{\text{CS2}}$ )

Input High Voltage	$V_{IH}$	$V_{SS} + 2.0$	—	$V_{CC}$	Vdc
Input Low Voltage	$V_{IL}$	$V_{SS} - 0.3$	—	$V_{SS} + 0.8$	Vdc
Input Leakage Current ( $V_{in} = 0$ to 5.25 Vdc)	$I_{in}$	—	1.0	2.5	$\mu\text{Adc}$
Capacitance ( $V_{in} = 0$ , $T_A = 25^{\circ}C$ , $f = 1.0\text{ MHz}$ )	$C_{in}$	—	—	7.5	pF

INTERRUPT OUTPUTS ( $\bar{\text{IRQA}}$ ,  $\bar{\text{IRQB}}$ )

Output Low Voltage ( $I_{Load} = 3.2\text{ mAdc}$ )	$V_{OL}$	—	—	$V_{SS} + 0.4$	Vdc
Output Leakage Current (Off State) ( $V_{OH} = 2.4\text{ Vdc}$ )	$I_{LOH}$	—	1.0	10	$\mu\text{Adc}$
Capacitance ( $V_{in} = 0$ , $T_A = 25^{\circ}C$ , $f = 1.0\text{ MHz}$ )	$C_{out}$	—	—	5.0	pF

## DATA BUS (D0-D7)

Input High Voltage	$V_{IH}$	$V_{SS} + 2.0$	—	$V_{CC}$	Vdc
Input Low Voltage	$V_{IL}$	$V_{SS} - 0.3$	—	$V_{SS} + 0.8$	Vdc
Three-State (Off State) Input Current ( $V_{in} = 0.4$ to 2.4 Vdc)	$I_{TSI}$	—	2.0	10	$\mu\text{Adc}$
Output High Voltage ( $I_{Load} = -205\text{ }\mu\text{Adc}$ )	$V_{OH}$	$V_{SS} + 2.4$	—	—	Vdc
Output Low Voltage ( $I_{Load} = 1.6\text{ mAdc}$ )	$V_{OL}$	—	—	$V_{SS} + 0.4$	Vdc
Capacitance ( $V_{in} = 0$ , $T_A = 25^{\circ}C$ , $f = 1.0\text{ MHz}$ )	$C_{in}$	—	—	12.5	pF

## PERIPHERAL BUS (PA0-PA7, PB0-PB7, CA1, CA2, CB1, CB2)

Input Leakage Current $R/\bar{W}$ , $\bar{\text{Reset}}$ , RS0, RS1, CS0, CS1, $\bar{\text{CS2}}$ , CA1, CB1, Enable ( $V_{in} = 0$ to 5.25 Vdc)	$I_{in}$	—	1.0	2.5	$\mu\text{Adc}$
Three-State (Off State) Input Current ( $V_{in} = 0.4$ to 2.4 Vdc)	$I_{TSI}$	—	2.0	10	$\mu\text{Adc}$
Input High Current ( $V_{IH} = 2.4\text{ Vdc}$ )	$I_{IH}$	-200	-400	—	$\mu\text{Adc}$
Darlington Drive Current $V_O = 1.5\text{ Vdc}$	$I_{OH}$	-1.0	—	-10	mAdc
Input Low Current ( $V_{IL} = 0.4\text{ Vdc}$ )	$I_{IL}$	—	-1.3	-2.4	mAdc
Output High Voltage ( $I_{Load} = -200\text{ }\mu\text{Adc}$ ) ( $I_{Load} = -10\text{ }\mu\text{Adc}$ )	$V_{OH}$	$V_{SS} + 2.4$ $V_{CC} - 1.0$	— —	— —	Vdc
Output Low Voltage ( $I_{Load} = 3.2\text{ mAdc}$ )	$V_{OL}$	—	—	$V_{SS} + 0.4$	Vdc
Capacitance ( $V_{in} = 0$ , $T_A = 25^{\circ}C$ , $f = 1.0\text{ MHz}$ )	$C_{in}$	—	—	10	pF

## POWER REQUIREMENTS

Power Dissipation	$P_D$	—	—	550	mW
-------------------	-------	---	---	-----	----



**PERIPHERAL TIMING CHARACTERISTICS** ( $V_{CC} = 5.0 \text{ V} \pm 5\%$ ,  $V_{SS} = 0 \text{ V}$ ,  $T_A = T_L$  to  $T_H$  unless otherwise specified.)

Characteristic	Symbol	MC6821		MC68A21		MC68B21		Unit	Reference Fig. No.
		Min	Max	Min	Max	Min	Max		
Peripheral Data Setup Time	$t_{PDSU}$	200	—	135	—	100	—	ns	8
Peripheral Data Hold Time	$t_{PDH}$	0	—	0	—	0	—	ns	8
Delay Time, Enable negative transition to CA2 negative transition	$t_{CA2}$	—	1.0	—	0.670	—	0.500	$\mu\text{s}$	5, 9, 10
Delay Time, Enable negative transition to CA2 positive transition	$t_{RS1}$	—	1.0	—	0.670	—	0.500	$\mu\text{s}$	5, 9
Rise and Fall Times for CA1 and CA2 input signals	$t_r, t_f$	—	1.0	—	1.0	—	1.0	$\mu\text{s}$	5, 10
Delay Time from CA1 active transition to CA2 positive transition	$t_{RS2}$	—	2.0	—	1.35	—	1.0	$\mu\text{s}$	5, 10
Delay Time, Enable negative transition to Peripheral Data Valid	$t_{PDW}$	—	1.0	—	0.670	—	0.5	$\mu\text{s}$	5, 11, 12
Delay Time, Enable negative transition to Peripheral CMOS Data Valid PA0-PA7, CA2	$t_{CMOS}$	—	2.0	—	1.35	—	1.0	$\mu\text{s}$	6, 11
Delay Time, Enable positive transition to CB2 negative transition	$t_{CB2}$	—	1.0	—	0.670	—	0.5	$\mu\text{s}$	5, 13, 14
Delay Time, Peripheral Data Valid to CB2 negative transition	$t_{DC}$	20	—	20	—	20	—	ns	5, 12
Delay Time, Enable positive transition to CB2 positive transition	$t_{RS1}$	—	1.0	—	0.670	—	0.5	$\mu\text{s}$	5, 13
Peripheral Control Output Pulse Width, CA2/CB2	$PW_{CT}$	550	—	550	—	500	—	ns	5, 13
Rise and Fall Time for CB1 and CB2 input signals	$t_r, t_f$	—	1.0	—	1.0	—	1.0	$\mu\text{s}$	14
Delay Time, CB1 active transition to CB2 positive transition	$t_{RS2}$	—	2.0	—	1.35	—	1.0	$\mu\text{s}$	5, 14
Interrupt Release Time, $\overline{IRQA}$ and $\overline{IRQB}$	$t_{IR}$	—	1.60	—	1.10	—	0.85	$\mu\text{s}$	7, 16
Interrupt Response Time	$t_{RS3}$	—	1.0	—	1.0	—	1.0	$\mu\text{s}$	7, 15
Interrupt Input Pulse Width	$PW_I$	500	—	500	—	500	—	ns	15
Reset Low Time*	$t_{RL}$	1.0	—	0.66	—	0.5	—	$\mu\text{s}$	17

\*The Reset line must be high a minimum of 1.0  $\mu\text{s}$  before addressing the PIA.

FIGURE 5 – TTL EQUIV. TEST LOAD

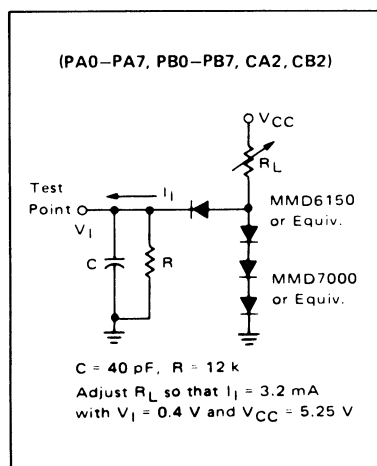


FIGURE 6 – CMOS EQUIV. TEST LOAD

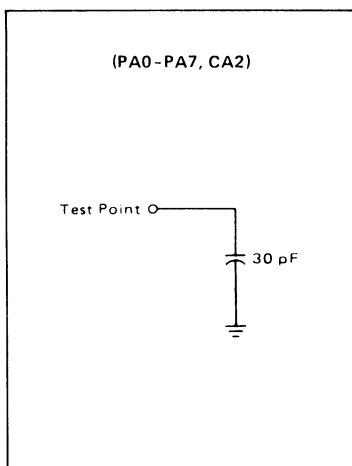
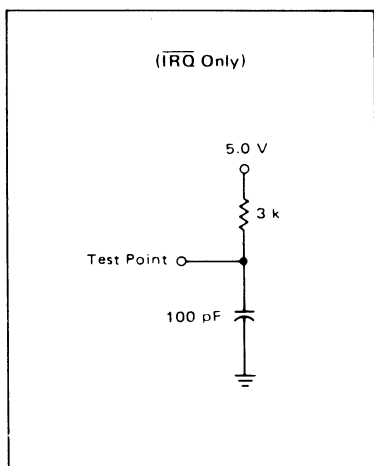


FIGURE 7 – NMOS EQUIV. TEST LOAD





**BUS TIMING CHARACTERISTICS** ( $V_{CC} = 5.0 \text{ V} \pm 5\%$ ,  $V_{SS} = 0$ ,  $T_A = T_L$  to  $T_H$  unless otherwise specified.)

Characteristic	Symbol	MC6821		MC68A21		MC68B21		Unit	Ref. Fig. No.
		Min	Max	Min	Max	Min	Max		
Enable Cycle Time	$t_{\text{cycE}}$	1000	—	666	—	500	—	ns	1
Enable Pulse Width, High	$PW_{EH}$	450	—	280	—	220	—	ns	1
Enable Pulse Width, Low	$PW_{EL}$	430	—	280	—	210	—	ns	1
Enable Pulse Rise and Fall Times	$t_{Er}, t_{Ef}$	—	25	—	25	—	25	ns	1
Setup Time, Address and R/W valid to Enable positive transition	$t_{AS}$	160	—	140	—	70	—	ns	2, 3
Address Hold Time	$t_{AH}$	10	—	10	—	10	—	ns	2, 3
Data Delay Time, Read	$t_{DDR}$	—	320	—	220	—	180	ns	2, 4
Data Hold Time, Read	$t_{DHR}$	10	—	10	—	10	—	ns	2, 4
Data Setup Time, Write	$t_{DSW}$	195	—	80	—	60	—	ns	3, 4
Data Hold Time, Write	$t_{DHW}$	10	—	10	—	10	—	ns	3, 4

FIGURE 1 — ENABLE SIGNAL CHARACTERISTICS

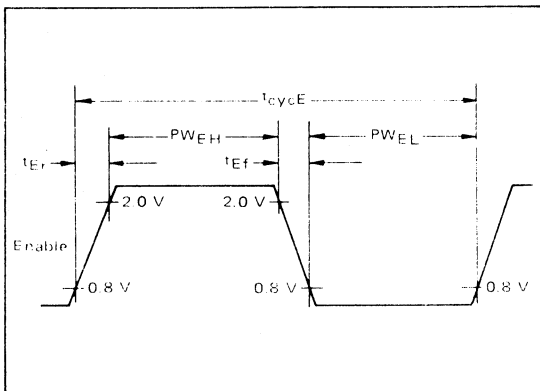
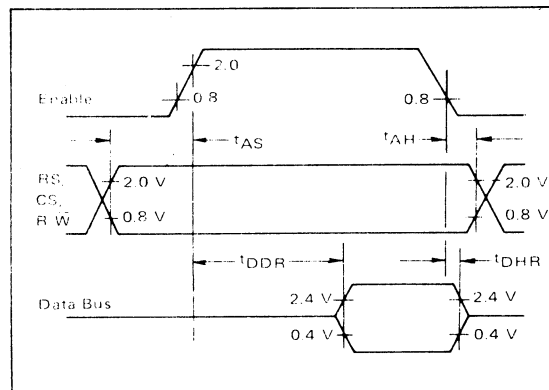
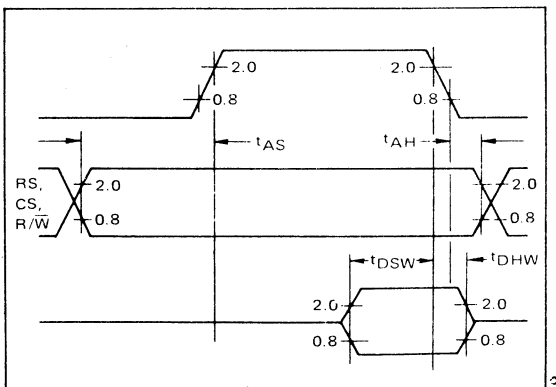
FIGURE 2 — BUS READ TIMING CHARACTERISTICS  
(Read Information from PIA)FIGURE 3 — BUS WRITE TIMING CHARACTERISTICS  
(Write Information into PIA)

FIGURE 4 — BUS TIMING TEST LOADS

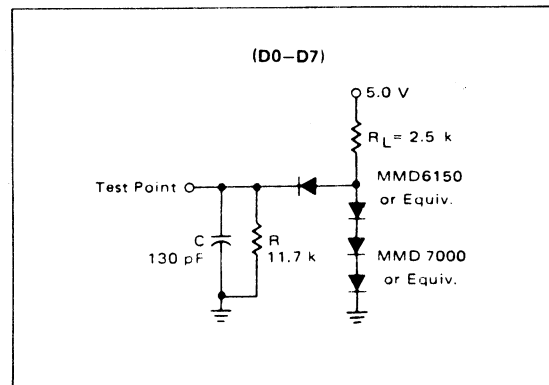
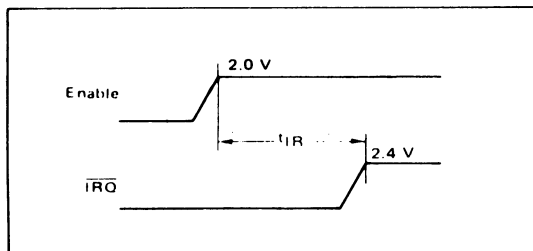
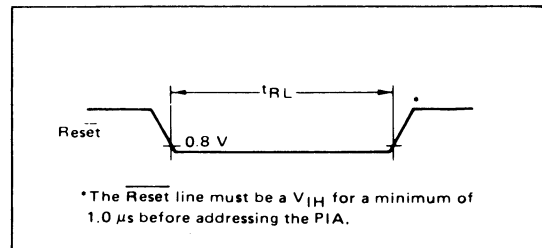
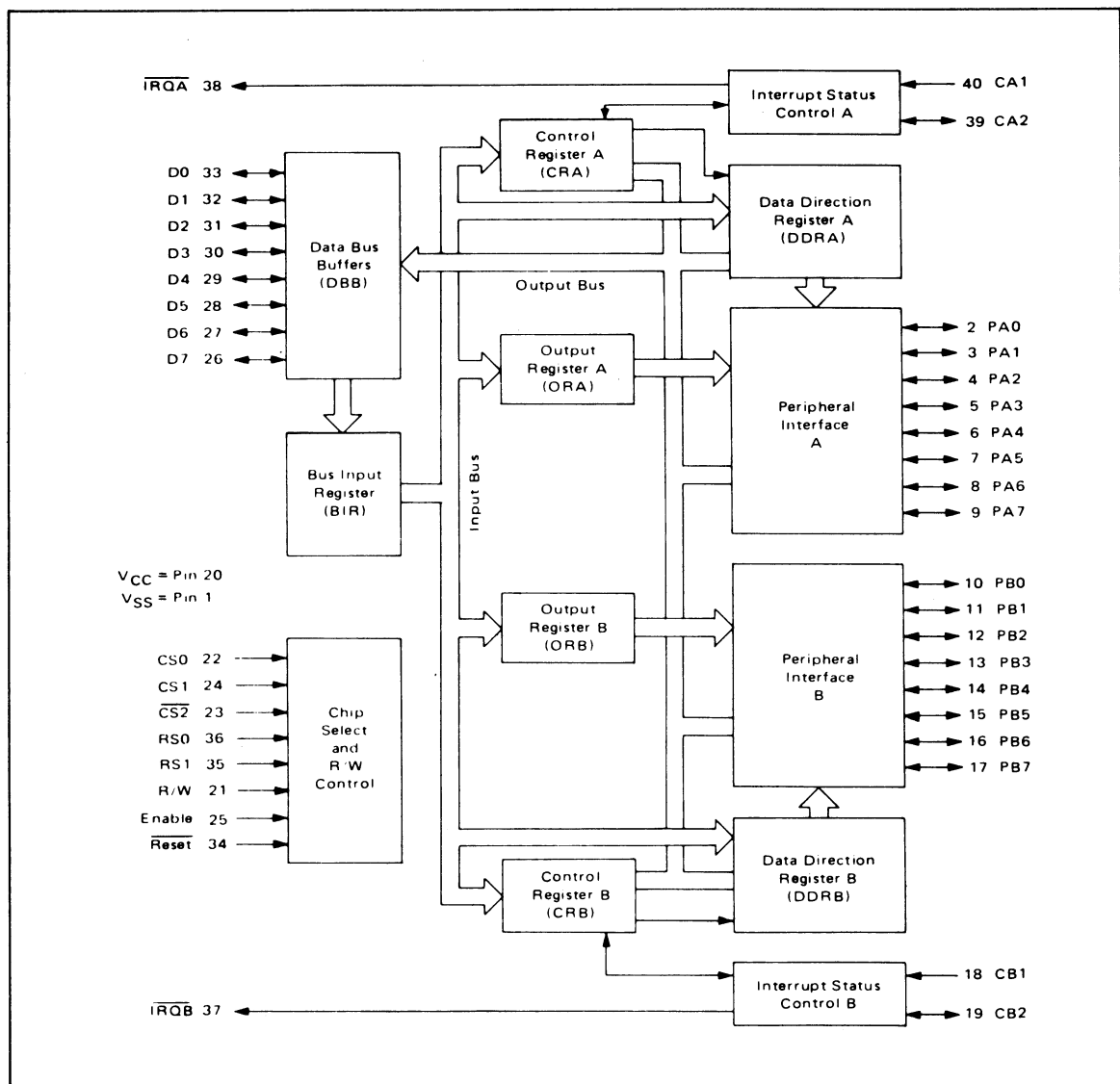


FIGURE 16 –  $\overline{\text{IRQ}}$  RELEASE TIMEFIGURE 17 –  $\overline{\text{RESET}}$  LOW TIME

## EXPANDED BLOCK DIAGRAM



## SERIEL DATA KOMMUNIKATION

Det er i et tidligere afsnit vist, at CPU'en arbejder med 8 bits i hver byte, og at byen flyttes i parallelformat, d.v.s. alle 8 bit i en given byte flyttes samtidig.

Der er tidspunkter, hvor data må transporteres via EN kommunikationsledning eller et medie, fordi dette er meget kostbart pr. enhed. Når dette er tilfældet, er det ønskeligt at nedsætte omkostningerne til transmissionsudstyret på bekostning af transmissionstiden.

Dette kan opnås ved at sende data i serieform på EN transmissionsledning fremfor i parallelform på 8 transmissionsledninger. At sende på serieform tager længere tid ved en bestemt signalhastighed (båndbredde) end afsendelse på parallelform.

Hvis signaler i forbindelse med microcomputere skal sendes over en afstand på mere end et par meter, vælges oftest seriel transmission.

### ACIA MC 6850

I 6800 familien kan problemet med DATAOMSÆTNING mellem SERIE og PARALLEL løses med hardwarekredsen MC 6850, der er en ASYNCHRONOUS COMMUNICATION INTERFACE ADAPTER (ACIA), også ofte omtalt som en UÅRT (Universal Asynchronous Receiver Transmitter). Kredsløbet omformer parallelle data på CPU-siden til serielle data på interfacesiden (Transmit) eller modtager serielle data på interfacesiden og omformer disse til parallelle data på CPU-siden (Receive). Desuden findes forskellige kontrolsignaler til kommunikationen på interfacesiden. Virkemåden af hardwaren kan fastlægges gennem programmering, d.v.s. entydig funktion for MC 6850 fastlægges gennem software.

To generelle anvendelsesformer for MC 6850 kan vises ved fig. 7.38 og 7.39 nedenfor.

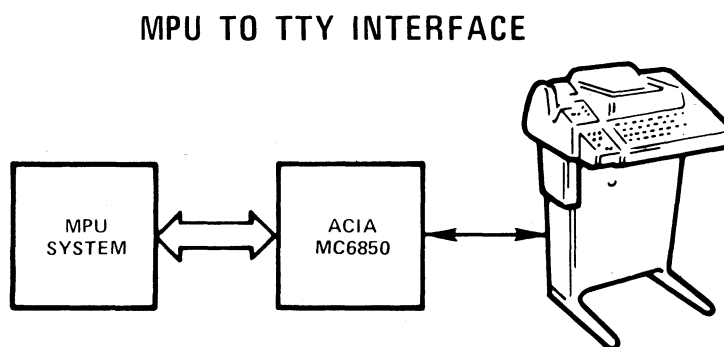


Fig. 7.38. (AA)

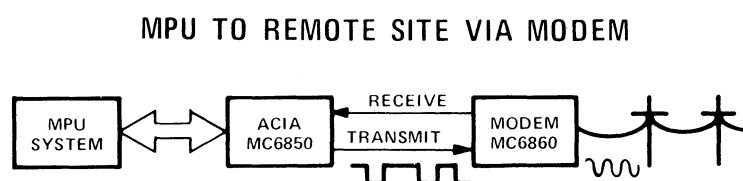


Fig. 7.39. (AA)

## ACIA BLOKDIAGRAM

For ACIA MC 6850 kan tegnes det i fig. 7.40 viste BLOKDIAGRAM med de anvendte bennavne:

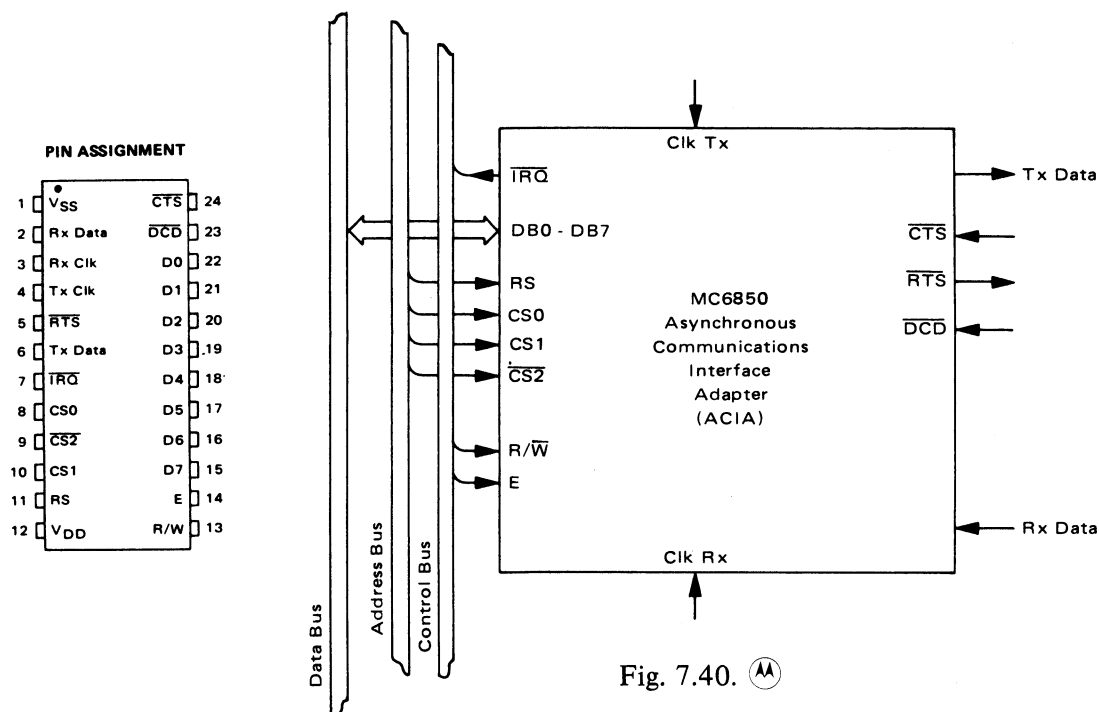


Fig. 7.40. (M)

Inde i ACIA'en findes følgende 4 registre, der er tilgængelige for CPU-programmet:

CONTROL	(RS = 0, R/ $\overline{w}$ = 0)
STATUS	(RS = 0, R/ $\overline{w}$ = 1)
TRANSMIT DATA	(RS = 1, R/ $\overline{w}$ = 0)
RECEIVE DATA	(RS = 1, R/ $\overline{w}$ = 1)

Disse kan ses af fig. 7.41.

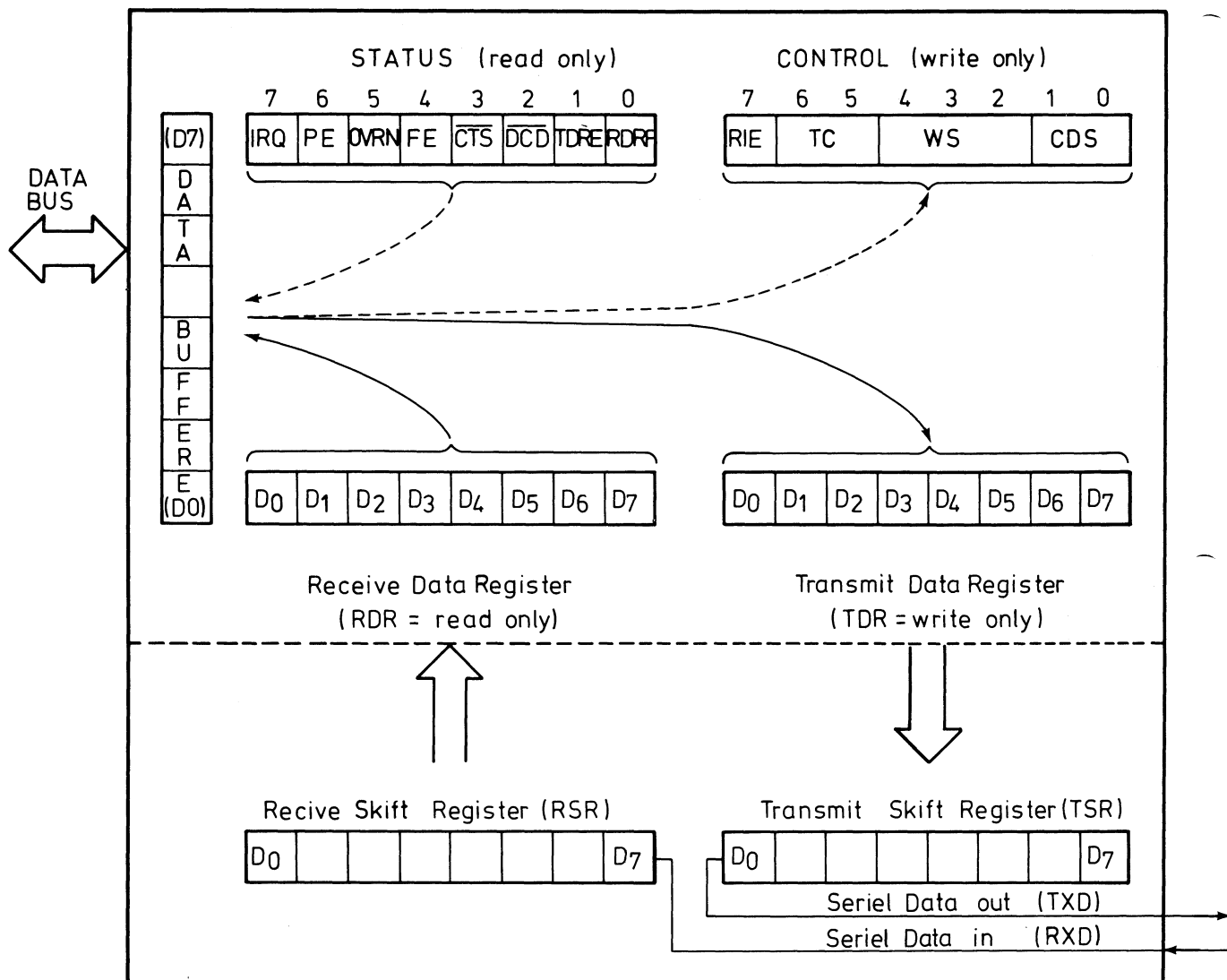


Fig. 7.41.

## ACIA ADRESSERING

De fire 8-bit registre i ACIA'en selekteres ved at føre adressebus-sens  $A_0$  til REGISTER SELECT (RS). Da der kun findes en register select ledning, opfatter CPU'en ACIA'en som liggende på to adresser i memory map.

Read/write signalet er med til at adskille ACIA'ens enkelte registre, idet CONTROL og STATUS ligger på en ADRESSE, hvilket også gælder for Transmit- og Receive-registrene.

RS	R/ $\bar{w}$	Register selected
0	0	Control Register
0	1	Status Register
1	0	Transmit Data Register
1	1	Receive Data Register

Fig. 7.42.

Det ses, at Control Register og Transmit Data Register kun kan SKRIVES i af CPU'en (WRITE ONLY). Status Register og Receive Data Register kan kun LÆSES af CPU'en (READ ONLY).

## REGISTRENES VIRKEMÅDE

### TRANSMIT DATA REGISTER (TDR)

TRANSMIT DATA REGISTER (TDR) bruges til at holde de data, der skal transmitteres næste gang. Når der skrives fra CPU'ens DATABUS ind i dette register, så CLEARES (0-stilles) TRANSMIT DATA REGISTER EMPTY (TDRE) i STATUS REGISTRET og ligeledes den tilhørende interrupt betingelse.

Karakteren (de 8 bit) i transmit data registret overføres til TRANSMIT SHIFT REGISTER (TSR) ET bits tid, før START BIT for denne karakter kan ses på TRANSMIT DATA output benet (TXD). Overførsel fra Transmit data register (TDR) til Transmit Shift register (TSR) sker på bagkanten af TRANSMIT CLOCK (TXC) 1, 16 eller 64 clockpulser (afhængig af clockdele forholdet programmeret i kontrolregistret) før START BIT for denne karakter ses på TRANSMIT OUTPUT DATA (TXD).

Hvis CPU'en skriver i transmit data registret (TDR), medens den foregående karakter er ved at blive udsendt fra transmit shift registret, så sker overførsel fra TDR til TSR på begyndelsen af det sidste stop bit i den netop udsendte karakter. Dette gøres for at opnå maximal transmissionshastighed. I andre tilfælde sker overførslen på den positive gående kant af den næste interne clock cycle, som er et nøjagtigt antal gange af senderens bit-tid siden det sidste MASTER RESET, som beskrevet nedenfor i afsnittet om CLOCK DELER FORHOLDET (kontrol reg. bit 0 og 1).

Efter overførslen fra (TDR) til (TSR), der sker på den faldende kant at den interne SENDER CLOCK PULS (der repræsenterer centret på det transmitterede databit), eller omkring midten af STOP BIT på den foregående karakter, går TDRE bit i STATUS REGISTRET på HIGH og forårsager et INTERRUPT ( $\overline{IRQ}$ ), hvis dette er ENABLED. Dette interrupt fjernes ved afslutningen af den WRITE CYCLE, som fra CPU'en skriver en ny karakter ind i TRANSMIT DATA REGISTER (TDR).

Hvis TDRE bit i status registret IKKE er sat på HIGH (d.v.s. den foregående karakter endnu ikke er blevet overført til TRANSMIT SHIFT REGISTER), når der skrives en ny karakter ind i TDR, vil registrets tidligere indhold gå tabt. Hvis det sker under overførsel til TSR, kan transmissionsforløbet blive uforudsigeligt. TDRE bit i STATUS REG. bliver altid sat på HIGH efter overførsel til (TSR). Derfor bør man vente på dette STATUS BIT, før der skrives en karakter ind i (TDR), for at undgå tvetydigheder.

Når ACIA'en er programmeret til 7 DATABIT, bliver MSB i Transmit Data Registret „overset” (udeladt).

Data transmitteres i SERIE med mindst betydende bit (LSB) først.

### RECEIVE DATA REGISTER (RDR)

RECEIVE DATA REGISTER (RDR) benyttes til at holde på datakarakteren efter, at den er modtaget. Der kræves en READING fra dette register til DATABUS for at kunne RESETTE nogen modtager interrupt tilstand, og RESET af RECEIVER DATA

REGISTER FULL (RDRF) bit i STATUS REGISTRET. RDR påvirkes kun af overførsel af en ny modtaget karakter, når RDRF i status er LOW. Overførslen sker i den sidste halvdel af det første STOP BIT i den modtagne karakter.

Når ACIA'en er programmeret til 7 databit, bliver MSB i RDR sat til NUL.

Data MODTAGES i SERIE med LSB først.

## CONTROL REGISTER I MC 6850

ACIA'ens kontrolregister er opdelt i 4 afsnit som vist i fig. 7.43.

ACIA Control Register Format

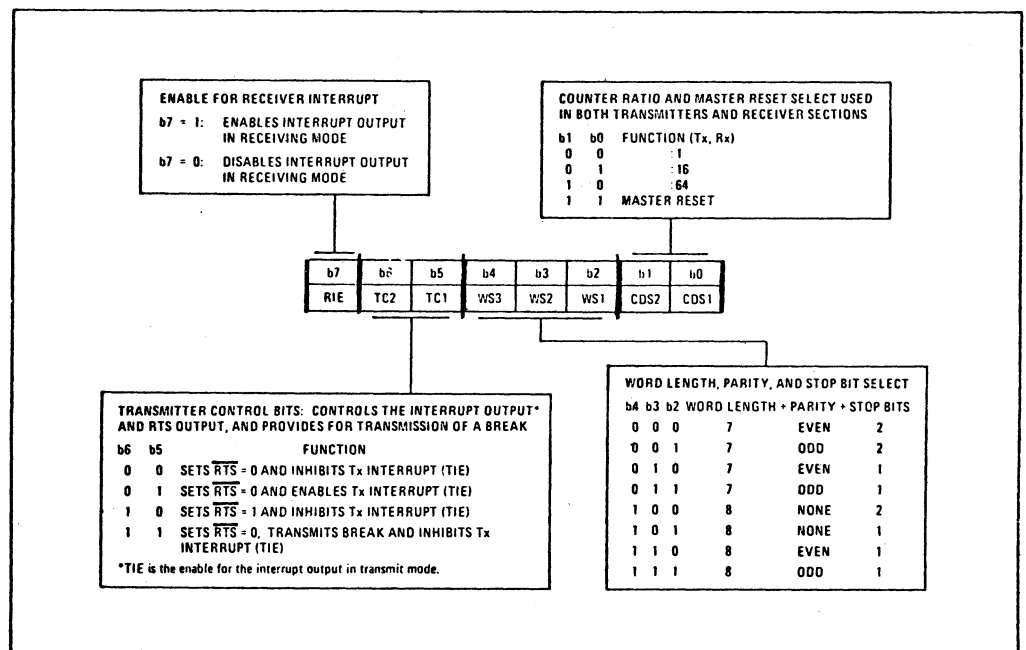


Fig. 7.43. (M)

Når der fra databus skrives ind i dette register, påvirkes kontrolfunktionerne for både TRANSMIT- og RECEIVE-SIDEN af ACIA'en.

## POWER ON RESET

Når forsyningsspændingen tilsluttes ACIA'en, initialiseres KONTROL-REGISTRET automatisk ved en intern POWER-ON-RESET. Registeret kommer til at indeholde: 010X XXXX. Herved er RECEIVER INTERRUPT DISABLED, REQUEST TO SEND er OFF (HIGH) og alle interne registre og latche er RESET.

## MASTER RESET

Den interne power-on-reset skal ophæves ved en MASTER RESET, før ACIA'en kan benyttes til kommunikation. MASTER RESET udføres ved at skrive XXXX XX11 til CONTROL REGISTRET.

Når der skrives ned i CONTROL REGISTRET, har dette en umiddelbar virkning. Nogle af controlregistrets bit har lille eller slet ingen indflydelse på virkningen af TRANSMIT- eller RECEIVE-siden i ACIA'en, andre bit kan øjeblikkeligt ændre virkemåden af den ene eller den anden side af ACIA'en og i visse tilfælde med ubehagelige følger for data, der er under modtagelse.

### BIT 0 og 1 (CDS)

Bit 0 og 1 i controlregistret definerer TÆLLERFORHOLDET (counter ratio) og MASTER RESET tilstanden. Når begge bit er 1-taller, er det defineret som MASTER RESET, som resetter alle interne registre og latche. Transmitterens clock-deler bliver sat på 0 og i Transmit Shift Register (TSR) sættes alle bit til 1 (svarende til MARK-tilstanden), selve senderen disables, alle interrupt betingelser slettes og modtager logikken indstilles til tomgangstilstanden, d.v.s. klar til at modtage START BIT. Master reset fjerner også power-on-reset-tilstanden.

Når kontrolregistrets bit 0 og 1 indeholder anden kombination end 11, definerer de clock-deler-forholdet for SENDER- og MODTAGER CLOCK. Der findes et ydre tilført clocksignal, som deles med det indstillede forhold, der kan være:

Counter ratio and Master reset select used in both transmitters and receiver sections		
b1	b0	Function (Tx,Rx)
0	0	1
0	1	16
1	0	64
1	1	MASTER RESET

Ved delerforholdet 1 antages det, at den ydre clock er SYNKRON med DATA.

Ved normal initialisering vil CPU'en først skrive to 1-taller i bit 0 og 1 for at give MASTER RESET, og derefter skrive det bitmønster, der passer til det valgte delerforhold af clock. Master Reset synkroniserer transmitter clock deleren på en sådan måde, at alle skift i sendersignalet databit vil ligge et lige antal gange delerforholdet fra AFSLUTNINGEN af Master Reset. Det vil sige et forhold på 16 (eller 64) perioder af den externe transmit-clock fra afslutningen af master reset.

Forholdet mellem extern clock og intern clock med 16 som delerforhold vises i fig. 7.44.

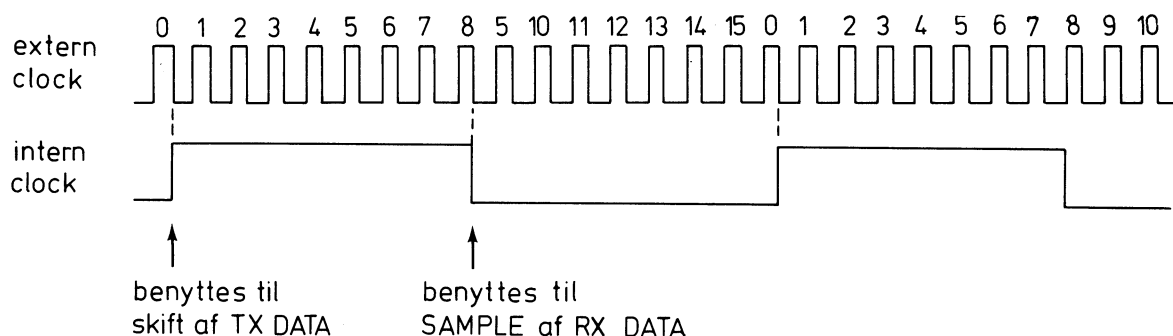


Fig. 7.44.

Intern clock i TX-siden er synkroniseret i forhold til OPHØR af MASTER RESET.



Intern clock i RX-siden holdes OFF, indtil der modtages et START BIT, hvor så intern clockforkant startes samtidig med start bits begyndelse.

Forholdet mellem DATA KARAKTER og INTERN CLOCK kan ses af fig. 745.

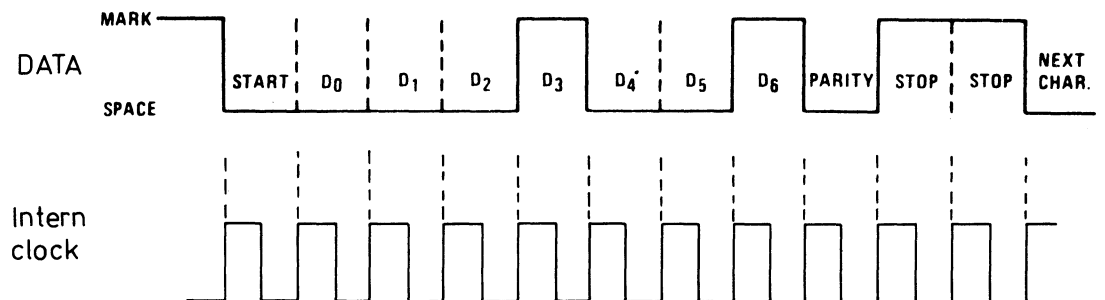


Fig. 7.45.

Ved start af modtagelse benyttes eksterne clockforkanter i den første halvperiode af den interne clock til hele tiden at undersøge, om START BIT = 0. Skulle der blive detekteret et 1-tal i denne første halvperiode, vil modtager INTERN clock blive „nulstillet”, da undersøgelsen viser „FALSK STARTBIT” på grund af 1-tallet, der blev detekteret.

#### BIT 2, 3 og 4 (WS)

Bit 2, 3 og 4 i kontrolregistret benævnes Word Select (WS) og benyttes til at fastlægge:

bit 4 ⇒ ANTAL BIT I ORDET [7 eller 8]  
 bit 3 ⇒ ANTAL STOP BIT [1 eller 2]  
 bit 2 ⇒ PARITETSFORMEN [lige/ulige/ingen]

Dette gælder for både TX og RX DATA.

Word Length, Parity, and Stop Bit Select					
b4	b3	b2	Word Length + Parity + Stop Bits		
0	0	0	7	Even	2
0	0	1	7	Odd	2
0	1	0	7	Even	1
0	1	1	7	Odd	1
1	0	0	8	None	2
1	0	1	8	None	1
1	1	0	8	Even	1
1	1	1	8	Odd	1

Fig. 7.46. AA

Ved LIGE paritet (EVEN PARITY) skal hele ordet incl. parity bit være et LIGE ANTAL 1-taller, og parity bit indstiller sig automatisk ved TRANSMISSION (TX), så dette opfyldes.

Ved modtagelsen (RX) af en karakter undersøges pariteten. Stemmer den IKKE overens med det i KONTROL REGISTRET programmerede, vil PARITY ERROR i STATUS REG. gå HIGH.

I modtageren undersøges pariteten i det øjeblik, karakteren overføres til RECEIVE DATA REGISTER (RDR).

I senderen genereres paritetsbit under udsendelsen af karakteren, og det skabes, lige før det skal udsendes.

For fig. 7.45 DATA gælder:

**7 BIT ASCII CHAR. "H"**  
**EVEN PARITY — 2 STOP BITS**  
**H = 48<sub>16</sub> = 1001000<sub>2</sub>**

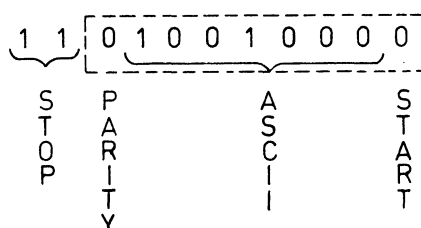


Fig. 7.47.

Der kan overføres 7 eller 8 DATA bit i hver karakter bestemt af bit 4 i kontrolregistret (se fig. 7.46).

Antal af STOP bit indstilles med bit 3 i kontrolregistret.

Der må ikke ændres på indholdet af bit 4, 3 og 2, når der foretages TX eller RX af data, idet dette kan forårsage fejl i dataindholdet.

## BIT 5 og 6 (TC)

Bit 5 og 6 i kontrolregistret benyttes til TX kontrol (Transmit Control).

Transmitter Control Bits: Controls the Interrupt Output* and RTS Output, and provides for Transmission of a Break		
b6	b5	Function
0	0	Sets $\overline{\text{RTS}} = 0$ and inhibits Tx interrupt (TIE)
0	1	Sets $\overline{\text{RTS}} = 0$ and enables Tx interrupt (TIE)
1	0	Sets $\overline{\text{RTS}} = 1$ and inhibits Tx interrupt (TIE)
1	1	Sets $\overline{\text{RTS}} = 0$ , Transmits Break and inhibits Tx interrupt (TIE)

\*TIE is the enable for the interrupt output in transmit mode.

Fig. 7.48. Ⓐ

Request to Send ( $\overline{\text{RTS}}$ ) er ENABLED i alle kombinationer PÅ NÆR 10, hvor der kommer HIGH på  $\overline{\text{RTS}}$  og „lukker” dette output fra ACIA'en.

Transmit Interrupt Enable (TIE) er kun ENABLED, når kombinationen 01 findes i bit 6 og 5.

Hvis de to bit begge er HIGH (11), transmitteres BREAK fra ACIA'en. Dette signal er udelukkende SPACE el. LOGISK 0, der udsendes som TX-DATA. Programmeres kontrolregistret til BREAK, vil dette være dominerende selv over master-reset.

### BIT 7 (RIE)

Bit 7 i kontrolregistret anvendes i forbindelse med RX interrupt. (RIE = Receive Interrupt Enable). Når bit 7 = 1, etableres der interrupt ENABLE for:

RECEIVE DATA REG. FULL	(STATUS bit 0 = RDR)
DATA CARRIER DETECT LOSS	(STATUS bit 2 = $\overline{\text{DCD}}$ )
OVERRUN	(STATUS bit 5 = OVRN)

Når bit 7 = 0, vil interrupt være DISABLED og ikke kunne påvirke STATUS REGISTRET eller  $\overline{\text{IRQ}}$  til CPU'en.

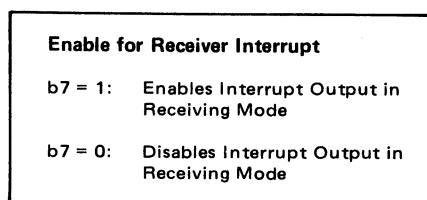


Fig. 7.49. (AA)

### STATUS REGISTER I MC 6850

De individuelle bit i STATUS REGISTRET husker de forskellige tilstande, der hører til virkningen af ACIA'en.

Nogle af STATUS BIT'ene kan føre til INTERRUPT af CPU'en. Andre er kun til informationsbrug.

Nogle af bittene har forbindelse med fejltilstande, der opstår i karakter-data og disse status bit reagerer ved overførsel af de enkelte karakterer. Andre status bit reagerer på tilstande, der er uafhængige af overførsel af karakterer. Hver enkelt status bit har sin egen funktion, det overvåger.

### BIT 0 (RDRF)

Bit 0 i status registret er flaget for RECEIVER DATA REGISTER FULL (RDRF). Et 1-tal i bit 0 markerer, at der er modtaget en karakter, som er overført fra (RSR) til (RDR), men endnu ikke ført videre til CPU'ens databus (se evt. fig. 7.41).

Bit 0 bliver 1, når overførslen fra (RSR) til (RDR) foregår. Bit 0 sættes igen til 0, når (RDR) læses ud på CPU'ens databus.

Ved OVERRUN (STATUS bit 5 = 1) sættes bit 0 = 1, selvom (RDR) ikke indeholder den nye karakter (der er tabt ved overrun), men stadig indeholder den tidligere karakter i (RDR).

Tab af RECEIVER CARRIER (kan ses som  $\overline{\text{DCD}}$  input til ACIA = HIGH) vil RESETTE ACIA'ens modtagerdel og hermed også sætte bit 0 = 0.

Hvis bit 7 = 1 i KONTROLREGISTRET (RIE = 1), vil et 1-tal i STATUS REG. bit 0 medføre,  $\overline{\text{IRQ}}$  går LOW og afgiver interrupt til CPU'en. Bit 7 i STATUS REG. går på HIGH ved  $\overline{\text{IRQ}}$  og benyttes som identifikationsbit ved en polling foretaget af CPU'en.

## ACIA Status Register Format

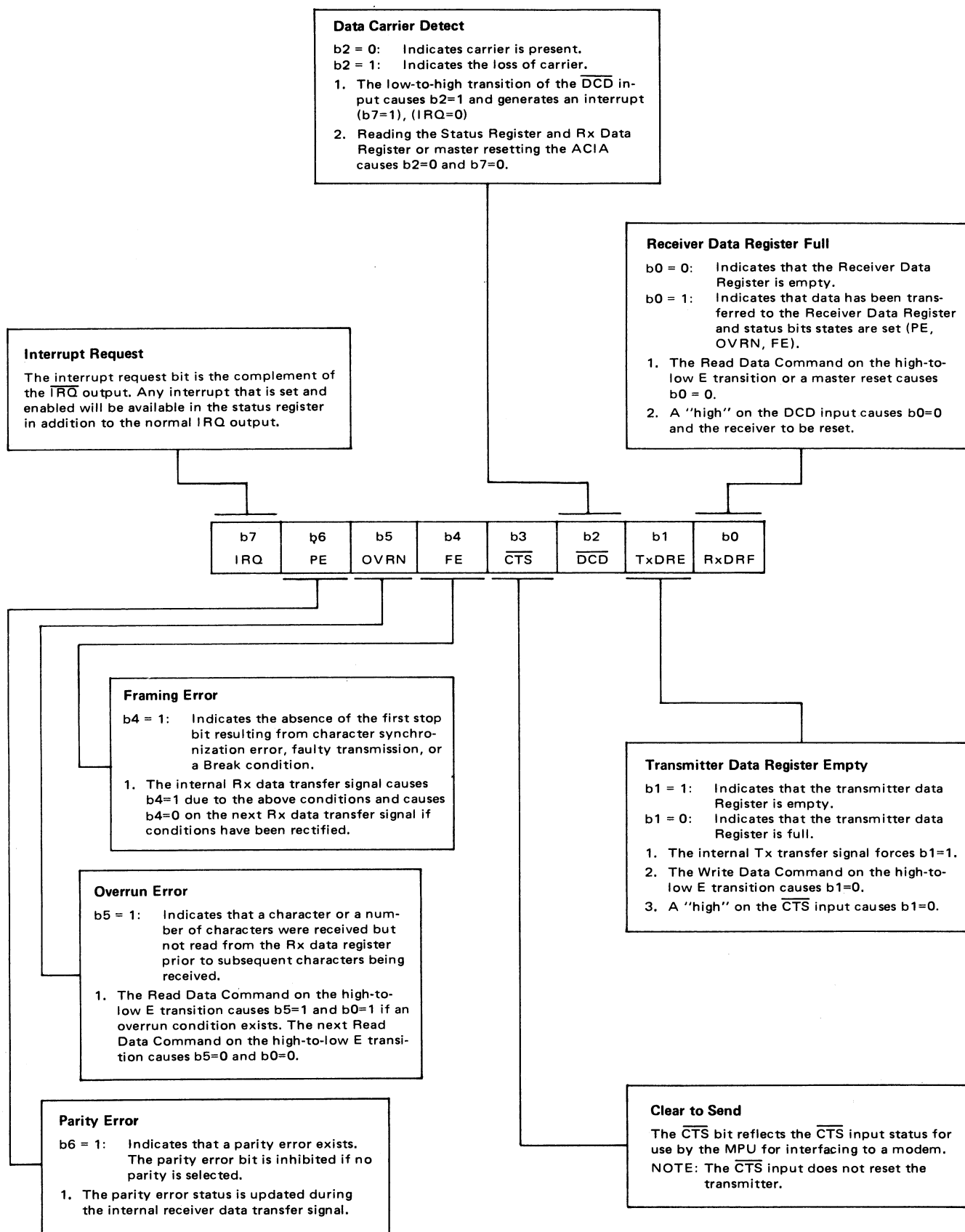


Fig. 7.50. AA

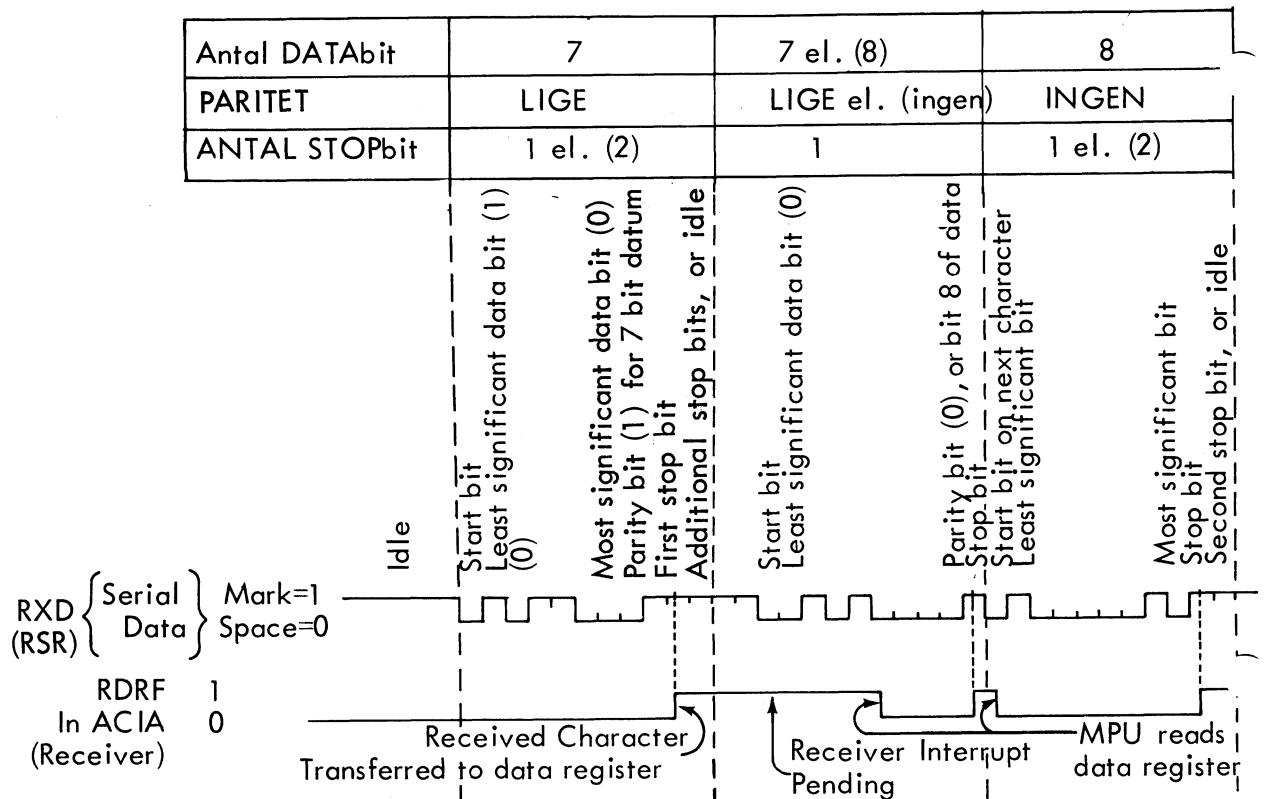


Fig. 7.51.

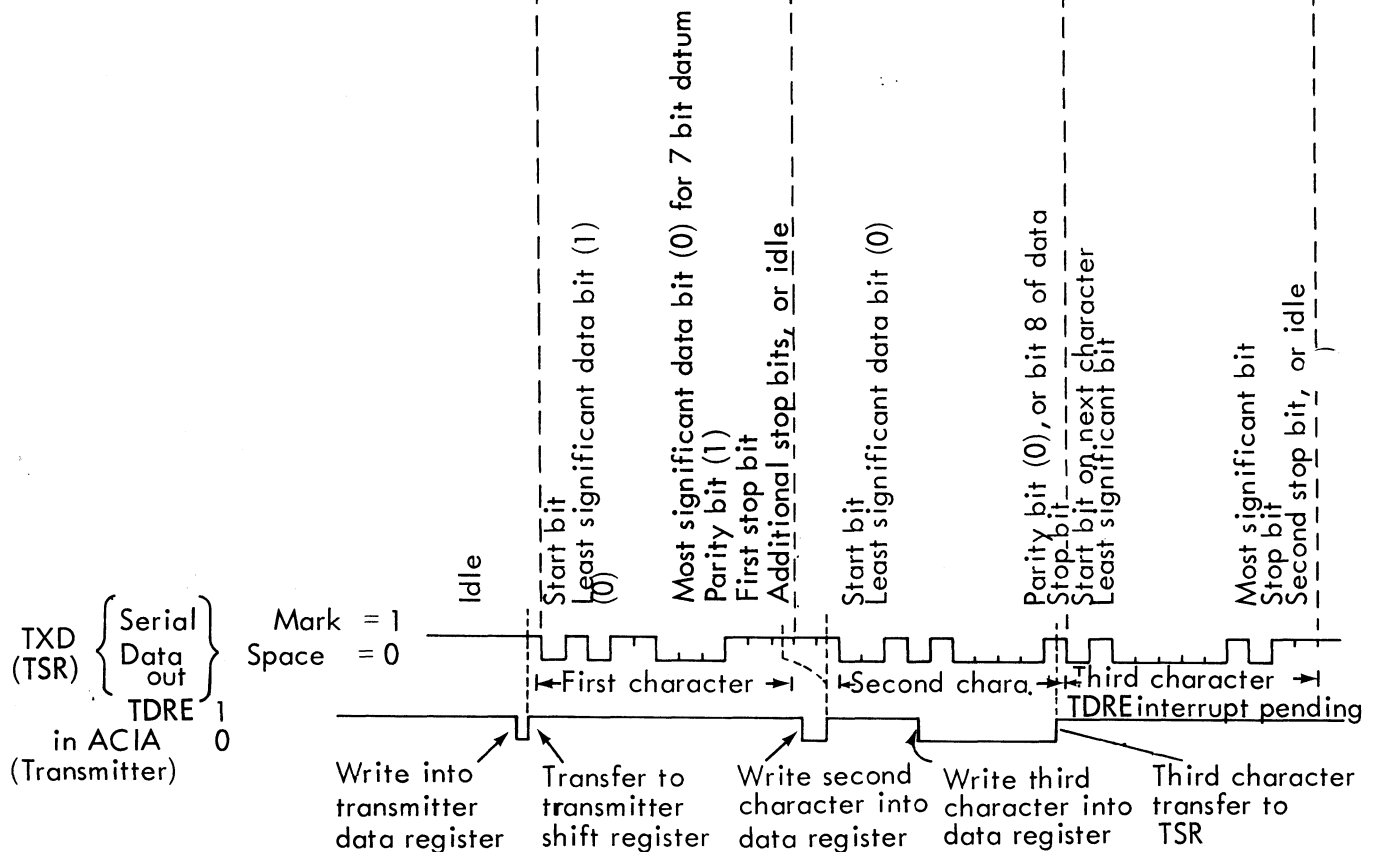


Fig. 7.52.

**BIT 1 (TDRE)**

Bit 1 i Status registret er flaget for TRANSMITTER DATA REGISTER EMPTY (TDRE). Et 1-tal i bit 1 markerer, at (TDR) er tomt og kan modtage den næste karakter, der skal afsendes.

Et 0 i bit 1 indikerer, at den sidst indskrevne karakter i (TDR) endnu ikke er overført til (TSR), eller at bit 3 i status registret (Clear to Send =  $\overline{CTS}$  = input til ACIA fra modem) er gået HIGH.

Når kontrolregistrets bit 6 og 5 bliver 01, ENABLES TRANSMITTER INTERRUPT (TIE) og så vil et 1-tal i TDRE forårsage et  $\overline{IRQ}$  til CPU'en.

Statusregistrets bit 1 (TDRE) går på 1 ved afslutningen af overførslen fra (TDR) til (TSR), hvilket kan ske, når halvdelen af det sidste stop bit i foregående karakter er transmitteret fra (TSR). Bit 1 går på 0 umiddelbart efter, at der er skrevet en ny karakter ind i (TDR).

Hvis bit 1 (TDRE) tvinges på 0, fordi bit 3 ( $\overline{CTS}$ ) går på 1, så vil bit 1 vende tilbage til 1, når bit 3 igen går på 0.

#### **BIT 2 ( $\overline{DCD}$ )**

Bit 2 i statusregistret viser tab af RECEIVER CARRIER fra modem. Skiftet fra LOW til HIGH på DATA CARRIER DETECT ( $\overline{DCD}$ ) input på ACIA'en sætter den interne flip flop for tabt carrier. Bit 2 = 1 i status registret viser tilstanden, at carrier flip-flop er 1 eller  $\overline{DCD}$  input er HIGH.

Når tabt-carrier-flip-flop går på 1, sker der RESET af modtager-siden (RX) i ACIA'en (se MASTER RESET bit 1 og 0 i CONTROL REGISTER). Dette 0-stiller også (RDRF) bit 1 i status registret, selvom der stadig kan være gyldige data i (RDR). Transmitter-siden (TX) i ACIA'en berøres IKKE af carrier-tab.

Tabt-carrier-flip-floppen kan 0-stilles (RESETTES) enten af et MASTER RESET eller ved at LÆSE STATUS REGISTER og derefter LÆSE RECEIVE DATA REG. (RDR) i den angivne rækkefølge.

Bit 2 i status registret vil forblive 1, indtil  $\overline{DCD}$  signalet går på 0.

#### **BIT 3 ( $\overline{CTS}$ )**

Bit 3 i status registret viser tilstanden på input signalet CLEAR TO SEND ( $\overline{CTS}$ ).

Et LOW input på  $\overline{CTS}$  viser sig som et 0 i bit 3 i status.

Når  $\overline{CTS}$  går HIGH, forhindrer det bit 1 i status (TDRE) i at gå på 1 og herved bliver TRANSMITTER INTERRUPT forhindret  $\Rightarrow$  ingen nye karakterer til (TDR). Dette har imidlertid ikke nogen virkning på den øvrige del af ACIA'en, og data i (TSR) vil fortsætte med at give output og ved afslutning af udsendelse vil (TDR), hvis det er fyldt op, blive overført til (TSR) og udsendt.

Derfor vil der ved normal virkemåde foregå udsendelse af EN eller TO KARAKTERER på TX-data, efter at  $\overline{CTS}$  er gået HIGH.

#### **BIT 4 (FE)**

Bit 4 i status registret benævnes FRAMING ERROR (FE) og viser, at den karakter, der lige nu befinder sig i (RDR) ikke blev fulgt af et STOP BIT på den korrekte måde. Dette kan ske, hvis karakter-synkroniseringen er ukorrekt eller hvis modtageledningen er i BREAK tilstanden. Hvis der modtages BREAK-signal, vil modtageren opfatte disse 0'er som start bit til en ny karakter.

Hvis BREAK tilstanden ophører i løbet af karaktertiden, vil ACIA'en opfatte det, som om der kom karakterer svarende til det modtagne bitmønster. Hvis BREAK fortsætter, vil hver karaktetid blive modtaget som en NUL-KARAKTER og i LIGE PARITET med en FRAMING ERROR (FE) og vil blive tolket tilsvarende, hvor det drejer sig om andre status-tilstande, f.eks. hvis parity er defineret ulige, vil parity error status bit 6 gå på 1, o.s.v.

For modtagne karakterer kræves der kun lidt over 1/2 stop bit for at forhindre FRAMING ERROR detekteret, idet alt over 1/2 stop bit af ACIA'en opfattes som en leder i tomgang og herved bliver „overset”.

#### BIT 5 (OVRN)

Bit 5 i status registret benævnes OVERRUN (OVRN) og viser, at (RDR) ikke indeholder den sidst modtagne karakter, men at en eller flere karakterer blev tabt på grund af manglende aflæsning af den tidligere datakarakter, før den næste karakter ankom. Status sættes = 1 efter den foregående „gode karakter” er blevet læst, så det genspejler den nuværende status af RDR.

Overrun (OVRN) tilstanden sætter RDRF-status til 1, hvilket forårsager en ny receiver data interrupt, selv om karakteren i RDR ikke har ændret sig.

I den normale virkemåde, vil læsningen af den foregående gode karakter i RDR ikke forårsage sletning af interrupt tilstanden, og der kræves en aflæsning nr. 2 fra RDR. Hvis RDR kun læses ud en gang ved betjeningen af RECEIVER INTERRUPT, vil interrupt stadig gælde, når CPU'en afslutter med en RTI instruktion. Dette vil forårsage en ny interrupt service sekvens.

Bit 5 i status registret sættes til 0, når der på korrekt måde overføres en gyldig karakter til RDR, eller når modtageren bliver RESET med MASTER RESET, eller ved tab af CARRIER (se status bit nr. 2).

#### BIT 6 (PE)

Bit 6 i status registret benævnes PARITY ERROR (PE) og viser, om den karakter der nu befinder sig i receive data register (RDR) blev modtaget med FØRKERT paritet. PE-bit er 0, når der ikke er specificeret paritetscheck for den modtagne karakter. BIT'en sættes = 1 eller = 0, når ny karakter transporteres til receive data register (RDR). PE bliver også cleared ved MASTER RESET eller ved tab af CARRIER (se status bit 2).

#### BIT 7 (IRQ)

Bit 7 i status registret er i overensstemmelse med output tilstanden på ACIA'ens interrupt request ( $\overline{\text{IRQ}}$ ). Hvis ACIA'en har et interrupt etableret, bliver  $\overline{\text{IRQ}}$ -output signalet trukket på LOW, hvorved bit 7 i status går på 1. Når årsagen til interrupt fjernes ved at læse eller skrive i det „nødvendige” register eller ved at DISABLE INTERRUPT, så går bit 7 tilbage til 0 og  $\overline{\text{IRQ}}$  signalledningen til Høj-impedans tilstand. Disabling af interrupt har ingen indvirkning på den indre interrupt tilstand, og hvis der senere ENABLES uden at fjerne den tilhørende interrupt betingelse (korrekt læsning eller skrivning i registre), så vil bit 7 gå til 1 og  $\overline{\text{IRQ}}$ -output aktiveret igen.

Dette bit benyttes i en INTERRUPT POLLING ROUTINE til at bestemme, hvem der har givet  $\overline{\text{IRQ}}$  til CPU'en.

## ACIA'ens SIGNALER TIL OG FRA DEN YDRE VERDEN (INTERFACE SIGNALS)

ACIA'en er konstrueret til at være tilsluttet et standard MODEM eller andet udstyr baseret på seriel datatransmission. KONTROL, DATA og CLOCK for dette interface beskrives herunder.

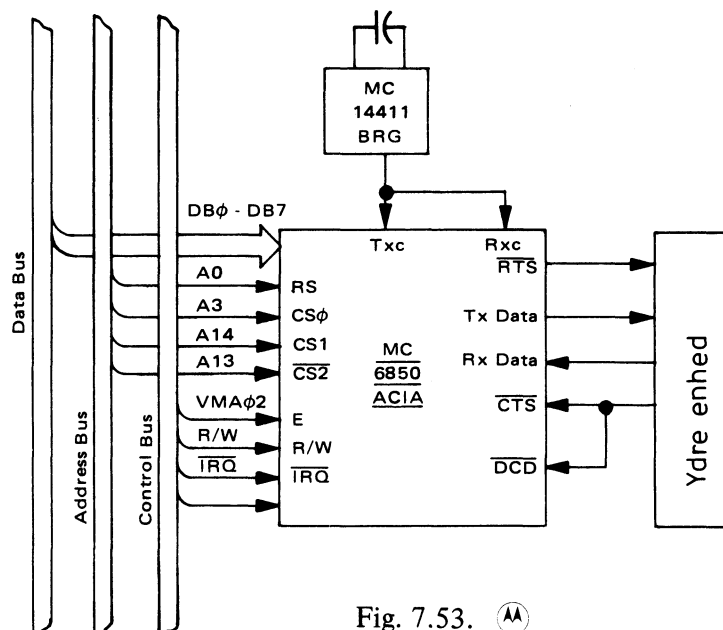



Fig. 7.53. 

### TRANSMIT CLOCK

Transmit clock (TX clock) er den tidsreference, der benyttes ved Data transmit (TXD) fra ACIA til ydre enhed.

Den negative kant på clock er den aktive ved TX.

Datahastigheden (Baudrate) kan programmeres til at være 1:1, 1:16 eller 1:64 af clockhastigheden. F.eks. kan baudrate = 4800:16 = 300 findes i KIT 2, hvor ACIA TX clock = 4800 Hz.

### RECEIVER CLOCK

Receiver clock (RX clock) benyttes til synkronisering af data, der modtages. Hvis der anvendes 1:1 mellem Data og Clock, skal synkronisering af RX data foretages EXTERNT.

Modtageren læser indkomne data på den positive kant af clock.

Clockfrekvens på 1, 16 eller 64 gange modtagersignalet baudrate kan indstilles i kontrolregistrets bit 0 og 1.

### RECEIVE DATA

Receive Data (RXD) ledningen er der, hvor SERIE DATA MODTAGES i ACIA'en. Synkronisering med en clock for at opnå datadektering udføres internt, når der benyttes clock på 16 eller 64 gange baudrate (datahastigheden). Der kan benyttes datahastigheder op til  $500 \cdot 10^3$  baud ved EXTERN synkronisering og op til  $50 \cdot 10^3$  baud ved intern ACIA clock = 1:16 af tilført clock (RX clock).

### TRANSMIT DATA

De modtagne data formes til ASYNKRONE karakterer, der består af et LOW START BIT, 7 eller 8 DATA BIT, et VALGFRIT PARITY BIT (bestemt af control registrets bit 3 og 4) og 1 eller 2 STOP BIT. Databit nr. 0 (LSB) er det først udsendte (og modtagne) efter start bit.



**CLEAR TO SEND ( $\overline{\text{CTS}}$ )**

Clear to send ( $\overline{\text{CTS}}$ ) input til ACIA'en benyttes til at kontrollere senderenden i en kommunikations-sløjfe. Et modem har et  $\overline{\text{CTS}}$  output, der er aktivt LOW. Hvis  $\overline{\text{CTS}}$  går på HIGH, vil dette få statusregistrets bit 1 (TDRE) til at gå på 0.

**REQUEST TO SEND ( $\overline{\text{RTS}}$ )**

Request to send ( $\overline{\text{RTS}}$ ) er et ACIA output signal, som kan benyttes til at kontrollere ydre kredsløb som f.eks. modems. Kontrolregistrets bit 5 og 6 i ACIA bestemmer niveau på  $\overline{\text{RTS}}$ .

**DATA CARRIER DETECTED (DCD)**

Data carrier detected ( $\overline{\text{DCD}}$ ) er et ACIA input signal, der benyttes ved modtagning i ACIA.  $\overline{\text{DCD}}$  signalet INHIBITS og INITIALISERER modtagerdelen af ACIA'en, når det er HIGH. Et skift fra LOW til HIGH på ( $\overline{\text{DCD}}$ ) afgiver et interrupt til CPU'en, hvis (RIE) receive interrupt enable er sat HIGH, for at markere tilstanden ved tab af carrier.

**PROGRAMMERING AF ACIA**

Det antages, at ACIA'en ligger på ADRESSE: 8010 (contr./status) og 8011 (TX/RX).

Der ønskes programmeret til : 64, 8 databit, ulige paritet og 1 stop bit. Der skal arbejdes mod en ydre enhed af MODEM-typen og ACIA skal være klar til at SENDE og MODTAGE (TRANSMIT og RECEIVE).

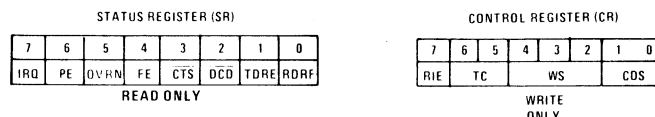


Fig. 7.54. Ⓐ

```

ACIA IN LDA A    #$03    %0000 0011    MASTER RESET
        STA A    $ 8010 (CONTROL REG. WRITE)
        LDA A    #$1E    %0001 1110    ÷64, 8 bit, O.P., 1 S
        STA A    $ 8010 (CONTROL REG. WRITE)
CHECK   LDA A    $ 8010 (READ STATUS REG.)
        AND A    #$0C    %0000 1100    (se på  $\overline{\text{CTS}}$  og  $\overline{\text{DCD}}$ )
        BNE     CHECK
        LDA A    #$BE    0/0 1011 1110 (RIE og TIE enable;  $\overline{\text{RTS}} = 0$ )
        STA A    $ 8010 (CONTROL REG. WRITE)
        JSR     OUTCH (output ACIA karakter)
        JSR     INCHR (input ACIA karakter)
        ⋮

```

**\*OUTPUT ACIA KARAKTER:**

```

OUTCH   PSH B
OUTC 1  LDA B    ACIA S          SAVE B
        ASR B                                IS DATA READY YET?
        ASR B                                (TDRE → carry)
        BCC     OUTC 1
        STA A    ACIA D
        PUL B
        RTS

```

**\*INPUT ACIA KARAKTER:**

```

INCHR  LDA A   ACIA S
        ASR A
        BCC    INCHR
        LDA A   ACIA D
        RTS

```

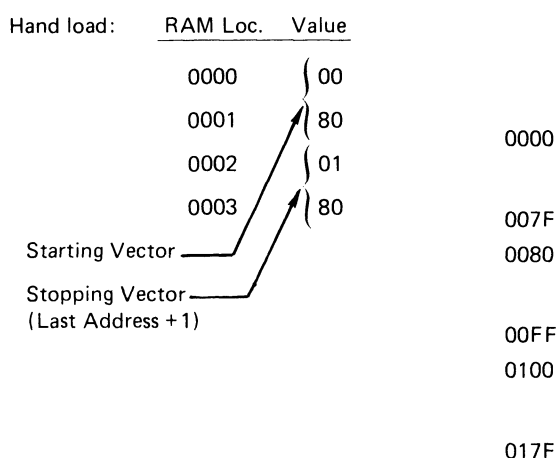
(RDRF → carry)  
(ER DATA READY ?)  
input karakterdata

**LOAD/DUMP VIA ACIA**

Det på de følgende sider viste program kan illustrere anvendelsen af ACIA i forbindelse med IND- og UDLÆSNING mellem microcomputer og båndoptager (ref. Motorola microcomputer course).

**ACIA MEMORY LOAD/DUMP PROGRAM****Example**

Assume three 128 x 8 MCM6810 RAMs in a system. It is desired to load a 256-byte program into the two upper RAMs starting with address 0080.



After the hand-loading of the starting and stopping vectors, the load program is executed by starting the MPU at program address 0900. When the program has finished loading, the CA2 line of PIA1 will go low. This signal can be used to stop the tape recorder or turn off a lamp to indicate the end of the loading process.

The *memory dump program* works as follows: The start-and-stop memory dump addresses or vectors are hand-loaded into RAM locations 0000, 0001, 0002, and 0003 in the same manner as in the previous load program. Program execution begins at memory address 094B. The characters AA and 55 are first dumped or placed on the tape in order to indicate the beginning of memory dump or listing. Each program character or byte is dumped via the ACIA and Modem until the last memory location has been addressed and dumped. When the dump operation is complete, the CA2 lead of PIA1 will go low, indicating dump complete.



Load via ACIA (MC6850)

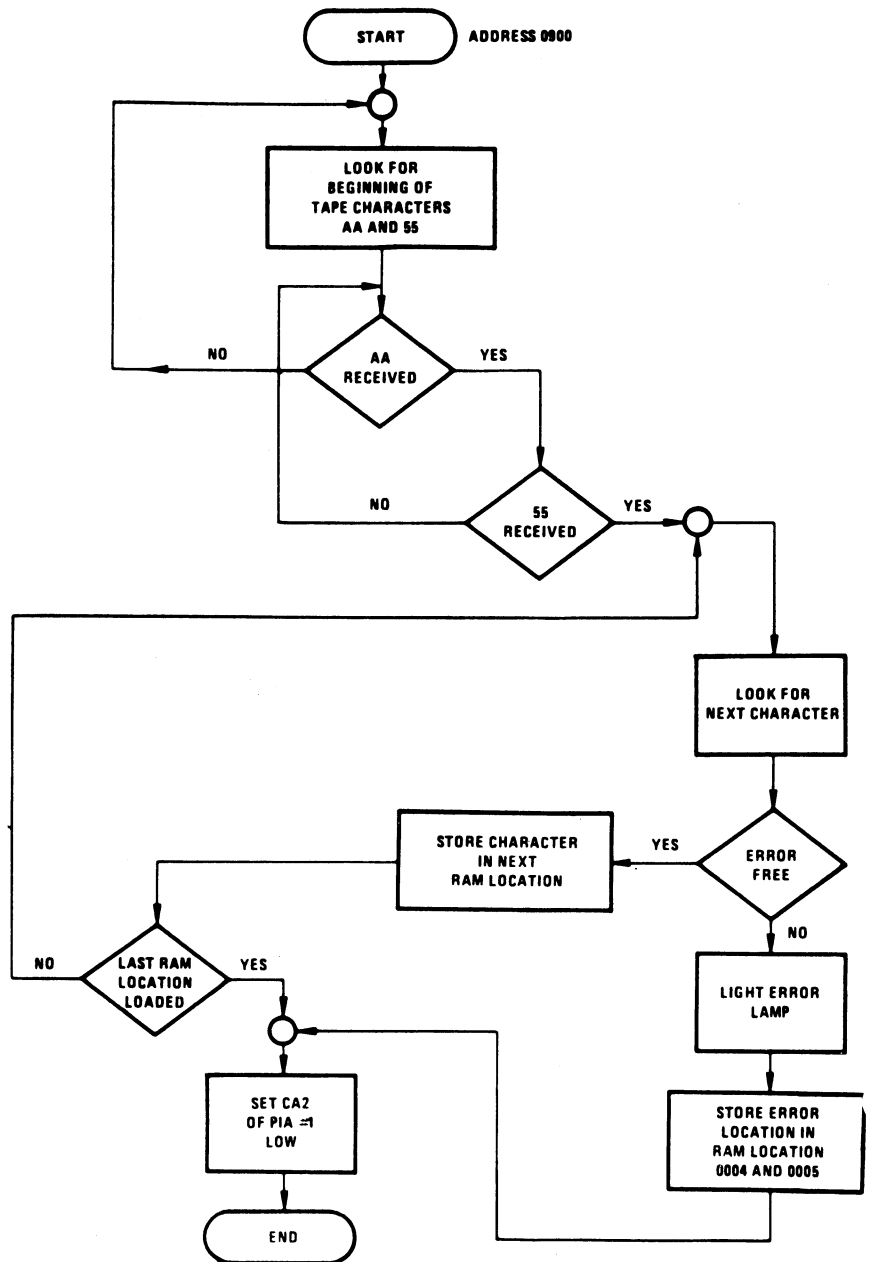
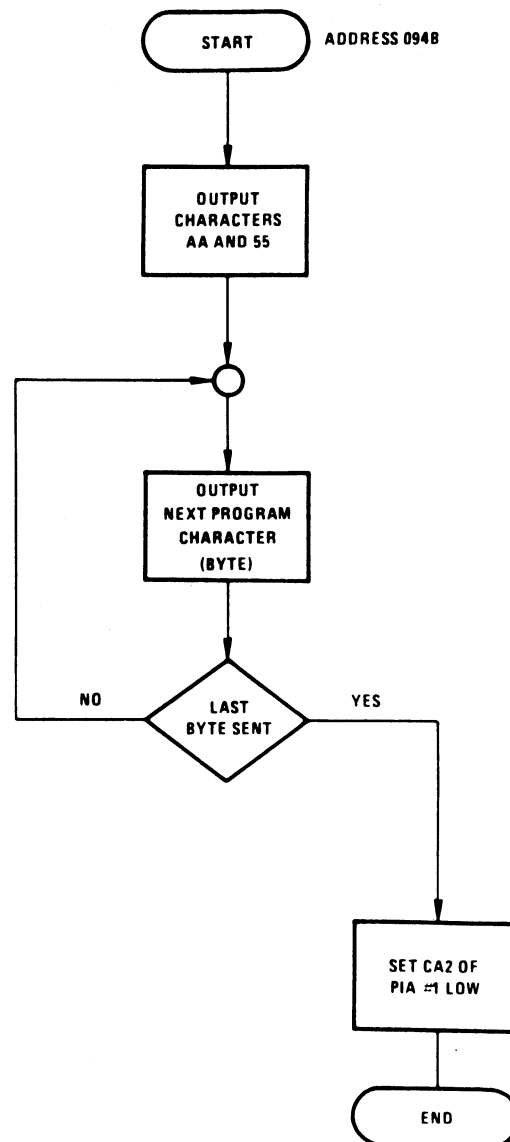



Fig. 7.55. Ⓐ

## Dump via ACIA (MC6850)


Fig. 7.56. 

## Source Program for Load/Dump via ACIA Program

```

1.000  NAM LDBOOT
2.000  OPT M
3.000  ♦ THIS PROGRAM LOADS OR DUMPS MEMORY
4.000  ♦ PLACE START ADDRESS IN LOC 00 & 01
5.000  ♦ PLACE END ADDRESS + 1 IN LOC 02 & 03
6.000  ♦ IF ERROR OCCURS, CHECK LOC 04 & 05 FOR ADDRESS.
7.000  ♦ CA2 STOPS DRIVE AT EDT OR ERROR.
8.000  ♦ CB2 GIVES ERROR INDICATION.
9.000  ♦ DUMP PROGRAM STARTS AT LOC 094B.
10.000 PIA1AC EQU $0805
11.000 PIA1BC EQU $0807
12.000 ACIAC EQU $0806
13.000 ACIAD EQU $0809
14.000 ORG $0900
15.000 LDA A #$03
16.000 STA A ACIAC ACIA MASTER RESET
17.000 LDX $00 LOAD START ADDRESS
18.000 LDA A #$19 ACIA 8 BITS EVEN PARITY
19.000 STA A ACIAC
20.000 LOOP LDA A ACIAC
21.000 ROR A
22.000 BCC LOOP RECEIVER FULL?
23.000 LDA A ACIAD
24.000 CMPA CMP A #$AA IS FIRST CHAR "AA"?
25.000 BNE LOOP BRANCH IF NOT
26.000 LOOP1 LDA A ACIAC
27.000 ROR A
28.000 BCC LOOP1
29.000 LDA A ACIAD
30.000 CMP A #$55 IS SECOND CHAR "55"?
31.000 BNE CMPA IF NOT, TRY FOR AN "AA"
32.000 LOOP2 LDA A ACIAC
33.000 TAB TRANSFER A TO B
34.000 AND B #$70
35.000 BNE ERROR BRANCH IF ERROR
36.000 ROR A
37.000 BCC LOOP2
38.000 LDA A ACIAD LOAD A CHAR FROM TAPE
39.000 STA A 0,X STORE IN MEMORY
40.000 INX INCREMENT ADDRESS
41.000 CPX $02 LOAD COMPLETED?
42.000 BNE LOOP2 GO GET MORE
43.000 END LDA A #$30
44.000 STA A PIA1AC TURN OFF CA2
45.000 BRA ♦
46.000 ERROR LDA A #$36
47.000 STA A PIA1BC TURN ON ERROR LIGHT
48.000 STX $04 STORE ADR OF ERROR
49.000 BRA END
50.000 PAGE
51.000 LDX $00 ♦START OF DUMP PROGRAM
52.000 LDA A #$19
53.000 STA A ACIAC
54.000 LDA A #$AA ♦OUTPUT CONTROL CHAR
55.000 STA A ACIAD
56.000 LOOP5 LDA A ACIAC
57.000 ROR A
58.000 ROR A
59.000 BCC LOOP5 ♦XMIT BUFFER EMPTY?
60.000 LDA A #$55 ♦OUTPUT SECOND CONTROL CHAR
61.000 STA A ACIAD
62.000 LOOP6 LDA A ACIAC
63.000 ROR A
64.000 ROR A
65.000 BCC LOOP6 ♦XMIT BUFFER EMPTY?
66.000 LOOP4 LDA A 0,X
67.000 STA A ACIAD ♦OUTPUT CHAR TO TAPE
68.000 LOOP3 LDA A ACIAC
69.000 ROR A
70.000 ROR A
71.000 BCC LOOP3 ♦XMIT BUFFER EMPTY?
72.000 INX
73.000 CPX $02
74.000 BNE LOOP4
75.000 BRA END
76.000 MON

```

Fig. 7.57. 

## Assembled Program for Load/Dump via ACIA Program

```

00010          NAM      LDBOOT
00020          OPT      M
00030          ♦ THIS PROGRAM LOADS OR DUMPS MEMORY
00040          ♦ PLACE START ADDRESS IN LOC 00 & 01
00050          ♦ PLACE END ADDRESS + 1 IN LOC 02 & 03
00060          ♦ IF ERROR OCCURS, CHECK LOC 04 & 05 FOR ADDRESS.
00070          ♦ CA2 STOPS DRIVE AT EOT OR ERROR.
00080          ♦ CB2 GIVES ERROR INDICATION.
00090          ♦ DUMP PROGRAM STARTS AT LOC 094B.
00100      0805  PIA1AC EQU    $0805
00110      0807  PIA1BC EQU    $0807
00120      0806  ACIAC  EQU    $0806
00130      0809  ACIAD  EQU    $0809
00140      0900          ORG    $0900
00150      0900 86 03          LDA A    #$03
00160      0902 B7 0806        STA A    ACIAC      ACIA MASTER RESET
00170      0905 DE 00          LDX      $00        LOAD START ADDRESS
00180      0907 86 19          LDA A    #$19        ACIA 8 BITS EVEN PARITY
00190      0909 B7 0806        STA A    ACIAC
00200      090C B6 0806 LOOP   LDA A    ACIAC
00210      090F 46            ROR A
00220      0910 24 FA          BCC      LOOP      RECEIVER FULL?
00230      0912 B6 0809        LDA A    ACIAD
00240      0915 81 AA          CMPA    #$AA      IS FIRST CHAR "AA"?
00250      0917 26 F3          BNE      LOOP      BRANCH IF NOT
00260      0919 B6 0806 LOOP1  LDA A    ACIAC
00270      091C 46            ROR A
00280      091D 24 FA          BCC      LOOP1
00290      091F B6 0809        LDA A    ACIAD
00300      0922 81 55          CMPA    #$55      IS SECOND CHAR "55"?
00310      0924 26 EF          BNE      CMPA      IF NOT, TRY FOR AN "AA"
00320      0926 B6 0806 LOOP2  LDA A    ACIAC
00330      0929 16            TAB
00340      092A C4 70          AND B    #$70
00350      092C 26 14          BNE      ERROR     BRANCH IF ERROR
00360      092E 46            ROR A
00370      092F 24 F5          BCC      LOOP2
00380      0931 B6 0809        LDA A    ACIAD      LOAD A CHAR FROM TAPE
00390      0934 A7 00          STA A    0,X      STORE IN MEMORY
00400      0936 08            INX
00410      0937 9C 02          CPX      $02      INCREMENT ADDRESS
00420      0939 26 EB          BNE      LOOP2     LOAD COMPLETED?
00430      093B 86 30          LDA A    #$30      GO GET MORE
00440      093D B7 0805        STA A    PIA1AC
00450      0940 20 FE          BRA      ♦
00460      0942 86 36          LDA A    #$36      TURN OFF CA2
00470      0944 B7 0807        STA A    PIA1BC
00480      0947 DF 04          STX      $04      TURN ON ERROR LIGHT
00490      0949 20 F0          BRA      END        STORE ADR OF ERROR
00510      094B DE 00          LDX      $00      ♦START OF DUMP PROGRAM
00520      094D 86 19          LDA A    #$19
00530      094F B7 0806        STA A    ACIAC
00540      0952 86 AA          LDA A    #$AA      ♦OUTPUT CONTROL CHAR
00550      0954 B7 0809        STA A    ACIAD
00560      0957 B6 0806 LOOP5  LDA A    ACIAC
00570      095A 46            ROR A
00580      095B 46            ROR A
00590      095C 24 F9          BCC      LOOP5     ♦XMIT BUFFER EMPTY?
00600      095E 86 55          LDA A    #$55      ♦OUTPUT SECOND CONTROL CHAR
00610      0960 B7 0809        STA A    ACIAD
00620      0963 B6 0806 LOOP6  LDA A    ACIAC
00630      0966 46            ROR A
00640      0967 46            ROR A
00650      0968 24 F9          BCC      LOOP6     ♦XMIT BUFFER EMPTY?
00660      096A A6 00          LDA A    0,X
00670      096C B7 0809        STA A    ACIAD      ♦OUTPUT CHAR TO TAPE
00680      096F B6 0806 LOOP3  LDA A    ACIAC
00690      0972 46            ROR A
00700      0973 46            ROR A
00710      0974 24 F9          BCC      LOOP3     ♦XMIT BUFFER EMPTY?
00720      0976 08            INX
00730      0977 9C 02          CPX      $02
00740      0979 26 EF          BNE      LOOP4
00750      097B 20 BE          BRA      END
00760          MON

```

Fig. 7.58. (M)



**MOTOROLA**  
**Semiconductors**

BOX 20912 • PHOENIX, ARIZONA 85036

**MC6850**

### ASYNCHRONOUS COMMUNICATIONS INTERFACE ADAPTER (ACIA)

The MC6850 Asynchronous Communications Interface Adapter provides the data formatting and control to interface serial asynchronous data communications information to bus organized systems such as the MC6800 Microprocessing Unit.

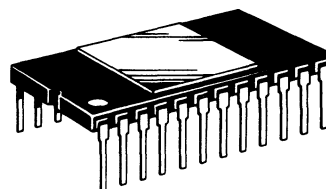
The bus interface of the MC6850 includes select, enable, read/write, interrupt and bus interface logic to allow data transfer over an 8-bit bi-directional data bus. The parallel data of the bus system is serially transmitted and received by the asynchronous data interface, with proper formatting and error checking. The functional configuration of the ACIA is programmed via the data bus during system initialization. A programmable Control Register provides variable word lengths, clock division ratios, transmit control, receive control, and interrupt control. For peripheral or modem operation three control lines are provided. These lines allow the ACIA to interface directly with the MC6860L 0-600 bps digital modem.

- Eight and Nine-Bit Transmission
- Optional Even and Odd Parity
- Parity, Overrun and Framing Error Checking
- Programmable Control Register
- Optional  $\div 1$ ,  $\div 16$ , and  $\div 64$  Clock Modes
- Up to 500 kbps Transmission
- False Start Bit Deletion
- Peripheral/Modem Control Functions
- Double Buffered
- One or Two Stop Bit Operation

**MOS**

(N-CHANNEL, SILICON-GATE)

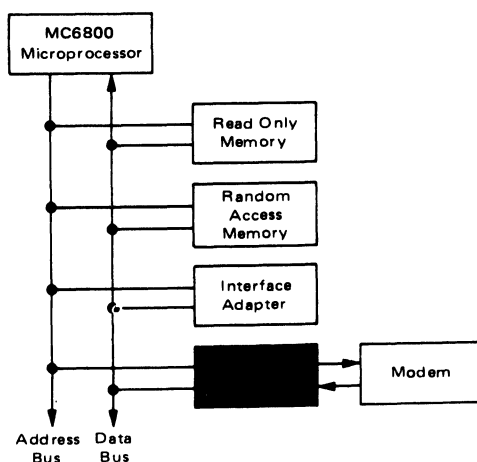
### ASYNCHRONOUS COMMUNICATIONS INTERFACE ADAPTER



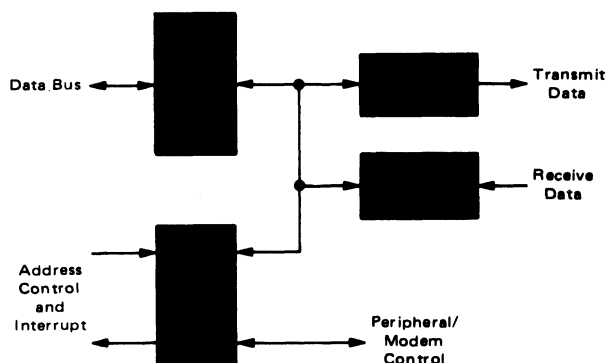
**L SUFFIX**  
CERAMIC PACKAGE  
CASE 716

NOT SHOWN: **P SUFFIX**  
PLASTIC PACKAGE  
CASE 709

**MC6800 MICROCOMPUTER FAMILY  
BLOCK DIAGRAM**



**MC6850 ASYNCHRONOUS COMMUNICATIONS INTERFACE ADAPTER  
BLOCK DIAGRAM**



## MAXIMUM RATINGS

Rating	Symbol	Value	Unit
Supply Voltage	V <sub>CC</sub>	-0.3 to +7.0	Vdc
Input Voltage	V <sub>in</sub>	-0.3 to +7.0	Vdc
Operating Temperature Range	T <sub>A</sub>	0 to +70	°C
Storage Temperature Range	T <sub>stg</sub>	-55 to +150	°C
Thermal Resistance	θ <sub>JA</sub>	82.5	°C/W

This device contains circuitry to protect the inputs against damage due to high static voltages or electric fields; however, it is advised that normal precautions be taken to avoid application of any voltage higher than maximum rated voltages to this high-impedance circuit.

## ELECTRICAL CHARACTERISTICS (V<sub>CC</sub> = 5.0 V ±5%, V<sub>SS</sub> = 0, T<sub>A</sub> = 0 to 70°C unless otherwise noted.)

Characteristic	Symbol	Min	Typ	Max	Unit
Input High Voltage	V <sub>IH</sub>	V <sub>SS</sub> + 2.0	—	V <sub>CC</sub>	Vdc
Input Low Voltage	V <sub>IL</sub>	V <sub>SS</sub> - 0.3	—	V <sub>SS</sub> + 0.8	Vdc
Input Leakage Current (V <sub>in</sub> = 0 to 5.25 Vdc)	I <sub>in</sub>	—	1.0	2.5	µAdc
Three-State (Off State) Input Current (V <sub>in</sub> = 0.4 to 2.4 Vdc)	I <sub>TSI</sub>	—	2.0	10	µAdc
Output High Voltage (I <sub>Load</sub> = -205 µAdc, Enable Pulse Width < 25 µs) (I <sub>Load</sub> = -100 µAdc, Enable Pulse Width < 25 µs)	V <sub>OH</sub>	V <sub>SS</sub> + 2.4 V <sub>SS</sub> + 2.4	— —	— —	Vdc
Output Low Voltage (I <sub>Load</sub> = 1.6 mAdc, Enable Pulse Width < 25 µs)	V <sub>OL</sub>	—	—	V <sub>SS</sub> + 0.4	Vdc
Output Leakage Current (Off State) (V <sub>OH</sub> = 2.4 Vdc)	I <sub>LOH</sub>	—	1.0	10	µAdc
Power Dissipation	P <sub>D</sub>	—	300	525	mW
Input Capacitance (V <sub>in</sub> = 0, T <sub>A</sub> = 25°C, f = 1.0 MHz)	C <sub>in</sub>	—	10 7.0	12.5 7.5	pF
Output Capacitance (V <sub>in</sub> = 0, T <sub>A</sub> = 25°C, f = 1.0 MHz)	C <sub>out</sub>	—	—	10 5.0	pF
Minimum Clock Pulse Width, Low (Figure 1)	PW <sub>CL</sub>	600	—	—	ns
Minimum Clock Pulse Width, High (Figure 2)	PW <sub>CH</sub>	600	—	—	ns
Clock Frequency	f <sub>C</sub>	—	—	500 800	kHz
Clock-to-Data Delay for Transmitter (Figure 3)	t <sub>TDD</sub>	—	—	1.0	µs
Receive Data Setup Time (Figure 4)	t <sub>RDSU</sub>	500	—	—	ns
Receive Data Hold Time (Figure 5)	t <sub>RDH</sub>	500	—	—	ns
Interrupt Request Release Time (Figure 6)	t <sub>IR</sub>	—	—	1.2	µs
Request-to-Send Delay Time (Figure 6)	t <sub>RTS</sub>	—	—	1.0	µs
Input Transition Times (Except Enable)	t <sub>r</sub> , t <sub>f</sub>	—	—	1.0*	µs

\*1.0 µs or 10% of the pulse width, whichever is smaller.

## BUS TIMING CHARACTERISTICS

### READ (Figures 7 and 9)

Characteristic	Symbol	Min	Typ	Max	Unit
Enable Cycle Time	t <sub>cycE</sub>	1.0	—	—	µs
Enable Pulse Width, High	PW <sub>EH</sub>	0.45	—	25	µs
Enable Pulse Width, Low	PW <sub>EL</sub>	0.43	—	—	µs
Setup Time, Address and R/W valid to Enable positive transition	t <sub>AS</sub>	160	—	—	ns
Data Delay Time	t <sub>DDR</sub>	—	—	320	ns
Data Hold Time	t <sub>H</sub>	10	—	—	ns
Address Hold Time	t <sub>AH</sub>	10	—	—	ns
Rise and Fall Time for Enable input	t <sub>Er</sub> , t <sub>Ef</sub>	—	—	25	ns

### WRITE (Figure 8 and 9)

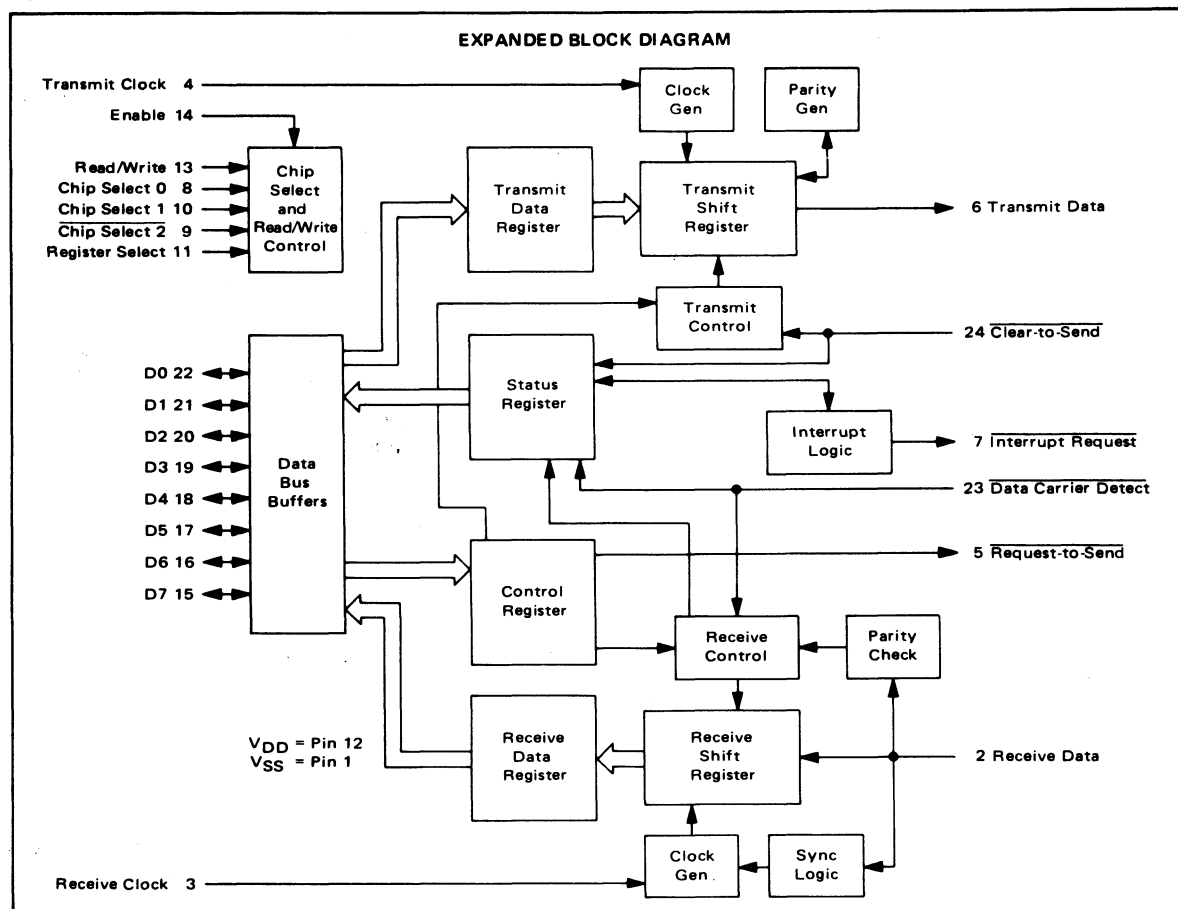
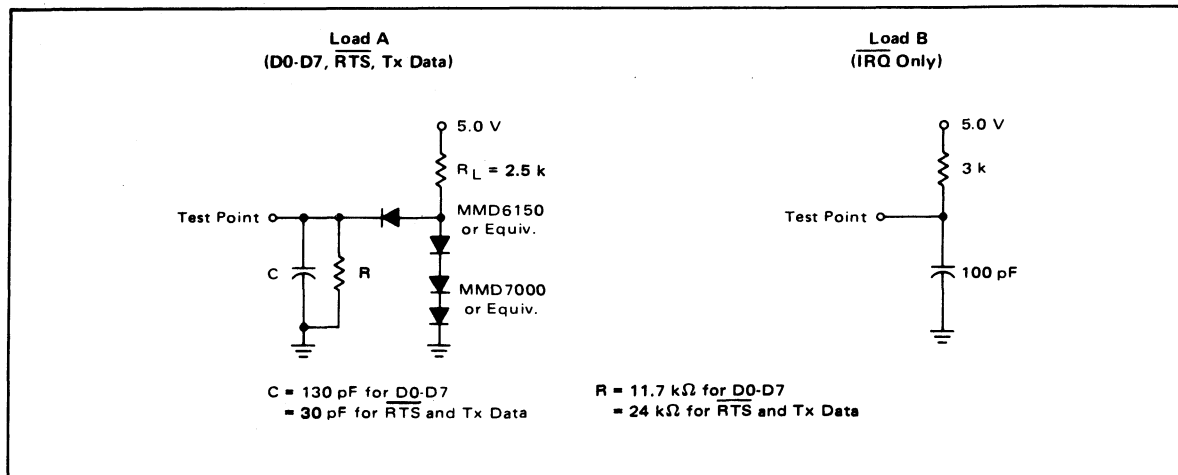
Characteristic	Symbol	Min	Typ	Max	Unit
Enable Cycle Time	t <sub>cycE</sub>	1.0	—	—	µs
Enable Pulse Width, High	PW <sub>EH</sub>	0.45	—	25	µs
Enable Pulse Width, Low	PW <sub>EL</sub>	0.43	—	—	µs
Setup Time, Address and R/W valid to Enable positive transition	t <sub>AS</sub>	160	—	—	ns
Data Setup Time	t <sub>DSW</sub>	195	—	—	ns
Data Hold Time	t <sub>H</sub>	10	—	—	ns
Address Hold Time	t <sub>AH</sub>	10	—	—	ns
Rise and Fall Time for Enable input	t <sub>Er</sub> , t <sub>Ef</sub>	—	—	25	ns



MOTOROLA Semiconductor Products Inc.



FIGURE 9 – BUS TIMING TEST LOADS

**DEVICE OPERATION**

At the bus interface, the ACIA appears as two addressable memory locations. Internally, there are four registers: two read-only and two write-only registers. The read-only

registers are Status and Receive Data; the write-only registers are Control and Transmit Data. The serial interface consists of serial input and output lines with independent clocks, and three peripheral/modem control lines.



FIGURE 1 – CLOCK PULSE WIDTH, LOW-STATE

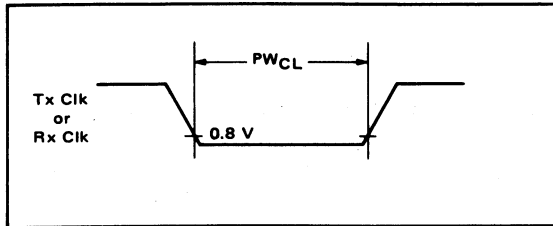


FIGURE 2 – CLOCK PULSE WIDTH, HIGH-STATE

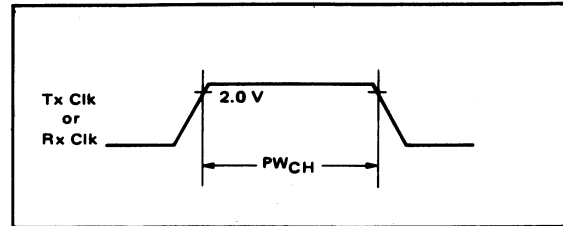
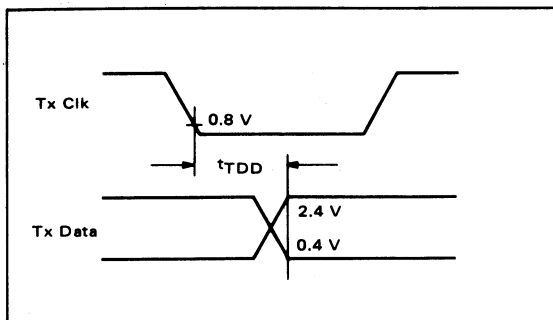
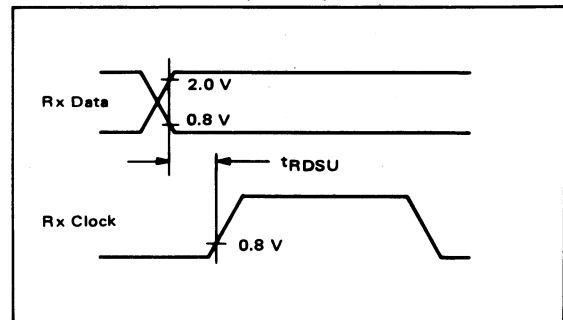
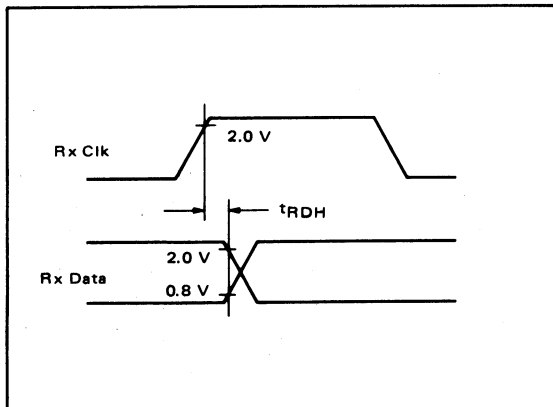
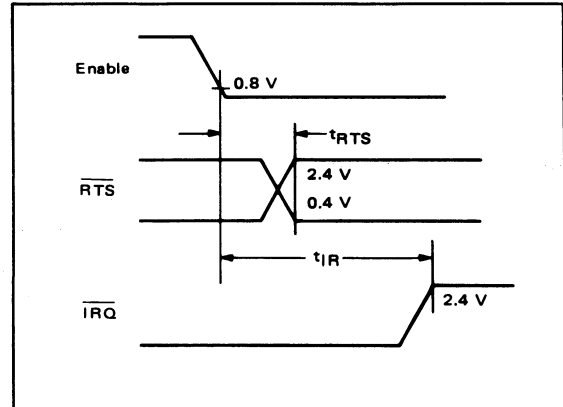
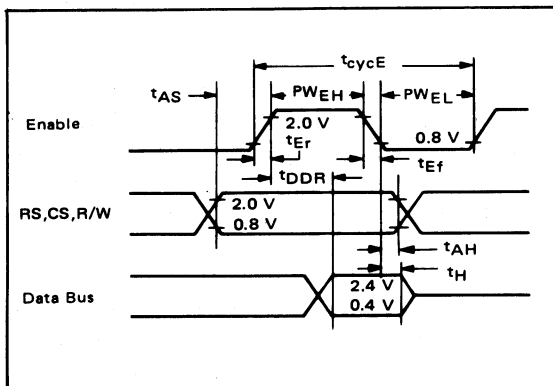
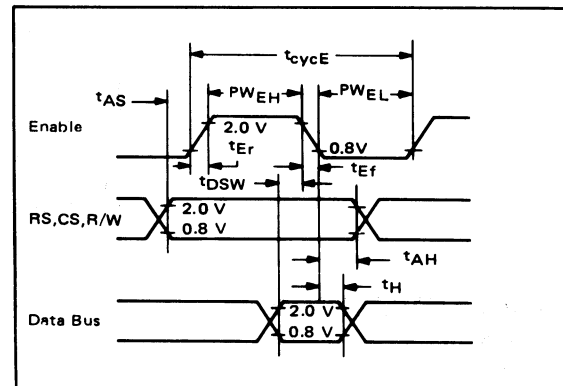


FIGURE 3 – TRANSMIT DATA OUTPUT DELAY

FIGURE 4 – RECEIVE DATA SETUP TIME  
(÷1 Mode)FIGURE 5 – RECEIVE DATA HOLD TIME  
(÷1 Mode)FIGURE 6 – REQUEST-TO-SEND DELAY AND  
INTERRUPT-REQUEST RELEASE TIMESFIGURE 7 – BUS READ TIMING CHARACTERISTICS  
(Read information from ACIA)FIGURE 8 – BUS WRITE TIMING CHARACTERISTICS  
(Write information into ACIA)

## PROGRAMMABLE READ ONLY MEMORY (EPROM)

De programmerbare ROMs, der anvendes idag, kan opdeles i hovedgrupperne som beskrevet i afsnit 3.

Her skal kun behandles de mest anvendte af UV-typerne. UV står for „sletning ved hjælp af ultraviolet lys” og hentyder til den påvirkning, kredsløbene udsættes for, når indholdet i hukommelsen skal slettes, før en ny programmering kan foretages. Udviklingen er gået over følgende typenumre:

**TABLE 2**  
**Typical Commercially Available MOS EPROMs (UV Type)**

Vendor	Model No.	Interchangeable ROM	Size	Organization	Access Time (max)	Power Supply	Maximum Current (mA) Active	Current (mA) Standby
Intel	1702A	1302	2k	256 x 8	1 $\mu$ s	5, -9 V	65	65
Intel	2704	—	4k	512 x 8	450 ns	12, $\pm$ 5 V	65, 45, 10	65, 45, 10
Intel	2708	2308	8k	1024 x 8	450 ns	12, $\pm$ 5 V	65, 45, 10	65, 45, 10
Intel	2716	2316E	16k	2048 x 8	450 ns	5 V	100	100
TI	2716*	—	16k	2048 x 8	450 ns	12, $\pm$ 5 V	45, 17, 6	45, 17, 6
Intel	2732	2332	32k	4096 x 8	300 ns	5 V	40	15
TI	2532	4732	32k	4096 x 8	450 ns	5 V	168	10

\*Interchangeable with Intel 2708, but being redesigned to be compatible with Intel 2716

og den nærmeste fremtid vil bringe

2732:  $4096 \times 8 = 32\text{k bits}$

og senere igen:

2764:  $8192 \times 8 = 64\text{k bits}$

## SLETNING AF EPROM

Til sletning af det binære indhold (programmet) i UV-EPROMer benyttes ULTRAVIOLET LYS med en bølglængde på ca. 4000 Ångström. Afgiver lampen ca. 12 mW pr.  $\text{cm}^2$  til chip'en i EPROM'en (d.v.s. afstanden mellem chip og lampe er 2,5 cm) tager sletningen 15–20 min., men det er „god latin” at belyse 30–40 min. for at være sikker på 100% sletning. Ved sletningen gendannes alle bit som 1-taller og genprogrammering er herefter mulig.

Sollyset og visse fluorescente lamper udsender lys af samme bølglængde, som benyttes til sletningen og kan derfor have slettende virkning på EPROM'er.

Lampernes påvirkning skal foregå over ca. 3 år, men sollyset vil i løbet af ca. 1 uge ved direkte belysning af chip'en have så meget indflydelse, at visse bits er ændret og indholdet i EPROM'en derfor ødelagt. For at modvirke denne bestråling forsynes vinduet i disse UV-EPROMs med et UV-filter, der kan klæbes på.

## PROGRAMMERING AF EPROM

Programmeringen af den enkelte EPROM fremgår af de respektive datablade. Generelt foregår programmeringen ved at påtrykke en DATA BYTE på dataoutputledningerne i EPROM'en. Den ADRESSE i EPROM'en, der ønskes at indeholde databyten, aktiveres ved hjælp af adresseindgangene. Når data og adresse er etableret, sendes en programmeringspuls ind på programmeringsterminalen. Programmeringsspændingen er for de mest anvendte typer på 25 volt.

Som eksempel på en programmering kan INTELS beskrivelse for 2716 benyttes:

### 2716 And 2758 Programming

Initially, and after each erasure, all bits of the 2716/2758 are in the "1" state. Information is introduced by selectively programming "0" into the desired bit locations. A programmed "0" can only be changed to a "1" by UV erasure.

The 2716/2758 is programmed by applying a 50 ms, TTL programming pulse to the  $\overline{\text{CE}}/\text{PGM}$  pin with the  $\overline{\text{OE}}$  input high and the  $V_{\text{PP}}$  supply at  $25\text{V} \pm 1\text{V}$ . Any location may be programmed at any time — either individually, sequentially, or randomly. The programming time for a single bit is only 50 ms and for all bits is approximately 100 and 50 seconds for the 2716 and 2758 respectively. The detailed programming specifications and timing waveforms are given in the following tables and figures.

**CAUTION:** The  $V_{\text{CC}}$  and  $V_{\text{PP}}$  supplied must be sequenced on and off such that  $V_{\text{CC}}$  is applied simultaneously or before  $V_{\text{PP}}$  and removed simultaneously or after  $V_{\text{PP}}$  to prevent damage to the 2716/2758. The maximum allowable voltage during programming which may be applied to the  $V_{\text{PP}}$  with respect to ground is +26V. Care must be taken when switching the  $V_{\text{PP}}$  supply to prevent overshoot exceeding the 26-volt maximum specification. For convenience in programming, the 2716/2758 may be verified with the  $V_{\text{PP}}$  supply at  $25\text{V} \pm 1\text{V}$ . During normal read operation, however,  $V_{\text{PP}}$  must be at  $V_{\text{CC}}$ .

#### PIN CONFIGURATION

A7	1	24	V <sub>CC</sub>
A6	2	23	A <sub>8</sub>
A5	3	22	A <sub>9</sub>
A4	4	21	V <sub>PP</sub>
A3	5	20	OE
A2	6	19	A <sub>10</sub>
A1	7	18	CE
A0	8	17	O <sub>7</sub>
O <sub>0</sub>	9	16	O <sub>6</sub>
O <sub>1</sub>	10	15	O <sub>5</sub>
O <sub>2</sub>	11	14	O <sub>4</sub>
GND	12	13	O <sub>3</sub>

#### PIN NAMES

A <sub>0</sub> –A <sub>9</sub>	ADDRESSES
CE/PGM	CHIP ENABLE/PROGRAM
OE	OUTPUT ENABLE
O <sub>0</sub> –O <sub>7</sub>	OUTPUTS

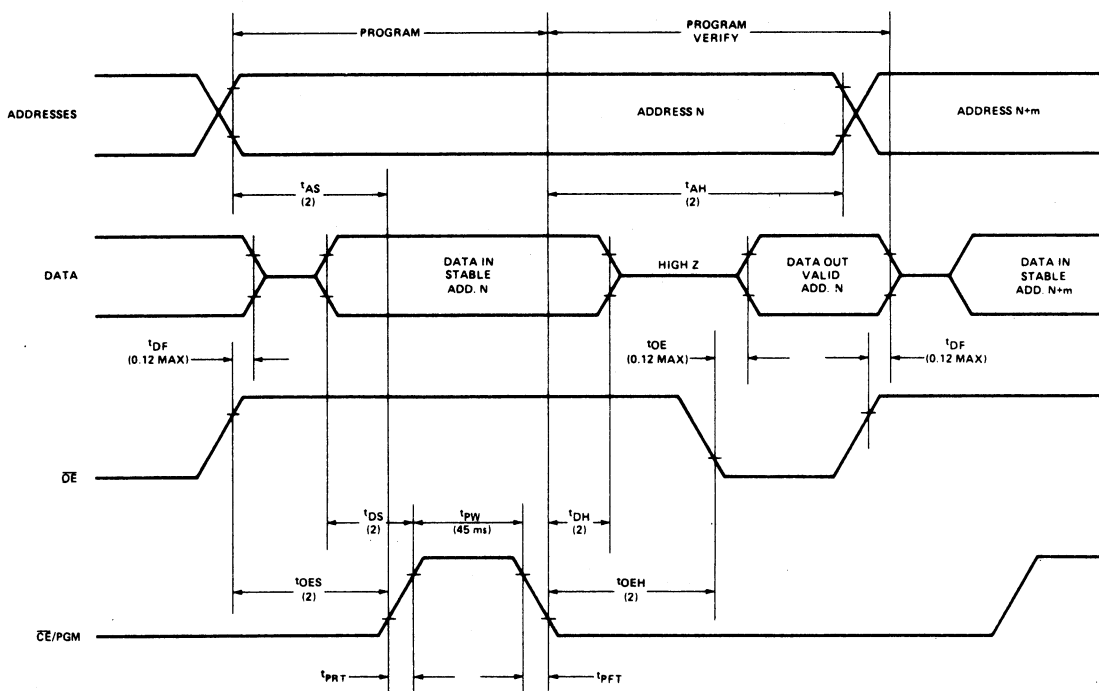
#### A.C. Conditions of Test:

$V_{\text{CC}}$  .....  $5\text{V} \pm 5\%$   
 $V_{\text{PP}}$  .....  $25\text{V} \pm 1\text{V}$   
 Input Rise and Fall Times (10% to 90%) ..... 20 ns

Input Pulse Levels ..... 0.8V to 2.2V  
 Input Timing Reference Level ..... 1V and 2V  
 Output Timing Reference Level ..... 0.8V and 2V

#### PROGRAMMING WAVEFORMS

$V_{\text{PP}} = 25\text{V} \pm 1\text{V}$ ,  $V_{\text{CC}} = 5\text{V} \pm 5\%$



NOTE: ALL TIMES SHOWN IN PARENTHESES ARE MINIMUM TIMES AND ARE  $\mu\text{SEC}$  UNLESS OTHERWISE NOTED.

Fig. 7.59. intel

Databladet for en af de mest benyttede EPROM: 2716 er fra INTEL vises herunder i uddrag:



## 2716\* 16K (2K × 8) UV ERASABLE PROM

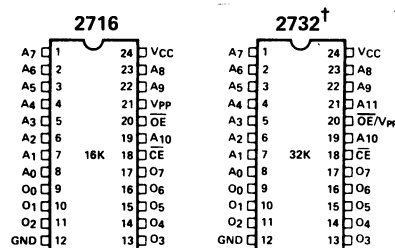
- **Fast Access Time**
  - 350 ns Max. 2716-1
  - 390 ns Max. 2716-2
  - 450 ns Max. 2716
- **Single +5V Power Supply**
- **Low Power Dissipation**
  - 525 mW Max. Active Power
  - 132 mW Max. Standby Power
- **Pin Compatible to Intel® 5V ROMs (2316E, 2332, and 2364) and 2732 EPROM**
- **Simple Programming Requirements**  
Single Location Programming Programs with One 50 ms Pulse
- **Inputs and Outputs TTL Compatible during Read and Program**
- **Completely Static**

The Intel® 2716 is a 16,384-bit ultraviolet erasable and electrically programmable read-only memory (EPROM). The 2716 operates from a single 5-volt power supply, has a static standby mode, and features fast single address location programming. It makes designing with EPROMs faster, easier and more economical. For production quantities, the 2716 user can convert rapidly to Intel's pin-for-pin compatible 16K ROM (the 2316E) or the new 32K and 64K ROMs (the 2332 and 2364 respectively).

The 2716, with its single 5-volt supply and with an access time up to 350 ns, is ideal for use with the newer high performance +5V microprocessors such as Intel's 8085 and 8086. The 2716 is also the first EPROM with a static standby mode which reduces the power dissipation without increasing access time. The maximum active power dissipation is 525 mW while the maximum standby power dissipation is only 132 mW, a 75% savings.

The 2716 has the simplest and fastest method yet devised for programming EPROMs — single pulse TTL level programming. No need for high voltage pulsing because all programming controls are handled by TTL signals. Now, it is possible to program on-board, in the system, in the field. Program any location at any time — either individually, sequentially or at random, with the 2716's single address location programming. Total programming time for all 16,384 bits is only 100 seconds.

### PIN CONFIGURATION\*



†Refer to 2732 data sheet for specifications

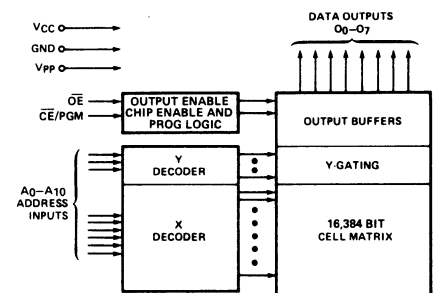
### PIN NAMES

A <sub>0</sub> –A <sub>7</sub>	ADDRESSES
CE/PGM	CHIP ENABLE/PROGRAM
OE	OUTPUT ENABLE
O <sub>0</sub> –O <sub>7</sub>	OUTPUTS

### MODE SELECTION

MODE	PINS	CE/PGM (18)	OE (20)	V <sub>pp</sub> (21)	V <sub>cc</sub> (24)	OUTPUT (9-11, 13-15)
Read		V <sub>IL</sub>	V <sub>IL</sub>	+5	+5	D <sub>OUT</sub>
Standby		V <sub>IH</sub>	Don't Care	+5	+5	High Z
Program		Pulsed V <sub>IL</sub> to V <sub>IH</sub>	V <sub>IH</sub>	+25	+5	D <sub>IN</sub>
Program Verify		V <sub>IL</sub>	V <sub>IL</sub>	+25	+5	D <sub>OUT</sub>
Program Inhibit		V <sub>IL</sub>	V <sub>IH</sub>	+25	+5	High Z

### BLOCK DIAGRAM



\*Pin 18 and pin 20 have been renamed to conform with the entire family of 16K, 32K, and 64K EPROMs and ROMs. The die, fabrication process, and specifications remain the same and are totally unaffected by this change.

## COMPATIBLE KREDSLØB

Efterhånden som fremstillingsprocessen for EPROM'er forbedres, kan der placeres flere og flere bit på hver chip, d.v.s. hver dual in line indeholder flere bytes nu end i de ældre typer. For at lette udskiftningen til nye og „større” typer i eksisterende udstyr, gøres der meget for, at nye kredsløb er PIN COMPATIBLE med de ældre. Ved pin kompatibilitet forstås, at placeringen af benene på DIL'en er så ensartet som muligt, når man ser på anvendelsen af kredsene.

## EPROM/ROM Compatibility

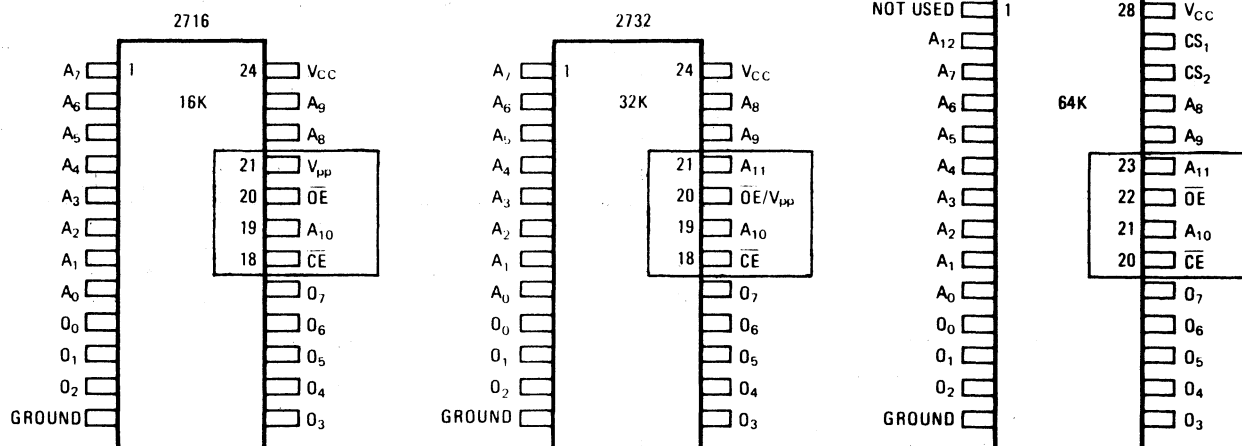


Fig. 7.60. intel

**INTEL 2716 ANVENDT I  
PRESS 8041**

Nedenfor angives benforbindelserne til 2716 anvendt i S & W  
PRESS 8041

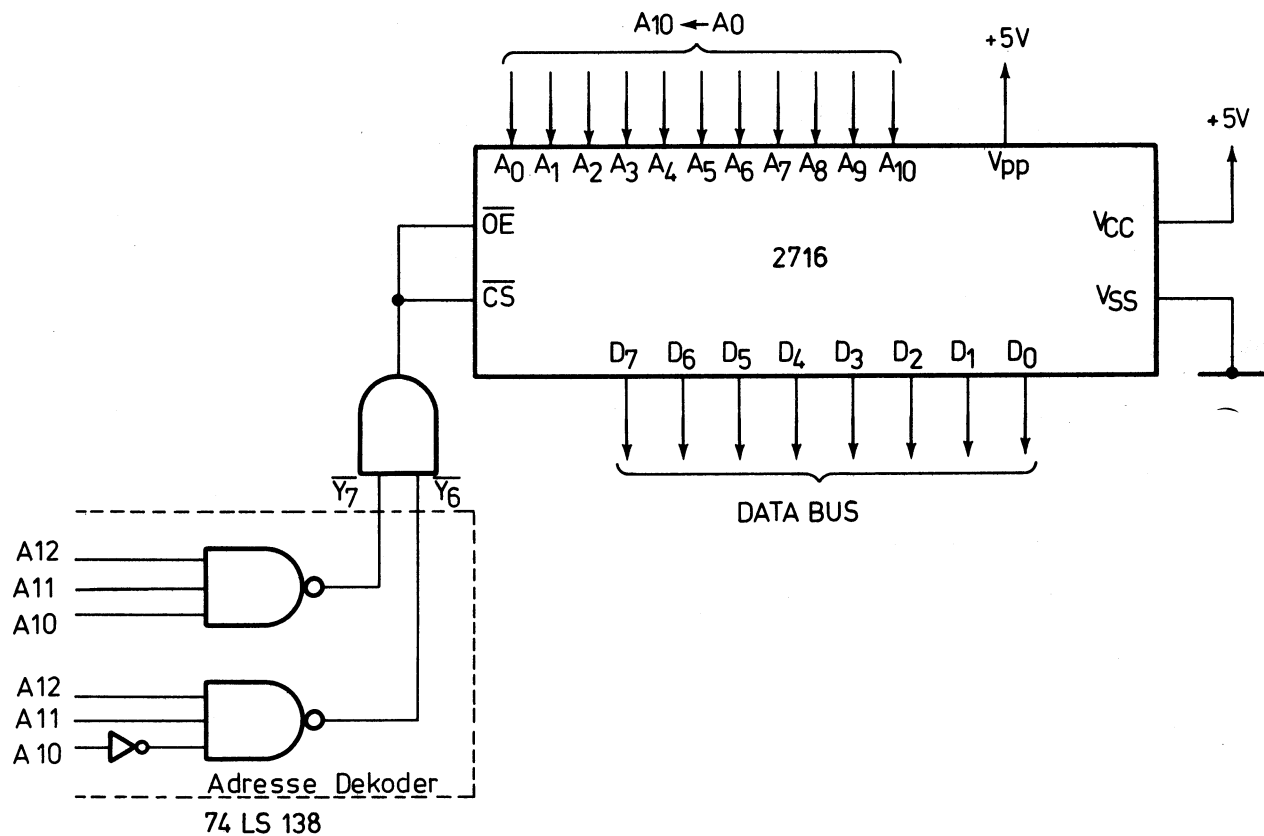


Fig. 7.61.

Chip select ( $\overline{CS}$ ) og output enable ( $\overline{OE}$ ) er aktive LOW input og fungerer når følgende betingelser er opfyldt:

A <sub>12</sub>	A <sub>11</sub>	A <sub>10</sub>	$\overline{Y_7}$	$\overline{Y_6}$	$\overline{CS} = \overline{OE}$
1	1	0	1	0	0
1	1	1	0	1	0

og dette fører frem til følgende memory map for 2716:

SYSTEM LAYOUT WORK SHEET																SYS	
MPU Address Lines (A0 – A15)																Address	
Device	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	From To
IC 301 2716 EPROM	-	-	-	CS	CS	x	x	x	x	x	x	x	x	x	x	x	1800 – 1FFF

x = variable, CS = aktive HI,  $\overline{\text{CS}}$  = aktive LO, - = not used

Fig. 7.62.

Adresseområdet for 2716 kan findes som:

0001    1xxx    xxxx    xxxx    (BINÆR)

1    8 → F    Ø → F    Ø → F    (HEX)

og bestemmes heraf til at ligge i området:

1800 → 1FFF    (HEX)

Da der ikke anvendes fuld dekodning af adressebussen, når  $A_{15} \leftarrow A_{13}$  ikke er anvendt, vil IC 301 totalt ligge på:

xxx1    1xxx    xxxx    xxxx    (BIN)

$\left\{ \begin{array}{l} \text{alle} \\ \text{ulige} \end{array} \right\} \cdot \left\{ 8 \rightarrow F \right\} \cdot \left\{ \emptyset \rightarrow F \right\} \cdot \left\{ \emptyset \rightarrow F \right\}$  (HEX)

hvilket betyder, at samme byte i 2716 kan kaldes af „forskellige” adresser på adressebussen f.eks.

1800 = 3800 = 5800 ————— B800 = D800 = F800

(se også memory mapping (side 7.62)).

## ADRESSEDEKODER

I de adresserbare kredsløb omkring en microprocessor (ROM, RAM og I/O) findes der internt kredsløb, som er i stand til at føle på processorens adresseledninger og herved „se”, om de bliver „udpeget”, d.v.s. adresseret af microprocessoren og desuden også, hvilken byte, der er udpeget. Disse interne kredsløb benævnes ADRESSEDEKODERE. De virker på den måde, at et vist antal INPUT ledninger er tilsluttet lige så mange adresseledninger på adressebussen, f.eks. 4 stk. (se fig. 7.63). Outputledningernes antal (Y) er bestemt af inputledningernes antal (n), idet

$$Y = 2^n$$

og det vil for eksempel sige  $Y = 2^4 = 16$ , hvilket ses af figuren.

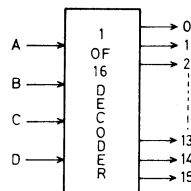
Af outputledningerne vil kun EN ad gangen være på aktivt niveau. Den viste adresse DEKODER med 4 input og 16 output kaldes:



## 4 lines to 16 lines decoder

eller


1 of 16 lines decoder.



D	C	B	A	ACTIVE OUTPUT
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	10
1	0	1	1	11
1	1	0	0	12
1	1	0	1	13
1	1	1	0	14
1	1	1	1	15

These address decoders are integrated into the individual circuits. They are available in different sizes as shown in the following table.

LINES INPUT	2	3	4	5	6	7	8	9	10	11
LINES OUTPUT	4	8	16	32	64	128	256	512	1024	2048

Fig. 7.63. **74LS138  
DEKODER**

Kredsløbet 74LS138 er

3 lines to 8 lines

**DECODER/DEMULTIPLEXER**

hvilket betyder, at der er 3 inputledninger, der modtager logiske signaler (binære værdier). For hver af de binære kombinationer, der kan skabes på input, er ET og KUN ET af de 8 output aktive. I 74LS138 er output aktiv LOW, hvilket giver, at kun ET OUTPUT er LOW ad gangen.

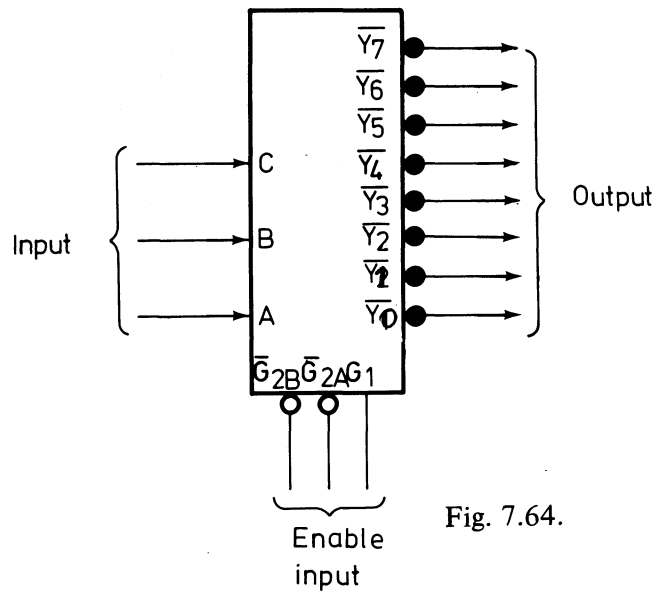


Fig. 7.64.

Fra T.I. databook er følgende data i uddrag hentet:

### TTL MSI

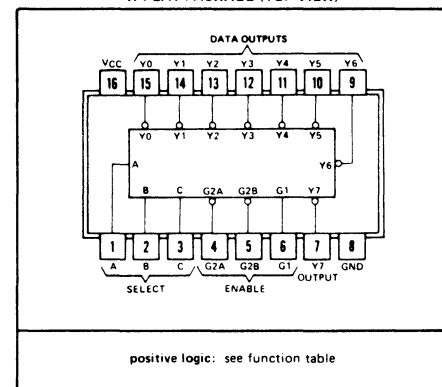
### TYPES SN54LS138, SN54LS139, SN54S138, SN54S139, SN74LS138, SN74LS139, SN74S138, SN74S139 DECODERS/DEMULPLEXERS

BULLETIN NO. DLS 7211804, DECEMBER 1972

- Designed Specifically for High-Speed: Memory Decoders Data Transmission Systems
- 'S138 and 'LS138 3-to-8-Line Decoders Incorporate 3 Enable Inputs to Simplify Cascading and/or Data Reception
- 'S139 and 'LS139 Contain Two Fully Independent 2-to-4-Line Decoders/ Demultiplexers
- Schottky Clamped for High Performance

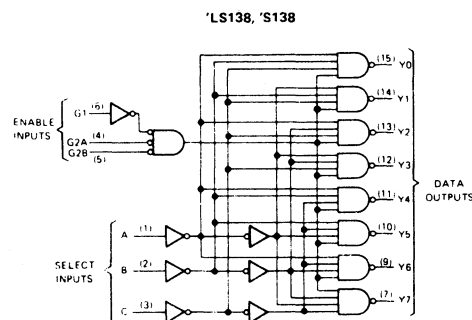
TYPE	TYPICAL PROPAGATION DELAY (3 LEVELS OF LOGIC)	TYPICAL POWER DISSIPATION
'LS138	22 ns	32 mW
'S138	8 ns	245 mW
'LS139	22 ns	34 mW
'S139	7.5 ns	300 mW

'LS138, 'S138  
J OR N DUAL-IN-LINE OR  
W FLAT PACKAGE (TOP VIEW)



positive logic: see function table

functional block diagrams and logic



'LS138, 'S138  
FUNCTION TABLE

ENABLE		SELECT			OUTPUTS							
G1	G2*	C	B	A	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
X	H	X	X	X	H	H	H	H	H	H	H	H
L	X	X	X	X	H	H	H	H	H	H	H	H
H	L	L	L	L	L	H	H	H	H	H	H	H
H	L	L	L	H	L	H	H	H	H	H	H	H
H	L	L	L	L	H	L	H	H	H	H	H	H
H	L	L	H	H	H	H	L	H	H	H	H	H
H	L	L	H	L	H	H	H	L	H	H	H	H
H	L	L	H	L	H	H	H	H	L	H	H	H
H	L	H	H	L	H	H	H	H	H	L	H	H
H	L	H	H	H	H	H	H	H	H	H	L	H
H	L	H	H	H	H	H	H	H	H	H	H	L
H	L	H	H	H	H	H	H	H	H	H	H	L

\*G2 = G2A + G2B

H = high level, L = low level, X = irrelevant

Fig. 7.65.

Kredsløbet benyttes ved microcomputerkredsløb som adressedekoder for chip select signaler, når der på de adresserbare kredse som RAM, ROM og I/O ikke findes tilstrækkelige chip select input.

Anvendes 'LS138, betyder det, at kredsløb med 1 stk. Chip Select (CS) „ser ud”, som om de var i besiddelse af 3 stk. chip select input, hvilket kan belyses med følgende eksempel:

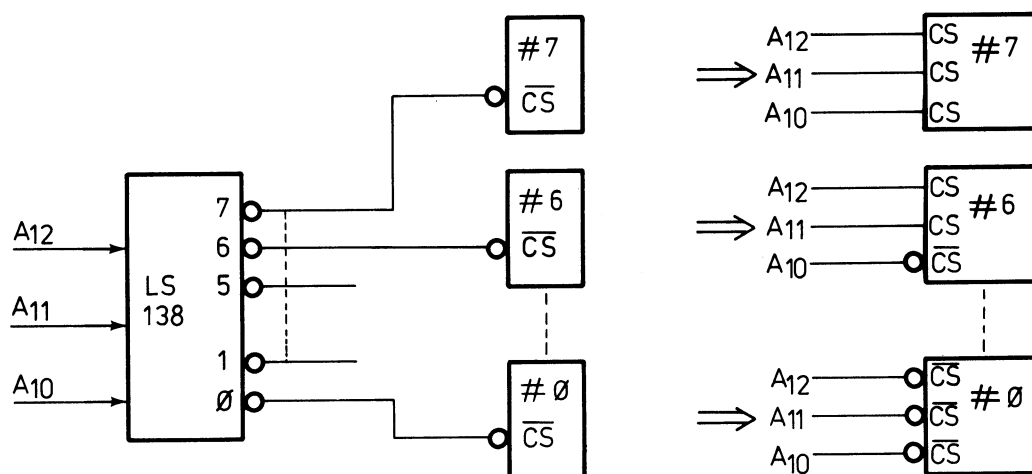


Fig. 7.66.

A <sub>12</sub>	A <sub>11</sub>	A <sub>10</sub>	'LS138 LO OUT	AKTIV KREDS
0	0	0	0	# 0
0	0	1	1	# 1
0	1	0	2	# 2
0	1	1	3	# 3
1	0	0	4	# 4
1	0	1	5	# 5
1	1	0	6	# 6
1	1	1	7	# 7

## MEMORY MAP

De adresserbare kredsløb har to typer af input-terminaler tilsluttet CPU'ens adressebus og disse to typer er:

1. Adresse-ledningerne
2. Chip Select-ledningerne

Ad 1:

Adresseledningernes opgave er at udpege det område (den byte), der ønskes INDENI DET ENKELTE KREDSLØB. Der kan her være tale om en enkelt memory byte eller et enkelt register i en I/O enhed.

Ad 2:

Chip Select-ledningerne benyttes til SELEKTIV UDVÆLGELSE af det enkelte kredsløb (Chip), som CPU'en ønsker at komme i kontakt med.

Som omtalt i foregående afsnit anvendes ofte et YDRE ADRESSE-DEKODNINGSKREDSLØB i forbindelse med chip select.

Som eksempel på chip select i et system kan S & W press 8041 anvendes. Dette system er vist i fig. 7.67, hvor der kun er medtaget signaler af betydning for Chip Select og intern adressering på chip'en.

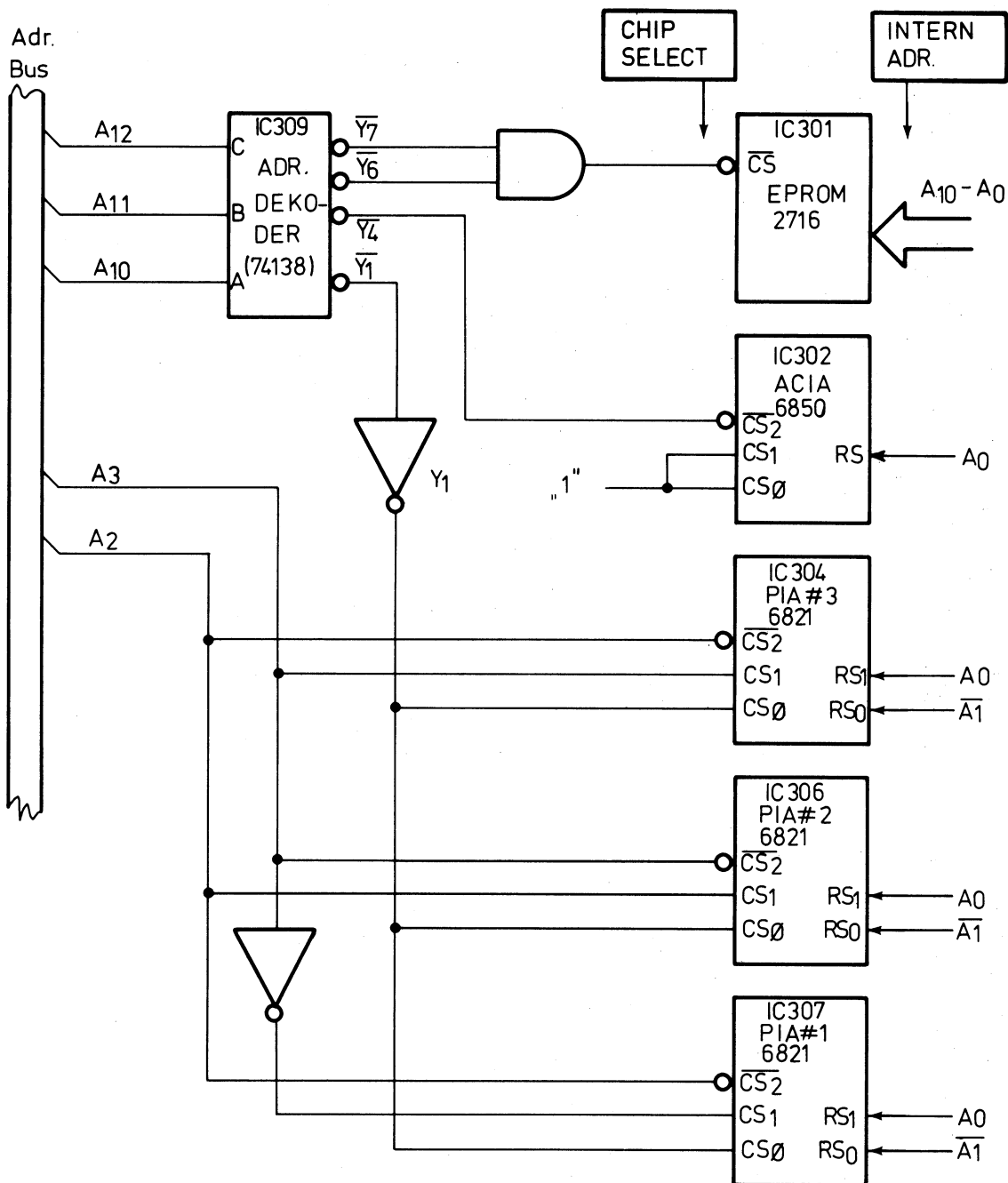


Fig. 7.67.

Ud fra fig. 7.67 kan memory map i fig. 7.68 udledes:

SYSTEM LAYOUT WORK SHEET																	Address	
MPU Address Lines (A0 A15)																	From	To
Device	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
301: 2716 EPROM	-	-	-	1	1	X	X	X	X	X	X	X	X	X	X	X	1800	→ 1FFF
302: 6850 ACIA	-	-	-	1	0	0	-	-	-	-	-	-	-	-	-	RS	1000	→ 1001
304: 6821 PIA#3	-	-	-	0	0	1	-	-	-	-	-	-	1	0	RS	RS1	0408	→ 040B
306: 6821 PIA#2	-	-	-	0	0	1	-	-	-	-	-	-	0	1	RS	RS1	0404	→ 0407
307: 6821 PIA#1	-	-	-	0	0	1	-	-	-	-	-	-	0	0	RS	RS1	0400	→ 0403

ADR. DECODE

Fig. 7.68.

Sammenhængen mellem fig. 7.67 og 7.68 kan f.eks. udtrykkes som følgende:

$$\begin{aligned}
 301: \quad \overline{CS} &= \overline{Y_7} \cdot \overline{Y_6} = \overline{A_{12} \cdot A_{11} \cdot A_{10}} \cdot \overline{A_{12} \cdot A_{11} \cdot A_{10}} \\
 &= \overline{A_{12} \cdot A_{11} \cdot A_{10}} + A_{12} \cdot A_{11} \cdot A_{10} \\
 &= \overline{A_{12} \cdot A_{11}}
 \end{aligned}$$

$$CS = A_{12} \cdot A_{11}$$

$$302: \quad \overline{CS2} = \overline{Y_4} = \overline{A_{12} \cdot A_{11} \cdot A_{10}}$$

$$CS2 = A_{12} \cdot A_{11} \cdot A_{10}$$

$$304: \quad CS = \overline{CS2} \cdot CS1 \cdot CS0 = \overline{A_2} \cdot A_3 \cdot Y_1$$

$$= \overline{A_2} \cdot A_3 \cdot (\overline{A_{12} \cdot A_{11} \cdot A_{10}})$$

$$CS = \overline{A_{12}} \cdot \overline{A_{11}} \cdot A_{10} \cdot A_3 \cdot \overline{A_2}$$

$$306: \quad CS = \overline{CS2} \cdot CS1 \cdot CS0 = \overline{A_3} \cdot A_2 \cdot Y_1$$

$$= \overline{A_3} \cdot A_2 \cdot (\overline{A_{12} \cdot A_{11} \cdot A_{10}})$$

$$CS = \overline{A_{12}} \cdot \overline{A_{11}} \cdot A_{10} \cdot \overline{A_3} \cdot A_2$$

$$307: \quad CS = \overline{CS2} \cdot CS1 \cdot CS0 = \overline{A_2} \cdot \overline{A_3} \cdot Y_1$$

$$= \overline{A_2} \cdot \overline{A_3} \cdot (\overline{A_{12} \cdot A_{11} \cdot A_{10}})$$

$$CS = \overline{A_{12}} \cdot \overline{A_{11}} \cdot A_{10} \cdot \overline{A_3} \cdot A_2$$

Den „reducerede” memory map kan ud fra dette tegnes som:

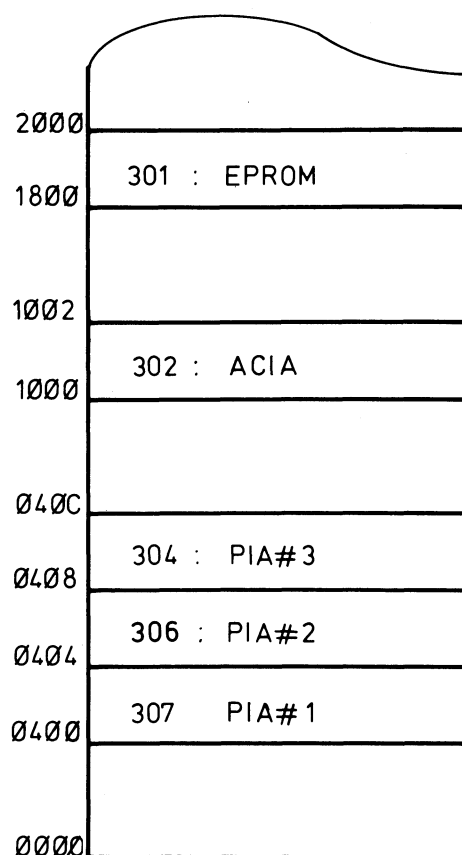


Fig. 7.69.

Som tidligere omtalt under 2716 EPROM anvendt i press 8041, er der ikke 100% adressedekodning til chip select og derfor FYLDER de enkelte kredsløb flere adresser end angivet i fig. 7.69.

Ved hjælp af skemaet i fig. 7.68 kan følgende siges om de enkelte kredsløbs adresser angivet ved 4 hexadecimale cifre:

$$\begin{aligned}
 301: & \left\{ \begin{array}{l} \text{alle} \\ \text{ulige} \end{array} \right\} \cdot \{8 \rightarrow F\} \cdot \{\emptyset \rightarrow F\} \cdot \{\emptyset \rightarrow F\} \\
 302: & \left\{ \begin{array}{l} \text{alle} \\ \text{ulige} \end{array} \right\} \cdot \{\emptyset \rightarrow 3\} \cdot \{\emptyset \rightarrow F\} \cdot \{\emptyset \rightarrow F\} \\
 304: & \left\{ \begin{array}{l} \text{alle} \\ \text{lige} \end{array} \right\} \cdot \{4 \rightarrow 7\} \cdot \{\emptyset \rightarrow F\} \cdot \{8 \rightarrow B\} \\
 306: & \left\{ \begin{array}{l} \text{alle} \\ \text{lige} \end{array} \right\} \cdot \{4 \rightarrow 7\} \cdot \{\emptyset \rightarrow F\} \cdot \{4 \rightarrow 7\} \\
 307: & \left\{ \begin{array}{l} \text{alle} \\ \text{lige} \end{array} \right\} \cdot \{4 \rightarrow 7\} \cdot \{\emptyset \rightarrow F\} \cdot \{\emptyset \rightarrow 3\}
 \end{aligned}$$

Der kan ses gentagelse af et vist mønster i adresseringen.

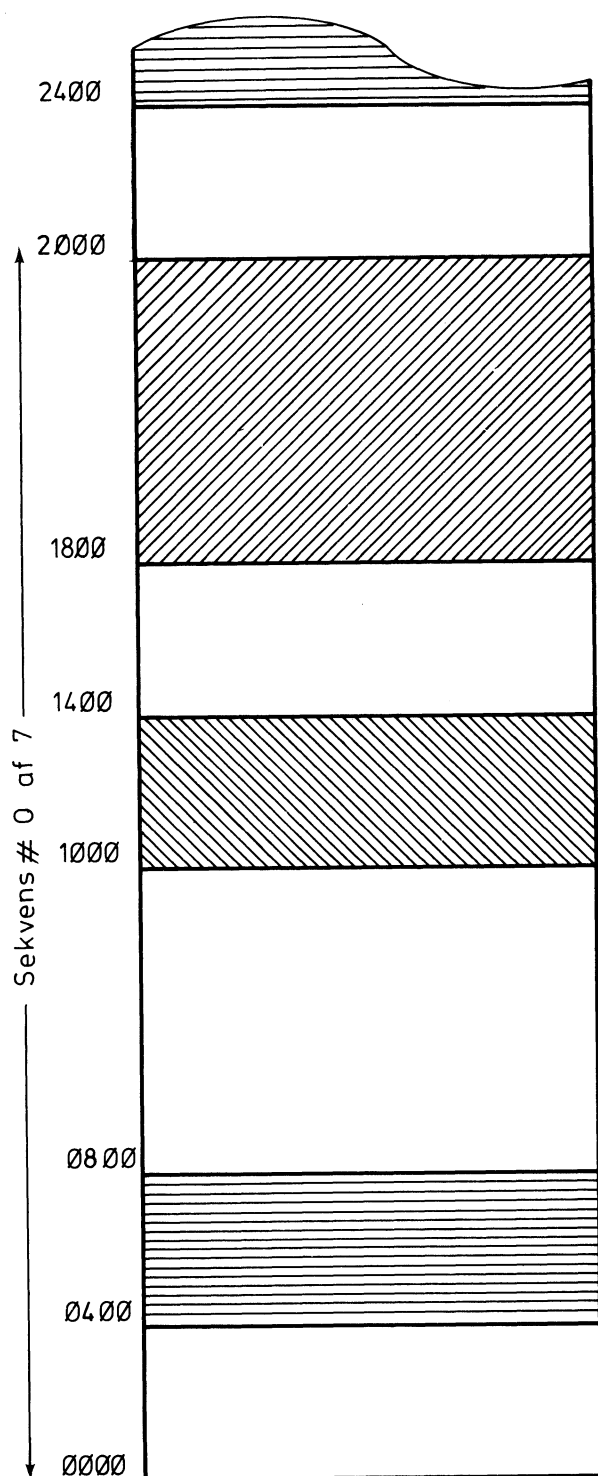
ADR.  
(HEX.)

Sekvens nr.

0000	# 7
E000	# 6
C000	# 5
A000	# 4
8000	# 3
6000	# 2
4000	# 1
2000	# 0
0000	# 0

og sekvens # 0 vist

Fig. 7.70.a

EPROM  
(1800 → 1FFF)ACIA  
(1000 → 13FF)

PIA #3: 0408 → 040B, 0418 → 041B, ..... 07F8 → 07FB  
 PIA #2: 0404 → 0407, 0414 → 0417, ..... 07F4 → 07F7  
 PIA #1: 0400 → 0403, 0410 → 0413, ..... 07F0 → 07F3

Fig. 7.70.b

## ANALOG/DIGITAL OG DIGITAL/ANALOG OM- FORMNING

### D/A OMFORMER

Analog  $\rightarrow$  Digital (A/D) converteren omformer et analogt signal til en repræsentativ digital værdi.

Digital  $\rightarrow$  Analog (D/A) converteren omformer et bit-mønster til en analog strøm- eller spændingsstørrelse.

Her skal et binært tal omformes til f.eks. en analog spænding.

En simpel løsning af problemet består i at generere en spænding for hver bit-position i det binære tal. Spændingens værdi er proportional med bit-værdien (vægten) af det enkelte bit. F.eks. kan bit 0 generere en spænding på  $2^0 E = 1 \cdot E$  volt, bit 1 på  $2^1 E = 2 E$ , bit 2  $\Rightarrow 2^2 E = 4 E$  o.s.v. Den resulterende spænding opnås ved at addere de spændinger, der svarer til de steder, hvor det binære tal har 1-taller.

Resultatet bliver så proportional med det oprindelige binære tal.

En 4-bit D/A converter ses i fig. 7.71. Den består af 4 switche, der styres af det logiske input ( $D_3 \leftarrow D_0$ ), således at et 1-tal lukker og et 0 åbner den enkelte switch. (Analog switch).

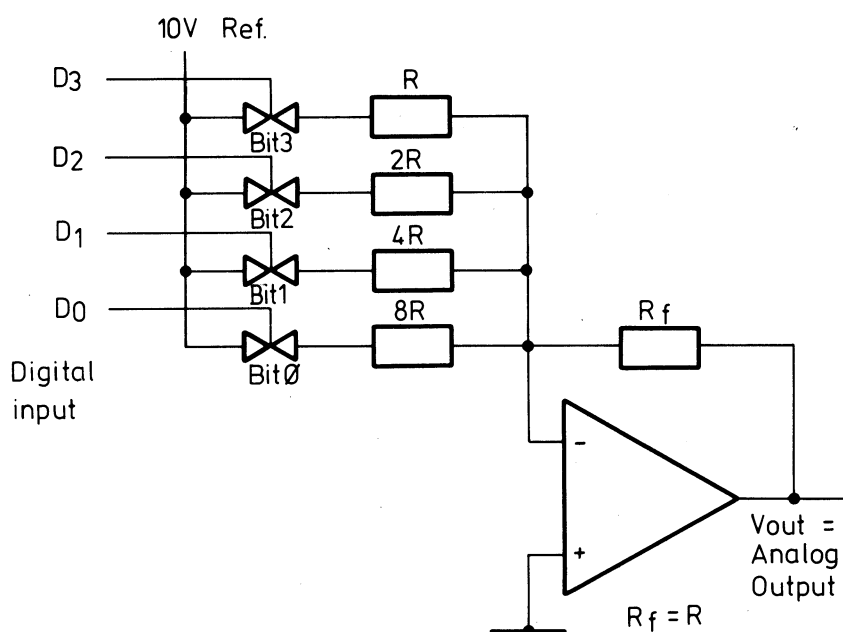


Fig. 7.71.

Operationsforstærkeren er koblet som summationsforstærker med forstærkningen for det enkelte bit bestemt af forholdet mellem LADDERMODSTANDEN (f.eks.  $8 R$ ) og tilbagekoblingsmodstanden ( $R_f$ ).

Med  $D_3 D_2 D_1 D_0 = 1111$ , fås:

$$V_{out} = -V_{REF} \left( \frac{R_f}{8R} + \frac{R_f}{4R} + \frac{R_f}{2R} + \frac{R_f}{R} \right)$$

$$V_{out} = -1.875 \cdot V_{REF}, \text{ idet } R_f = R.$$



Her bliver  $V_{out\ max.} = -1.875 \cdot V_{ref} = -18,75\ V$  for  $D_3 D_2 D_1 D_0 = 11$ . Ændringen forårsaget af LSB angiver OPLØSNINGSEVNEN og kan her beregnes til

$$\frac{1}{8} \cdot V_{REF} = \frac{10}{8} = 1,25\ V.$$

Dette svarer til ca.  $\frac{1}{16}$  af  $V_{out\ max.}$ , idet

$$\frac{18,75}{16} = 1,17\ volt.$$

Da det er lettere at switche strømme end spændinger, benyttes i praktiske monolitiske kredsløb oftest summation af strømme. Bitvægten opnås ved at benytte R – 2 R LADDER NETWORK som vist i fig. 7.72.

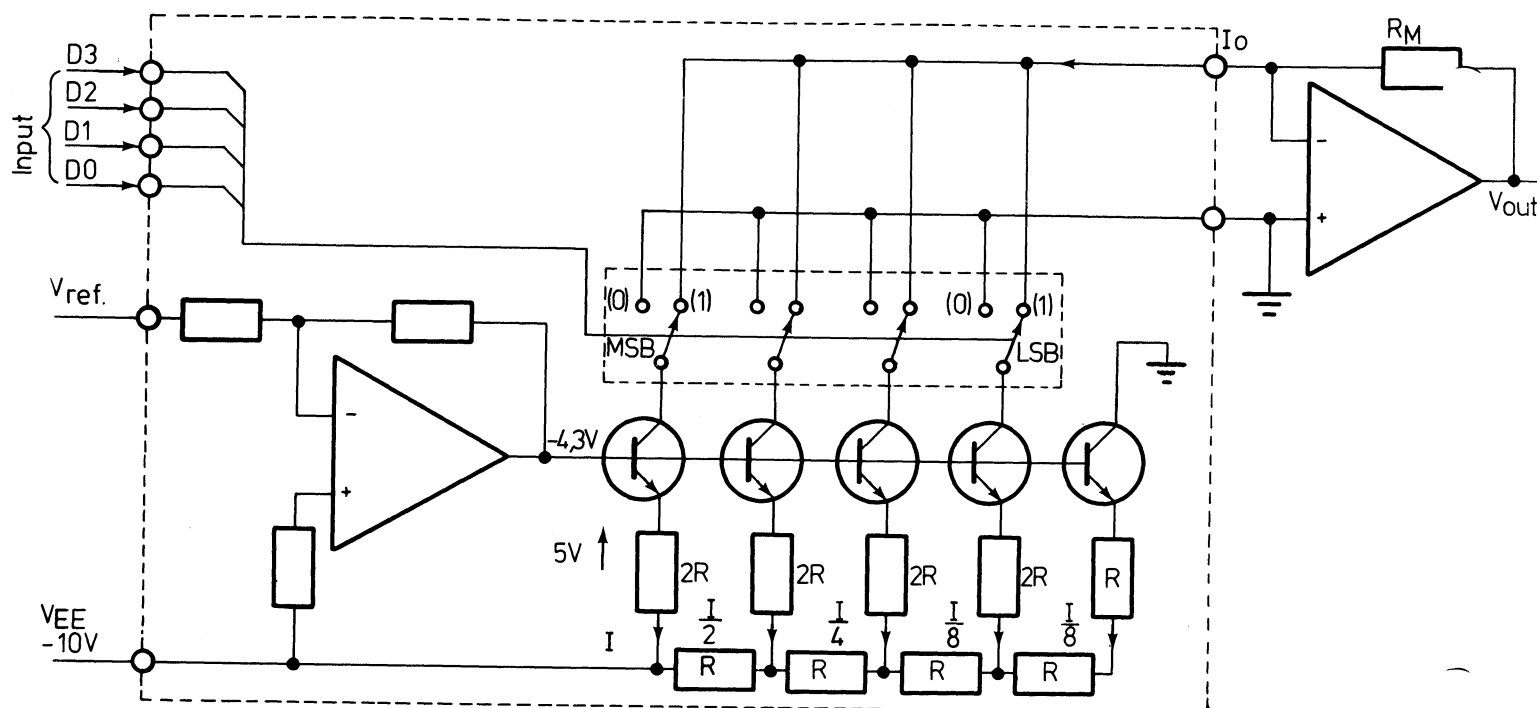


Fig. 7.72.

Med  $V_{REF}$  indstilles biasstrøm til transistorerne. Emitterspændingen er ens for samtlige transistorer og dermed også topspændingen i R – 2 R netværket. I fig. 7.72 tænkes spændingen over emittermodstanden i referencestrømgeneratoren at være på 5 V og sættes R til  $500\ \Omega$ , fås:

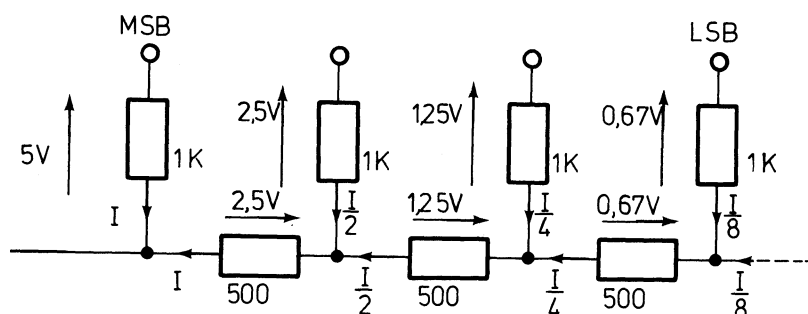


Fig. 7.73.

$$I_{REF} = \frac{5}{10^3} = 5 \text{ mA}$$

$$I = \frac{5}{10^3} = 5 \text{ mA}$$

$U_R = I \cdot R = 5 \cdot 10^{-3} \cdot 500 = 2,5 \text{ V}$  o.s.v.  $\Rightarrow$  de på skitsen viste spændinger (se fig. 7.73).

Strømmen i ladder-network vil være den samme hele tiden. Den binære værdi på input bestemmer stillingen på „omskifterne” i kollektorerne.

Kontakter, der står i stilling „1”, trækker strøm gennem den ydre modstand  $R_m$ .

Heraf fås:

$$V_{out} = -I_0 \cdot R_m$$

hvor

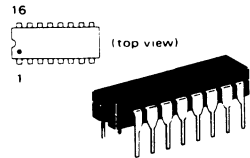
$$I_0 = I \cdot D_3 + \frac{I}{2} \cdot D_2 + \frac{I}{4} \cdot D_1 + \frac{I}{8} \cdot D_0$$

$$I_0 = I_{REF} \left( D_3 + \frac{D_2}{2} + \frac{D_1}{4} + \frac{D_0}{8} \right)$$

idet  $I = I_{REF}$  og  $D$  = binære input værdier.

## MC 1408

Et eksempel på en eksisterende D/A converter findes i Motorola's MC 1408, hvis datablad her er angivet i uddrag.

<div style="display: flex; justify-content: space-between; align-items: center;"> <div style="text-align: center;"> <p><b>MC1508L-8</b></p> <p><b>MC1408L-8</b></p> <p><b>MC1408L-7</b></p> <p><b>MC1408L-6</b></p> </div> <div style="text-align: right;"> <p><b>D-TO-A CONVERTER</b></p> </div> </div>	
<p style="text-align: center;"><b>Specifications and Applications Information</b></p> <div style="border: 1px solid black; padding: 5px; margin: 5px 0;"> <p style="text-align: center;"><b>EIGHT-BIT MULTIPLYING DIGITAL-TO-ANALOG CONVERTER</b></p> <p>... designed for use where the output current is a linear product of an eight-bit digital word and an analog input voltage.</p> <ul style="list-style-type: none"> <li>Relative Accuracy: <math>\pm 0.19\%</math> Error maximum (MC1508L-8, MC1408L-8)</li> <li>Seven and Six-Bit Accuracy Available (MC1408L-7, MC1408L-6)</li> <li>Fast Settling Time – 300 ns typical</li> <li>Noninverting Digital Inputs are MTTL and CMOS Compatible</li> <li>Output Voltage Swing – +0.5 V to -5.0 V</li> <li>High-Speed Multiplying Input Slew Rate 4.0 mA/<math>\mu</math>s</li> <li>Standard Supply Voltages: +5.0 V and -5.0 V to -15 V</li> </ul> </div>	<div style="border: 1px solid black; padding: 5px; margin: 5px 0;"> <p style="text-align: center;"><b>EIGHT-BIT MULTIPLYING DIGITAL-TO-ANALOG CONVERTER</b></p> <p style="text-align: center;"><b>SILICON MONOLITHIC INTEGRATED CIRCUIT</b></p> </div> <div style="border: 1px solid black; padding: 5px; margin: 5px 0;">  <p style="text-align: center; font-size: small;">L SUFFIX CERAMIC PACKAGE CASE 620</p> </div>



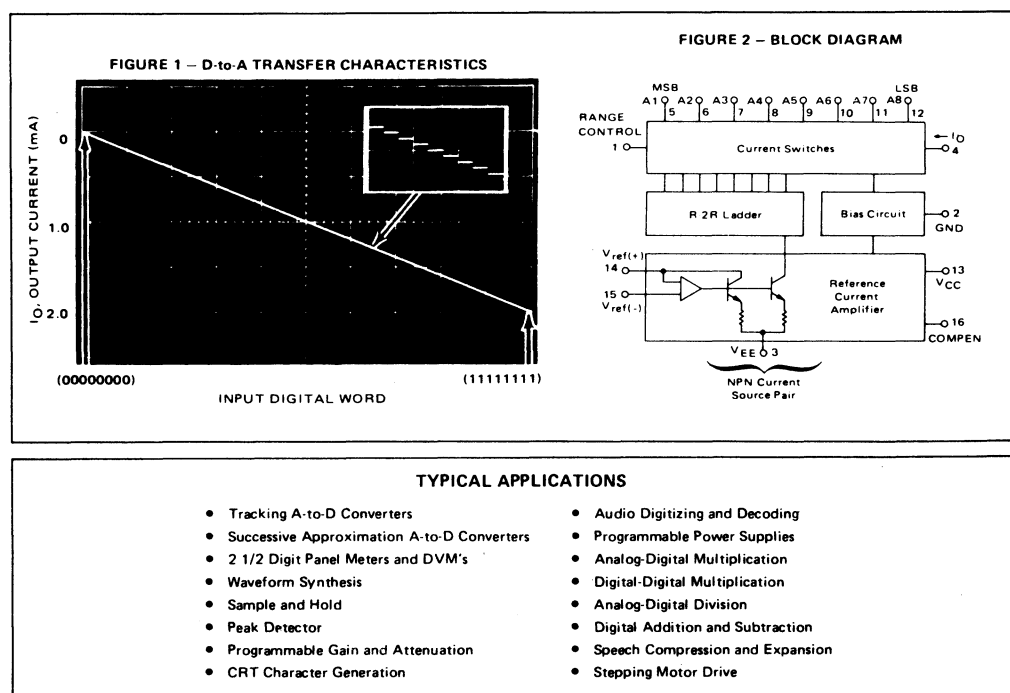


Fig. 7.74. (AA)

Et anvendelseseksempel er angivet i fig. 7.75 fra databladets app. notes:

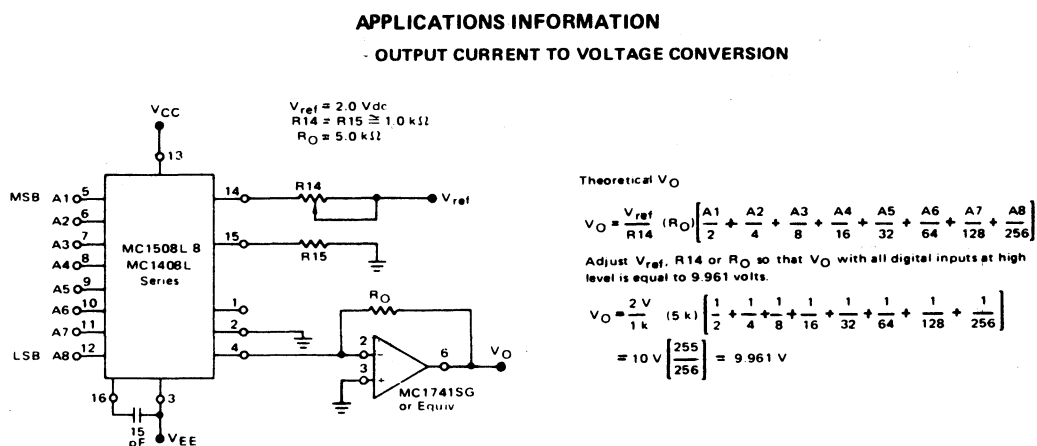


Fig. 7.75. (AA)

## A/D OMFORMER

I omformning fra et analogt signal til en digital information (binært tal) benyttes oftest en af følgende metoder:

1. successive approximation
2. integration
3. Direct Comparison.

### SUCCESSIVE APPROXIMATION

A/D omformningen kan udføres ved at benytte D/A konverteren omtalt tidligere.

Det ukendte ANALOG INPUT sammenlignes med et „GÆT” udført af D/A-konverteren og sammenlignes i en komparator, der beslutter, om den digitale information, der udgør næste „GÆT” til D/A, skal forøges eller formindskes.

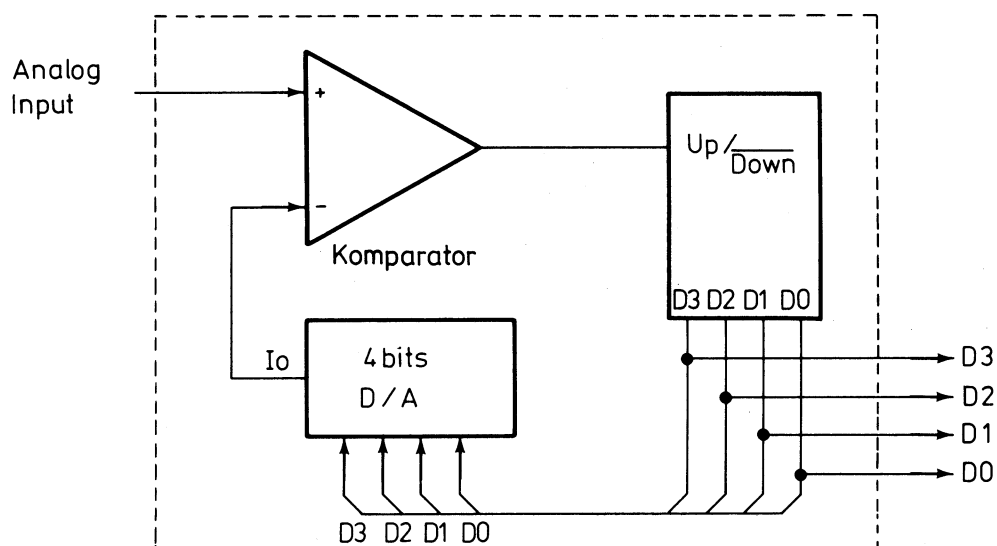


Fig. 7.76.

Gætningen foretages ved at sætte MSB på 1 og se, om output fra D/A er større eller mindre end det analoge input. Hvis D/A output er mindre end analogt input, beholdes 1-tallet, men er D/A output blevet større end analogt input, sættes dette bit til 0 igen.

Herefter går til det næste bit i rækken og samme undersøgelse foretages. Dette gentages indtil LSB nås i talrækken. Fig. 7.77 viser forløbet af D/A output i forhold til tiden og ANALOG INPUT er stiplede som reference:

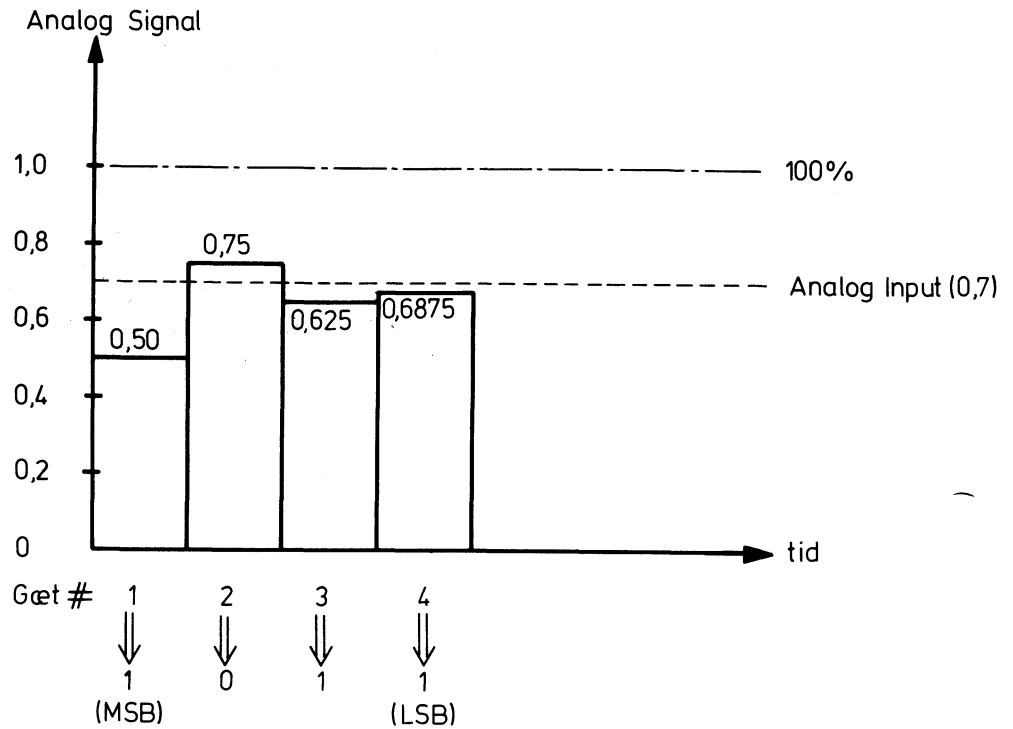


Fig. 7.77.

Metoden, der anvendes, betegnes **SUCCESSIVE-APPROXIMATION ANALOG TO DIGITAL CONVERSION** og er den mest anvendte metode. Ved anvendelsen skal der foretages lige så mange gæt, som der er antal bit i det digitale output. I praksis kan skiftefunktionen udføres ved hjælp af hardware som f.eks. vist i fig. 7.78.

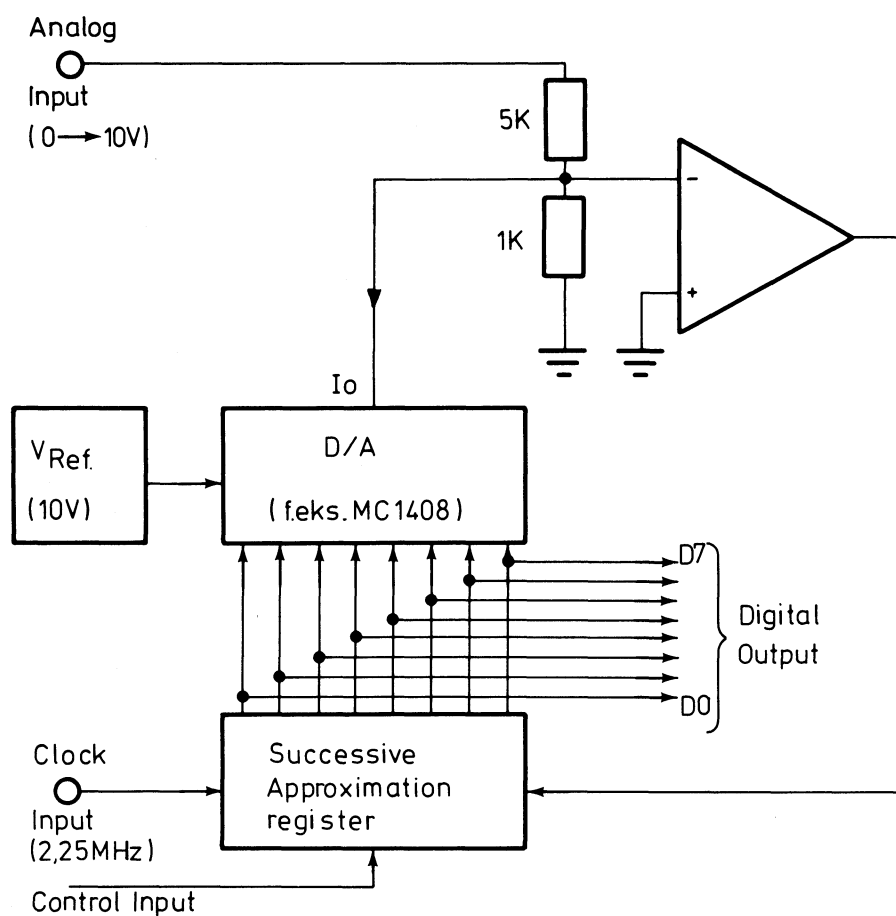
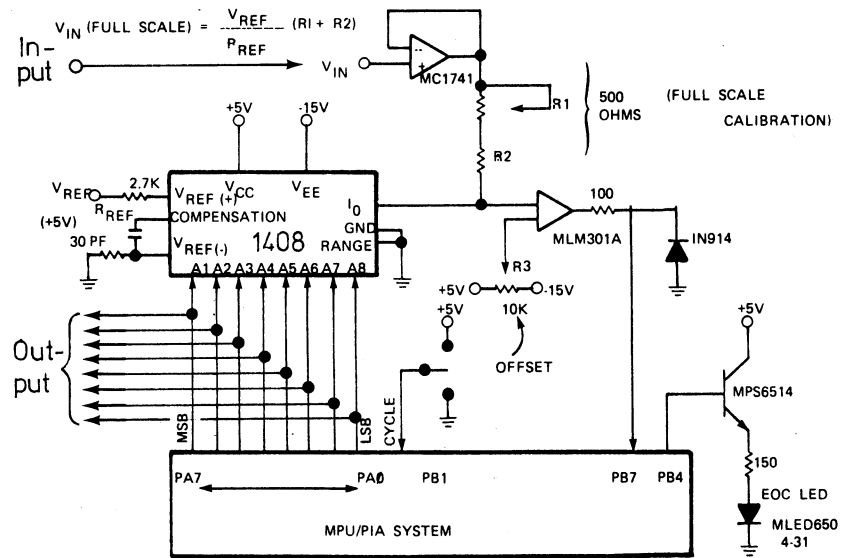


Fig. 7.78.

Her foretages bit-skift og styring i enheden benævnt Successive Approximation Register (SAR).

Løsningen kan også foretages som en blandet H.W. & S.W. løsning f.eks. som vist i fig. 7.79, hvor det er PIA 6821, der sammen med MPU'en udgør SAR-enheden.



Using a D/A for Successive Approximation A/D

Fig. 7.79. (AA)

Successive approximation metoden har middelgod konverterings-hastighed og -nøjagtighed. Skal hastigheden være større og kan det tillades, at nøjagtigheden bliver mindre, kan DIRECT COM-PARISON metoden benyttes.

Skal derimod nøjagtigheden være stor og hastigheden kan tillades lille, benyttes INTEGRATIONSMETODEN (f.eks. DUAL SLOPE).

Når der tales om UDVIKLINGSUDSTYR til microcomputere, tænkes der næsten altid på SOFTWARE UDVIKLING.

Hovedparten af softwaren til en microcomputer er det PROGRAM, der ligger i ROM. Udviklingen og tildels også afprøvning og fejlfinding i SOFTWAREN, bør overlades til personer, der har speciel indsigt i PROGRAMMERING og FEJLFINDING på den type microprocessor, der er anvendt.

Til hjælp ved PROGRAMUDVIKLINGEN og til den senere AFPRØVNING og FEJLFINDING (TEST and DEBUG) kan „programmøren” betjene sig af forskellige former for udstyr.

## UDVIKLINGSUDSTYR

Til selve programmets udvikling og afprøvning anvendes UDVIKLINGSUDSTYR, der kan spænde lige fra de billigste EVALUATION KITS (pris ca. kr. 1.000,- → 3.000,-) til de største udviklingsudstyr specielt fremstillet til en bestemt microprocessor (pris ca. kr. 100.000,-) og til de avancerede minicomputeranlæg, hvor der kan anskaffes software til de enkelte microprocessortyper. Alt efter udbygningen vil disse anlæg ligge i prisleje fra 100.000,- til flere millioner kroner.

## EVALUATION KITS

De simpleste udviklingsudstyr koster omkring kr. 2.000,- og består som regel af et enkelt printkort med følgende enheder monteret (evt. leveret som samlesæt):

CPU'en  
1-2k bytes ROM m. MONITORPROGRAM  
256 bytes RAM  
1 a 2 stk. parallel I/O enheder  
1 stk. seriel I/O enhed  
Hexadecimalt tastatur  
7-segment display

og diverse interfacekredsløb, der skal være med for at få minimum-systemet til at virke.

Systemerne kan ofte udvides med op til 1k bytes RAM og evt. også 1-2k bytes EPROM, men herudover ikke mere.

Systemerne er udmærkede til oplæring i grundlæggende programmering og grundlæggende hardwarekendskab. Desuden kan de benyttes som microcomputerløsning ved mindre styrings- og reguleringsopgaver, selvom denne anvendelse sjældent ses med udviklingsudstyret, men derimod ofte vil blive løst med single board computer.

Fordelene ved EVALUATION KITS er den lave anskaffelsespris. Herved opnås, at mange kan få lejlighed til at sætte sig ind i problematikken ved programmering og anvendelse af microcomputere, uden at skulle investere store beløb.

Ulempen er den simple og langsommelige programmering i MASKINKODE (HEXADECIMAL OBJECT CODE) og de besværligheder, dette medfører, når programmerne bliver større end ca. 50 STATEMENTS.



I forbindelse med EVALUATION KITS ses ofte tilslutningsmulighed til ydre enheder som BÅNDOPTAGER og/eller TELETYPE (TTY).

## MELLEMSTORE UD- VIKLINGSUDSTYR

For en pris på 20.000,- til 40.000,- kroner kan man anskaffe sig et udviklingsudstyr, der er i stand til at udføre programudvikling af lidt større programmer (50 → 1000 statements).

Grundudgaven kan typisk bestå af CPU-kort, ROM kort med MONITOR, RESIDENT ASSEMBLER og -EDITOR, 8k eller 16k RAM kort og diverse I/O og interface kort.

Systemet kan ofte tilsluttes følgende terminalkonfigurationer:

ASCII keyboard og TV skærm  
CRT terminal  
TTY terminal  
Båndoptager

og kan desuden ofte udvides med ekstra RAM lager og tilsluttes floppy disk ved anskaffelse af de ekstra printkort og enheder, dette kræver.

## STØRRE UDVIKLINGS- UDSTYR

Af større udviklingsudstyr findes to hovedgrupper:

1. Microcomputerbaseret specialudstyr for bestemt processor
2. Microcomputer med software specielt udviklet til den enkelte microprocessor.

For type 1 gælder det, at de enkelte microcomputerfabrikanter ofte leverer „DISK OPERATIV SYSTEMER” for deres respektive microcomputere og disse består ofte af:

CPU enhed  
ROM enhed (monitor)  
RAM enhed  
I/O enheder  
Interface enheder

alle samlet i „MAIN FRAME”. Desuden benyttes oftest

CRT terminal  
Floppy disk unit  
Linie printer

som eksterne enheder og software til systemet findes på floppy disk.

Som options til disse systemer fås ofte

IN CIRCUIT EMULATOR (ICE)  
REAL TIME PROTOTYPE ANALYZER  
MACRO ASSEMBLER  
BASIC, FORTRAN og/eller PASCAL COMPILER

Prisen for grundsystemet med CRT og Floppy er i 1979 ca. 100.000 → 200.000,- kr. og alle udbygninger (udvidelser) af systemet er relativt kostbare. Med et sådant system vil man være i stand til at udvikle,

afprøve og fejlfinde et microcomputersystem på en effektiv og rationel måde.

Da SOFTWARE omkostningerne er de dominerende idag og fremover vil udgøre en større og større procentdel af udviklingsomkostningerne, vil det være mere og mere påkrævet, at de store udviklingssystemer anvendes, selv i mindre virksomheder, når disse baserer deres produktion på anvendelse af microcomputere.

Ved type 2 af de større udviklingssystemer er grundlaget normalt, at der i firmaet findes en større computer i forvejen eller der er adgang til en større computer på TIME SHARING basis.

De større firmaer indenfor microprocessorer kan levere CROSS-ASSEMBLER og CROSS COMPILERE til de mest almindelige minicomputere. Med denne software vil man blive i stand til at udvikle programmerne til sine microcomputere og ofte kan disse programmer afprøves i en SIMULATOR på den samme minicomputer.

Derimod vil der næsten aldrig kunne foretages EMULERING eller REAL TIME PROTOTYPE TEST på en minicomputer, hvilket jo nok i fremtiden vil bevirke, at type 1 foretrækkes frem for type 2.

## TESTING

Hvad gør man, når det ikke virker? Hvad var det, der gik galt? og hvorfor? DEBUGGING processing, også kendt som afprøvning og fejlfinding, er en integreret del af ethvert system design. Murphy's lov holdes altid: „Hvis noget kan gå galt, vil det også ske”.

Når man står overfor et system, der ikke opfører sig korrekt, er der forskellige mulige teknikker til rådighed til at identificere og rette problemet.

I det følgende vil årsagen til alm. problemer samt deres løsning blive behandlet. Det drejer sig om problemer som: Komponentfejl, softwarefejl og støjinducerede fejl.

De værktøjer, der anvendes til at identificere og lokalisere disse problemer, vil også blive beskrevet. Det gælder Universalinstrument, Logiske Prober, Signaturanalysatorer, Oscilloskoper, Digitale Analyzere, Simulatorer, Emulatorer og In Circuit Emulatorer.

## HVAD GÅR GALT?

Fire væsentlige problemer kan opstå i et system:

1. Forbindelsesfejl: en kortslutning eller en afbrydelse
2. Komponentfejl: incl. forkert komponentværdi
3. Software „lus”:
4. Støj og interferens: enten INTERN eller EXTERN

## FORBINDELSSESFEJL

Forbindelsesfejl konstateres ved en modstandsmåling fra punkt til punkt i systemet. Afprøv hver eneste ledning. Vær sikker på, at den går til den rigtige forbindelse og ikke til andre på det integrerede kredsløb. Vær sikker på, at kredsløbsforbindelserne er dobbelt-kontrolleret. Vær ikke sikker på, at tegningerne er rigtige, før systemet arbejder.

FORBINDELSSESFEJL er de mest almindelige og besværlige problemer. De kan let løses – men de tager megen tid.

## KOMPONENTFEJL

Komponenter som modstande, kondensatorer, spoler, transformatorer, transistorer, dioder, integrerede kredsløb, stikforbindelser og printkort kan alle udvise fejl. Modstande afbryder, kapaciteter lækker. Kort sagt: ingen komponenter er ideelle. Alt fejler før eller siden.

Hver komponent kan gives et KVALITETSTAL, kendt som MEAN TIME BETWEEN FAILURES eller MTBF. Dette er en statistisk forudsigelse af, hvor lang tid (i timer) en komponent vil holde i et givet milieu.

I fig. 8.1 vises en tabel med fejlhyppighed angivet i procent pr. 1000 timer gældende for militært flyudstyr.

Komponent	Fejlhyppighed ( $\lambda$ ) (% pr. 1000 timer)
1. Kapacitet	0,02
2. Modstand	0,002
3. Variabel modstand	0,01
4. Transformator	0,5
5. Diode	0,013
6. Integreret kredsløb (SSI, MSI, LSI)	0,015
7. Transistor	0,04
9. Quartz krystal	0,05
10. Lodning	0,0002
11. Wire-wrapning	0,00002

Fig. 8.1

Tabellen angiver middelværdier og det er en forudsætning, at komponenterne anvendes korrekt i forhold til specifikationer fra fabrikanterne.

Kendes fejlhyppigheden hos de enkelte komponenter i et system kan det give fejlhyppigheden på hele systemet. Dette opnås ved at addere alle komponenternes fejlhyppighed.

Middeltid mellem fejl findes som det reciprokke af fejlhyppigheden

$$\text{Middeltid} = \frac{1}{\text{fejlhyppighed}}$$

$$\text{MTBF} = \frac{1}{\lambda}$$

I et system kan der for et enkelt PC-kort beregnes, idet de angivne komponenter befinder sig på dette kort:

4 stk. IC $\Rightarrow 4 \times 0,015$	= 0,06
1 stk. krystal	= 0,05
10 stk. modst. $\Rightarrow 10 \times 0,002$	= 0,02
10 stk. kapaciteter $\Rightarrow 10 \times 0,02$	= 0,2
1 stk. transformer	= 0,5
4 stk. dioder $\Rightarrow 4 \times 0,013$	= 0,05
1 stk. PC med 10 connectors og 500 loddepunkter $\Rightarrow 10 \times$ $10 \times 0,005 + 500 \times 0,0002$	= 0,15

$$\lambda \text{ total pr. 1000 timer} = \underline{\underline{1,03\%}}$$

$$\text{MTBF} = \frac{1}{\lambda_{\text{tot}}} = \frac{1}{\frac{1,03}{100} \cdot \frac{1}{1000}}$$

$$= 9,7 \cdot 10^4 \text{ timer}$$

$$\text{MTBF} = \underline{\underline{97.000 \text{ timer}}}$$

Antages det, at der udføres 1000 stk. af denne enhed og disse anvendes på den specificerede måde, så vil der efter 1000 timers drift være 10 stk., der har fejlet og efter 10.000 timer være 100, der har udvist fejl.

Hvor ofte sker der fejl? Dette enkle spørgsmål er besvaret som en middelværdi i det foregående, men fortæller intet om FORDELINGEN af fejlene. De fleste systemer følger den i fig. 8.2 viste „livsfunktion“:

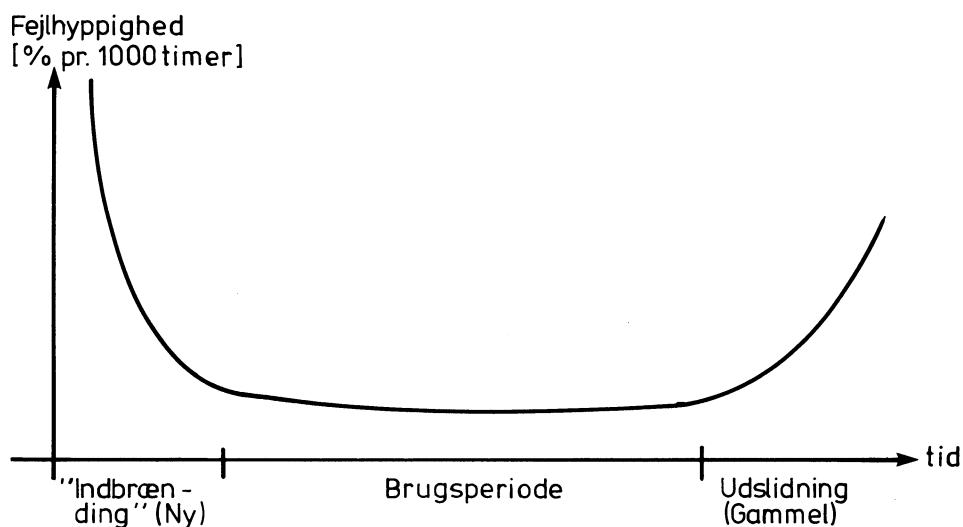


Fig. 8.2

Initialfejlene søges undgået ved at udføre en „indbrændingstest“ af udstyret hos fabrikanten, inden det sendes til forbrugerne.

Anvendelsen af udstyret har stor betydning ved beregning af MTBF.

F.eks. vil et bestemt system anvendt i underholdningsindustrien kunne udvise en MTBF på flere år, medens det samme anvendt i f.eks. et luftfartøj kunne give en MTBF på 0,1 time! Dette er tilfældet, fordi de to formål opstiller FORSKELLIGE FEJLHYPPIGHEDSKRAV og derved arbejder med forskellige  $\lambda$ -værdier for de samme komponenter.

## SOFTWAREFEJL

Software (programmet) kan også være behæftet med fejl. En fejl kan være „skjult” i en programdel, der sjældent benyttes og dette kan betyde, at udstyret i lang tid ser ud til at fungere perfekt. Pludselig kan der på grund af softwarefejl i en bestemt rutine opstå fejl flere gange i løbet af kort tid.

SOFTWARE-FEJL eller „LUS” (BUGs) er ofte de sværeste at finde. Denne fejltype er desværre den oftest forekomne i microcomputer-systemer, idet intet program er perfekt.

Et program er begrænset i præcision, hastighed og fleksibilitet. Den dygtige programmør er en uovertruffen pessimist med hensyn til sin software, indtil den har kørt i et par år.

## STØJFEJL

Støj findes overalt. Hvorsomhelst, der er strøm i en ledning, er der et elektromagnetisk felt. Derfor er der felter fra transformatorer, motorer og ledninger overalt. Hertil kommer, at enhver ledning vil virke som antenne for radio-, TV- og amatørstationernes signaler. Man skal huske, at støj ikke alene kan komme fra de ydre omgivelser, men også kan genereres inde i selve systemet.

Der kan gives nogle eksempler:

1. Når IC'er skiftes fra et logisk niveau til det modsatte, genereres små strømændringer på grund af deres ændring i strømforbrug. Hvis for mange kredsløb skifter samtidig, kan det påvirke forsyningsspændingen så meget, at det får indvirkning på andre kredsløb. For at undgå denne type af støj benyttes AFKOB-LINGSKAPACITETER ved hvert af de integrerede kredsløb.
2. Hvis to ledninger ligger tæt sammen, vil en puls, der udsendes på den ene, kunne induceres over på den anden leder på grund af transformatorvirkningen mellem de to ledere. Den inducerede puls kan herved påvirke kredsløb til at skifte eller ændre data, så de ikke længere er korrekte. For at undgå denne virkning snos lederpar og beskyttes med skærm.
3. Støjproblemer opstår ofte i forbindelse med brum fra DC-forsyningen. Ved stor belastning af ensretterkredsløb kan ripple på forsyningen forårsage f.eks. uønsket læsning – eller skrivning – i RAM lageret.

Andre problemer er SWITCH transienter fra NET-FORSYNINGEN. Ind- og udkobling af relæer og motorer (f.eks. en TTY) kan give spikes på netledningen på flere tusinde volt. For at undgå disse støjproblemer bør der anvendes NET-FILTRE og SKÆRMEDE POWER TRAFO'er.

## VÆRKTØJER TIL PROBLEMLØSNING

I det følgende vil der blive omtalt, hvilke værktøjer, der er til rådighed for løsningen af microcomputer-problemer og hvilken type problemer der kan løses med de enkelte værktøjer.

Fig. 8.4 viser summarisk sammenhængen mellem værktøjerne og de problemer, disse kan løse. I fig. 8.3 er der vist den relative tid, det vil tage at løse et problem med forskellige sammensætninger af værktøjerne:

PROBLEMETS LØSNING	Kræver mindst følgende udstyr:
TIDSKRÆVENDE	UNIVERSALINSTRUMENT LOGISKE PROBER OSCILLOSKOP
MIDDELTID	UNIVERSALINSTRUMENT OSCILLOSKOP DIGITAL DOMAIN ANALYSATOR
HURTIGSTE MÅDE	UNIVERSALINSTRUMENT OSCILLOSKOP DIGITAL DOMAIN ANALYSATOR IN CIRCUIT EMULATOR

Fig. 8.3

PROBLEMER DER KAN LØSES	UDSTYR TIL RÅDIGHED:						
	1	2	3	4	5	6	7
	UNIVERSAL- INSTRU- MENT	LOGISKE PROBER	SIGNATUR ANALYSA- TOR	OSCILLO- SKOP	DIGITAL DOMAIN ANALYSA- TOR	IN CIRCUIT EMULATOR (ICE)	SOFTWARE EMULATOR (SIMULATOR)
KORTSLUTNING AFBRYDELSE FORKERT SPÆNDING	JA	MÅSKE	NEJ	JA	MÅSKE	MÅSKE	NEJ
DÅRLIG MODSTAND EL. KAPACITET o.s.v.	JA	NEJ	NEJ	JA	NEJ	NEJ	NEJ
UKENDTE LOGISKE SIGNA- LER NÅR FORKERT PRO- GRAMFORLØB ER SKET	JA	JA	JA	JA	JA	NEJ	NEJ
UKENDTE LOGISKE SIGNA- LER NÅR DER KAN SKE FORKERT PROGRAM- FORLØB	JA (TIDSKRÆVENDE)	JA	NEJ	JA (TIDSKRÆ- VENDE)	JA	JA	NEJ
SOFTWAREPROBLEMET	NEJ	NEJ	NEJ	MÅSKE	JA	JA	JA

Fig. 8.4

Anvendelse af UNIVERSALINSTRUMENTET og OSCILLOSKOPET vil ikke blive behandlet her, idet disse instrumenter og deres anvendelse forudsættes kendte.

Simple problemer som manglende forsyningsspænding, kortslutninger og afbrydelser er de hyppigste årsager til, at et system overhovedet ikke „kører”. Denne STATISKE fejltipe findes hurtigst ved hjælp af universalinstrument og/eller oscilloskop.

På samme måde findes også de periodiske fejl, der ofte skyldes stikforbindelserne (og sokler) eller dårlige lodninger, lettest ved at anvende oscilloskop evt. af STORAGE TYPEN.

Er der mistanke om fejl i integrerede kredsløb, kan disse afprøves ved at blive placeret i en REFERENCE-ENHED, som man ved er i orden med det oprindelige kredsløb. Virkningen efter indplacering af det mistænkte kredsløb vil afgøre, om mistanken er berettiget.

## KONSTRUKTIONS- PROBLEMER

Konstruktionsfejl kan deles i to hovedgrupper:

1. Forkert anvendelse
2. Forkerte specifikationer

som kan illustreres med et par eksempler.

### Forkert anvendelse:

Sendes der for meget strøm gennem en modstand, vil den brænde op. Tilføres der for høj spænding til en kapacitet, vil den kortslutte. Enhver komponent har sin begrænsning. Problemet med FOR MEGET er det mest almindelige problem. F.eks. for mange belastninger tilsluttet en dataledning kan forårsage, at der læses „forkerte” data.

### Forkerte specifikationer:

Hvis man tror, at et kredsløb er i stand til at drive 30 belastningsenheder, men det kun kan drive 20, er der tale om FORKERTE SPECIFIKATIONER. Der er simpelthen ikke set i databladet, hvad specifikationerne for kredsløbet var.

Mere underfundigt er det, hvis TIMING af en bestemt kreds misforstås. F.eks. hvis adressen til en hukommelseskreds skal være stabil 20 nanosek. før DATA og WRITE ankommer. Dette kan være overset, således at DATA og WRITE ankommer før eller samtidig med adressen, og dette vil føre til forkert virkemåde.

Konstruktionsproblemer kræver hele spektret af udstyr til passende fejlfinding, men universalinstrument og oscilloskop vil være tilstrækkelig, hvis tiden ikke er af betydning.

Ved overbelastning af bus-ledninger ses fejlen som INTERMITTERENDE (periodiske) FEJL og ved OVERSPÆNDING og -STRØM som brændende og rygende komponenter. Disse sidste problemer med ild og røg løses lettest — der skal foretages nyt design med større enheder eller mindre strøm/spænding.

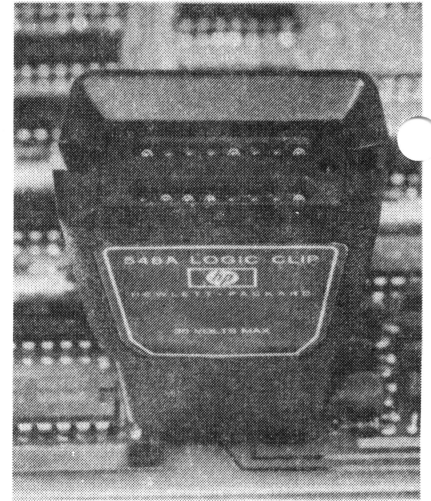
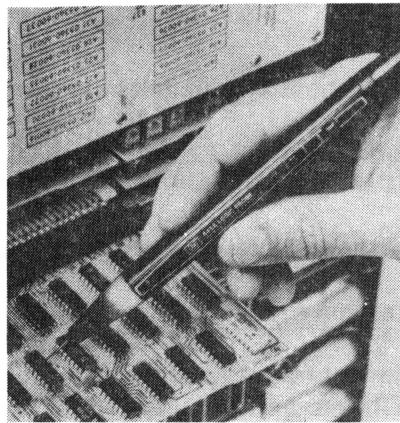
De periodiske problemer kræver kontrol af databladene for de enkelte kredsløb for at finde eventuelle belastningsfejl både med



hensyn til output og input på kredsløbene. Der kan evt. anvendes varierende temperaturer for systemet for at finde de komponenter, der optræder med „periodiske” fejl, idet disse temperaturvariationer ofte får udløst den periodiske tilstand.

## LOGISKE PROBER

Logiske prober anvendes til at vise de logiske niveauer på en nem og hurtig måde. Proberne kan vise, om et punkt er logisk 0 eller 1 eller er i en ubestemt tilstand ved at forskellige LED's i proben lyser. Ved fejlfinding skal man især være opmærksom på den ubestemte tilstand, som kun kan forekomme på tri-state ledninger. Er der ikke tale om tri-state vil en UBESTEMT TILSTAND ofte være et tegn på fejl i systemet. Fig. 8.5 viser forskellige logiske prober.



# LOGIC PROBES    LOGIC CLIPS

Fig. 8.5 

## DYNAMISKE PROBLEMER

Ved drift virker systemet ikke! Universalinstrument og logiske prober kan ikke vise tidsfølge og er derfor af mindre værdi ved undersøgelse af dynamiske problemer.

Vi behøver udstyr, der kan vise de logiske niveauers tidsforløb korrekt.

Det mest anvendte instrument i denne situation er oscilloskopet, men ved microcomputerudstyr kan det være meget tidskrævende at skulle fejlfinde i dynamiske forhold med et oscilloskop.

## LOGIC ANALYZER

Til microcomputere er der udviklet specielle måleinstrumenter. Et af de mest generelle i anvendelse er LOGIC ANALYZER eller DIGITAL DOMAIN ANALYZER, som den egentlig burde kaldes.

Med DIGITAL DOMAIN ANALYZERs kan man se på op til 32 signalledninger samtidig, og de fleste af disse analysatorer er indrettet således, at man frit kan vælge at få præsenteret et af følgende displays:

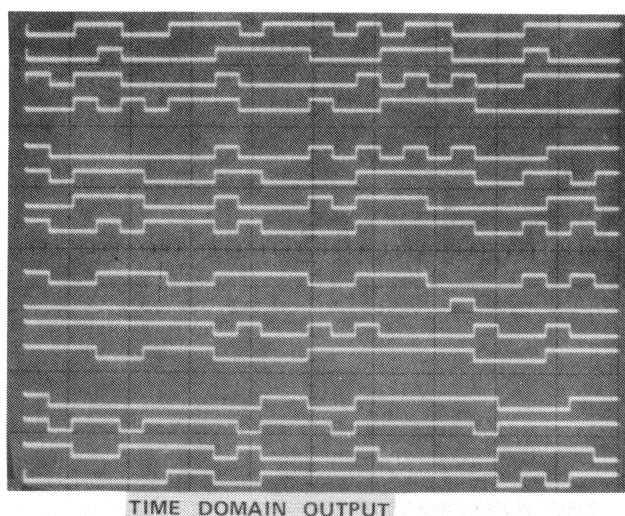


Fig. 8.6

115	1011	0100	0111	1101	B47D
116	0000	0000	0100	1100	004C
117	0110	1101	0010	1100	6D2C
118	1110	0111	0010	0101	E725
119	0110	1000	0100	0101	6845
120	1000	0000	0110	1101	806D
121	1000	1001	1110	1100	89EC
122	1000	1001	1110	1100	89EC
123	0111	1111	0010	0001	7F21
124	1010	0010	0100	0101	A245
125	0000	0011	1011	0001	03B1
126	0000	0011	1011	0001	03B1
127	0101	1001	1010	1100	59AC
128	0000	0000	1000	1000	0088
129	1111	0101	1111	1101	F5FD
130	1110	1010	1011	1001	EAB9

DATA DOMAIN PRESENTATION

Fig. 8.7

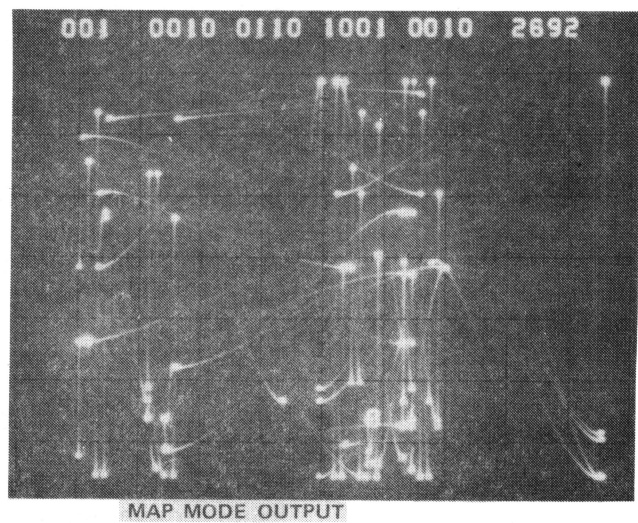
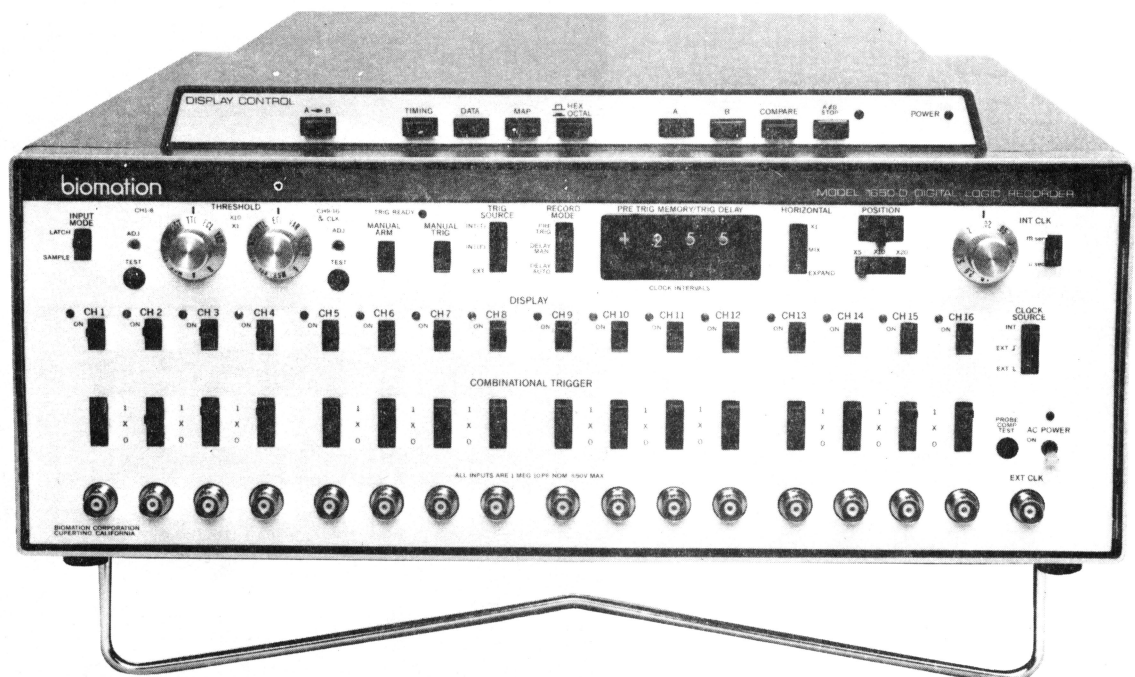


Fig. 8.8

Som eksempel på en analyser, der tilsluttet et X-Y-skop kan give de foregående billeder, ses fig. 8.9 og 8.10.

# biomation



THE NEW MODEL 116 DISPLAY CONTROL ACCESSORY PROVIDES SELECTABLE OUTPUT DISPLAY FORMATTING FOR THE MODEL 1650-D LOGIC ANALYZER.

Fig. 8.9

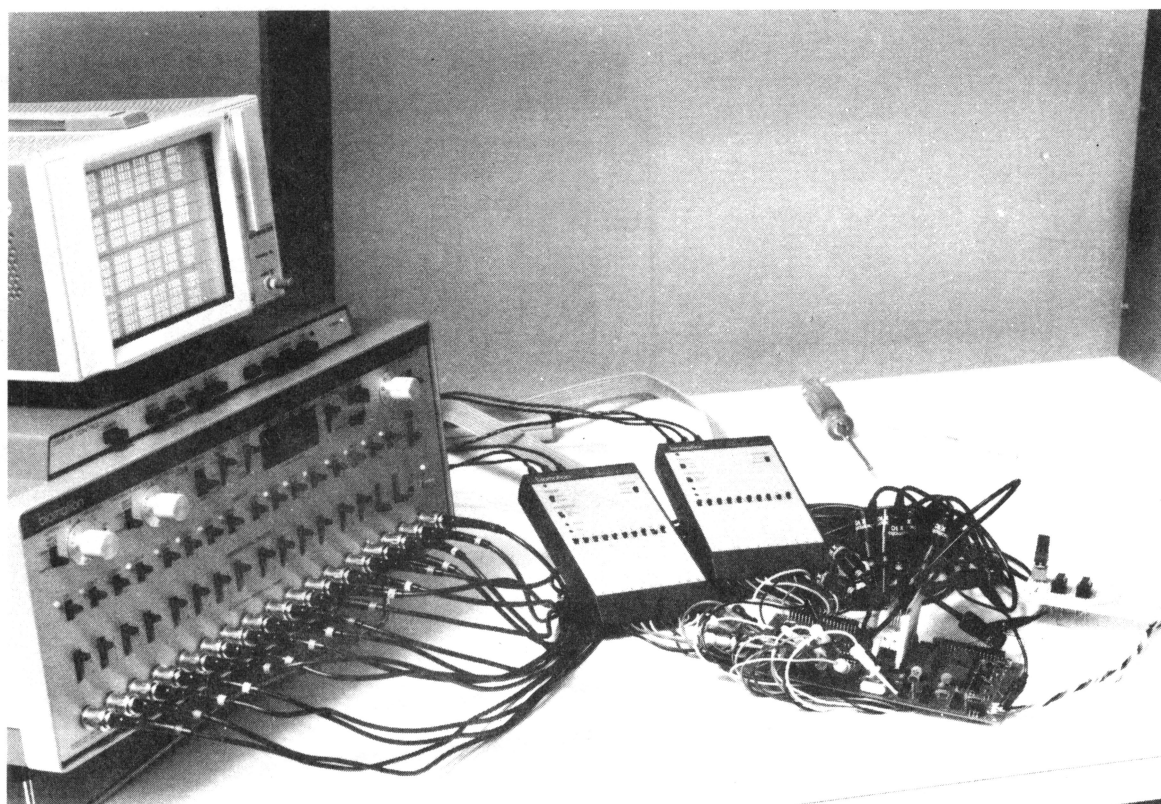


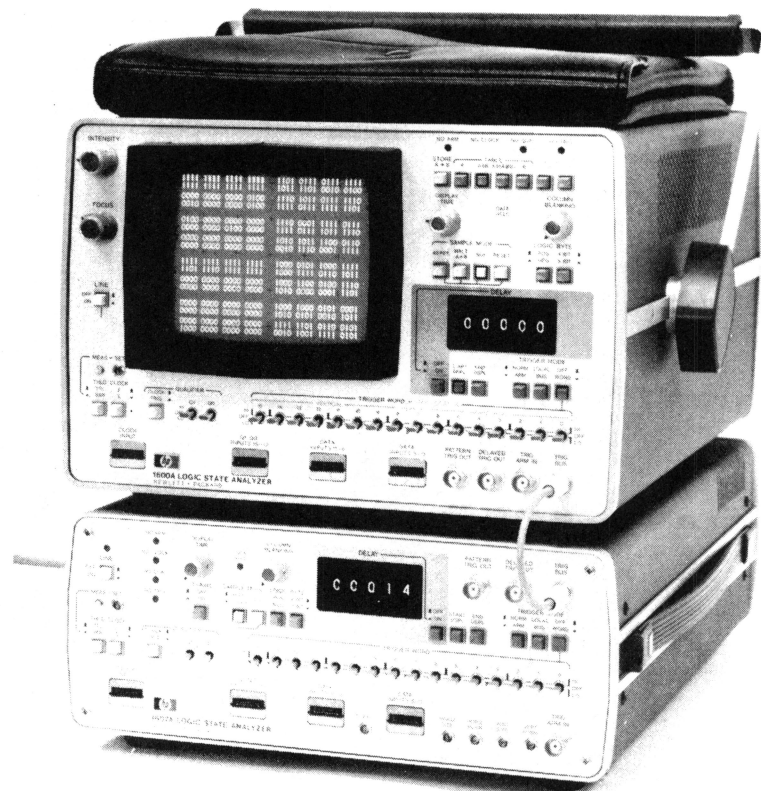

Fig. 8.10

Som det ses af fig. 8.6–8.8 kan der vises 16 kanaler samtidig enten som:

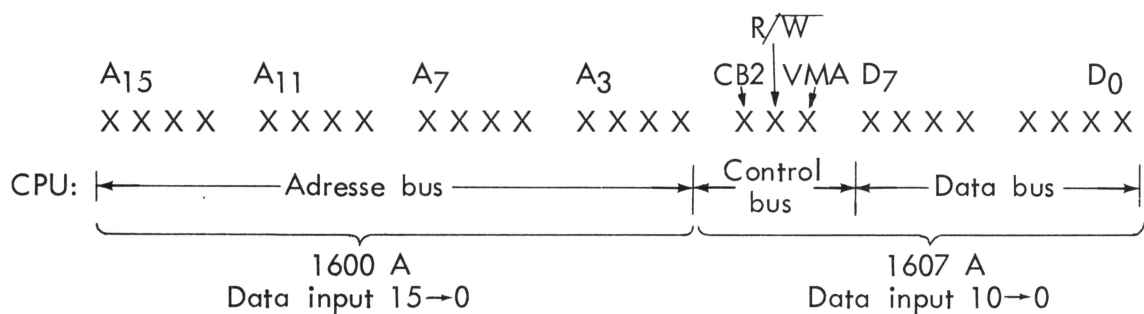
Puls plan (time domain)	fig. 8.6
Binært, octal, HEX (data domain)	fig. 8.7
Koordinater (map mode)	fig. 8.8

På analysatoren INDSTILLES op til 16 bit som TRIGGERORD. Når måleudstyret ser dette bitmønster, sker der trigning. Der kan forud indstilles, om man i time- eller data domain, på oscilloskopet ønsker at se 16 clock-pulser FØR TRIGGER eller 16 clockpulser EFTER TRIGGER. Desuden kan der forudindstilles et DELAY, som angiver, at display ønskes startet et bestemt antal clockpulser EFTER, at TRIGGERORDET er genkendt.

Et større system haves med HP 1600 A sammen med HP 1607 A, der ialt giver 32 bit triggerord og display. Dette system ses i fig. 8.11.

Fig. 8.11 

Som eksempel på anvendelsen kan der undersøges en INTERRUPT ROUTINE, idet interrupt systemet benyttes som trigger. Til analyser-proberne (data input) føres følgende signaler fra microcomputeren MC 6800, der her er benyttet:



og ved interrupt fås f.eks. det i fig. 8.12 viste display. Hvert statement svarer til en  $\phi_2$  clock cycle, idet CPU clock  $\phi_2$  tilføres analysatorens clockterminal.

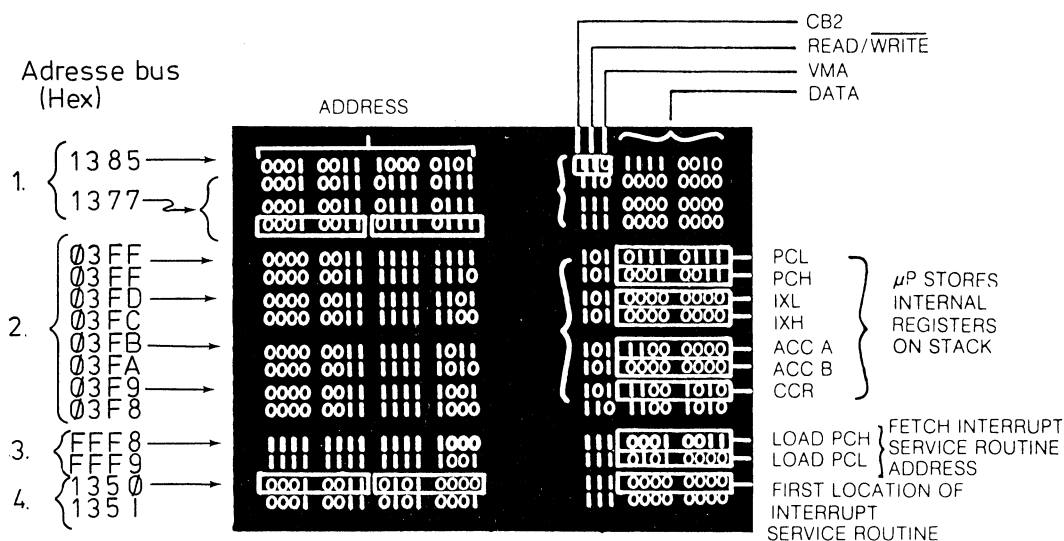


Fig. 8.12

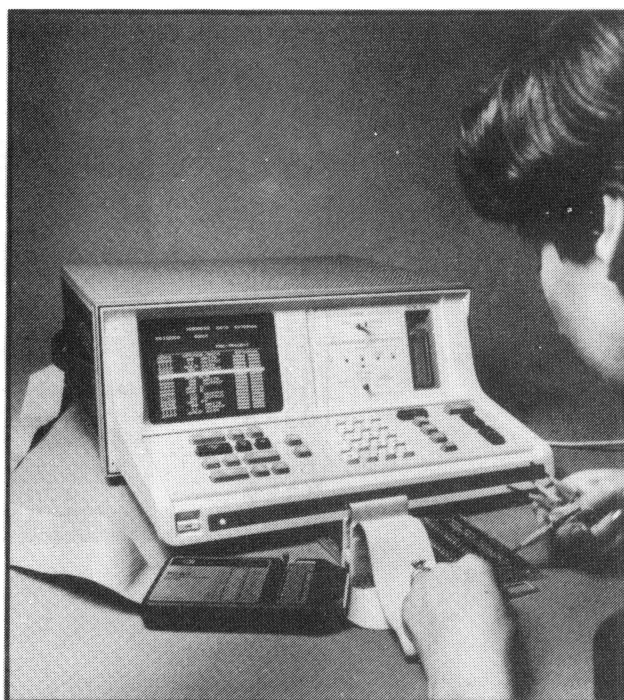

Der gennemløbes følgende faser, som kan ses på display:

1. Den igangværende instruktions-cycle afsluttes. (Det har været en BRANCH-instruktion med OFFSET = F2, idet adressen ændres med størrelsen F2 (hex 2 komplement), der svarer til springet fra adr. 1385 til adr. 1377.
2. De næste linier fra adr. 03FF og til adr. 03F8 viser, at CPU'en SKRIVER sine registre ud i STACK.
3. Herefter udsendes IRQ-ADRESSERNE fra CPU'en: ADR = FFF8 og FFF9. Data = IRQ vektoren på disse to adresser, ses at være 13<sub>(H)</sub> og 50<sub>(H)</sub>.
4. IRQ-vektoren (1350) ses på ADRESSEBUS og data på denne adresse er den første byte i IRQ rutinen, d.v.s. første instruktion.

Ved at følge disse trin kan man se, hvor microcomputeren VAR, hvor den ER og hvor den GÅR HEN. På de enkelte ANALYZER-fabrikater findes egenskaber ud over de her beskrevne, men som fællesnævner for dem alle kan det siges, at de kan give brugeren mulighed for at se, hvad der sker i et microcomputersystem, når SOFTWAREN udføres. Det er herved muligt at finde frem til de steder, hvor der opstår forkerte adresser, findes uønskede data eller forkerte kontrolsignaler.

Logic Analyzers udvikles meget i disse år og mere avancerede modeller fremkommer med jævne mellemrum. Som eksempel på en af disse modeller kan vises HP 1611 A i fig. 8.13 display fra samme i fig. 8.14.




Fig. 8.13 

ADDRESS DATA EXTERNAL		
TRIGGER 0145		
ADRS	OPCODE/DATA	EXTERNAL
0145	01	0000 0000
0154	CALL 0981	0000 0000
37FD	01 WRITE	0000 0000
37FC	57 WRITE	0000 0000
0981	LXI D, 34C0	0000 0000
0984	CALL 076E	0000 0000
37FB	09 WRITE	0000 0000
37FA	87 WRITE	0000 0000
076E	LDAX B	0000 0000
0862	97 READ	0000 0000
076F	CPI FE	0000 0000
0771	RZ	0000 0000
0772	JC 077B	0000 0000
077B	STAX D	0000 0000
34C0	97 WRITE	0000 0000
077C	INX D	0000 0000

(a)

ADDRESS DATA EXTERNAL		
TRIGGER 0145		
ADRS	OPCODE/DATA	EXTERNAL
0145	01	0000 0000
0146	54 READ	0000 0000
0147	01 READ	0000 0000
0154	CD OPCODE	0000 0000
0155	81 READ	0000 0000
0156	09 READ	0000 0000
37FD	01 WRITE	0000 0000
37FC	57 WRITE	0000 0000
0981	11 OPCODE	0000 0000
0982	C0 READ	0000 0000
0983	34 READ	0000 0000
0984	CD OPCODE	0000 0000
0985	6E READ	0000 0000
0986	07 READ	0000 0000
37FB	09 WRITE	0000 0000
37FA	87 WRITE	0000 0000

(b)

Fig. 8.14 

## SIGNATURANALYSE

Der findes specielt måleudstyr, som kun kan benyttes, når et grundsystem er opbygget og afprøvet. Dette udstyr refererer til den kendte virkemåde i grundsystemet og kan fortælle, hvad der gik galt i det udstyr, der nu måles på.

Måleudstyret benævnes oftest SIGNATURANALYSATORER.

Princippet i signatur-analysen er det, at man sammenligner de data, som optræder i de forskellige knudepunkter i kredsløbet med dem, som er angivet i diagrammet for det pågældende måleobjekt. Fremgangsmåden er identisk med det der sker, når de målte spændinger i et kredsløb sammenlignes med de værdier, som er angivet på dia-

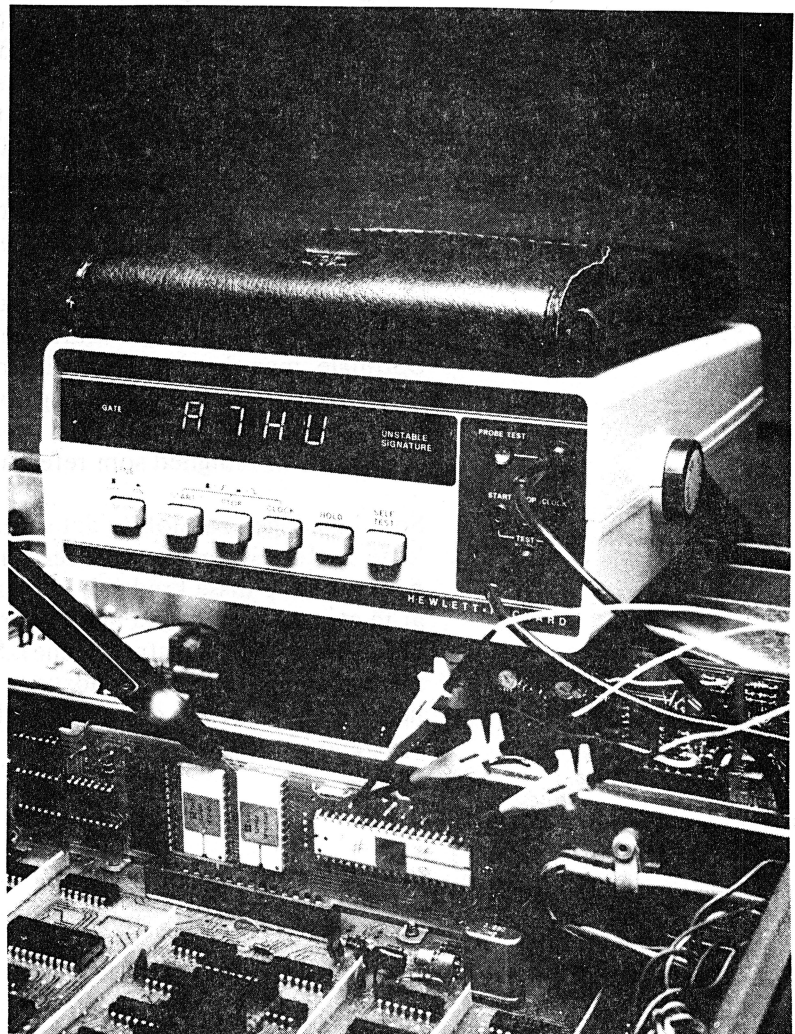

grammet. Når det er digitale kredsløb, det drejer sig om, lyder det som en kompliceret og uoverskuelig fremgangsmåde. For at opnå en overskuelig præsentation af de forskellige data er det nødvendigt at foretage en reduktion af data på en sådan måde, at der så vidt muligt er en entydig sammenhæng mellem de faktiske data og de præsenterede data. Det vil være ret generende, hvis fejlbehæftede data i kredsløbet kunne give en indikation, som ville være korrekt. Hewlett-Packard har under udnyttelsen af denne filosofi frembragt et instrument, som kaldes en signaturanalysator. De forskellige data reduceres til 4 hexadecimalle tal, der er „måleresultatet”. Sammenligner man signatur-værdierne forskellige steder i kredsløbet med de tilsvarende angivelser i diagrammet, vil det være muligt at indkredse den fejlbehæftede komponent.

En signaturanalysator kan godt bruges, uden at kredsløbet er specielt forberedt herpå. Resultaterne vil selvfølgelig langt fra være optimale, men dog brugbare. En fremgangsmåde beroende på sammenligning mellem et defekt og godt apparat tilbyder sig umiddelbart. Ulempen er, at processen er langsommelig, og at der skal være et apparat til rådighed som reference.

Som i analoge kredsløb er det særligt vanskeligt at finde fejl i lukkede sløjfer. Sådanne optræder i rigt mål i mikroprocessorstyringer. Netop disse kredsløb er de mest intrikante at fejlsøge. For at få et overblik over hændelsesforløbene er det nødvendigt at bryde sløjferne (datastrømmene). Skal dette kunne gøres under fejlsøgning og service, må udstyret være konstrueret med henblik herpå. Ønsker man det bedst mulige resultat, bør der ofres de nødvendige ekstra komponenter.



Fig. 8.15 viser HP 6004A signaturanalysator:

Fig. 8.15 

## SOFTWAREAFPRØVNING

Det grundlæggende princip i al afprøvning er at sammenligne et eksisterende kredsløb, komponent eller system med, „hvad det burde være”. Problemet kan selvfølgelig være at vide, hvad det burde være, men ellers at udføre en fornuftig procedure for sammenligningen på en systematisk måde.

De afprøvningsinstrumenter og -metoder, der anvendes ved en sådan sammenligning, er beskrevet på de foregående sider. Men som sædvanligt i computer-tilfælde, kan problemer løses enten som hardware eller som software. I det følgende vises nogle software-løsninger.

## SELF-DIAGNOSTIC

I den metode, der betegnes som SELF-DIAGNOSTIC, beslutter microcomputeren selv, om den er „i orden”, og hvis dette ikke er tilfældet, fortæller den, hvor fejlen findes.

Det grundlæggende princip ved SELF-DIAGNOSTIC er at udføre

en „WORST-CASE” sekvens i computeren og se på resultatet af denne. Er der tale om selve CPU'en, kan en worst-case sekvens af instruktioner normalt oplyses fra fabrikanten. En sådan sekvens vil typisk afprøve alle maskinens instruktioner i en forud bestemt rækkefølge. Udover dette kan den indeholde nogle kritiske instruktionssekvenser, som i visse tilfælde kan fejle.

SELF-DIAGNOSTIC programmer benyttes i udstrakt grad på systemer, hvor der er ledig CPU-tid. Det er en simpel opgave at skrive et testprogram, der benytter de fleste af maskinens instruktioner og lægge dette program i en ubenyttet del af programhukommelsen (ROM). Testprogrammet kan køres på alle tidspunkter, når processoren er „ubeskæftiget”, d.v.s. har tid til overs, og den kan vise, om virkemåden er korrekt. Hvis SELF-DIAGNOSTIC programmet køres over et vist tidsrum, kan det være med til at afsløre periodiske fejl.

Er der ikke plads til programmet i programhukommelsen (ROM), kan det fra en ydre enhed læses ind i RAM og være med til at afprøve udstyret ved, at det eksekveres fra dette RAM-lager i en afprøvningsprocedure.

SELF-DIAGNOSTIC benyttes også til at afprøve hukommelser eller input-output enheder. RAM test omtales senere i afsnittet med ”ALGORITME – MØNSTER – GENERERING”.

Når der er tale om ROM, er den mest enkle form for SELF-DIAGNOSTIC den såkaldte CHECKSUM PRØVE. Ved denne teknik efterfølges hver datablok bestående af 16, 32 eller 256 ord af en ONE BYTE- eller TWO BYTE CHECKSUM. Typisk udregnes en sådan checksum ved at summere (addere) alle HALF-BYTENE i hver af blokkens ORD, idet der benyttes hexadecimal aritmetik. Summen afkortes derefter til kun at indeholde de sidste 4 binære cifre og CHECKSUM-KARAKTEREN er ASCII udtrykket for det resulterende hexadecimale ciffer.

Et simpelt program i et sikkert område af ROM'en, d.v.s. et område der anses for at være fejlfrit, kan læse indholdet i resten af ROM'en, beregne checksummen og derefter sammenligne resultatet med en i programmet lagret værdi. Hvis der findes afvigelse i sammenligningen, så er der fundet en ROM-fejl.

Testning af input-output interface og I/O kredsløb er sædvanligvis vanskelig, set ud fra de uoverskuelige timing forhold, der eksisterer. Det er imidlertid muligt at foretage en grov afprøvning for at verificere generel virkemåde af selve kredsløbene. Forudsat at der er mulighed for feedback fra enhederne, kan microcomputer-programmet udføre en ordre som: „slut relæ A”. Der kan så via feedback ses, om kontakten i relæet er sluttet. På denne måde kan systemet afprøve alle de ydre kontrolenheder og verificere deres generelle virkemåde. Ligeledes kan der under alm. drift foretages „FORNUFTS-TEST” på indkommende signaler. Sådanne tests vil sammenligne input-signalerne med nogle i hukommelsen oplagrede tabel-værdier og heraf beslutte, om input data er „FORNUFTIGE”. F.eks. vil målte vandtemperaturer under 0°C og over 100°C blive udpeget som ufornuftige.

Naturligvis kan tests blive meget mere følsomme og også mere omfattende end dette eksempel. Disse ”fornufts-tests” vil opdage pe-

riodiske og permanente fejl på inputkredsløb og kan herefter udsende en „alarm” til den ydre verden.

## **STORED RESPONDS**

Ved den afprøvningsmetode, der benævnes STORED RESPONDS, benyttes en stor computer til at EMULERE eller SIMULERE det system eller det kredsløbskort, der er til afprøvning. Først benyttes et program til at måle KARAKTERISTIKKEN af systemet eller kredsløbet, der testes. Dette foretages helst under dynamiske forhold. Data indspilles og skal senere benyttes af SAMMENLIGNERPROGRAMMET. Sammenlignerprogrammet tilføres derefter systemet eller kortet. Det skaber input-signalerne. Output måles og sammenlignes med det tidligere respons fra systemet, idet dette er gemt i et lager. I sådanne systemer er det nødvendigt med to gennemkørsler. Den FØRSTE FASE er en PRÆGEFASE, hvor computeren benyttes til at „optage” vigtige svar fra systemet og disse benyttes senere som referencer. Når disse svarkarakteristikker er opnået, vil computeren i FASE TO kun køre i SAMMENLIGNINGSMODE og udføre et bestemt testprogram og måle svarene.

Metoden anvendes hovedsageligt i PRODUKTION og ved INDGANGSTEST. Prisen på det system, der kan give effektiv STORED RESPONDS testning inklusive programmet kan ligge fra 500.000,- til 5.000.000,- d.kr.

## **ALGORITME MØNSTER GENERERING**

ALGORITME - MØNSTER – GENERERING benyttes hovedsagelig til at prøve en RAM-hukommelse. Princippet består i at skrive et mønster ind i hukommelsen og derefter kontrollere, at:

1. det bliver skrevet korrekt.
2. intet blev skrevet andre steder end ønsket på grund af forkert RAM-virkemåde.

De to grundlæggende mønstergenererings-principper, der anvendes ved RAM-afprøvning, er:

### **FAST MØNSTER TEST GALOPERENDE MØNSTER TEST**

## **FAST MØNSTER TEST**

Ved FIXED PATTERN TEST skrives på hver af hukommelsens adresser efterhånden IDENTISKE, VARIERENDE og CYKLISKE mønstre, som derefter udlæses. Dette vil vise, om der er GROVE RAM-fejl. Det vil dog ikke vise MØNSTER FØLSOMME problemer. MØNSTER FØLSOMHED er en typisk fejkilde ved CHIP's med stor pakketæthed. På grund af Chip'ens geometriske layout kan nogle bit-kombinationer indskrevet på bestemte tidspunkter i hukommelsescellerne forårsage, at nogle andre bit positioner andre steder i hukommelsen sættes ON eller OFF. Disse problemer kan opstå i selve RAM-hukommelsen eller inden i microprocessoren. Når som helst problemet optræder i en microprocessor, er det en grundlæggende konstruktionsfejl, og der er ikke meget for brugeren at gøre ved det. Det bedste, som brugeren kan gøre, er at køre et af fabrikanten leveret WORST CASE program, som har vist sig at fremkalde lignende fejl på grund af den specielle rækkefølge af de implicerede instruktioner. Problemerne vil ikke blive behandlet her, idet de anses for at opstå yderst sjældent, når en chip har været på markedet i mere end et år.

Når der er tale om hukommelser og her især de, der har stor pakketæthed, så er MØNSTER FØLSOMHED et almindeligt tilbagevendende problem, som dog relativt let undersøges med den her omtalte ALGORITME MØNSTER GENERERING.

## GALOPERENDE MØNSTER TEST

Den galoperende mønster test forkortes ofte til GALPAT (af GALopping - PATtern test). Princippet i denne testmetode er efterhånden at skrive binære værdier ind i hukommelsescellerne, for derefter at sammenligne dem med resten af hukommelsen, før der går til næste hukommelseslokation.

På denne måde vil der ske en detektering ved hjælp af testen, hvis skrivning i hukommelsescelle 0 har påvirket indholdet i celle 102. testen. I en typisk GALPAT vil hukommelsen blive startet med et bestemt kendt bitmønster enten alle på 1 eller alle på 0. Den grundlæggende test-algoritme er følgende:

1. Indholdet af lokation (N-1) testes mod indholdet af alle de øvrige hukommelseslokationer. De skal matche.
2. Adresse (N-1) forøges herefter med en og trin 1 udføres igen. Således fortsættes, indtil alle hukommelseslokationer er afprøvet.
3. Start datamønstret INVERTERES herefter og man går tilbage til 1.

Det er muligt at udføre mange forskellige variationer af denne grundlæggende GALPAT. De har fået øgenavne som:

- „Marcherende 1'er og 0'er". (Marching Ones and Zeros)
- „Gående 1'er og 0'er" (walking Ones and Zeros)
- „Galoperende mønstre" (galloping patterns)

Det idelle ville være, om man kunne skrive alle de mulige mønstre i hver af hukommelseslokationerne og efter at have skrevet mønstret i et ORD, så afprøve alle de andre ord i hukommelsen for at se, om de var blevet ændret. Ligeledes burde man efter check af hvert af ORDENE hoppe tilbage til det originale ORD for at se, om det var blevet ændret af testen. Man kunne nemlig tænke sig, at den oprindelige lokation ændrede sig ved hver test mod de øvrige celler og tilfældigvis var rigtig igen, når man senere vender tilbage til den. Disse sidste afprøvningsmetoder vil tage meget lang tid at udføre på f.eks. en 32k RAM hukommelse og bør derfor kun anvendes ved den første grundige undersøgelse (initial test), eller hvis der er begrundet mistanke til et kredsløb.

## SIMULERING og EMULERING

Lad os først definere de to begreber:

SIMULERING er funktionserstatningen for et stykke hardware, og udføres af et program. Det betyder, at hardwaren (opstillingen, udstyret) bliver SIMULERET af noget software (et program). Programmet vil på de samme input signaler generere de samme output-signaler, som hardwaren ville have gjort. Uheldigvis vil SIMULATOREN udføre denne simulering meget LANGSOMMERE end den originale hardware var i stand til.

EMULATION er en simulering udført på VIRKELIG TID (eng. REAL TIME SIMULATOR).

I virkeligheden vil mange emulatorer kunne udføre simuleringen hurtigere end selve modellen, der simuleres. F.eks. er der mange bit-slice systemer, der emulerer instruktionssættet til en anden computer.

Simulering benyttes i forbindelse med to vigtige kredsløb: selve microprocessoren og ROM-hukommelsen.

ROM-simulering (eller emulering) udføres ved at køre programmet ud fra RAM, som om denne var en ROM. Dette gøres normalt ved alle programmer i udviklingsfasen. Det er helt sikkert, at det første program, der skrives, indeholder et antal fejl (bugs), og af den grund bør det ikke fra starten skrives i ROM eller PROM. I et typisk udviklingsudstyr vil sådan et program blive placeret i RAM-hukommelse og der blive afprøvet og fejlsøgt (de-bugged). De to store problemer er at omforme adressen for det endelige program til de, der kræves af ROM-hukommelsen og at opnå ensartet hastighed.

Typisk sidder RAM-området på en fastlagt fysisk adresse, som ikke passer til ROM-adressen i det endelige system. Det andet problem er et synkroniseringsproblem, når der benyttes et langsommere RAM-lager til udviklingen og programmet til sin tid placeres i en hurtigere ROM. I større udviklingssystemer vil der normalt være taget hensyn til disse forhold ved emuleringen eller simuleringen.

Simulering eller emulering af selve microprocessoren er meget mere kompleks. Simulering af processoren benyttes i to tilfælde:

1. Hvis selve microprocessoren ikke kan fremskaffes.
2. For at lette fejlsøgningen.

## KRYDSPROGRAMMER

Når programmer udvikles på større computersystemer, benyttes såkaldte krydsprogrammer (Cross programs). En kryds-assembler på en IBM 370 maskine kan f.eks. være indrettet til at generere 6800 kode. Det er nødvendigt at afprøve den korrekte udførelse af den resulterende 6800 kode. Dette gøres med en 6800 simulator, som ligger i hovedmaskinen (IBM 370). Denne simulator udfører alle 6800 instruktionerne i simuleret tid. På den måde bliver hele logikken i programmet tested. Den vigtige begrænsning, der ligger i en sådan simulering, er det forhold, at ingen input-output funktion kan afprøves med mindre brugeren anbringer kendte data på det rigtige tidspunkt i udvalgte hukommelsesceller. Input-output registre kan på den måde simuleres af hukommelsescellerne.

Uheldigvis er timing af I/O enheder ofte tilfældig og næsten altid kompleks. Af den grund benyttes en simulering kun til at afprøve den almindelige logik i et program. Simulering er velegnet til at afprøve numeriske algoritmer, men højst uegnet til at fejlsøge kompleks input-output interface.

## IN CIRCUIT EMULATOR

I ethvert system, hvor brugeren må afprøve virkelige input-output i virkelig tid, er et af de vigtigste „test instrumenter” EFTERLIGNINGEN (emulation) af selve microprocessoren. Når dette gøres, kaldes det:

### IN CIRCUIT EMULATION (ICE)

ICE blev indført af INTEL på deres MDS (Microcomputer Development System) og fås nu på alle førende udviklingsudstyr fra microcomputerfabrikanter og endda fra uafhængige computervirksomheder. Et eksempel på udviklingsudstyr med ICE ses på fig. 8.16. Der er her tale om INTEL's system „Intellec MDS”.

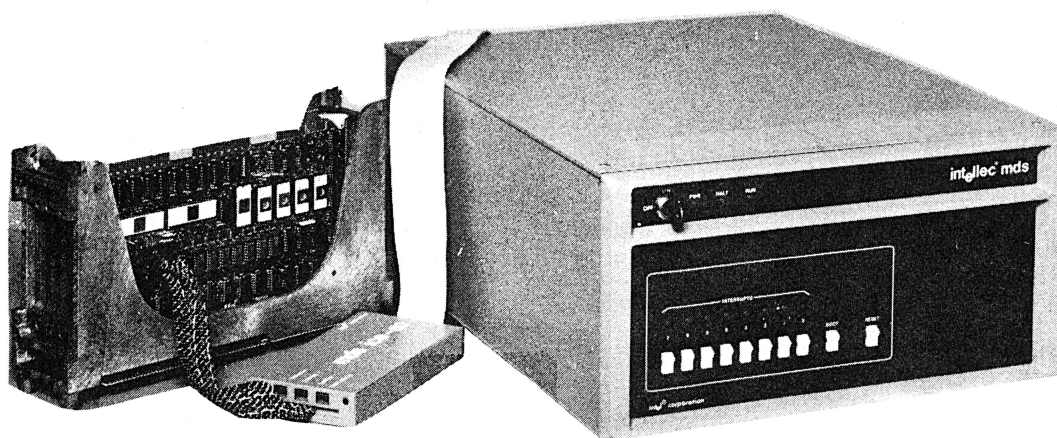


Fig. 8.16 intel

Selve udviklingssystemet forsynes med et ekstra printkort og en ydre kontrolprobe. Fra kontrolproben går der en „navlestreng” hen til det kort, der skal afprøves. „Navlestrengen” er afsluttet med et stik, der indsættes i CPU'ens sokkel og udviklingsudstyr inkl. probe udgør herefter CPU'en. Fordelen ved dette er, at EMULATOR i udviklingsudstyret nu kan have total kontrol over hele systemet, der skal afprøves. Operatøren kan fra konsollen bestemme, hvilke funktioner og enheder, der skal afprøves. Man kan stoppe CPU'en og se på indholdet i registre og hukommelser. Indholdet kan ændres, hvis dette ønskes. Der kan benyttes „break points” til automatisk at stoppe CPU'ens programeksekvering, når den når til et bestemt punkt i programmet og/eller når der opstår en bestemt signalkombination på busserne. Man kan sidde ved udviklingsudstyrets tastatur og udføre input-output og andre funktioner ved blot at aktivere tasterne — og alt dette kan foretages med de samme kommandoer, som blev benyttet i programudviklingens første faser, idet OPERATIVSYSTEMET er det samme om der køres SOFTWARE-UDVIKLING eller EMULERING på udviklingssystemet. Desuden findes ofte ANALYSATORER i forbindelse med emulatoren. Fremgangsmåden ved EMULERING er ofte opdelt i flere faser:

#### Fase 1.

Udviklingsudstyret indeholder de fleste af software-funktionerne og udfører disse internt. De eneste YDRE signaler er CLOCK og spændingsforsyning.

#### Fase 2.

Memory overgives efterhånden fra udviklingssystemet til den hardware, der er under afprøvning. Overdragelsen foretages mange gange i „blokke”, men kan også ske en byte ad gangen.

**Fase 3.**

I/O enhederne overdrages fra udviklingsudstyrets emulator til den ydre hardware. Denne fase afsluttes med, at udviklingsudstyret kun udgør selve processoren.

Til slut er der kun at skille udviklingsudstyret fra enheden, der skulle testes og derefter indsætte den virkelige CPU – og så kører systemet!

## LITTERATURLISTE

1. Microprocessors from chips to systems  
Rodnay Zaks  
SYBEX C201, 1977. Teknisk Forlag.
2. Microprocessor interfacing Techniques  
Austin Lesea & Rodnay Zaks  
SYBEX C207, 1978. Teknisk Forlag.
3. Microprocessor Lexicon  
Acronyms and definitions  
SYBEX, 1978. Teknisk Forlag.
4. Microprocessor Course  
Motorola Semiconductor Products Inc., 1979.
5. Microcomputer Components  
Motorola Semiconductor Products Inc., 1979.
6. Semiconductor Data Library, CMOS  
Vol. 5/series B  
Motorola Semiconductor Products Inc., 1979.
7. M6800 Microcomputer System Design Data  
Motorola Semiconductor Products Inc., 1976.
8. AMI MOS Products Catalogue Winter 1979  
American Microsystems Inc., 1978.
9. The TTL Databook for design engineers  
Texas Instruments, 1973.
10. Microcomputere, teori og praksis  
Steen Hannibal og Lars Birkvad  
Teknisk Forlag, 1978
11. Microcomputere – grundlæggende interfacing  
Steen Hannibal  
Teknisk Forlag, 1979.
12. Digital – 79  
B & O internt kursus  
Bang & Olufsen, 1979.
13. MC 6802 datablad  
Motorola Semiconductor Products Inc., 1977.
14. Microcomputerkursus 1 og 2  
B. Thestrup & P.E. Melsen  
Århus tekniske Skole, 1978.
15. AMI S 6800 microprocessors  
Hardware reference manual  
American Microsystems Inc., 1976.
16. Intel Component Data Catalog 1978  
Intel Corporation, 1978.
17. EPROM's  
Electronics 16. Aug. 1979, side 126



18. MC 1408/1508 datablad  
Motorola Semiconductors Products Inc.
19. The IC trouble shooters  
Hewlett Packard, 1976.
20. MODEL 116 displ. contr. accessory  
Biomation datablad, 15/9/76
21. Hewlett-Packard Journal  
Hewlett-Packard, January 1977.
22. Intel 8080 microcomputer systems  
Users manual. Sept. 1975
23. Electronic Design. 12 June 1979.
24. BYTE. Feb. 1978.
25. Digitalteknik  
P.E. Melsen  
Århus tekniske Skole
26. Fairchild F6800  
Programming Manual.
27. Electronics. August 16, 1979.
28. M6800 Microprocessor Application Manual.
29. What is an interrupt?  
BYTE, March 1979, side 230

## ACCESS TIME

Den tid, der kræves til at gøre lagrede data tilgængelige (brugbare). Udtrykket anvendes hovedsageligt ved hukommelser.

## ACIA

Asynchronous Communications Interface Aapter. I/o kredsløb, der udefra modtager serieinformation og omformer dette til paralleldata til CPU'ens DATABUS.

ACCUMULATOR  
(AC)

Det register (i CPU) som indeholder resultater af udførte operationer eller modtager /sender data til/fra CPU.

## ASCII

American Standard Code for Information Interchange.

En 8-bit kode til datatransmission udviklet af The American Standard Association for at opnå tilpasning (kompatibilitet) mellem dataudstyr. (7 databit + 1 paritetsbit).

*Teletype*  
ASR 33

Betegnelse for TTY med 20 mA current loop.

## ADDRESS

Adresse, f.eks.

CURRENT ADDRESS = adressen til nuværende programtrin.

RETURN ADDRESS = Den adresse til hvilken maskinen skal vende tilbage efter udført sub-rutine.

ARITHMETIC UNIT  
(ALU)

Aritmethic Logic Unit (ALU) =

aritmetriskenhed - en kreds som udfører aritmetriske operationer og/eller logiske funktioner. ALU sidder normalt inde i CPU'en.

ASSEMBLER

Anordning til at sammensætte (danne) maskinprogrammet. Oversætter fra ASSEMBLER sprog til MASKINSPROG.

Assembleren kan bestå af SOFT- eller HARDWARE. Assembler er ofte sammenbygget med SIMULATOR.

BAUD  
(BAUD RATE)

Enhed for signalhastighed. Hastigheden i baud er antallet af diskrete tilstande eller signalskift pr. sekund (denne betegnelse anvendes kun om de aktuelle signaler på en kommunikationslinie).

Hvis hvert signalskift kun repræsenterer en bit tilstand, er BAUD det samme som BIT pr. sek. ellers ikke.

BIT

BINARY DIGIT. Den mindste del af en information i binært notationssystem. Et bit er enten et NUL (0) eller et ET (1).

BRANCH (ING)

Hop i programmet.

CONDITIONAL B. = hoppet er betinget af visse afsøgte tilstande (se FLAG).

UNCONDITIONAL B. = hoppet (til en sub-rutine) sker ubetinget samtidig med at nuværende adresse lagres i hukommelsen (STACK).

	<p>BRANCH BACK = hoppe fra en sub-rutine tilbage til hovedrutinen</p> <p>2-, 3-, MULTI-WAY B = betingede hop til flere forskellige program-adresser.</p>
BOOTSTRAP	<p>Bootstrapping = "Lyfte sig i håret"</p> <p>BOOTSTRAP LOADER = Et i ROM indskrevet program, der loader et ønsket program ind i RAM. Denne type load anvendes i SIMULATORER og MINIDATAMATER, hvor hovedparten af programmet lagres i RAM.</p>
BUS LINE (BUS)	<p>Ledninger som flere kredsløb er fælles om at anvende.</p> <p>DATA BUS = overfører data</p> <p>ADDRESS BUS = overfører adresseordrer</p> <p>POWER BUS = overfører forsynings-energi</p> <p>SYNC BUS = overfører synkroniserings-og nulstillingspulser.</p>
BYTE	Ord som omfatter 8 BIT (se også NIBBLE).
CARRY	Hukommelsesciffer (mente)
CHIP SELECT	Valg af bestemt(e) kreds(e) (se også ENABLE)
CLEAR	Rense, nulstille.
CODE	<p>Kode (se også NUMBER SYSTEMS). Ofte benyttes specielle tegn til angivelse af kodetypen, f.</p> <p>\$ = HEX CODE</p> <p>@ = OCTAL CODE</p> <p>% = BINARY CODE</p>

COMPILER ( CROSS COMBILER)	Oversætter program fra højniveau-sprog til maskinkode, f.eks. fra ASSEMBLER- til MASKINKODE. CROSS C. angiver, at en computertype benyttes til oversættelse og anden computer-type anvender det oversatte.
CP	Forkortelse for clock pulse.
CPU	CENTRAL PROCESSING UNIT (CPU) = kommandocentralen i en datamat.
CROSS ASSEMBLER	ASSEMBLERINGEN foretages af en anden maskintype end den, hvortil programmet skal anvendes
CS	Forkortelse for CHIP SELECT.
CYCLE TIME	CYKLUSTID (for datamat). Den tid, der er nødvendig for en enkelt ordre (instruktion) at blive modtaget og udført. Visse ordrer kan kræve to eller flere CYCLES, hvilket gør det svært at sammenligne maskinernes hastighed. Læg mærke til, at ordlængden har indflydelse på maskinens totale hastighed. Ved hastighedssammenligninger anvendes ofte additionstiden for et antal decimale cifre.
DATA POINTER	Se POINTER
DEBUGGING	"Aflusning", d.v.s. rette fejl i PROGRAM eller HARDWARE.
DECIMAL ADJUST	(af ACCUMULATOR) - operation ved hvilken akkumulatoren forøges med 6 (0110) i de tilfælde, hvor CARRY er 1 eller hvis den i ACCUMULATOR lagrede værdi er større end 9 (1001).

DECODE	Et kredsløb, der accepterer kodede input DATA og aktiverer et specificeret output i overensstemmelse med koden, der i øjeblikket er på input.
DECREMENT	Formindske med en enhed. (-1)
DMA	<u>D</u> irect <u>M</u> emory <u>A</u> ccess. En metode, hvor der er mulighed for adgang til MEMORY fra eksterne enheder via I/o kredsløbene.
DP	Forkortelse for DATA POINTER.
DUMP	At lægge program og data fra memory ud på externt medie, f.eks. hulstrimmel eller tape.
EBCDIC	8 bit kode, der anvendes til datatransmission.
EDITOR	Et program, der hjælper ved tilrettelægning af SOURCE PROGRAM ved at tillade nem manipulation eller editering af tekstmateriale.
ENABLE	Muliggøre, sætte i stand til (se CHIP SELECT..
ENCODE	At lave til kode. F.eks. sker omformning af skrivemaskinens tastslag til ASCII-kode v.h. a. en ENCODER.

EXECUTE	Udføre. F.eks. PROGRAM EXECUTE = at udføre det, der angives i programmet.
EAPROM	se MEMORY
EPROM	se MEMORY
EXCHANGE	Ombytte indholdet af to registre (hukommelser) uden at data mistes (se også LOAD).
EXPANDER	Indretning til at forøge antallet af ind- eller udgange.
FETCH	<p>Hente frem. I visse tilfælde anvendes FETCH i betydningen LOAD.</p> <p>F. IMMEDIATE = F. DIRECT = Data hentes direkte fra den eller de hukommelsesceller, som angives af adressen (som normalt følger efter selve ordren).</p> <p>F. INDIRECT = Den adresse, som følger umiddelbart efter ordren angiver en registerhukommelse, hvori der er angivet den adresse, hvor data skal hentes.</p>
FIRMWARE	Det begreb, der dækker, at fabrikanten leverer SOFTWARE i form af HARDWARE, f.eks. at et program (Software) leveres i en ROM (Hardware) - Mellemtning mellem (eller blanding af) SOFT- og HARDWARE.
FIXED INSTRUCTIONS	<p>Fast instruktionsliste, bestemt af datamat- (CPU) fabrikanten.</p> <p>(se også MICROPROGRAMABLE COMPUTER)</p>

FLAG	Flag, et eller flere bistabile trin, som markerer visse tilstande, f.eks. CARRY, AC = 0, tilstanden på direkte indgang (INTERRUPT) o.s.v. Flagene TESTES ved betingede hop (CONDITIONAL JUMPS).
FSK	Frequency - Shift Signaling. <u>F</u> requency - <u>S</u> hift <u>K</u> eying. En frekvens modulations metode, hvor frekvensen varierer i den tid, der er af betydning for informationen.
HANDSHAK (-ING)	Udveksling af forud bestemte signaler mellem maskiner, der er forbundet v.h.a. en kommunikationskanal. Gøres for at sikre, at de begge er i forbindelse med hinanden.
HARDWARE	De virkelige fysiske kredsløb. (det modsatte er SOFTWARE).
HARD WIRING	At sammensætte de fysiske kredsløb.
HEXADECIMAL	Se NUMBER SYSTEMS.
INCREMENT	Forøge med en enhed (+1).
INDEX REGISTER	En hukommelse (register), der kan adresseres med korte instruktioner således, at data hurtigt kan føres til eller tages ud af registre. INDEX REGISTER anvendes til lagring af rene cifferverdier (tal) samt til lagring af adresser ved indirekte adressering.
INSTRUCTION	Programtrin, der fortæller, hvad der skal udføres. Instruktionen kan bestå af en eller flere BYTES. Instruktionerne = programmet er normalt lagret i ROM og tilføres CPU'ens INSTRUKTIONSDEKODER.



INTERRUPT

Afbryde. INTERRUPT anvendes, når vigtig information til datamaten skal have prioritet, f.eks. ordre om nødstop eller lignende. INTERRUPT kan være gradueret, så forskellige indgange får forskellig høj grad af prioritet. Ved INTERRUPT afslutter datamaten den CYCLE (programtrin), der er i gang og markerer derefter, at den er klar til "indhoppet". Ved INTERRUPT må man ofte gennem en serie programtrin redde de DATA, som ligger i forskellige registre.

I/O

Forkortelse af INPUT/OUTPUT network.

INTERFACE

Tilpasning, ledninger til tilpasning.

IRQ

Interrupt Request. Krav om afbrydelse (se INTERRUPT).

JUMP

Hoppe, f.eks. til en ny programadresse.

CONDITIONAL J. = Betinget hop. Hoppet udføres kun, hvis visse betingelser er opfyldt.

UNCONDITIONAL J. = Ubetinget hop (til en subroutine). Den gamle adresse gemmes således, at der kan hoppes tilbage (BRANCH BACK).

JUMP DIRECT = i hopordren gives direkte den adresse, til hvilket hoppet skal foretages.

JUMP INDIRECT = i hopordren gives adresser på en hukommelse (normalt i INDEX REGISTER). Denne hukommelse giver besked om adressen til hvilken hoppet skal foretages.

## LABEL

Etikette. En given række i SOURCE PROGRAM gives et navn eller en forkortelse (etiketten). Rækken kan f.eks. være første programtrin i en SUBROUTINE, og etiketten bliver da "navnet" på hele subroutinen.

Hvis blot etiketten skrives på korrekt måde, kan ASSEMBLER kende den igen. Man kan så senere i SOURCE PROGRAM kalde SUBROUTINE ved blot at angive dens etikette og behøver ikke at kunne den virkelige adresse, udtrykt i maskinsprog.

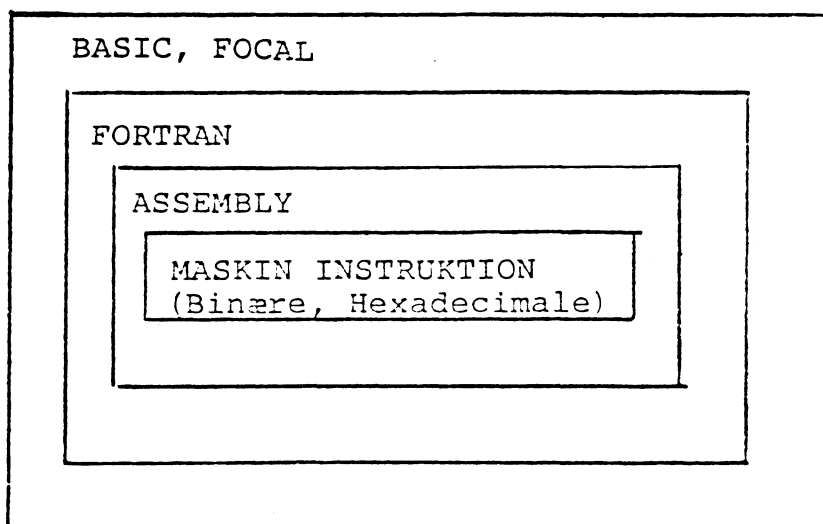
## LANGUAGE

Sprog, som omfatter et antal ord og en grammatik, som angiver hvorledes disse ord skal kombineres med hinanden.

Der findes sprog på højere eller lavere niveau. Jo højere niveau, sproget har, desto lettere er det at skrive den ønskede programsekvens. Der findes oversætterprogrammer, med hvilket en datamat kan oversætte højniveausproget til maskininstruktioner.

Desværre er disse automatiske oversættelser ikke helt optimale - man får unødigt lange maskinprogrammer - forøgelsen er sædvanligvis fra 20% til 80% - hvilket medfører stort "ROM-areal", og at tiden for at fuldføre et bestemt program bliver unødigt lang.

Med hensyn til niveau kan sproget grupperes, som vist nedenstående. Bemærk, at ASSEMBLY - og MASKINSPROGET er helt afhængig af, hvilken datamat (CPU), der vælges.



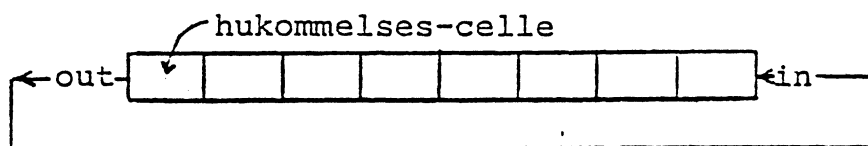
LIFO	<p><u>L</u>ast <u>I</u>n - <u>F</u>irst <u>O</u>ut. Hukommelsesregister (memory), hvor sidste lagrede information kommer først ud ved en efterfølgende udlæsning. Benyttes bl. andet ved lagring af CPU-indhold i forbindelse med INTERRUPT.</p> <p>(Se også STACK).</p>
LOAD	<p>Indsæt, lagre data i hukommelsesceller (f.eks. LOAD AC <math>\Rightarrow</math> ACC. forsynes med data)</p> <p>Data som før LOAD befinder sig i hukommelsescellerne (registret) går tabt ved LOAD.</p>
LOOK AHEAD	<p>Se fremad, en kreds, som hurtigt tager de signaler frem, der skal tilbagelægge store "afstande" f.eks. carry ved addition af mangede tal (Look Ahead Carry).</p>
LSB	<p>Forkortelse for <u>L</u>EAST <u>S</u>IGNIFICANT <u>B</u>IT, som er den binære enhed, der har laveste værdi (enheden længst til højre i tallet).</p>
MEMORY	<p>Hukommelse</p>
RAM	<p><u>R</u>ANDOM <u>A</u>CC<u>E</u>SS <u>M</u>EMORY er en hukommelsestype, som der kan skrives ind i og læses ud fra (rent elektrisk). Typen anvendes til informationer, som ønskes ændret hurtigt, f.eks. IN- og OUT DATA. (Se også SCHRATCH PAD) RAM er VOLATILE.</p>
ROM	<p><u>R</u>EAD <u>O</u>NLY <u>M</u>EMORY er en hukommelsestype, som kun kan lagre faste værdier f.eks. et fast program eller konstanter. Efter montering i maskinen kan den kun udlæses (aflæses). ROM er non-VOLATILE</p>
PROM	<p><u>P</u>ROGRAMABLE <u>R</u>OM er en hukommelsestype, som kan fast programmeres af brugeren.</p>
EPROM	<p><u>E</u>RASEABLE <u>P</u>ROM. En PROM, som kan slettes v. h.a. ultraviolet lys.</p> <p>Efter sletning kan den programmeres igen.</p>

EAPROM

ELECTRICAL ALTERABLE PROM. En elektrisk sletbar PROM.

SR

SHIFT REGISTER er en hukommelsestype (register), hvor DATA lagres i serieform og kan skiftes rundt STEP BY STEP i en hukommelses-sløjfe.



MICROCOMPUTER

Microdatamat, en datamat som er sammensat af et fåtal højkomplekse integrerede halvleder-kredse. Grænsen mod MINIDATAMAT er flydende. I de senere år er minicomputere blevet mere universelle og har sit aktuelle ordreprogram lagret i RAM.

MICROCOMPUTERENS program er sædvanligvis fast og lagret i ROM.

MICROPROGRAMMABLE  
COMPUTER

En datamat, hvor kunden kan specificere visse specielle aritmetriske operationer, hvilke da indgår i datamatens instruktionsliste. Microprogrammering kræver intimt samarbejde mellem bruger og fremstiller.

MINICOMPUTER

Lille datamat, minidatamat (se også MICRO-COMPUTER).

MODEM

En forkortelse for Modulator - Demodulator. Udtrykket kan benyttes, når modulator og demodulator sidder i samme signal-converter-kasse.

MNEMONIC	<p>"hjælp til hukommelsen" mnemoteknik (Mnemosyne = hukommelsens gudinde og mor til muserne i den græske mytologi).</p> <p>I datateknik udtryk for korte bogstavsbetegnelser i ASSEMBLY-sproget, som er lettere at huske end den direkte maskinordre.</p> <p>Eksempler:</p> <p>JUN = Jump unconditionally (ubetinget hop)</p> <p>DAC = Decrement Accumulator.</p> <p>For at MNEMONIC skal være til virkelig nytte, kræves det, at man tænker på de engelske ord, som forkortelserne står for</p>
MPU	Micro Processor Unit (se CPU).
MSB	Forkortelse af <u>M</u> OST <u>S</u> IGNIFICANT <u>B</u> IT , det binære ciffer længst til venstre i ciffergruppen.
MUX	Forkortelse af <u>M</u> ULTIPLEXER, indretning til at placere data i tidsmultiplex efter hinanden på en eller flere måder.
NESTING	<p>"Redebygning" - en metode, hvor der ved hjælp af et hovedprogram og SUBROUTINER sammensættes et komplet program.</p> <p>Med begrænset kapacitet i ADDRESS-STACK kan man ikke tage SUBROUTINE efter SUBROUTINE alt for mange gange, idet alle returadresser skal kunne lagres.</p>
NIBBLE	Et ord med 4 bit (se også BYTE).
NMI	<u>N</u> on <u>M</u> askable <u>I</u> nterrupt. Ikke maskerbar afbrydelse. En afbrydelse, der altid vil påvirke CPU, uanset hvad den er igang med at udføre.

## NUMBER SYSTEMS

BINARY	DECIMAL	HEXADECIMAL	OCTAL
0000	0	00	00
0001	1	01	01
0010	2	02	02
0011	3	03	03
0100	4	04	04
0101	5	05	05
0110	6	06	06
0111	7	07	07
1000	8	08	10
1001	9	09	11
1010	10	0A	12
1011	11	0B	13
1100	12	0C	14
1101	13	0D	15
1110	14	0E	16
1111	15	0F	17
10000	16	10	20

## OBJECT PROGRAM

Maskinprogram. Det program som ASSEMBLEREN tager frem fra kildeprogrammet. (SOURCE PROGRAM)

## OCTAL NUMBER

Se NUMBER SYSTEMS.

## OPCODE

Forkortelse for OPERATION CODE. Opcode benyttes i forbindelse med programmering som angivelse af MNEMONIC- eller MACHINE CODE for OPERATOR og OPERAND eller OPERATOR alene. (f.eks.: LDI = Load Immediate = C4 = 11000100. LDI = OPERATOR MNEMONIC; C4 = OPCODE.)

## OPERAND

I ASSEMBLER-sproget lig med detailspecifikation af OPERATOR, der angiver, hvad der skal gøres. Hvis f.eks. OPERATOR er et betinget hop, så angiver OPERANDEN vilkårene for hoppet.

## OPERATOR

I ASSEMBLER-sproget lig med den hovedopgave, som skal udføres (se også OPERAND).

OPTION (OPTIONAL)	Valg (Efter eget valg. Frit valg. Frivilligt).
OVERFLOW	Flyde over, ikke plads til. I visse tilfælde også benyttet med betydningen: give mente.
PAGE	<p>Side. Programadresser kan f.eks. indeholde totalt 12 bit. De mest betydende. f.eks. 4 bit kan angive SIDE, medens de efterfølgende 8 bit angiver detail-adressen på denne side.</p> <p>Ofte lagres et antal sider i hver sin PROM. Visse hopordrer kan være begrænset til at gælde for en og samme side, og i dette tilfælde sendes de mest betydende cifre i program-adressen ikke ud.</p> <p>Hvis datamaten er opbygget efter PAGE-system, må man med en ordresekvens "blade" den ønskede side frem.</p>
PC	Forkortelse af PROGRAM COUNTER.
PIA	<p><u>P</u>eripheral <u>I</u>nterface <u>A</u>dapter. Interfaceenhed mellem CPU DATA BUS og ydre data linier.</p> <p>Arbejder normalt som parallel ind og parallel ud og er bi-directional.</p>
POINTER	<p>Visere, et hukommelses-indhold som angiver, hvor den aktuelle adresse findes lagret.</p> <p>DATA POINTER (D.P.) = angiver hvor DATA skal hentes og er ofte lagret i et INDEX REGISTER.</p> <p>PROGRAM POINTER (PP) = kan være en POINTER, som angiver, hvor i RAM en aktuel adresse ligger.</p>

	PROGRAM COUNTER (PC) = Adresse pointer, der viser hen til næste adresse i programudførelsen.
POP (Pull)	Hente data fra STACK i memory til ACCUMULATOR i CPU.
PRECHARGE	Forlade. Oplade en leder eller et ledningsnet til en vis tærskelværdi, således at lave energiniveauer kan overføres. PRECHARGE anvendes hovedsageligt i dynamiske hukommelser.
PROGRAM COUNTER	En hukommelse som indeholder aktuel adresse (programtrin). PROGRAM COUNTER kan enten stige en enhed (+1) automatisk, eller indstilles til en ønsket værdi ved hop i programsekvensen.
PSEUDO OPERATOR	Hjælpeinstruktion, der anvendes til tydeliggørelse. Et D kan f.eks. angive, at alle tal, der følger, er decimaletal, indtil andet angives. PSEUDOOPERATORER forsvinder, når ASSEMBLEREN oversætter til det endelige maskinsprog.
PUSH	Sende data fra ACCUMULATOR i CPU til STACK i memory.
RAM	Se MEMORY.
REFRESH	Opfriske indholdet i en dynamisk hukommelse. Genetablere tabt ladning i hukommelseskapaciteter.
REGISTER	En samstilling af hukommelsesceller. (se også MEMORY-SR).



REMOTE

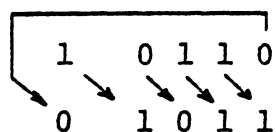
Langt fra. Fjernt beliggende REMOTE CONTROL  
= Fjernbetjening.

ROM

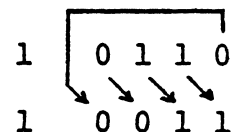
se MEMORY

ROTATE

Få til at rotere. En ordre som skifter de  
binære cifre i ACCUMULATOR rundt. Ordren kan  
indeholde besked om skifteretningen - ROTATE  
LEFT, ROTATE RIGHT. F. eks.:



Rotate Right  
through carry



Rotate Right  
not through carry

RS232

Omformning fra TTL niveau til  $\pm 12V$  niveau.  
(EIA standards).

*EIA standard for serial data trans-  
mission  $\pm 12V$  niveau*

SCHRATCH PAD

Hukommelse af RAM-typen. Normalt lille hukom-  
melses kapacitet, men hurtigt arbejdende.  
Benyttes til at lagre DATA og INSTRUCTIONS,  
der kun er brug for i kort tid.

SHIFT REGISTER  
(SR)

Se MEMORY

STORE	Lagre (i hukommelse).
SUBROUTINE	Underprogram. SUBROUTINE er en programsekvens, der ofte benyttes i hovedprogrammet. I stedet for at skrive programsekvensen hver gang, den skal benyttes i hovedprogrammet, skrives den som et underprogram, der kan kaldes fra hovedprogrammet v.h.a. hopordre (JUMP).
TAG	Etikette (se LABEL).
TIME SHARE	Tids-MULTIPLEX på tilkoblede enheder. Hver enhed (computerterminal) får tildelt en bestemt brøkdel af computerens I/O tid, og i denne tid kan terminalen kommunikere med computeren.
TTY	Forkortelse af <u>TeleType</u> = fjernskriver. Som teletype anvendes sædvanligvis typen ASR-33 (20 mA current loop).
TWO's COMPLEMENT	Metode til at ændre de binære cifre 0 og 1 således, at subtraktion kan udføres i en ARITHMETIC UNIT, der kun kan udføre addition.
VOLATILE	Flygtig. Anvendes om RAM-hukommelser, hvor det binære indhold forsvinder (er VOLATILE), når forsyningsspændingen forsvinder. NON-VOLATILE vil sige, at hukommelsesindholdet <u>bevares</u> , når spændingen forsvinder. ROM er non-volatile hukommelser.

## SIMULATOR

Simulator. Indretning til at prøve, simulere en kreds eller et program.

SIMULATOR kan have form af SOFTWARE, f.eks. som en hulstrimmel, hvorved simuleringen sker v.h.a. en større datamat. SIMULATOR kan også være udformet som HARDWARE, f.eks. en micro-computer, hvis RAM LOADES med programmet. SIMULATOR kan endvidere være sammenbygget med ASSEMBLER.

SIMULERING udføres for at få indikation af evt. fejl i programmet.

## SOFTWARE

"Alt skrivebordsarbejdet". Kan f.eks. være hulstrimmel, udfyldte programark, manualer o.l. Det modsatte er HARDWARE.

## SOURCE PROGRAM

Kildeprogram. Et oprindeligt program, f.eks. skrevet i ASSEMBLER-sprog og senere oversat i maskinsprog, og derved blevet til OBJECT PROGRAM.

## STACK

Stakken (lageret). Almindeligvis en lagerhukommelse. For microcomputere en lagerhukommelse, i hvilken man gemmer en serie adresser, som er nødvendige for tilbagehop fra SUBROUTINER. Denne STACK er af typen LIFO (Last In - First Out) og benævnes af og til som push down stack. Tilføres den for mange adresser, bliver de(n), der ligger dybest (den ældste) tabt.

## STATEMENT

Opgørelse. Udsagn. I dataterminologien f.eks. en række i ASSEMBLER-sproget med følgende udseende:

(LABEL)	OPERATOR	OPERAND	COMMENTS
---------	----------	---------	----------

UART

Universal Asynchronous Reciever Transmitter.  
Asynkron udgave af USRT (se denne).

USRT

Universal Synchronous Reciever Transmitter  
Interfaceenhed, der omformer signaler mellem  
parallel controller eller data terminal og  
synkron bit-serie kommunikationsnetværk.

VMA

Valid Memory Address. Signal, der fortæller,  
hvornår adressen er etableret på det ønske-  
de kredsløb.