LUXOR

Dokumentnamn/Name of document

Blad nr/Sheet no
( )

Från/From	Datum/Date	Beteckning/Reference
Rubrik, ärende/Subject	Ert datum/Your date	Er beteckning/Your reference
	Gäller fr o m/Effective date	Ersätter/Replaces
	Distribution	

Servicemanual och Teknisk beskrivning

ABC 1600

PRELIMINÄR

Innehållsförteckning

1 Allmänbeskrivning ABC 1600

ABC 1600 är ett avancerat datorsystem utvecklat för att motsvara inte bara dagens utan även morgondagens krav av datoriserad arbetsplats.

Trots att ABC 1600 får ses som ett avancerat enanvändarsystem finns alla ingredienser där för att den ska kunna användas som fleranvändardator. Till detta bidrar opereativsystemet som är UNIX-baserade och kallas ABCenix, genom vilket flera användare och flera processer (jobb) samtidigt kan hanteras av systemet. Det välkända och kraftfulla operativsystemet gör att programvara utvecklade på andra datorsystem med små eller inga förändringar kan köras på ABC 1600 och vice versa. För att ytterligare förenkla transport av programvara kan ABCenix med hjälp av speciella filhanterare läsa och generera filformat, som används av andra operativsystem som exv UFD-DOS, OS-8, CP/M och MS-DOS.

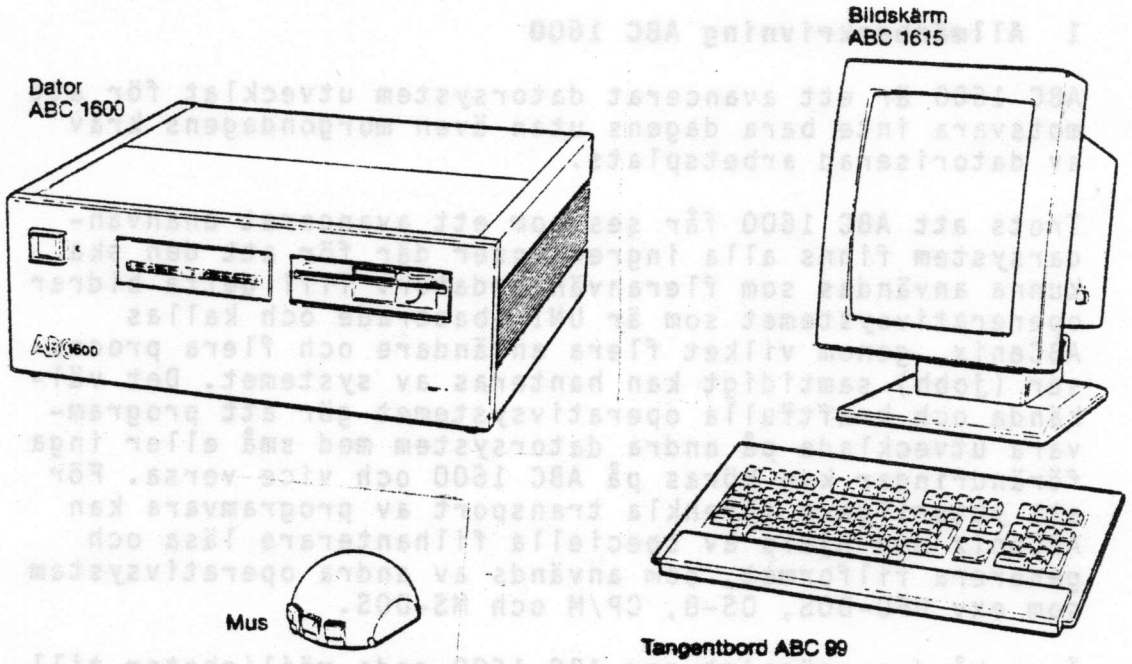
Även hårdvarumässigt ger ABC 1600 goda möjligheter till kommunikation, utbyggnad och anpassning genom expansionsbuss och de generella seriekommunikationsportarna. Den kan härigenom anslutas till många olika typer av massminnen, nätverk typ LUX-NET och som terminal till stordatorsystem.

Systemet är också försett med en fönsterhantere med vars hjälp man lätt hanterar de olika programvarorna och systemrutinerna. Den ger möjlighet att samtidigt ha kontroll över flera programkörningar och ger användaren möjlighet att själv bestämma uppdelningen av bildskärmen.

Genom den ergonomiska utformning av tangentbord och bildskärm uppfyller ABC 1600 även mycket högt ställda krav på arbetsplatsutformning. Tangentbordet ABC 99 gör med sin låga profil och generella utformning att den kan användas till många olika tillämpningar vad gäller applikationsprogram och placeringar.

Tangentbordet är också försett med en sk MUS med vilken man lätt hanterar menyer, grafikeditorer och förflyttnignar av data på bildskärmen.

Bildskärmen ABC 1615 som kan användas både i landskaps- och porträttmod, dess höga upplösning (768x1024 pixels), hög bildrepetitionsfrekvens (ger flimmerfri bild) och rättvänd eller inverterad text förstärker systemets ergonomiska egenskaper.



Förutom tangentbord och bildskärm består ABC 1600 av datorenheten som förutom processorenhet och videoenhet även innehåller en winchesterenhet för lagring av operativsystem, programvaror och data samt en 130 mm flexskiveenhet för säkerhetskopiering eller transport av data och programvaror till eller från systemet. Här ingår även en nätdel (150 W), som ger kraftmatning till datordelens olika delar.

Datordelens huvudblock är följande:

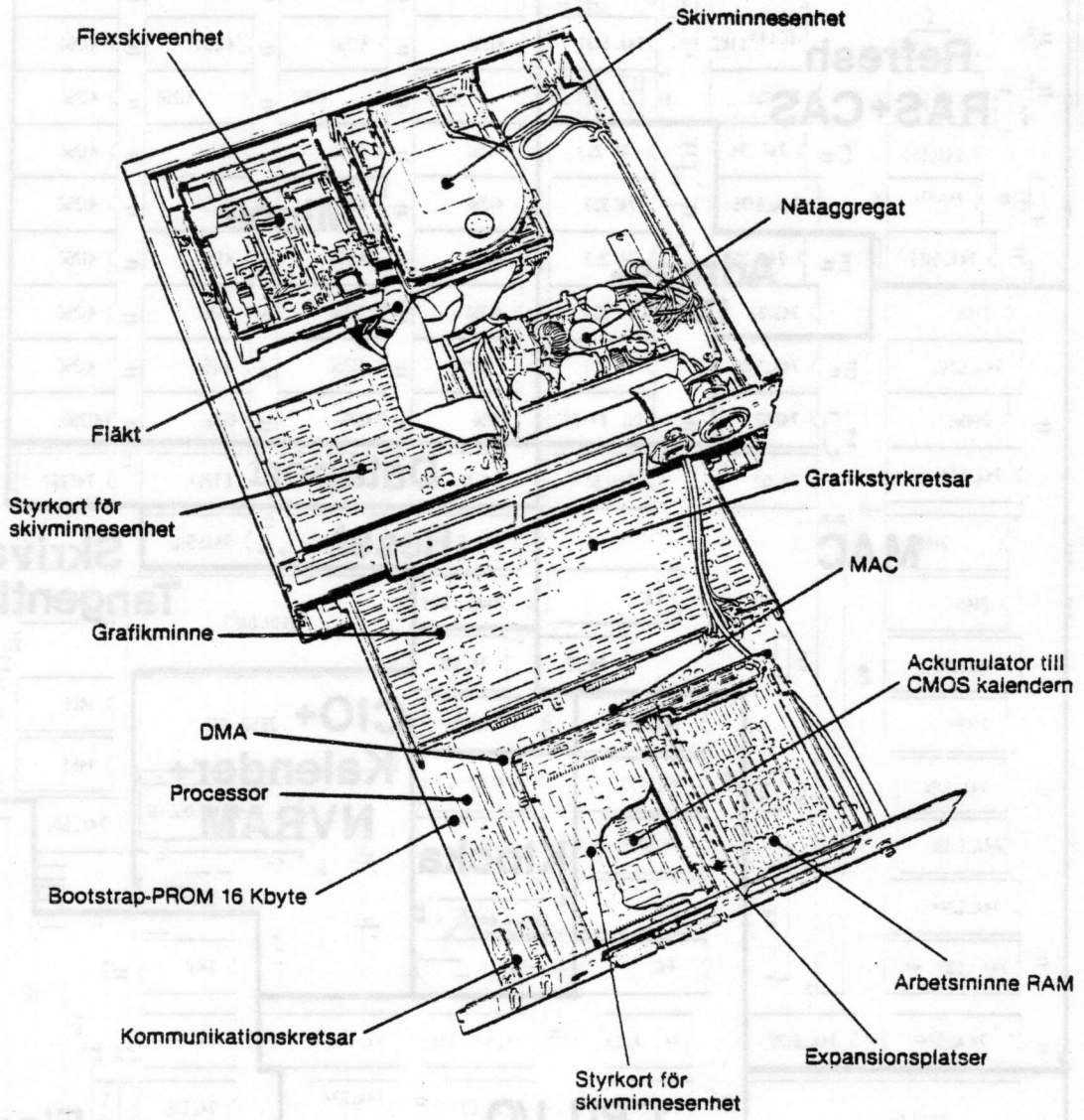
Processordel

Videodel

Winchesterenhet

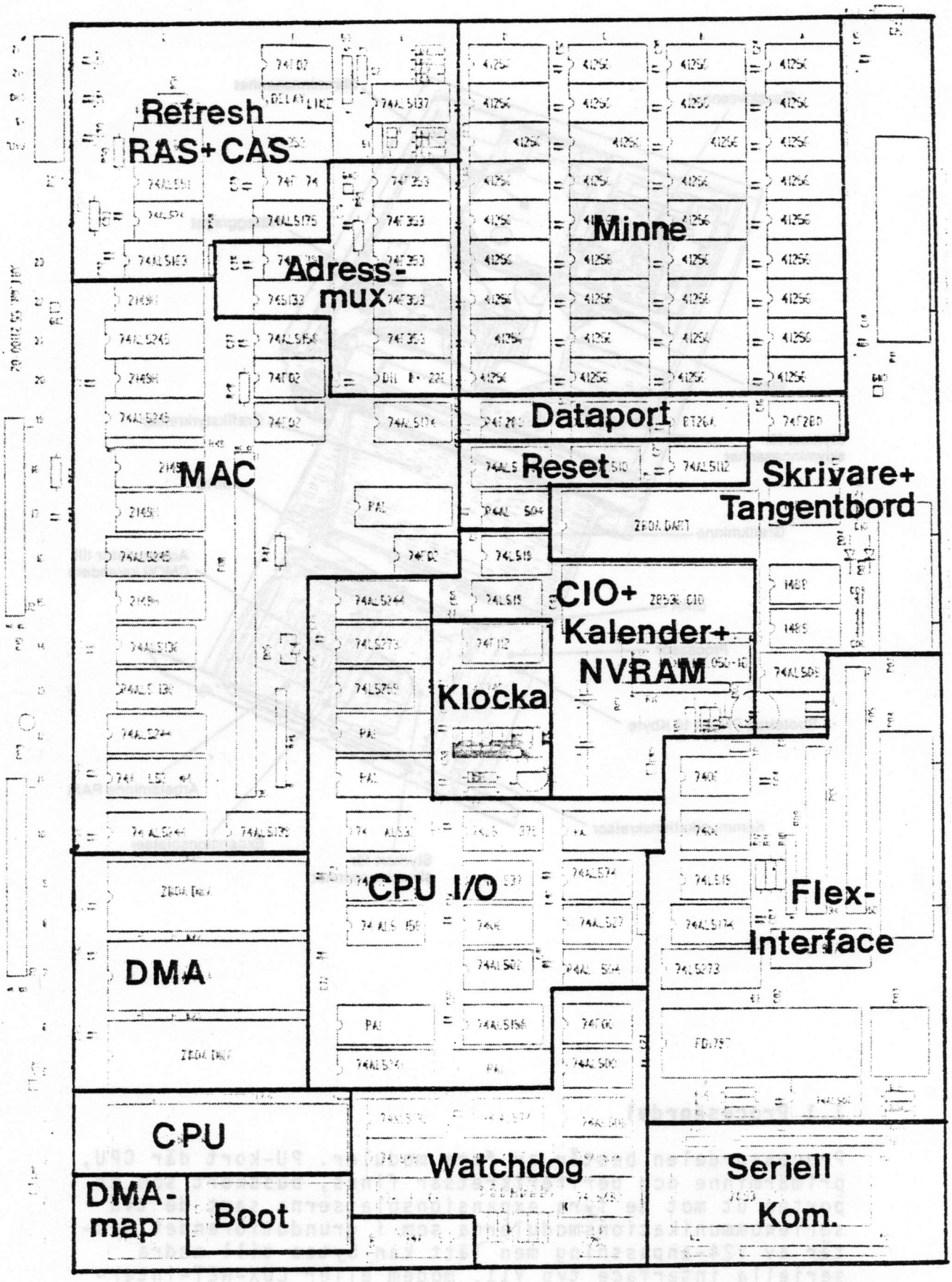
Flexskiveenhet

Nätdel

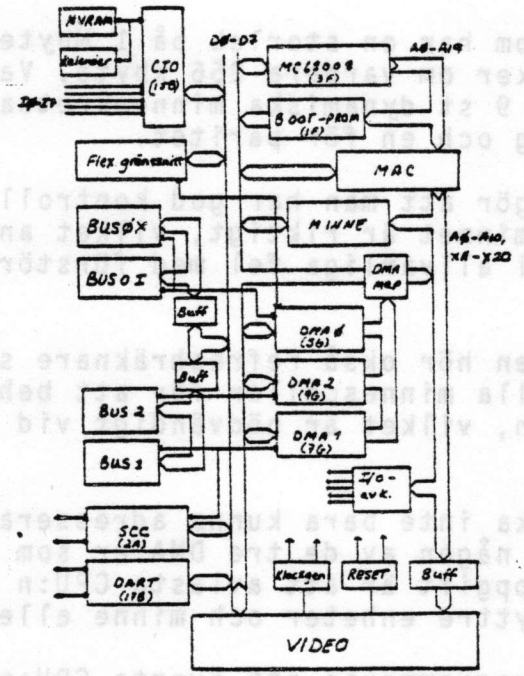


1.1 Processordel

Processordelen består av fyra moduler. PU-kort där CPU, primärminne och periferikretsar finns, busskort som är porten ut mot de fyra expansionsplatserna samt de två seriekommunikationsmodulerna som i grundutförandet består av V24-anpassning men lätt kan bytas till andra seriella interface typ V11, modem eller LUX-NET-interface.


Refresh
RAS+CAS
Adress-
mux
Minne
Dataport
MAC
Reset
Skrivare+
Tangentbord
CIO+
Kalender+
NVRAM
Klocka
CPU I/O
Flex-
Interface
DMA
CPU
Watchdog
Seriell
Kom.
DMA-
map
Boot

PU-delen



Den avgjort största modulen av dessa är PU-kortet. Här sitter CPU:n MC68008 som styr och administrerar all hantering av data och program i systemet. Den tillhör den nya generationens asynkrona CPU:er som hanteras med kraftfulla instruktioner och ger bra stöd för flera-användar-OS typ ABCenix.

MC68008 kan hantera 1 Mbyte direkt adresserbart minne men får i ABC 1600 hjälp med adresseringen av den sk MAC:en.

MAC står för "Memory Access Control", vilket betyder att den omvandlar de adresser som CPU:n sänder ut till fysiska minnesadresser och gör på detta sätt att den programvara som körs av CPU:n inte själv behöver veta vilken minnesarea den egentligen arbeter. Detta gör det enklare att kombinera olika programvaror då de ska användas samtidigt i samma system.

MAC:en styrs av operativsystemet som med MAC:ens hjälp snabbt kan koppla om mellan 16 olika arbeten (TASK), som var och en arbetar i sin egen minnesarea. Den kan också kontrollera om någon process försöker arbeta mot minnesarea som inte har tilldelats denna process samt om processen försöker skriva i någon skrivskyddad minnesarea, exv programarea. På detta sätt hjälper MAC:en till att öka säkerheten i systemet så att inga data förstörs p g a otillåten skrivning.

MAC:en ökar också CPU:ns adresseringsområde från 1 Mbyte till 2 Mbyte, där den första delen (0-1 Mbyte) används för primärminnesadressering under det att den andra delen (1-2 Mbyte) adresserar grafikminne och I/O.

Primärminnet som har en storlek på 1 Mbyte utgörs av fyra minnesbanker om vardera 256 Kbyte. Varje minnesbank består av 9 st dynamiska minneskretsar varav åtta för datalagring och en för paritet.

Paritetsbiten gör att man har god kontroll på att data som hämtas ur minnet är riktigt, vilket annars skulle kunna leda till allvarliga fel med förstörda data som följd.

Till minnesdelen hör också refreshräknare som kontinuerligt läser i alla minnesceller för att behålla de data som skrivits in, vilket är nödvändigt vid dynamiska minnen.

Primärminnet ska inte bara kunna adresseras av CPU:n utan även från någon av de tre DMA:er som finns i systemet. Deras uppgift är att avlasta CPU:n vid dataöverföring mellan yttre enheter och minne eller vice versa.

DMA:erna kan programmeras att överta CPU:ns funktion, dvs de kan generera nödvändiga adresser och kontrollstrobar för att hantera minne och I/O-enheter.

De tre DMA:erna är uppdelade så att DMA0 hanterar expansionsbussarna BUS0I och BUS0X samt den inbyggda flexskiveenheten, DMA1 seriell kommunikation via SCC:n samt BUS1 och DMA2 busskontakt BUS2 där normalt winscheterinterfacet är anslutet.

Då DMA:ernas adresseringsförmåga endast är 64 Kbyte får de hjälp av en speciell DMA-MAP som utökar adressområdet till 2 Mbyte vilket för att de kan nå alla I/O och minnesareor vi en dataöverföring.

Den seriella datakommunikationen hanteras av kommunikationskretsarna SCC och DART.

DART:en svarar för den enklare typen av kommunikation och har sina två portar anslutna till skrivarutgången samt tangentbordet. Kommunikationen med skrivaren sker via en 25-pol V24-anslutning under det att tangentbordet kan anslutas på två olika sätt. Dels via bildskärmen genom samma kablage som H-synk och V-synk distribueras men också direkt via en 7-pol DIN-kontakt på själva processorkortet.

SCC:n hanterar den mer avancerade kommunikationen och kan användas för kommunikation med andra datorsystem eller för anslutning av terminaler då flera vill använda datorkraften i ABC 1600. SCC:n kan till skillnad från DART:en hantera synkron kommunikation med NRZ/NRZI-kodning och har inbyggda baudrategeneratorer för klockgenerering.

Dataöverföring till och från SCC:n sker via de två kommunikationsmodulerna som enkelt kan bytas ut då speciella behov av seriell kommunikation föreligger.

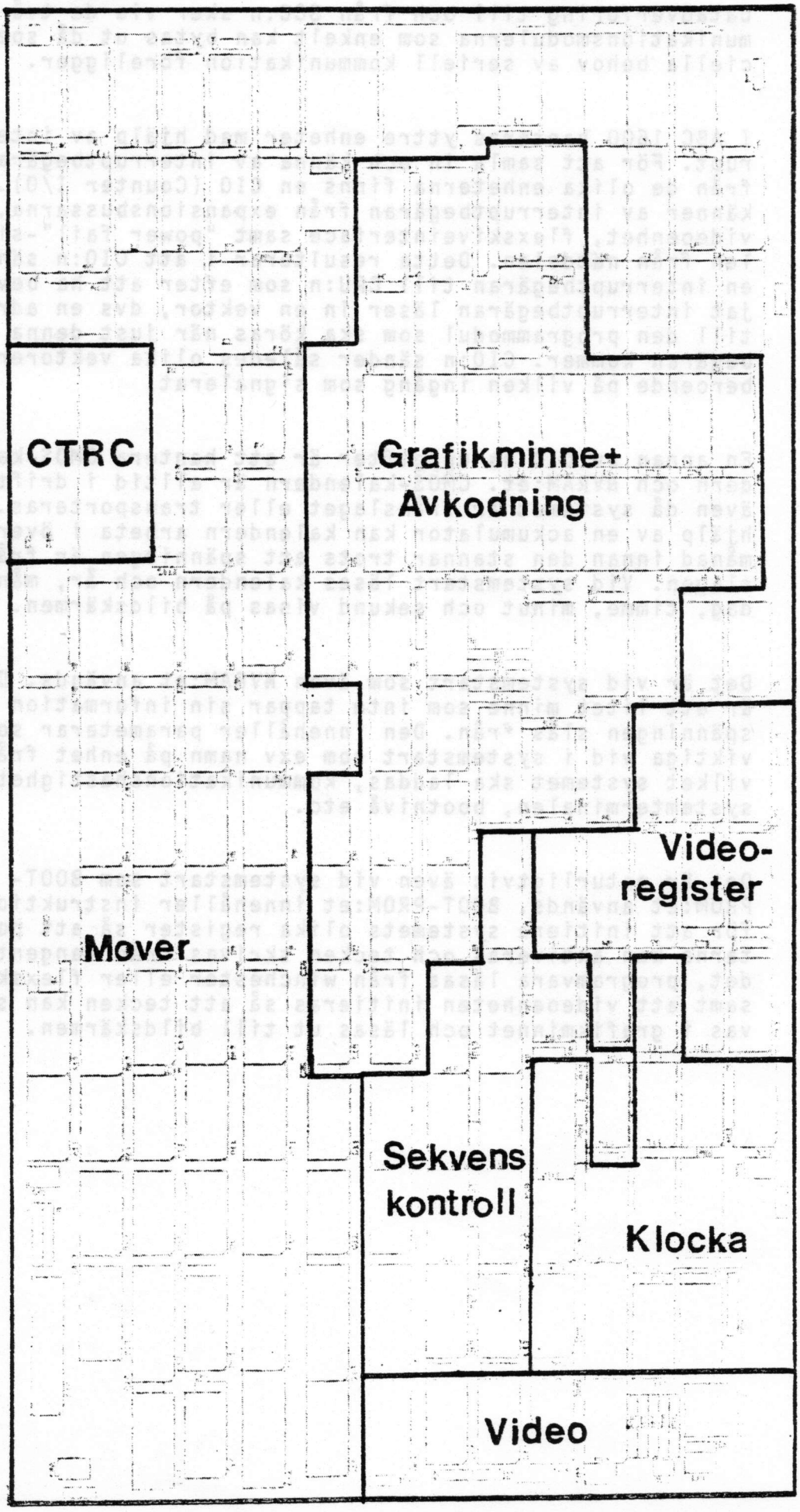
I ABC 1600 hanteras yttre enheter med hjälp av interrupt. För att samla in och känna av interruptbegäran från de olika enheterna finns en CIO (Counter I/O). Den känner av interruptbegäran från expansionsbussarna, videoenhet, flexskiveinterface samt "power fail"-signalen från nätdelen. Detta resulterar i att CIO:n sänder en interruptbegäran till CPU:n som efter att ha beviljat interruptbegäran läser in en vektor, dvs en adress till den programmodul som ska köras när just denna begäran kommer. CIO:n sänder således olika vektorer beroende på vilken ingång som signalerat.

En annan av CIO:ns uppgifter är att hantera CMOS-kalendern och NVRAM:et. CMOS-kalendern är alltid i drift, även då systemet är frånslaget eller transporteras. Med hjälp av en ackumulator kan kalendern arbeta i över en månad innan den stannar trots att spänningen är frånslagen. Vid systemstart läses kalendern och år, månad, dag, timme, minut och sekund visas på bildskärmen.

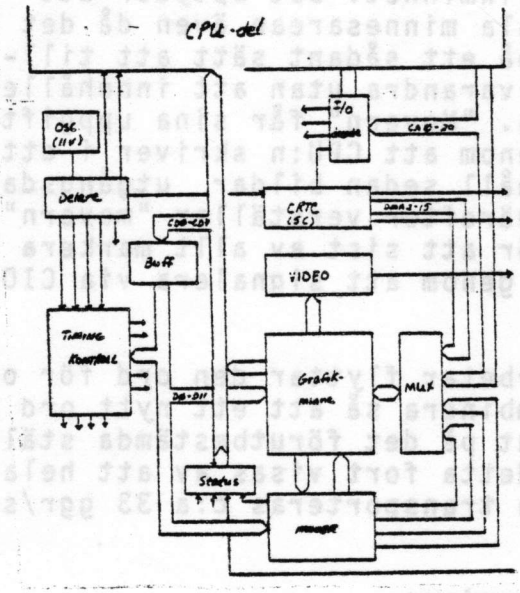
Det är vid systemstart som även NVRAM:et används. Det är ett litet minne som inte tappar sin information då spänningen slås från. Den innehåller parameterar som är viktiga vid i systemstart som exv namn på enhet från vilket systemet ska laddas, kommunikationshastighet mot systemterminalen, bootnivå etc.

Det är naturligtvis även vid systemstart som BOOT-PROM:et används. BOOT-PROM:et innehåller instruktioner för att initiera systemets olika register så att portarna kan aktiveras och tecken skrivas från tangentbordet, programvara läsas från winchester eller flexskiva samt att videoenheten initieras så att tecken kan skrivas i grafikminnet och läsas ut till bildskärmen.

1.2 Videodel


CTRC
Grafikminne+
Avkodning
Video-
register
Mover
Sekvens
kontroll
Klocka
Video

Videodelen ansluts till processordelen via två 40-poliga stiftlister. Härigenom kan adress- och databuss samt kontrollsignaler och klockor distribueras mellan modulerna. Videodelens uppgift är att lagra den bild som ska visas på bildskärmen och i lämplig takt sända data till denna tillsammans med synkroniseringssignaler. På videoenheten finns också systemets klockgenerering samt en "mover" vars uppgift är att snabbt förflytta data från en position till en annan i grafikminnet.



Klockgeneringen består av en oscillatorkrets som lämnar en 64MHz-klocka varifrån alla andra klockfrekvenser utgår. Genom delare och synkroniseringssteg ser den till att alla delar i systemet aktiveras synkront och i den takt som är lämplig.

Sekvenskontrollen hanterar signaler till grafikminnet så att då olika enheter arbetar mot det genereras olika timing. Normalt styrs den så att en tidsfördelning erhålls mellan bildskärmsutläsning och MOVER/CPU-tid. Men då bildskärmsutläsning inte förekommer, exv under synkåtergång eller vid släckt bildskärm, får MOVER/CPU hela tiden för sig själva.

Grafikminnet som i standradförande är på 128 Kbyte (organiserat 64Kx16bit) är helt pixelorienterat dvs det lagrar en unik information om varje pixel. Detta gör att man kan blanda text och grafik som man önskar. Minnet kan sägas vara uppdelad i två delar, synlig area och dold area. Den synliga arean är på 96 Kbyte och är den minnesdel som visas på bildskärmen under det att den dolda arean används för att lagra tecken och symboler som sedan snabbt kan förflytta till den synliga delen med hjälp av "movern".

Grafikminnet kan byggas ut till att omfatta en total area av 512 Kbyte vilket räcker för att lagra fyra bilder eller för lagring av symboler vid exv CAD-program.

Minnet adresseras från flera olika håll, dels måste CRT-Controllern som hanterar de adresser som via grafikminnet läser ut data till bildskärmen. Dels måste CPU:n komma åt att läsa och skriva/modifiera data, samt "movern" som måste ge en frånadress (anger var data ska hämtas) och en till-adress (anger var data skall skrivas).

"Moverns" uppgift är att snabbt flytta rektangulära ytor av valfri storlek till/från valfri plats inom ramen för grafikminnet. Det betyder att "movern" kan arbeta inom hela minnesarean även då det är utbyggt. Data flyttas på ett sådant sätt att till- och frånytan kan överlappa varandra utan att innehållet förstörs eller påverkas. "Movern" får sina uppgifter om förflyttningen genom att CPU:n skriver i ett antal register vars innehåll sedan bildar utgångsdata för datatransporten. Därefter verställer "movern" hela förflyttningen för att sist av allt markera till CPU:n att allt är klart genom att signalera via CIO:n.

Då "movern" arbetar flyttar den ord för ord genom att skifta och kombinera så att ett nytt ord bildas. Detta skrivs sedan ut på det förutbestämda stället. Att "movern" gör detta fort visas av att hela bildskärmsinnehållet kan transporteras c:a 33 ggr/sek.

1.3 Winchesterminne

UNIX-system i allmänhet och så även ABCenix arbetar ganska mycket mot massminnen, de är så att säga "diskintensiva". Detta gör behovet av ett snabbt massminne med god lagringskapacitet till en nödvändighet. I ABC 1600 finns därför ett skivminne av winchestertyp inbyggt i dataenheten. I sitt grundutförande har detta en lagringskapacitet på 13 Mbyte men kan vid behov förstärkas med yttre skivminnen.

Winchesterenheten är en "slimline"-typ, vilket innebär att den tar liten plats och har låg effektförbrukning. För att styra skivminnet finns en styrenhet, XEBEC 1410, där data kan buffras vid skrivning eller läsning samt en del som upptäcker och korrigeras ev fel vid läsning pga "dropouts" eller liknande. Detta gör också att skivminnet är mycket säkert ur datasynpunkt.

Mellan styrenheten och datorns expansionsbuss sitter ett interfacekort som kallas SASI-interface. Dess uppgift är att omvändla ABC 1600:s styrsignaler så att de kan tolkas av styrenheten och omvänt.

1.4 Flexskivminne

Flexskiveenheten är till för att ta säkerhetskopia av de data som finns lagrade på winchesterenheten men också att tranportera data och program till och från den. Flexskiveenheten är av "slimlinetyp" och har en lagringskapacitet på 640 Kbyte /flexskiva. Den styrs av det inbyggda interfacet på processorkortet som också via en yttre 25-pol kontakt kan hantera ytterligare två flexskiveenheter av valfritt format. Parametrar som skivstorlek, antal spår, enkelsidig/dubbelsidig samt filformat hanteras helt med hjälp av mjukvaran då skivan monteras, vilket gör att byglingar och andra mekaniska omställnigar ej behövs.

1.5 Nätdel

Kraft till systemet levereras av en primärswitchad nätdel som ger tillräckligt med kraft för att försörja hela datordelen inkl skivminnen. Den har även reservkapacitet för att driva expansionskort och inbyggnadsmoduler som kan förekomma i ett utbyggt system. Nätdelen lämnar totalt en effekt av 150W som kan fördelas valfritt mellan +5 och +12V men kan även lämna en mindre strömmängd -12V. Den motsvarar de säkerhetskrav som uppställts av exv SEMKO men även kraven på luftoch nätburen strålning enligt VDE 0871/B. Detta gäller för övrigt hela ABC 1600-systemet.

2 Demontering/Montering Datorenhet ABC 1600

Datorenhet ABC 1600 är konstruerad med tanke på att service och utbyggnad ska vara enkel att utföra. Den är uppbygd helt i metall (aluminiumprofil) för att ge god stabilitet och goda skärmningsegenskaper mot strålning.

För att behålla dess egenskaper är det viktigt att DU som servicetekniker ser till att alla skruvar och andra förband är korrekt monterade och utförda för att enheten även efter ingreppet bibehåller sina egenskaper.

Vid vissa kabelmontage som hör till spänningsmatningen är det även av vikt att kablage, brickor och muttrar monteras i rätt ordning för att de säkerhetsnormer enligt vilken apparaten är klassad skall fortsätta att gälla.

Var därför observant på den ursprungliga monteringen.

2.1 Demontering av dator- och videodel.

1. Lossgör alla kablage till datorenheten.

2. Demontera de åtta märkta skruvarna på datorenhetens baksida samt även luckan. (Se bild 1).

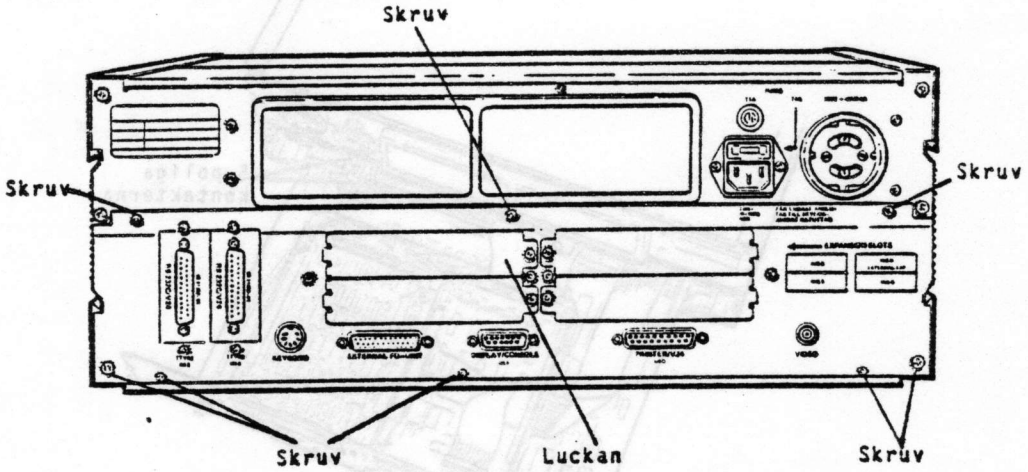


Bild 1

Enheten kan nu försiktigt föras utåt c:a 20 cm likt en byrålåda (bild 2).

> OBS!
> Se till att alltid ha stöd för datordelen mot bordsytan.

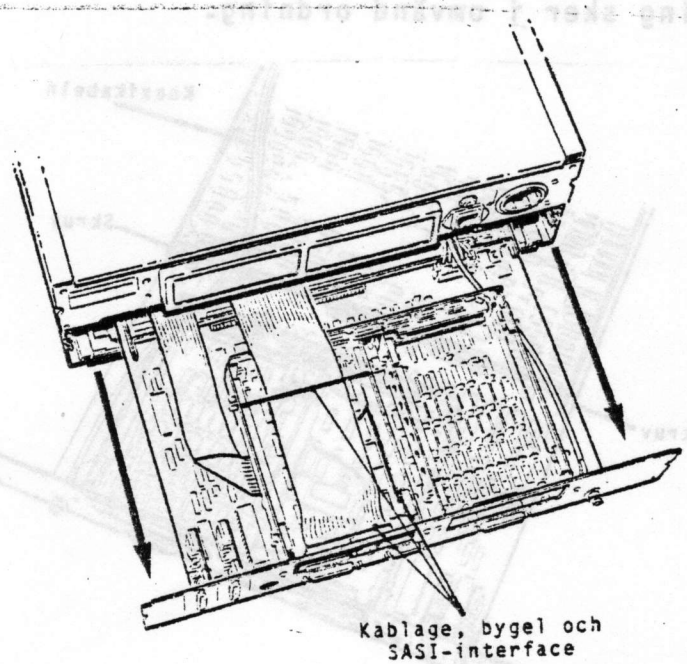


Bild 2

3. Demontera det 50-poliga kablaget, bygel och SASI-interfacet.

4. Demontera den 40-poliga kontakten samt de två 5-poliga kontakterna för spänningsmatning. (Bild 3)

   Dator och videodel kan nu försiktigt avlägsnas från datorlådan.

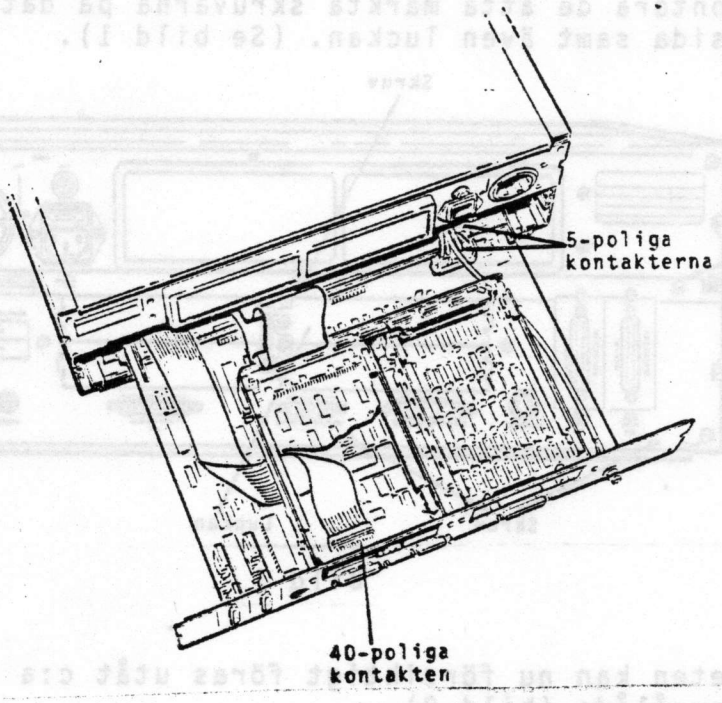


Bild 3

5. Skilj dator och videodel åt genom att lossa de två skruvarna och koaxialkabeln. Drag sedan försiktigt isär de två modulerna.(Bild 4).

   Montering sker i omvänd ordning.

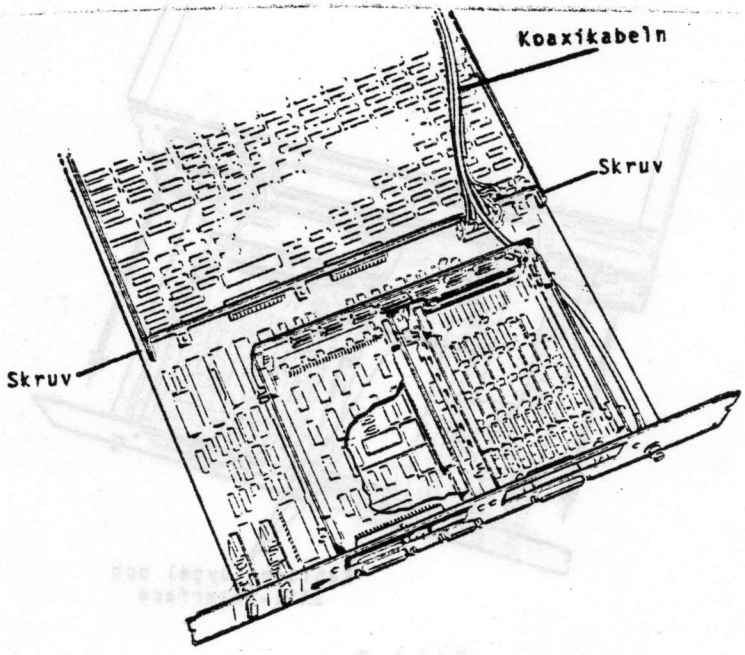


Bild 4

2.2 Demontering av skivenheterna och nätdel

1. Demontera de 13 skruvarna på datorenhetens baksida (Bild 5).

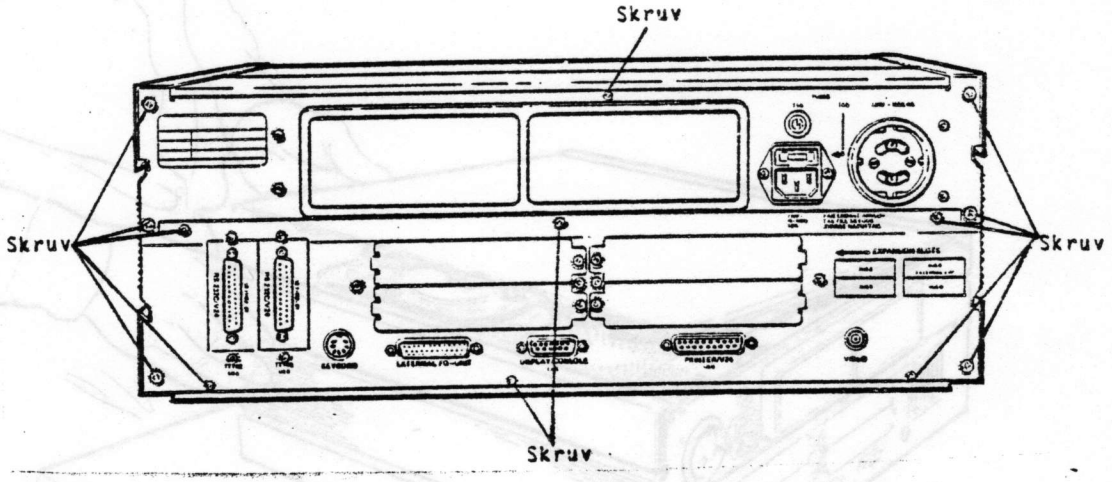


Bild 5

2. Lossa fronten genom att låta datorenhetens front skjuta över bordskanten några cm och lossa de två skruvarna på frontens undersida. (Bild 6)

3. För frontens nedre del framåt och lyft uppåt så att den kan lyftas av.

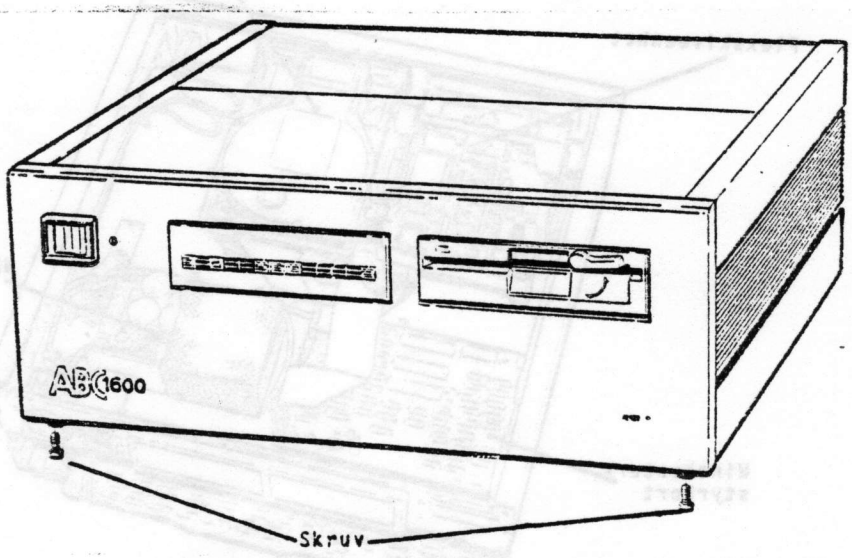


Bild 6

4. Datorenhetens gavlar kan nu skjutas lätt isär och locket demonteras. (Bild 7)

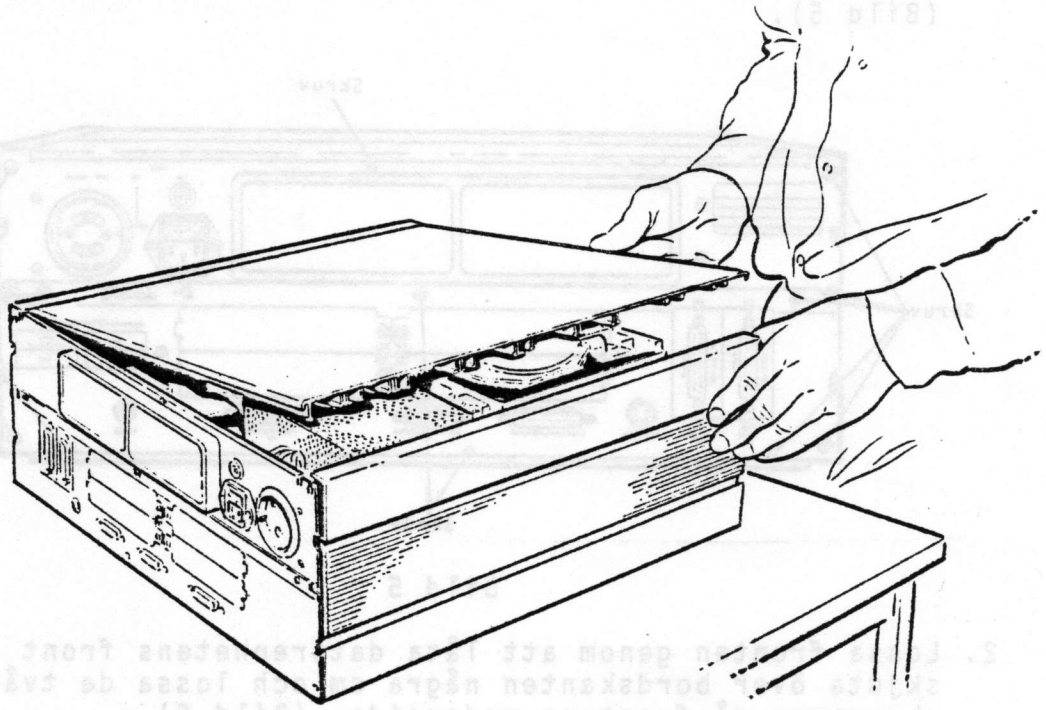


Bild 7

Flexskiveenhet, winchesterenhet, winchesterstyrkort och nätdel kan nu demonteras. (Bild 8).

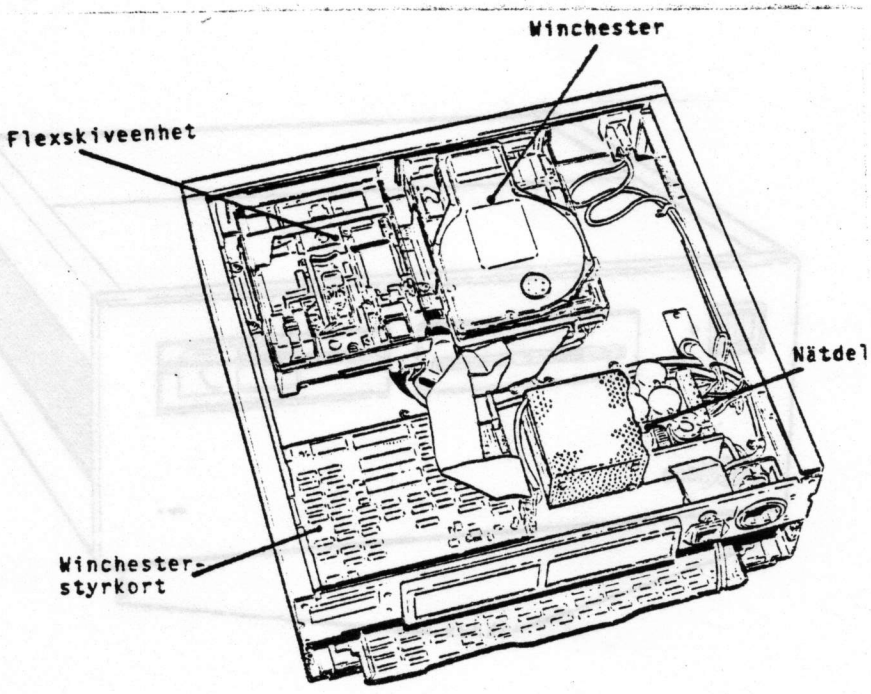


Bild 8

3 Tekninskbeskrivning - datorenhet

3.1 Processorenhet

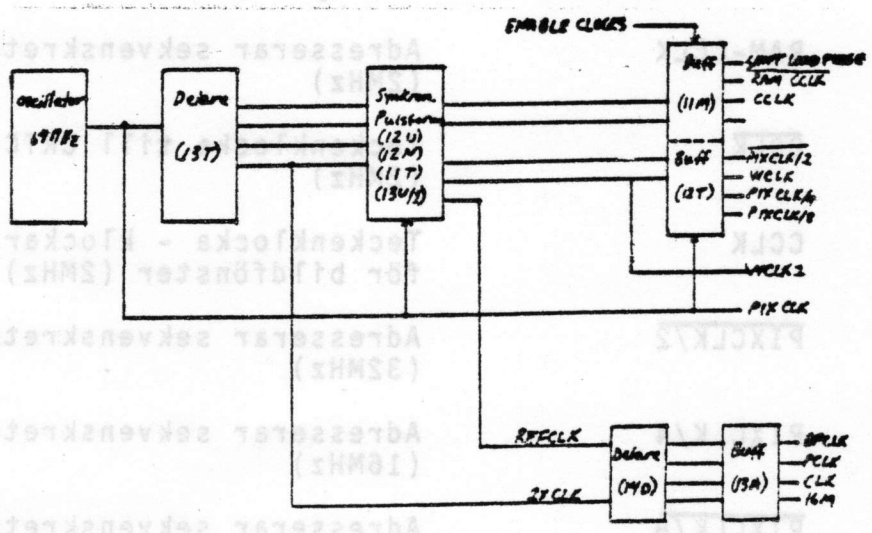
3.1.1 Systemklocka

Systemklockans uppgift är att leverera klockor och pulståg till systemet så att dess olika delar kan utföra sina uppgifter synkront med varandra eller i en förutbestämd sekvens. I ett så komplext datorsystem som ABC 1600 är det särskilt viktigt att alla olika klockfrekvenser är säkert låsta till varandra för att säkerställa enhetens funktion. Risken är annars uppenbar att s k "löptidsfenomen" beroende på exempelvis temperatur, kretstyp och kretsleverantör kan omintetgöra hela systemts funktion.

I ABC 1600 har detta lösts på så sätt att alla delade frekvenser via vippor är låsta till den ursprungliga klockan (64MHz).

Klockan, som till största delen är belägen på videodelen kan sägas bestå av fem huvudblock:

- oscillator
- delare
- synkronisering (Belägen på Videokortet)
- port för videoklockor
- delare för CPU-klockor (Belägen på PU-kort)



Videodelens klockor

Oscillatorn består av en fast oscillatorkrets (11V) med inbyggd kristall, som levererar en symetrisk utsignal på 64 MHz. För att oscillatorn ej skall lastas för hårt buffras utsignalen i inverterarna 11U/3 och 4.

Signalen används nu för att klocka räknaren (13T) på vars utgångar de delade frekvenserna 32MHz (Pin 14), 16MHz (Pin 13), 8 MHz (Pin 12) och 4 MHz (Pin 11) bildas. Dessa signaler går nu två vägar, dels till bufferten (13N/2), dels till NAND-grinden 13U/1 på vars utgång en låg signal genereras för var sextonde klockpuls.

Efter diverse synkronisering och fördröjning i vipporna (11T) bildar denna signal SHIFT-LOAD-PULSE som laddar videoskiftregistret. Från bufferten (13/N2) går de delade signalerna till datavipporna (12N och 12V) för fördröjning och synkronisering till 64MHz-klockan, för att sedan påföras portkretsarna 11M och 12T.

Portkretsarnas uppgift är att blockera vissa klockor till videodelen under uppstart av systemet innan registren för sekvenskontroll är programmerade. När dessa klockor ska släppas fram bestäms av initieringsprogrammet, som via I/O-hanteringen aktiverar stroben $\overline{\text{ENABLE-CLOCKS}}$. Denna strob påverkar set-ingången på D-vippan 13M/I så att portkretsarna öppnas.

Med hjälp av resetsignalen $\overline{\text{RES1}}$ till ovannämnda vippa ser man till att portkretsarna är blockerade vid systemstart.

De klockor som berörs av portkretsarna är:

SHIFT-LOAD-PULSE	Laddar videoskiftregistret med nytt ord.
RAM-CCLK	Adresserar sekvenskretsarna (2MHz)
$\overline{\text{CCLK}}$	Teckenklocka till CRTC (2MHz)
CCLK	Teckenklocka - klockar vippa för bildfönster (2MHz)
$\overline{\text{PIXCLK/2}}$	Adresserar sekvenskretsarna (32MHz)
PIXCLK/4	Adresserar sekvenskretsarna (16MHz)
$\overline{\text{PIXCLK/8}}$	Adresserar sekvenskretsarna (8MHz)
WCLK	Adresserar sekvenskretsarna (4MHz)

Två klocksignaler påförs videodelen omedelbart efter spänningstillslag nämligen WCLK (4MHz) samt PIXCLK (64MHz).

Till PU-kortet levereras klockorna 2*CLK (16MHz) och REFCLK 8MHz) av vilka den senare är synkroniserad till 64MHz-klockan.

På PU-kortet sitter också det femte blocket för klockgenerering och innehåller delare, buffert och inverterare för att generera rätta frekvenser och faslägen på de klockor som ska användas av CPU, periferikretsar, businterface och minnen.

Processordelens klockor

På PU-kortet buffras och inverteras klocksignalen 2*CLK och bildar signalen 16M som används av flexskiveinterfacets kontrollkretsar samt för att klocka J/K-vipporna 14D/1 och 2. I 14D/2 påföres REFCLK till J-ingången, vilket resulterar i att utgångarnas frekvens blir 8MHz.

Från den inverterade utgången får man klockan PRECLK, som används för klockning av PAL-kretsarna 10C och 12E och via buffring och invertering bildas klockan CLK d v s inverterad CPU-klocka. Den icke inverterade utgången på 14D/2 bildar insignal till vippan 14D/1 samt efter invertering och buffring CPU-klockan CLK(8MHz).

Efter vippan 14D/1 har frekvensen delats till 4MHz och efter buffring och invertering samt fördelning i 13D bildas 4MHz-signalerna PCLK och PCLK till periferikretsar och deras grindlogik samt BPCLK och BPCLK till bussinterfacen och deras logik.

Då vissa periferikretsar exempelvis DMA kan vara känsliga vad gäller klocksignalers nivå har en hjälptransistor (T01) införts parallellt med bufferten till PCLK-signalen.

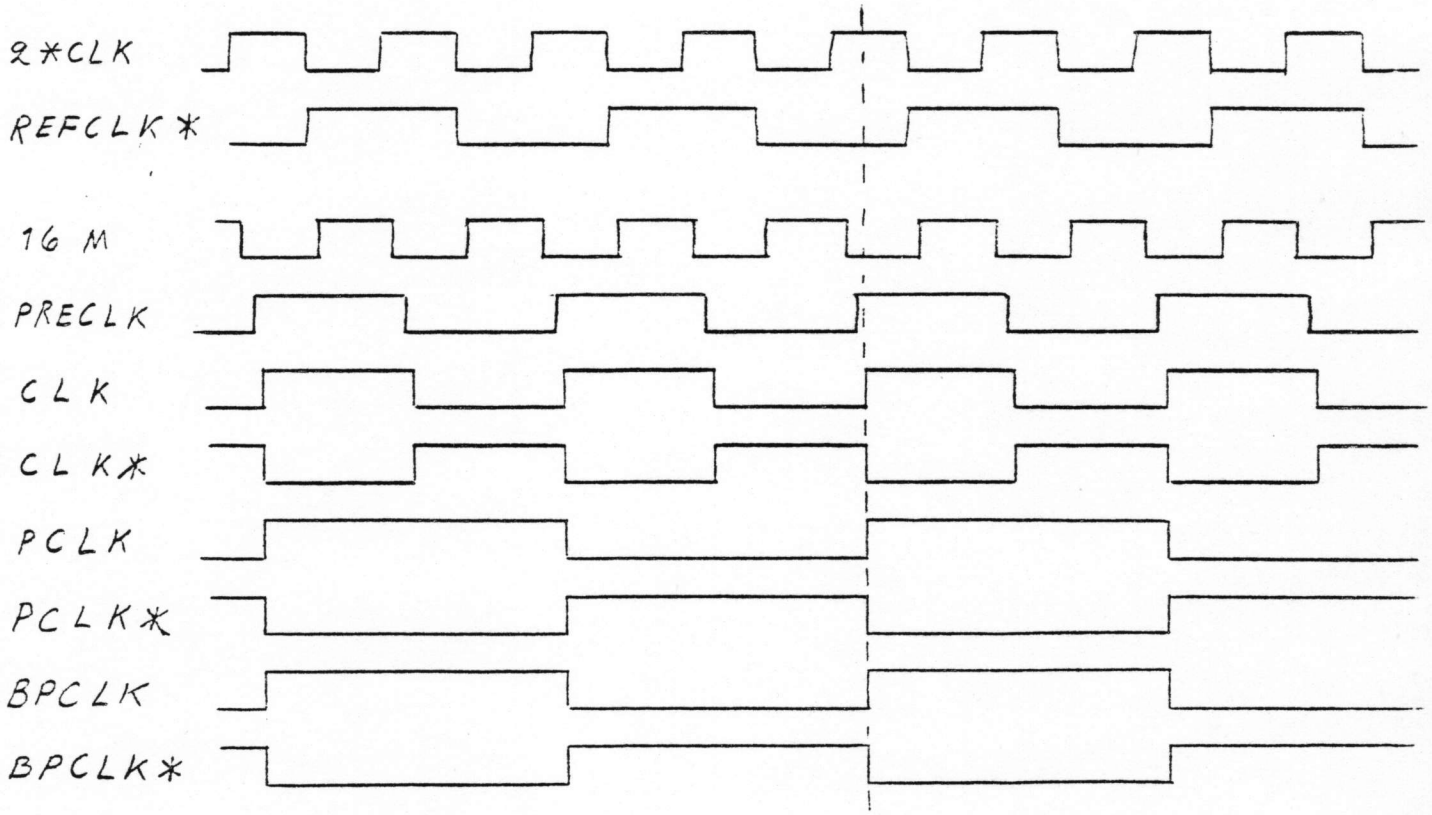
Ytterligare en klocksignal bör nämnas i detta sammanhang, nämligen E-klockan, som genereras i CPU:n och egentligen är avsedd för periferikretsar i 6800-familjen. Klockan som är 1/10 av CPU-klockan (här 800KHz) användas i ABC 1600 till "watchdog" och refreshklocka för dynamiska minnen.

Fig 6

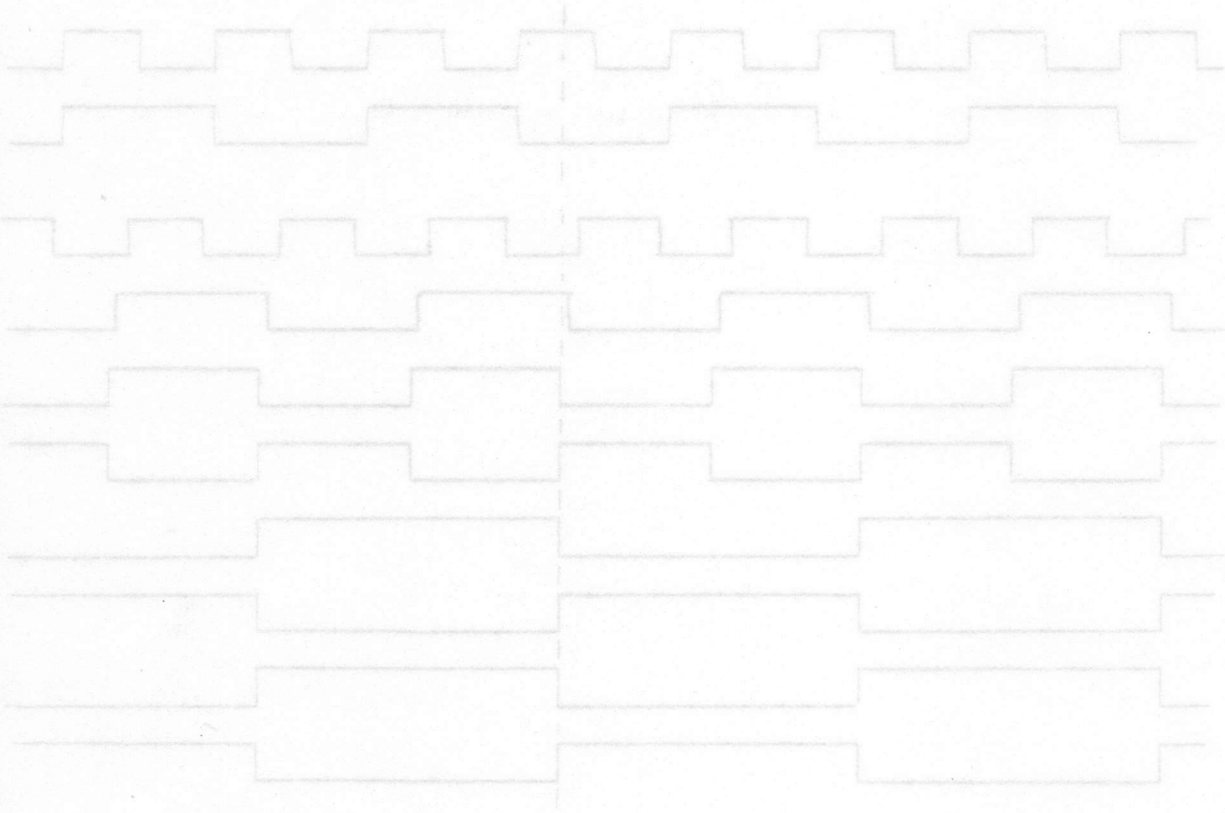
fig 6.)

Klocksignaler på CPU-kortet (schemablad 1)

1 cm = 31,25 ns.



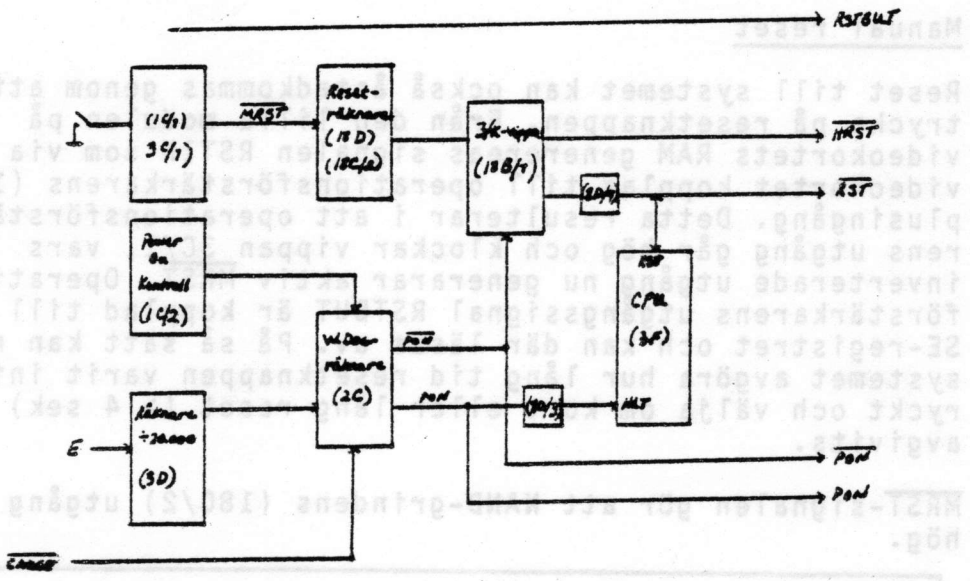
CLK, CLK*, PCLK, PCLK*, BPCLK och BPCLK* växlar nivå samtidigt. PRECLK växlar nivå cirka 4 ns före dessa.



3.1.2 Systeminitiering

För att säkerställa att ett visst bestämt läge råder efter spänningstillslag eller efter en tryckning på resetknappen finns initieringslogik som ser till att alla periferikretsar och register sätts i sina ursprungslägen. Logiken skall också se till att resetpulsens längd är tillräcklig för att exempelvis CPU:n skall få ett säkert starttillstånd efter spänningstillslag. I ABC 1600 finns fyra olika sätt att initiera systemet:

- Spänningstillslag = hela systemet initieras
- Tryck på resetknapp = används då systemet låst sig, eller vid övergång till BOOT-nivå 0 (vid lång reset)
- "Watchdog". = programvaran spårat ur så att läsning av CAUSE-registret ej sker.
- Körning av instruktionen RESET = genererar reset till alla enheter anslutna till CPU:ns resetledning. Obs! CPU:n påverkas ej.



Spänningstillslag

Vid spänningstillslag kommer spänningen att byggas upp enligt en ramp- eller exponentialfunktion, ej språngmässigt. Detta gör att operationsförstärkarens (1C/2) plusingång med sin linjära spänningsdelare snabbare når en högre nivå än minusingången, där först en zenerdiods genombrottsspänning ska uppnås innan denna ingång skiljs från 0-nivå.

Detta resulterar i att operationsförstärkarens utgång i första ögonblicket ger en hög utnivå. Efter ytterligare en kort stund, då matningsspänningen är på väg mot sitt slutvärde, passerar minusingångens nivå plusingången, vilket gör att operationsförstärkarens utgång återgår till låg nivå.

Den positiva puls som nu bildats räcker till att nollställa dekadräknaren 2C/2 samt att sätta utgångarna på räknaren 2C/1 till värdet 9. Det betyder att utgång 3 pin 7 antar en hög nivå och signalerna PON (Power ON) och $\overline{\text{PON}}$ blir aktiva.

Dessa signalers varaktighet bestäms av E-klockan (1/10 av CPU:ns dvs 800 KHz) som via delaren 3D (delas 20.000 ggr) klockar räknaren 2C/2, vilken då denna räknat till åtta kommer den att generera klockpuls till 2C/1, som nu slås över till noll och PON-signalerna deaktiveras.

Tiden blir 0,4 sek. vilket är tillräckligt då CPU:n, som behöver längst tid för systemreset efter spänningstillslag och måste ha minst 100ms.

Då PON-signalerna är aktiva sätts J/K-vippan 18B/1 så att resetsignalerna $\overline{\text{RST}}$ och $\overline{\text{HRST}}$ blir aktiva samt via inverteraren 8D/3 även $\overline{\text{HLT}}$-ingången på CPU:n, vilket är nödvändigt för att CPU:n skall göra en systemstart. PON-signalen nollställer också det s k CAUSE-registret (14E).

Manual reset

Reset till systemet kan också åstadkommas genom att trycka på resetknappen. Från den lilla modulen på videokortets RAM genereras signalen RSTIN som via videokortet kopplas till operationsförstärkarens (1C/1) plusingång. Detta resulterar i att operationsförstärkarens utgång går hög och klockar vippan 3C/1, vars inverterade utgång nu genererar aktiv $\overline{\text{MRST}}$. Operationsförstärkarens utgångssignal RSTBUT är kopplad till CAUSE-registret och kan där läsas av. På så sätt kan nu systemet avgöra hur lång tid resetknappen varit intryckt och välja om kort eller lång reset (> 4 sek) avgivits.

$\overline{\text{MRST}}$-signalen gör att NAND-grindens (18C/2) utgång går hög.

OBS!
RC-utgången på räknaren 18D ligger i vila på hög nivå vilket gör att räknaren 18D ej längre blockeras och vid första klockpuls kommer nu RC-utgången att gå låg.

Då RC-utgången påverkar J/K-ingångarna på vippan 18B/1 kommer denna att slå om vid nästa klockpuls och generera $\overline{\text{RST}}$ och $\overline{\text{HRST}}$ till systemet. $\overline{\text{HRST}}$-signalen kommer nu att nollställa vippan 3C/1 och signalen MRST deaktiveras.

Med hjälp av grinden 18C/2 kommer räkningen i 18D att fortsätta tills RC-signalen åter går hög och resetpulsen blir inaktiv. Tiden för resetpulsen blir 2 ms.

$\overline{RST}$ och $\overline{HRST}$ - signalen utgår från samma vippa och efter inverteraren 8D/4 har signalerna också samma fasläge, detta gör att reset, som avges genom tryckning på reset-knappen påverkar alla delar av systemet. Genom att RST-signalen genereras via en "open collector"-utgång kan flera utgångar vara förbundna till samma ledning. I detta fall är det CPU:ns dubbelriktade resetanslutning som på detta sätt kan aktivera alla till denna ledning anslutna enheter, exempelvis periferikretsar och businterface.

"Watchdog"

"Watchdog" är den tredje orsaken till att systemreset genereras. Avsikten med "watchdog" är att kontrollera om operativsystemet spårat ur så att den kontinuerliga avsläsningen av CAUSEregistret upphör. CAUSE-signalen, (hur den bildas behandlas senare), som egentligen är lässtroben till detta register, nollställer räknaren 2C/1. Skulle CAUSE-signalen utebli mer än 1,6 sek sker systemreset som genereras genom att PON signalen blir aktiv.

CPU:n genererar reset

Det sista sättet att generera reset är att CPU:n kör en resetinstruktion. Denna reset påverkar alla enheter anslutna till $\overline{RST}$-ledningen. CPU:n har genom programvaran på detta sätt möjlighet att "väcka" yttre enheter utan att påverka de interna registren och de enheter som anslutits till $\overline{HRST}$-signalen. Du kan läsa mera om CPU:ns resetinstruktion i CPU-beskrivningen.

3.1.3 BOOT

BOOT-PROM:ets programvara är den enda som ligger resident lagrad i ABC 1600. Dess huvuduppgift är att initiera datorns olika register så att laddning av operativsystem och programvaror är möjlig. Till de register som måste initieras hör Task-register, segment-RAM och page-RAM i MAC.

Taskregistret är efter reset eller spänningstillslag nollställt, vilket innebär att process "0" (systemets process) samt enablestroben till BOOT-PROM:et (BOOTE) är aktiva.

BOOT-programmet

Segment-RAM och page-RAM sätts upp så att MAC:en blir "transparant" dvs alla CPU:ns adresser går igenom MAC:en utan att förändras (A11 = X11, A12 = X12 osv.). Genom uppsättningen av MAC:en har BOOT-programvaran tillgång till primärminnet och en av de första uppgifterna blir nu att kopiera BOOT-prommets innehåll till RAM-minnet samt att kontrollera tillgänglig minnesarea.

Efter kopiering och testning att kopian blev riktig, ändras TASK-registret så att stroben BOOTE blir inaktiv. Uppstartsförloppet fortsätter med övriga register som specialkontrollregistret, videodelens sekvenskontroll och CRT-Controller. Efter initiering av videodelen aktiveras de från starten blockerade klockorna med stroben CLOCKS ENABLE och efter rensning av grafikminnet också läsning av detsamma med hjälp av DISP ENABLE.

BOOT-programmet måste dessutom initiera periferikretsar som Counter I/O (CIO) för att kunna läsa startparametrar ur NVRAM och CMOS-kalender samt initiera interruptvektorer och DART för att hämta tecken från tangentbordet.

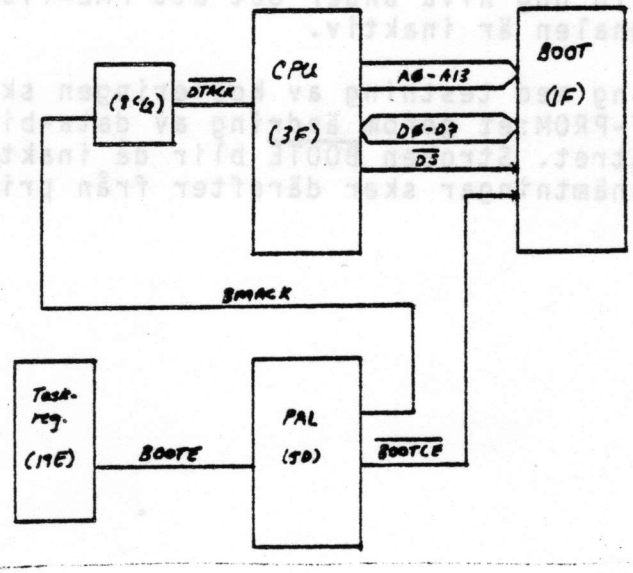
Programvaran måste också läsa av CAUSE-registret, dels för att undvika att en ny uppstartsekvens påbörjas genom att "watchdogen" aktiveras och dels för att avgöra resetknappens läge. Detta har betydelse, då uppstarten beror av en tryckning på resetknappen, då man måste avgöra om en kort eller lång reset (>4 sek) avgivits.

Vid lång reset avbryts uppstartförloppet innan någon programvara laddats från yttre minnesenheter och en promt (M68000 prom) skrivs ut på bildskärmen. Här stoppar nu programmet och väntar på att namnet på den enhet där programvaran finns för fortsättningen av startrutinerna och operativsystemet ska anges från tangentbordet.

För att kunna aktivera yttre enheter måste drivrutiner för att hantera dessa finnas redan i BOOT-programvaran. I ABC 1600 kan start ske från tre olika enheter, den inbyggda winchesterenheten via BUS 2, från flexskiva via det inbyggda flexskiveinterfacet och från magnetband.

De två sistnämnda kräver också DMA-support för att fungera varför både uppsättning av DMA-registren och DMA-MAP måste hanteras av BOOT-programmet.

Då rätt drivrutin aktiverats läses sektor "0" från aktuell enhet där bl a uppgifter om var efterföljande programvaras position finns.



Elektronik runt BOOT-PROM

Nu ska elektroniken runt BOOT-PROM:et förklaras och hur det vid uppstart aktiveras. BOOT-PROM:et är vid sidan av MAC:en den enda del i systemet som påförs adresser direkt från CPU:n vilket är nödvändigt, då MAC:en ej är initierad vid uppstart. Från CPU:n får prommet adresserna A0-A13, databussens åtta bitar samt datastroben $\overline{DS}$ som fungerar som enableingång. (När datastroben genereras kan du läsa om i beskrivningen om CPU:n).

För att BOOT-PROM:et ska aktiveras måste också den andra enableingången aktiveras. Den signalen som kallas $\overline{BOOTCE}$ utgår från PAL 5D och ur PAL-listan får man följande villkor som ska vara uppfyllt för att signalen ska genereras.

$$\overline{BOOTCE} = R * \overline{BOOTE} * \overline{A17} * \overline{A18} * \overline{A19}$$

I klartext betyder detta att R/W-signalen måste stå i läge "Read", samt att $\overline{BOOTE}$ som kommer från TASK-registret (19E) är aktiv. Detta sker genom att resetknappen $\overline{HRST}$ nollställer registret vid uppstart av systemet. Vidare avkänns de tre högsta adresserna från CPU:n för att underlätta kopieringen till primärminnet.

När dessa villkor är uppfyllda kan CPU:n göra en läsning i BOOT-PROM:et, men då CPU 68008 är en asynkron processor kräver den en kvittens signal vid varje skrivning eller läsning.

Vid läsning i BOOT-PROM:et heter kvittenssignalen BMACK (Boot Memory ACKnowledge. Den genereras i PAL 5D och ger via NOR-grinden 8C/2, $\overline{\text{DTACK}}$-signal (Data Transfer ACKnowledge) till CPU:n. I PAL-listan kan du studera vilka villkor som styr BMACK men observera att signalen är aktiv vid hög nivå under det att PAL-listan beskriver då signalen är inaktiv.

Då kopiering med testning av kopieringen skett, deaktiveras BOOT-PROM:et genom ändring av data-bit D6 i TASK-registret. Stroben $\overline{\text{BOOTE}}$ blir då inaktiv och instruktionshämtningar sker därefter från primärminnesarean.

3.1.4 MAC

MAC:ens (Memory Access Control) uppgift är att omvandla CPU:ns adresser till fysiska adresser som sedan används till att adressera minnen eller I/O. Till MAC:ens uppgift hör också att dela upp hela den tillgängliga minnesarean i små delar, så att en process kan tilldelas just så mycket minne som den för tillfället behöver. Den ska också öka ut CPU:ns adresseringsförmåga så att grafikminne och I/O inte stjäl utrymme från primärminnesarean. MAC:en ska också se till att de skilda processerna inte kan komma åt varandras dataareor så att dessa förstörs eller skrivs över.

Genom denna omvandling av adresser och uppdelning kan de olika programvarorna skenbart arbeta i samma minnesareor och behöver på så sätt inte känna till varandras existens, vilket underlättar framställningen av program.

MAC:ens arbetssekvenser

MAC:en består av snabba bipolära RAM-minnen, som kan programmeras till rätt funktion av operativsystemet i systemmod. Den är uppdelad i två nivåer "segment" och "page".

Segment-RAM:et delar upp minnet i 32 Kbytesblock med hjälp av CPU:ns adresser A15-A19 samt processnummer (0-15) vilka bildar adresser till segmentRAM:et.

Från segement-RAM:et genereras sex databitar, som bildar adresser till page-RAM:et tillsammans med A11-A14. Detta gör att minsta minnesblock som kan tilldelas är 2Kbyte. Ur page-RAM:et får vi de omvandlade adresserna X11-X20 samt kontrollbitarna Wp (Write protect) och NONX, (Non existent). Detta gör att vi nu kan adressera upp till 2Mbyte i stället för processorns 1 Mbyte och att vi kan låsa ut vissa areor för skrivning samt för alla former av minnesoperationer.

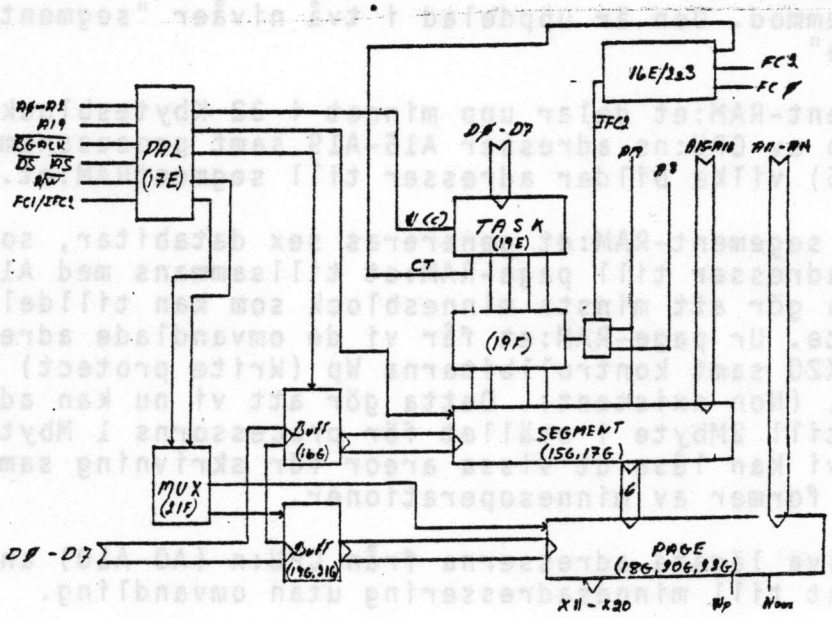
De elva lägsta adresserna från CPU:n (A0-A10) används direkt till minnesadressering utan omvandling.

Man kan arbeta med MAC:en på tre olika sätt.

1. Användarmod = I användarmod d v s då en användarprocess ska utnyttja MAC:en, används CPU:ns alla adresser som adresser till minnet tillsammans med processnumret (1-15). Användarprocesserna adresserar alltid genom MAC:en men kan inte påverka dess innehåll och således inte tilldela sig själv något extra minnesutrymme.

2. Systemmod = Alltid processnr 0. I denna mod används adress A19 (CPU:ns högsta adressbit) för att avgöra om adresseringen sker genom MAC:en, (systemet använder sin data eller programarea i minnet) eller om skrivning eller läsning ska ske i MAC:en. Då A19 är låg adresserar systemet genom MAC:en.

3. Systemmod = A19 hög. Systemet avser nu att uppdaterna innehållet i MAC:en exempelvis tilldela en användarprocess mer minne eller skrivskydda en area. För att kunna programmera även A19:s funktion lånar man tillfälligtvis adressbit A8 som vid denna programmering får ersätta A19. All avkodning som behövs för att kunna generera rätta strobar till MAC:en sker i PAL 17E.



MAC:ens huvudblock är följande:

TASK-register
SEGMENTRAM
PAGERAM
DATAPORT
STROBGENERERING

TASK-register

TASK-registret (19E) är ett register som innehåller numret på aktuell process d v s den process om just nu körs. Med sitt fyrabitarsregister kan de hantera 16 processer (0-15) där nr 0 är avsedd för operativsystemet, de övriga är användarprocesser.

Då operativsystemet vill aktivera en ny process läggs processnummet ut på databussen samt rätt adress så att klockstroben W(c) aktiveras och registret uppdateras. W(C) bildas i PAL-kretsen 17 E och vilka signaler som styr den kan avläsas PAL-listorna under kap. . Vid reset eller systemstart nollställs registret av $\overline{\text{HRST}}$-signalen vilket betyder att process 0 aktiveras.

Innan processnummret påförs segment-RAM:et som adress, passeras en port av NOR-grindar (19F), som portsignal används IFC2, som i detta läge är en invertering av CPU:ns FC2-signal dvs den funktionskod som talar om att processorn arbetar i systemmod. Detta gör att process 0 aktiveras automatiskt då processorn går in i denna mod och man slipper ta upp tid med att först skriva om TASK-registret. Ett undantag är biten CT som också har sitt ursprung i samma register som processnumren.

Med hjälp av NOR-grindarna 16E/2 och 3 och funktionskoderna FC0 och FC2 kan CT påverka processporten så att man i användarmod kan nå systemmodens programvara samtidigt med användarprocessens egna dataarea. Då systemet vill komma åt någon användarprocess dataarea aktiveras, således CT genom skrivning i taskregistret och därefter sker övergång till användarmod genom ändring av statusregistret.

När modifieringen av dataarean är genomförd sker återgång till systemmod via någon instruktion exempelvis TRAP som automatiskt ger systemmod. CT-biten kan nu återställas och normal instruktionkörning återupptas.

Segment-RAM:et som består av de två bipolära RAM-minnena 15G och 17G, adresseras av taskregistrets fyra bitar för processnummer samt av CPU:ns adressledningar A15-19. Sex av de åtta tillgängliga databitarna från minnena kommer nu att bilda nya adresser. Dessa adresser som påförs page-RAM:arna delar upp minnet i 32 Kbytesblock och beroende på vilket datamönster som tidigare skrivits till segment-RAM:et kan nu godtyckliga 32 Kbytesblock kopplas till processerna.

För att aktivera segment-RAM:et måste enable-stroben till minnena vara aktiva. Generellt kan sägas att segment-RAM:et alltid är aktivt då CPU:n har busskontroll.

Enable-strob samt skriv-strob genereras i PAL 17E och finns kommenterad i PAL-listan.

För att skriva och läsa i segment-RAM:et finns en dataport (16G). Denna port aktiveras endast i systemmod då ju endast systemet har rätt att ändra data i MAC:en.

Även denna signal bildas i PAL 17E medan portens riktning styrs av CPU:ns R/$\overline{W}$-signal. Med hjälp av denna dataport kan också läget av CT-biten avläsas.

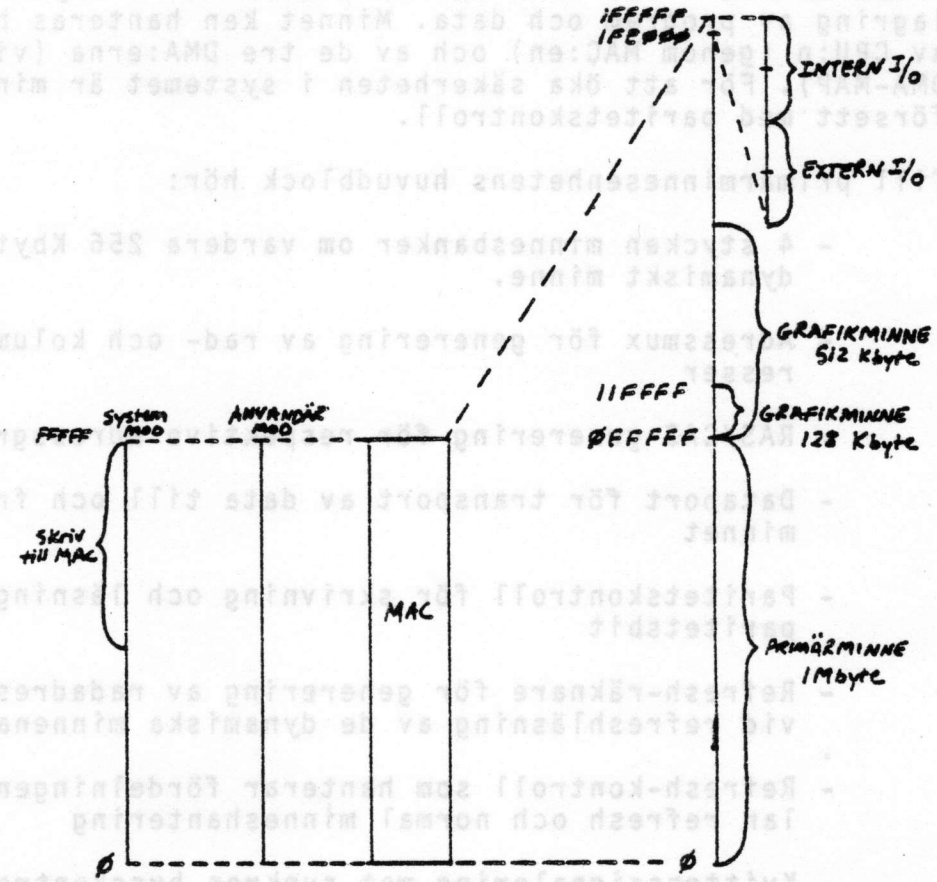
"Page-RAM"

Efter uppdelningen i 32 Kbytesblock sker ytterligare en uppdelning av minnet nämligen i page-RAM:et. Här delas blocken till 2 Kbytesblock med hjälp av CPU:ns adressledningar A11-A14, som tillsammans med de sex ombildade adressledningarna från segment-RAM:et får bilda adress till page-RAM:et.

Page-RAM:et består av de tre minnena 18G, 20G och 22G. Funktionen hos dessa minnen är likartad med segment-RAM:ets, men har på grund av databredden (12 bitar), två dataportar. För att hantera detta har en mux (21F) införts, som efter strobgenerering i 17E delar upp skrivstroben till minnena och enable till dataportarna. Adress A0 används som muxsignal och fördelar således hög- och lågdel i de ord som skrivs eller läses från page-RAM:et.

De data som kommer ut från page-RAM:et utgör de ombildade adresserna X11-X20 samt signalerna Wp och Nonx. Med hjälp av X11-X20 sker nu all adressering, då CPU:n har busskontroll både till minnen och I/O. För att adressera inom ett 2 Kbytesblock används CPU:ns egna adressledningar A0-A10 utan omvandling. Med hjälp av signalerna Wp och Nonx som skrivs in i page-RAM:et på samma sätt som adresstilldelningen, kan systemet skrivskydda (Wp) resp blockera ut Nonx vilket block som helst. Detta utgör ett bra skydd mot obehörig åtkomst av program och data och hindrar på så sätt olika processer att påverka eller förstöra varandras minnesareor.

Ett försök att exempelvis skriva i skrivskyddad area eller att utnyttja minne, som ej är tilldelat kommer i ABC 1600 att leda till att "BUS-error"-ingången på CPU:n aktiveras och tas där omhand av en speciell programvara, som hanterar misstaget. Vidare se "BUS-error" under "exception processing" i CPU-beskrivningen.


1FFFFF
1FE000
INTERN I/O
EXTERN I/O
GRAFIKMINNE
512 Kbyte
11FFFF
GRAFIKMINNE
128 Kbyte
ØEFFFF
FFFFF
System mod
ANVÄNDAR mod
Skriv till MAC
MAC
PRIMÄRMINNE
1Mbyte
Ø
Ø

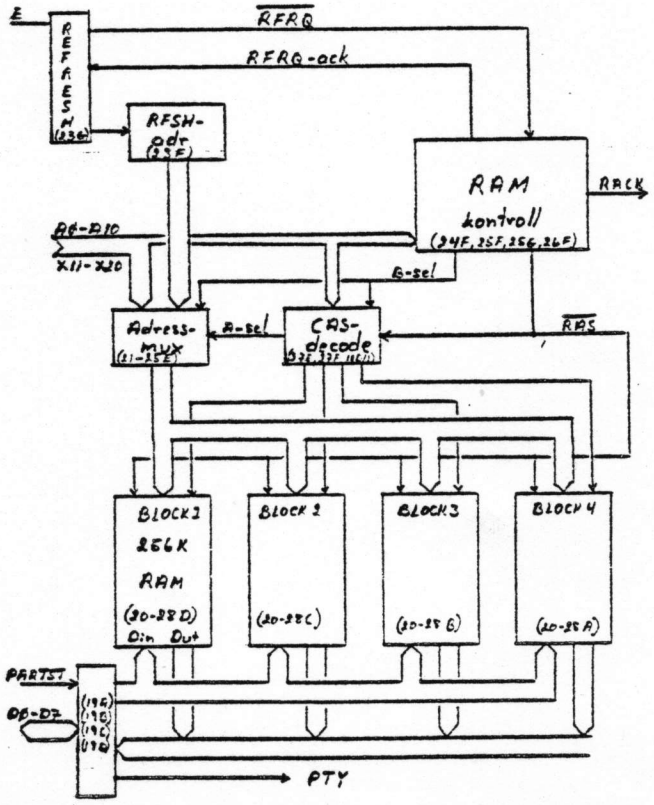
3.1.5 Primärminne

Primärminnet i ABC 1600 har en storlek av 1 Mbyte för lagring av program och data. Minnet kan hanteras både av CPU:n (genom MAC:en) och av de tre DMA:erna (via DMA-MAP). För att öka säkerheten i systemet är minnet försett med paritetskontroll.

Till primärminnesenhetens huvudblock hör:

- 4 stycken minnesbanker om vardera 256 Kbyte dynamiskt minne.
- Adressmux för generering av rad- och kolumnadresser
- RAS/CAS-generering för respektive adressgrupp
- Dataport för transport av data till och från minnet
- Paritetskontroll för skrivning och läsning av paritetsbit
- Refresh-räknare för generering av radadresser vid refreshläsning av de dynamiska minnena
- Refresh-kontroll som hanterar fördelningen mellan refresh och normal minneshantering
- Kvittenssignalering mot synkron busskontroll
- Adressområdeskontroll + Byglingsfält

PRIMÄRMINNE



Minnesblock

Varje minnesblock består av nio stycken 256 K x 1Bit dynamiska RAM varav åtta är till för databitarna D0-D7 och den resterande för paritet. Förutom databit in och ut finns nio adressledningar (MEMA 0-8), lässkrivsignal samt radadresstrob (RAS) och kolumnadresstrob (CAS) till varje minneskrets. Adresser och radadresstrob genereras alltid till alla minneskretsar vid läsning/skrivning i minnet, under det att kolumnstroben endast är aktiv för ett minnesblock åt gången. Detta gör att man vid refreshläsning av minnet kan aktivera alla blocken samtidigt då endast radadress och radadresstrob genereras till minnet i detta läge.

Minneskretsarna är lokaliserade enligt följande:

Minnesblock 1 (0-256 Kbyte)

D0	i pos 28 D
D1	i pos 27 D
D2	i pos 26 D
D3	i pos 25 D
D4	i pos 24 D
D5	i pos 23 D
D6	i pos 22 D
D7	i pos 21 D
Paritet	i pos 20 D

Minnesblock 2 (256-512K):

D0	i pos 28 C
D1	i pos 27 C
	.
	.
Paritet	i pos 20 C

Minnesblock 3 (512K-768K):

D0	i pos 28 B
D1	i pos 27 B
	.
	.
Paritet	i pos 20 B

Minnesblock 4 (768K-1 Mbyte):

D0	i pos 28 A
D1	i pos 27 A
.	
.	
Paritet	i pos 20 A

Adressmuxar

CPU:n (eller någon av de tre DMA-kretsarna) lägger via MAC:en (eller DMA-MAP) ut en adress till den minnesposition som ska aktiveras. Adresserna (A0-A10, X11-X17) påförs nu adressmuxarna (Pos 21E-25E) som är av typen "1 av 4" för att i dessa tidsmässigt fördelas av muxsignalerna ASEL och BSEL. Av de fyra ingångarna på varje mux används tre.

ASEL	BSEL	
Låg	Låg	Ej använd
Låg	Hög	Kolumnadress till minnena
Hög	Låg	Refreshadress till minnena
Hög	Hög	Radadress till minnena

Då minnet befinner sig i vila d v s varken CPU, DMA eller Refresh-räknarna adresserar minnet är ASEL och BSEL båda höga. Detta gör enligt tabellen att radadresser kan genereras till minneskretsarna. Vad som nu behövs för att minnet ska uppfatta detta som en adress är radadresstrob (RAS). Denna genereras av att stroben ADS kopplas via ingång 3 på mux 26F. Detta kräver att selektingångarna A och B är höga. Då B-ingången är kopplad till BSEL och A-ingångens signal bildas av BSEL i D-vippan 24F, skiljer endast en klockpuls mellan dessa signaler, vilket innebär att ingång 3 på muxen i detta läge är den aktiva.

Stroben ADS som är aktivt hög bildas i NAND-grindarna 6C/1,3 och 4, då signalerna $\overline{DS}$, $\overline{AS}$ är aktiva samt då signalen $\overline{NORAM}$ ej är aktiv. $\overline{AS}$ och $\overline{DS}$ aktiveras av CPU:n och $\overline{NORAM}$ som bildas i PAL 5D är aktiv endast vid läsning i BOOT-PROM, skrivning eller läsning i MAC samt vid "interrupt acknowledge". (studera gärna PAL - listan). Vid normal minneshantering blir således ADS aktiv då datastroben ($\overline{DS}$) och adresstroben (AS) är aktiva.

RAS - CAS

Adresserna (A0-A8) finns nu fram till minneskretsarnas adressingångar och då radadresstrob ($\overline{RAS}$) genereras, låses dessa adresser fast i minnena som radadress.

Radadresstroben ger efter fördröjningslinjen 27F upphov till att ASEL går låg. Detta innebär att ingång 2 på adressmuxarna kopplas som adress till minnena, dvs adresserna A9, A10, X11-X17. ASEL är också en av de signaler som behövs för att aktivera avkodaren 27 E som genererar kolumnadresstrobar. Den andra enablesignalen bildas i fördröjningslinjen 27F som efter att radadresstroben fördröjts 70 ns öppnas avkodaren via latchen 10D/1 och NOR-grinden 28F/2. Latchen och NOR-grinden ser till att ingen kolumnadresstrob genereras under refresh.

Som tidigare omtalats kommer kolumnadresstroben att avgöra vilket minnesblock som ska aktiveras. Avkodningen sker därför med hjälp av adresserna X18 och X19. Dessa adresser kommer nu att avgöra vilken kolumnadresstrob (CAS0-3) som blir aktiv.

När nu rätt adresser med sina strobar aktiverat minnet återstår att koppla in dataport samt kvittera till CPU:ns $\overline{DTACK}$-ingång.

Kvittenssignal

Kvittenssignalen från minnet heter RACK och genereras i D-vippan 25F/2. För att RACK ska bli aktiv måste adressområdet vara det rätta samt att signalen BSEL ska vara hög (RACK ska ej genereras vid refresh). Detta kontrolleras i NOR-grinden 26G/2, vars utgång bildar dataingång på 25F/2, men används också till att aktivera mux 26F för att släppa fram skrivstrob till minnena (se nedan). Är resetingången på 25F/2 inte aktiverad kommer kvittenssignal nu att genereras till CPU:n.

Observera att resetingången är aktiverad under refreshläsning samt då radadresstroben är aktiv.

Skrivstrob

Vad som ovan sagts gäller både vid läsning och skrivning i minnet, men för att kunna skriva måste också skrivstroben ($\overline{MEMWR}$) ges. Stroben kommer från mux 26F som kopplar signalen MEMW till minnet. MEMW bildas i NOR-grinden 26G/1 genom kombination av signalerna R/$\overline{W}$ och $\overline{DS}$ från CPU, WP och Nonx från MAC samt den tidigare omnämnda $\overline{NORAM}$-signalen. Uppfylls villkoren kan data skrivas till minnet.

Paritetskontroll/dataport

För att hantera data till och från minnena finns dataportarna 19B och 19C. Normalt ligger porten riktad så att data kan skrivas till minnena, men kan riktas utåt genom att signalen MRS blir aktiv. Detta kräver att R/$\overline{W}$-signalen står i läge "Read", $\overline{DS}$ är aktiv samt att minnets kvittensignal $\overline{RACK}$ också är aktiv.

Då data skrivs till minnena kopplas databussens data efter invertering i dataporten till minnenas dataingång och vid läsning kopplas minnenas datautgång till databussen via den inverterade dataporten.

För att uppnå en bättre kontroll på de data som läses ur minnena genereras en paritetsbit för varje utläst byte data. Varje minnesblock består av nio minneskretsar, åtta för databitarna (D0-D7) och en för paritet. Paritetsbiten beräknas vid skrivning i 19D som är en speciell krets för beräkning av paritet.

Kretsen genererar en utsignal som är beroende av de nio insignalernas data. Åtta av de nio ingångssignalerna utgörs av indata under det att den 9:e, signalen PARTST kommer från specialkontrollregistret, (se vidare I/O--hantering), och används vid testning av minnet. Som utsignal från paritetskretsen får man signalen PTYWD som utgör data in till minneskrets nummer 9.

Vid läsning ur minnet kommer de nio minneskretsarnas utdata att bilda insignal till paritetskontrollkretsen 19A. Är alla databitar riktiga blir utsignalen $\overline{PTY}$ läge men om något fel inträffat, aktiveras $\overline{PTY}$-signalen och NMI (interuptnivå 7) genereras till CPU:n.

Detta sker genom att $\overline{PTY}$-signalen via NOR-grinden 8C/1 ger J-ingången på J/K-vippan 18B/2 en hög nivå. (En förutsättning är dock att paritetsteststroben PARTST ej är aktiv). J/K-vippan klockas av MRS-signalen som aktiveras vid läsning i minnet och dess utsignaler slår om. Den inverterade utgången $\overline{SETNMI}$, påverkar set-ingången på D-vippan 4D/2 vars utgång via PAL 5D ger NMI till CPU:n.

Den icke inverterade utgången på 18B/2 klockar CAUSE-registret. CAUSE-registret laddas nu med adressinformation från adresserna X16-X20 samt signalen $\overline{DMAOK}$. På detta sätt kan man skaffa sig information om i vilket adressområde paritetsfelet inträffade samt om någon DMA haft busskontrollen. Då CAUSE-registret klockas tänds också en lysdiod på kretskortet för att på ett mer påtagligt sätt indikera att ett paritetsfel inträffat. Med hjälp av den tidigare nämnda CAUSE-signalen kan nu CPU:n läsa CAUSE-registret och på så sätt få reda på orsaken till ett paritetsfel.

Obs!
De första 200 producerade ABC 1600-datorerna gav reset ej NMI vid paritetsfel.

Refresh

Alla dynamiska minnen av den typ som bildar primärminnet i ABC 1600 kräver att man med jämna mellanrum läser i alla celler. Minneselementen består i princip av en kondensator som tappar sin information om den ej aktiveras. Vid vissa tillämpningar som exempelvis grafikminnet, aktiverar man kontinuerligt alla celler vid utläsning av data till bildskärmen och behöver där ingen speciell refresh-läsning. I ett primärminne däremot kan man inte garantera denna kontinuitet och måste där ha speciell elektronik för att åstadkomma refresh. Det måste också finnas kontrollogik som ser till att en refreshläsning inte sker då CPU:n skriver eller läser och vise versa.

Vid refresh av minnet placeras en adress från refreshräknarna via adressmuxen till minnenas adressingång. Därefter aktiveras radadresstroben $\overline{\text{RAS}}$, varvid en läsning sker, inte som vid en normal CPU-läsning av en byte, utan av en hel "rad" i minnet.

256-Kbits-kretsarna som sitter i ABC 1600 har en inre organisation som motsvarar 256 rader varför vi aktiverar 1/256 av minnet vid varje refreshläsning. Det behövs således räknare som kan generera en 8-bitars adress (0-255) och dessa räknare återfinner vi i pos 23F/1-2.

Klockan till dessa räknare heter $\overline{\text{RFRCLK}}$ och har sitt ursprung i E-klockan, som via vippan 24G/1 och räknaren 23G delas till lämplig frekvens. Som tidigare omtalats under avsnittet systemklockor, utgör E-klockan 1/10 av CPU-klockan, i detta fall 800 KHz vid 8 MHz CPU-klocka. Efte-r halvering i 24G/1 sker delning med 6 i räknaren 23G, vilket ger oss ca 67 KHz för $\overline{\text{RFRCLK}}$.

Nu är refreshadressen kopplad till minnet och det återstår att ordna rätt timing samt att hantera fördelningen mellan CPU-läsning/skrivning och refresh. Detta handhas av kretsarna 24F, 24G/2, 25F/1 och 25G/2. Då det är tid för refresh, klockas D-vippan 24G/2, så att den inverterade utgången går låg. Denna förändring ger efter två klockningar ($\overline{\text{CLK}}$) av 24F att utgång 2 går låg och 3 hög. Utgång 2 bildar data in till vippan 25F/1, som vid nästa klockpuls ($\overline{\text{CLK}}$) ger den inverterade utgången hög nivå.

Detta gäller under förutsättning att vippans SET-ingång ej är aktiverad, dvs adresstroben $\overline{\text{AS}}$ är aktiv. Att $\overline{\text{AS}}$ är aktiv, tyder på att CPU:n eller någon DMA är mitt i en läsning/skrivning och får inte brytas av refresh. Därför hålls utgången låg tills adresstroben blir inaktiv och efterföljande klockpuls kan sätta den till hög nivå.

Tillsammans med utgång 3 från 24F, som redan tidigare lagts till hög nivå, ger detta omslag av AND-NOR-grinden 25/2, så att dess utgång BSEL går låg. BSEL som påverkar B-ingången till muxen 26F och efter klockning genom D-vippan 24F även A-ingången, gör att ingång 0 på muxen kopplas till utgången och radadresstrob avges. Då radadresstroben aktiveras är BSEL låg och ASEL fortfarande hög, vilket gör att adressmuxarna har ingång 1 dvs refreshräknarens adresser kopplade till minnet.

Då A-ingången på muxen 26 F aktiveras, genereras också reset till 24G/2, vars utgång genom två klockningar i 24F avslutar refreshläsningen då BSEL åter går hög.

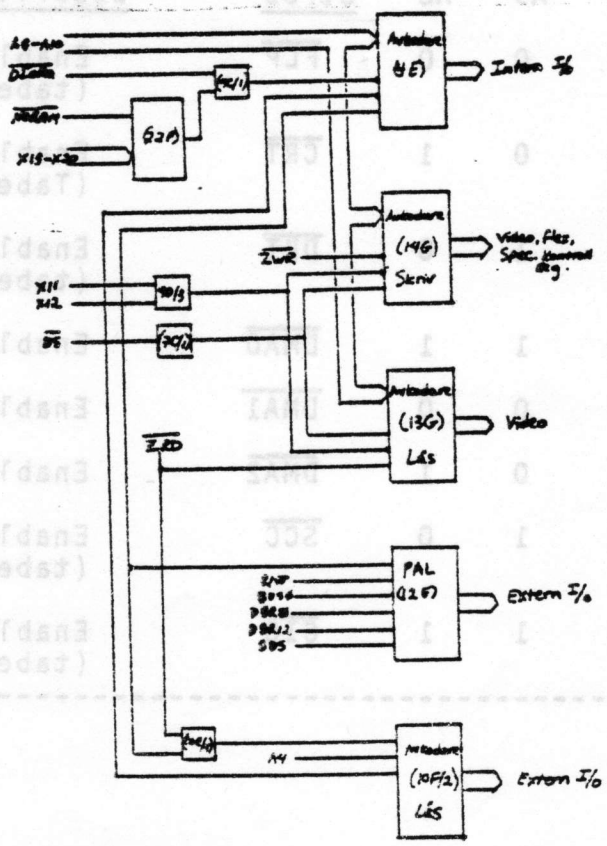
Efter ytterligare en klockning släpps resetingången på 24G/2 och muxen återgår till sitt ursprungliga läge och vid följande klockning ges ånyo möjlighet att generera kvittenssignal till CPU:n. Minnena är nu åter redo att ta emot eller leverera data till CPU:n.

3.1.6 I/O-adressering

I/O-adresseringen används till att skriva och läsa i de olika register som finns i ABC 1600. Det kan exempelvis vara allt från att generera enablestrobar till periferikretsar, till att skriva in startvärden i videoenhetens "mover"-register. I CPU:n 68008 skiljer man inte på minnes och I/O-hantering (jfr Z80), utan den ser all hantering som minnesadresseringar. Detta gör att systemkonfigurationen avgör hur stor del av tillgängligt adresseringsområde, som avsätts för I/O.

I ABC 1600 har ett utrymme på 8 Kbyte högst upp i det adresserbara utrymmet avsatts. Detta styrs av en adressavkodare (NAND-grinden 22F) där adresserna X13X20 samt signalen NORAM känns av. Då alla dessa har hög nivå genereras utsignalen I/O, som direkt eller via grindning med andra signaler styr I/O-portarna. Både CPU:n (via MAC:en) eller någon av de tre DMA:erna (via DMA--MAP) kan aktivera signalen.

Det på detta sätt reserverade, 8 Kbyte stora adressblocket, delas först upp på två huvudblock om vardera 4 Kbyte med hjälp av adress X12. Då adressledning X12 har låg nivå adresseras extern I/O dvs någon av de fyra busskontakterna med strobar som RCSB, EXP, E0, E12, CSA etc. Ges X12 hög nivå adresseras intern I/O som exempelvis videodel och periferikretsar. Här sker också en uppdelning med hjälp av X11, så att X11 låg ger adressering av videodel + specialkontrollregister + flexskiveregister och då X11 är hög, de inre periferikretsarna och deras interna register.



Adress	X12	X11	
1FE000H-1FE7FFH	0	0	BUS-interface
1FE000H-1FEFFFH	0	1	
1FF000H-1FF7FFH	1	0	Periferikretsar
1FF800H-1FFFFFH	1	1	Videoenhet+spec. kontr.reg.+flexreg.+ DMA-MAP

Vid I/O-adressering kan man generellt säga att X11-X12 adresserar I/O-gruppen under det att A8-A10 adresserar enheten inom gruppen. De lägsta adresserna A0-A4 används sedan för att adressera enheternas enskilda register samtidigt, som databussen levererar data till aktuellt register.

Intern I/O

Nedan följer en sammanställning i tabellform som beskriver adresserna till de olika registren.

TABELL 1: Intern I/O, periferikretsar

$\overline{I/O}$ + R/$\overline{W}$ X12 = "1" X11 = 0 Adressintervall IFF000H-IFF7FFH

Adress	A10	A9	A8	Strob	Beskrivning
1FF0XX	0	0	0	$\overline{FLP}$	Enable 1797 (tabell 4)
1FF1XX	0	0	1	$\overline{CRT}$	Enable CRT (Tabell 5)
1FF2XX	0	1	0	$\overline{DRT}$	Enable DART (tabell 6)
1FF3XX	0	1	1	$\overline{DMA0}$	Enable DMA0
1FF4XX	1	0	0	$\overline{DMA1}$	Enable DMA1
1FF5XX	1	0	1	$\overline{DMA2}$	Enable DMA2
1FF6XX	1	1	0	$\overline{SCC}$	Enable SCC (tabell 7)
1FF7XX	1	1	1	$\overline{CIO}$	Enable CIO (tabell 8)

TABELL 2: Intern I/O, video, flexskiva, DMAMAP, spec. kontr.reg.

$\overline{I/O}$ + R/$\overline{W}$($\overline{W}$)x12=1, X11=1 Adressintervall 1FF800H-1FFFFFH

Adress	A10	A9	A8	Strob	Beskrivning
1FF8XX	0	0	0	IOWR0	Skrivstrob 0 Video(tabell 9)
1FF9XX	0	0	1	IOWR1	Skrivstrob 1 Video(tabell 10)
1FFAXX	0	1	0	IOWR2	Skrivstrob 2 Video(tabell 11)
1FFBXX	0	1	1	FW	Kontrollreg. flexskiveint. (tabell 13)
1FFCXX	1	0	0	Ej Använd	
1FFDXX	1	0	1	DMAMAP	Skriv/läs DMA-MAP (tabell 16)
1FFEXX	1	1	0	EN.SP. CONT.REG.	Skriv spec.-kontrollreg. (tabell 17)
1FFFXX	1	1	1	Ej använd	

TABELL 3: Intern I/O, läsning, video

$\overline{I/O}$ R/$\overline{W}$ X12=1, X11=1 Adressintervall 1FF800H-1FFFFFH

Adress	A10	A9	A8	Strob	Beskrivning
1FF8XX	0	0	0	IORD0	Läs statusregister (tabell 18)
1FF9XX	x	x	x	Ej använd	
1FFFXX					

TABELL 4: Intern I/O, flexskiva

$\overline{FLP}$ Adressintervall 1FF000H-1FF0FFH

Adress	A2	A1	Register
1FF000	0	0	Kommando/Status
1FF002	0	1	Spår
1FF004	1	0	Sektor
1FF006	1	1	Data

TABELL 5: Intern I/O, CRTC

$\overline{CRT}$ Adressintervall 1FF100H-1FF1FF

Adress	A0	Register
1FF100	0	Välj Register
1FF101	1	Skriv/läs register

TABELL 6: Intern I/O, DART

$\overline{DRT}$ Adressintervall 1FF200-1FF2FF

Adress	A2	A1	Register
1FF200	0	0	Kontrollregister port B
1FF202	0	1	Dataregister port B
1FF204	1	0	Kontrollregister port A
1FF206	1	1	Dataregister Port A

TABELL 7: Intern I/O, SCC

$\overline{\text{SCC}}$ Adressintervall 1FF600-1FF6FF

Adress	A2	A1	Register
1FF600	0	0	Kontrollregister port B
1FF602	0	1	Dataregister port B
1FF604	1	0	Kontrollregister port A
1FF606	1	1	Dataregister port A

TABELL 8: Intern I/O, CIO

$\overline{\text{CIO}}$ Adressintervall 1FF700-1FF7FF

Adress	A2	A1	Register
1FF700	0	0	Dataport C
1FF702	0	1	Dataport B
1FF704	1	0	Dataport A
1FF706	1	1	Kontrollport

TABELL 9: Intern I/O, Skriv video

$\overline{\text{IOWR0}}$ Adressinteval1 1FF800-1FF8FF

Adress	CA2	CA1	CA0	Signal
1FF800	0	0	0	LDSX HB
1FF801	0	0	1	LDSX LB
1FF802	0	1	0	LDSY HB
1FF803	0	1	1	LDSY LB
1FF804	1	0	0	LDTX HB
1FF805	1	0	1	LDTX LB
1FF806	1	1	0	LDTY HB
1FF807	1	1	1	LDTY LB

TABELL 10: Intern I/O, Skriv video

$\overline{\text{IOWR1}}$ Adressintervall 1FF900-1FF9FF

Adress	CA2	CA1	CA0	Signal
1FF900	0	0	0	$\overline{\text{LDFX HB}}$
1FF901	0	0	1	$\overline{\text{LDFX LB}}$
1FF902	0	1	0	$\overline{\text{LDFY HB}}$
1FF903	0	1	1	$\overline{\text{LDFY LB}}$
1FF904	1	0	0	Ej använd
1FF905	1	0	1	$\overline{\text{WRML}}$
1FF906	1	1	0	Ej använd
1FF907	1	1	1	$\overline{\text{WRDL}}$

TABELL 11: Intern I/O, Skriv video

$\overline{\text{IOWR2}}$ Adressintervall 1FFA00-1FFAFF

Adress	CA2	CA1	CA0	Signal
1FFA00	0	0	0	WRMASK STROBE HB
	0	0	1	" " LB
LB				
	0	1	0	ENABLE CLOCKS
	0	1	1	FLAG STROBE
tabell 12				
	1	0	0	ENDISP
	1	0	1	Ej använd
	1	1	0	Ej använd
	1	1	1	Ej använd

TABELL 12: Intern I/O, flaggregister video

FLAG STROBE	Adress 1FFA03
Databit	Signal
D0	L/P FLAG
D1	BLANK FLAG
D2	PIX POL
D3	FRAME POL
D4	HOLD FY
D5	HOLD FX
D6	COMP MOVE FLAG
D7	REPLACE/SET & RESET

TABELL 13: Intern I/O, flexskiveregister

$\overline{FW}$ Adressområde 1FFB00-1FFBFF

Adress	A7	A0	Signal	
1FFB00	0	0	FW0	(Tabell 14)
1FFB01	0	1	FW1	(Tabell 15)

TABELL 14: Intern I/O, kontrollregister, flexskiveinterface

FW0	Adress 1FFB00
Databit	Signal
D0	Master reset till 1797
D1	Densitet Dubbel/Enkel
D2	Head Load Timing 1797
D3	MINI Input (9229) 8"/5.25"
D4	Head Load Input (9229)
D5	Precompensation select P0
D6	" " P1
D7	" " P2

TABELL 15: Intern I/O, kontrollreg. flexskiveenhet

FW1 Adress 1FFB01

Databit	Signal
D0	Driveselect 1
D1	" 2
D2	" 3
D3	Motor on
D4	Post Compensation
D5	Low Current
D6	Ej använd
D7	" "

TABELL 16: Intern I/O, DMAMAP

DMAMAP Adressområde 1FFD00-1FFDFF

Adress	A2	A1	A0	Register
1FFD00	0	0	0	DMA-MAP register 2 (låg byte)
1FFD01	0	0	1	" " (hög byte)
1FFD02	0	1	0	Ej använd
1FFD03	0	1	1	" "
1FFD04	1	0	0	DMA-MAP register 1 (låg byte)
1FFD05	1	0	1	DMA-MAP register 1 (hög byte)
1FFD06	1	1	0	DMA-MAP register 0 (låg byte)
1FFD07	1	1	1	DMA-MAP register 0 (hög byte)

TABELL 17 :Intern I/O, specialkontrollregister

EN SPEC. CONTR. REG.	Adress 1FFE00
Databit	Signal/Funktion
D0	CS7
D1	Ej använd
D2	$\overline{\text{BTCE}}$
D3	$\overline{\text{ATCE}}$
D4	PARTST
D5	$\overline{\text{DMADIS}}$
D6	SYSSCC
D7	SYSFS

Ur tabellerna kan vi se vilka adresser som aktiverar de olika I/O-enheterna, men det är också viktigt för förståelsen att titta på skrivlässignaler och andra styrande signaler.

Om man börjar med enablesignaler till interna periferikretsar (tabell 1), så aktiveras de som tidigare setts av X12 = hög och X11 = låg, men också av $\overline{\text{I/O}}$-stroben och $\overline{\text{DIORQ}}$ genom OR-grinden 4C/1. I/O-stroben genereras enligt ovan då adresserna X13 - X20 har hög nivå under det att $\overline{\text{DIORQ}}$-signalen är beroende av om CPU:n eller någon DMA har busskontrollen. Då CPU:n har kontrollen genereras $\overline{\text{DIORQ}}$ i PAL6E och aktiveras då CPU:n vill skriva eller läsa i DART:ens eller någon av DMA:ernas interna register. $\overline{\text{DIORQ}}$ har i detta läge ingen betydelse för aktivering av strobarna.

Har däremot någon av DMA:erna kontrollen genereras $\overline{\text{DIORQ}}$ av denna DMA och fungerar då på samma sätt som CPU:ns $\overline{\text{I/O}}$-signal. Detta betyder att både CPU och DMA kan aktivera avkodaren 4E oberoende av skriv eller lässekvens.

Nästa avkodargrupp (Tabell 2 och 3) omfattar I/O-hantering av video, specialkontrollregister och registren i flexskiveinterfacet utom de interna registren i kontrollkretsen 1797. Huvudavkodarna i gruppen 13G (för lässtrob) och 14G (för skrivstrobar) aktiveras genom en kombination av signalerna X11, X12, $\overline{\text{DS}}$ samt vid generering av lässtrob $\overline{\text{ZRD}}$ och för skrivstrob $\overline{\text{ZWR}}$. De två adresserna X11 och X12 grindas i NAND-grinden 9D/3 och datastroben $\overline{\text{DS}}$ från CPU eller DMA via inverteraren 7C/1 aktiverar var sin enableingång på de båda avkodarna.

ZRD och ZWR bildas i PAL 11E och används bl a till DART och BUS-interface men här utgör de enableingångar ZRD till 13G och ZWR till 14G. Se vidare i PAL-listan.

Med hjälp av adresserna A8-A10 väljer man nu vilken strob som ska aktiveras. De tre första skrivstrobarna tillsammans med den enda lässtroben aktiverar register i videodelen. Då IORD0 aktiveras kan videodelens statusregister läsas under det att skrivstrobarna IOWR0-2 styr laddning av moverregister, flaggregister, aktivering av klockor och utläsning till bildskärm mm. Mer om videodelens I/O kan du läsa om i beskrivningen över videodelen.

Utgång nr 3 från avkodaren ger enable till en ny avkodare 10F/1. Med hjälp av adresserna A7 och A0 styrs de två strobarna FW0 och FW1 som aktiverar var sitt register i flexskiveinterfacet. När de två registren 7B och 8B klockas av FW0 och FW1 läses data in från databussen som sedan får styra flexskiveinterfacets olika funktioner mot flexskiveenheter, kontrollkrets och dataseparator. Se vidare beskrivning över flexskiveinterfacet.

Från huvudavkodarens utgång 5 kan CPU:n styra signalen DMAMAP som via mux 1E aktiverar RAM-kretsarna i DMA-MAP så att CPU:n kan skriva in nya adresser för DMA-överföring.

Utgång nr 6 slutligen aktiverar specialkontrollregistret 13E. Med hjälp av databitarna D0-D2 väljer man vilken utgång som ska kopplas till den gemensamma ingången som utgörs av databit D3. Specialkontrollregistret ger sju utgångssignaler. Dessa utgörs av CS7 som ger en extra kortadressledning till BUS-interfacet, ATCE och BTCE som styr klockomkoppling på de två V24-modulerna, PARTST som aktiveras vid paritetstest av primärminnet, DMADIS, som blockerar möjligheten för DMA:erna att begära busskontrollen, SYSSCC och SYSFS för att styra om SCC:n respektive flexskivan ska ha företräde till sina respektive DMA-kretsar eller om BUS-interfacen ska ges möjlighet att nyttja DMA:erna.

Extern I/O

Nu har I/O-områdets övre halva behandlats (dvs då adress X12 är hög). I den nedre halvan (X12="0") sker all I/O-hantering av de fyra BUS-interfacen. Strobavkodningen hanteras företrädesvis av avkodaren 10F/2 och PAL 12E, men en hel del avkodning sker också ute på själva busskortet. Busskortet behandlas i ett senare avsnitt.

Till avkodare 10F/2 genereras enablesignal via OR-grinden 10E/4, till vilken signalerna X12 och ZRD kommer.

När båda dessa signaler har låg nivå kommer avkodaren att aktiveras. Adresserna A4 och X11 får nu styra vilken av utgångarna som ska bli aktiv. Då endast de två övre utgångarna används inser man att X11 måste vara hög för att någon av dessa ska aktiveras, under det att adress A4 styr vilken av strobarna RCSB och EXP som genereras. RCSB används för att läsa det register på busskortet som talar om vilken position i expansionslådan eller i vilken av de fyra platserna i ABC 1600 som ett visst kort befinner sig i. EXP används för ett liknande syfte men bara när antalet expansionsplatser i en yttre expansionsenhet överstiger fem kortplatser.

I PAL 12E genereras ytterligare fem strobar till busskortet. Dessa är E0, E12, CSA, CSP och DIR. Ur PAL - listan kan du se när respektive signal är aktiv.

E0 håller dataport och adressport för strob-generering öppna för kommunikation med BUSOI och BUSOX. Den fungerar som enablesignal till dessa busskontakter. Detsamma gäller E12 men nu för BUS1 och BUS2. CSA lägger ut en kortadress på databussen för att adressera ett speciellt expansionskort.

CSP ger CS-strob (Card Select) till alla expansionskort. Detta utföres automatiskt av PAL 12E varje gång ett expansionskort adresseras då kortadressen finns som en del av I/O-adressen (A5-A10).

DIR styr dataportarnas riktning, skrivning eller läsning mot expansionskorten samt medverkar vid strob-generering på busskortet.

3.1.7 Interrupthantering

I ett datorsystem är det programvaran som via CPU:n styr kommunikationen och dataflödet mellan systemets olika delar. För att inte belasta CPU:n för hårt låter man ofta periferienheterna själva övervaka sin uppgift utan inblanding av CPU:n. Det kan vara en kommunikationskrets som får kontrollera om några data är på väg in till systemet eller att övervaka om kraftmatningen till datorn plötsligt försvinner. I båda fallen måste den yttre elektroniken kunna meddela sig med CPU:n för att påtala att något inträffat som kräver CPU:ns engagemang. Dessa meddelanden orsakar ett avbrott i CPU:ns normala programkörning och kallas följaktligen "interrupt".

Då periferienheterna hela tiden arbetar oberoende av varandra kan det mycket väl hända att flera enheter samtidigt vill att CPU:n ska ta hand om just deras problem. I detta läge måste det finnas en viss förutbestämd prioritetsordning. Det säger sig självt att ett paritetsfel exv är betydligt allvarligare än då ett tecken kommer från tangentbordet. För att hantera detta kan CPU:n känna av tre olika nivåer av interrupt. Med hjälp av dessa delas elektroniken in i tre prioritetsgrupper som inom sig också kan ha en viss prioritetsordning.

I ett så komplext system som ABC 1600 där så många olika yttre enheter kan generera interrupt, räcker inte de tre prioritesnivåerna till för att CPU:n ska kunna veta vilken enhet som ska betjänas. För att få fram bättre uppgifter och för att kvittera att begäran accepterats sänder CPU:ns kvittenssignaler som bekräftar begäran samt vilken prioritetsnivå kvittensen gäller. Som svar på detta skickar den begärande enheten en s k "vektor" tillbaka till CPU:n. Vektorn levereras som en byte på databussen och används av CPU:n som pekare i en tabell där alla interruptvektorer ligger. I den tabellposition som vektorn pekat ut finns startadressen till den rutin (program) som hanterar just den enheten som begärt interrupt.

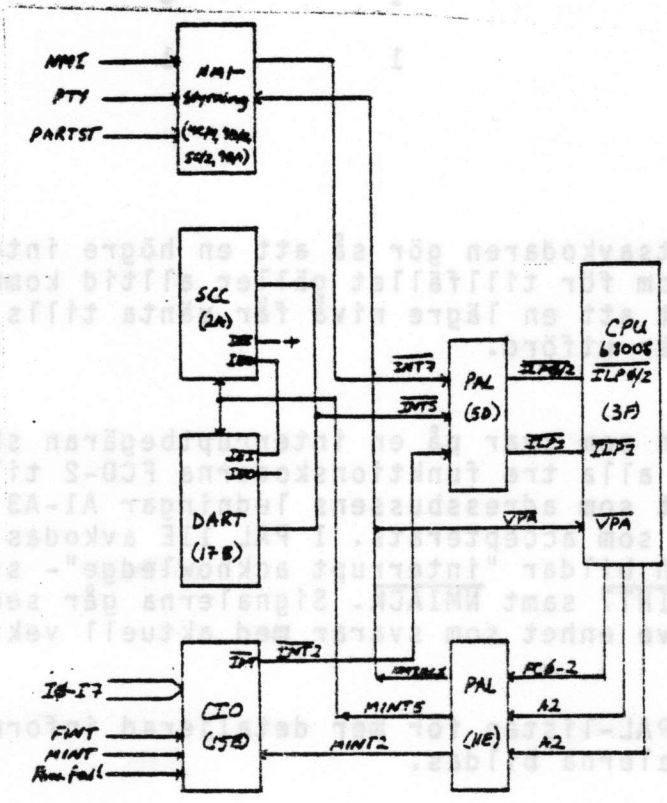
Enklare logik saknar förmåga att själva leverera en vektor, men genom att signalera till en speciell ingång på CPU:n kan detta ändå hanteras då CPU:n i detta fall själv kan hämta en startadress på en på förhand bestämd position i tabellen.

Vektortabell ABC 1600

Vector Number(s)	Address Dec	Address Hex	Space	Assignment
0	0	000	SP	Reset: Initial SSP[2]
	4	004	SP	Reset: Initial PC[2]
2	8	008	SD	Bus Error
3	12	00C	SD	Address Error
4	16	010	SD	Illegal Instruction
5	20	014	SD	Zero Divide
6	24	018	SD	CHK Instruction
7	28	01C	SD	TRAPV Instruction
8	32	020	SD	Privilege Violation
9	36	024	SD	Trace
10	40	028	SD	Line 1010 Emulator
11	44	02C	SD	Line 1111 Emulator
12[1]	48	030	SD	(Unassigned, Reserved)
13[1]	52	034	SD	(Unassigned, Reserved)
14[1]	56	038	SD	(Unassigned, Reserved)
15	60	03C	SD	Uninitialized Interrupt Vector
16-23[1]	64	040	SD	(Unassigned, Reserved)
	95	05F		—
24	96	060	SD	Spurious Interrupt[3]
25	100	064	SD	Level 1 Interrupt Autovector
26	104	068	SD	Level 2 Interrupt Autovector
27	108	06C	SD	Level 3 Interrupt Autovector
28	112	070	SD	Level 4 Interrupt Autovector
29	116	074	SD	Level 5 Interrupt Autovector
30	120	078	SD	Level 6 Interrupt Autovector
31	124	07C	SD	Level 7 Interrupt Autovector
32-47	128	080	SD	TRAP Instruction Vectors[4]
	191	0BF		
48-63[1]	192	0C0	SD	(Unassigned, Reserved)
	255	0FF		—
64-255	256	100	SD	User Interrupt Vectors
	1023	3FF		—

NOTES:

1. Vector numbers 12, 13, 14, 16 through 23, and 48 through 63 are reserved for future enhancements by Motorola. No user peripheral devices should be assigned these numbers.
2. Reset vector (0) requires four words, unlike the other vectors which only require two words, and is located in the supervisor program space.
3. The spurious interrupt vector is taken when there is a bus error indication during interrupt processing. Refer to Paragraph 5.5.2.
4. TRAP #n uses vector number 32+n.



Till de enheter som kan generera ett interrupt hör CIO (Counter I/O), som samlar interruptsignaler från expansionsbussarna, flexskiveinterface, "mover" och "power fail"-signal. Dessutom kan CIO:ns egna Counter/Timer generera interrupt med vissa tidsintervall. CIO:n har interruptnivå nr 2.

Kommunikationskretsarna SCC och DART har gemensamt interruptnivå 5 men har också en intern prioritetsordning där SCC:n har högst prioritet. Av CPU:ns interruptnivåer är nr 7 eller NMI (Non Maskable Interrupt) den högsta. Den kan genereras av paritetsfel och från den expanderbara busskontakten BUSOX, men används här bara vid felsökning i systemnära programvara typ operativsystem.

Interruptnivåer

De tre interruptsignalerna $\overline{\text{INT2}}$, $\overline{\text{INT5}}$ och $\overline{\text{INT7}}$ samlas i PAL5D som bl a tjänstgör som prioritetsavkodare för interrupt. Utgångarna som på detta vis styrs heter $\overline{\text{ILP0/2}}$ och $\overline{\text{ILP1}}$ och går direkt till CPU:ns interruptingångar. Signalerna styrs enligt följande:

	$\overline{\text{ILP0/2}}$	$\overline{\text{ILP1}}$	Enhet
Inget interrupt	0	0	
$\overline{\text{INT2}}$	0	1	CIO
$\overline{\text{INT5}}$	1	0	SCC,DART
$\overline{\text{INT7}}$	1	1	Paritet, NMI från BUSOX

Prioritetsavkodaren gör så att en högre interruptnivå än den som för tillfället gäller alltid kommer genom, under det att en lägre nivå får vänta tills interruptrutinen är utförd.

När CPU:n som svar på en interruptbegäran skall kvittera läggs alla tre funktionskoderna FC0-2 till hög nivå samtidigt som adressbussens ledningar A1-A3 visar vilken nivå som accepterats. I PAL 11E avkodas dessa signaler och bildar "interrupt acknowledge"- signalerna, $\overline{\text{MINT2}}$, $\overline{\text{MINT7}}$ samt $\overline{\text{NMIACK}}$. Signalerna går sedan till respektive enhet som svarar med aktuell vektor.

Studera PAL-listan för mer detaljerad information om hur signalerna bildas.

NMI

NMI-signalen (INT7) genereras av D-vippan 4D/2 genom att signalen SETNMI aktiveras eller att vippan klockas av NMI-signalen från BUSOX som klockas vid paritetsfel,(se avsnitt om paritet).
Då man inte vet om NMI-signalen är ansluten via expansionsbussen, säkerställs dess nivå av PON-signalen. Den ser vid systemstart till att NMI-signalen är inaktiv genom att ge ett definierat läge åt NAND-grinden 5C/2. Detta läge råder sedan tills NMI-ingången aktiveras.

Med hjälp av HRST-signalen (vid reset eller uppstart) och med NMIACK som båda grindas via AND-grinden 4C/4 kan vippan 4D/2 återställas så att NMI-signalen blir inaktiv. NMIACK är också ansluten till CPU:ns VPA-in - gång och gör där så att CPU:n hämtar en intern interruptvektor vid NMI.

Interrupt från seriell kommunikation

En interruptbegäran från kommunikationskretsarna SCC och DART är något mer kompliserat då de själva kan generera en interruptvektor som ska läsas av CPU:n. Båda periferikretsarna har egna vektorregister som dessutóm kan modifieras vid sändning så att flera olika vektorer kan avges beroende på vilken intern del som vill begära interrupt. Dessa är dessutom ordnade i prioritet enligt följande:

Intern prioritet vid interrupt SCC och DART

Interruptkälla	Prioritet
Mottagning kanal A	Hög
Sänding " "	
Status " "	
Mottagning kanal B	
Sänding " "	
Status " "	Låg

Beroende på vilken källa som nu påkallar uppmärksamhet kommer vektorbytens databitar D1-D3 att påverkas så att olika vektorer genereras.

Periferikretsar enligt ZILOG-modell är utrustade med ett par signaler så att de kan bilda en kedja med avseende på interrupt, varmed man kan bestämma deras inbördes prioritet. Dessa signaler heter IEI (Interrupt Enable In) och IEO (Interrupt Enable Out) och kopplas så att IEO på en krets med högre prioritet ansluts till IEI till nästa i prioitetsordning etc.

Då i vårt fall SCC:n begär interrupt genom att aktivera INT påverkas också IEO så att DART:en förhindras att själv begära interrupt tills SCC:n är färdig.

När nu SCC:n aktiverat sin INT-utgång, påförs signalen, som heter INT5, prioritetsavkodaren och vidare till CPU:n. CPU:n svarar nu med att lägga alla tre funktionskoderna (FC0-2) höga, vilket innebär interrupt acknowledge samt ge adresserna A1-A3 värdet fem. I PAL 11E avkodas signalerna så att MINT5 aktiveras. Denna signal går till SCC:ns ingång INTACK och talar om att nu är det dags att lägga ut interruptvektorn på bussen. CPU:n läser in vektorn och aktiverar rätt programrutin.

DART:en arbetar på ett liknande sätt men har ingen speciell "interrupt acknowledge"-ingång utan använder en kombination av M1 och IORQ. M1 påverkas direkt av MINT5 under det att IORQ bildas i PAL 11E och aktiveras bl a under en sådan här sekvens.

Interrupt från CIO och expansionsbuss

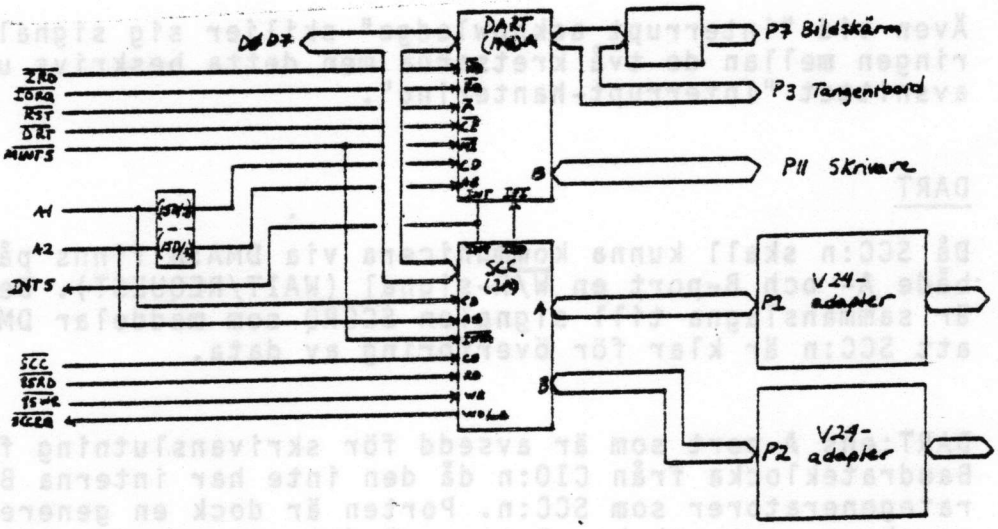
CIO:n, som arbetar mot interruptnivå 2, har ett liknande sätt att hantera interrupt som SCC:n, men har tre vektorregister, ett vardera för port A och B, samt ett för "Counter/Timers". För port A och B gäller att interruptvektorn kan modifieras beroende på vilken insignal (PA0-7 eller PB0-7) som aktiverats. Detta sker genom att vektorns tre databitar (D1-D3) visar nummret på insignalen. I den tredje vektorn visar D1-D2 vilken "Counter/Timer", som begär interrupt. CIO:n kan således ge en mängd olika vektorer och fungerar här som en samlare av externa interruptsignaler. Här kommer interruptsignaler från busskontakterna (I0-I7), flexskiveinterface (FINT), "mover" (MINT) och signalen som indikerar att nätspänningen är på väg att försvinna ("power fail").

CIO:ns INT-utgång aktiverar vid interruptbegäran signalen INT2 som efter prioritetsavkodaren 5D kommer till CPU:n. CPU:n svarar som ovan med "interrupt acknowledge" som avkodas i 11E och bildar kvittenssignalen MINT2. Med denna signal aktiveras ingången INTACK på CIO:n och då lässtroben (85RD) går låg, lägger CIO:n ut den aktuella vektorn på databussen och ett förlopp enligt SCC:n upprepas.

3.1.8 Seriell kommunikation

Den seriella kommunikationen handhas i ABC 1600 av de två kommunikationskretsarna DART (Z80-DART, Dual Asynchronius Receiver/Transmitter) och SCC (Z8530, Serial Communications Controller). DART:en, som är den enklare av de båda, kan hantera asynkron kommunikation och används mot tangentbord och skrivare. Mot den 25 - poliga kontakten P11 sker kommunikationen med skrivare enligt RS232C. Mot tangentbordet kan DART:en kommunicera via två kontakter. Dels via den 7-poliga DIN-kontakten P3 för direktanslutning av tangentbordet till datordelen med TTL-snitt, dels via bildskärmen ABC 1615 med den 15-poliga kontakten P7 som har buffertkretsar av RS232 C-snitt.

SCC:n hanterar den mer avancerade och generella kommunikationen via de båda seriella kommunikationsmodulerna. Dessa moduler innehåller i standardutförande V24-buffertar men kan vid behov bytas ut mot exv inbyggnadsmodem, RS422-moduler eller interface mot LUX-NET. SCC:n supportar inte bara asynkron kommunikation utan även synkron av typen SDLC/HDLC med NRZ-eller NRZI-kodning. Kommunikationen mot modulerna sker genom de på datakortet placerade anslutningarna P1 och P2. Till skillnad mot DART:en kan data till och från SCC:ns portar också hanteras av DMA. SCC:n har också två inbyggda Baudrate-generatorer, en för varje kanal, vilket eliminerar behovet av speciella klockor för detta ändamål.



De båda periferikretsarnas sätt att kommunicera med CPU:n (i SCC:ns fall även DMA) är likartade men ej identiska. Gemensamt är dock databussen (D0-D7), där kommunikationsdata och registerdata transporteras till och från kretsarna. Val mellan port A och port B sker genom adress A2 och mellan kontroll och data med hjälp av A1. I DART:ens fall dock efter invertering av signalerna i inverterarna 15D/2 och 3. Enable-strob till kretsarna ges med hjälp av de två signalerna $\overline{DRT}$ (för DART) och $\overline{SCC}$ (för SCC).

Dessa signaler har tidigare beskrivits i avsnittet "I/O-hantering". Gemensamt för både SCC och DART är också klockan PCLK på 4MHz, som kommer från PU-kortets klockgenerering.

SCC

Övriga signaler för att generera reset, skriv/läs-strobar och "interrupt acknowledge" skiljer något mellan de båda kretsarna. SCC har separata ingångar för skriv- resp lässtrob under det att DART:en har en gemensam. DART:ens kombinerade läs/skriv-strob heter ZRD under det att SCC:ns lässtrob heter 85RD och skrivstrob 85WR.

Då ZRD, som bildas i PAL 11E är låg och även IORQ är aktiv, kan läsning ske från DART. Vid skrivning är IORQ aktiv men ZRD hög. SCC:ns separata strobar bildas i PAL 10C och är aktiva vid skrivning resp läsning från både CPU och DMA. Någon separat IORQ-signal finns ej men 85RD och 85WR är endast aktiva under en sådan sekvens vilket kontrolleras då signalerna bildas i PAL-kretsen.

Vid reset av SCC:n aktiveras både skriv- och läs-strob samtidigt, under det att DART:en har en speciell reset-ingång. Studera PAL-listorna för mer information om hur signalerna bildas.

Även vid "interrupt acknowledge" skiljer sig signaleringen mellan de två kretsarna men detta beskrivs under avsnittet "Interrupt-hantering".

DART

Då SCC:n skall kunna kommunicera via DMA:n finns på både A- och B-port en W/R-signal (WAIT/REQUEST). Dessa är sammanslagna till signalen SCCRQ som meddelar DMA:n att SCC:n är klar för överföring av data.

DART:ens A-port som är avsedd för skrivanslutning får Baudrateklocka från CIO:n då den inte har interna Baudrategeneratorer som SCC:n. Porten är dock en generell asynkron port och kan även användas som exv terminalanslutning då fler än en vill arbeta mot datorsystemet.

B-porten däremot är speciellt kopplad för tangentbordsanslutning och kommunicerar endast via ett fåtal signaler. Klockan levereras av tangentbordet via KB-TRXC, data från tangentbordet via KB-RXD, data till tangentbordet vid KB-TXD, och DCD via KB-KEYDOWN som indikerar att någon tangent är nedtryckt.

Via AND-grindarna 13A grindas tangentbordssignalerna ihop från bildskärm- och tangentbords-anslutningarna, så att man kan använda dem alternativt.

Kommunikationsmoduler

Från SCC:ns A- och B-portar går signalerna direkt till de två kommunikationsmodulerna. Där kan signalerna omvandlas och påverkas lite olika beroende på vilken slags modul som är ansluten. I leveransutförande sitter två moduler för V24-anslutning. På korten finns också portkretsar som styrs av signalerna $\overline{\text{ATCE}}$ (V24-kort 1) och $\overline{\text{BTCE}}$ (V24-kort 2) som kommer från specialkontroll-registret 13E. Med dessa kontrollsignaler kan interna klockor eller externa klockor kopplas till SCC:n.

Då porten är ställd för interna klockor kommer SCC:ns TRXC-klocka att dels ge TXC (Transmit Clock) till V24-kontakten och samtidigt via räknarna 2B (på V24-kortet) och byglingsfältet, RTXC-klocka tillbaka. På detta sätt får vi "asynchronous/split speed"-kommunikation, där byglingsfältet avgör vilken hastighet mottagningskloc-kan ska ha. För "single speed"-kommunikation ignoreras RTXC-signalen, vilket görs genom omprogrammering av SCC:n, så att samma klocka används både för mottagning och sändning.

Då kontrollsignalen ($\overline{\text{ATCE}}$ eller $\overline{\text{BTCE}}$) ges låg nivå kommer istället de externa klockorna från den 25-poliga kontakten att kopplas till SCC:ns RTXC och TRXC-ingångar så att de styrs av de utifrån kommande klockorna (synkron kommunikaion möjlig). Alla signaler nivåomvandlas till/från standard V24-nivåer i buffertkretsarna 1A, 1B och 1C.

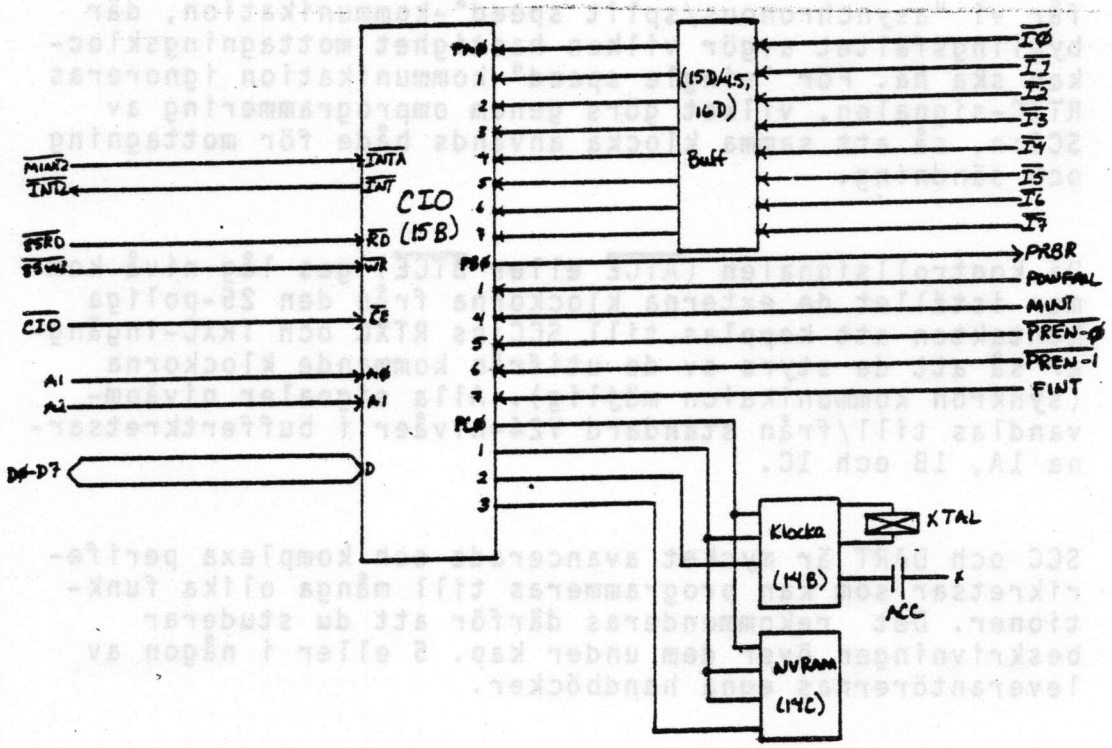
SCC och DART är mycket avancerade och komplexa periferikretsar som kan programmeras till många olika funktioner. Det rekommenderas därför att du studerar beskrivningen över dem under kap. 5 eller i någon av leverantörernas egna handböcker.

3.1.9 CIO + kalender och NVRAM

CIO:n (Counter I/O Z8536) är en periferikrets vars funktion kan programmeras inom vida gränser. Den har tre stycken dubbelriktade parallellportar (två 8-bitars och en 4-bitars), där funktionen kan programmeras för varje bit. Den har också tre "Counter/Timers" som kan lämna klockor till externa enheter.

I ABC 1600 används CIO:n för att samla ihop interruptsignaler från de fyra expansionsbussarna, flexskiveinterface, "mover" och "powerfail". Den levererar också baudrateklocka till DART:ens skrivaranslutning samt hanterar CMOS-kalender och NVRAM.

Då överföring av information via CIO:n är av ringa omfattning hanteras den ej av DMA utan endast av CPU:n. Den tillhör samma familj av periferikretsar som SCC:n och hanteras signalmässigt på samma sätt.



CIO:ns portar

PA-porten (PA0-7) är programmerad att ta mot interruptsignaler från expansionsbussarna enligt följande:

Port	Signal	In/Utgång
PA0	Interrupt BUS2	I
PA1	Interrupt BUS1	I
PA2	Interrupt BUSOX*2	I
PA3	Interrupt BUSOX*3	I
PA4	Interrupt BUSOX*4	I
PA5	Interrupt BUSOX*5	I
PA6	Interrupt BUSOX	I
PA7	Interrupt BUSOI	I

PB-porten används förutom att samla interrupt även för att generera baudrateklocka till DART:ns skrivarutgång och känna av enable-strobar för DMA-överföring till BUS0 och 1.

Port	Signal	In/Utgång
PB0	"Printer baudrate" (PRBR)	U
PB1	"Power fail"/ok (POWERFAIL)	I
PB2	Ej använd	
PB3	Ej använd	
PB4	"Mover interrupt" (MINT)	I
PB5	"Peripheral enable" BUS1 ($\overline{\text{PREN-1}}$)	I
PB6	"Peripheral enable" BUS0 ($\overline{\text{PREN-0}}$)	I
PB7	"Floppy" interrupt (FINT)	I

CMOS-kalender

PC-porten används för att hantera CMOS-kalender (14B) och NVRAM (14C). CMOS-kalendern kan med hjälp av kristallen X1 och sin laddningsbara ackumulator fortsätta sitt arbete efter det att spänningen till datorsystemet slagits från. Pga CMOS-teknologin i kombination med den relativt låga frekvensen (32.768 KHz) kan accumulatorn driva kalendern i minst 30 dagar efter det att spänningen slagits från.

Då datorsystemet är påslaget,laddas ackumulatorn kontinuerligt. Det gör att aktuell tid (år, månad, dag, timme, minut, sekund) alltid finns tillgänglig då systemet skall användas.

Med hjälp av ett speciellt program "/etc/setbclock" kan klockan sättas och med kommandot "cat /dev/bclock" kan den läsas.

NVRAM

NVRAM:et (Non Volative RAM) är ett litet RAM-minne (16x16 bitar) som har den egenskapen att det inte tappar informationen då spänningen slås av. NVRAM:et innehåller data (parametrar) som är av stor betydelse då systemet skall startas och kallas därför ibland parameterminne.

Följande parametrar ingår:

- -Systemterminalens baudrate (9600 Baud)
- -BOOT-nivå, dvs vid vilken nivå (0-3) som operativsystemet skall stanna, vid automatisk start (normal start)
- -BOOT-enhet, den enhet (exv Winchester eller flexskiva) som operativsystemet ska laddas från
- -Tidszon, anges i förhållande till GMT (Greenwich Mean Time)
- -CPU:ns klockfrekvens (här 8MHz)

Vid 4-bits-porten PC sköts all hantering av både NVRAM och CMOS-kalender. Data till och från enheterna sker seriellt och klockning vid läsning och skrivning genereras av "Counter/Timer" 3 i CIO:n. Signalerna fördelar sig enligt följande:

Port	Signal
PC0	Klocka (för klockning av data)
PC1	Dubbelriktad seriell data
PC2	Enable CMOS-kalender
PC3	Enable NVRAM

Studera beskrivningen av CIO:n och avsnittet "interrupthantering".

3.1.10 DMA

DMA:ns (Z80 DMA - Direct Memory Access) främsta uppgift är att öka överföringshastigheten mellan två enheter samt att avlasta systemets CPU. Data kan överföras mellan två minnesareor eller mellan minne och I/O men också mellan två I/O-portar. DMA:erna hanterar överföringen utan inblandning av CPU:n dvs de kan själva kontrollera och styra data- och adressbuss och de strobar som normalt kontrolleras av CPU:n. I DMA:n finns ett flertal olika register som före en överföring måste uppdateras och som beskriver hur överföringen ska gå till och mellan vilka till- och från-enheter den ska ske. Dessa register vars innehåll bestäms av CPU:n kan du läsa mera om i kretsbeskrivning över DMA:n.

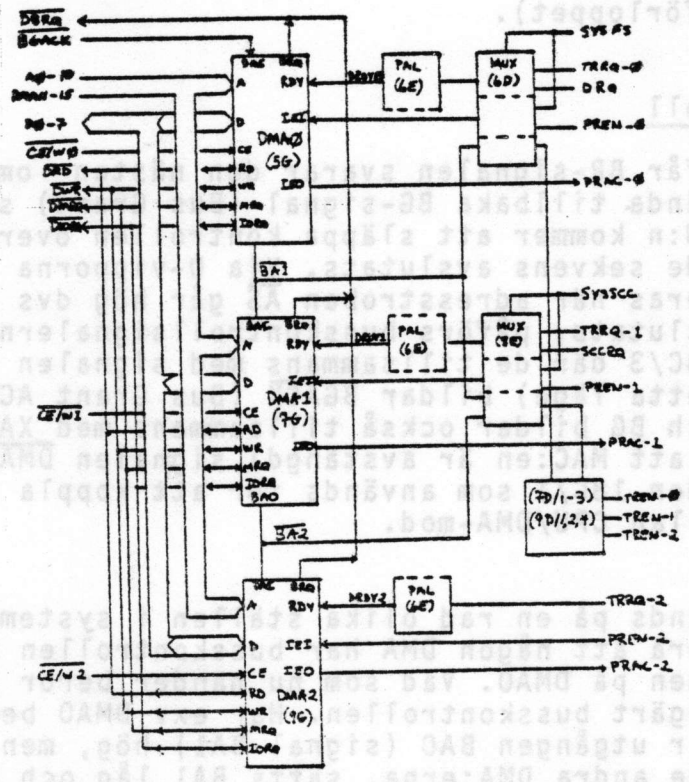
I ABC 1600 finns tre DMA:er (DMA0-2) som är uppdelade så att de hanterar olika I/O-enheter, men kan alla arbeta mot de båda minnesareorna primär- och grafikminne.

Uppdelningen på I/O-enheter är enligt följande:

DMA0 = BUS0I, BUS0X och flexskiveinterface

DMA1 = BUS1 och SCC (Seriell kommunikation)

DMA2 = BUS2



När flera I/O-enheter kan arbeta mot samma DMA (DMA0 och 1) bestäms vilken av enheterna som ska ha tillgång till "sin" DMA med signalerna SYSFS och SYSSCC som styrs från specialkontrollregistret. Se "I/O-hantering".

DMA-adresser och handskakning

DMA:erna kan via sina 16 adressledningar (A0-A10, DMA11-15) endast addressera ett 64 Kbyte stort område. För att kunna nå alla minnesoch I/O-positioner finns därför en enhet som i likhet med CPU:ns MAC, kan omvandla DMA:ns adresser till fysiska minnespositioner. Enheten som kallas DMA-MAP består av två små bipolära RAM vars innehåll skrivs in av CPU:n före varje överföring.

Förutom adress-, data- och kontrollbuss har DMA:n också signaler för handskakning mot CPU:n och de I/O-enheter som hanteras av respektive DMA. Då någon av I/O-enheterna som är anslutna till DMA:n aktiverar RDY-ingången, svarar DMA:n med att begära busskontrollen genom att aktivera DBRQ-signalen (DMA Bus Request).

$\overline{\text{DBRQ}}$-signalerna är sammankopplade på de tre DMA:erna och via OR-grinden 10E/3 ges $\overline{\text{BR}}$-signal (Bus Request) till CPU:n under förutsättning att DMADIS från specialkontrollregistret ej är aktiv. (DMADIS sätts aktiv vid reset eller systemstart och är aktiv under själva uppstartsförloppet).

Busskontroll

Då CPU:n får BR-signalen svarar den nästan omedelbart med att sända tillbaka BG-signal (Bus Grant) som talar om att CPU:n kommer att släppa kontrollen över bussarna då pågående sekvens avslutats. Via D-vipporna i 10D/2, som aktiveras när adresstroben $\overline{\text{AS}}$ går hög dvs då sekvensen avslutats, påförs busskontrollsignalerna NAND-grinden 18C/3 där de tillsammans med signalen IOC (ej aktiv i detta läge) bildar $\overline{\text{BGACK}}$ (Bus Grant ACKnowledge). BR och BG bildar också tillsammans med $\overline{\text{XADR-OFF}}$ (talar om att MAC:en är avstängd) signalen $\overline{\text{DMAOK}}$ via NAND-grinden 18C/1 som används för att koppla om DMA-MAP:en mellan CPU/DMA-mod.

$\overline{\text{BGACK}}$ används på en rad olika ställen i systemet för att indikera att någon DMA har busskontrollen bl a till BAI-ingången på DMA0. Vad som nu händer beror på vilken DMA som begärt busskontrollen. Har exv DMA0 begärt bussen förblir utgången BA0 (signal BA1) hög, men är det någon av de andra DMA:erna, sätts BA1 låg och BAI-ingången på DMA1 går låg osv. De nu bildade signalerna BA1 och BA2 används för att till systemet signalera vilken av de tre DMA:erna som har busskontrollen.

BGACK används också för att stänga av CE/W-signalerna (Chip Enable/Wait) till DMA-kretsarna från I/O-avkodningen i portkretsen 5E/2. I DMA-mod används CE/W-signalerna istället som WAIT-signaler som via porten 5E/1 påverkas av DTACK-signalen. DTACK-signalen utgör den samlade kvittenssignalen från bl a minne, I/O och videoenhet. Detta gör att enheter med olika åtkomsttid kan anslutas till DMA:erna.

CPU:n skriver i DMA:ns register

Då CPU:n förbereder en överföring med hjälp av DMA måste en hel del uppgifter skrivas in bl a från vilken adress data ska överföras, om adressen ska räknas upp eller ner för varje byte (från minne) eller om adressen ska vara fast (I/O). Även till adressen måste programmeras på samma sätt, dessutom måste anges hur timingen i överföringen ska vara och hur data ska överföras (en byte i taget, ett antal bytes eller med s k sökmod.)

För att skriva in dessa data aktiverar CPU:n enablestroben (DMA0, DMA1 eller DMA2) för aktuell DMA samtidigt som DIORQ (samma som IORQ i CPU-mod) och DWR aktiveras och adressen till det avsedda registret anges på adressbussens lägre halva (A0-A7). Data kan nu skrivas till registren och DMA:n göras klar för överföringen. På samma sätt kan data läsas ur DMA:ns register om lässtroben DRD aktiveras i stället för skrivstroben DWR.

> OBS!
> Om DMA-kretsarna av någon anledning "spårar ur" måste datorsystemets strömbrytare slås av, då varken CPU:n med hjälp av resetinstruktion eller manuell reset påverkar kretsarna.

DMA0

Med hjälp av de två signalerna SYSFS och SYSSC avgörs vilken av IO-enheterna som ska få tillgång till "sin" DMA. Signalerna påverkar muxarna 6D (SYSFS styr växling mellan flexskiveinterface och BUS0) och 8E (SYSSCC styr växling mellan SCC och BUS1) som kopplar om signalerna för handskakning och busskontroll till respektive DMA. Då SYSFS är hög kopplas signalen DRQ (Data Request från flexskiveminnet) till utgång 4 och påförs PAL 6E där den bildar signalen DRDY0, som aktiverar RDY-signalen till DMA0.

Utgång 7 (6E) blockerar via NOR-grinden 7D/2 och NAND-grinden 9D/4 möjligheten att TREN-0 till bussinterfacet avges. Via utgång 9 blockeras också signalen DBRQ som via PAL12E medverkar vid signalgenerering till bussinterfacet. Utgång 12 slutligen kopplar SYSFS-signalen till IEI-ingången på DMA0 som via IEO ger signalen PRAC-0.

Observera att IEI och IEO inte används för interruptprioritet då DMA-erna ej använder interruptsinalering utan till att generera PRAC-signaler. Då SYSFS är låg dvs då BUS0 har möjlighet att utnyttja DMA:n kopplar utgång 4 TRRQ-0 till RDY på DMA:n, utgång 7 signalen BA1 till NOR-grinden 7D/2. BA1 hög, ger att TREN-0 aktiveras under det att BA1 låg gör att signalen blockeras. (Då BA1 har låg nivå innebär detta att någon annan DMA har begärt bussen). Utgång 9 aktiverar DBR0 då DMA0 begärt bussen och utgång 12 ger via IEI och IEO på DMA0 aktiv PRACK-0-signal.

DMA1

På liknande sätt sker omkopplingen till DMA1 via stroben SYSCC och mux 8E. Då SYSCC är hög (SCC:n har tillgång till DMA:n) har utgång 4 låg nivå och IEI/IEO på DMA1 aktiverar PRACK-1. Utgång 7 som aktiveras av SCCRQ ger DRDY1 via PAL6E och utgång 9 kopplar BA2 till DBR12. (Detta innebär att då SCC:n begärt DMA:n kommer DBR12 ej att aktiveras, under det att då BUS2 begär tillgång till sin DMA kommer signalen att aktiveras i detta läge). Utgång 12 slutligen blockerar signalen TREN-1 till BUS1 via NOR-grinden 7D/3 och NAND-grinden 9D/2.

Då SYSFS är låg kommer BUS1 att ha tillgång till DMA1 och utgång 4 på mux 8E att ge PRAC-1 via IEI/IEO på DMA1. Utgång 7 påverkas av TRRQ-1 från BUS1 och ger via PAL6E, DRDY1 till DMA:n och utgång 9 som aktiveras då BA1 är låg ger DBR12 till PAL12E. Utgång 12 kopplar BA2 till NOR-grind 7D/3, där låg nivå på BA2 blockerar TREN-1 under det att BA2 hög ger möjlighet till att signalen aktiveras.

DMA2

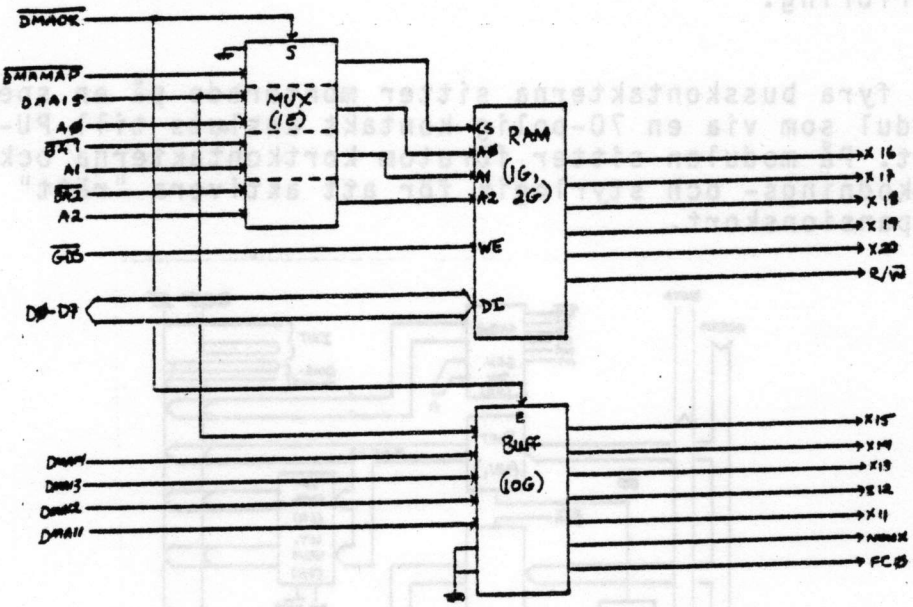
DMA2 har endast en I/O-enhet att hantera, BUS2. Detta gör att någon mux ej behövs för att koppla om signalerna utan de kan påföras DMA:n direkt efter vissa kontroller. PREN-2 från BUS2 går via inverteraren 15D/6 till DMA:ns IEI-ingång. Utgången IEO lämnar nu PRAC-2 tillbaka till I/O-kortet. Från kortet kommer nu TRRQ-2 till PAL6E där den resulterar i DRDY2 efter vissa kontroller. DRDY-2 påverkar RDY-ingången på DMA2 som nu kommer att begära bussen.

Då DMA2 nu fått kontroll över bussarna lägger den ut DRD och DMRQ som bland annat avkodas i NOR-grinden 7D/1 och ger hög signal till NAND-grinden 9D/1. Den andra ingången på 9D/1 är den inverterade signalen av BA2 som är aktiv då DMA2 får möjlighet att ta busskontrollen. Då båda signalerna är höga in på NAND-grinden resulterar detta i att TREN-2 till BUS2 blir aktiv och överföringen kan börja.

Dessa muxar och grindar som nu beskrivits kontrollerar att handskakningen från periferikretsar och bussar till respektive DMA hanteras på ett riktigt sätt. Den logiska ordningen mellan handskakningssignalerna är således: PREN > PRAC > TRRQ > TREN.

DMA-MAP

Som tidigare omtalats räcker inte DMA:erns adresseringsförmåga till för att täcka hela minnes och I/O-området. På grund av detta har man infört en DMA-MAP som ska peka ut det minnesblock, som data ska transporteras till eller från.



Via muxen 1E kan CPU:n adressera och skicka data till DMA-MAP:ens minnen 1G och 2G. Muxen styrs av signalen DMAOK som sköter växlingen mellan CPU/DMA-mod. I CPU-mod kopplas adresserna A0-A2 till minnenas adressingångar under det att stroben DMAMAP från specialkontrollregistret via CS-ingången, aktiverar minnet och GDS från PAL12E ger möjlighet att skriva genom att aktivera WE-ingången. På detta sätt skrivs sex adresser in som lagras i minnena (två för varje DMA). Detta gör att en överföring av data kan ske över två 64-Kbytes blockgränser. De data som skrivs in resulterar sedan i adresserna X16-X20 samt skriv/lässtroben R/W.

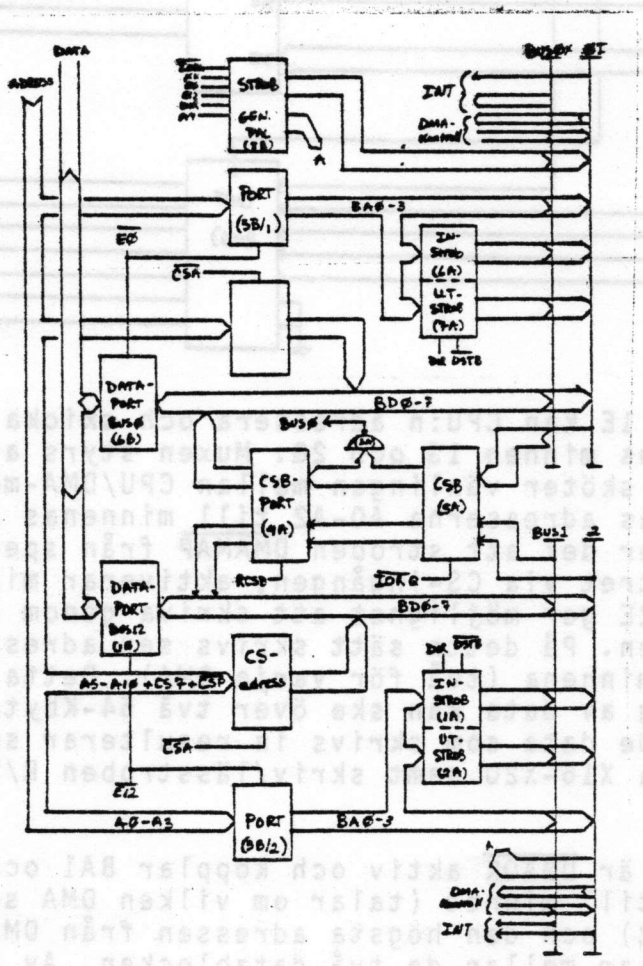
I DMA-mod är DMAOK aktiv och kopplar BA1 och BA2 som adresser till minnet (talar om vilken DMA som adresserar minnet) och den högsta adressen från DMA:erna, DMA15 växlar mellan de två datablocken. Av de lägre adresserna kopplas adresserna DMA11-DMA15 via portarna 10G/1 och 2 och bilder här X11-X15 under det att A0-A10 från DMA:erna adresserar minnet direkt. Från portkretsarna får man också definierade nivåer på signalerna NONX och FC0.

3.1.11 Expansionsbussen

För att öka flexibilitet och användbarhet är ABC 1600 försedd med fyra stycken expansionsplatser. De kan utnyttjas för att ansluta olika former av interface som exv Winchester- och Flexskivestyrkort, kommunikationsprocessor för LUXNET-anslutning eller för I/O-kort vid mät- och styrtillämpningar.

Med hjälp av expansionsbussen kan man konfigurera systemet så att det passar för en speciell tillämpning. De fyra kortplatserna hanteras med hjälp av interrupt/CSB-signalering och diverse I/O-strobar samtidigt som data överförs via en 8-bitars databuss. De 64-poliga kortkontakterna innehåller också klocka och speciella handskakningssignaler som används vid DMA-överföring.

De fyra busskontakterna sitter monterade på en speciell modul som via en 70-polig kontakt ansluts till PU-kortet. På modulen sitter förutom kortkontakterna också avkodnings- och styrlogik för att aktivera "rätt" expansionskort.



Styrningen av korten kan hanteras från programvaran via en speciell "driver" (/dev/DBinoutb). Dock har enheter typ winchester, magnetband och LUX-NET egna drivers. För att systemet ska veta vilket kort som påkallar uppmärksamhet har alla busskontakterna separata interruptledningar som samlas ihop av CIO:n, som in sin tur skickar interruptbegäran till CPU:n och en vektor till en programmodul som sedan kan hantera just detta kort.

Behöver man fler än de fyra kortplatser som finns i ABC 1600, kan man koppla till en expansionslåda. Den kortplats som är avsedd att hantera expansionslådan är BUSOX och skiljer sig lite från de tre andra. Den är utrustad med fem interruptledningar och lika många för CSB. Det gör att fem expansionskort kan anslutas till denna busskontakt vilket totalt gör åtta stycken i systemet. Är man i behov av ännu fler kortplatser kan man använda en expansionslåda utrustad med interrupt- och CSB-avsökning med vars hjälp upp till 32 kort kan hanteras.

De fyra kortplatserna som benämns BUSOI, BUSOX, BUS1 och BUS2 är på busskortet uppdelade i två grupper där BUSOI och BUSOX utgör den ena (BUSO) och BUS1 och BUS2 den andra (BUS12). De båda grupperna har separat logik för avkodning och separata dataportar som aktiveras av signalerna E0 respektive E12.

Vid DMA-överföring till och från busskontakterna hanteras BUSOI och BUSOX av DMA0 (används även av Flexskiveinterfacet), BUS1 av DMA1 (används även av SCC:n), och BUS2 av DMA2 som är ensam användare av denna DMA. Det är till denna kortplats som Winchesterinterfacet är ansluten, vilket gör att "datatrafiken" normalt är mycket intensiv just här.

Kortadressering

Den metod som här används för att hantera expansionsbussen bygger på att man vet vilken position ett visst I/O-kort har, dvs i vilken kortplats det sitter. Systemet kan själv ta reda på detta genom att sända ut en kortadress på databussen via porten 1C/1 och 2 (gäller BUS1 och BUS2) och porten 6C/1 och 2 (för BUSOI och BUSOX).

Kortadressen utgörs av CPU-adresserna A5-A10 samt CS7 som genereras via specialkontrollregistret. Genom portarna kopplas också signalen $\overline{CSP}$ (Card Select Pulse) från PAL12E på PU-kortet som bildar kortvalstrobarna $\overline{CS0}$ (BUS0) och $\overline{CS12}$ (BUS12). De båda portarna aktiveras av signalen $\overline{CSA}$ (Card Select Adress) som också bildas i PAL12E.

Ute på de olika I/O-korten jämförs nu den mottagna kortadressen med kortets egen adress och det kort där de båda adresserna stämmer överens aktiverar sin CSB-signal. CSB-signalerna från de olika kortplatserna läses in i registret 5A med hjälp av signalen $\overline{IORQ}$. I NAND-grinden 3A avkodas sedan alla CSBsignaler från BUSOI och BUSOX så att om någon av dess signaler aktiverats blir också utgången från grinden, signalen BUSO, aktiv.

BUSO-signalen tas omhand i PAL 12E på PU-kortet där den används för att avgöra om BUSOI eller BUSOX (enablestrob E0) eller BUS1 eller BUS2 (enablestrob E12) ska aktiveras. Strobarna E0 och E12 aktiverar dataportarna till respektive halva på busskortet och då $\overline{RCSB}$-strob avges till CSB-porten (4A/1 och 2) öppnas den (se "I/O-hantering") och CSBdata kan nu avläsas på databussen genom någon av dataportarna 6B (styrs av E0) eller 1B (styrs av E12) beroende på BUSO-signalens läge. Systemet vet nu vilken kortadress som motsvaras av en viss position.

Utökning av antalet kortplatser

Om en expansionsenhet med mer än fem kortplatser förutom kabeln är ansluten till BUSOX kan systemet inte direkt avgöra positionen utan måste göra några kompletterande avläsningar. Med hjälp av $\overline{EXP}$-stroben och bussadresserna BA0-4 kan systemet klara av att skilja på upp till 32 olika kort och fastställa deras position i expansionsenheten. BUSEN-signalen styr portarna i expansionskabeln.

När det aktuella kortet är uppvalt kan register och portar på kortet styras av in- och ut-strobar från avkodarna 6A och 7A (BUSOI och BUSOX) eller 1A och 2A (BUS1 och BUS2). Avkodarna i sin tur aktiveras av signaler från PAL8B och av DIR-signalen från PAL12E som också styr riktningen på dataportarna 1B och 6B. Då DIR-signalen är låg ligger portarna riktade utåt så att skrivning till bussen kan ske och vice versa.

Strobgenerering

För att strobar ska kunna avges måste avkodarnas tre enable-ingångar aktiveras.

DSTB-0
BA4-0
DIRW/$\overline{R}$-0 — 7A — Utstrobar till BUSOI och X

DSTB-0
BA4-0
DIRW/$\overline{R}$-0 — 6A — Instrobar till BUSOI och X

DIR
DSTB-12
BA4-12
DIRW/$\overline{R}$-12 — 2A — Utstrobar till BUS1 och 2

DIR
DSTB-12
BA4-12
DIRW/$\overline{R}$-12 — 1A — Instrobar till BUS1 och 2

När en av avkodarna aktiverats kommer adresserna BA1 - BA3 att styra vilken strob som avges. Tabellen nedan beskriver vilka strobar som bildas för de olika adresserna och i PAL-listan kan du studera när enablestrobarna aktiveras.

Tabell över utstrobar BUSOI och X (7A)

DSTB-0 = låg, BA4-0 = låg, DIRW/$\overline{R}$-0 = hög

BA3	BA2	BA1	Strob
0	0	0	OUT-0
0	0	1	Ej använd
0	1	0	$\overline{C1-0}$
0	1	1	$\overline{C2-0}$
1	0	0	C3-0
1	0	1	C4-0
1	1	0	Ej använd
1	1	1	Ej använd

Tabell över instrobar BUSOI och X (6A)

DSTB-0 = låg, BA4-0 = låg, DIR = hög

BA3	BA2	BA1	Strob
0	0	0	$\overline{\text{INP-0}}$
0	0	1	$\overline{\text{STAT-0}}$
0	1	0	$\overline{\text{OPS-0}}$
0	1	1	Ej använd
1	0	0	Ej använd
1	0	1	Ej använd
1	1	0	Ej använd
1	1	1	Ej använd

Tabell över utstrobar BUS1 och 2 (2A)

DSTB-12 = låg, BA4-12 = låg, DIRW/$\overline{\text{R}}$-12 = hög

BA3	BA2	BA1	Strob
0	0	0	OUT-12
0	0	1	Ej använd
0	1	0	$\overline{\text{C1-12}}$
0	1	1	$\overline{\text{C2-12}}$
1	0	0	$\overline{\text{C3-12}}$
1	0	1	$\overline{\text{C4-12}}$
1	1	0	Ej använd
1	1	1	Ej använd

Tabell över instrobar BUS1 och 2 (1A)

DSTB-12 = låg, BA4-12 = låg, DIR = hög

BA3	BA2	BA1	Strob
0	0	0	$\overline{\text{INP-12}}$
0	0	1	$\overline{\text{STAT-12}}$
0	1	0	$\overline{\text{OPS-12}}$
0	1	1	Ej använd
1	0	0	Ej använd
1	0	1	Ej använd
1	1	0	Ej använd
1	1	1	Ej använd

DMA-hantering av expansionsbussen

Då överföring från bussarna ska ske med hjälp av DMA, finns ett par speciella signaler för handskakning. Dessa signaler är PREN, PRAC, TRRQ och TREN. PREN (PEripheral ENable) genereras av I/O-kortet då kortet är uppvalt och CPU:n bestämt att den kommande överföringen ska ske med hjälp av DMA.

Via DMA:ernas IEO-utgångar aktiveras PRAC-signalen (PEripheral ACKnowledge) som talar om för I/O-kortet att börja insamling av data eller att göra sig färdig för att ta mot data. När detta är klart aktiverar kortet TRRQ (TRansfer ReQuest) (för varje byte som överförs) som efter diverse kontroller aktiverar aktuell DMA (den DMA som är kopplad för att hantera denna busskontakt) genom att påverka dess RDY-ingång.

Resultatet av detta blir att DMA:n begär busskontroll. När sedan DMA:n fått busskontroll och styr förloppet själv, avges TREN (TRansfer ENable) till busskortet och byten överförs.

Studera listorna över PAL 8B och PAL 12E så får du en mer detaljerad insikt i de olika bussignalernas inbördes beroende samt avsnittet "DMA" som beskriver hur handskakningen går till.

Bussanslutning i expansionsplats

A Funktion	Pin nr	B Funktion
—12 V	1	—12 V
0 V	2	0 V
BPCLK•	3	BPCLK
0 V	4	0 V
INT•	5	0 V
D7	6	0 V
D6	7	
D5	8	
D4	9	
D3	10	XINT•5
D2	11	XINT•4
D1	12	XINT•3
D0	13	XINT•2
CSB•	14	XCSB•2
BRST•	15	XCSB•3
STAT•	16	XCSB•4
INP•	17	XCSB•5
C4•	18	
C3•	19	
C2•	20	
C1•	21	EXP•
OUT•	22	BUSEN•
CS•	23	DSTB•
NMI•	24	0 V
OPS•	25	A4
	26	A3
TREN•	27	A2
TRRQ•	28	A1
PRAC•	29	A0
PREN•	30	DIRW/R•
+5 V	31	+5 V
+12 V	32	+12 V

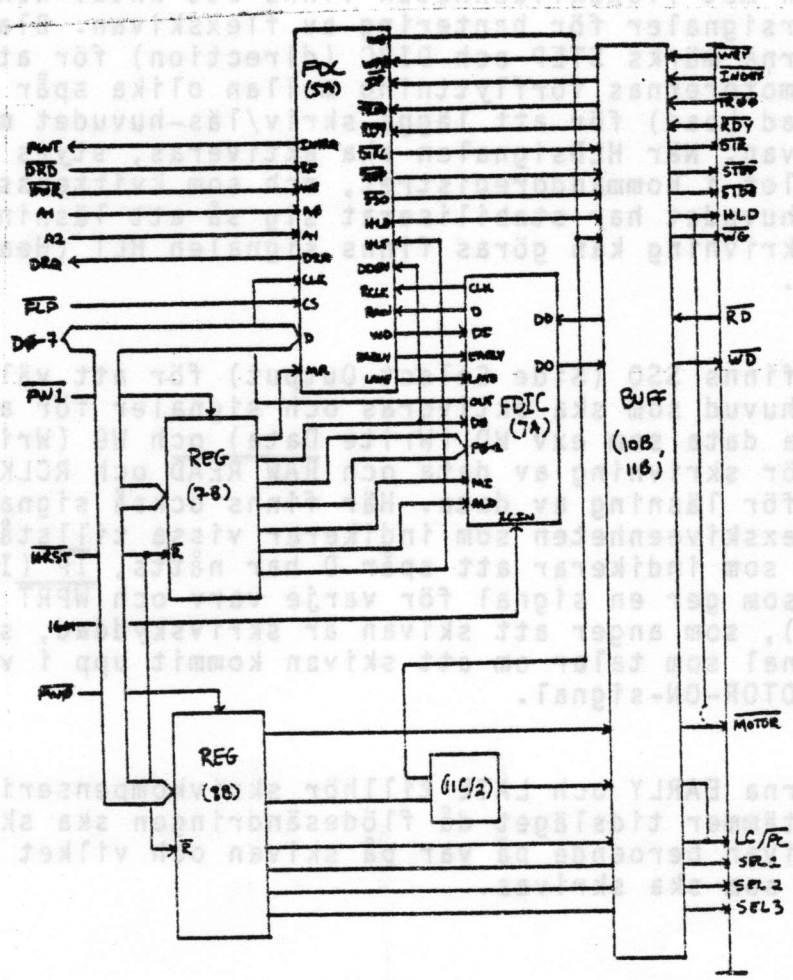
BUSEN•, EXP•, XINT•, XCSB• och NMI• finns bara i expansionsplats 0.

3.1.12 Flexskiveinterface

Det inbyggda flexskiveinterfacet i ABC 1600 kan hantera tre flexskiveenheter. De parametrar som skiljer mellan olika flexskiveformat och ev även leverantörer kan styras med mjukvara vilket gör att interfacet kan hantera olika flexskiveformat utan ombygling.

Interfacet används företrädesvis till den inbyggda 130 mm (5.25") stora flexskiveenheten men via den 25-poliga kontakten P4 kan även flexskiveminnen av typ ABC 830, 832, 834 och 838 anslutas. Observera att den interna flexskiveenheten måste vara byglad som enhet nr2, för att inte omöjliggöra anslutning av ett externt flexskivminne, då dessa sedan tidigare har adress 0 och 1.

Styrningen av interfacet och därigenom de olika flexskiveenheterna hanteras av operativsystemet via en sk "driver". (En "driver" är en speciell programmodul som hanterar yttre enheter typ skivminne, flexskiveminnen, bandstationer, terminaler etc). I "/dev"-biblioteket kopplas sedan en viss "driver" tillsammans med enhetstyp, portnummer och enhetsnamn och man får en programmodul som kan hantera exv flexskiveminnen.



Hårdvarumässigt består interfacet av en styrkrets (FDC1797,) en dataseparator (FDC9229), två register för styrparametrar (7B och 8B) samt ett antal buffertar mot flexskiveenheterna.

Styrkretsen 1797 är en speciell periferikrets för hantering av flexskiveenheter. Den kan programmeras att hantera flexskivor med olika antal spår, kodningsformat (FM och MFM), sektorstorlek och skrivkompensering. Mot CPU/DMA-sidan finns signaler som hanterar data (DAL0-7), skriv- och lässtrobar (WE och RE), enablestrob (CS), resetsignal (MR) och två adressingångar (A0 och A1) för adressering av de interna registren. Här finns också två signaler som hanterar interrupt (INTRQ eller FINT som signalen kallas i ABC 1600) som avges då ett kommando utförts och "data request" (DRQ) som ges till DMA:n då det är dag att begära busskontrollen för en dataöverföring.

Signalering mot flexskiveenhet

Det är CPU:n i ABC 1600 som skriver i de interna registren för att sätta upp villkoren för överföringen under det att DMA:n hanterar all överföring av data från/till flexskiveenheten via 1797:an.

På sidan mot flexskiveenheten finns ett antal kontroll- och styrsignaler för hantering av flexskivan. Bland signalerna märks STEP och DIRC (direction) för att styra stegmotorernas förflyttning mellan olika spår och HLD (Head Load) för att lägga skriv/läs-huvudet mot flexskivan. När HLDsignalen ska aktiveras, styrs av innehållet i kommandoregistret, och som kvittenssignal på att huvudet har stabiliserat sig så att läsning eller skrivning kan göras finns signalen HLT (Head Load Timing).

Vidare finns SSO (Side Select Output) för att välja vilket huvud som ska aktiveras och signaler för att överföra data som exv WD (Write Data) och WG (Write Gate) för skrivning av data och RAW READ och RCLK (Read CLock) för läsning av data. Här finns också signaler från flexskiveenheten som indikerar vissa tillstånd exv TRACK00 som indikerar att spår 0 har nåtts, IP (Index Pulse) som ger en signal för varje varv och WPRT (Write PRotect), som anger att skivan är skrivskyddad, samt RDY-signal som talar om att skivan kommit upp i varv efter MOTOR-ON-signal.

Signalerna EARLY och LATE tillhör skrivkompenseringen och bestämmer tidsläget då flödesändringen ska skrivas till skivan beroende på var på skivan och vilket bitmönster som ska skrivas.

Styrkrets för flexskiva

Hur de olika kontroll- och styrsignalerna ska behandlas styrs av innehållet i 1797:ans interna register.

Register i FDC1797

A1	A0	Läsning ($\overline{RE}$)	Skrivning $\overline{WE}$
0	0	Statusregister	Kommandoregister
0	1	Spårregister	Spårregister
1	0	Sektorregister	Sektorregister
1	1	Dataregister	Dataregister

Då data ska transporteras till eller från flexskivan sätts kommandoregistret upp för sökning och spår- och sektor-registren till den avsedda positionen.

Flexskiveenhetens stegmotor flyttar nu skriv/läs-huvudet till rätt position och då detta är klart signalerar 1797:an $\overline{INTRQ}$ (FINT) till CPU:n som nu kan ge ytterligare kommandon som exv att läsa eller skriva data på flexskivan. När 1797:an utfört även detta kommando och data finns att hämta i dataregistret (läsning) eller dataregistret är tomt (vid skrivning) skickas DRQ till DMA0 och DMA:n tar över busskontrollen och hanterar överföringen av data.

I beskrivningen över FDC1797 kan du studera vilka kommandon och parametrar som kan användas.

Dataseparator

Till sin hjälp att separera data-signalen från flexskiveenheten till en klocksignal och en datasignal och för att skriva data till enheten med rätt skrivkompensering finns dataseparatorn FDC9229. Den hanterar också klockgenerering så att 1797:an får rätt klockfrekvens beroende på flexskivetyp (130 eller 200 mm) och kodningsformat (FM eller MFM).

Huvudklocka till flexskiveinterfacet är 16M (16 MHz-klocka) som i 9229:an delas till 1 eller 2 MHz innan den bildar klocka till FD1797. Frekvensdelning beror på ingångarna till MINI och DENS som anger miniflexskiva (130 mm) eller standardflexskiva (200 mm) och kodningsformatet (FM eller MFM).

Tabell över klockdelning i FDC9229

DENS	MINI	Klockfrekvens(MHz)	Flexskive-typ/kodning
0	0	2	SF/DD
0	1	1	MF/DD
1	0	2	SF/SD
1	1	1	MF/SD

För att hantera skrivkompenseringen finns fem ingångar EARLY, LATE och P0-P2. EARLY och LATE genereras i FD1797 och är beroende av bitströmmen av data ut och talar om vilka flödesändringar som ska tidigareläggas och vilka som ska fördröjas i förhållande till normalläget.

Detta görs för att flödesändringarna vid läsning av flexskivan ska ge så tydliga indikeringar som möjligt och kompensera för läs/skrivhuvudets "tröghet". Med hjälp av signalerna P0-P2 bestäms tidsförskjutningen i förhållande till normalläget.

Dataflödet till och från styrkretsen FD1797 går alltid via dataseparatorn. Vid skrivning kommer data från WD på styrkretsen till ingången DI(WDIN) på dataseparatorn och efter eventuell kompensation skrivs data ut via DO-utgången (WDOUT) till flexskivan. Data från flexskivan kommer in till dataseparatorn på ingången DD ($\overline{\text{DSKD}}$) och efter att ha bahandlats i en intern PLL delas datasignalen upp så att mottagna data och dataklocka kan levereras till styrkretsen som separata signaler (RAW och RCLK).

Parameterregister

De två registren för styrparametrar 7B och 8B, innehåller både parametrar för flexskiveenheten och kontrollparametrar för FD1797 och FDC9229B. Med hjälp av klockstrobarna FW0(8B) och FW1(7B) laddas registren från databussen. Båda registren påverkas också av resetsignalen $\overline{\text{HRST}}$ vilket gör att de nollställs vid reset eller systemstart.

I tabellerna på nästa sidorna beskrivs utsignalerna samt vilken databit (D0-D7) som påverkar dem.

Tabell över register 7B styrs av FW1

Data in	Utsignal	Beskrivning
D0	MR	(Master Reset) ger reset till FDC1797.
D1	DDEN	(Double DENsity) till FDC 1797 och FDC9229B Låg nivå ger Double Density och hög nivå Single Density
D2	HLT	(Head Load Timing) till FDC1797
D3	MINI	Till FDC9229B. Hög nivå indikerar miniflexskiva (130 mm), låg nivå ger standardflexskiva 200 mm)
D4	HLD	(Head Load) till FDC9229B
D5	P0	Ger tillsammans med P1 och P2 tidsförskjutningens storlek vid skrivkompensering.
D6	P1	Se P0
D7	P2	Se P0

Tabell över register 8B styrs av FWO

Data in	Utsignal	Beskrivning
D0	SEL1	Aktiverar Flex. nr 0 vid hög nivå
D1 " " "	SEL2	" " 1
D2 " " "	SEL3	" " 2
D3	MOTOR	Ger start av drivmotor vid hög nivå
D4	LC/PC	Ger låg skrivström till läs/skrivhuvudet vid hög nivå. Används endast till standard-flexskivor (200 mm) och endast från spår 43 och inåt på skivan.
D5	LC/PC	Samma som ovan men grindad med WG (Write Gate).

Mer om detta finns i beskrivningen över FDC1797 och FDC9229 under kap. 5 eller någon av leverantörernas handböcker.

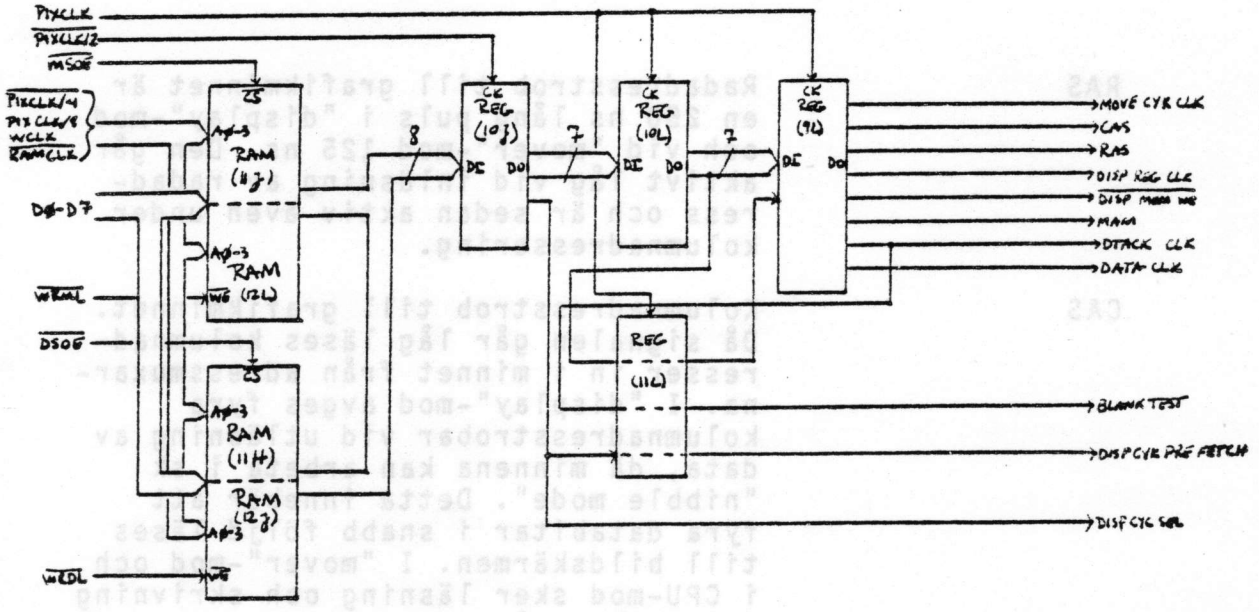
3.2 Videoenhet

3.2.1 Sekvenskontroll videoklockor

Sekvenskontrollens uppgift är att leverera styrsignaler till videoenhetens olika delar och att ge en riktig tidsfördelning mellan dessa. De styrsignaler som behövs är dessutom olika, beroende på vilken del som för tillfället är aktiv mot grafikminnet. Då data ska läsas ut till videoregistret för vidare transport till bildskärmen, behövs en viss tids-sekvens under det att då "movern" eller CPU:n flyttar data inom grafikminnet, behövs en annan. För att lösa denna olikhet vad gäller tids-sekvens finns en tabell belägen i fyra små snabba RAM-minnen om vardera 16 x 4 bitar.

Två av minnena genererar styrsekvenser i "display"-mod och de andra två i "mover" och CPU-mod. Minnena adresseras vid drift av fyra klockor från klockkretsarna och data från minnena utgör de nya styrsignalerna. Dessa styrsignaler kopplas via ett par s k "pipelinesteg" för att synkroniseras mot pixelklockan.

Vid systemstart måste minnena laddas med rätt data innan några styrsignaler kan avges, därför blockeras alla adresser från klockgenereringen via en portkrets (se avsnitt systemklocka"). Då data skrivits in i minneskretsarna med hjälp av databussen och CPU:ns adressbuss, släpps klockorna fram och kan adressera minnena.



Programmering av klockregister

Då man vid uppstartsförloppet ska programmera tidstabellen (kretsarna 11H, 11J, 12J och 12L kopplas CPU-adresserna CA4-CA7 till minnena via portkretsen 10M som i sin tur aktiveras av signalen CLOCKS DISABLED. Den signalen skapar i D-vippan 13M/1 och aktiveras då resetsignalen RESI går låg, d v s under systemstart eller reset.

Med de två I/O-strobarna WRML (Write Move Low byte strobe) och WRDL (Write Display Low byte strobe) väljer man nu vilka minnen som ska skrivas då de påverkar WE-ingångarna på respektive minnespar. Dessa signaler påverkar också enable-ingångarna genom att via SET -resp RESET-ingången på J/K-vippan 13J/1 aktivera MSOE (Move State Output Enable) och DSOE (Display State Output Enable).

Efter avslutad programmering deaktiveras WRML resp WRDL och I/O-stroben ENABLE CLOCKS aktiveras så att klockorna från portkretsarna 11M och 12T tillåts adressera minnena och styrsignalerna kan börja arbeta.

Via "pipelinestegen" 10J, 10L och 9L samt D-vipporna i 11L fördröjs och synkroniseras styrsignalerna så att de uppträder i rätta tidslägen.

Klocksignaler

Nedan följer en beskrivning av signalerna samt tidsdiagram, då "mover" resp "display" är aktiva.

RAS	Radadresstrob till grafikminnet är en 250 ns lång puls i "display"-mod och vid "mover"-mod 125 ns. Den går aktivt låg vid inläsning av radadress och är sedan aktiv även under kolumnadressering.
CAS	Kolumnadresstrob till grafikminnet. Då signalen går låg läses kolumnadresser in i minnet från adressmuxarna. I "display"-mod avges fyra kolumnadresstrobar vid utläsning av data, då minnena kan arbeta i sk "nibble mode". Detta innebär att fyra databitar i snabb följd läses till bildskärmen. I "mover"-mod och i CPU-mod sker läsning och skrivning med en bit/gång.

MAM

Muxsignalen (Memeory Adress Mux) kopplar om adressmuxarna till grafikminnet så att radadresser eller kolumnadresser når adressingångarna på minneskretsarna.

DISP REG CLK

Klockar videoregistret vid utläsning av data från grafikminnet. Signalen aktiveras fyra gånger i rad vid laddning av videoregistret men är något fördröjd i förhållande till CAS. I likhet med RAS och CAS buffras och inverteras signalen i 8L/1 innan den används. De fyra klock - pulserna laddar videoregistrets tre "pipelinesteg" och pixelskiftregistret. Den avges också två gånger till under "mover"- eller CPU-sekvens för att flytta data framåt i "pipelinen"

DATA CLK

Används i "mover"-mod för att via NAND-grindarna 12H/2 och 10H/2 klocka registret 6H, som levererar data till "movern" vid utläsning från grafikminnet. Klockan används också för att läsa in data i registren 1V och 2H som innehåller s k "gammalt ord" i "moverns" cirkelskiftare.

MOVE CYK CLK

Aktiv under "mover"-sekvens. Klockan gör via D-vippan 12M/2 så att varannan moveraktivitet mot minnet blir en läsning och varannan skrivning. Se vidare avsnittet "MOVER".

DISP MEM WE

Signal som via OR-grindarna 7J/1, 8J/1 och 8J/" blockerar CPU:ns och "moverns" mjöligheter att skriva i minnet under en "display"-sekvens.

Obs!
Bildskärmsutläsningen har alltid högsta prioritet.

DTACK CLK

Klockar D-vipporna 11A/1, 11A/2 och 11G/2. Vipporna används för att generera kvittenssignalen PACK till CPU:n vid skrivning eller läsning. Är adressområdet det rätta och "movern" ej är aktiv får CPU:n möjlighet att skriva i minnet. Genom att "seriekoppla" 11A/1 och 11A/2 erhålls olika timing vid skrivning resp läsning. Se vidare avnsittet "Grafikminne".

BLANK TEST

BLANK TEST klockar D-vippan 11N/2. I denna kontrolleras om signalen DISP EN från CRTC:en är aktiv. Då bildskärmen har sitt aktiva läge ligger GATE OUT CH REG LOAD från D-vippan 11N hög och möjliggör utskiftning av data till videoskiftregistret. Via J/12-vipporna 13J/1 och 2 samt AND-grindarna 13L/2 och 3 hanteras fördelningen mellan MSOE och DSOE. Då DISP EN är aktiv sker en fördelning av tiden mellan de två enablesignalerna, under det att då DISP EN är inaktiv kommer MSOE att få hela tiden. BLANK TEST är identisk med DTACK CLK men fördröjs två klock - pulser av PIXCLK i 11F.

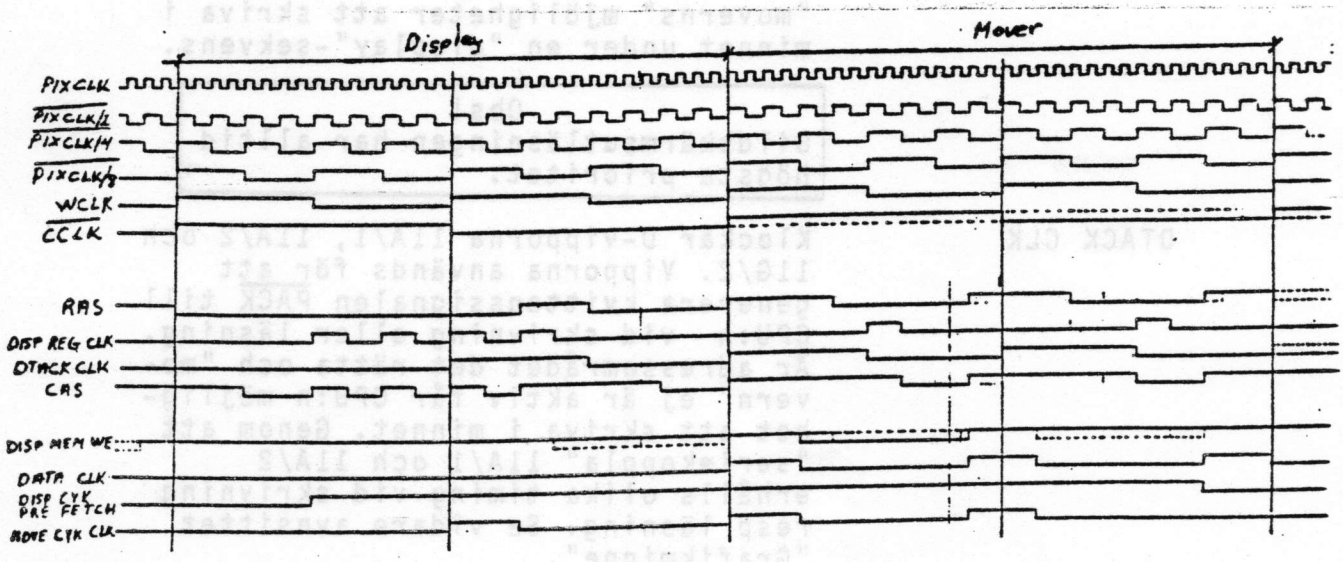
DISP CYC SEL

Används för att ställa in adressmuxarna så att "display"-adresserna kopplas till minnet. Signalen kommer från det första "pipelinesteget" 10J och aktiveras därför först av klocksignalerna. Via NAND-grindarna 10A/1,2 och 4 blockerar signalen CPU:ns och "moverns" möjligheter att adressera minnet under en bildskärmssekvens.

DISP CYK PRE FETCH

Följer DISP CYC SEL men är fördröjd en klockpuls i 11L. Signalen ser via NANDgrinden 8M/1 till att de inre adressmuxarna (fördelning mellan radadress och kolumnadress) aktiveras då CRTC:n adresserar minnet.

Tidsdiagram



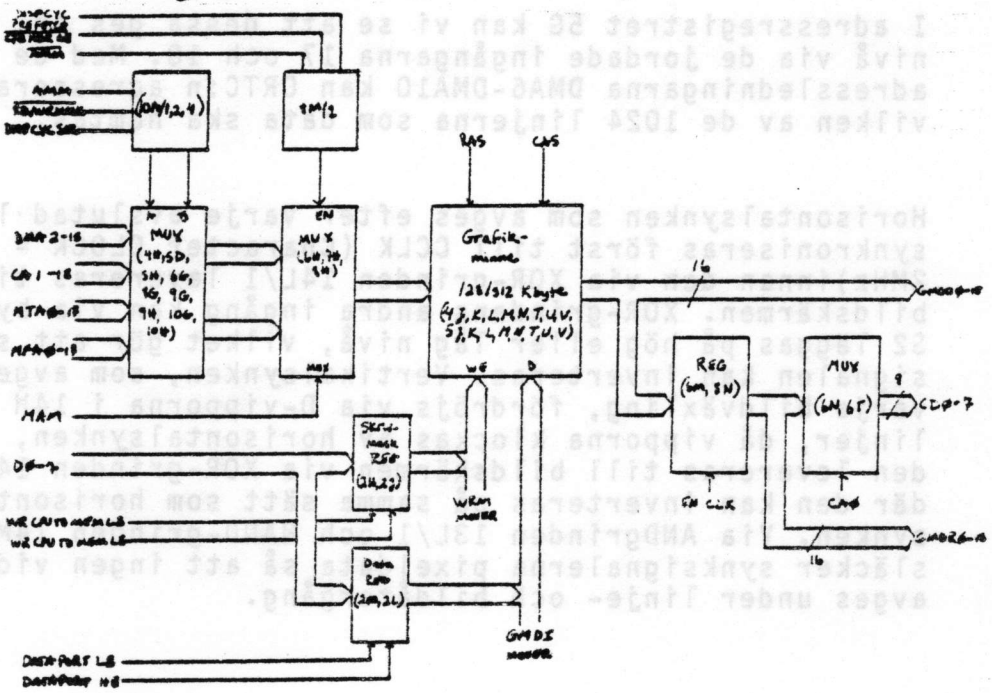
3.2.2 CRTC + Video ut

Idag ställs allt högre krav på arbetsplatsens utformning. En viktig länk i detta utgörs av bildskärmens egenskaper. Det får ej finnas något märkbart flimmer samtidigt som text ej får bli suddig vid rullning och detta innebär att bildrepetitionsfrekvensen måste vara hög. Till detta kommer att upplösningen bör vara hög för att ge god teckenutformning och tydliga bilder vid grafisk presentation. Hög upplösning i kombination med hög bildrepetitionssfrekvens gör att data till bildskärmen måste sändas med mycket hög hastighet.

I ABC 1600 är upplösningen 768 punkter x 1024 linjer vilket gör nästan 800.000 punkter (eg 786832). Med en bildrepetetionsfrekvens på ca 60 Hz gör detta att utläsningen av data måste ske med en hastighet av 64MHz. För att klara av denna hastighet är den del som hanterar pixeldata i seriell form, uppbyggd med snabb logik sk FAST-logik. Den höga hastigheten ställer också krav på att timingen ska vara exakt vilket särskilt gäller videoregistret där data omvandlas från parallell form till seriell.

Enhetens huvudbeståndsdelar är följande:

- CRTC för generering av adresser till grafikminnet samt vertikal- och horisontalsynk.
- Grafikminne för lagring av aktuell bild (behandlas separat)
- Videoregister för mellanlagring och omvandling av parallelldata till seriedata.
- Pixelkontroll för släckning och invertering av pixeldata samt nivå och impedansanpassning till koaxialanslutning.



CRTC

CRT-Controllern 6845E är en intelligent kontrollkrets för bildskärmshantering. Den innehåller ett antal register som kan programmeras med uppgifter om hur minnesadresseringen ska gå till, timing och pulslängd för vertikal och horisontalsynk mm.

Programmering av kretsen sker vid systemstart genom att CPU:n via I/O-adressering aktiverar stroben $\overline{CRT}$ som påverkar CRTC:ns CS-ingång. Med hjälp av adress CA0 och databussen kan nu de enskilda registren skrivas på så sätt att då CA0 är låg anger databussen en adress till det register som ska uppdateras och vid nästa sekvens då CA0 är hög anger databussen det värde som registret ska innehålla.

Genom att upprepa processen kan nu alla registren ges önskade data och därmed CRT-Controllern sin önskade funktion. CRTC:ns enableingång styrs av $\overline{IORQ}$ via grinden 10A/3 och skriv/läs-strob är 6845 R/$\overline{W}$ som motsvarar CPU:ns R/W-signal. Under själva programmeringssekvensen är CRTC:ns möjligheter att adressera minnet avstängda då resetingången hålls aktiva av D-vippan 14 J/1 tills CPU:n aktiverar stroben $\overline{ENDISP}$ vilket inte sker förrän alla videoenhetens register är programmerade.

När CRTC:n aktiveras kommer den att leverera adresserna DMA2-15 till grafikminnet, av vilka DMA2-5 utgör X-adresser, dvs de adresserar orden inom en linje. Som tidigare omtalats, arbetar grafikminnet vid utläsning av data i sk "nibble mode" och detta gör att endast vart fjärde ord inom linjen ska adresseras. Det gör också att de två minst signifikenta adressledningar till grafikminnet ej behöver avges, utan kan läggas till en fast nivå då CRTC:n adresserar minnet.

I adressregistret 5G kan vi se att dessa ges en låg nivå via de jordade ingångarna 17 och 18. Med de tio adressledningarna DMA6-DMA10 kan CRTC:n adressera från vilken av de 1024 linjerna som data ska hämtas.

Horisontalsynken som avges efter varje avslutad linje, synkroniseras först till CCLK (Character CLOCK = 2MHz)innan den via XOR-grinden 14L/1 levereras till bildskärmen. XOR-grindens andra ingång kan via bygeln S2 läggas på hög eller låg nivå, vilket gör att synksignalen kan inverteras. Vertikalsynken, som avges vid varje bildväxling, fördröjs via D-vipporna i 14H fyra linjer, då vipporna klockas av horisontalsynken, innan den levereras till bildskärmen via XOR-grinden 14L/2 där den kan inverteras på samma sätt som horisontal - synken. Via ANDgrinden 13L/1 och NAND-grinden 14M/4 släcker synksignalerna pixeldata så att ingen video avges under linje- och bildåtergång.

CRTC:n sänder också ut signalen DISPEN (DISPlay ENable) som talar om att när den adresserar inom ett aktivt område av bildskärmen. Genom att utnyttja denna signal till styrningen av sekvenskontrollen, får "mover" och CPU längre tid att arbeta på när CRTC:n inte adresserar bildminnet.

Videoregister

Videoregistret ska i snabb följd kunna ta hand om och lagra 64 pixels (4 ord) från grafikminnet och överföra dessa till seriell form. För detta ändamål finns tre "pipelinesteg" bestående av registren 7T, 7U, 7V, 9M, 9N, 9T och skiftregistren 9U, 9V, 10U och 10V. De tre "pipelinestegen" klockas av signalen DISPREG CLK som har sitt ursprpung i sekvenskontrollenheten. Denna signal avges fyra gånger i följd, i takt med att nytt data finns tillgängligt från grafikminnet men avges också vid två senare tillfällen under mover eller CPU-sekvensen då data flyttas framåt i "pipelinen". Med hjälp av SHIFT LOAD PULSE läses data från sista stegets utgångar in i skiftregistren. SHIFT LOAD PULSE bildas vid klockgenereringen och ger en kort puls för var sextonde pixelklockpuls.

I AND-grinden 10N/1 grindas SHIFT LOAD PULS med signalen GATE OUT CH REG LOAD som vid låg nivå stänger grinden så att skiftregistren inte kan laddas. Detta gör man utanför den aktiva bildytan för att få ett definitivt läge på utdata trots att inget relevant indata finns tillgängligt. När inte skiftregistren laddas med nytt data kommer den seriella ingången på 10V att få stå med indata dvs signalen FRAME POL. Den kommer från flaggregistret 13H som kan sättas av CPU:n, och ger att ramen runt den aktiva bildytan kan påverkas så att den blir svart eller vit.

Videosignal

Data ut från skriftregistret levereras av 9V och signalen kallas PIFSR (Pixel From Shift Register). I XOR-grinden 14L/4 grindas den med signalen PIX POL från flaggregistret som vid hög nivå inverterar alla pixels innan de levereras till bildskärmen. Från XOR-grindens utgång kopplas pixelinformationen till NAND-grinden 14M/4 där en låg nivå på den andra ingången gör att signalen blockeras. I AND-grinden 13L/1 samlas de signaler som ska blockera utskrivning av data till bildskärmen. Till dessa hör vertikal och horisontalsynk för att återgångarna ska vara osynliga.

Vid uppstart (innan ENDISP aktiverats) hålls set-ingången på 14D/1 låg vilket genom pin 6 ger en blockering av pixelsignalen. Då uppstartsförloppet är över kan CPU:n via signalen BLANKFLAG från flaggregistret också släcka signalen genom att ändra dataingången på 14D/1 och på så sätt få vippan att slå om.

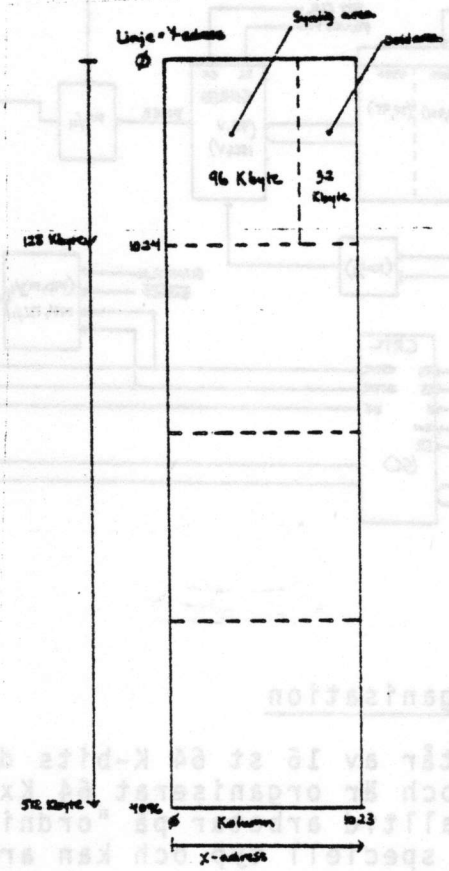
Från NAND-grinden 14M/4 kopplas signalen via 14N/2 till videoutgångens koaxialkontakt. 14N/2 har "Open Collek - torutgång" som får sin spänningsmatning från transistorn TR1. Efter spänningens nivåanpassning i TR1 går matningen via ett 47 ohms motstånd som ger utgången c:a 50 ohms impedans.

Studera avsnitt "Grafikminne" och "I/O-hantering" som ger en del kompletterande upplysningar samt beskrivning över 6845E under kap. 5.

3.2.3 Grafikminne

Grafikminnets uppgift är att lagra den bildinformation som ska visas på bildskärmen. Minnet är pixelorienterat dvs man lagrar informationen separat för varje punkt (pixel), till skillnad från exv textminnen där endast en kod lagras för att beskriva mönstret inom en viss given bildyta. Detta gör att minnesbehovet ökas men ger istället att text och bilder kan blandas valfritt och att olika textstilar och symboler kan skapas och visas samtidigt på bildskärmen.

Minnesstorleken är 128 Kbyte och innehåller information om en (1) bild men kan vid behov byggas ut till 512 Kbyte och kan då lagra fyra (4) bilder. Av dessa 128 Kbyte utnyttjas 96 Kbyte för den pixelinformation som ska visas (synlig area) under det att resterande 32 Kbyte kan användas som symbolbibliotek och för att lagra tecken (dold area).

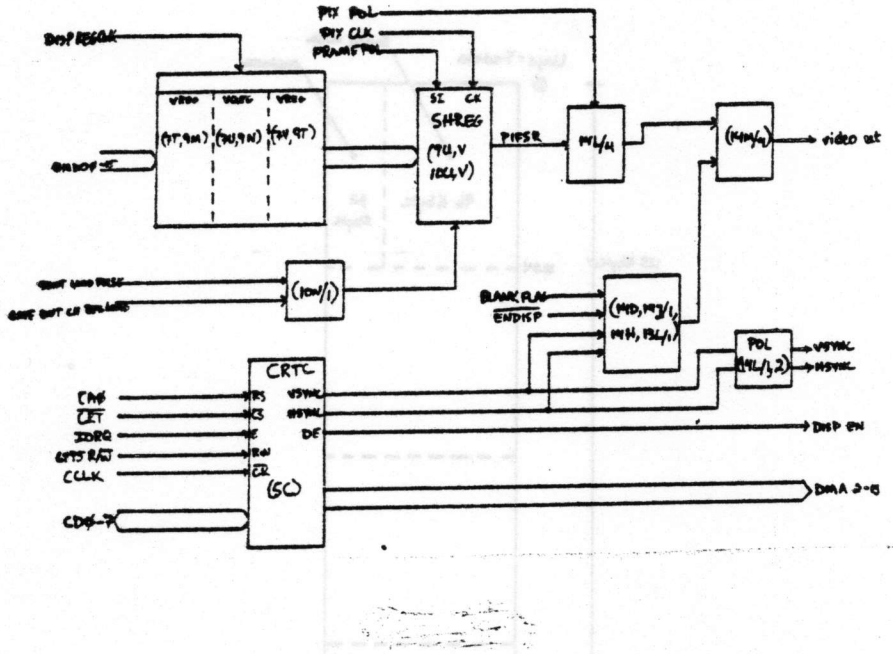


Enheter som använder grafikminnet

Det är flera enheter som måste komma åt att arbeta i grafikminnet. "CRT-Controllern" ska kunna adressera minnet då data skiftas ut till videoregistren och vidare till bildskärmen. CPU:n ska uppdatera bilder och text men måste också komma åt att läsa den information som finns i grafikminnet.

"Movern" måste både kunna skriva och läsa ur minnet för att flytta ytor mellan dess olika delar, som exv att hämta ett tecken från den dolda arean och flytta in den i den synliga arean vid inmatning av text. Allt detta gör att man måste ha adressmuxar och dataportar som kan hanteras av resp. enhet, men också en viss inbördes prioritet som talar om när de olika enheterna får tillgång till minnet.

Högsta prioritet har utläsning av data till bildskärmen som alltid måste ske rätt ögonblick då en viss tidsförskjutning här skulle ge störningar i bilden. Därnäst följer "mover" och CPU, som får dela på den återstående tiden. Då utskiftning av data går snabbt blir det dels tid över mellan varje utläsning men också vid horisontal och vertikalåtergång då utskiftning av data stoppas och "mover" och CPU ensamma har tillgång till grafikminnet.

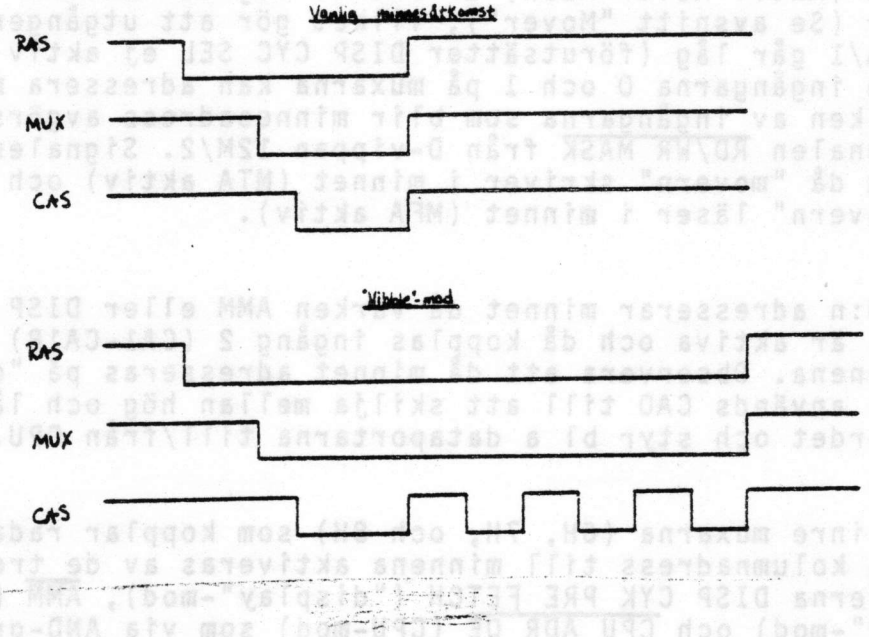


Grafikminnets organisation

Grafikminnet består av 16 st 64 K-bits dynamiska minnen (vid 128 Kbyte) och är organiserat 64 Kx16 bitar, vilket gör att man alltid arbetar på "ordnivå" (16 bit). Minnena är av en speciell typ och kan arbeta i sk "Nibble mode", vilket innebär att fyra databitar kan läsas eller skrivas i en följd med endast en adressering. Detta tillgår så att radadress och kolumnadress på vanligt sätt kopplas till minnenas adressingångar under det at RAS resp CAS avges.

Genom att hålla RAS aktiv och samtidigt ge ytterligare tre kolumnadresstrobar, adresserar minnet själv tre nya kolumnadresser, vilket gör att totalt fyra bitar kan skrivas eller läsas från minnet i snabb följd.

Detta sätt att arbeta utnyttjas då data läses till videoregistret och gör att 16x4 pixels snabbt kan flyttas från minnet till registret. "Mover" och CPU får nu längre tid för skrivning/läsning i minnet.



Någon speciell "refreshlogik" behövs ej till grafikminnet då utläsningen till bildskärmen sker på alla rad - adresser med en viss
periodisitet som gott och väl räcker till för "refresh" av minnena.

Adressering av grafikminnet

I adressmuxarna samlas minnesadresserna från CPU, "mover" och CRTC. Muxarna är av typen 4-1 och kopplar adresserna DMA2-15 (Display Memory Adress), CA1-18 (CPU Adress), MTA0-17 (Mover To Adress) och MFA0-17 (Mover From Adress) till minnena.

Mellan adressmuxarna och minnena sitter ytterligare en nivå muxar (2-1), nämligen de som kopplar om mellan radadress och kolumnadress. De nio yttre muxarna (4H, 5D, 5H, 6G, 8G, 9G, 9H, 10G, 10H) väljer ingång med hjälp av selektingångarna A och B.

Dessa ingångar styrs i sin tur av NAND-grindarna 10A/1,2 & 4 vilka styrs av signalerna AMM, RD/WR MASK och DISP CYC SEL. DISP CYC SEL kommer från register 10J i sekvenskontrollenheten och är aktiv då data ska läsas ut till videoregistret. Då denna signal har låg nivå sätts utgångarna på 10A1 och 2 båda till hög nivå, vilket gör att ingångarna 3 ("displayadresserna" DMA2-15) på muxarna kopplas till utgångarna.

(Observera att de två minst segnifikanta adressledningar ej behövs här pga minnenas "nibble mode", samt att adressledning DMA 15 är den högsta vilket gör att endast den lägre av de fyra minnesdelarna kan visas på bildskärmen vid utbyggt minne, 512 Kbyte).

AMM (Aktiv Mover Mask) är aktivt hög då "movern" arbetar (Se avsnitt "Mover"), vilket gör att utgången på 10A/1 går låg (förutsätter DISP CYC SEL ej aktiv = hög) och ingångarna 0 och 1 på muxarna kan adressera minnet. Vilken av ingångarna som blir minnesadress avgörs av signalen RD/WR MASK från D-vippan 12M/2. Signalen är hög då "movern" skriver i minnet (MTA aktiv) och låg då "movern" läser i minnet (MFA aktiv).

CPU:n adresserar minnet då varken AMM eller DISP CYC SEL är aktiva och då kopplas ingång 2 (CA1-CA18) till minnena. Observera att då minnet adresseras på "ordnivå" används CA0 till att skilja mellan hög och låg byte i ordet och styr bl a dataportarna till/från CPU.

De inre muxarna (6H, 7H, och 8H) som kopplar radadress och kolumnadress till minnena aktiveras av de tre signalerna DISP CYK PRE FETCH ("display"-mod), AMM ("mover"-mod) och CPU ADR OE (CPU-mod) som via AND-grinden 8M/1 påverkar muxens enable-ingång. För att muxarna ska aktiveras krävs att någon av dessa signaler är aktiva.

DISP CYK PRE FETCH är aktivt låg och kommer från D-vippan 11L i sekvenskontrollenheten och är aktiv då data läses till videoregistret. AMM är aktivt låg och kommer från 6D i "mover"-kontrollenheten och är aktiv under en "mover"-sekvens. (Se under "Mover"). CPU ADR OE är aktivt låg då CPU:n adresserar minnet. Signalen kommer från D-vippan 11G/2 och aktiveras om rätt adressområde aktiveras och då DTACK CLK från sekvenskontrollenheten går hög. På detta sätt kan muxarna aktiveras under alla tre sekvenserna.

Med hjälp av MAM-signalen från sekvenskontrollenheten sker sedan omkoppling mellan radadress och kolumnadress så att då MAM har hög nivå läses radadressen och vid låg nivå, kolumnadressen.

Datastrobar

När nu adresserna har påförts minnena och RAS/CAS från sekvenskontrollenheten är det dags att se hur data förs till eller från minnena.

Varje minneskrets har en dataingång (GMDI=Grafic Memory Data Input) där en bit åt gången kan skrivas till minnet och en datautgång (GMDO=Grafic Memory Data Output), där likaså en bit levereras då man läser ut minnet.

Minnena har också en ingång som indikerar om skrivning till eller läsning från minnet gäller, $\overline{WE}$ (Write Enable).

Då data läses ur minnet till videoregistret ligger $\overline{WE}$ hög och data läggs på utgångarna GMDO-15, varifrån de kan läsas till videoregistret med hjälp av klockan DISPREG CLK.

"Movern" hanterar grafikminnet

När "movern" läser data ur minnet klockas dessa in i registren 6M och 8N av klockan EN CLK. Data från dessa register kallas nu GMDRO-15 och bildar indata till "moverns" cirkelskiftare. Signalen EN CLK kommer från AND-grinden 10N/2 där det krävs att ingång 5 är hög för att DATA CLK på ingång 4 ska komma genom grinden. Detta uppfylls då någon av ingångarna till NAND-grinden 12H/2 har låg nivå.

Den ena signalen AMM är alltid hög då "movern" är aktiv under det att den andra DATA MASK växlar under en "mover"-sekvens så att den är låg då "movern" läser ett ord men hög då "movern" skriver in data igen. Detta gör att data klockas in i registret då "movern" läser men också då CPU:n läser för då är även AMM-signalen låg.

CPU:n kan dock inte läsa in alla 16 bitarna i ordet samtidigt utan tar hjälp av muxarna 6N och 6T för att dela upp ordet i två bytes. Muxarna aktiveras av signalen $\overline{MEM\ DATA\ OE}$ från OR-grinden 11B/4 och för att signalen ska få låg nivå krävs att rätt adressområde aktiveras samt att läsning gäller. Muxsignalen utgör den minst signifikant adressbiten CA0, som här liksom i andra sammanhang får skilja lågdel och högdel av ordet.

CPU:n hanterar grafikminnet

Då CPU:n skriver eller läser i minnet måste den ha kvittenssignal tillbaka. När videoenheten aktiveras kommer kvittenssignalen $\overline{PACK}$ (Picture ACKnowledge) från AND-NOR-grinden 7B/1. För att denna signal ska avges måste ett antal villkor vara uppfyllda bl a att rätt adressområde aktiverats. Detta kontrolleras i grindarna 7J/3, 9B/3 och 8B/1 som avkodar adresserna CA17-CA20. Med hjälp av byglarna S3 och S4 kan det avkodade området ändras då utbyggnad av grafikminnet sker.

Då adressen ligger inom grafikminnesområdet ges 8B/1 en låg nivå på utgången som kopplas till OR-grinden 11B/1 och NOR-grinden 9B/1. De båda grindarnas andra ingångar får signal från D-vippan 11G/1 signalerna NODAS (No Delayed Adress Strobe) och DAS (Delayed Adress Strobe) gör att utsignalen ligger hög en kort stund efter det att adresstroben avgivits men går sedan låg då DAS blir aktiv.

Detta gör att utgången på 9B/1 är låg vilket i sin tur innebär att utgången på AND-grinden 11C/4 är låg och de tre vippornas (11A/1o2, 11G/2) SET-ingångar är aktiva. I detta läge kan varken enable-signaler eller kvittenssignaler avges.

Då DAS går låg slår utgångarna på 11B/1 och 9B/1 om vilket gör att SET-ingångarna på vipporna ej längre aktiveras. Vipporna kan nu klockas av DTACK CLK från sekvenskontrollenheten så att 11A/1 och 11G/2 slår om. Detta gör att signalen $\overline{\text{CPU ADR OE}}$ aktiveras dvs de inre adressmuxarna aktiveras. Gäller skrivning kommer stroben $\overline{\text{WR FROM CPU}}$ att avges samt kvittenssignalen PACK.

Om läsning gäller kommer det att dröja till nästa klockning av DTACK CLK, som får 11A/2 att slå om, före PACK till CPU:n avges. Vid läsning aktiveras också signalen $\overline{\text{MEM DATA OE}}$ från ORgrinden 11B/4 som öppnar dataportarna 6H och 6T för läsning. En förutsättning för kvittenssignal är också att det inte är "movern" som aktiverar minnet vilket kontrolleras med hjälp av signalen $\overline{\text{AMM}}$ till 7B/1.

Skrivning i grafikminnet kan ske från "movern" och CPU:n genom att respektive indataport aktiveras samt att $\overline{\text{WE}}$-ingången på minnena läses på låg nivå. I vissa lägen kan det vara nödvändigt att inte skriva i ordets alla 16 bitar utan man vill låta vissa vara opåverkade. Detta kan göras genom att skapa en skrivmask som endast aktiverar de utvalda minnenkretsarnas WE-ingångar.

Dataportarna som aktiveras från CPU:n är registren 2M och 2L, vars enableingångar styrs av signalerna DATAPORT LB och DATAPORT HB. Dessa signaler kommer fån Mux 9J som dessutom skapar signalerna WRPORT LB och WRPORT LB som klockar skrivmaskregistren 2H och 2J.

Då CPU:n vill sätta upp en skrivmask aktiverar den genom I/O-adressering stroben WRMASKSTROBE LB eller HB från 9F beroende på vilket register som ska skrivas. Står muxens selektingång i REPLACE (=låg, vilket den normalt gör) kommer WRPORTLB eller HB att aktiveras och klocka resp skrivmaskregister. För att registren ska kunna skrivas måste också enableingångarna aktiveras. Detta sker med hjälp av WR CPU TO MEM LB resp HB som aktiveras av WR FROM CPU och CAO genom OR-grinden 8J.

När skrivmasken är uppsatt skrivs data till minnena genom att öppna dataporten och klocka in data. Enable till portarna ges då AMM är låg, dvs då "movern" är inaktiv. Data klockas med hjälp av strobarna DATAPORT LB resp HB från mux 9 J.

Strobarna aktiveras då BDS (Buffrad datastrob) samt signalen WR FROM CPU som via OR-grinden 11 B/2 öppnar 9B/2 o 4 så att CAO kan bestämma om DATAPORT LB eller HB ska aktiveras. Data kan nu skrivas till minnena med hjälp av dataportarnas innehåll och skrivmasken.

Ett speciellt läge inträffar de selektingången på mux 9J sätts till hög nivå. Då skriver man istället in ett mönster i dataportregistren ex FFH ("SET-mod") eller OOH ("RESET-mod") och aktiverar skrivmasken vid skrivning i bildminnet.

Även "movern" kan producera en skrivmask, vilket oftast är nödvändigt för första och sista ordet på varje linje samt om den yta som ska flyttas är så liten att den ej upptar ett helt ord. Skrivmasken genereras i 1G och 1J, som programmerats med en tabell som talar om hur skrivmasken ska se ut under olika förutsättningar.

Data från "movern" till grafikminnet kommer från PAL-kretsarna 1M,1N, 1T och 2T som i detta läge fungerar som 2-1-muxar med separata muxsignaler så att "gammalt" och "nytt ord" kombineras beroende på skiftfaktorn.

Läs mer under avsnittet "Mover".

3.2.4 "Mover"

"Moverns" egenskaper

Vid pixelorienterade bildminnen av den typ som finns i ABC 1600 måste en mängd information lagras om bildens innehåll i jämförelse med teckenorienterade bildminnen. Det gör att varje gång en bild ska uppdateras måste en hel del data transporteras, vilket tar lång tid i anspråk. Lång uppdateringstid är irriterande i synnerhet vid inmatning av text då man förväntar sig att tecknet omedelbart ska visa sig.

För att komma runt problemet har ABC 1600 försetts med en sk "mover". Med hjälp av den kan man inte bara flytta tecken till rätt position på bildskärmen utan vilken godtycklig rektangulär yta som helst från en storlek av en (1) pixel till hela bildskärmsinnehållet. Movern kan inte arbeta i datorns primärminne varför hela teckenuppsättningen vid start skrivs in i grafikminnets dolda area (den del som ej visas på bildskärmen), varifrån tecken sedan kan hämtas till godtycklig position i den synliga delen.

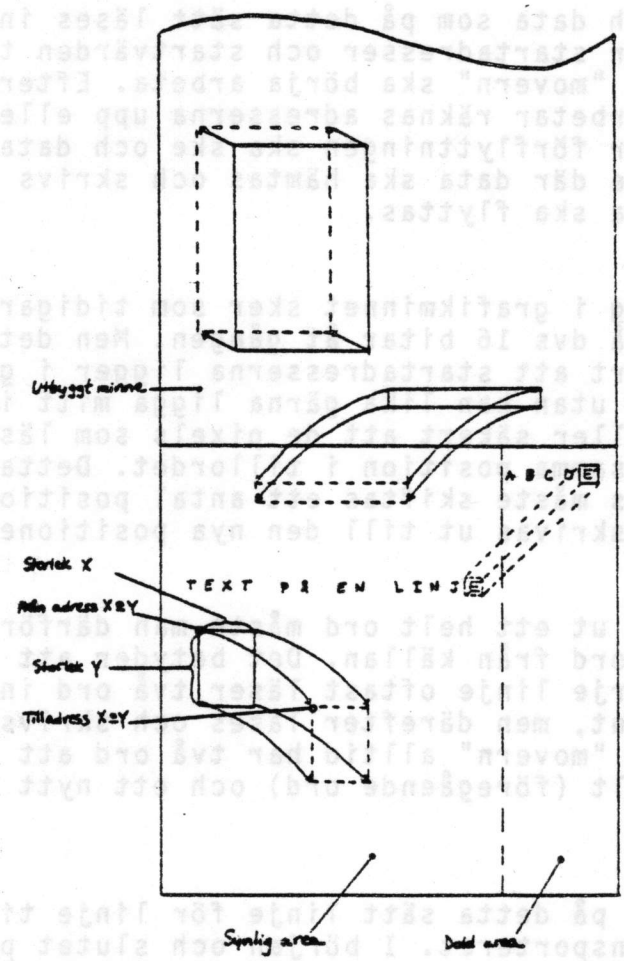
"Movern" kan således arbeta inom hela grafikminnesarean och ser ingen skillnad mellan dold och synlig area. Även då minnet är utbyggt till 512 Kbyte kan den arbeta inom hela detta område, vilket är nödvändigt då man vill lagra flera bilder som snabbt ska kunna skiftas.

Adresseringen från CRTC:n sker alltid i samma area vilket gör att då en ny bild ska visas måste den först flyttas till detta område innan den kan visas. Men även om man inte är i behov av flera bilder kan det ibland vara nödvändigt med utbyggt minne, då det kan användas som symbolbibliotek eller liknande vid exv CAD-tillämpningar.

Tidigare visades att utskiftning av pixels till bildskärmen inte tar hela den tillgängliga tiden i anspråk som finns för minnesåtkomst. Under visningstid delar "mover" och CRTC på tillgängligtid (fördelas av sekvenskontrollen) men då utskiftning av data till skärmen ej sker (under strålåtergång) får "moverns" hela tiden för sig själv. Fördelningen mellan visningstid och släckt tid är 74% tänd och 26% släckt.

Då "movern" flyttar 24 Mpixels/s under visningstid och 32 Mpixels/s under släcktiden gör detta ett genomsnitt av 26.1 Mpixels/s. Tar vi hänsyn till den synliga delens area (strax under 0.8 Mpixel) gör det att bilden kan flyttas över 33 ggr/s. Då "movern" flyttar grafikminnesdata ord för ord går det naturligtvis ännu mycket snabbare att flytta en mindre del av innehållet.

Den yta som "movern" ska flytta måste vara rektangulär men den kan hämtas från vilken position som helst och levereras till godtycklig plats inom ramen för minnesmatrisen. Det gör att ytan kan flyttas uppåt, nedåt, åt höger eller vänster och till- och frånyta kan överlappa varandra utan att data förstörs eller skrivs över.



Då "movern" ska flytta ett block får den uppgifter från CPU:n om frånposition, tillposition samt blocket storlek i X och Y-led. Dessa uppgifter laddar CPU:n i några av "moverns" register med hjälp av I/O-hantering och då sista uppgiften är levererad startar "movern" automatiskt flyttningen. Det är normalt sex register som ska skrivas med tolv bytes data.

Dessa är:

Storlek	X-led	(2 bytes)
Storlek	Y-led	"
Tilladress	X-led	"
Tilladress	Y-led	"
Frånadress	X-led	"
Frånadress	Y-led	"

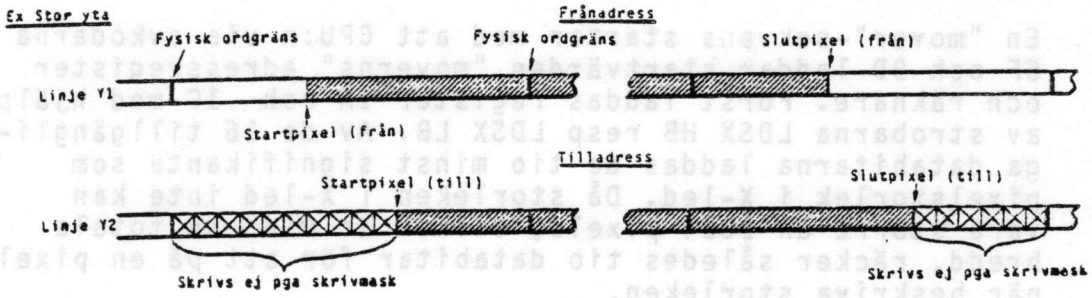
Ett specialfall är då "movern" flyttar text från den dolda arean till den rad där inmatningen för tillfället sker. I detta fall är storleken i X och Y-led given samt även tilladresserna då de utgörs av nästa teckenposition, vilket gör att endast frånadresserna måste anges dvs 4 bytes. Vid byte av rad anges dock även tilladresserna för första teckenpositionen vilket gör att CPU:n i detta fall skickar 8 bytes.

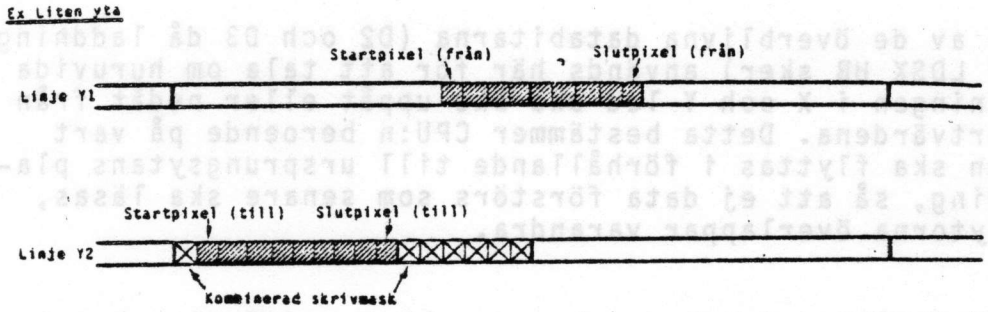
De adresser och data som på detta sätt läses in i registren utgör startadresser och startvärden till de positioner där "movern" ska börja arbeta. Efter hand som "movern" arbetar räknas adresserna upp eller ner beroende på hur förflyttningen ska ske och data läses från det ställe där data ska hämtas och skrivs till det ställe dit data ska flyttas.

All adressering i grafikminnet sker som tidigare omtalats på ordnivå dvs 16 bitar åt gången. Men det är ju inte alls säkert att startadresserna ligger i gränser mellan två ord utan kan lika gärna ligga mitt i ordet. Det är inte heller säkert att de pixels som läses i frånordet har samma position i tillordet. Detta gör att data troligtvis måste skiftas ett antal positioner innan det kan skrivas ut till den nya positionen.

För att skriva ut ett helt ord måste man därför oftast kombinera två ord från källan. Det betyder att "movern" i början på varje linje oftast läser två ord innan det första skrivs ut, men därefter läses och skrivs ett ord i taget så att "movern" alltid har två ord att arbeta med, ett gammalt (föregående ord) och ett nytt (senast inlästa ord).

Movern arbetar på detta sätt linje för linje tills hela blocket är transporterat. I början och slutet på varje linje är det viktigt att inte förstöra informationen utanför det område som data ska flyttas till. Här använder sig "mover"n av möjligheten att lägga en skrivmask (dvs man aktiverar inte WE-ingångarna på alla minneskretsar) över det ord som ska skrivas ut. De skrivmasker som är aktuella för "movern" finns inskrivna i en PROM-tabell som adresseras av de fyra minst signifikanta bitarna i X-till-adressen och X-storleken berättar vilken pixelposition i ordet som blockets början resp slut har.





Moverns huvudblock är följande:

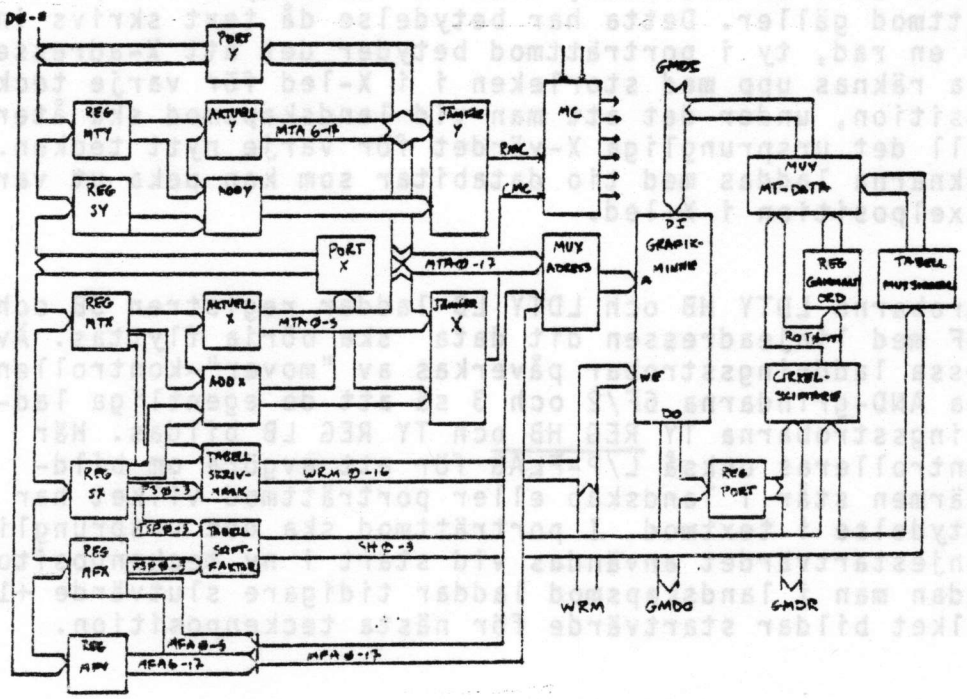
Adressräknare från-adress

Adressräknare till-adress

Cirkelskiftare

"Mixer"

"Mover"-kontroll



Laddning av "mover"-register

En "mover"-sekvens startar med att CPU:n via avkodarna 5F och 9D laddar startvärden "moverns" adressregister och räknare. Först laddas register 1A och 1C med hjälp av strobarna LDSX HB resp LDSX LB. Av de 16 tillgängliga databitarna laddas de tio minst signifikanta som pixelstorlek i X-led. Då storleken i X-led inte kan vara större än 1024 pixels, vilket är minnets totala bredd, räcker således tio databitar för att på en pixel när beskriva storleken.

Två av de överblivna databitarna (D2 och D3 då laddning med LDSX HB sker) används här för att tala om huruvida räkningen i X och Y-led ska ske uppåt eller nedåt från startvärdena. Detta bestämmer CPU:n beroende på vart ytan ska flyttas i förhållande till ursprungsytans placering, så att ej data förstörs som senare ska läsas, om ytorna överlappar varandra.

När storleken i X-led och upp- eller nedräkning bestämts laddas storleken i Y-led med strobarna LDSY HB och LDSY LB i registren 11D resp 13 F. Här används 12 av de 16 databitarna då storleken i Y-led kan vara upp till 4096 linjer vid utbyggt minne (normalt 1024 linjer vid 128 Kbyte minne). Därefter laddas räknarna 2C, 2F och 3C med strobarna LDTX LB resp LDTX HB. Dessa strobar används inte direkt utan grindas 7C/2 oh 3 till signalerna TX REG LB resp TX REG HB, då X-räknarna inte bara ska laddas med ursprungsdata en gång, utan för varje ny linje.

Detta styrs av "mover"-kontrollenheten som måste kontrollera när det är dags för en ny linje, när flyttningen av data är avslutad, samt om landskap eller porträttmod gäller. Detta har betydelse då text skrivs in på en rad, ty i porträttmod betyder det att X-adressen ska räknas upp med storleken i i X-led för varje tecken position, under det att man vid landskapsmod ska återgå till det ursprungliga X-värdet för varje nytt tecken. Räknarna laddas med tio databitar som kan peka ut varje pixelposition i X-led.

Strobarna LDTY HB och LDTY LB laddar registren 6B och 14F med linjeadressen dit data ska börja flyttas. Även dessa laddningsstrobar påverkas av "mover"-kontrollen via AND-grindarna 6F/2 och 3 så att de egentliga laddningsstrobarna TY REG HB och TY REG LB bildas. Här kontrolleras också L/P-FLAG för att avgöra om bildskärmen står i landskap eller porträttmod vilket har betydelse i textmod. I porträttmod ska det ursprungliga linjestartvärdet användas vid start i ny teckenpositon, medan man i landskapsmod laddar tidigare slutvärde +1, vilket bildar startvärde för nästa teckenposition.

För att veta när linjen resp hela flyttningen är klar, kontrolleras också CMC och RMC från jämförarna 3D och 10 B. Startvärdesregistren innehåller 12 bitar för att kunna adressera totalt 4096 linjer.

Alla data om storlek och tilladresser finns nu lagrade. I textmod behöver dessa register ej skrivas av CPU:n då de uppräknas automatiskt till nästa teckenposition, förutom den första positionen på varje linje, då även till-adresserna måste anges. Det som däremot alltid måste anges är frånadresserna. Med hjälp av strobarna LDFX HB och LDFX LB kan registren 4D och 3G laddas med startpositionen i X-led, varifrån data ska hämtas. För varje ny linje som flyttas, laddas startvärdet från dessa register på nytt.

De sista uppgifterna som CPU:n sänder till "movern" är från-data i Y-led. De båda strobarna LDFY HB och LDFY LB grindas via AND - grindarna 9C/3 och 4 så att klockstrobarna CLK Y FROM HB och CLK Y FROM LB bildas. Dessa klocksstrobar får påverka räknarna 10F, 11F och 12F och tillsammans med signalen LDY ladda dessa med startlinjen. Då "movern" arbetar aktiveras klockstrobarna så att linje för linje kan adresseras.

LD FY LB klockar också D-vippan 8F/1 som i sin tur via 6D aktiverar signalen AMM (Active Mover Mode). AMM-signalen kommer nu att vara aktiv under hela förflyttningen och släpper inte förrän hela förflyttningen är klar.

"Mover"-adressering

Registren är nu laddade med startvärden och "movern" har satts i sitt aktiva läge och det är dags att titta på hur minnet adresseras av "movern".

Vi börjar med "mover"-frånadresserna (MFA 0-17) där MFA0-5 utgörs av X-adressen dvs adressering inom en linje och MFA6-17 för adressering av aktuell linje. I register 3G och 4D finns redan startvärdet i X-led lagrat och med hjälp av signalen LDX, laddas detta värde in i räknarna 3F och 4F så att de innehåller adressen till det första ordet som ska läsas.
Räknarna kommer sedan under hela "mover"-sekvensen att peka ut aktuell frånadress.

De fyra minst signifikanta bitarna (FSP 0-3) talar om från vilken pixelposition i ordet som flyttningen ska börja. Dessa används sedan för att få fram skiftfaktorn (talar om positionsförhållandet mellan till och frånadress inom ordet). MFA0-MFA5 räknas sedan upp eller ned beroende på hur data skall förflyttas.

Då sista ordet av de som ska flyttas har bearbetats, laddas startvärdet in från registren igen och "movern" börjar arbeta på nästa linje.

Ibland kan det vara nödvändigt att låsa räknarna under en viss tid. Det gör man bl a under den tid då laddning av räknarna sker med hjälp av HOLD XY, men kan också användas då man vill kopiera ett mönster i ett ord till en större yta. Då låser man Xoch Y-positionerna på frånräknarna under det att tillräknarna får fortsätta att arbeta. Detta används exv då hela bildskärmen rensas från sitt innehåll. Signalerna som låser räknarna heter HOLD FX, HOLD FY och kommer från flaggregistret som aktiveras av CPU:n.

I Y-led lämnar de tre räknarna 10F, 11F och 12F från-adresserna MFA6-MFA17 direkt till adressmuxarna. De räknas upp eller ner beroende på nivån av U/$\overline{D}$ Y och aktiveras av $\overline{CMC}$ som talar om att sista X-positionen som ska flyttas är uppnådd. Med hjälp av HOLD FY kan räknarna låsas så att samma linje läses hela tiden. Oavsett om räknarna hålls fast eller räknas, pekar de alltid ut aktuell linje som data ska läsas från.

Att beräkna tilladresserna är något mer kompliserat då man i textmod måste koppla ihop in och utgångar för att ladda slutvärdet från förra förflyttningen som startvärde för nästa. Här sker också all kontroll av om slutvärdet uppnåtts så att "mover"-sekvensen ska avslutas. I stora drag kan man säga att startvärdet laddas i ett par räknare som innehåller aktuell adress dvs den adress som just då ska skrivas. Denna adress jämförs med det adderade värdet av startadress och storlek vilket ger oss slutvärdet. Är adresserna lika betyder det att den adress som nu adresserar minnet är det sista.

Från jämförarna ges nu signalerna CMC, från X-adresserna som talar om att nästa linje ska adresseras samt RMC från Y-adresserna som talar om att sista linjen nu adresseras. När båda signalerna är aktiva avslutas "mover"-sekvensen då sista ordet på sista linjen adresseras.

De fyra minst signifikanta adresserna från startvärdet TSP0-3 anger pixelpositionen inom ordet och används för att bilda skiftfaktorn tillsammans med FSP0-3. TSP0-3 ger tillsammans med U/D X-signalen indata för beräkning av skrivmasken vid utskrift av det första ordet på varje linje och tillsammans med de fyra minst signifikanta bitarna ur storleksregistret PS0-3 indata för skrivmasken till det sista ordet på varje linje.

Tidigare visades det att startvärdet i X-led laddas i räknarna 2C, 2F och 3C med hjälp av LD X REG och klockpulserna TX REG LB och TX REG HB. Med hjälp av signalen LD X och klockan TX FX CLK laddas räknarna 2E och 3E med det tidigare inlästa startvärdet. Dessa räknare aktiveras en gång för varje ord och innehåller på detta sätt "aktuellt ord" dvs det ord som för tillfället adresseras av "movern".

Med signalen U/D X kan de fås att räkna uppåt eller nedåt beroende på flyttningriktningen och med HOLD TX WORD används i första positionen på varje linje för att "movern" ska ha två ord att arbeta med innan något skrivs.

HOLD XY används i tidsintervallet mellan det att registren laddats från CPU:n, tills dess att "mover"-moden blir aktiv genom AMM. Utsignalen från räknarna utgörs av MTA0-5 som ger X-adressen till det ord som för närvarande adresseras inom linjen. I adderarna 1B, 1E och 2D adderas start-adressen med storleken så att slutadressen erhålls. Denna slutadress jämförs med aktuellt ord i jämföraren 3D, vars utsignal CMC blir aktiv då sista ordet adresseras på linjen.

Från adderarna kan data läsas genom portkretsarna 1D/1 o 2 samt 2B/1 till den 12 bitar breda databussen som förbinder "mover"registren. Att bussens bredd är 12 bitar beror på att den högsta adress som "moverns" olika delar använder är 12 adressledningar. Från videoenhetens interna databuss kan data från adresserna läsas in som startadresser då man arbetar i textmod och har bildskärmen i porträttläge. För att öppna utportarna måste signalen X BUFF OE från "mover"-kontrollen aktiveras.

Principen för adressbehandling i Y-led är liknande den ovan beskrivna, men levererar linjeadresserna MTA6-17. Här har adressen till startlinjen laddats i registren 6B och 14 F. Med hjälp av TY CLK och LDY laddas denna till räknarna 6C, 13B och 14C som innehåller aktuell linjeadress. TY CLK kommer att räkna upp aktuell linje då CMC är aktiv dvs då X-adressen har nått sin sista position på nuvarande linje. Linjeräknaren kan också räknas upp eller ner med U/D Y beroende på förflyttningsriktningen eller stoppas med hjälp av HOLD XY innan "mover"-moden aktiverats. Då bildskärmen är i landskapsläge kan utdata från dessa register bilda startadresser för nästa tecken som X-adresserna i porträttmod. För att portarna 6A/1, 13C/1 och 2 ska öppna sig, krävs att signalen Y BUFF OE från "mover"-kontrollen aktiveras.

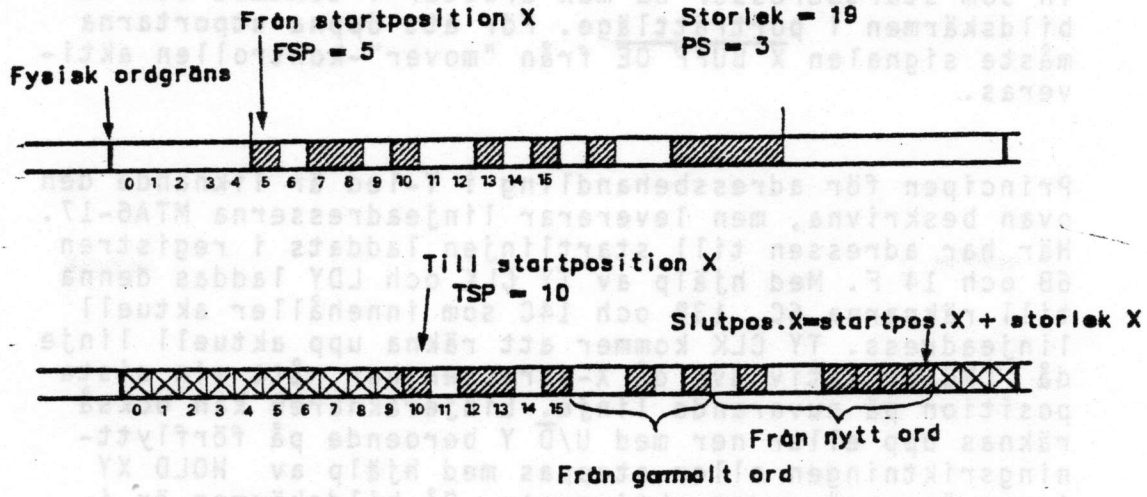
I adderarna 10C, 13D och 14G adderas startadressen med storleken så att adressen slutligen erhålls och jämförarna 10B och 13 G kontrollerar om aktuell linje är identisk med slutlinjen eller ej. Då dessa adresser är lika avger jämförarna signalen $\overline{RMC}$ som till "mover"-kontrollen indikerar att flyttning av sista linjen pågår.

Skiftfaktor

MTA och MFA-adresserna hanterar den egentliga adresseringen av minnet under det att de minst signifikanta adresserna bestämmer skiftfaktor och skrivmask.

Skiftfaktorn anger hur många positioner som skiljer mellan tilloch från-adress inom ordet. Den erhålls som utdata från PROM 1F, vars innehåll är en tabell som adresseras av TSP0-3, FSP0-3 samt U/$\overline{D}$ X. På utgångarna SH0-3 får man ett värde från 0 till 15 som är beroende på skillnaden mellan från- och tillordens pixelpositioner. Ur tabellen får man också uppgift om huruvida man måste läsa två ord innan ett kan skrivas ut eller om innehållet i första frånordet räcker för att skriva ett tillord. Detta indikeras med signalen HOLD 1W CYK som används av "mover"-kontrollen. Nedan följer två exempel på hur skiftfaktorn bildas:

Exempel på skiftning vid flyttning av data



Skiftfaktorn används dels till cirkelskiftaren där den styr hur många positioner frândata ska flyttas för att få tilldatas position och dels vid skrivning av data till grafikminnet för att avgöra vilka pixels från det gamla ordet och vilka från det senast inlästa som ska bilda tillordet.

Skrivmasken används då det första och sista ordet på varje linje skrivs ut till grafikminnet. Detta för att data i samma ord men utanför tilladresserna ska förbli opåverkade. I likhet med skiftfaktorn bildas skrivmasken med hjälp av en tabell som denna gång finns i kretsarna 1G och 1J. Adresserna till tabellen utgörs av TSP0-3, som talar om vilken pixel första ordet börjar på, PS0-3, som tillsammans med TSP-adressen talar om den sista pixelpositionen, U/$\overline{D}$ X, som talar om åt vilket håll räkningen sker så man vet om orden ska maskas före eller efter den aktuella pixelpositionen.

Skrivmask

De tre återstående adresserna WRMS0, WRMS1 samt ANDED MASKS talar om när första resp sista ordet uppträder samt om den area som ska flyttas är så liten att den ryms inom ett ord så att skrivmaskerna måste kombineras i ett och samma ord.

Dessa adresser ska också se till att alla andra ord utom det första och sista får en skrivmask så att samtliga minneskretsar skrives. Utgångarna från tabellen är anslutna direkt till minneskretsarnas WE-ingångar, vilket gör att de måste kunna stängas av då "movern" inte är aktiv samt ge en hög signal då "movern" läser i minnet. Detta löses genom att $\overline{AMM}$-signalen kopplas till CS-ingångarna på kretsarna vilket gör att de aldrig är aktiva annat än under en "mover"-sekvens.

Sedan får MOVE MEMWR-signalen, som är aktiv då "movern" eller CPU:n har möjlighet att skriva i grafikminnet, styra om skrivning eller läsning gäller. Tillsammans ser dock dessa signaler till att skrivning endast kan ske då "movern" är aktiv samt då skrivning ska utföras. Vid läsning ser pullup-motstånden R4 och R5 till att WE - signalen ligger på hög nivå då kretsarnas utgångar ej är aktiverade.

"Moverdata" - behandling av dataflödet

När "movern" läser ett ord ur minnet klockas det in i registren 6M och 8N så att stabila data finns på dess utgångar tills nästa läsning görs. Dessa data som nu kallas GMDR0-15 (Grafic Memory Data Registered) bildar indata till cirkelskiftaren. Den består av åtta stycken skiftregister (2U, 2V, 3N, 3T, 3U, 3V, 6U och 6V) som är så kopplade att de snabbt kan skifta runt ett 16-bitars ord från 0 till 15 positioner.

Skiftregistren är uppbyggda på grindlogik, vilket gör att de ej behöver klockas för att skifta ordet utan det nya ordet finns tillgängligt efter endast några få nanosekunder.

Det som avgör hur många steg som data ska skiftas är skiftfaktorn SH0-3 från tabellen i 1F. Genom att låta ordet, så att säga "bita sig själv i svansen" likt ett hjul, kan man skifta ordet önskat antal positioner utan att förstöra något av innehållet. Den främre registergruppen bestående av 3T, 3N, 6V och 3V kan med hjälp av SH0 och SH1 skifta orden 0,1,2 eller tre steg under det att den efterföljande gruppen 6V, 3V, 2U och 2V kan skifta ordet i 4-bitarssteg dvs 0, 4, 8 eller 12 beroende på SH2 och SH3. Genom att kombinera dessa uppnår man att data kan skiftas från 0-15 steg beroende på den totala skiftfaktorn.

Cirkelskiftare

Från cirkelskiftaren erhålls det skiftade ordet som signalerna ROT0-ROT15. Detta ord innehåller data som ligger i rätt postion inom det ord som ska skrivas ut till grafikminnet, men det är inte säkert att det stämmer med de ordgränser man måste ha för att skriva ett helt ord till minnet.

Genom att kombinera en del av den nyss skiftade ordet (nytt ord) med resterande del av det i sekvensen innan skiftade ordet (gammalt ord) erhåller man ett ord som i sin helhet kan skrivas till minnet. Det gamla ordet finns lagrat i registren 1V och 2N och bytes vid varje ny inläsning med hjälp av klockan DATA CLK. I PAL:arna 1M, 1T, 1N och 2T muxas det gamla och det nya ordet så att rätt fördelning uppnås.

Varje PAL är programmerad så att den innehåller fyra identiska muxar med separata muxsignaler, vilket gör att varje enskild utsignal kankomma från gammalt eller nytt ord oberoende av någon annan. Med hjälp av styrsignalen COMP MOVE FLAG från flaggregistret kan samtliga utsignaler inverteras så att ytan samtidigt som den flyttas också inverteras. Då utsignalerna från PALkretsarna kopplas direkt till minnenas ingångar, måste man också kunna sätta utgångarna högimpediva, vilket sker då signalen HOLD XY är hög dvs aktiv. Det som avgör vilken del av det nya resp gamla ordet som får bilda data till minnet avgörs av skiftfaktorn SH0-3 samt räknestroben $U/\overline{D}$ X. Via tabellerna i PROM-kretsarna 1K och 1L genereras de individuella muxsignalerna som krävs för att styra data på ovan angivet sätt.

"Mover"-kontroll

Till sist beskrivas den del som håller samman och styr hela flyttningen av data i grafikminnet, nämligen "mover"-kontrollen. Den samlar in uppgifter från sekvenskontrollen, flaggregister och "moverns" alla delar och sänder ut styrsignaler till adressering och dataflöde beroende på insignalernas status. "Mover"-kontrollen styr framförallt det som ska hända vid början och slutet av varje linje samt start och avslutning av hela "mover"-sekvensen. Signalerna kan grovt delas in i följande fyra kategorier:

- Signaler som startar/stoppar en "mover"-sekvens
- Signaler som laddar och klockar register
- Signaler som fördröjer eller håller ett förlopp
- Strobar för skrivmask

En "mover"-sekvens initieras då CPU:n laddar sista registret med startadresser. Det är från-Y-registret som laddas med stroben LDFY LB från I/O-avkodningen. Stroben klockar också D-vippan 8F/1 vars utsignal går hög och påverkar ingång 12 på registret 6D. Detta register klockas med AMM CLK som skapas i 12G/2 genom klockning av MOVE CYK CLK från sekvenskontrollen.

Då AMM CLK klockar registret blir AMM aktiv som indikerar att "mover"-sekvensen har börjat. AMM eller den inverterade signalen $\overline{\text{AMM}}$ distribueras till en stor del av videoenheten för att indikera att "mover"-moden är aktiv. I detta läge börjar "movern" att arbeta självständigt och sekvensen avslutas inte förrän adressjämförarna samtidigt aktiverar CMC och RMC. Dessa signaler påförs OR-grinden 12D/1 vars utsignal går låg och vidare genom 12D/4 där den eventuellt efter en viss fördröjning kopplas till D-vippan 7D/2. Den klockas av $\overline{\text{AMM-CLK}}$ som gör att dess utsignaler slår om och via AND-grinden 6F/4 påverkar resetingången på 8F/1 och "mover"-moden avslutas då klockning av AMM-registret 6D sker.

Att "mover"-moden inte avslutas direkt beror på att det sista ordet måste skrivas ut samt att ett antal startadressregister ska laddas automatiskt med nya data för att vara förberedda att flytta data till nästa teckenposition. Hur registren laddas beror på bildskärmens läge, dvs landskap eller porträttmod.

Ett antal av de laddningsstrobar som CPU:n använder för att ladda startvärden i adressregistren används inte direkt utan grindas i "mover"-kontrollen först. Det beror på att dess register även ska kunna laddas av andra orsaker än av laddningsstrobarna.

Först gäller det alla X-adressregister som måste kunna laddas med nytt startvärde vid varje linjestart men även Y-adresser som tillsammans med X-adresserna automatiskt ska peka på nästa position efter avslutad "mover"-sekvens.

Följande utsignaler och klockor styr laddning av adressregistren:

- TX FX CLK ----- Klockpuls som stegar fram X-adresserna ett steg för varje nytt ord på linjen. Bildas i AND-grinden 9C/2 och aktiveras då AMM är aktiv.

- CLK Y FROM HB ----- Klockar linjeräknaren 11F och bildar AND-grinden 9C/3. Aktiveras av laddningsstroben LD FY HB samt av RD/WR MASK (liknar AMM CLK) då AMM är aktiv. Klockningen sker en gång vid varje ord men används endast vid linjeslut beroende på räknarens enable-signaler.

- CLK Y FROM LB ----- Identisk funktion med ovanstående men klockar registren 10F och 12F och aktiveras även av laddningspulsen LD FY LB.

- TY CLK ----- Klockar räknarna som innehåller aktuell linje. Laddar detta register då AMM aktiveras samt en gång för varje linje. Bildas i AND-grinden 7C/4 där den dels aktiveras av LDTY LB genom D-vippan 8F/2 samt av R/W MASK då "movern" är aktiv.

  TY REG HB och TY REG LB laddar startadressen i linjeräknarna med hjälp av LDFY HB resp LDFY LB. Genom AND-grinden 6F/2 och 3 aktiveras de också då "mover"-moden avslutas om bildskärmen står i landskapsläge.

- $\overline{\text{LD X REG}}$ ----- Laddar startadressen i X-led vid början av "mover"-sekvensen och med bildskärmen i porträttläge även vid avslutning. Bildas i AND-grinden 9C/1 där den påverkas av LDTX HB via D-vippan 8D/1 samt eventuellt vid avslut av 7D/2.

- TX REG HB och TX REG LB klockar X-startregistren och aktiveras av laddningspulserna LDTX HB resp LDTX LB genom AND-grinden 7C/2 och 3. Med bildskärmen i porträttläge avges klockpuls även vid avslut med hjälp av AMM CLK och L/P FLAG genom OR-grindarna 8C/4 och 10D/2.

- $\overline{\text{LD X}}$ ----- Laddar aktuellt ord från startadressregistren vid slutet av varje linje. Bildas i OR-grinden 12D/2 av $\overline{\text{CMC}}$ där den eventuellt kan fördröjas något men går sedan via muxen 7F till registren både i landskap och porträttläge.

- LD Y ----- Aktiverar linjeregistren så att laddning kan ske. Aktiveras automatiskt vid "mover"-start men även vid avslutning i porträttläge för att återgå till startlinjen för tilladressen så att den automatiskt får rätt position för nästa tecken.

- X BUFF OE ----- Öppnar dataportarna från X-till-adresserna vid avslutning så att den sista adressen får bilda startadress till nästa teckenposition i landskapsläge. Signalen bildas i ORgrinden 8C/1 av L/P FLAG och "moverslutpulsen" från 7D/2.

- Y BUFF OE ----- Samma funktion som ovanstående men i porträttläge. Bildas av L/P FLAG samt "moverslutpulsen".

För att få tillräckligt med data att bearbeta och för att vid "mover"-slut hinna skriva ut data och ladda registren kan man ibland låsa det aktuella läget under en kortare tid. Det som styr detta är bl a om man måste läsa ett eller två ord innan något kan skrivas ut eller om ytan är så liten att den ryms inom ett ord vilket gör att jämföraren indikerar slut innan något skrivits.

Från skifttabellen får vi signalerna HOLD 1W CYK och dess inverterade signal. De talar om att två ord måste läsas för att ett helt ord ska kunna skrivas.

Från D-vippan 12B/2 får vi signalen ANDED MASKS som vid hög nivå indikerar att hela ytan ryms inom ett ord. I AND-grinden 12C/1 kontrolleras om både HOLD 1W CYK och ANDED MASKS är aktiva under en "mover"-sekvens och då så är fallet ger de upphov till fördröjningssignalerna DELAY IF HOLD och HOLD IF ANDED. DELAY IF HOLD blockerar via OR-grinden 12D/4 så att "moverslutsignalen" från utgången på samma grind fördröjs en läs/skrivsekvens. HOLD IF ANDED ser via OR-grinden 8C/3 och AND/NOR-grinden 7B/2, tillsammans med HOLD 1W CYK till att signalen HOLD TX WORD aktiveras. HOLD TX WORD låser här tilladresserna i X-led genom att stoppa räknarna för "aktuell ordadress" under en läs/skrivsekvens. Signalen är också aktiv innan AMM aktiveras samt då HOLD 1W CYK är aktiv och WRMS1 ej hunnit klockas genom AMM-registret. Detta gör att HOLD TX Word aktiveras under en läs/skriv-sekvens i början av linjen.

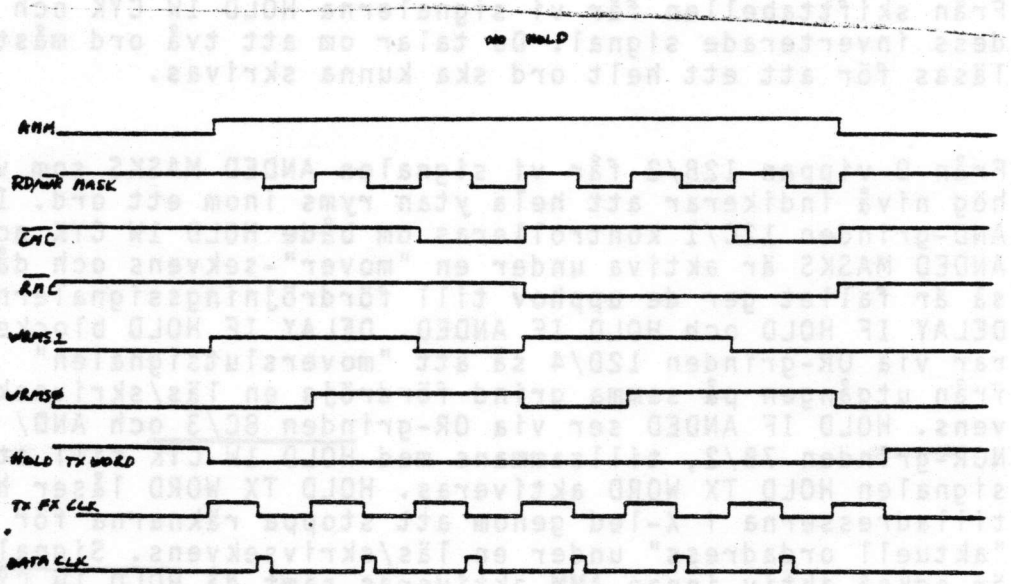
Signalen HOLD XY blockerar alla räknare under tiden AMM inte är aktiv fram tills AMM klockats genom D-vippan 7D/1 av AMM CLK. Signalen öppnar också mux 7F så att LDX och LDY kan aktiveras samt PAL-kretsarnas utgångar för skrivning av data till grafikminnet.

Från "mover"-kontrollen levereras också adresser till skrivmasktabellen. med signalerna WRMS0 och WRMS1 erhålls fyra olika kombinationer som indikerar följande lägen:

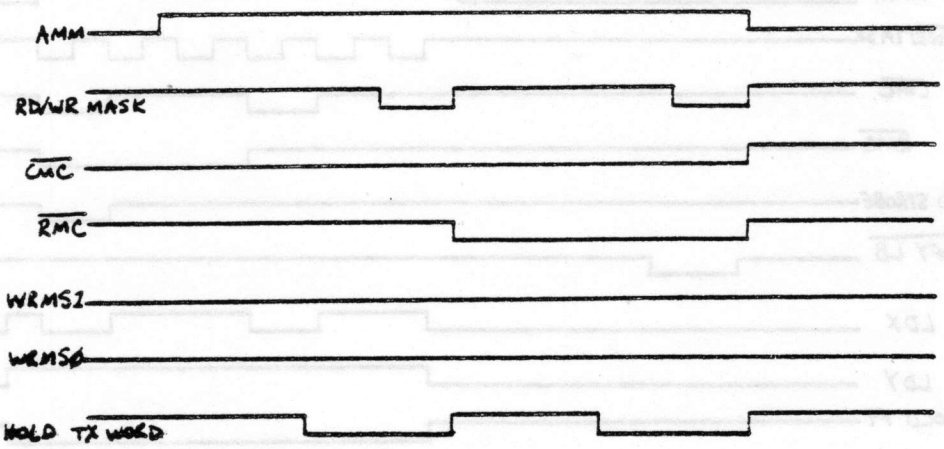
WRMS1	WRMS0	Funktion
LÅG	LÅG	Början och slut i samma ord
LÅG	HÖG	Sista ordet skrivs ut
HÖG	LÅG	Första ordet skrivs ut
HÖG	HÖG	"Movern" aktiv, mitt i en linje

WRMS1 styrs direkt av $\overline{\text{CMC}}$ och AMM via AND-grinden 6F/1 vilket gör att signalen går hög då "movern" aktiveras och ligger så tills jämföraren med $\overline{\text{CMC}}$ indikerar att sista ordet på linjen adresseras. WRMS0 bildas i grindsystemet 7G/1, 3 och 4 som ser till att utsignalen går hög, en eller två läs/skrivsekvenser efter WRMS1 beroende på om HOLD 1W CYK är aktiv eller ej.
I nedanstående tidsdiagram visas några olika "mover"-sekvenser som kan förekomma:

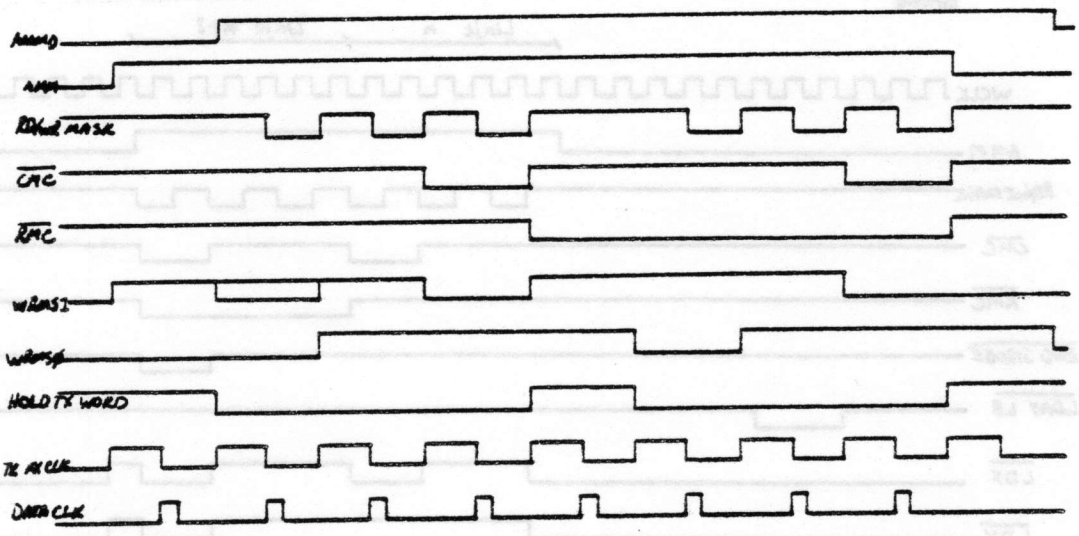
Tidsdiagram



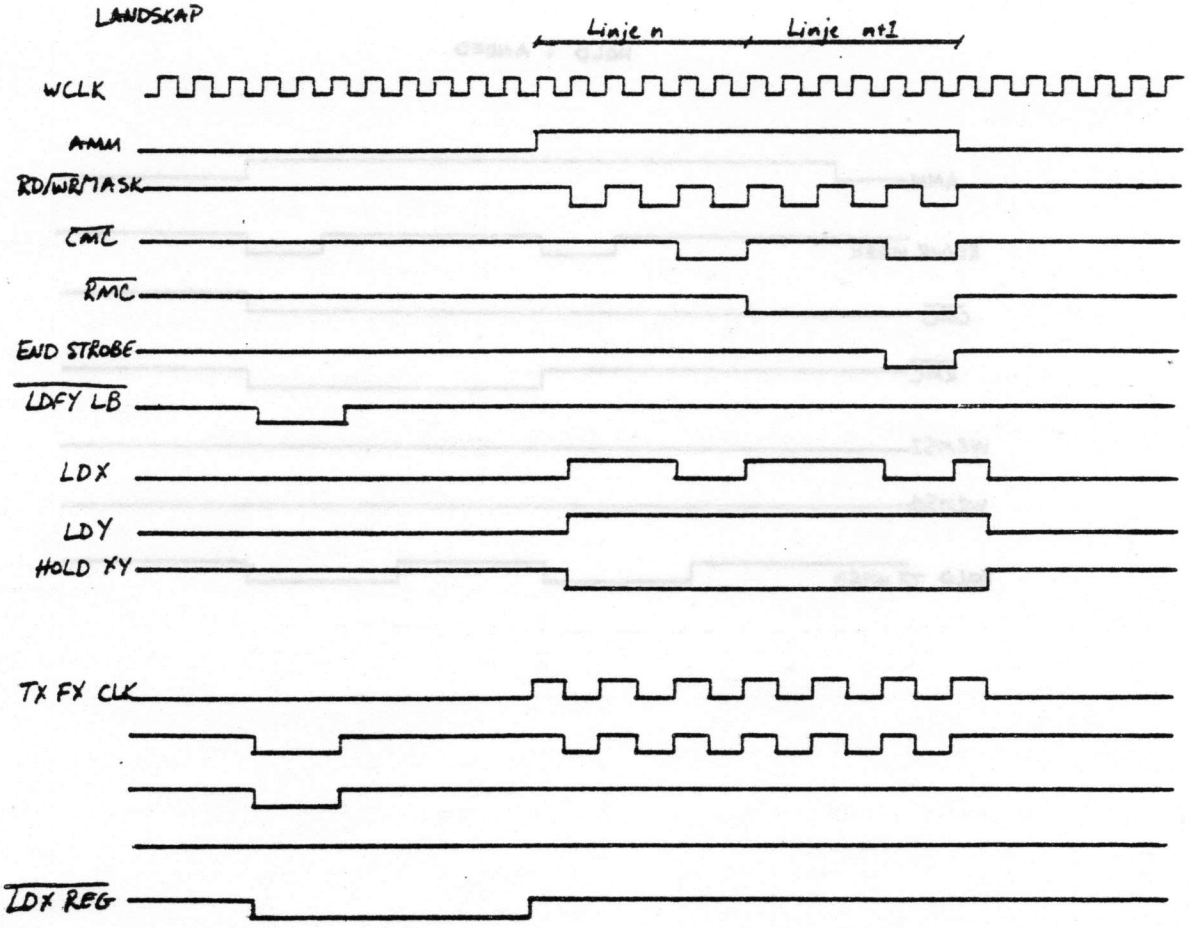
HOLD + ANDED



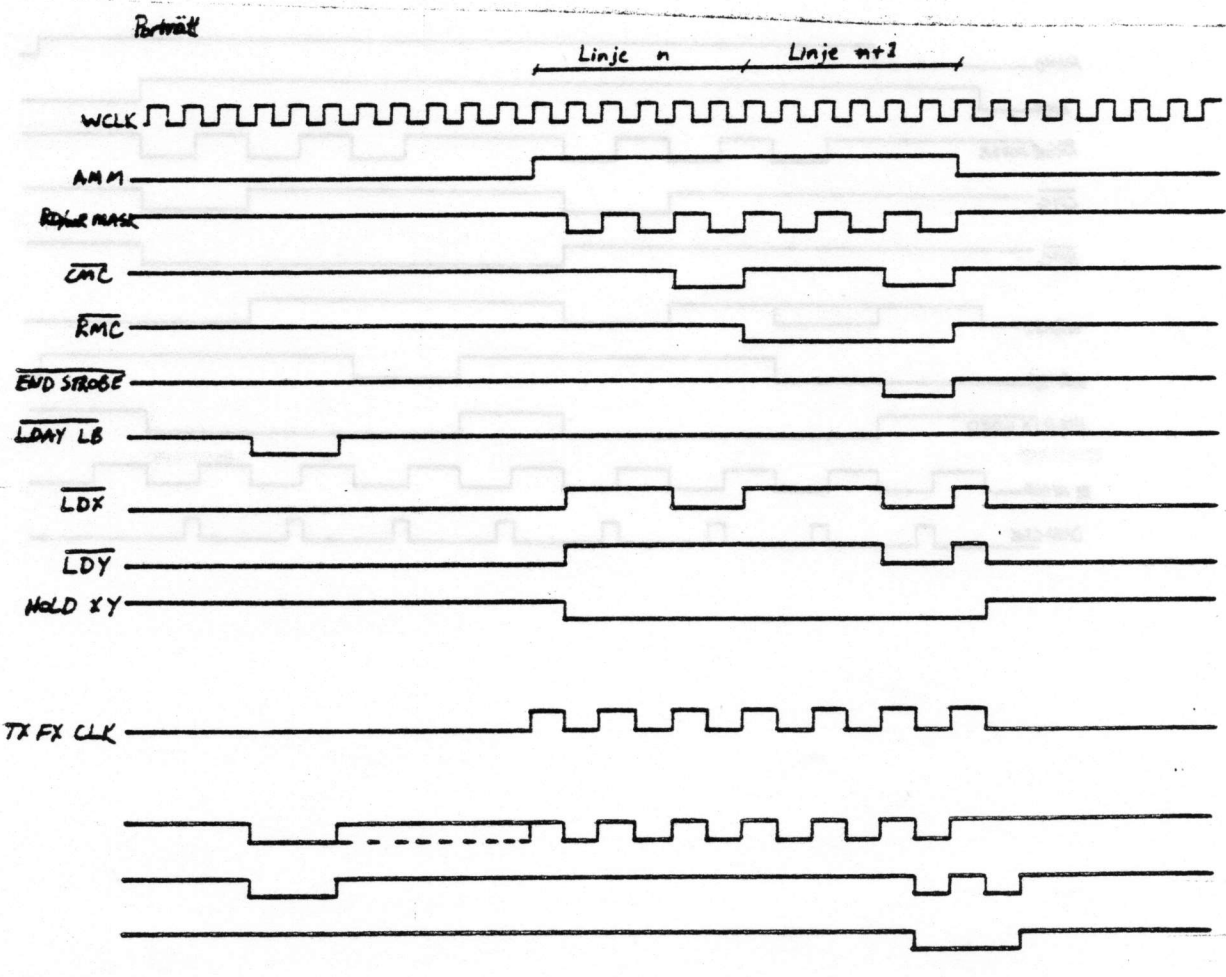
HOLD



LANDSKAP



Porträtt



4 PAL-kretsar

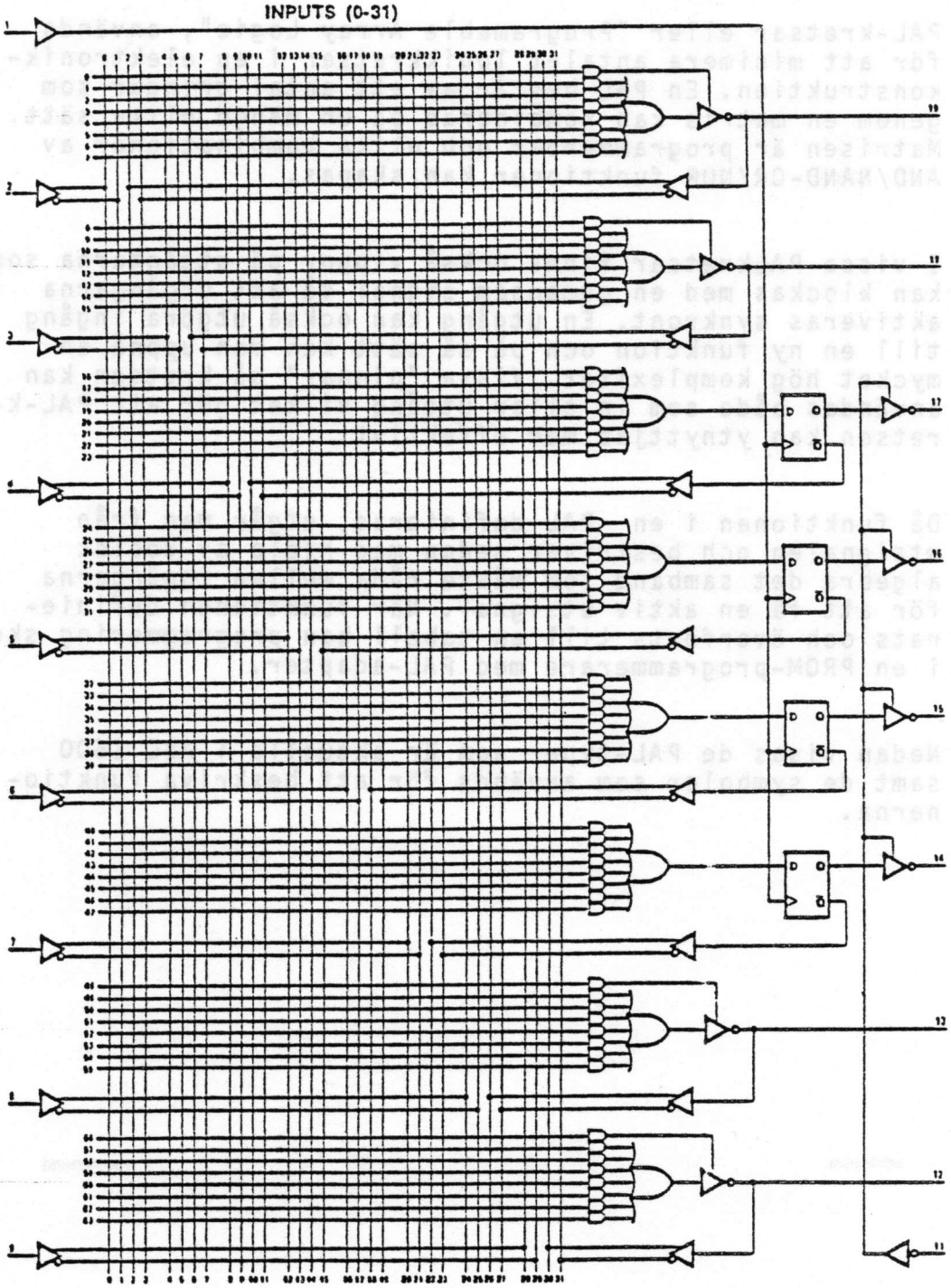
PAL-kretsar eller "Programable Array Logic", används för att minimera antalet logikkretsar i en elektronikkonstruktion. En PAL består av ett antal grindar som genom en matris kan kombineras på en mängd olika sätt. Matrisen är programmerbar och olika kombinationer av AND/NAND-OR/NOR funktioner kan skapas.

I vissa PALkretsar finns också vippor på utgångarna som kan klockas med en gemensam signal så att utgångarna aktiveras synkront. En utgång kan också utgöra ingång till en ny funktion och på så sätt kan man uppnå en mycket hög komplexitet. Vissa "pinnar" på kretsen kan användas både som in eller utgång vilket gör att PAL-kretsen kan ytnyttjas mer effektivt.

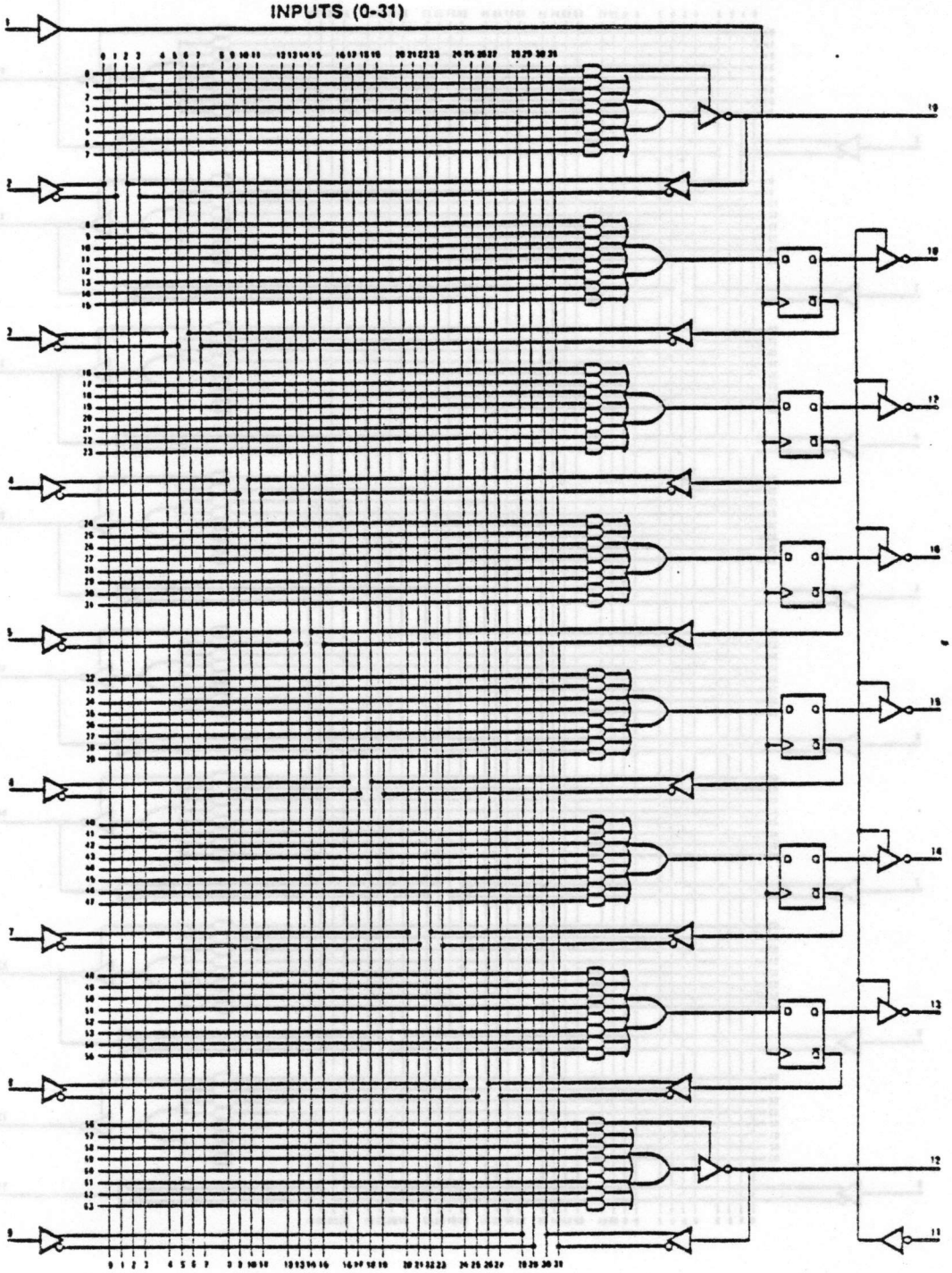
Då funktionen i en PAL definieras, utgår man från utsignalen och beskriver sedan med hjälp av Boolsk algebra det samband som måste råda mellan ingångarna för att få en aktiv utsignal. När funktionen definierats och överförts till en tabell kan programmering ske i en PROM-programmerare med PAL-adapter.

Nedan visas de PAL-typer som är aktuella i ABC 1600 samt de symboler som används för att beskriva funktionerna.

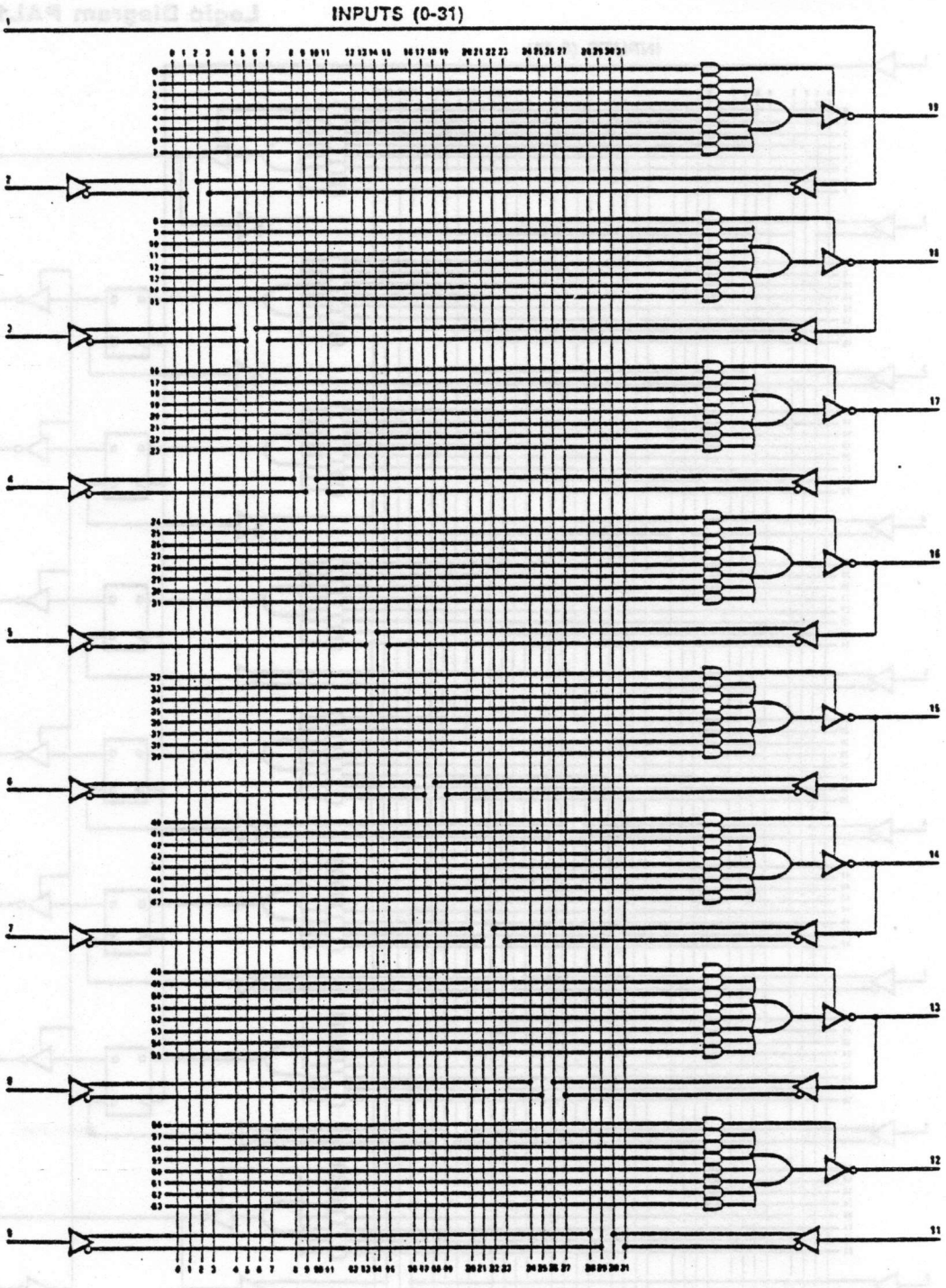
Logic Diagram PAL16R4



Logic Diagram PAL16R6



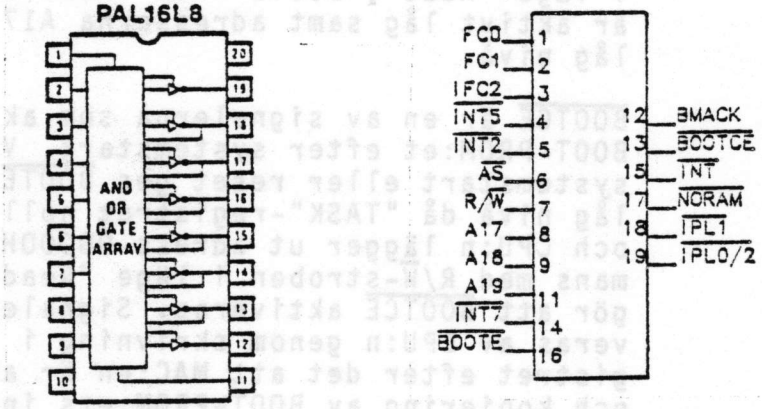
Logic Diagram PAL16L8



4.1 PAL 5D

UPPGIFT: Prioritetsavkodare för "interrupt enable"
BOOT-PROM
Kvittenssignal BOOT-PROM
Kontroll RAM-area

TYP: PAL 16L8



INSIGNALER:

FC0	-	Funktionskod 0 från CPU
FC1	-	Funktionskod 1 från CPU
IFC2	-	Inverterad funktionskod 2
$\overline{INT5}$	-	Interruptbegäran från SCC och DART
$\overline{INT2}$	-	" " CIO
$\overline{AS}$	-	Adresstrob från CPU
R/$\overline{W}$	-	Läs/skrivsignal från CPU

A17 - Adress 17 från CPU

A18 - Adress 18 från CPU

A19 - Adress 19 från CPU

$\overline{INT7}$ - NMI från paritetsfel eller BUSOX

$\overline{BOOTE}$ - BOOT-enable från TASK-register

UTSIGNALER:

$$\overline{BOOTCE} = R * \overline{BOOTE} * \overline{A17} * \overline{A18} * \overline{A19}$$

BOOTCE är aktivt låg då R/W-signalen står i läge "Read", $\overline{BOOTE}$ från TASK-registret är aktivt låg samt adresserna A17-A19 har låg nivå.

$\overline{BOOTCE}$ är en av signalerna som aktiverar BOOT-PROM:et efter systemstart. Vid systemstart eller reset ges $\overline{BOOTE}$ aktiv låg nivå då "TASK"-registret nollställs och CPU:n lägger ut adress 00000H tillsammans med R/$\overline{W}$-stroben i läge "Read", vilket gör att $\overline{BOOTCE}$ aktiveras. Signalen deaktiveras av CPU:n genom skrivning i TASKregistret efter det att MAC:en är aktiverad och kopiering av BOOT-PROM:ets innehåll skett till RAM-arean.

$$\overline{BMACK} = AS +$$
$$BOOTCE * \overline{A19} +$$
$$BOOTCE * IFC2 +$$
$$BOOTCE * FC1 +$$
$$BOOTCE * \overline{FC0} +$$

BMACK-signalen ger $\overline{DTACK}$ till CPU:n, då BOOT-PROM:et aktiveras eller vid skrivning och läsning i MAC. Observera att signalen är aktiv vid hög nivå inte vid låg nivå som uttrycket beskriver. Det gör att signalen är aktiv i alla lägen utom de angivna.

Då BOOT-PROM:et aktiveras kommer BMACK att vara aktiv då adresstroben $\overline{AS}$ och $\overline{BOOTCE}$ båda går låga. De fyra sista uttrycken är avsedda att kontrollera att kvittenssignal ska avges då skrivning eller läsning i MAC sker. Vi finner att BMACK är hög (aktiv) då A19 är hög i systemmod och då dataarea adresseras. (Jmf då MAC aktiveras).

$\overline{INT} = \overline{IFC2} * FC1 * FC2$

$\overline{INT}$ (Interrupt acknowledge) avkodas från CPU:ns funktionskoder och då alla har hög nivå är signalen aktiv. Observera att IFC2 är det inverterade värdet av FC2.

$\overline{ILP1} = \overline{INT7} +$

$\overline{INT2} * INT5$

$\overline{ILP1}$ är en av de två signalerna som ges till CPU:n vid interruptbegäran. Den är aktiv vid NMI ($\overline{INT7}$) och då CIO:n (INT2) begär interrupt. Vid interruptnivå 2 kontrolleras att nivå 5 från SCC:n ej avgivits, då CPU:n i detta läge kunde fått nivå 7.

$\overline{ILP2} = \overline{INT7} +$

$\overline{INT5}$

$\overline{ILP2}$ avkodas tillsammans med $\overline{ILP1}$ av CPU:n för att avge interruptnivån. Signalen är aktiv vid NMI (INT7) och vid interruptbegäran från SCC:n ($\overline{INT5}$).

$\overline{NORAM} = \overline{BOOTE} * \overline{A17} * \overline{A18}\ \overline{A19} +$ (1)

$A19 * \overline{IFC2} * \overline{IFC1} +$ (2)

$\overline{IFC2} * FC1 * FC0$ (3)

$\overline{NORAM}$-signalen indikerar att ingen RAM-area ska aktiveras. Signalen är aktiv i tre olika lägen:

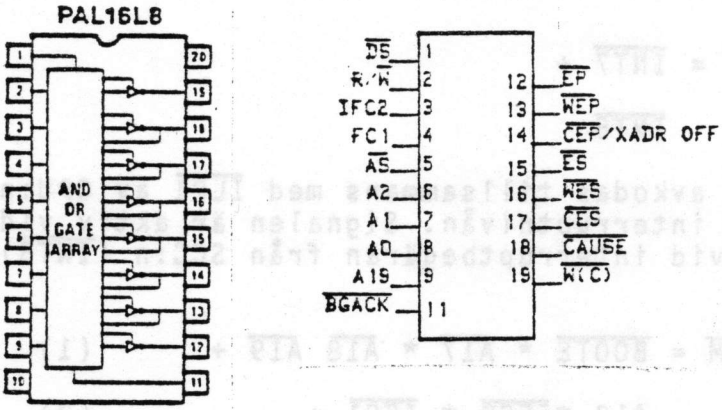
- BOOT-PROM:et aktiveras (1)
- Vid skrivning/läsning i MAC (2)
- "Interrupt acknowledge" (3)

4.2 PAL17E

UPPGIFT:

Levererar enable-signaler och strobar till MAC:ens register, minnen och dataportar samt lässtrobar till CAUSE-registret.

TYP: PAL16L8



INSIGNALER:

$\overline{DS}$	=	Datastrob från CPU
R/$\overline{W}$	=	Skriv/lässignal från CPU
IFC2	=	Inverterad funktionskod 2
FC1	=	Funktionskod 1 från CPU
$\overline{AS}$	=	Adresstrob från CPU
A2	=	Adress 2 från CPU
A1	=	" 1 " "
A0	=	" 0 " "
A19	=	" 19 " "
$\overline{BGACK}$	=	"Bus Grant ACKnowledge" från busskontroll

UTSIGNALER:

$$\overline{EP} = IFC2 + \overline{FC1} + \overline{A19} + A2 + A1 + \overline{DS}$$

EP aktiverar dataportarna för skrivning och läsning i "page"-RAM och fördelas till de två portarna med hjälp av mux 21F som fördelar signalerna med hjälp av A0. I muxen inverteras signalen så att den här måste tolkas då den har hög nivå, d v s inverterad funktion. Detta gör att systemmod, dataarea, A19 hög, A1 och A2 låga samt $\overline{DS}$ aktiv, alla måste gälla för att EP ska aktiveras.

$$\overline{WEP} = IFC2 + \overline{FC1} + \overline{A19} + A2 + A1 + R/W + AS$$

WEP aktiverar $\overline{WE}$ -ingångarna (Write Enable) på "page"-minnet. Signalen behandlas på samma sätt som EP och måste därför tolkas inverterad. Följande villkor måste vara uppfyllda för att signalen ska aktiveras:

Systemmod, dataarea, A19 hög, A1 och A2 låg nivå, $\underline{R/W}$ i läge "Write" d v s låg nivå samt aktiv adresstrob $\overline{AS}$.

$$\overline{CEP} =$$

$$IFC2 * BGACK + \quad (1)$$

$$\overline{A19} * BGACK + \quad (2)$$

$$\overline{IFC2} * \overline{FC1} * A19 * R/\overline{W} * BGACK + \quad (3)$$

$$\overline{IFC2} * \overline{FC1} * A19 * \overline{A2} * \overline{A1} * \overline{R/W} * \overline{DS} * \overline{AS} * BGACK \quad (4)$$

$\overline{CEP}$ (Chip Enable Page) eller XADR OFF, som signalen också kallas, aktiverar "page"-minnet i MAC:en. Signalen kontrollerar också då någon DMA har begärt busskontroll för att se om MAC:en slutat sända adresser.

För att $\overline{CEP}$ ska vara aktiv krävs att CPU:n har busskontroll, vilket kontrolleras med hjälp av BGACK. $\overline{CEP}$ aktiveras alltid i användarmod (adressering sker genom MAC) (1), alltid då $\overline{A19}$ är låg (även systemet adresserar genom MAC (2), vid läsning i "page"-minnena (systemmod, dataarea, A19 hög samt R/$\overline{W}$ i läge "Read" (3) och vid skrivning i "page"-minnena (samma som vid läsning med undantag för R/$\overline{W}$ i läge "Write", adresskontroll av A1 och A2 samt aktiv data och adresstrob (4).

$$\overline{ES} = \overline{IFC2} * \overline{FC1} * A19 * \overline{A2} * A1 * A0 * \overline{DS}$$

$\overline{ES}$ (Enable Segement-RAM) aktiverar dataporten till "segment"-minnet. Detta sker vid både skrivning och läsning under förutsättning att följande villkor samtidigt är uppfyllda:

> Systemmod, dataarea, A19 är hög, A2 är låg, A0 och A1 båda höga och att datastroben är aktiv.

$$\overline{WES} = \overline{IFC2} * \overline{FC1} * A19 * \overline{A2} * A1 * A0 * \overline{R/W} * \overline{AS}$$

$\overline{WES}$ (Write Enable Segment-RAM) är aktiv då skrivning sker i "segment"-RAM:arna. Samma förutsättningar gäller som för $\overline{ES}$ med undantag för R/$\overline{W}$-stroben som måste stå i läge "Write" samt att adresstroben ($\overline{AS}$) ska vara aktiv.

$\overline{CES}$ =

$IFC2$ + (1)

$\overline{A19}$ + (2)

$\overline{IFC2} * A19 * R/\overline{W} * BGACK$ + (3)

$\overline{IFC2} * \overline{FC1} * A19 * \overline{A2} * \overline{A1} * BGACK$ + (4)

$\overline{IFC2} * \overline{FC1} * A19 * \overline{A2} * A1 * A0 * \overline{R/W} * \overline{DS} * \overline{AS} * BGACK$ (5)

$\overline{CES}$ (Chip Enable Segment-RAM) dels då adressering sker genom MAC, dels då systemet skriver eller läser i "segement"-minnena men även då systemet skriver eller läser "page"-RAM:et då detta måste ha aktiva adresser vid uppdatering. $\overline{CES}$ aktiverar alltid i användarmod (1), alltid då $\overline{A19}$ är låg (systemet adresserar genom MAC) (2), då systemet läser i "segment"-minnet (3), då "page"-minnet läser eller skriver (4) samt då systemet skriver i "segment"-minnet (5).

$\overline{CAUSE} = \overline{IFC2} * \overline{FC1} * A19 * A2 * A1 * A0 * \overline{DS} * \overline{R/W} * BGACK$

CAUSE-stroben används för att läsa CAUSE-registret som innehåller uppgifter om adressen till den minnesarea där ett paritetsfel inträffat men används också som återställningssignal för "watchdog". Signalen aktiveras endast under förutsättning att följande samband råder: systemmod, adressering i dataarea, A19 hög, A0-A2 höga, datastrob aktiva, R/$\overline{W}$-signalen i läge "Read" och BGACK inaktiv.

$W(C) = \overline{IFC2} * \overline{FC1} * A19 * A2 * \overline{A1} * A0 * \overline{R/W} * \overline{DS} * BGACK$

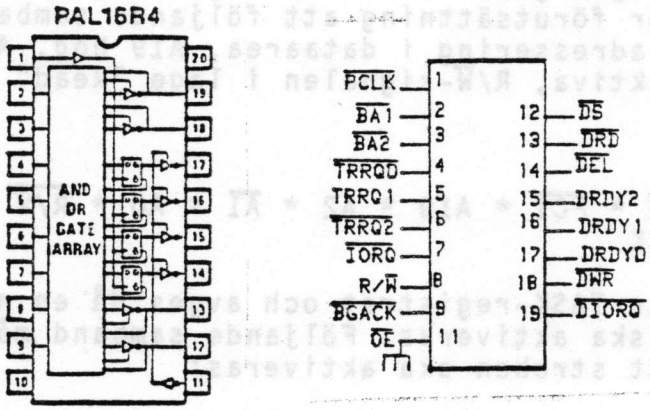
W(C) klockar TASK-registret och avges då en ny användarprocess ska aktiveras. Följande samband måste alla råda för att stroben ska aktiveras:

Systemmod, adressering i dataarea, A19 hög, A0-A2 höga, datastrob aktiv, R/$\overline{W}$ i läge "Write" och BGACK ej aktiv.

4.3 PAL6E

Uppgift: Hanterar strobomvandlingen mellan CPU/DMA samt genererar RDY-signaler till DMA.

Typ: PAL 16R4



Insignaler: $\overline{PCLK}$ Periferiklocka från klockgenerering (4MHz)

$\overline{BA1}$ Aktiv efter en $\overline{BGACK}$-signal då DMA1 eller DMA2 har begärt busskontroll.

$\overline{BA2}$ Aktiv efter en $\overline{BGACK}$-signal då DMA2 har begärt busskontroll.

$\overline{TRRQ0}$ "Transfer Request"-begäran från Bus0 eller flexskiveinterface till DMA0.

$\overline{TRRQ2}$ "Transfer Request"-begäran från BUS 2 (Winchesterinterface) till DMA2.

$\overline{IORQ}$ "I/O Request"-signal från PAL 11E

R/$\overline{W}$ Läs/skrivstrob från CPU.

$\overline{BGACK}$ "Bus Grant Acknowledge" från 18C/3

Utsignaler:

$$\overline{DS} = DIORQ * \overline{DRD} * \overline{DEL} + DIORQ * \overline{DWR} * \overline{DEL} + DIORQ * \overline{DRD} * R/\overline{W} \quad \text{(Om } \overline{BGACK}\text{)}$$

Signalen aktiveras endast då $\overline{BGACK}$ är aktiv dvs. då någen DMA tagit busskontrollen. Då ersätter den CPU:ns datastrob och är aktiv då DMA:n skriver eller läser mot minnet eller någen yttre enhet.

$$\overline{DRD} = \overline{IORQ} * R/\overline{W} \quad \text{(Om BGACK)}$$

Som utsignal används den endast då CPU:n har busskontrollen och läser ur DMA:ns interna register.

Då någon DMA har busskontrollen sätts denna utgång högimpediv då DMA:n själv driver signalen.

$\overline{DWR} = \overline{IORQ} * \overline{R/W}$ (Om BGACK)

Som ovan enligt $\overline{DRD}$ men gäller vid skrivning.

$\overline{DIORQ} = \overline{IORQ}$

Signalen följer den i PAL 11E bildade IORQ-signalen och används då CPU:n skriver eller läser i någon DMA:s interna register.

$\overline{DEL} = \overline{DRD} * \overline{BGACK} + \overline{DWR} * \overline{BGACK}$

Signalen är aktivt låg då någon DMA har busskontrollen och gör en skrivning eller läsning. Den används endast internt i kretsen och ger en fördröjd signal som används för att få rätt klockning.

$$\begin{aligned}\overline{DRDY0} = {} & \overline{TRRQ0} * BA1 * DWR * DRD * DEL + \quad (1)\\ & \overline{DRDY0} * DWR * DRD + \quad (2)\\ & \overline{TRRQ0} * \overline{BGACK} + \quad (3)\\ & \overline{TRRQ0} * \overline{BA1} \quad (4)\end{aligned}$$

Signalen ger RDY-signal till DMA0 och används inverterad dvs. då signalen är hög. $\overline{DRDY0}$ kan således aktiveras under förutsättning att $\overline{BGACK}$ är aktiv (3), men DMA0 får inte ha lämnat över busskontrollen till någon annan DMA (4). Dessutom ska signalena inte bli aktiva innan DMA:n aktiverats (1). Samband (2) ser till att signalen blir inaktiv då läs och skrivstrobarna gått tillbaka till sitt inaktiva läge. Som sammanfattning kan vi säga att signalen aktiveras då DMA0 tagit över busskontrollen, BUS0 eller flexskiveinterfacet har skickat TRRQ och DMA0 lagt ut skriv- eller lässtrob beroende på vad den programmerats att utföra.

$$\begin{aligned}\overline{DRDY1} = {} & TRRQ1 * \overline{BA1} * BA2 * DWR * DRD * DEL\\ & + \\ & \overline{DRDY1} * DWR * DRD +\\ & TRRQ1 * \overline{BA1} +\\ & TRRQ1 * \overline{BA2}\end{aligned}$$

$\overline{DRDY1}$ lämnar RDY-signal till DMA1 dvs. då BUS1 eller SCC begär "Transfer Request".

Signalen används som ovanstående inverterad och observera också att TRRQ-signalen här är aktivt hög. Ingången som kontrolleras vid busskontrollen är BA1 och dess utgång är BA2. I övrigt samma funktion som ovanstående.

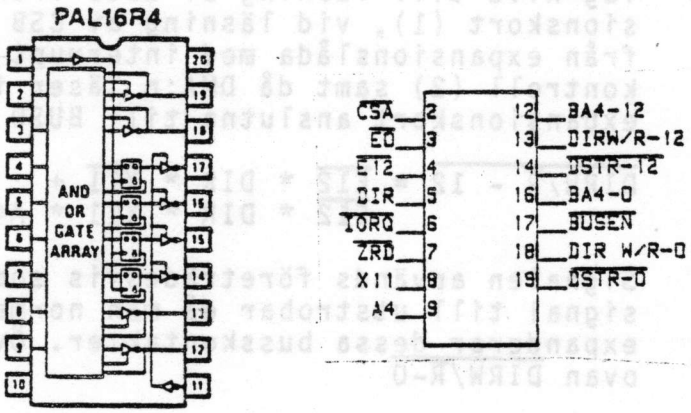
$$\overline{DRDY2} = \overline{TRRQ2} * \overline{BA1} * \overline{BA2} * DWR * DRD * DEL + \overline{DRDY2} * DWR * DRD + \overline{TRRQ2} * BA1 + \overline{TRRQ2} * BA2$$

$\overline{DRDY2}$ lämnar RDY-signal till DMA2 dvs. då BUS2 begär "Transfer Request". Funktionen enligt DRDY0 med undantag för busskontrollen som kräves att både BA1 och BA2 har låg nivå för att DMA2 ska vara aktuell. I övrigt samma funktion som ovan.

4.4 PAL8B

Uppgift: Strobgenerering BUS-interface

Typ: PAL 16R4



Insignaler:

$\overline{CSA}$ Strob för kortadress från PAL 12E

EO Enable BUSO1 och BUSOX från PAL 12E

E12 Enable BUS1 och BUS2 från PAL 12E

DIR Ger portriktning skrivning/läsning från PAL 12E

$\overline{IORQ}$ "I/O Request" från PAL 11E

$\overline{ZRD}$ Lässtrob från PAL 11E

X11 Adress X11 från MAC eller DMA-MAP

A4 Adress A4 från CPU eller DMA

Utgångar:

$\overline{BA4-0}$ = $\overline{A4}$ (Då $\overline{EO}$ är aktiv)

$\overline{BA4-0}$ levereras till bussinterface BUSO1 och BUSOX som adressledning 4. BA4-0 följer adress A4 från CPU eller DMA under förutsättning att enablestroben $\overline{EO}$ är aktiv (låg).

$\overline{BA4-12}$ = $\overline{A4}$ (Då $\overline{E12}$ är aktiv)

$\overline{BA4-12}$ levereras till bussinterface BUS1 och BUS2 som adressledning 4. BA4-12 följer adress A4 från CPU eller DMA under förutsättning att enablestroben $\overline{E12}$ är aktiv (låg).

$$\overline{DIR\ W/R\text{-}0} = \overline{EO} * DIR * \overline{X11} + \quad (1)$$
$$\overline{EO} * DIR * X11 * A4 + \quad (2)$$
$$\overline{EO} * DIR * \overline{TREN\text{-}0} \quad (3)$$

Signalen ger enable-signal till avkodaren för utstrobar vid hög nivå. Levereras till BUS0-interfacen där den styr portriktningen hos elektroniken i expansionskabeln. Ger låg nivå till läsning av data från expansionskort (1), vid läsning av CSB eller INT från expansionslåda med interrupt och CSB-kontroll (2) samt då DMA:n läser från expansionskort anslutna till BUS0 (3).

$$\overline{DIRW/R - 12} = \overline{E12} * DIR * \overline{X11} +$$
$$\overline{E12} * DIR * X11 * A4$$

Signalen används företrädesvis som enable-signal till utstrobar då man normalt ej expanderar dessa busskontakter. Övrigt se ovan $\overline{DIRW/R\text{-}0}$

$$\overline{BUSEN} = \overline{CSA} + \quad (1)$$
$$\overline{EO} * \overline{DIR} + \quad (2)$$
$$\overline{EO} * DIR * \overline{X11} + \quad (3)$$
$$\overline{EO} * DIR * X11 * A4 + \quad (4)$$
$$\overline{EO} * DIR * \overline{TREN\text{-}0} \quad (5)$$

Signalen aktiverar elektroniken i expansionskabeln till BUS0X. Den är aktivt låg då kortadressen sändes ut (1), vid skrivning till något expansionskort (2), då data läses från ett expansionskort (3), vid läsning av CSB och INT från expasionslåda med interrupt eller CSB-kontroll (4), samt då DMA:n läser från expansionskort anslutna till BUS0.

$$\overline{DSTB\text{-}0} = \overline{EO} * \overline{IORQ} * DIR +$$
$$EO * \overline{IORQ} * \overline{ZRD} * DIR * \overline{X11}$$

Datastrob som aktiverar strobavkoderna både vid skrivning och läsning. Signalen är aktivt låg vid all hantering av BUS0, utom vid avkänning av CSB och INT samt vid DMA-hantering.

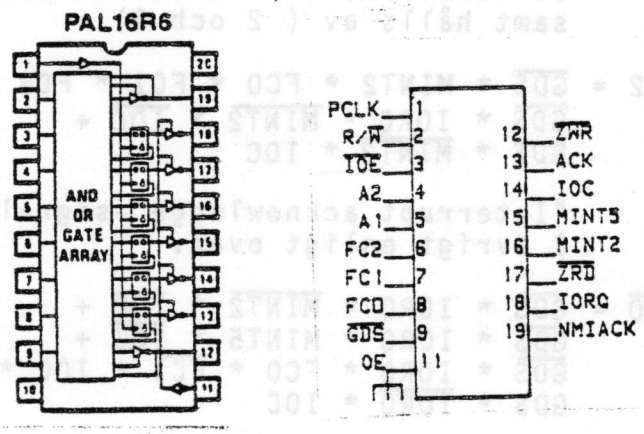
$$\overline{DSTB\text{-}12} = \overline{E12} * \overline{IORQ} * \overline{DIR} +$$
$$\overline{E12} * \overline{IORQ} * \overline{ZRD} * DIR * \overline{X11}$$

Aktivt låg då BUS1 eller BUS2 aktiveras. I övrigt lika med $\overline{DSTB\text{-}0}$.

4.5 PAL 11E

Uppgift: Att generera "interrupt acknowledge" till rätt enhet samt att leverera kvittenssignal till CPU:n vid I/O-hantering.

Typ: PAL 16R6



Insignaler:

PCLK Periferiklocka (4MHz) från klockgenerering

R/$\overline{W}$ Läs/skrivstrob från CPU.

$\overline{IOE}$ SCC:ns "hold"-indikering från PAL10C.

A2 Adress A2 från CPU.

A1 Adress A1 från CPU.

FC2 Funktionskod 2 från CPU.

FC1 Funktionskod 1 från CPU.

FC0 Funktionskod 0 från CPU.

GDS Datastrob från PAL 12E.

OE "Output Enable", alltid aktiv.

Utsignaler:

$\overline{NMIACK}$ = FC0 * FC1 * FC2 * A1 * A2 * $\overline{GDS}$

NMIACK är kvittensen på att CPU:n accepterat en interruptbegäran av högsta prioritet.

Signalen aktiveras då PAL:en avkodat funtionskoderna (alla höga indikerar "interrupt acknowledge") samt att adresserna A1-A3 indikerar nivå 7. (Obs då 68008 endast har interruptnivå 2,5 och 7 behöver här endast A1 och A2 avkodas).

GDS ger rätt klockning åt signalen.

$$\overline{MINT5} = \overline{GDS} * MINT5 * \overline{FC0} * FC1 * FC2 * A1 * A2 + \quad (1)$$
$$\overline{GDS} * \overline{IORQ} * \overline{MINT5} * \overline{IOC} + \quad (2)$$
$$\overline{GDS} * \overline{MINT5} * IOC \quad (3)$$

MINT5 är kvittenssignal till SCC och DART från CPU:n som accepterat en interruptbegäran. Signalen aktiveras efter avkodning av funktionskoderna och adresserna (1) samt hålls av (2 och 3)

$$\overline{MINT2} = \overline{GDS} * MINT2 * \overline{FC0} * FC1 * FC2 * \overline{A1} * A2 +$$
$$\overline{GDS} * \overline{IORQ} * \overline{MINT2} * \overline{IOC} +$$
$$GDS * \overline{MINT2} * IOC$$

"Interrupt acknowledge"-signal till CIO. I övrigt enligt ovan.

$$\overline{IORQ} = \overline{GDS} * IORQ * \overline{MINT2} * \overline{IOC} + \quad (1)$$
$$\overline{GDS} * IORQ * MINT5 * \overline{IOC} + \quad (2)$$
$$\overline{GDS} * \overline{IORQ} * FC0 * \overline{FC1} * IOC * \overline{IOE} + \quad (3)$$
$$GDS * \overline{IORQ} * IOC \quad (4)$$

"I/O Request"-strob till I/O enheter. Aktiv vid alla I/O-sekvenser (3) samt vid "interrupt acknowledge" nivå 2 (1) och nivå 5 (2). (4) avslutar stroben efter två periferiklockpulser.

$$\overline{IOC} = \overline{IORQ} + \quad (1)$$
$$\overline{MINT2} * IOC + \quad (2)$$
$$\overline{MINT5} * IOC \quad (3)$$

Används intern i PAL-kretsen för klockning och avslut samt vid generering av BGACK i 18C/3 som ej kan aktiveras då signalen är aktiv. Följer normalt en klockpuls efter IORQ (1), men aktiveras en klockpuls före IORQ vid "interrupt acknowledge" nivå 2 (2) och nivå 5 (3).

$$\overline{ZRD} = R/\overline{W} * \overline{GDS} * IORQ * FC0 * \overline{FC1} * IOC * \overline{IOE} + \quad (1)$$
$$R/\overline{W} * \overline{GDS} * \overline{ZRD} * IOC \quad (2)$$

Lässtrob till videoenhet, bussinterface och DART. Klockas samtidigt med IORQ-signalen då dataarea aktiveras samt CPU:ns läs/skrivsignal i läsläge (1).(2) håller signalen ytterligare en klockpuls.

$$\overline{ZWR} = \overline{R/W} * \overline{IORQ}$$

Skrivstrob från CPU:n till ovan omtalade enheter.
Aktiv då CPU:ns läs/skriv-strob i läge skriv samt "I/O Request" aktiv.

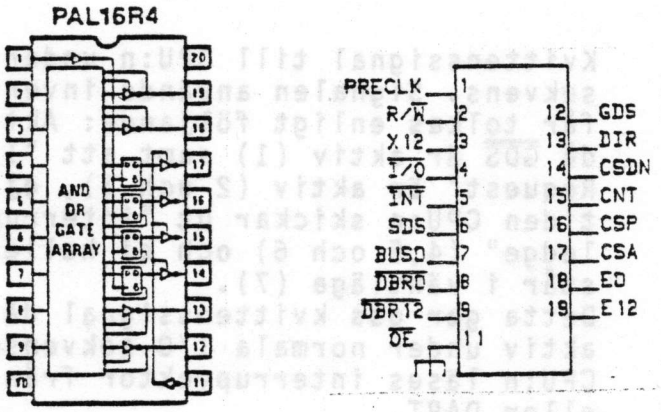
$\overline{ACK}$ = GDS + (1)
R/$\overline{W}$ * IORQ + (2)
R/$\overline{W}$ * IORQ * $\overline{IOC}$ + (3)
R/$\overline{W}$ * MINT2 * MINT5 * $\overline{FC0}$ * $\overline{IOC}$ + (4)
IORQ * MINT2 * MINT5 * $\overline{FC0}$ + (5)
IORQ * $\overline{FC0}$ * FC1 * FC2 * IOC + (6)
IOE * $\overline{FC1}$ (7)

Kvittenssignal till CPU:n under en I/O-sekvens. Signalen används inverterad och får tolkas enligt följande: Aktiv endast då $\overline{GDS}$ är aktiv (1) samt att "I/O Request" är aktiv (2 och 3), ej under tiden CPU:n skickar ut "interrupt acknowledge" (4,5 och 6) och ej heller då SCC:n står i väntläge (7).
Detta ger oss kvittenssignal då IORQ är aktiv under normala I/O-sekvenser samt då CPU:n läses interrupvektor från CIO, SCC eller DART.

4.6 PAL 12E

Uppgift: Genererar styrsignaler och hanterar "timing" mot bussinterface.

Typ: PAL 16R4



Insignaler:

PRECLK	Från klockgenerering (8MHz), samma fas som CPU-klocka.
R/$\overline{W}$	Läs/skrivstrob från CPU.
X12	Adress X12 från MAC eller DMA - MAP.
$\overline{I/O}$	Indikerar I/O-områdena från 22F.
$\overline{INT}$	"Interrupt acknowledge" från PAL 5D.
SDS	Datastrob synkroniserad med PCLK.
BUSO	Indikerar att det addresserade kortet finns på BUSOI eller BUSOX. Från 3a på busskort.
$\overline{DBRO}$	Indikerar att DMA0 fått buss - kontrollen.
$\overline{DBR12}$	Indikerar att DMA1 eller DMA2 fått busskontrollen.
$\overline{OE}$	"Output Enable", alltid aktiv.

Utsignaler:

$$\overline{GDS} = SDS * DBR12 * DBRO * \overline{INT} + \quad (1)$$
$$SDS * DBR12 * DBRO * \overline{I/O} * X12 + \quad (2)$$
$$SDS * DBR12 * DBRO * I/O * CSDN \quad (3)$$

GDS är en datastrob som avges vid vissa tillfällen.

CPU:n måste ha busskontrollen, DBRO och DBR12 har hög nivå. Datastroben som är synkroniserad med periferiklockan PCLK ska vara aktiv. Den avges under en "interrupt acknowledge"-sekvens (1) och vid intern I/O, exv. då periferikretsarnas register skrives/läses (2) och vid extern I/O, expansionsbussen (3).

$$\overline{DIR} = \overline{DBRO} * R/\overline{W} + \quad (1)$$
$$\overline{DBR12} * R/\overline{W} + \quad (2)$$
$$DBRO * DBR12 * \overline{R/W} \quad (3)$$

DIR styr riktningen av dataportarna till busskorten. Låg nivå då DMA:n läser från någon av bussarna (1 och 2) samt då CPU:n skriver till bussarna. Observera att vid låg nivå är dataportarna riktade utåt. Att denna riktning gäller vid läsning i DMA-mod beror på att DMA:n i detta fall genererar lässtrob till minnet som data ska överföras från. DMA:n arbetar i sök-mod vilket gör att den läser ur minnet samtidigt som data skrivs till någon av bussarna. Vid hög nivå är riktningarna omkastade.

$$\overline{CSDN} = SDS * \overline{INT} + \quad (1)$$
$$SDS * I/O + \quad (2)$$
$$SDS * X12 + \quad (3)$$
$$CNT * \overline{CSDN} \quad (4)$$

Signalen CSDN används endast internt i PAL-kretsen.
Den går hög efter det att expasionskortet adresseras för att indikera att kortvalet är avslutat. Signalen återgår till låg nivå igen vid "interrupt acknowledge" (1) eller vid adressering utanför I/O-området (2) eller vid adressering av intern I/O (3). (4) ser till att signalen går hög efter avslutad kortadressering.

$$\overline{CSA} = SDS * \overline{I/O} * \overline{X12} * INT * \overline{CSDN} * CNT * DBR12 * DBRO$$

Då CSA är låg öppnas adressportarna på busskortet för kortvalsadresser. För att portarna ska aktiveras krävs att datast-roben SDS är aktiv, extern I/O adresseras ($\overline{I/O} * \overline{X12}$),ej "interrupt acknowledge", ännu inte dags att avsluta sekvensen ($\overline{CSDN}$*CNT) samt att någon DMA ej har busskontrollen.

$$\overline{CSP} = CSP * \overline{CSA} * CNT$$

Kort kortvalsstrob till bussinterfacen då $\overline{CSA}$ är aktiv dvs. kortadress finns till-gänglig.

Stroben avslutas efter en klockpuls av att kontrollera den egna nivån och återkommer ej under sekvens p.g.a. CNT.

$\overline{CNT} = \overline{CSP}$

Kort puls som avslutar en kortadressering. Samma timing som CSP men fördröjd en klockpuls. Används intern i PAL-kretsen.

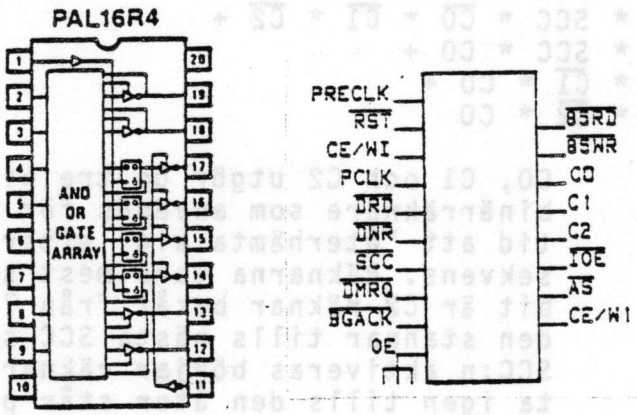
$\overline{EO} = BUSO * \overline{I/O} * \overline{X12} * CSDN * INT * DBR12 * DBRO * DBR12 * \overline{DBRO} * I/O$

Aktiverar den halva av busskortet som hanterar BUS1 och BUS2. Identiskt med ovanstående med undantag för BUSO-signalen som ska vara inaktiv dvs. låg, samt att signalen aktiveras och hålls då DMA1 eller DMA2 har busskontrollen.

4.7 PAL 10C

Uppgift: Att generera en riktig "timing" till SCC och vid skriv- och lässtrobar till CIO.

Typ: PAL 16R4



Insignaler:

PRECLK	Från klockgenerering (8MHz), samma fas som CPU-klocka.
$\overline{RST}$	Resetsignal från knapp eller CPU.
CE/WI	(Chip Enable/Wait In) Enable-signalen till DMA1.
PCLK	Periferiklocka från klockgenerering.
$\overline{DRD}$	Lässtrob från DMA eller PAL 6E vid CPU-mod.
$\overline{DWR}$	Skrivstrob från DMA eller PAL 6E vid CPU-mod.
$\overline{SCC}$	Enablesignal till SCC:n från I/O-hantering.
$\overline{DMRQ}$	"Memory Request"-signal från DMA.
$\overline{BGACK}$	"Bus Grant Acknowledge" från 18 C/3.
$\overline{OE}$	"Output Enable", alltid aktiv.

Utsignaler:

$$\overline{C2} = PCLK * \overline{C2} +$$
$$\overline{PCLK} * DRD * DWR * BGACK * \overline{C0} * \overline{C1} * \overline{C2} +$$
$$\overline{PCLK} * SCC * \overline{C0} * \overline{C1} * \overline{C2} * +$$
$$\overline{PCLK} * \overline{C0} * \overline{C1} * \overline{C2} +$$
$$\overline{PCLK} * \overline{C0} * \overline{C2} * +$$
$$PCLK * C1 * \overline{C2}$$

$$\overline{C1} = \overline{PCLK} * \overline{C1} + \overline{PCLK} * DRD * \overline{DWR} * \overline{BGACK} * \overline{C0} * \overline{C1} * \overline{C2} + \overline{PCLK} * SCC * \overline{C0} * \overline{C1} * \overline{C2} + \overline{PCLK} * \overline{C0} * \overline{C1} + \overline{PCLK} * C0 * \overline{C1}$$

$$\overline{C0} = \overline{PCLK} * \overline{C0} + \overline{PCLK} * DRD * DWR * BGACK * \overline{C0} * \overline{C1} * \overline{C2} + \overline{PCLK} * DRD * \overline{DWR} * \overline{C0} + \overline{PCLK} * SCC * \overline{C0} * \overline{C1} * \overline{C2} + \overline{PCLK} * SCC * C0 + \overline{PCLK} * \overline{C1} * C0 + PCLK * \overline{C2} * C0$$

C0, C1 och C2 utgör de tre bitarna i en binärräknare som används för att ge SCC:n tid att "återhämta sig" efter en utförd sekvens. Räknarna vars mest signifikanta bit är C2 räknar bakåt från 7 till 0 där den stannar tills nästa SCC-sekvens. Då SCC:n aktiveras börjar räknaren att arbeta igen tills den åter står på 0. Om SCC:n skulle aktiveras innan räknare nått 0, bromsas sekvensen tills detta värde är uppnått.

$$\overline{85RD} = \overline{BGACK} * \overline{DRD} * C0 * C1 * C2 + \quad (1)$$
$$BGACK * \overline{DRD} + \quad (2)$$
$$\overline{RST} \quad (3)$$

$\overline{85RD}$ är lässtrob till SCC och CIO. Den aktiveras då DMA:n har busskontrollen och gör en läsning samt då räknarna aktiverats till värdet 7.

Observera att detta endast gäller SCC:n då DMA:n ej arbetar mot CIO (2). Då CPU:n läses från SCC:n eller CIO (2). Signalen aktiveras också vid reset då den tillsammans med skrivstroben ger reset till SCC och CIO då de båda är aktiva samtidigt.

$$\overline{85WR} = \overline{BGACK} * \overline{DWR} * C0 * C1 * C2 + BGACK * \overline{DWR} + \overline{RST}$$

Signalen aktiveras vid skrivning. I övrigt samma som ovan.

$$\overline{CE/W1} = \overline{CE/W1} + \quad (1)$$
$$\overline{BGACK} * \overline{SCC} * \overline{C0} + \quad (2)$$
$$\overline{BGACK} * \overline{SCC} * \overline{C1} + \quad (3)$$
$$\overline{BGACK} * \overline{SCC} * \overline{C2} \quad (4)$$

Då CPU:n är aktiv genererar signalen "Chip Enable" till DMA1 (1). Då DMA1 har busskontrollen levereras "Wait" till den så länge räknaren ej hunnit komma till läge 7 (2,3 och 4).

$\overline{AS} = \overline{DMRQ}$ (Om $\overline{BGACK}$)

Adresstrob genereras i DMA-mod av "DMA Memory Request".

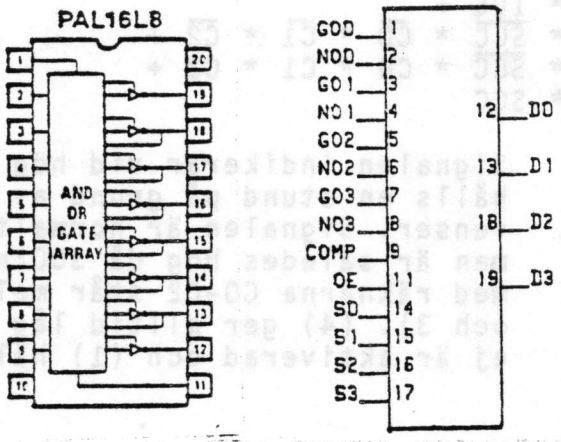
$$\overline{IOE} = \overline{PCLK} * \overline{\overline{IOE}} + \quad (1)$$
$$PCLK * \overline{SCC} * \overline{C0} * \overline{C1} * \overline{C2} + \quad (2)$$
$$PCLK * \overline{SCC} * C0 * C1 * C2 + \quad (3)$$
$$PCLK * SCC \quad (4)$$

Signalen indikerar vid hög nivå att SCC:n hålls en stund på grund av för täta sekvenser. Signalen är normalt alltid låg men är således hög då SCC:n är aktiverad med räknarna C0-C2 står mellan 1 och 6 (2 och 3). (4) ger alltid låg nivå då SCC:n ej är aktiverad och (1) håller signalen.

4.8 PAL 1N, 1M, 1T, 2T

Uppgift: Videoport och "mixer" vid utläsning av data från "movern".

Typ: PAL 16L8



OBS!
Några signalnamn kan ej lämnas då samma PAL-innehåll återfinns i fyra identiska PAL-kretsar men med olika inoch ut-signaler. Signaltypen är dock identisk och beskrivs allmänt här nedan.

Insignaler:

GOO-3 Data från register som innehåller "gammalt" ord.

NOO-3 Data direkt från cirkelskiftare, innehåller "nytt" ord.

SO-3 Muxsignal från muxtabellen.

DO-3 Data ut till grafikminnet.

COMP Från flaggregistret, kan invertera alla pixels.

OE "Output Enable", aktiverar utdata.

Utsignaler:

$$\overline{DO} = \overline{COMP} * \overline{SO} * GOO \quad (Om\ \overline{OE})$$
$$\overline{COMP} * SO * NOO$$
$$COMP * \overline{SO} * GOO$$
$$COMP * SO * NOO$$

Utsignalen DO lämnar en databit till grafikminnet i "mover"-mod. Den styrs av data G00 eller N00 beroende på läget av S0 som fungerar som muxsignal och kopplar om mellan G00 och N00. Med hjälp av COMP kan alla utgångar inverteras samtidigt. Sammanfattningsvis utgör varje PAL-krets fyra st 2-1 muxar med separata muxsignaler och gemensam invertering. Övriga utgångar identisk funktion.

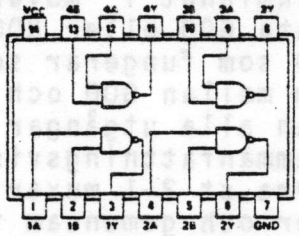
5 Kretsbeskrivning

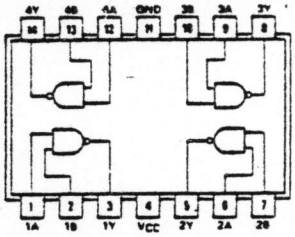
5.1 MSI-kretsar

QUADRUPLE 2-INPUT
POSITIVE-NAND GATES

00

positive logic:
$Y = \overline{AB}$





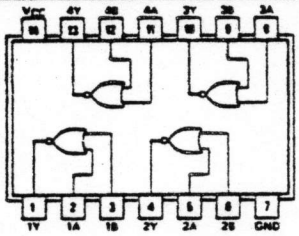
SN5400 (J) SN7400 (J, N)
SN54H00 (J) SN74H00 (J, N)
SN54LS00 (J, W) SN74LS00 (J, N)
SN54S00 (J, W) SN74S00 (J, N)
SN54ALS00A (J) SN74ALS00A (N)
PLANNED PRODUCT: SN54AS00 (J) SN74AS00 (N)

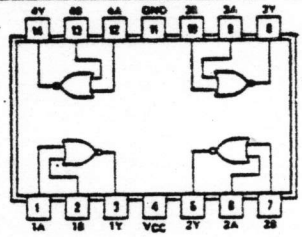
SN5400 (W)
SN54H00 (W)

QUADRUPLE 2-INPUT
POSITIVE-NOR GATES

02

positive logic:
$Y = \overline{A+B}$



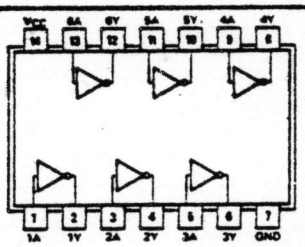


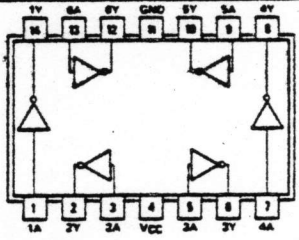
SN5402 (J) SN7402 (J, N)
SN54LS02 (J, W) SN74LS02 (J, N)
SN54S02 (J, W) SN74S02 (J, N)
SN54ALS02 (J) SN74ALS02 (N)
PLANNED PRODUCT: SN54AS02 (J) SN74AS02 (N)

HEX INVERTERS

04

positive logic:
$Y = \overline{A}$





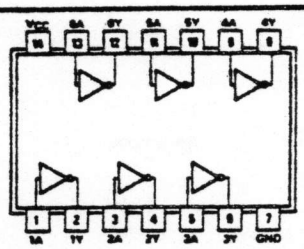
SN5404 (J) SN7404 (J, N)
SN54H04 (J) SN74H04 (J, N)
SN54LS04 (J, W) SN74LS04 (J, N)
SN54S04 (J, W) SN74S04 (J, N)
SN54ALS04 (J) SN74ALS504 (N)

SN5404 (W)
SN54H04 (W)

HEX INVERTER BUFFERS/DRIVERS
WITH OPEN-COLLECTOR
HIGH-VOLTAGE OUTPUTS

06

positive logic:
$Y = \overline{A}$

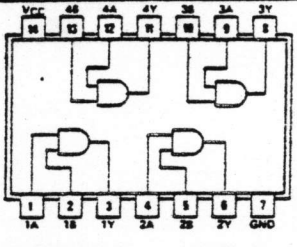


SN5406 (J, W) SN7406 (J, N)
SN54LS06 (J, W) SN74LS06 (J, N)

QUADRUPLE 2-INPUT
POSITIVE-AND GATES

08

positive logic:
$Y = AB$

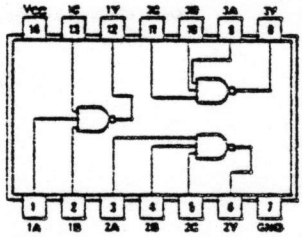


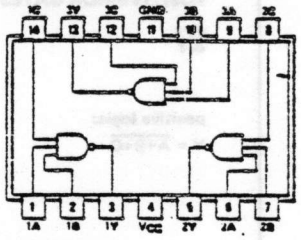
SN5408 (J, W) SN7408 (J, N)
SN54LS08 (J, W) SN74LS08 (J, N)
SN54S08 (J, W) SN74S08 (J, N)
SN54ALS08 (J) SN74ALS08 (N)
PLANNED PRODUCT: SN54AS08 (J) SN74AS08 (N)

TRIPLE 3-INPUT POSITIVE-NAND GATES

10

positive logic:
$Y = \overline{ABC}$





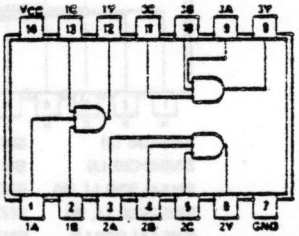
SN5410 (J) SN7410 (J, N)
SN54H10 (J) SN74H10 (J, N)
SN54LS10 (J, W) SN74LS10 (J, N)
SN54S10 (J, W) SN74S10 (J, N)
SN54ALS10 (J) SN74ALS10 (N)
PLANNED PRODUCT: SN54AS10 (J) SN74AS10 (N)

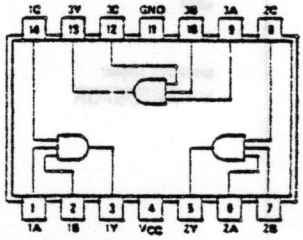
SN5410 (W)
SN54H10 (W)

TRIPLE 3-INPUT POSITIVE-AND GATES

11

positive logic:
$Y = ABC$





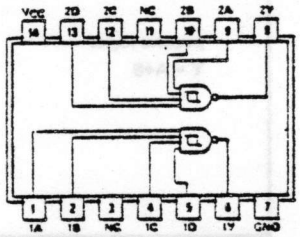
SN54H11 (J) SN74H11 (J, N)
SN54LS11 (J, W) SN74LS11 (J, N)
SN54S11 (J, W) SN74S11 (J, N)
SN54ALS11 (J) SN74ALS11 (N)
PLANNED PRODUCT: SN54AS11 (J) SN74AS11 (N)

SN54H11 (W)

DUAL 4-INPUT POSITIVE-NAND SCHMITT TRIGGERS

13

positive logic:
$Y = \overline{ABCD}$



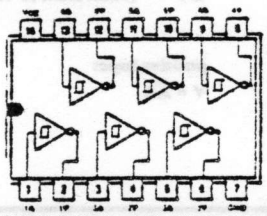
SN5413 (J, W) SN7413 (J, N)
SN54LS13 (J, W) SN74LS13 (J, N)

NC—No internal connection

SCHMITT TRIGGER INVERTER WITH TOTEM POLE OUTPUT

19

positive logic:
$Y = \overline{A}$

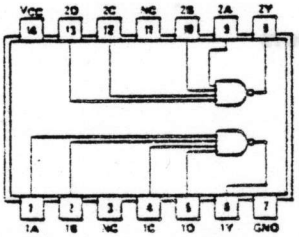


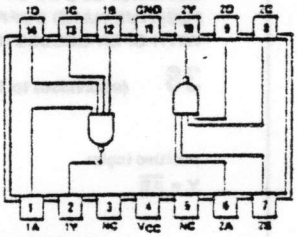
SN54LS19 (J or W) SN74LS19 (J or N)

DUAL 4-INPUT POSITIVE-NAND GATES

20

positive logic:
$Y = \overline{ABCD}$





SN5420 (J) SN7420 (J, N)
SN54H20 (J) SN74H20 (J, N)
SN54LS20 (J, W) SN74LS20 (J, N)
SN54S20 (J, W) SN74S20 (J, N)
SN54ALS20A (J) SN74ASLS20A (N)
PLANNED PRODUCT: SN54AS20 (J) SN74AS20 (N)

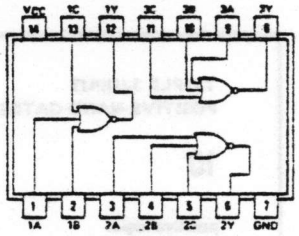
SN5420 (W)
SN54H20 (W)

NC—No internal connection

TRIPLE 3-INPUT
POSITIVE-NOR GATES

27

positive logic:
$Y = \overline{A+B+C}$

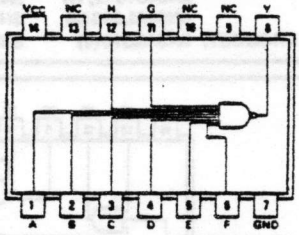


SN5427 (J, W) SN7427 (J, N)
SN54LS27 (J, W) SN74LS27 (J, N)
SN54ALS27 (J) SN74ALS27 (N)
PLANNED PRODUCT: SN54AS27 (J) SN74AS27 (N)

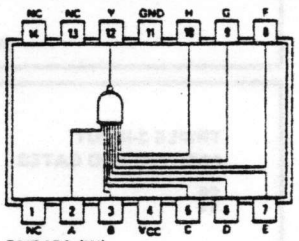
8-INPUT
POSITIVE-NAND GATES

30

positive logic:
$Y = \overline{ABCDEFGH}$



SN5430 (J) SN7430 (J, N)
SN54H30 (J) SN74H30 (J, N)
SN54LS30 (J, W) SN74LS30 (J, N)
SN54S30 (J, W) SN74S30 (J, N)
SN54ALS30 (J) SN74ALS30 (N)
PLANNED PRODUCT: SN54AS30 (J) SN74AS30 (N)



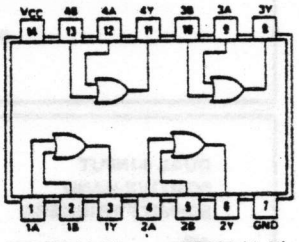
SN5430 (W)
SN54H30 (W)

NC—No internal connection

QUADRUPLE 2-INPUT
POSITIVE-OR GATES

32

positive logic:
$Y = A+B$

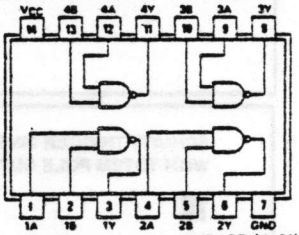


SN5432 (J, W) SN7432 (J, N)
SN54LS32 (J, W) SN74LS32 (J, N)
SN54S32 (J, W) SN74S32 (J, N)
SN54ALS32 (J) SN74ALS32 (N)
PLANNED PRODUCT: SN54AS32 (J) SN74AS32 (N)

QUADRUPLE 2-INPUT
POSITIVE-NAND BUFFERS

37 (equivalent to 1000)

positive logic:
$Y = \overline{AB}$

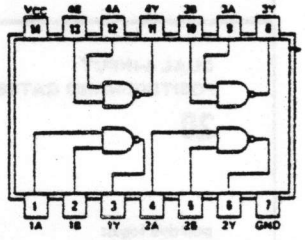


SN5437 (J, W) SN7437 (J, N)
SN54LS37 (J, W) SN74LS37 (J, N)
SN54S37 (J, W) SN74S37 (J, N)
SN54ALS37 (J) SN74ALS37 (N)

QUADRUPLE 2-INPUT
POSITIVE-NAND BUFFERS
WITH OPEN-COLLECTOR OUTPUTS

38 (equivalent to 1003)

positive logic:
$Y = \overline{AB}$

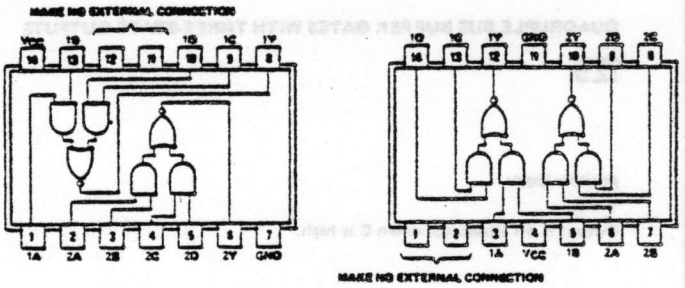


SN5438 (J, W) SN7438 (J, N)
SN54LS38 (J, W) SN74LS38 (J, N)
SN54S38 (J, W) SN74S38 (J, N)
SN54ALS38 (J) SN74ALS38 (N)

AND-OR-INVERT GATES

51

'S51, 'H51, 'S51
DUAL 2-WIDE 2-INPUT
positive logic:
$Y = \overline{AB+CD}$

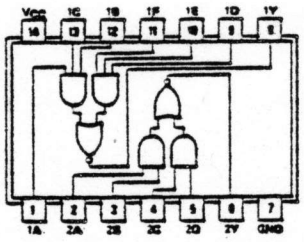


SN5451 (J)
SN54H51 (J)
SN54S51 (J, W)

SN7451 (J, N)
SN74H51 (J, N)
SN74S51 (J, N)

SN5451 (W)
SN54H51 (W)

'LS51
2-WIDE 3-INPUT,
2-WIDE 2-INPUT
positive logic:
$1Y = \overline{(1A \cdot 1B \cdot 1C)+(1D \cdot 1E \cdot 1F)}$
$2Y = \overline{(2A \cdot 2B)+(2C \cdot 2D)}$



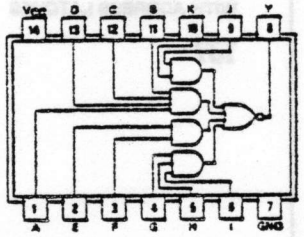
SN54LS51 (J, W)
SN74LS51 (J, N)

4-2-3-2 INPUT AND-OR-INVERT GATES

64 TOTEM-POLE OUTPUT

65 OPEN-COLLECTOR OUTPUT

positive logic: $\overline{Y = ABCD+EF+GHI+JK}$



SN54S64 (J, W)
SN54S65 (J, W)

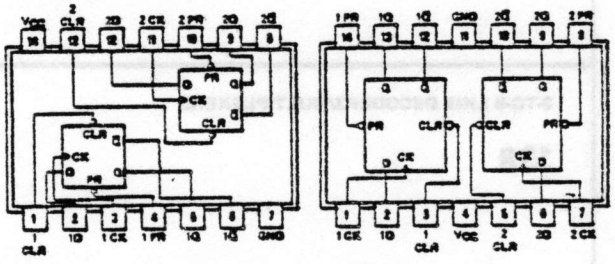
SN74S64 (J, N)
SN74S65 (J, N)

DUAL D-TYPE POSITIVE-EDGE-TRIGGERED FLIP-FLOPS WITH PRESET AND CLEAR

74

FUNCTION TABLE

INPUTS				OUTPUTS	
PRESET	CLEAR	CLOCK	D	Q	$\bar{Q}$
L	H	X	X	H	L
H	L	X	X	L	H
L	L	X	X	H*	H*
H	H	↑	H	H	L
H	H	↑	L	L	H
H	H	L	X	Q_0	$\bar{Q}_0$



SN5474 (J)
SN54H74 (J)
SN54LS74A (J, W)
SN54S74 (J, W)
SN54ALS74 (J)

SN7474 (J, N)
SN74H74 (J, N)
SN74LS74A (J, N)
SN74S74 (J, N)
SN74ALS74 (N)

SN5474 (W)
SN54H74 (W)

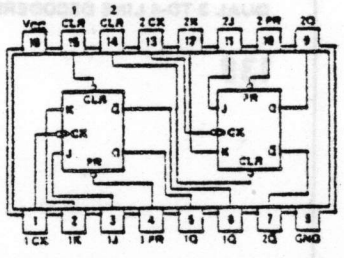
PLANNED PRODUCT: SN54AS74 (J)
SN74AS74 (N)

DUAL J-K NEGATIVE-EDGE-TRIGGERED FLIP-FLOPS WITH PRESET AND CLEAR

112

FUNCTION TABLE

INPUTS					OUTPUTS	
PRESET	CLEAR	CLOCK	J	K	Q	$\bar{Q}$
L	H	X	X	X	H	L
H	L	X	X	X	L	H
L	L	X	X	X	H*	H*
H	H	↓	L	L	Q_0	$\bar{Q}_0$
H	H	↓	H	L	H	L
H	H	↓	L	H	L	H
H	H	↓	H	H	TOGGLE	
H	H	H	X	X	Q_0	$\bar{Q}_0$



SN54LS112A (J, W)
SN54S112 (J, W)
SN54ALS112 (J)

SN74LS112A (J, N)
SN74S112 (J, N)
SN74ALS112 (N)

PLANNED PRODUCT: SN54AS112 (J)
SN74AS112 (N)

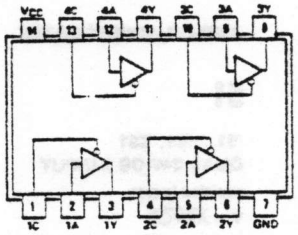
QUADRUPLE BUS BUFFER GATES WITH THREE-STATE OUTPUTS

125

positive logic:

Y = A

Output is off (disabled) when C is high.



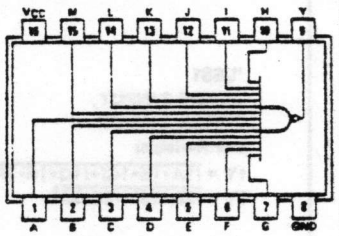
SN54125 (J, W) SN74125 (J, N)
SN54LS125A (J, W) SN74LS125A (J, N)

13-INPUT POSITIVE-NAND GATES

133

positive logic:

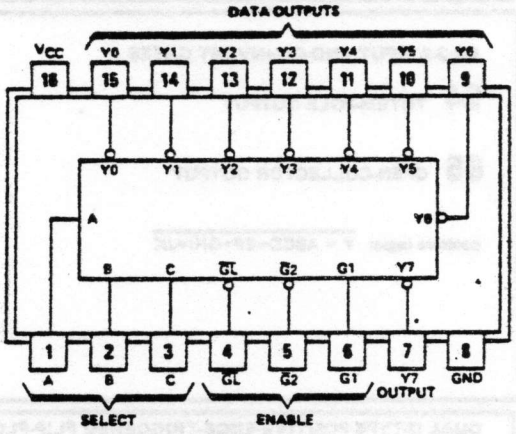
$Y = \overline{ABCDEFGHIJKLM}$



SN54S193 (J, W) SN74S133 (J, N)
SN54ALS133 (J) SN74ALS133 (N)

3-LINE TO 8-LINE DECODER/DEMULTIPLEXER WITH ADDRESS LATCHES

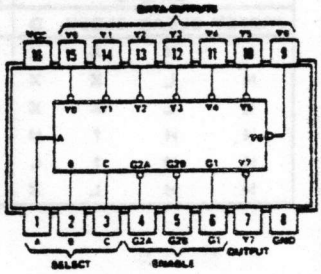
137



SN54LS137 (J or W) SN74LS137 (J or N)
SN54ALS137 (J) SN74ALS137 (J or N)

3-TO-8 LINE DECODERS/MULTIPLEXERS

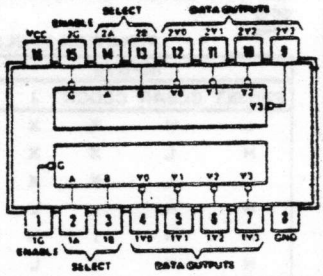
138



SN54LS138 (J, W) SN74LS138 (J, N)
SN54S138 (J, W) SN74S138 (J, N)
PLANNED PRODUCT: SN54ALS138 (J, W) SN74ALS138 (N)

DUAL 2-TO-4 LINE DECODERS/MULTIPLEXERS

139



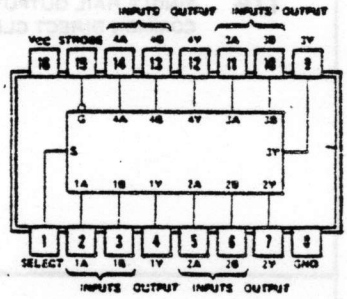
SN54LS139A (J, W) SN74LS139A (J, N)
SN54S139 (J, W) SN74S139 (J, N)
PLANNED PRODUCT: SN54ALS139 (J) SN74ALS139 (N)

QUAD 2-TO 1-LINE DATA SELECTORS/MULTIPLEXERS

157 NONINVERTED DATA OUTPUTS

158 INVERTED DATA OUTPUTS

See page 7-169



	SN54ALS157 (J, W)	SN74ALS157 (N)
	SN54LS158 (J, W)	SN74LS158 (J, N)
	SN54S158 (J, W)	SN74S158 (J, N)
	SN54ALS158 (J, W)	SN74ALS158 (N)
PLANNED PRODUCTS:	SN54AS157 (J)	SN74AS157 (N)
	SN54AS158 (J)	SN74AS158 (N)

SYNCHRONOUS 4-BIT COUNTERS

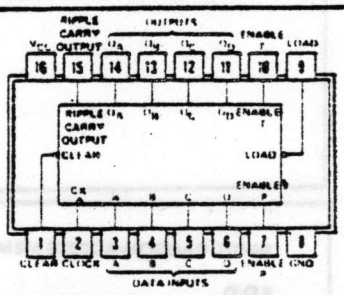
160 DECADE, DIRECT CLEAR

161 BINARY, DIRECT CLEAR

162 DECADE, SYNCHRONOUS CLEAR

163 BINARY, SYNCHRONOUS CLEAR

See page 7-177



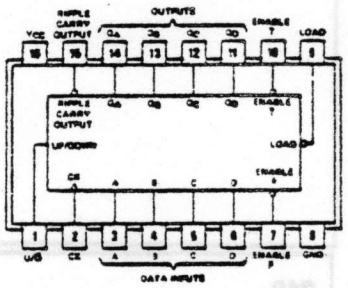
	SN54160 (J, W)	SN74160 (J, N)
	SN54LS160A (J, W)	SN74LS160A (J, N)
	SN54161 (J, W)	SN74161 (J, N)
	SN54LS161A (J, W)	SN74LS161A (J, N)
	SN54162 (J, W)	SN74162 (J, N)
	SN54LS162A (J, W)	SN74LS162A (J, N)
	SN54S162 (J, W)	SN74S162 (J, N)
	SN54163 (J, W)	SN74163 (J, N)
	SN54LS163A (J, W)	SN74LS163A (J, N)
	SN54S163 (J, W)	SN74S163 (J, N)
	SN54ALS160 (J)	SN74ALS160 (N)
	SN54ALS161 (J)	SN74ALS161 (N)
	SN54ALS162 (J)	SN74ALS162 (N)
	SN54ALS163 (J)	SN74ALS163 (N)
PLANNED PRODUCTS:	SN54AS160 (J)	SN74AS160 (N)
	SN54AS161 (J)	SN74AS161 (N)
	SN54AS162 (J)	SN74AS162 (N)
	SN54AS163 (J)	SN74AS163 (N)

4-BIT UP/DOWN SYNCHRONOUS COUNTERS

168 DECADE

169 BINARY

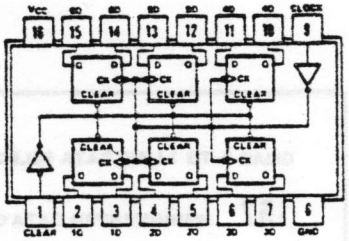
See page 7-212



	SN54S168 (J, W)	SN74S168 (J, N)
	SN54LS169B (J, W)	SN74LS169B (J, N)
	SN54S169 (J, W)	SN74S169 (J, N)
PLANNED PRODUCTS:	SN54ALS168 (J)	SN74ALS168 (N)
	SN54ALS169 (J)	SN74ALS169 (N)
PLANNED PRODUCTS:	SN54AS168 (J)	SN74AS168 (N)
	SN54AS169 (J)	SN74AS169 (N)

HEX D-TYPE FLIP-FLOPS

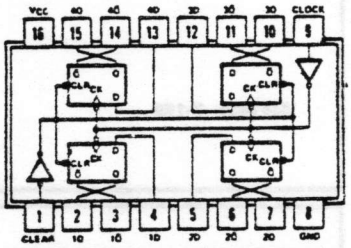
174 SINGLE RAIL OUTPUTS
COMMON DIRECT CLEAR



SN54174 (J, W) SN74174 (J, N)
SN54LS174 (J, W SN74LS174 (J, N)
SN54S174 (J, W) SN74S174 (J, N)
PLANNED PRODUCTS: SN54ALS174 (J, W) SN74ALS174 (N)
SN54AS174 (J) SN74AS174 (N)

QUAD D-TYPE FLIP-FLOPS

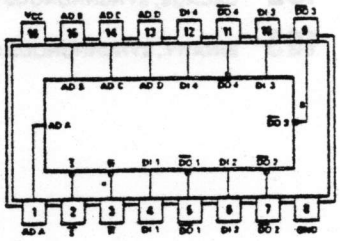
175 COMPLEMENTARY OUTPUTS
COMMON DIRECT CLEAR



SN54175 (J, W) SN74175 (J, N)
SN54LS175 (J, W) SN74LS175 (J, N)
SN54S175 (J, W) SN74S175 (J, N)
PLANNED PRODUCTS: SN54ALS175 (J, W) SN74ALS175 (N)
SN54AS175 (J) SN74AS175 (N)

64-BIT RANDOM-ACCESS MEMORIES

189 16 4-BIT WORDS
THREE-STATE OUTPUTS

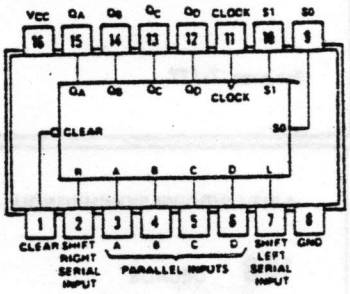


SN54S189 (J, W) SN74S189 (J, N)

See Bipolar Microcomputer Components Data Book, LCC5831

4-BIT BIDIRECTIONAL UNIVERSAL SHIFT REGISTERS

194



SN54194 (J, W) SN74194 (J, N)
SN54LS194A (J, W) SN74LS194A (J, N)
SN54S194 (J, W) SN74S194 (J, N)
PLANNED PRODUCTS: SN54AS194 (J) SN74AS194 (N)

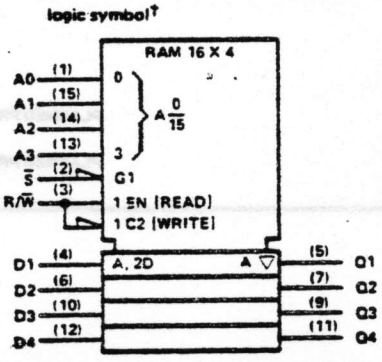
219

64-BIT RANDOM-ACCESS MEMORIES
(16 words of 4 bits each; three-state non-inverting output)
typical performance

TYPE	ADDRESS TIME	ENABLE TIME	POWER/ BIT
'LS219A	50 ns	35 ns	2.7 mW

SN54LS219A (J,FH) SN74LS219A (J,N,FN)

logic symbol†

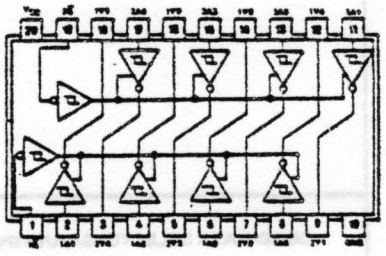


pin assignments

J, N PACKAGES				FH, FN PACKAGES			
1	A0	9	Q3	1	nc	11	nc
2	S̄	10	D3	2	A0	12	Q3
3	R/W̄	11	Q4	3	S̄	13	D3
4	D1	12	D4	4	R/W̄	14	Q4
5	Q1	13	A3	5	D1	15	D4
6	D2	14	A2	6	nc	16	nc
7	Q2	15	A1	7	Q1	17	A3
8	GND	16	V_{CC}	8	D2	18	A2
				9	Q2	19	A1
				10	GND	20	V_{CC}

OCTAL BUFFERS/LINE DRIVERS/LINE RECEIVERS

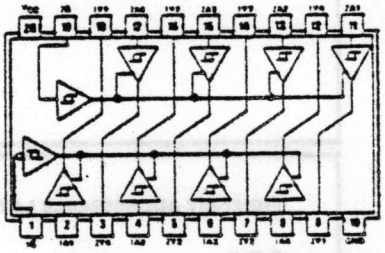
240 INVERTED 3-STATE OUTPUTS



	SN54LS240 (J)	SN74LS240 (J, N)
	SN54S240 (J)	SN74S240 (J, N)
	SN54ALS240 (J)	SN74ALS240 (N)
PLANNED PRODUCTS:	SN54AS240 (J)	SN74AS240 (N)

OCTAL BUFFERS/LINE DRIVERS/LINE RECEIVERS

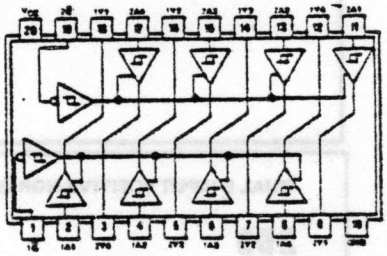
241 NONINVERTED 3-STATE OUTPUTS



	SN54LS241 (J)	SN74LS241 (J, N)
	SN54S241 (J)	SN74S241 (J, N)
	SN54ALS241 (J)	SN74ALS241 (N)
PLANNED PRODUCT:	SN54AS241 (J)	SN74AS241 (N)

OCTAL BUFFERS/LINE DRIVERS/LINE RECEIVERS

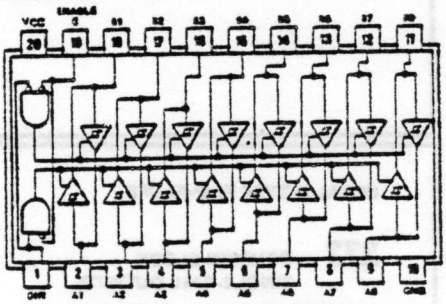
244 NONINVERTED 3-STATE OUTPUTS



	SN54LS244 (J)	SN74LS244 (J, N)
	SN54ALS244 (J)	SN74ALS244 (N)
	SN54S244 (J)	SN74S244 (J, N)
PLANNED PRODUCT:	SN54AS244 (J)	SN74AS244 (N)

OCTAL BUS TRANCEIVERS

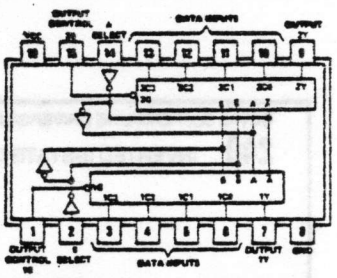
245 NONINVERTED 3-STATE OUTPUTS



	SN54LS245 (J)	SN74LS245 (J, N)
	SN54ALS245 (J)	SN74ALS245 (N)

DUAL DATA SELECTORS/MULTIPLEXERS

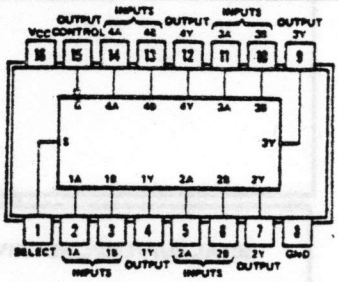
253 3-STATE OUTPUTS



SN54LS253 (J, W) SN74LS253 (N)
PLANNED PRODUCTS: SN54ALS253 (J) SN74ALS253 (N)
SN54AS253 (J) SN74AS253 (N)

QUAD DATA SELECTORS/MULTIPLEXERS

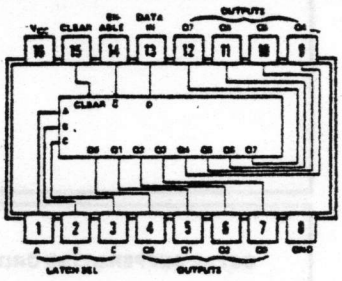
257 NONINVERTED 3-STATE OUTPUTS



SN54LS257B (J, W) SN74LS257B (J, N)
SN54S257 (J, W) SN74S257 (J, N)
PLANNED PRODUCT: SN54ALS257 (J, W) SN74ALS257 (N)

EIGHT-BIT ADDRESSABLE LATCHES

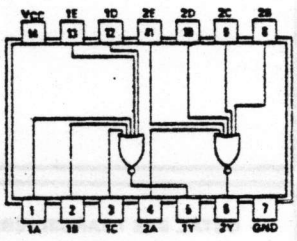
259



SN54259 (J, W) SN74259 (J, N)
SN54LS259 (J, W) SN74LS259 (J, N)
PLANNED PRODUCT: SN54ALS259 (J) SN74ALS259 (N)

DUAL 5-INPUT POSITIVE NOR GATES

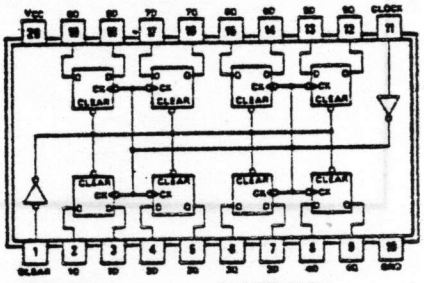
260



SN54S260 (J, W) SN74S260 (J, N)

OCTAL D-TYPE FLIP-FLOPS

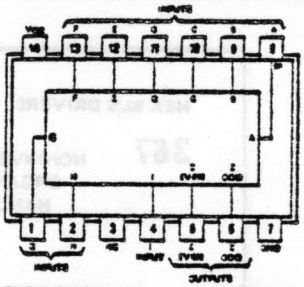
273 COMMON CLOCK
SINGLE-RAIL OUTPUTS



SN54273 (J) SN74273 (J, N)
SN54LS273 (J) SN74LS273 (J, N)
SN54ALS273 (J) SN74ALS273 (N)

9-BIT ODD/EVEN PARITY GENERATORS/CHECKERS

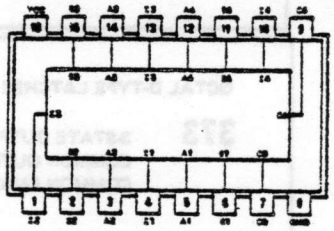
280 N-BIT CASCADEABLE



SN54LS280 (J, W) SN74LS280 (J, N)
SN54S280 (J, W) SN74S280 (J, N)
PLANNED PRODUCT: SN54AS280 (J) SN74AS280 (N)

4-BIT SINARY FULL ADDERS

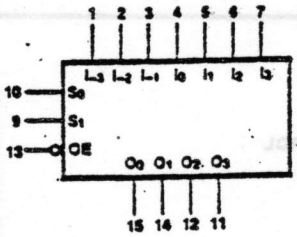
283

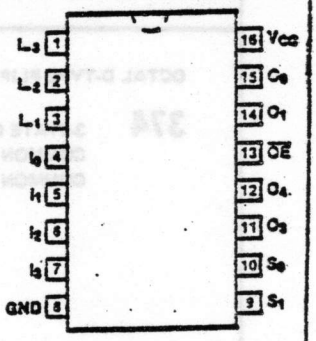


SN54283 (J, W) SN74283 (J, N)
SN54LS283 (J, W) SN74LS283 (J, N)
SN54S283 (J) SN74S283 (J, N)

Logic Symbol

350

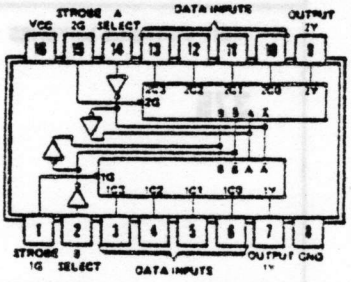




Vcc = Pin 16
GND = Pin 8

DUAL 4-LINE TO 1-LINE DATA SELECTORS/MULTIPLEXERS

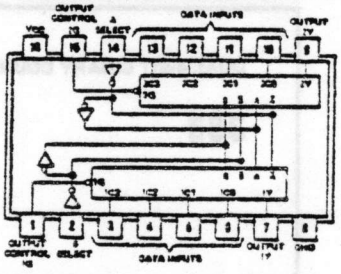
352 INVERTING VERSION OF 'LS153



SN54LS352 (J, W) SN74LS352 (J, N)
PLANNED PRODUCTS: SN54ALS352 (J) SN74ALS352 (N)
SN54AS352 (J) SN74AS352 (N)

DUAL 4-LINE TO 1-LINE DATA SELECTORS/MULTIPLEXERS

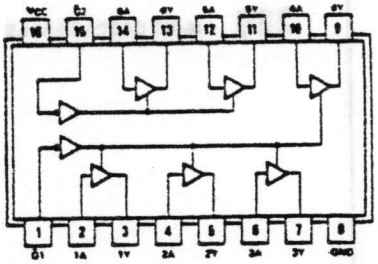
353 3-STATE OUTPUTS
INVERTING VERSION OF 'LS253



SN54LS353 (J, W) SN74LS353 (J, N)
PLANNED PRODUCTS: SN54ALS353 (J) SN74ALS353 (N)
SN54AS353 (J) SN74AS353 (N)

HEX BUS DRIVERS

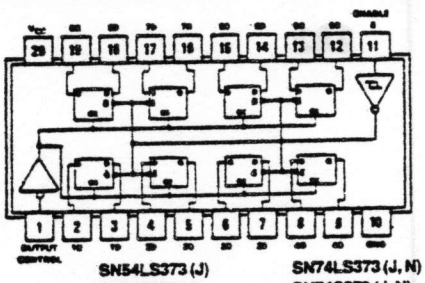
367 NONINVERTED 3-STATE OUTPUTS ORGANIZED TO FACILITATE HANDLING OF 4-BIT DATA



SN54367A (J, W) SN74367A (J, N)
SN54LS367A (J, W) SN74LS367A (J, N)

OCTAL D-TYPE LATCHES

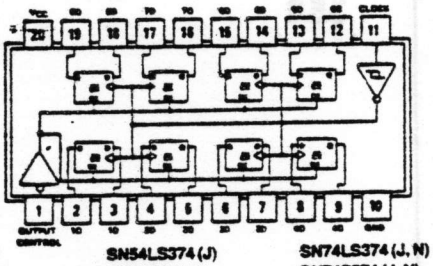
373 3-STATE OUTPUTS
COMMON OUTPUT CONTROL
COMMON ENABLE



SN54LS373 (J) SN74LS373 (J, N)
SN54S373 (J) SN74S373 (J, N)
PLANNED PRODUCTS: SN54ALS373 (J) SN74ALS373 (N)
SN54AS373 (J) SN74AS373 (N)

OCTAL D-TYPE FLIP-FLOPS

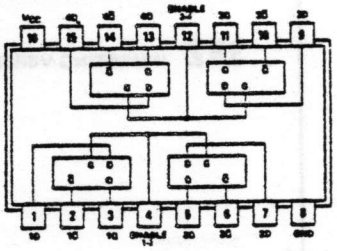
374 3-STATE OUTPUTS
COMMON OUTPUT CONTROL
COMMON CLOCK



SN54LS374 (J) SN74LS374 (J, N)
SN54S374 (J) SN74S374 (J, N)
PLANNED PRODUCTS: SN54ALS374 (J) SN74ALS374 (N)
SN54AS374 (J) SN74AS374A (N)

4-BIT BISTABLE LATCHES

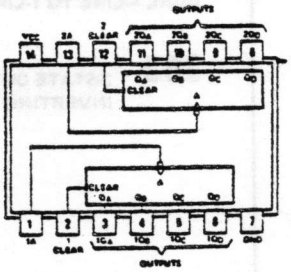
375



SN54LS375 (J, W) SN74LS375 (J, N)

DUAL 4-BIT BINARY COUNTERS

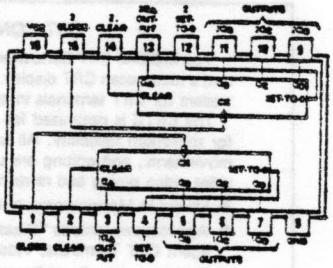
393



SN54393 (J, W) SN74393 (J, N)
SN54LS393 (J, W) SN74LS393 (J, N)

DUAL DECADE COUNTERS

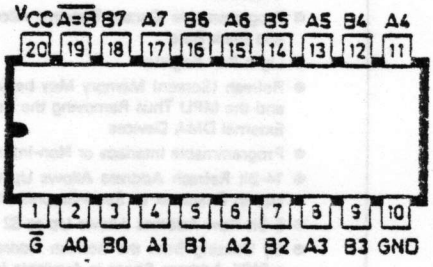
490



SN54490 (J, W) **SN74490 (J, N)**
SN54LS490 (J, W) **SN74LS490 (J, N)**

OCTAL COMPARATOR

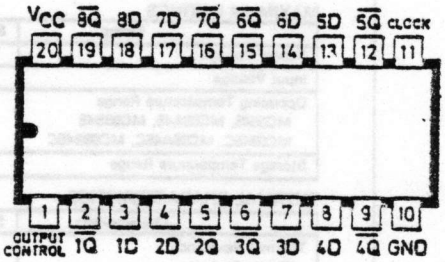
521



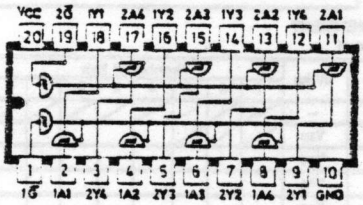
PLANNED PRODUCT: **SN54ALS521 (J, W)** **SN74ALS521 (N)**

OCTAL D-TYPE FLIP FLOPS – INVERTING

534 (INVERTED 374)



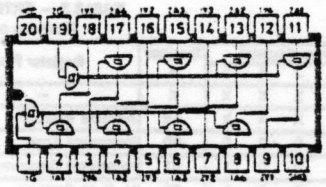
PLANNED PRODUCTS: **SN54ALS534 (J, W)** **SN74ALS534 (N)**
SN54AS534 (J) **SN74AS534 (N)**



OCTAL BUFFERS / LINE DRIVERS / LINE RECEIVERS

1240 INVERTED 3-STATE OUTPUTS

SN54ALS1240 (J) **SN74ALS1240 (N)**



OCTAL BUS TRANCEIVERS

1241 NON-INVERTED 3-STATE OUTPUTS

SN54ALS1241 (J) **SN74ALS1241 (N)**

5.2 LSI-kretsar

5.2.1 Övriga kretsar

MC6845

CRT CONTROLLER (CRTC)

The MC6845 CRT Controller performs the interface between an MPU and a raster-scan CRT display. It is intended for use in MPU-based controllers for CRT terminals in stand-alone or cluster configurations.

The CRTC is optimized for the hardware/software balance required for maximum flexibility. All keyboard functions, reads, writes, cursor movements, and editing are under processor control. The CRTC provides video timing and refresh memory addressing.

- Useful in Monochrome or Color CRT Applications
- Applications Include "Glass-Teletype," Smart, Programmable, Intelligent CRT Terminals; Video Games; Information Displays
- Alphanumeric, Semi-Graphic, and Full-Graphic Capability
- Fully Programmable Via Processor Data Bus. Timing May Be Generated for Almost Any Alphanumeric Screen Format, e.g., 80 × 24, 72 × 64, 132 × 20
- Single +5 V Supply
- M6800 Compatible Bus Interface
- TTL-Compatible Inputs and Outputs
- Start Address Register Provides Hardware Scroll (by Page, Line, or Character)
- Programmable Cursor Register Allows Control of Cursor Format and Blink Rate
- Light Pen Register
- Refresh (Screen) Memory May be Multiplexed Between the CRTC and the MPU Thus Removing the Requirements for Line Buffers or External DMA Devices
- Programmable Interlace or Non-Interlace Scan Modes
- 14-Bit Refresh Address Allows Up to 16K of Refresh Memory for Use in Character or Semi-Graphic Displays
- 5-Bit Row Address Allows Up to 32 Scan-Line Character Blocks
- By Utilizing Both the Refresh Addresses and the Row Addresses, a 512K Address Spece is Available for Use in Graphics Systems
- Refresh Addresses are Provided During Retrace, Allowing the CRTC to Provide Row Addresses to Refresh Dynamic RAMs
- Programmable Skew for Cursor and Display Enable (DE)
- Pin Compatible with the MC6836

MOS

(N-CHANNEL, SILICON-GATE)

CRT CONTROLLER (CRTC)

L SUFFIX
CERAMIC PACKAGE
CASE 715

S SUFFIX
CERDIP PACKAGE
CASE 734

P SUFFIX
PLASTIC PACKAGE
CASE 711

☆ = Package Suffix

MAXIMUM RATINGS

Rating	Symbol	Value	Unit
Supply Voltage	V_{CC}*	−0.3 to +7.0	V
Input Voltage	V_{in}*	−0.3 to +7.0	V
Operating Temperature Range MC6845, MC68A45, MC68B45 MC5845C, MC68A45C, MC68B45C	T_A	T_L to T_H 0 to 70 −40 to +85	°C
Storage Temperature Range	T_{stg}	−55 to +150	°C

THERMAL CHARACTERISTICS

Characteristic	Symbol	Value	Rating
Thermal Resistance Plastic Package Cerdip Package Ceramic Package	θ_{JA}	 100 60 50	°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. For proper operation it is recommended that V_{in} and V_{out} be constrained to the range $V_{SS} \le (V_{in}$ or $V_{out}) \le V_{CC}$.

FIGURE 1 — PIN ASSIGNMENTS

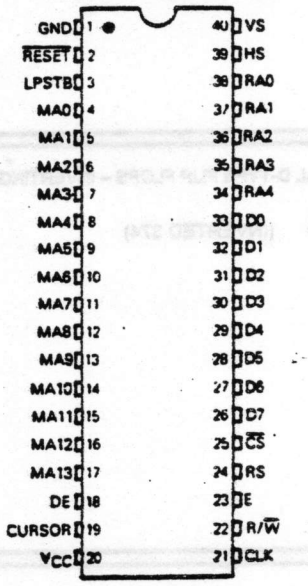


TABLE 2 — CRTC INTERNAL REGISTER ASSIGNMENT
(Features of the MC6845-1 have 1 subscript)

CS	RS	Address Register 4	3	2	1	0	Register #	Register File	Program Unit	Read	Write	Number of Bits 7	6	5	4	3	2	1	0
1	X	X	X	X	X	X	X	–	–	–	–								
0	0	X	X	X	X	X	AR	Address Register	–	No	Yes								
0	1	0	0	0	0	0	R0	Horizontal Total	Char.	No	Yes								
0	1	0	0	0	0	1	R1	Horizontal Displayed	Char.	No	Yes								
0	1	0	0	0	1	0	R2	H. Sync Position	Char.	No	Yes								
0	1	0	0	0	1	1	R3	Sync Width	–	No	Yes	V_1	V_1	V_1	V_1	H	H	H	H
0	1	0	0	1	0	0	R4	Vertical Total	Char. Row	No	Yes								
0	1	0	0	1	0	1	R5	V. Total Adjust	Scan Line	No	Yes								
0	1	0	0	1	1	0	R6	Vertical Displayed	Char. Row	No	Yes								
0	1	0	0	1	1	1	R7	V. Sync Position	Char. Row	No	Yes								
0	1	0	1	0	0	0	R8	Interlace Mode and Skew	Note 1	No	Yes	C_1	C_1	D_1	D_1			I	I
0	1	0	1	0	0	1	R9	Max Scan Line Address	Scan Line	No	Yes								
0	1	0	1	0	1	0	R10	Cursor Start	Scan Line	No	Yes		B	P			(Note 2)		
0	1	0	1	0	1	1	R11	Cursor End	Scan Line	No	Yes								
0	1	0	1	1	0	0	R12	Start Address (H)	–	Yes	Yes	0	0						
0	1	0	1	1	0	1	R13	Start Address (L)	–	Yes	Yes								
0	1	0	1	1	1	0	R14	Cursor (H)	–	Yes	Yes	0	0						
0	1	0	1	1	1	1	R15	Cursor (L)	–	Yes	Yes								
0	1	1	0	0	0	0	R16	Light Pen (H)	–	Yes	No	0	0						
0	1	1	0	0	0	1	R17	Light Pen (L)	–	Yes	No								

PIN DESCRIPTION

PROCESSOR INTERFACE

The CRTC interfaces to a processor bus on the bidirectional data bus (D0-D7) using $\overline{CS}$, RS, E, and R/$\overline{W}$ for control signals.

Data Bus (D0-D7) – The bidirectional data lines (D0-D7) allow data transfers between the internal CRTC register file and the processor. Data bus output drivers are high-impedance state until the processor performs a CRTC read operation.

Enable (E) – The Enable signal is a high-impedance TTL/MOS compatible input which enables the data bus input/output buffers and clocks data to and from the CRTC. This signal is usually derived from the processor clock. The high-to-low transition is the active edge.

Chip Select ($\overline{CS}$) – The CS line is a high-impedance TTL/MOS compatible input which selects the CRTC, when low, to read or write to the internal register file. This signal should only be active when there is a valid stable address being decoded from the processor.

Register Select (RS) – The RS line is a high-impedance TTL/MOS compatible input which selects either the address register (RS = "0") or one of the data register (RS = "1") of the internal register file.

Read/Write (R/$\overline{W}$) – The R/$\overline{W}$ line is a high-impedance TTL/MOS compatible input which determines whether the internal register file gets written or read. A write is defined as a low level.

CRT CONTROL

The CRTC provides horizontal sync (HS), vertical sync (VS), and display enable (DE) signals.

Vertical Sync (VS) and Horizontal Sync (HS) – These TTL-compatible outputs are active high signals which drive the monitor directly or are fed to the video processing circuitry to generate a composite video signal. The VS signal determines the vertical position of the displayed text while the HS signal determines the horizontal position of the displayed text.

Display Enable (DE) – This TTL-compatible output is an active high signal which indicates the CRTC is providing addressing in the active display area.

REFRESH MEMORY/CHARACTER GENERATOR ADDRESSING

The CRTC provides memory addresses (MA0-MA13) to scan the refresh RAM. Row addresses (RA0-RA4) are also provided for use with character generator ROMs. In a graphics system, both the memory addresses and the row addresses would be used to scan the refresh RAM. Both the memory addresses and the row addresses continue to run during vertical retrace thus allowing the CRTC to provide the refresh addresses required to refresh dynamic RAMs.

Refresh Memory Addresses (MA0-MA13) – These 14 outputs are used to refresh the CRT screen with pages of data located within a 16K block of refresh memory. These outputs are capable of driving one standard TTL load and 30 pF.

Row Addresses (RA0-RA4) – These five outputs from the internal row address counter are used to address the character generator ROM. These outputs are capable of driving one standard TTL load and 30 pF.

OTHER PINS

Cursor – This TTL-compatible output indicates a valid cursor address to external video processing logic. It is an active high signal.

Clock (CLK) – The CLK is a TTL/MOS-compatible inout used to synchronize all CRT functions except for the processor interface. An external dot counter is used to derive this signal which is usually the character rate in an alphanumeric CRT. The active transition is high-to-low.

Light Pen Strobe (LPSTB) – A low-to-high transition on this high-impedance TTL/MOS-compatible input latches the current Refresh Address in the light pen register. The latching of the refresh address is internally synchronized to the character clock (CLK).

V_{CC}, V_{SS} – These inputs supply +5 Vdc ±5% to the CRTC.

$\overline{RESET}$ – The $\overline{RESET}$ input is used to reset the CRTC. A low level on the $\overline{RESET}$ input forces the CRTC into the following state:

(a) All counters in the CRTC are cleared and the device stops the display operation.

(b) All the outputs are driven low.

(c) The control registers of the CRTC are not affected and remain unchanged.

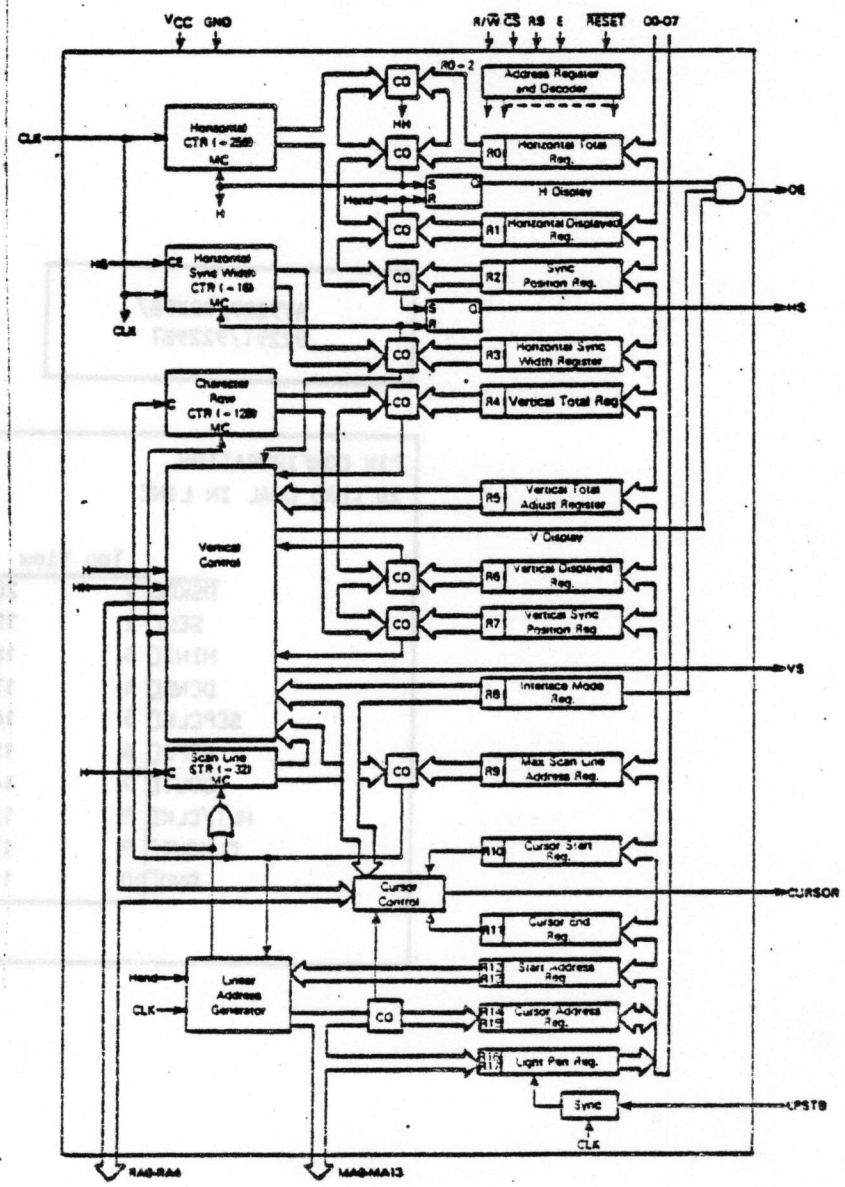
Functionality of $\overline{RESET}$ differs from that of other M6800 parts in the following functions:

(a) The $\overline{RESET}$ input and the LPSTB input are encoded as shown in Table 1.

(b) After $\overline{RESET}$ has gone low and (LPSTB = "0"), MA0-MA13 and RA0-RA4 will be driven low on the falling edge of CLK. $\overline{RESET}$ must remain low for at least one cycle of the character clock (CLK).

(c) The CRTC resumes the display operation immediately after the release of $\overline{RESET}$. DE is not active until after the first VS pulse occurs.

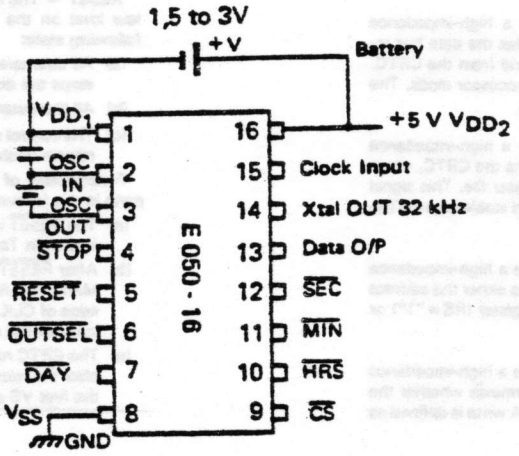
CRTC BLOCK DIAGRAM



E 050 - 16

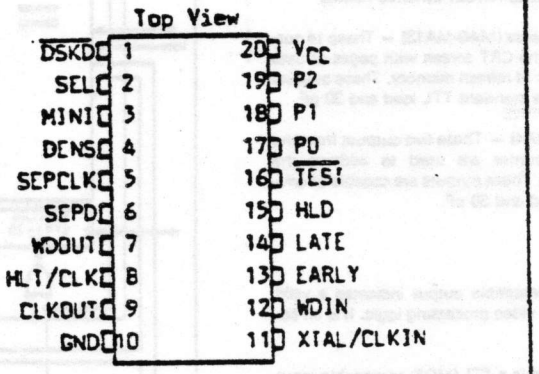
PACKAGE

16 Pin Dual in Line



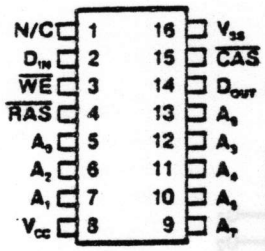
AY9229/9229B/
9229T/9229BT

PIN CONFIGURATION
20 LEAD DUAL IN LINE



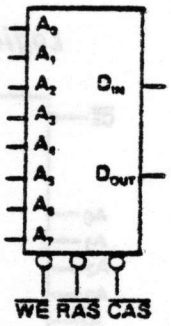
IMS2600 64K x 1 bit dynamic RAM

PIN CONFIGURATION

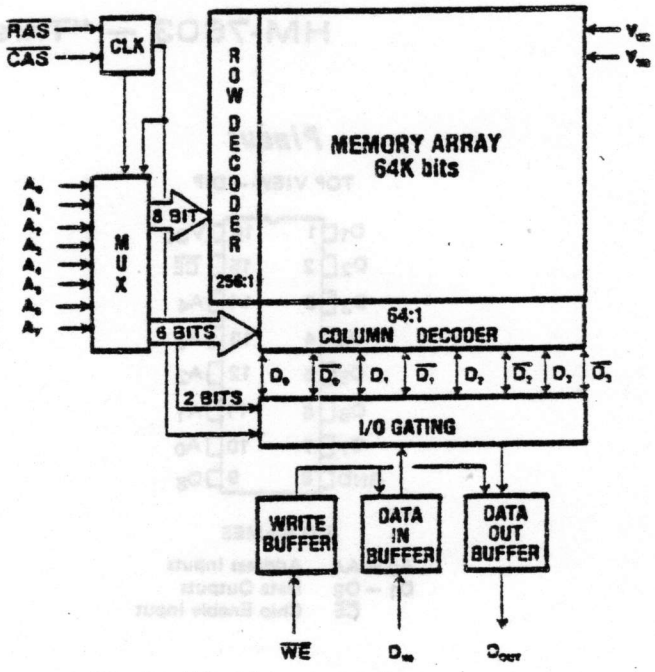


DIP

LOGIC SYMBOL



BLOCK DIAGRAM



PIN NAMES

A_0-A_7	ADDRESS INPUTS
$\overline{CAS}$	COLUMN ADDRESS STROBE
$\overline{RAS}$	ROW ADDRESS STROBE
D_{IN}	DATA IN
D_{OUT}	DATA OUT
$\overline{WE}$	WRITE ENABLE
V_{CC}	+5 VOLT SUPPLY INPUT
V_{SS}	GROUND

494 256-Bit Serial Electrically Erasable Programmable Memory

Block and Connection Diagrams

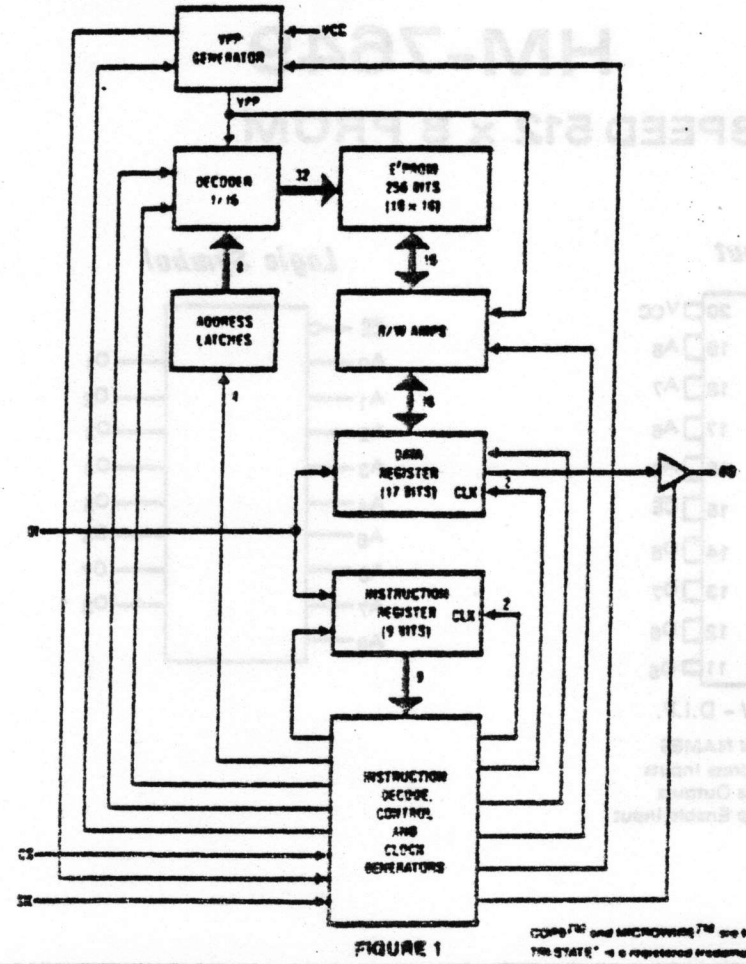


FIGURE 1

Dual-In-Line Package

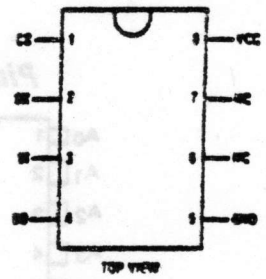


FIGURE 2

Pin Names

CS	Chip Select
SK	Serial Data Clock
DI	Serial Data Input
DO	Serial Data Output
VCC	Power Supply
GND	Ground

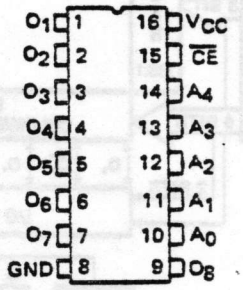
HM-7602/03

32 x 8 PROM

HM-7603 — "Three State" Outputs

Pinout

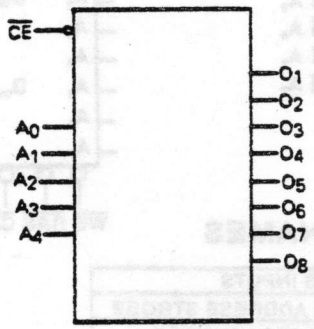
TOP VIEW — DIP



PIN NAMES

$A_0 - A_4$	Address Inputs
$O_1 - O_8$	Data Outputs
$\overline{CE}$	Chip Enable Input

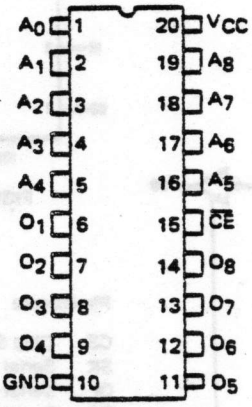
Logic Symbol



HM-7649

HIGH SPEED 512 x 8 PROM

Pinout

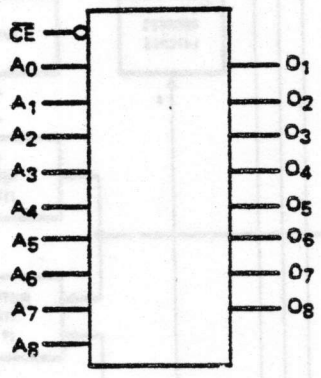


TOP VIEW - D.I.P.

PIN NAMES

A_0–A_8	Address Inputs
O_1–O_8	Data Outputs
$\overline{CE}$	Chip Enable Input

Logic Symbol



2732A
32K (4K x 8) UV ERASABLE PROM

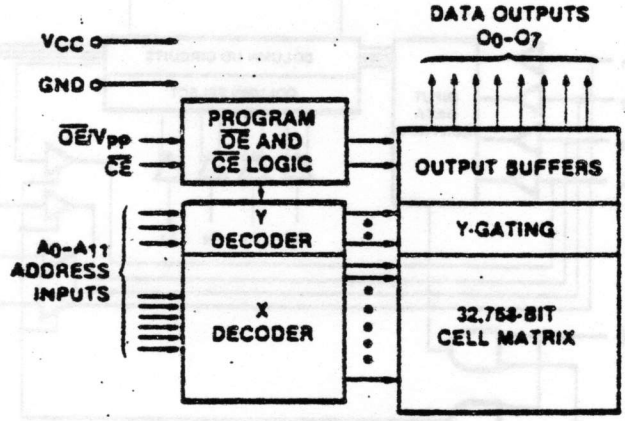


Figure 1. Block Diagram

PIN NAMES

A_0-A_{11}	ADDRESSES
$\overline{CE}$	CHIP ENABLE
$\overline{OE}/V_{PP}$	OUTPUT ENABLE/ V_{PP}
O_0-O_7	OUTPUTS

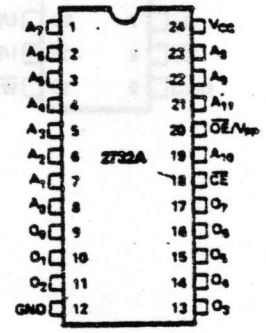


Figure 2. Pin Configuration

27128
128K (16K x 8) UV ERASABLE PROM

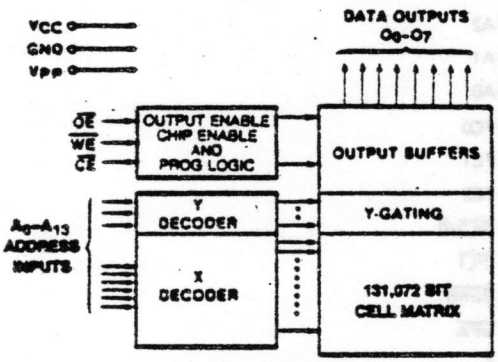


Figure 1. Block Diagram

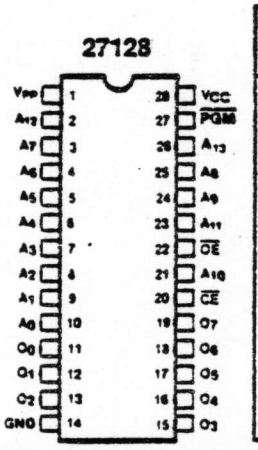
MODE SELECTION

Mode \ Pins	$\overline{CE}$ (20)	$\overline{OE}$ (22)	$\overline{PGM}$ (27)	A_9 (24)	V_{PP} (1)	V_{CC} (28)	Outputs (11-13, 15-19)
Read	V_{IL}	V_{IL}	V_{IH}	X	V_{CC}	V_{CC}	D_{OUT}
Output Disable	V_{IL}	V_{IH}	V_{IH}	X	V_{CC}	V_{CC}	High Z
Standby	V_{IH}	X	X	X	V_{CC}	V_{CC}	High Z
Program	V_{IL}	V_{IH}	V_{IL}	X	V_{PP}	V_{CC}	D_{IN}
Verify	V_{IL}	V_{IL}	V_{IH}	X	V_{PP}	V_{CC}	D_{OUT}
Program Inhibit	V_{IH}	X	X	X	V_{PP}	V_{CC}	High Z
Intelligent Identifier	V_{IL}	V_{IL}	V_{IH}	V_H	V_{CC}	V_{CC}	Code
Intelligent Programming	V_{IL}	V_{IH}	V_{IL}	X	V_{PP}	V_{CC}	D_{IN}

NOTES:
1. X can be V_{IH} or V_{IL}
2. V_H = 12.0V ± 0.5V

*HMOS is a patented process of Intel Corporation.

27256	2764	2732A	2716
V_{PP}	V_{PP}		
A_{12}	A_{12}		
A_7	A_7	A_7	A_7
A_6	A_6	A_6	A_6
A_5	A_5	A_5	A_5
A_4	A_4	A_4	A_4
A_3	A_3	A_3	A_3
A_2	A_2	A_2	A_2
A_1	A_1	A_1	A_1
A_0	A_0	A_0	A_0
O_0	O_0	O_0	O_0
O_1	O_1	O_1	O_1
O_2	O_2	O_2	O_2
Gnd	Gnd	Gnd	Gnd



2716	2732A	2764	27256
		VCC	VCC
		PGM	A14
VCC	VCC	N.C.	A13
A8	A8	A8	A8
A9	A9	A9	A9
VPP	A11	A11	A11
OE	OE/VPP	OE	OE
A10	A10	A10	A10
CE	CE	CE	CE
O7	O7	O7	O7
O6	O6	O6	O6
O5	O5	O5	O5
O4	O4	O4	O4
O3	O3	O3	O3

NOTE. INTEL "UNIVERSAL SITE" COMPATIBLE EPROM PIN CONFIGURATIONS ARE SHOWN IN THE BLOCKS ADJACENT TO THE 27128 PINS

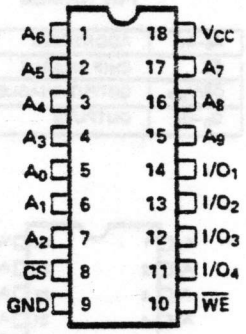
Figure 2. Pin Configurations

PIN NAMES

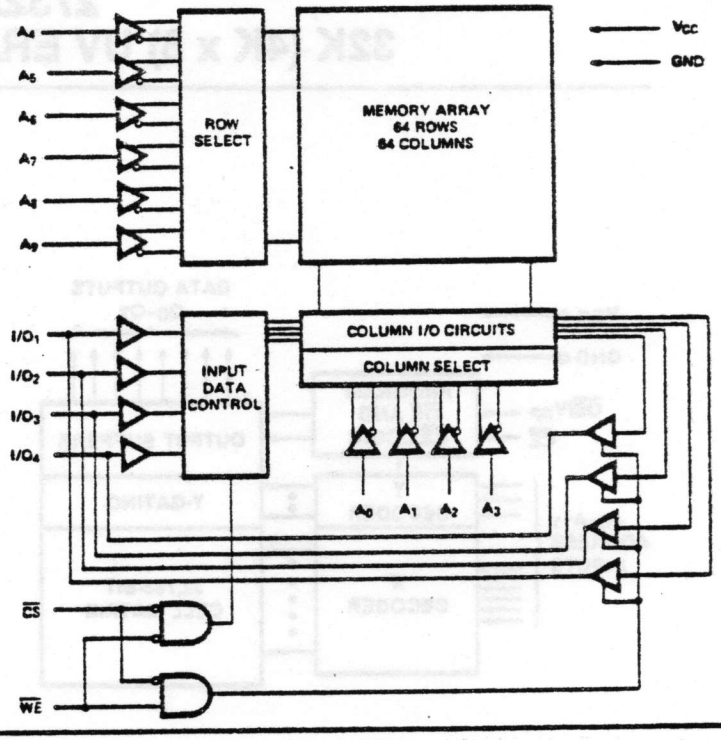
A_0-A_{13}	ADDRESSES
$\overline{CE}$	CHIP ENABLE
$\overline{OE}$	OUTPUT ENABLE
O_0-O_7	OUTPUTS
$\overline{PGM}$	PROGRAM
N.C.	NO CONNECT

SY2149H

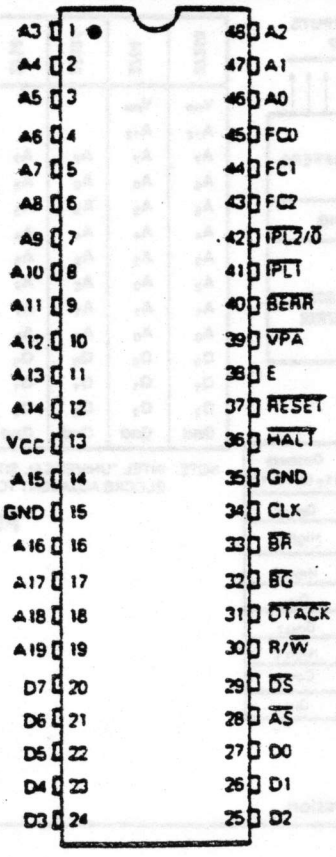
Pin Configuration



Block Diagram

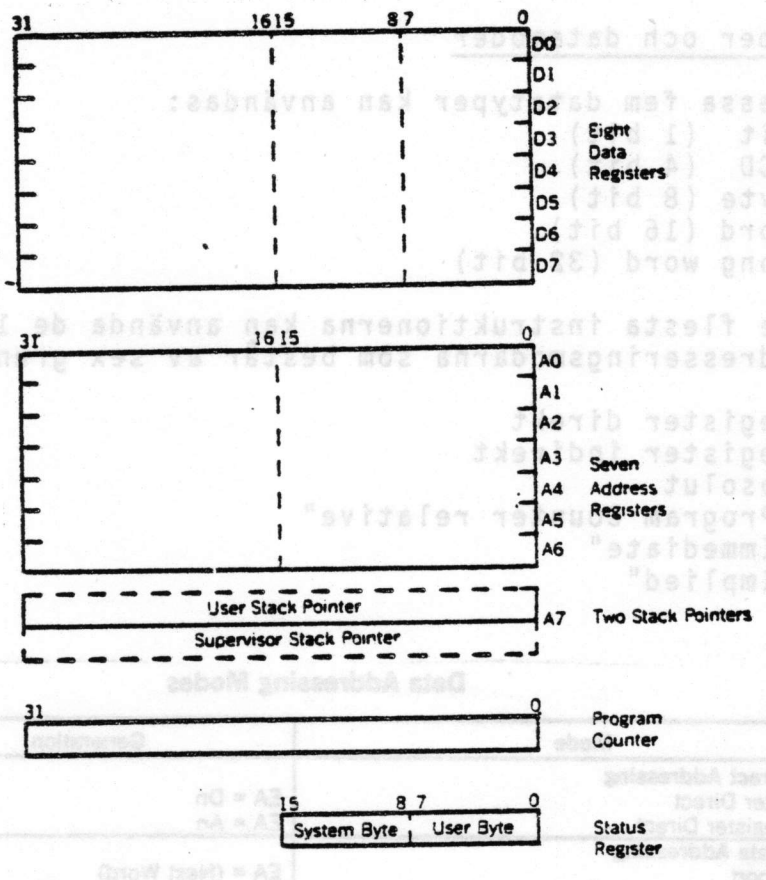


CPU MC 68008



5.2.2 MC 68008

CPU:n i ABC 1600 heter MC 68008 och är en efterföljare till MC 68000. MC 68008 är ur programmeringssynpunkt helt kompatibel med 68000, men med 8-bitars databuss för att enklare kunna anpassas till idag befintliga periferikretsar. Även adressbussen är något begränsad så att det adressbara minnesutrymmet är 1 Mbyte mot 68000 som har 16 Mbyte direkt adresserbart minne. Den interna uppbyggnaden är identisk hos de båda processorerna vilket innebär 17 st 32-bitars data och adressregister, 56 grundinstruktioner och 14 adresseringsmoder.

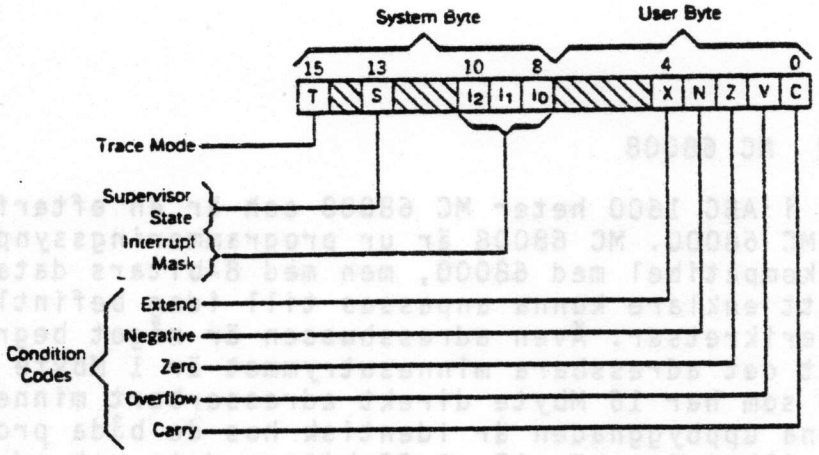


Programming Model

Register

Processorns 17 register består av 8 st 32-bitars dataregister (D0-07) som kan användas för "byte" (8 bitar), "word" (16 bitar) eller "Long word" (32 bitar) data. 7 adressregister (A0-A6) samt två register för stackpekare. Stackpekarna används en åt gången beroende på om processorn befinner sig i systemmod eller användarmod.

Förutom dessa register finns en 32-bitars programmräknare och ett 16-bitars statusregister.



Status Register

Datatyper och datamoder

Dessa fem datatyper kan användas:
Bit (1 bit)
BCD (4 bit)
Byte (8 bit)
Word (16 bit)
Long word (32 bit)

De flesta instruktionerna kan använda de 14 olika adresseringsmodarna som består av sex grundtyper.

Register direkt
Register indirekt
Absolut
"Program Counter relative"
"Immediate"
"Implied"

Data Addressing Modes

Mode	Generation
Register Direct Addressing	
Data Register Direct	EA = Dn
Address Register Direct	EA = An
Absolute Data Addressing	
Absolute Short	EA = (Next Word)
Absolute Long	EA = (Next Two Words)
Program Counter Relative Addressing	
Relative with Offset	EA = (PC) + d_{16}
Relative with Index and Offset	EA = (PC) + (Xn) + d_8
Register Indirect Addressing	
Register Indirect	EA = (An)
Postincrement Register Indirect	EA = (An), An ← An + N
Predecrement Register Indirect	An ← An - N, EA = (An)
Register Indirect With Offset	EA = (An) + d_{16}
Indexed Register Indirect With Offset	EA = (An) + (Xn) + d_8
Immediate Data Addressing	
Immediate	DATA = Next Word(s)
Quick Immediate	Inherent Data
Implied Addressing	
Implied Register	EA = SR, USP, SP, PC

NOTES:
EA = Effective Address
An = Address Register
Dn = Data Register
Xn = Address or Data Register used as Index Register
SR = Status Register
PC = Program Counter
d_8 = Eight-bit Offset (displacement)
d_{16} = Sixteen-bit Offset (displacement)
N = 1 for Byte, 2 for Words and 4 for Long Words
( ) = Contents of
← = Replaces

Översikt av instruktionsset

Nedan följer en översikt av processorns instruktionsset för att kunna flytta data, göra aritmetriska och logiska beräkningar, skifta och rotera data, bitmanipulering, BCD-beräkningar och för program och systemkontroll.

Instruction Set

Mnemonic	Description
ABCD	Add Decimal with Extend
ADD	Add
AND	Logical And
ASL	Arithmetic Shift Left
ASR	Arithmetic Shift Right
Bcc	Branch Conditionally
BCHG	Bit Test and Change
BCLR	Bit Test and Clear
BRA	Branch Always
BSET	Bit Test and Set
BSR	Branch to Subroutine
BTST	Bit Test
CHK	Check Register Against Bounds
CLR	Clear Operand
CMP	Compare
DBcc	Test Cond., Decrement and Branch
DIVS	Signed Divide
DIVU	Unsigned Divide
EOR	Exclusive Or
EXG	Exchange Registers
EXT	Sign Extend
JMP	Jump
JSR	Jump to Subroutine
LEA	Load Effective Address
LINK	Link Stack
LSL	Logical Shift Left
LSR	Logical Shift Right
MOVE	Move
MOVEM	Move Multiple Registers
MOVEP	Move Peripheral Data
MULS	Signed Multiply
MULU	Unsigned Multiply
NBCD	Negate Decimal with Extend
NEG	Negate
NOP	No Operation
NOT	One's Complement
OR	Logical Or
PEA	Push Effective Address
RESET	Reset External Devices
ROL	Rotate Left without Extend
ROR	Rotate Right without Extend
ROXL	Rotate Left with Extend
ROXR	Rotate Right with Extend
RTE	Return from Exception
RTR	Return and Restore
RTS	Return from Subroutine

Instruction Set (continued)

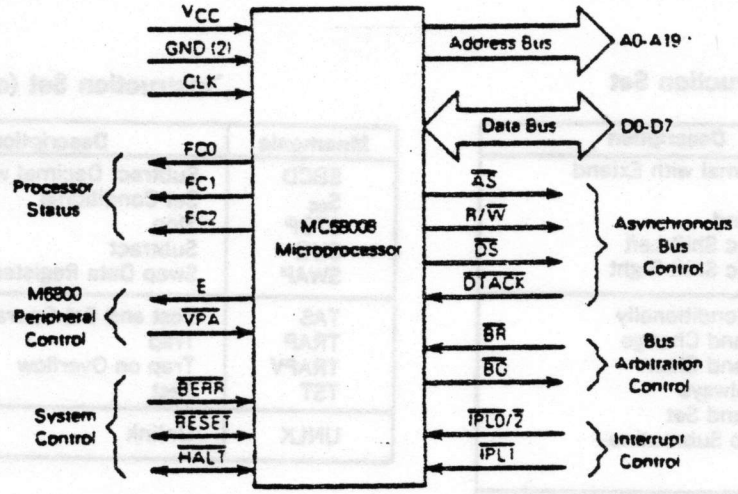
Mnemonic	Description
SBCD	Subtract Decimal with Extend
Scc	Set Conditional
STOP	Stop
SUB	Subtract
SWAP	Swap Data Register Halves
TAS	Test and Set Operand
TRAP	Trap
TRAPV	Trap on Overflow
TST	Test
UNLK	Unlink

Variations of Instruction Types

Instruction Type	Variation	Description
ADD	**ADD**	Add
	ADDA	Add Address
	ADDQ	Add Quick
	ADDI	Add Immediate
	ADDX	Add with Extend
AND	**AND**	Logical And
	ANDI	And Immediate
CMP	**CMP**	Compare
	CMPA	Compare Address
	CMPM	Compare Memory
	CMPI	Compare Immediate
EOR	**EOR**	Exclusive Or
	EORI	Exclusive Or Immediate
MOVE	**MOVE**	Move
	MOVEA	Move Address
	MOVEQ	Move Quick
	MOVE from SR	Move from Status Register
	MOVE to SR	Move to Status Register
	MOVE to CCR	Move to Condition Codes
	MOVE USP	Move User Stack Pointer
NEG	**NEG**	Negate
	NEGX	Negate with Extend
OR	**OR**	Logical Or
	ORI	Or Immediate
SUB	**SUB**	Subtract
	SUBA	Subtract Address
	SUBI	Subtract Immediate
	SUBQ	Subtract Quick
	SUBX	Subtract with Extend

Beskrivning av signaler MC 68008

Nedan följer en kortfattad beskrivning av signaler och signalgrupper hos processor MC 68008.



Input and Output Signals

Adressbuss:

Tjugo adressledningar (A0-A19) för adressering av minnen och yttre enheter. Kan sättas i högimpediv mod då någon annan enhet begär att få kontroll. Då processorn accepterat ett interrupt används (A1-A3) för att visa interruptnivå (1-7) medan övriga adressledningar är "1"-ställda (A0, A4-A19).

Databuss:

Dubbelriktad 8-bitars buss för att transportera data till och från processor. Då en interruptbegäran accepterats används databussen för att hämta ett vektornummer från en yttre enhet.
Högimpediv då inga data är tillgängliga på bussen.

Adresstrob:

Högimpediv signal som indikerar att en adress finns tillgänglig på adressbussen. Används också som lås under en "read modify-write"sekvens som används av instruktionen TAS. (test and set)

"Read/Write" (R/W):

Högimpediv signal som indikerar om nuvarande sekvens är en läs eller skrivsekvens. Denna signal används också i kombination med datastroben (DS).

Datastrob (DS):

Högimpediv signal som kontrollerar flödet på databussen. Då R/W - signalen går hög kommer processorn att läsa data från data-bussen och då R/W - signalen är låg, lägger processorn ut data på bussen.
Då datastroben är hög indikeras att inga data finns tillgängliga på databussen.

"Data Transfer Acknowledge" (DTACK):

Ingång som indikerar att dataöverföringen är avslutad. Då processorn upptäcker DTACK under en lässekvens hämtas data och sekvensen avslutas. Under en skrivsekvens avslutas sekvensen genast.

Busskontroll:

MC 68008 kan med två kontrolledningar (BR, BG) samarbeta med andra intelligenta enheter om att adressera bussen. De yttre enheterna kan vara kedjekopplade (daisy-chain) eller sammankopplade till samma ledning (wired-OR) eller en kombination av dessa.

Bussbegäran (BR):

Ingång dit alla enheter som kan begära kontroll över bussarna är sammankopplade. Signaler indikerar att någon annan enhet vill ta kontrollen och kan uppträda när som helst under en instruktions eller data-sekvens.

Bussmedgivande:(BG)

Utgång som indikerar att CPU:n kommer att släppa sin kontroll över bussarna då pågående sekvens avslutats.

Interrupt kontroll: (IPL0/IPL2,IPL1)

Dessa ingångar indikerar prioritetsnivån på den enhet som begär ett interrupt. I 68000/68010 finns tre ingångar dvs åtta (0-7) nivåer kan avkännas men hos 68008 är ILP0 och ILP2 sammankopplade och kan därför endast känna av fyra nivåer(0, 2, 5 ,7). Nivå 0 indikerar att ingen enhet begär interrupt och nivå 7 icke maskerbart interrupt. Med undantag för nivå 7 måste den begärande enhetens nivå vara högre än den nivå som indikeras av processorns statusregister innan en begäran accepteras. Då en interruptbegäran accepteras, visas detta med hjälp av funktionskoderna (FC0-FC2).

Systemkontroll:

Dessa kontrollsignaler används för att generera reset, halt till processorn eller för att indikera att något gått fel vid överföring av data.

Bussfel: (BERR)

Denna ingång talar om för processorn att något fel inträffat i den sekvens som just genomförts. Orsakerna kan vara följande:

1. aktuell enhet svarar ej.
2. otillåten minnesbegäran.
3. andra applikationsberoende fel.

BERR-signalen samarbetar med HALT-signalen för att kunna avgöra om denna sekvens skall repeteras eller om en speciell undantagssekvens skall utföras.

Reset:

Dubbelriktad. Då resetsignalen genereras externt kommer processorn att starta sin initieringssekvens. En internt genererad reset (genom RESET-instruktionen) gör att alla externa enheter ges reset. En total systemreset åstadskommes genom att både resetoch halt-ledningarna aktiveras samtidigt.

Halt:

Då denna dubbelriktade signal aktiveras utifrån kommer processen att stanna så snart pågående sekvens avslutas och kontrollsignaler och bussar läggs högimpendiva.

MC 6800 periferi kontroll:

Används för att kunna ansluta periferikretsar ur 6800-familjen till processorn. (Används i vårt fall på ett annat sätt.)

Enable (E):

Klocka för synkronisering som är 1/10 av systemklockan till 68008. Sex perioder låg, fyra hög.

VPA:

Ingång som indikerar att data till/från yttre enhet ska synkroniseras med "E". Vid interupt ska processorn använda automatisk vektor.

Processtatus:(FC0-FC2)

Dessa tre utsignaler indikerar vilken mod (användar eller system) processen befinner sig i, samt vilken typ av sekvens som utföres. Informationen finns tillgänglig då adresstroben är aktiv.

Function Code Outputs

FC2	FC1	FC0	Cycle Type
Low	Low	Low	(Undefined, Reserved)
Low	Low	High	User Data
Low	High	Low	User Program
Low	High	High	(Undefined, Reserved)
High	Low	Low	(Undefined, Reserved)
High	Low	High	Supervisor Data
High	High	Low	Supervisor Program
High	High	High	Interrupt Acknowledge

Klocka(CKL)

Systemklocka

VCC+GND:

+5V /0V

Sammanställning signaler.

Signal Summary

Signal Name	Mnemnoic	Input/Output	Active State	Hi-Z	
				on $\overline{HALT}$	on $\overline{BGACK}$
Address Bus	A0 A19	Output	High	Yes	Yes
Data Bus	D0-D7	Input/Output	High	Yes	Yes
Address Strobe	$\overline{AS}$	Output	Low	No	Yes
Read/Write	R/$\overline{W}$	Output	Read-High Write-Low	No No	Yes Yes
Data Strobes	$\overline{DS}$	Output	Low	No	Yes
Data Transfer Acknowledge	$\overline{DTACK}$	Input	Low	No	No
Bus Request	$\overline{BR}$	Input	Low	No	No
Bus Grant	$\overline{BG}$	Output	Low	No	No
Interrupt Priority Level	$\overline{IPL2:0}$, $\overline{IPL1}$	Input	Low	No	No
Bus Error	$\overline{BERR}$	Input	Low	No	No
Reset	$\overline{RESET}$	Input/Output	Low	No*	No*
Halt	$\overline{HALT}$	Input/Output	Low	No*	No*
Enable	E	Output	High	No	No
Valid Peripheral Address	$\overline{VPA}$	Input	Low	No	No
Function Code Output	FC0, FC1, FC2	Output	High	No	Yes
Clock	CLK	Input	High	No	No
Power Input	VCC	Input	-	-	-
Ground	GND	Input	-	-	-

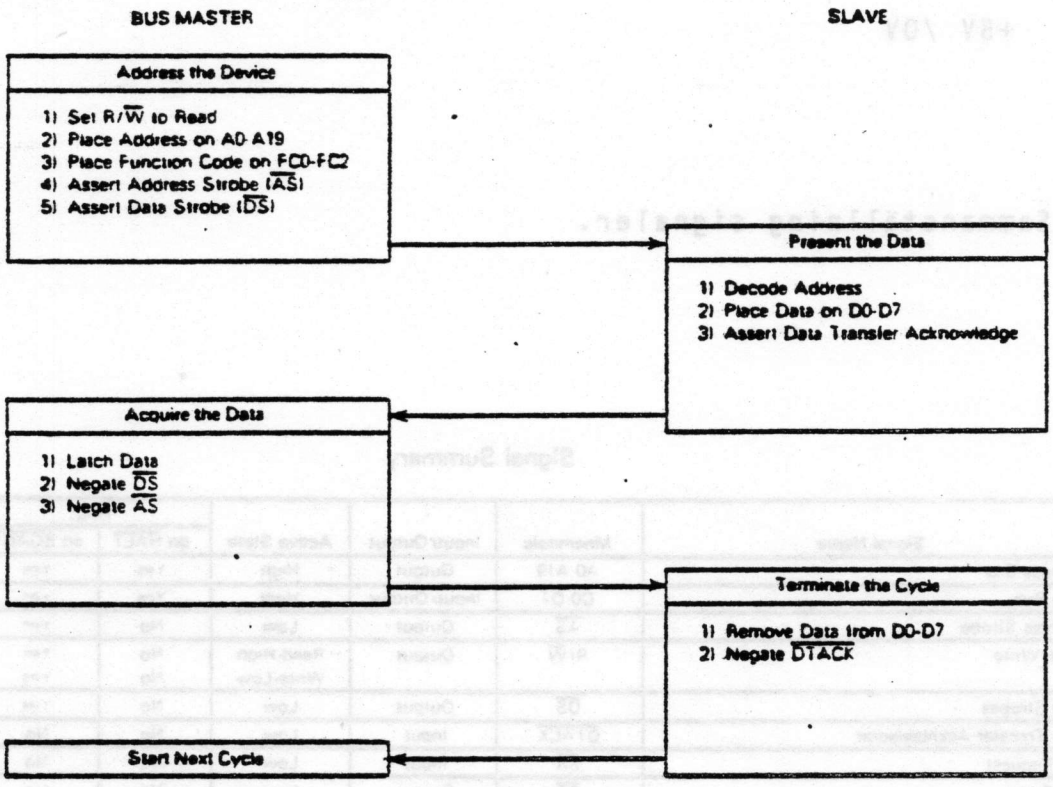
*Open Drain

Processorns arbetssekvenser

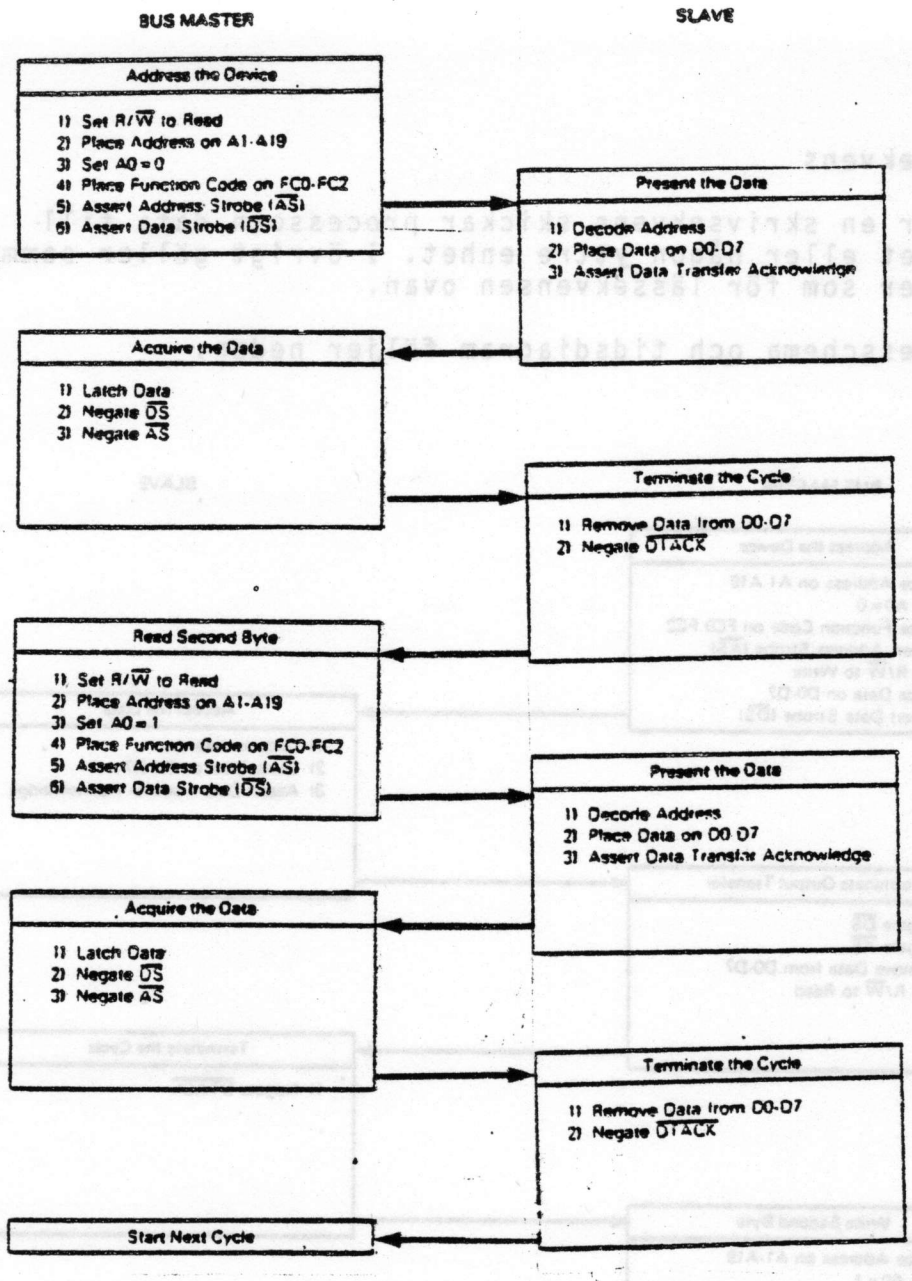
Detta avsnitt behandlar CPU:ns sätt att hantera skrivning - läsning av data, signalering vid bussövertagande, halt och reset samt "buserror".

Lässekvens

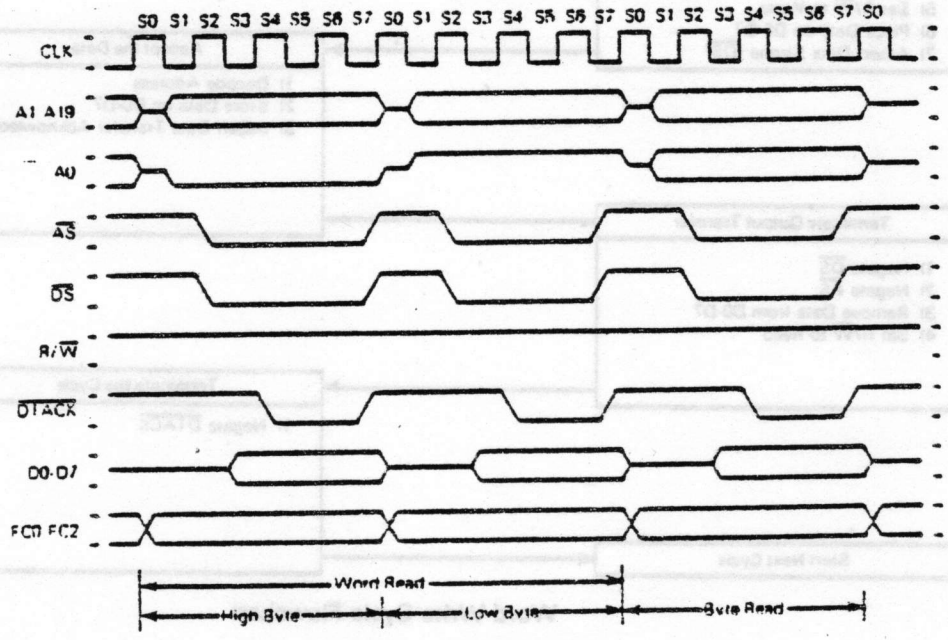
Under en lässekvens mottar processorn data från minnet eller en yttre enhet. Processorn läser alltid byte och om en instruktion specifiserar "word" eller "long word" kommer processorn att läsa två resp. fyra bytes. Då instruktionen specifiserar "byte", använder processorn adress A0 för att visa vilken byte som ska läsas och lägger sedan ut datastrob. Nedan följer tidsdiagram och schema över sekvenserna.



Byte Read Cycle Flowchart



Word Read Cycle Flowchart

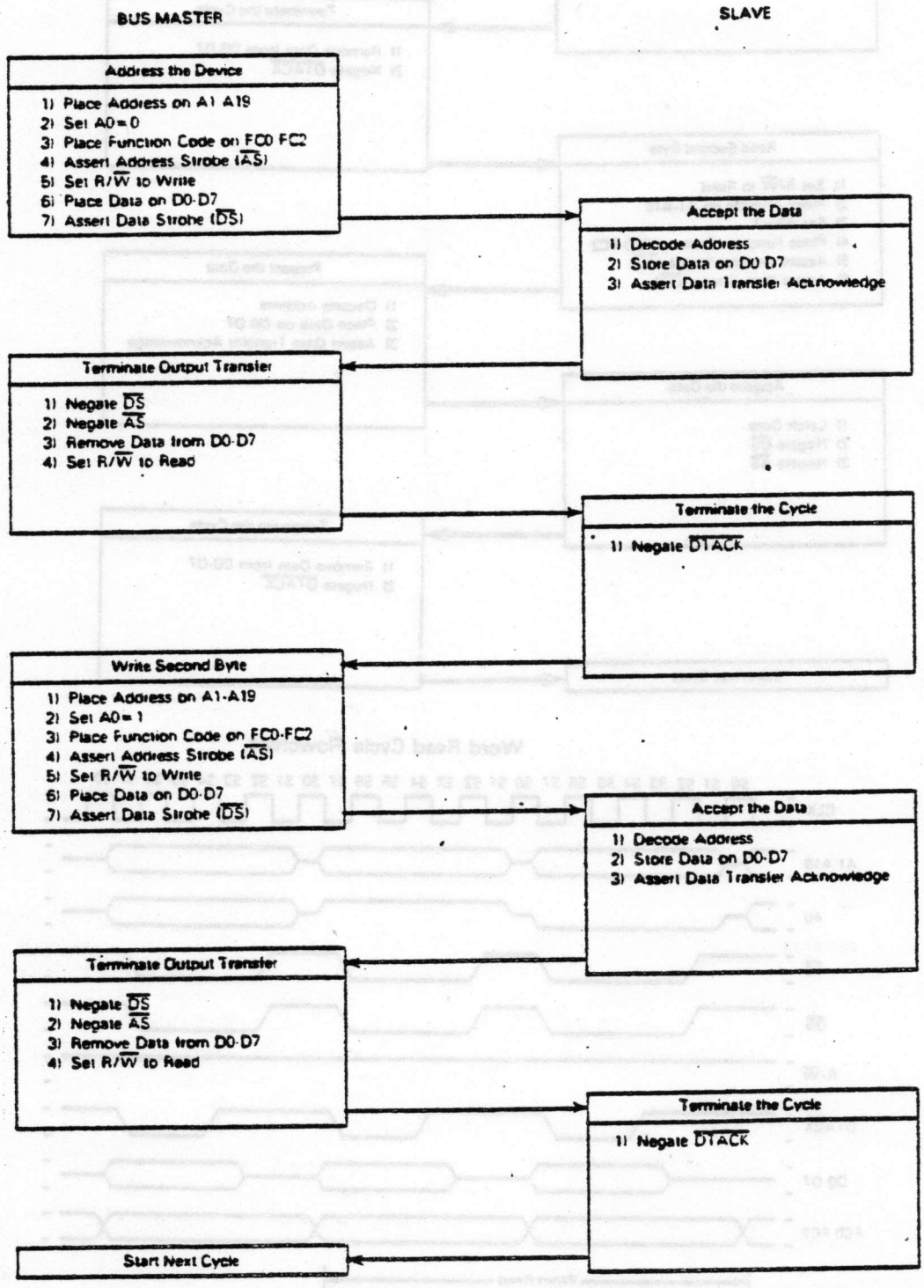


Word and Byte Read Cycle Timing

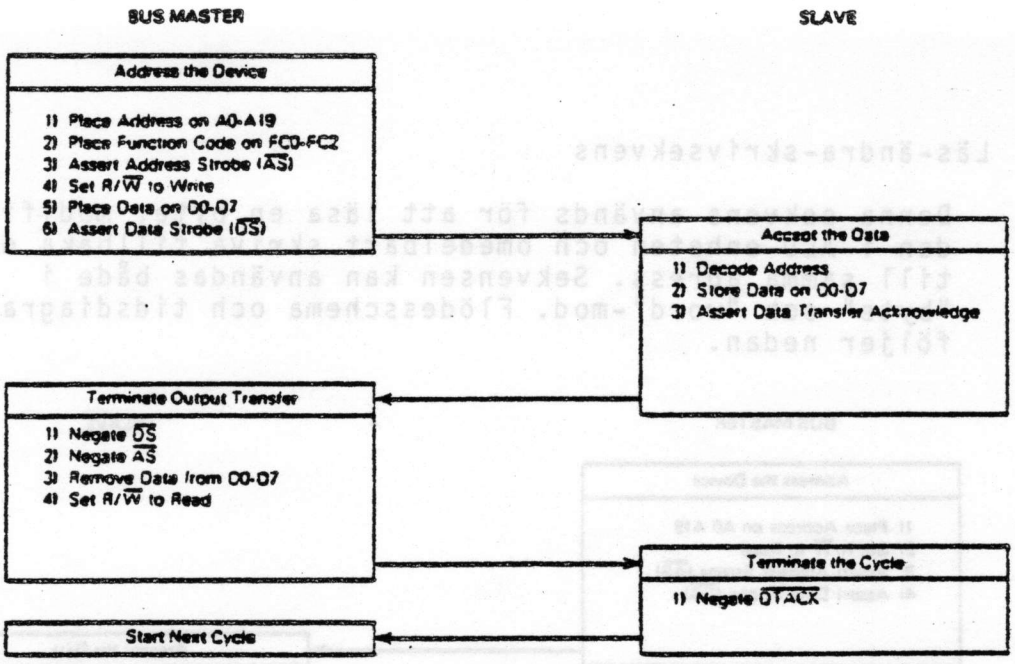
Skrivsekvens

Under en skrivsekvens skickar processorn data till minnet eller någon yttre enhet. I övrigt gäller samma regler som för lässekvensen ovan.

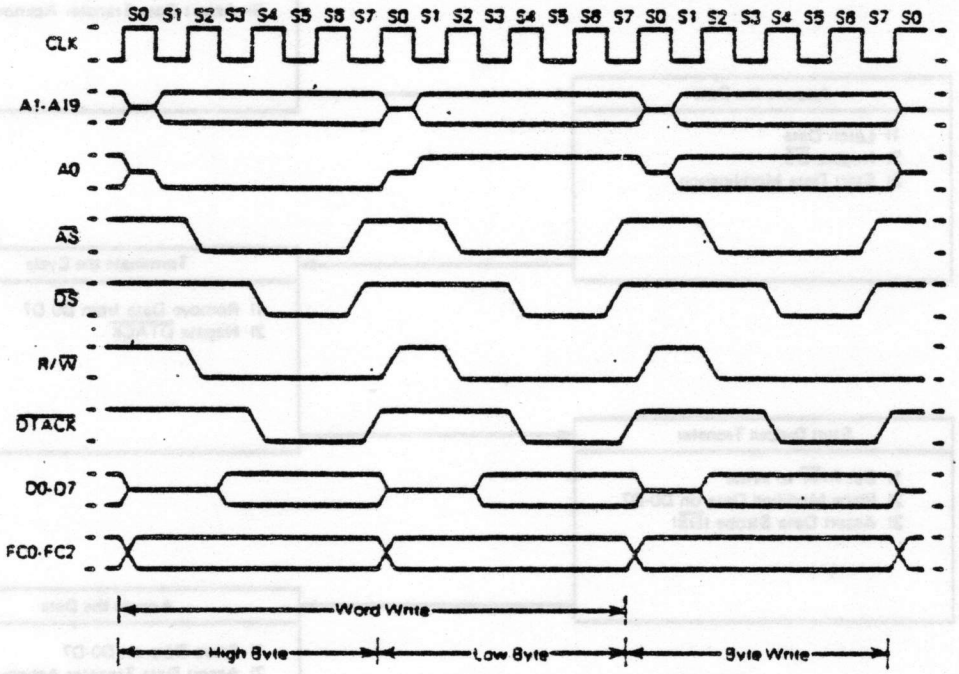
Flödesschema och tidsdiagram följer nedan.



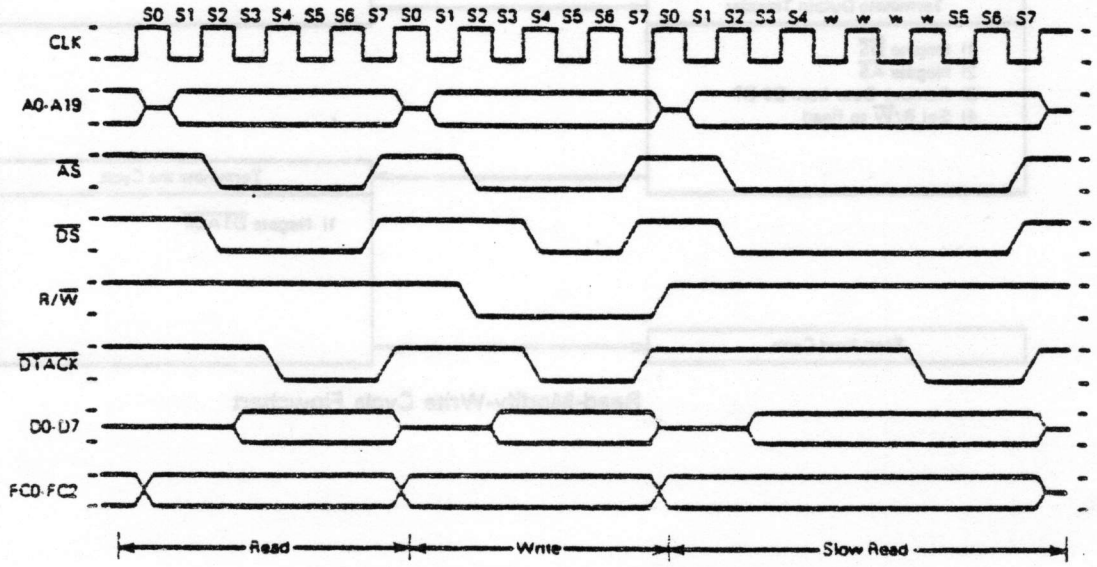
Word Write Cycle Flowchart



Byte Write Cycle Flowchart



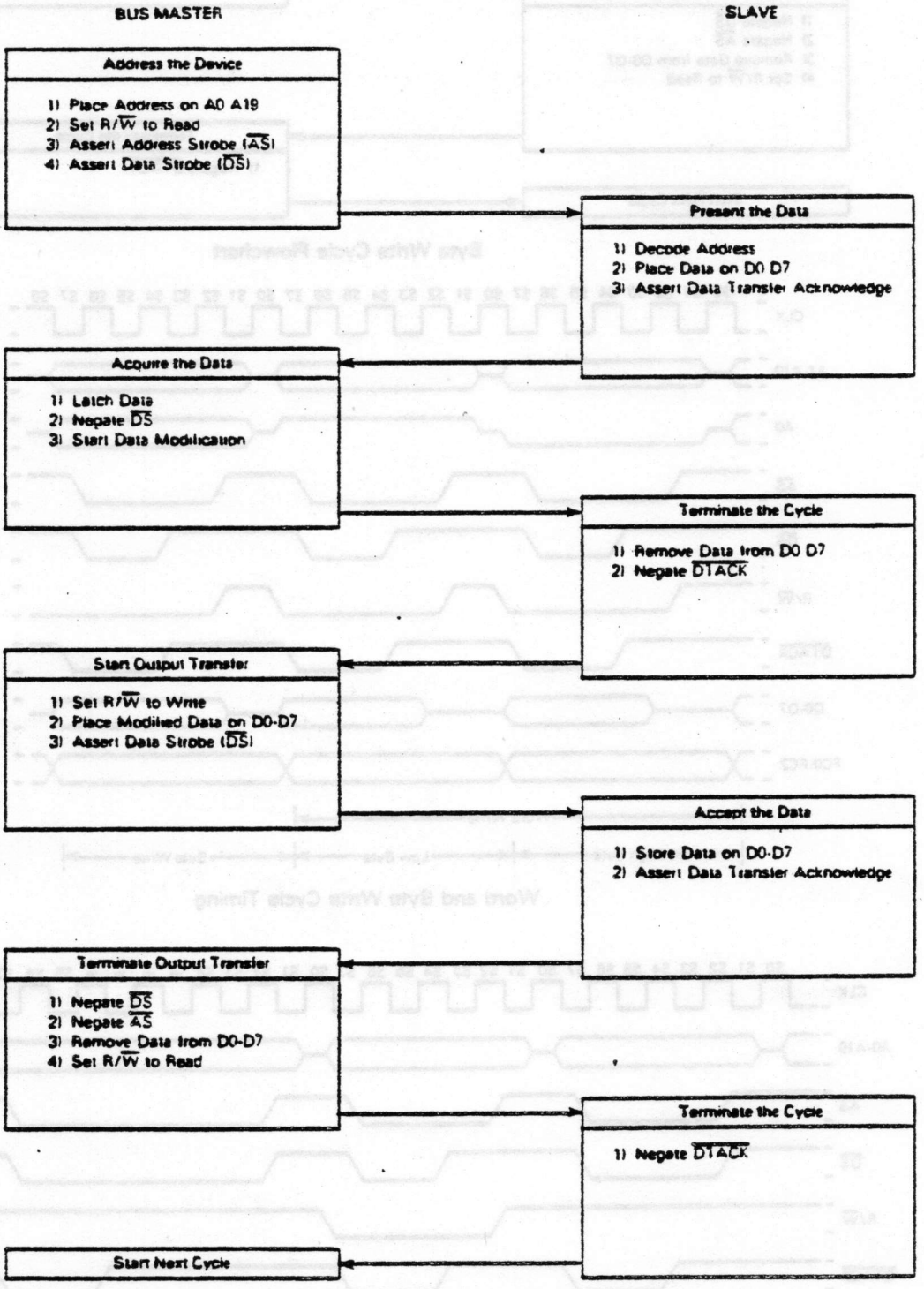
Word and Byte Write Cycle Timing



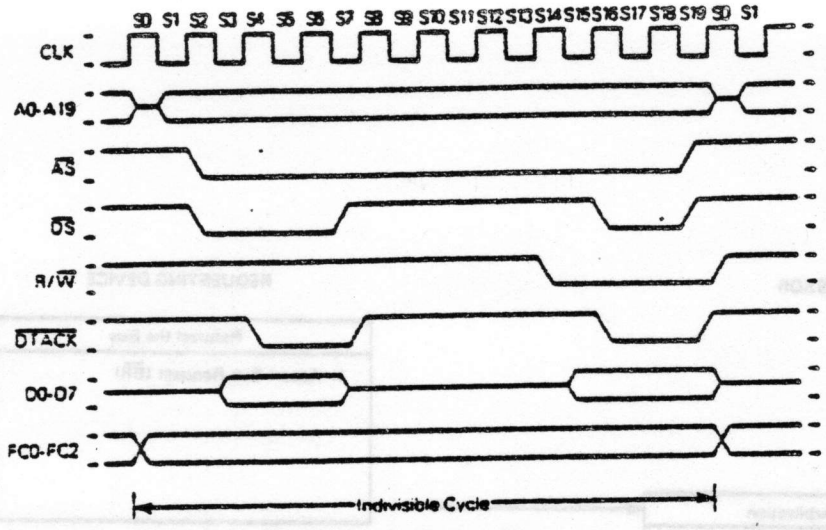
Read and Write Cycle Timing Diagram

Läs-ändra-skrivsekvens

Denna sekvens används för att läsa en byte, modifiera den i ALU-enheten och omedelbart skriva tillbaka den till samma adress. Sekvensen kan användas både i "byte" och "word"-mod. Flödesschema och tidsdiagram följer nedan.



Read-Modify-Write Cycle Flowchart



Read-Modify-Write Cycle Timing

Bussövertagande:

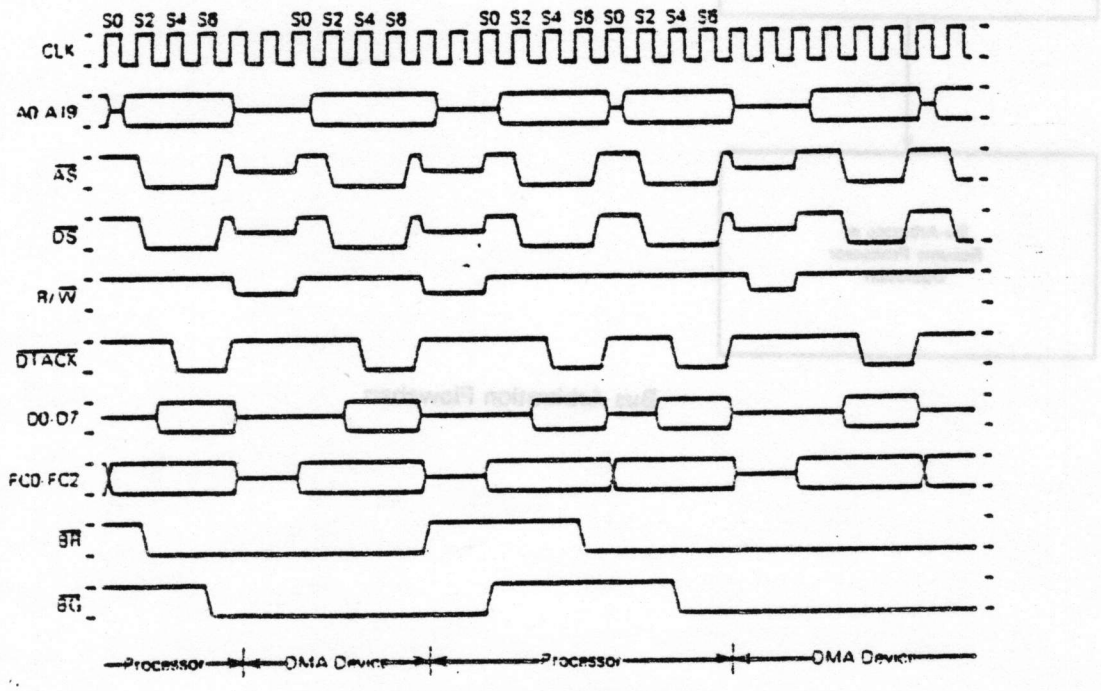
Bussövertagande är en teknik som används då det till processorn finns sådana enheter anslutna som själv kan generera adress, data och kontrollsignaler. En sådan enhet är exempelvis DMA.

Att begära busskontroll:

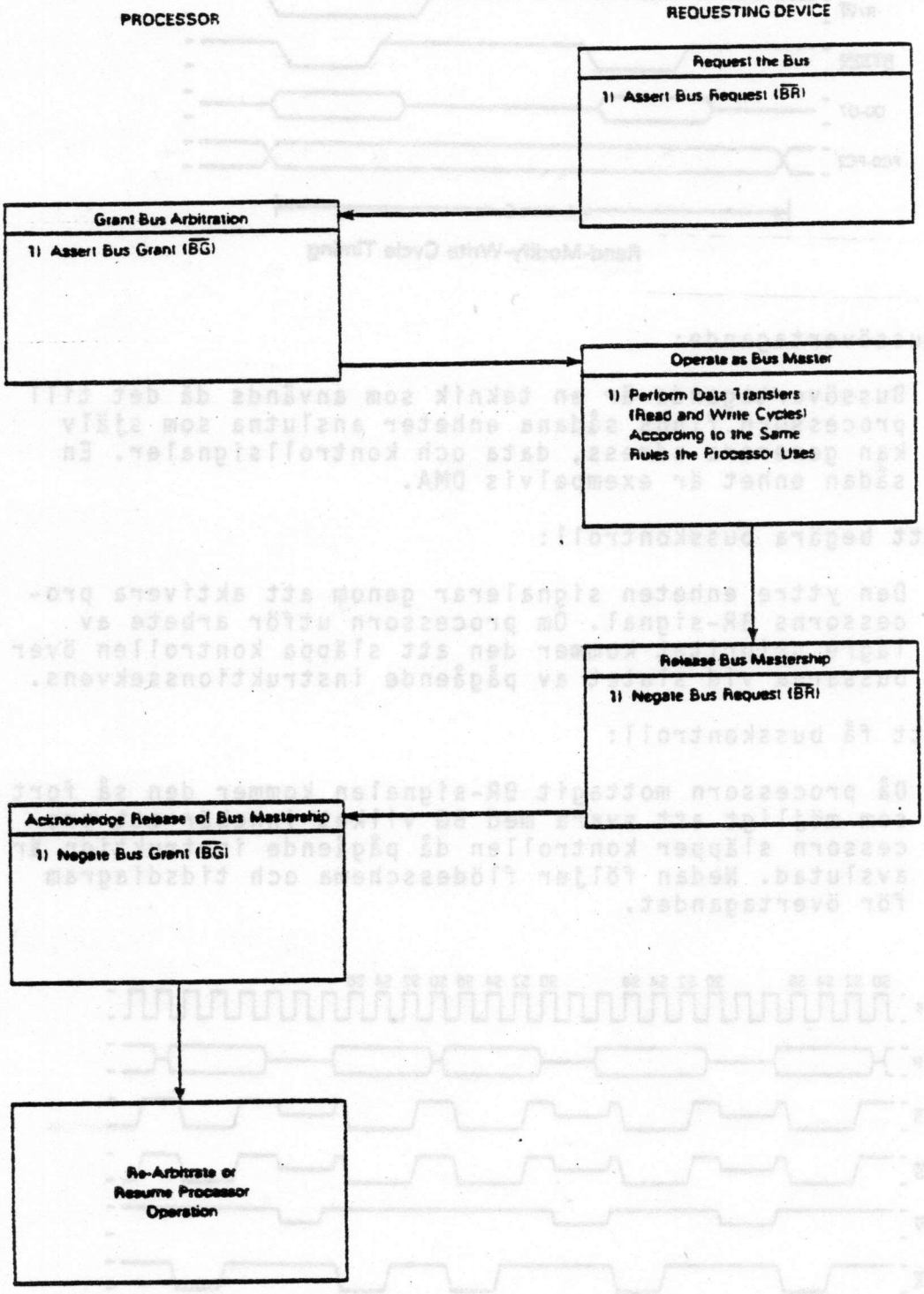
Den yttre enheten signalerar genom att aktivera processorns BR-signal. Om processorn utför arbete av lägre prioritet kommer den att släppa kontrollen över bussarna vid slutet av pågående instruktionssekvens.

Att få busskontroll:

Då processorn mottagit BR-signalen kommer den så fort som möjligt att svara med BG vilket innebär att processorn släpper kontrollen då pågående instruktion är avslutad. Nedan följer flödesschema och tidsdiagram för övertagandet.



Bus Arbitration Timing



Bus Arbitration Flowchart

Undantag från normal körning ("Exception processing")

Processorn kan befinna sig i tre olika lägen, normal (den vanliga programvaran körs), undantag (något har inträffat som gör att processorn lämnar den normala programkörningen för att köra en speciell rutin) och stoppad.

De undantagsrutiner som kan förekomma är exv. interrupt, hårdvaran signalerar bussfel (busserror), reset, adressfel, otillåten instruktion (kan vara instruktion som ej finns eller att en instruktion som är avsedd för systemmod försöker köras i avändarmod eller någon av instruktionerna TRAP, TRAPV, CHK, STOP eller RESET.

Då ett undantag uppträder sker normalt följande:

1. en kopia av statusregistret görs och statusregistret sätts upp för att hantera undantagsläget.
2. Undantagsvektor bestäms beroende på vad som inträffat.
3. Processorregistrens innehåll sparas.
4. Det nya registerinnehållet läses in och instruktionskörningen startar.

Beroende på vilken typ av undantag som utlöst sekvensen kommer en vektor att produceras som pekar på en viss position i vektortabellen.

Interrupt

En interruptbegäran startar med att interruptingångarna ILP0/2 och ILP1 aktiveras. Beroende på hur dessa ingångar aktiveras får begäran en viss prioritet. I MC 86008 kan på detta sätt tre olika interruptnivåer avkännas. Då en sådan interruptbegäran kommit till CPU:n genereras "interrupt acknowledge" genom att lägga de tre funktionskoderna (FC0-FC2) till hög nivå samt med adressledningarna A1-A3 visa vilken nivå som accepterats. Den begärande enheten svarar nu med att lägga ut en interruptvektor på databussen som CPU:n använder för att i vektortabellen hitta den rutin som ska köras vid just denna interruptbegäran.

Bussfel

Hos alla busskontrollsystem som arbetar med handskakning finns en möjlighet att något går fel eller att svar helt enkelt uteblir. Då olika yttre enheter kräver olika lång tid på sig för att ta mot eller leverera data måste också de yttre enheterna bestämma när något är fel.

Den yttre enheten flaggar då till processorn med BERR-signalen och processorn måste nu avgöra om felet är allvarligt dvs. om en speciell felsekvens skall aktiveras eller om man ska försöka överföra data en gång till.

Då signalen kommer på BERR-ledningen men ej på HALT kommer processorn att starta en speciell undantagssekvens så snart den yttre enheten släppt BERR-signalen igen.

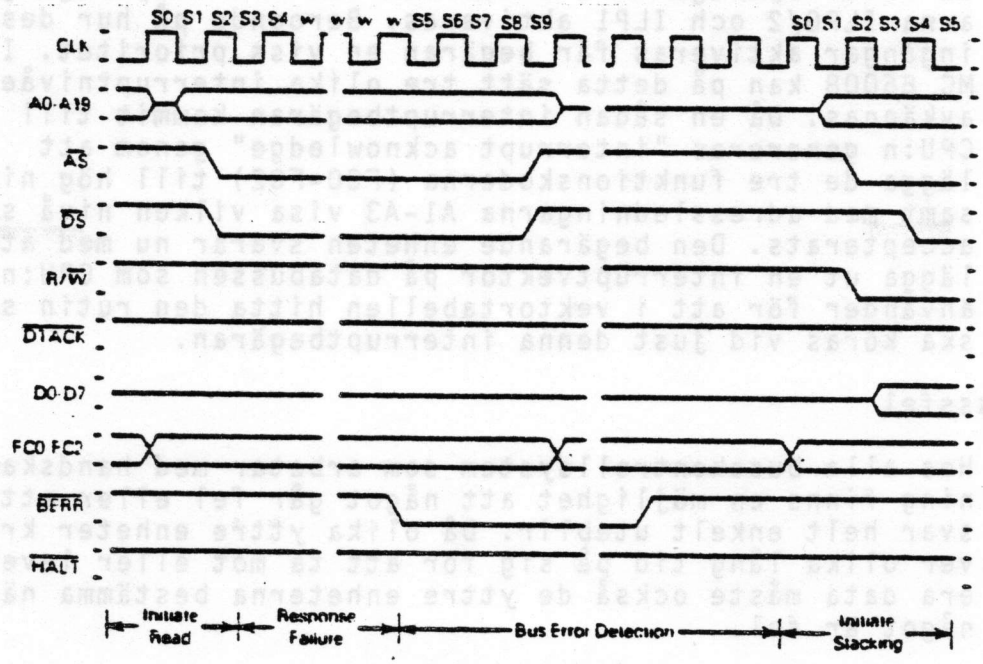
Sekvensen är som följer:

a/ programräknare och startregister sparas.
b/ spara felinformation.
c/ läs vektorn till tabellen för bussfel.
d/ kör rutinen för bussfel.

Då HALT-signalen aktiveras under det att BERRsignalen är låg kommer processorn att avsluta pågående sekvens och vänta tills HALT-signalen släppts. Nu börjar processorn med att åter köra den nyss avbrutna sekvensen från början igen.

Exception Grouping and Priority

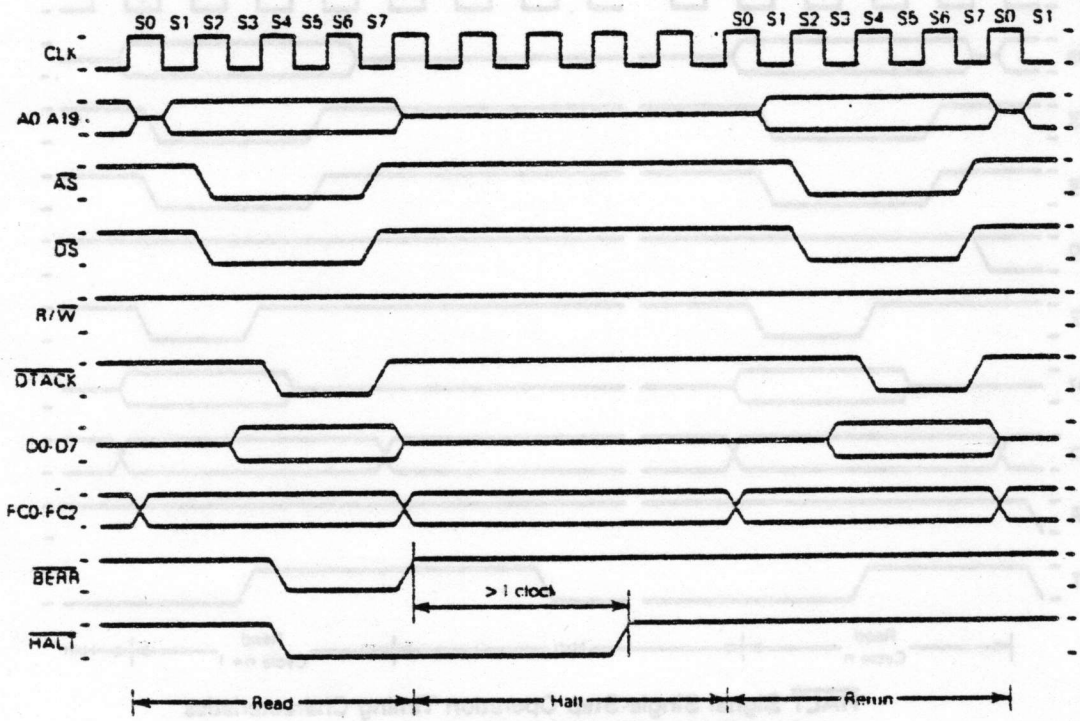
Group	Exception	Processing
0	Reset Bus Error Address Error	Exception processing begins at the next minor cycle
1	Trace Interrupt Illegal Privilege	Exception processing begins before the next instruction
2	TRAP, TRAPV, CHK, Zero Divide	Exception processing is started by normal instruction execution



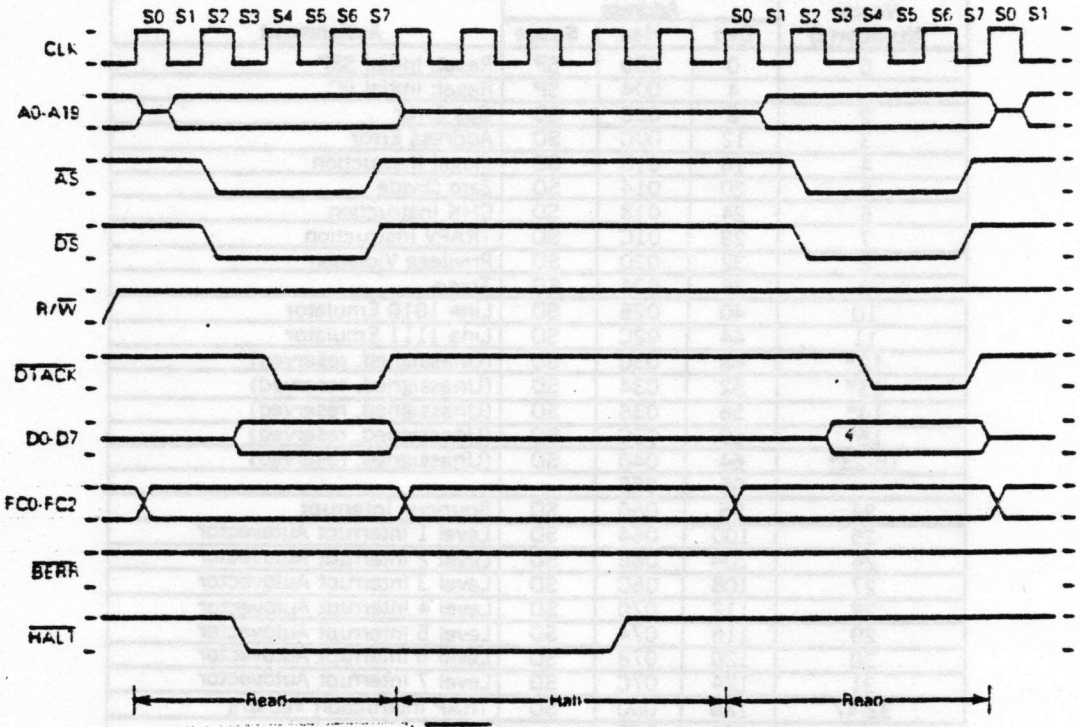
Bus Error Timing Diagram

Exception Vector Assignment

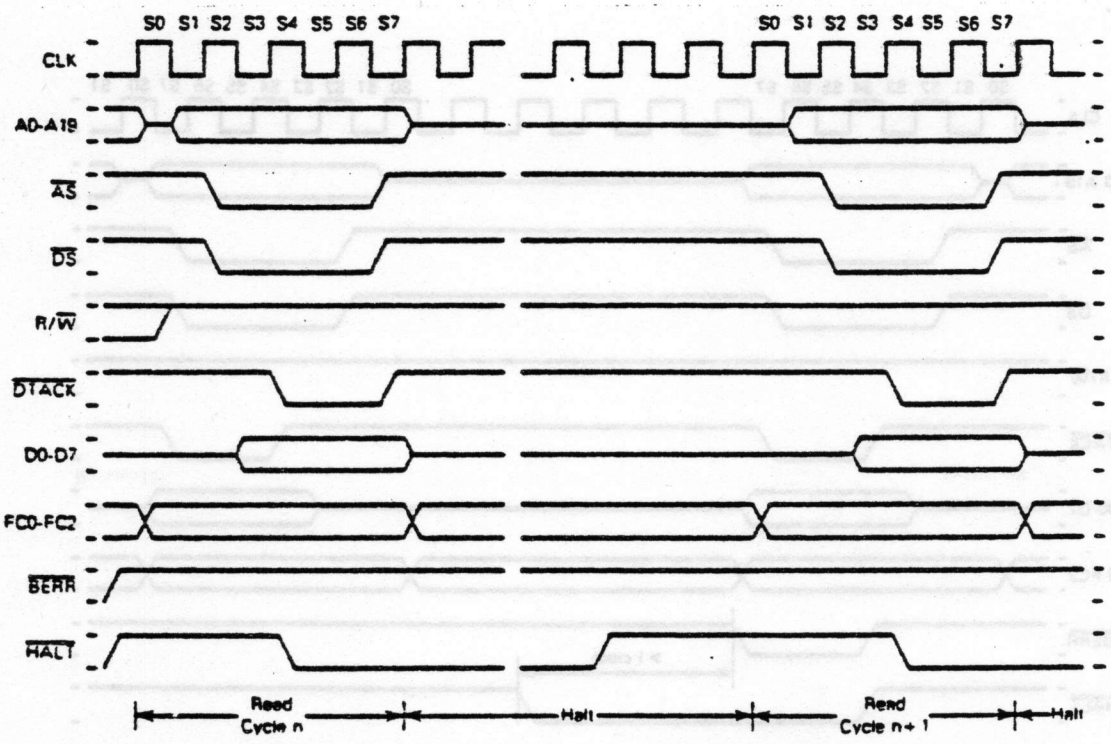
Vector Number(s)	Address			Assignment
	Dec	Hex	Space	
0	0	000	SP	Reset: Initial SSP
	4	004	SP	Reset: Initial PC
2	8	008	SD	Bus Error
3	12	00C	SD	Address Error
4	16	010	SD	Illegal Instruction
5	20	014	SD	Zero Divide
6	24	018	SD	CHK Instruction
7	28	01C	SD	TRAPV Instruction
8	32	020	SD	Privilege Violation
9	36	024	SD	Trace
10	40	028	SD	Line 1010 Emulator
11	44	02C	SD	Line 1111 Emulator
12*	48	030	SD	(Unassigned, reserved)
13*	52	034	SD	(Unassigned, reserved)
14*	56	038	SD	(Unassigned, reserved)
15*	60	03C	SD	(Unassigned, reserved)
16-23*	64	040	SD	(Unassigned, reserved)
	95	05F		—
24	96	060	SD	Spurious Interrupt
25	100	064	SD	Level 1 Interrupt Autovector
26	104	068	SD	Level 2 Interrupt Autovector
27	108	06C	SD	Level 3 Interrupt Autovector
28	112	070	SD	Level 4 Interrupt Autovector
29	116	074	SD	Level 5 Interrupt Autovector
30	120	078	SD	Level 6 Interrupt Autovector
31	124	07C	SD	Level 7 Interrupt Autovector
32-47	128	080	SD	TRAP Instruction Vectors
	191	0BF		—
48-63*	192	0C0	SD	(Unassigned, reserved)
	255	0FF		—
64-255	256	100	SD	User Interruot Vectors
	1023	3FF		—



Re-Run Bus Cycle Timing Information



HALT Operation Timing Diagram



HALT Signal Single-Step Operation Timing Characteristics

Halt under bussfel:

Då HALT-signalen aktiveras kommer processorn att sluta köra instruktioner tills signalen åter släpps. Detta kan används då man av någon anledning vill stoppa processorn under en viss tid eller för att stega fram genom programmet en instruktion i taget. Då processorn på detta sätt är stoppad befinner sig adress och databussar i högimpediv mod men signaleringen för bussövertagandet fungerar fortfarande.

$\overline{BERR}$ and $\overline{HALT}$ Negation Results

Conditions of Termination in Table 4-4	Control Signal	Negated on Rising Edge of State			Results — Next Cycle
		N		N + 2	
Bus Error	$\overline{BERR}$	●	or	●	Takes bus error trap
	$\overline{HALT}$	●	or	●	
Re-run	$\overline{BERR}$	●	or	●	Illegal sequence; usually traps to vector number 0.
	$\overline{HALT}$	●			
Re-run	$\overline{BERR}$	●			Re-runs the bus cycle
	$\overline{HALT}$			●	
Normal	$\overline{BERR}$	●			May lengthen next cycle
	$\overline{HALT}$	●	or	●	
Normal	$\overline{BERR}$			●	If next cycle is started it will be terminated as a bus error
	$\overline{HALT}$	●	or	none	

● = Signal is negated in this bus state

Dubbla bussfel:

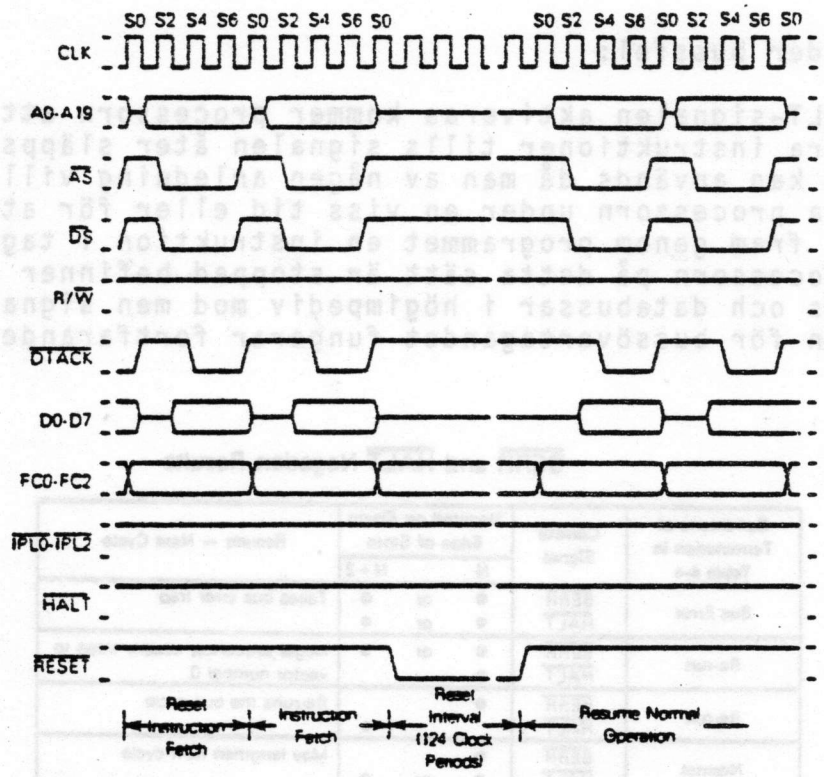
Om ett bussfel uppträder under tiden processorn hanterar ett tidigare uppkommet bussfel kommer processorn att stoppas då något stort fel troligtvis inträffat. En processor som stoppats på detta sätt kan endast fås att starta igen med hjälp av extern reset.

Reset:

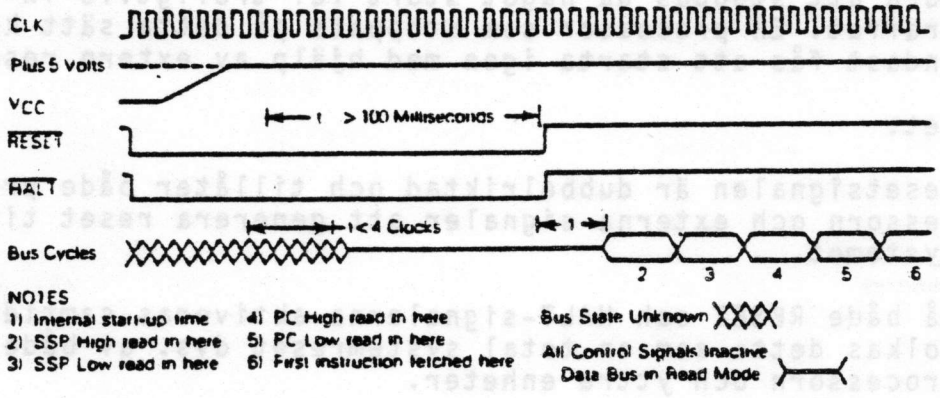
Resetsignalen är dubbelriktad och tillåter både processorn och externa signaler att generera reset till systemet.

Då både RESET och HALT-signalerna aktiveras samtidigt tolkas detta som en total systemreset dvs. av både processorn och yttre enheter.

Processorn svarar i detta läge med att läsa resetvektorn på adress 00000 och laddar denna i systemstackpekaren (SSP) och fortsätter med att läsa från adress 00004 och ladda denna i programräknaren. Därefter sätts processorn till interupnivå sju och start av resetrutinen börjar. Då processorn själv aktiverar resetledningen genom att köra en resetinstruktion kommer alla enheter som är anslutna till denna signal att nollställas utan processorn som behåller sitt registerinnehåll intakt.



Reset Operation Timing Diagram



System Reset Timing Diagram

5.2.3 Z80 DMA

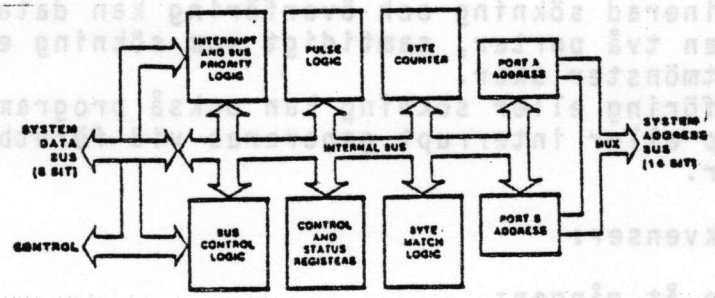
Allmän beskrivning

Z80 DMA (Direct Memory Access) är en kraftfull och mångsidig periferikrets för att styra överföring av data. Dess främsta uppgift är att handha CPU oberoende av överföring mellan två enheter, som med hjälp av dess speciella egenskaper, optimerar överföringshastigheten. Den kontrollerar med liten eller ingen inblandning av extern logik i system som använder en 8- eller 16 "bitars" databus och med en 16"bitars" adressbuss.

Överföring görs mellan valfria enheter, källa och destination inkl. minne till I/O, minne till minne och I/O till I/O. Dubbelportadresserna genereras automatiskt för varje överföring och kan vara antingen fast eller ökande/minskande. Dessutom kan den samtidigt som överföring sker, jämföra en förutbestämd bitmask med transporterat data.

Stora programmeringsmöjligheter inkl variabla tidssekvenser och automatisk återstart minimerar CPU:ns programvara. Dessa egenskaper är speciellt värdefulla då DMA:n kan användas till ett brett utbud av minnen, I/O och CPU-typer.

Z80 DMA kan kontrollera såväl adress och data som kontrollbuss.



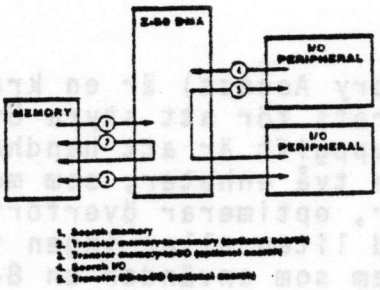
Funktionsbeskrivning

Funktionsgrupper:

Z80 DMA har tre primära funktioner:

- Överföring av data mellan två portar (Minne eller I/O
- Sökning efter en speciell 8-bits-kombination på databussen från minnet eller en I/O-enhet
- Sökning samtidigt som data överförs mellan två enheter

Figuren visar hur primära funktioner kan användas.



Under en överföring har DMA:n kontroll över adress och databussar. Data läses från en port "byte för byte". Portarna kan programmeras till antingen minnes- eller I/O-enheter. Således kan ett datablock skrivas från en area av primärminnet till en annan, eller frin en I/O-enhet till primärminnet och vice versa.

Då endast sökning ska utföras läses data från källan och jämförs "byte för byte" med DMA:ns internregister, som innehåller en programmerbar jämförelsebyte. Denna kan valfritt kombineras så att endast vissa bitar inom jämförelsebyten jämförs.

Sökningshastigheter på upp till 1,25 Mbyte/sekund kan uppnås med 2,5 MHz Z80 DMA eller 2 Mbyte/sekund med 4 MHz Z80A DMA:n.

Vid kombinerad sökning och överföring kan data överföras mellan två portar, samtidigt som sökning efter ett visst bitmönster sker.
Dataöverföring eller sökning kan också programmeras så att stopp eller interrupt genereras vid förutbestämda händelser.

Arbetssekvenser:

En byte åt gången:

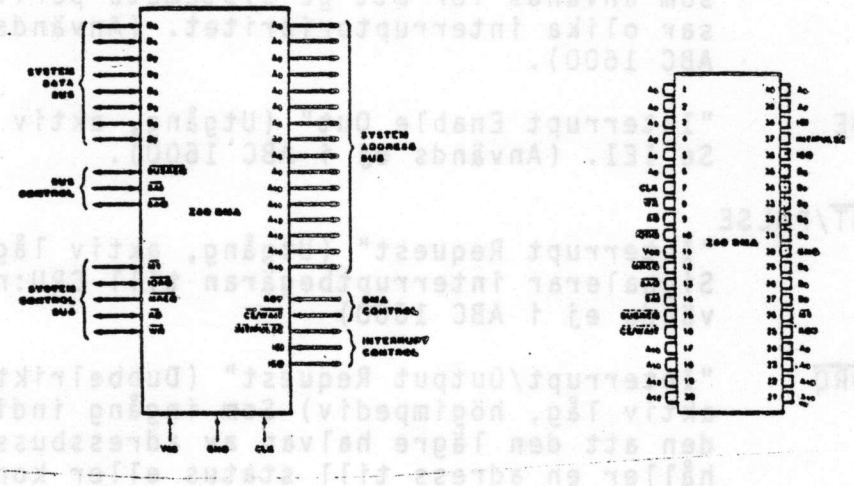
Data överförs en byte åt gången varefter DMA:n släpper busskontrollen som måste begäras på nytt för att överföringen ska fortsätta.

Ett antal bytes åt gången:

DMA:n fortsätter att överföra data tills dess RDY-ingång deaktiveras, varefter busskontrollen släpps.

Kontinuerlig överföring:

Data fortsätter att överföras tills det i registret angivna blocket är överfört. Går RDY-ingången låg (inaktiv) under tiden, stannar DMA:n överföringen temporärt tills den återigen går hög och överföringen fortsätter.



In- och utgångar:

A0-A15 "System Adress Bus" (Utgång högimpediv)
Innehåller den adress som DMA:n sänder till källa och datainstruktion.

$\overline{\text{BAI}}$ "Bus Aknowledge In" (Ingång aktiv låg)
Signalerar att CPU:n släppt busskontrollen. Används tillsammans med BAO för att bilda en kedja då flera DMA:er används i ett system. Den högst prioriterade DMA:n ansluts i detta fall till CPU:ns "Bus Acknowledge-signal".

BAO "Bus Acknowledge Out" (Utgång aktiv låg)
Används för att föra vidare CPU:ns "Bus Acknowledge"-signal då flera DMA:er används i systemet. BAO på den högst priviligerade DMA:n förbinds med BAI på nästa o.s.v.

BUSREQ "Bus Request" (dubbelriktad, aktiv låg, öppen kollektor)
Används som utgång för att begära busskontrollen, som ingång för att känna av att någon annan DMA begärt busskontrollen.

$\overline{\text{CE/WAIT}}$ "Chip Enable/WAIT (Ingång, aktiv låg)
Används normalt som CE-signal då CPU:n vill skriva eller läsa i DMA:ns register men fungerar efter bussövertagandet som WAIT-signal då DMA:n arbetar mot långsammare enheter.

CLK "System Clock" (Ingång)
4MHz (Z80A DMA) klocksignal

D0-D7 "System Data Bus" (Dubbelriktad, Högimpediv)
8-bitars databuss för transport av data till och från DMA:n.

IEI "Interrupt Enable In" (Ingång, aktiv låg)
Bildar tillsammans med IEO en interruptkedja som används för att ge systemets periferikretsar olika interruptprioritet. (Används ej i ABC 1600).

IOE "Interrupt Enable Out" (Utgång, aktiv hög)
Se IEI. (Används ej i ABC 1600).

$\overline{\text{INT}}$/PULSE "Interrupt Request" (Utgång, aktiv låg)
Signalerar interruptbegäran till CPU:n. (Används ej i ABC 1600)

$\overline{\text{IORQ}}$ "Interrupt/Output Request" (Dubbelriktad, aktiv låg, högimpediv) Som ingång indikerar den att den lägre halvan av adressbussen innehåller en adress till status eller kontrollregister som ska skrivas eller läsas av CPU:n.

Som utgång, efter det att DMA:n tagit busskontrollen indikerar den att adressbussen innehåller en adress till en I/O-enhet som ska aktiveras av DMA:n.

$\overline{\text{M1}}$ "Machine Cycle One" (Ingång, aktiv låg)
(Används i ABC 1600).

$\overline{\text{MREQ}}$ "Memory Request" (Utgång, aktiv låg högimpediv)
Indikerar att adressbussen innehåller en adress till en minnesposition för skrivning eller läsning från DMA:n.

$\overline{\text{RD}}$ "Read" (Dubbelriktad, aktiv låg högimpediv)
Som ingång indikerar den att CPU: vill läsa DMA.ns statusregister, som utgång då DMA:n tagit busskontrollen, att DMA:n läser från minne eller I/O.

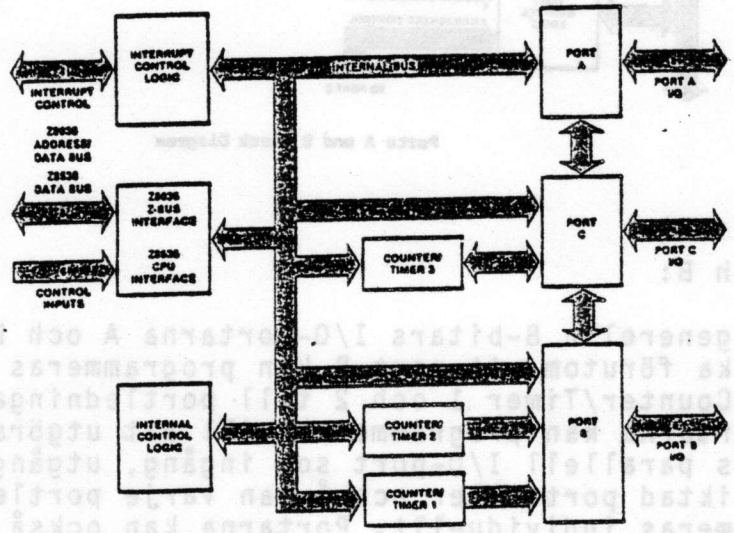
RDY "Ready" (Ingång, programmerbar hög eller låg)
Aktiveras av de yttre enheter som är anslutna till DMA:n för att indikera att de är klara att börja överföringen av data. DMA:n svarar med att begära busskontrollen.

WR "Write" (Dubbelriktad, aktiv låg högimpediv)
Används på samma sätt som $\overline{\text{RD}}$ fast vid skrivning.

Mer uppgifter om DMA:ns funktioner och hur den programmeras återfinns i leverantörens manualer.

5.2.4 CIO (Z8536)

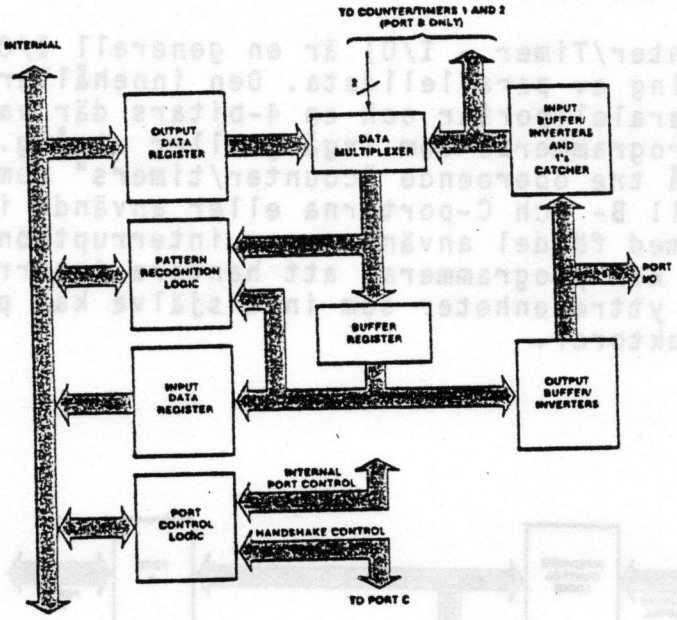
CIO:n (Counter/Timer - I/O) är en generell I/O-krets för hantering av parallelldata. Den innehåller två 8-bitars paralellportar och en 4-bitars där varje ledning kan programmeras som ingång eller utgång. I CIO:n finns också tre oberoende "counter/timers" som kan kopplas till B- och C-portarna eller används internt. CIO:n kan med fördel användas som interruptkontrollenhet då den kan programmeras att hantera interruptbegäran från yttre enheter som inte själva kan prestera interruptvektorer.



Z8036/Z8536 Z-CIO/CIO Block Diagram

CIO:ns huvudbeståndsdelar är följande:

- CPU-interface
- Tre I/O-portar (två 8-bitars, en 4-bitars)
- Tre "Counter/Timers"
- Interruptkontrollogik
- Intern Kontrollogik



Ports A and B Block Diagram

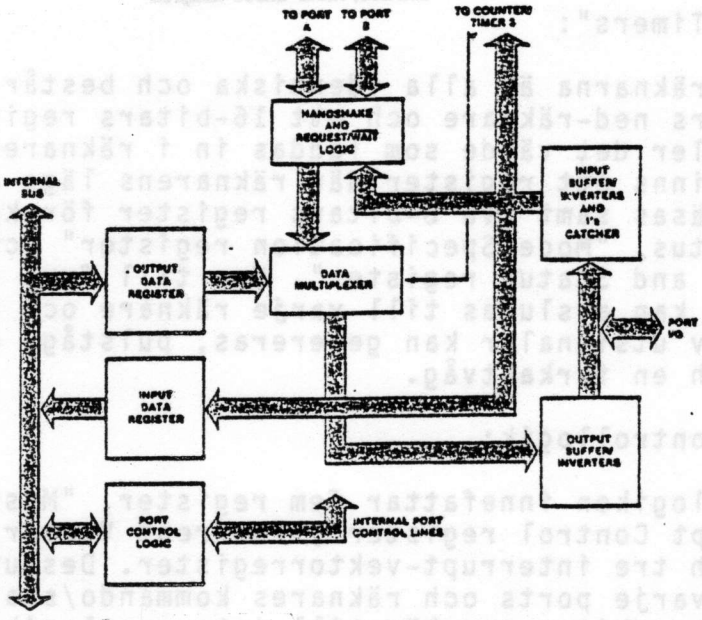
Port A och B:

De två generella 8-bitars I/O-portarna A och B är identiska förutom att port B kan programmeras att koppla Counter/Timer 1 och 2 till portledningarna. Båda portarna kan programmeras till att utgöra en 8-bitars parallell I/O-port som ingång, utgång eller dubbelriktad port eller också kan varje portledning programmeras individuellt. Portarna kan också programmeras att känna igen vissa portkombinationer och kan på så sätt generera olika interrupt beroende på mönstret. De kan också programmeras så att man får prioritetsskillnad mellan de olika ingångskombinationerna och bildar på så sätt även en prioritetsavkodare då den arbetar som interruptkontrollenhet. Portarna A och B kan också arbeta tillsammans så att de formar en 16-bitars port.

Varje port har 12 kontroll och statusregister som kontrollerar ovanstående funktioner. Tre av dessa register innehåller data som transporteras genom porten nämligen "Interrupt Data register", "Output Data register" och "Buffer register". "Mode Specification register" anger hur porten ska användas, alla åtta ledningarna som en byte eller var för sig, "Handshake Specification register" vilken typ av handskakning som ska användas eller om handskakning ej ska användas. Det mönster som exempelvis ska generera interrupt och kännas igen hanteras av de tre register "Pattern Polarity register", Pattern Transition register" och "Pattern Mask register".

Detaljerad information om hur varje portledning ska fungera finns i register "Data Path Polarity register", "Data Direction register" samt "Special Control register".

För varje port finns ett övergripande kommando- och statusregister och efter det att portens register programmerats är detta register normalt det enda som kontinuerligt behöver skrivas eller läsas.

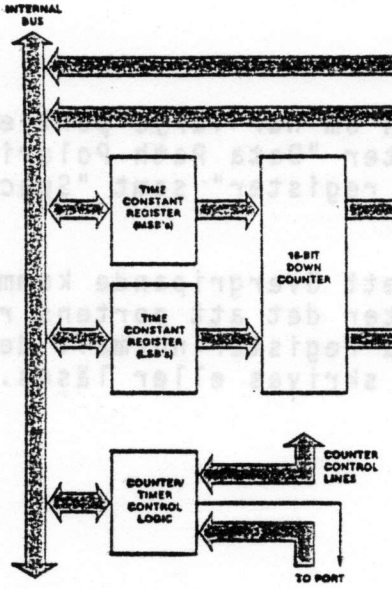


Port C Block Diagram

Port C

Port C:s funktion är i vissa fall beroende på portarna A och B. Portarna genererar handskakningssignaler när de andra portarna behöver det. En "REQUEST/$\overline{\text{WAIT}}$"-signal kan genereras så att överföring från eller till A och B-portarna kan synkroniseras mot DMA eller CPU. Varje ledning som ej används för handskakningen kan användas som I/O-ledning eller som utgång för "Counter/Timer" 3.

Då port C:s funktion normalt beror av port A och B finns endast tre register, de som definierar portledningarnas funktion, "Data Path Polarity register" "Data Direction register" och "Special I/O Control register".



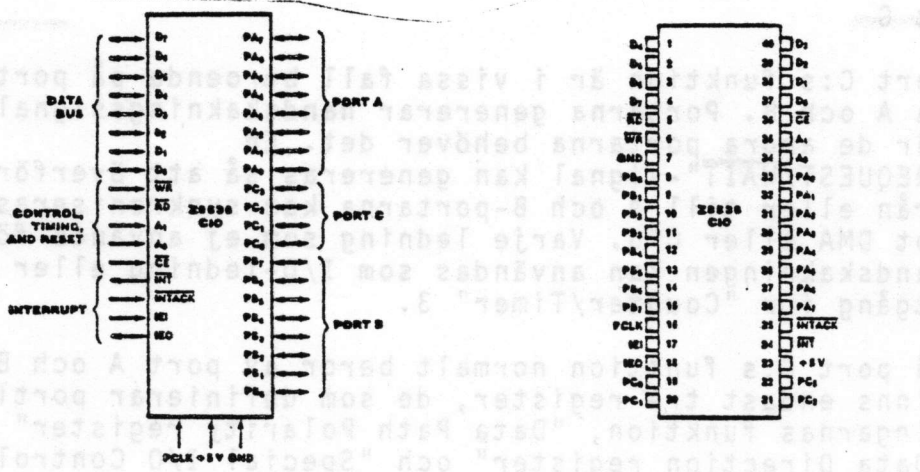
Counter/Timer Block Diagram

"Counter/Timers":

De tre räknarna är alla identiska och består av en 16-bitars ned-räknare och ett 16-bitars register som innehåller det värde som laddas in i räknaren. Dessutom finns ett register där räknarens läge alltid kan avläsas samt två 8-bitars register för kontroll och status, "Mode Specification register" och "C/T Command and Status register". Upp till fyra ledningar/port kan avslutas till varje räknare och tre olika typer av utsignaler kan genereras, pulståg, en enkelpuls och en fyrkantvåg.

Interrupkontrollogik:

Kontrollogiken innefattar fem register, "Master interrupt Control register", "Current Vector register" och tre interrupt-vektorregister. Dessutom innehåller varje ports och räknares kommando/status-register tre bitar som hör till interruptlogiken, "Interrupt Pending (IA)", "Interrupt Under Service (IUS)" och "Interrupt Enable (IE).



Signalbeskrivning:

A0-A1 Adressledningar (ingång). Dessa två ledningar används för att välja vilket register som CPU:n vill ha tillgång till - Port A:s dataregister, Port B:s dataregister, Port C:s dataregister eller Kontrollregistret.

$\overline{CE}$ "Chip Enable" (ingång, aktivt låg). Låg nivå aktiverar kretsen för skrivning eller läsning.

D0-D7 Data-bussen (dubbelriktad, högimpendiv). Dessa åtta ledningar används för att transportera data mellan CPU och CIO.

IEI "Interupt Enable In" (Ingång aktivt hög). Används tillsammans med IOE för att forma en interruptkedja. Hög nivå indikerar att ingen enhet av högre prioritet begärt interrupt.

IOE "Interrupt Enable Out" (utgång, aktiv hög). IOE är hög om IEI är hög och CIO:n ej gjort en interruptbegäran. IOE ansluts till IEI på närmaste enhet i interruptkedjan för att blockera möjligheten till interruptbegäran från lägre prioriterade enheter.

$\overline{INT}$ "Interupt Request" (utgång, aktiv låg). INT går låg då CIO:n gör en interruptbegäran.

$\overline{INTACK}$ "Interrupt Acknowledge" (ingång, alltid låg). Ges som svar på en interruptbegäran.

PA0-PA7 Port A:s I/O-signaler (dubbelriktade, högimpediva eller "open-drain"). Dessa åtta I/O-ledningar transporterar information mellan CIO:ns port A och externa enheter.

PB0-PB7 Port B:s I/O-signaler (dubbelriktade, högimpediva eller "open-drain"). Dessa I/O-ledningar transporterar information mellan CIO:ns port B och externa enheter. Den kan också användas att koppla "Counter/Timer" 1 och 2 till externa enheter.

PC0-PC3 Port C:s I/O-signaler (dubbelriktade, högimpediva eller "open-drain"). Dessa fyra I/O-ledningar användes normalt för handskakning för port A och B men kan också användas som generella I/O-signaler. "Counter/Timer" 3 är kopplad till denna port.

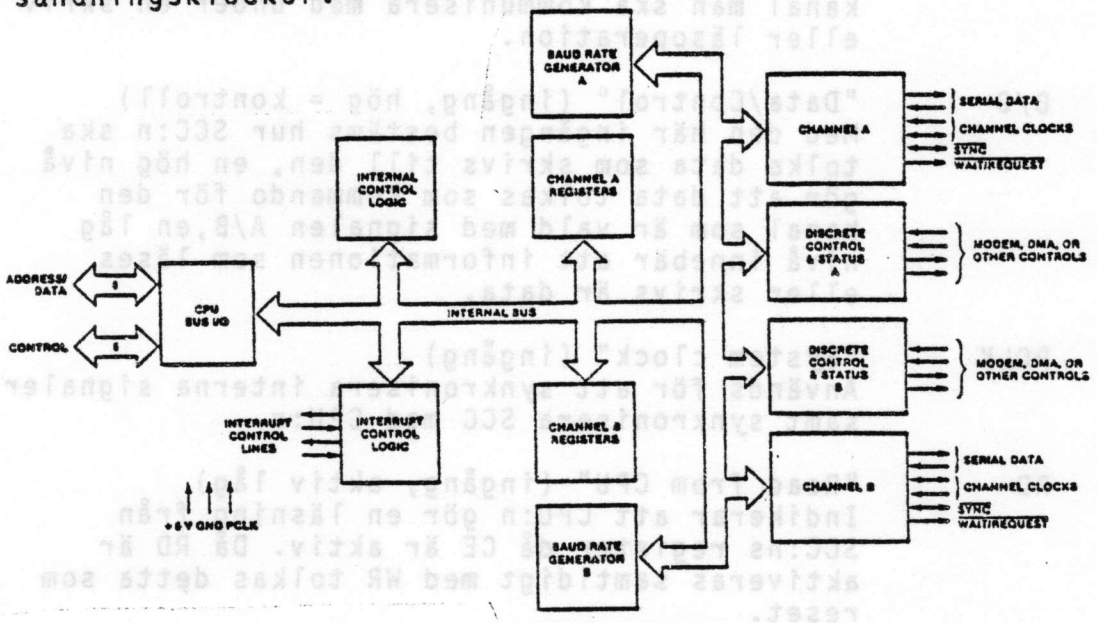
PCLK Periferiklocka (ingång). Används av intern logik samt av "Counter/Timers".

$\overline{RD}$ Lässtrob (ingång, aktiv låg). Indikerar att CPU:n läser i CIO:ns register. Då denna signal används tillsammans med $\overline{WR}$ tolkas detta som reset av CIO:n.

$\overline{WR}$ Skrivstrob (ingång, aktiv låg). Används då CPU:n skriver i CIO:ns register. Ger tillsammans med $\overline{RD}$ reset till CIO:n.

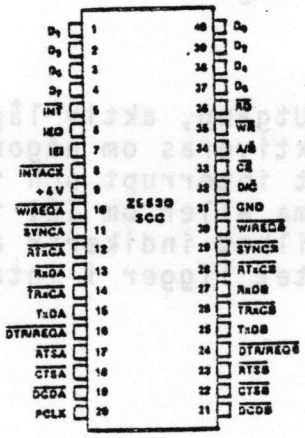
5.2.5 SCC (Z8530)

SCC:n är en generell seriekommunikationskrets med två kanaler som kan programmeras att hantera såväl tecken-,byte- som bit-orienterande protokoll. Den kan hantera alla former av asynkron och synkron kommunikation inklusive kodnings-format som NRZ/NRZI och FM. SCC:n innehåller också två baudrategeneratorer, en för varje kanal, vilka själva genererar den timing som behövs för mottagnings- och sändningsklockor.

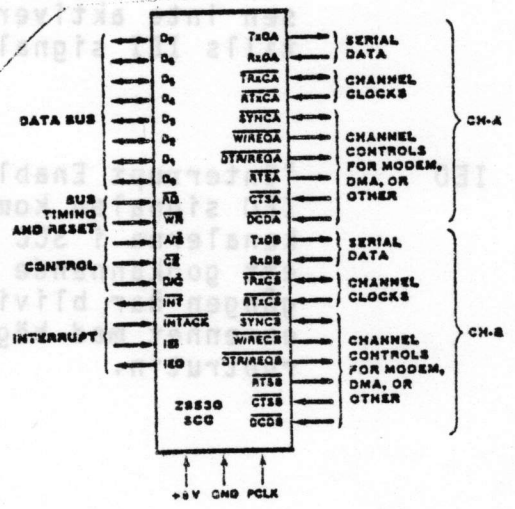


Z-SCC Block Diagram

Blockschema



Pin Designation for Z8530 SCC



Pinning

In- och utsignaler

D0-D7 Data buss. (dubbelriktad, tri-state)
Bussen används för att överföra data och kommando ord mellan CPU och SCC.

CE "Chip Enable" (ingång, aktiv låg)
Aktiverar SCC för att ta emot data eller kommando under en skrivoperation eller lägga ut data på bussen under en läsoperation.

A/B Kanalval A eller B (ingång, hög väljer kanal A). Med den här ingången bestämmer man vilken kanal man ska kommunisera med under en skriv eller läsoperation.

D/C "Data/Control" (ingång, hög = kontroll)
Med den här ingången bestäms hur SCC:n ska tolka data som skrivs till den, en hög nivå gör att data tolkas som kommando för den kanal som är vald med signalen A/B,en låg nivå innebär att informationen som läses eller skrivs är data.

PCLK "System clock" (ingång)
Används för att synkronisera interna signaler samt synkronisera SCC med CPU:n.

RD "Read from CPU" (ingång, aktiv låg)
Indikerar att CPU:n gör en läsning från SCC:ns register då CE är aktiv. Då RD är aktiveras samtidigt med WR tolkas detta som reset.

IEI "Interrupt Enable In" (ingång,aktiv låg)
Den här signalen används för att i ett system inordna olika enheter i en kedja med olika prioritet för att begära interrupt. En hög nivå talar om för kretsen att ingen annan enhet med högre prioritet har avgett interrupt till CPU:n, om den varit låg hade kretsen inte aktiverat sin INT signal utan väntat tills IEI signalen blivit hög.

IEO "Interrupt Enable Out". (Utgång, aktiv låg)
IEO signalen kommer att aktiveras om någon av kanalerna i SCC har avgett interrupt och fått ett godkännande på densamma eller om IEI ingången har blivit aktiv vilket indikerar att en enhet med högre prioritet ligger i interruptrutin.

INT "Interrupt Request". (Utgång, aktiv låg)
När SCC har programmerats för att avge interrupt, signalerar den en interruptbegäran till CPU:n.

W/REQ "Wait/Request". (Utgång)
Kan programmeras för att fungera antingen som en "WAIT"- signal för synkronisering med CPU:n eller som "REQUEST"-signal vid DMA överföring.

CTS "Clear To Send". (Ingång, aktiv låg)
Kan programmeras som "Auto enable" för sändning av data vilket innebär att CTS signalen måste vara låg för att data ska sändas på. Nivån på CTS kan läsas av via ett register. SCC:n kan programmeras så att den avger interrupt när ett omslag sker på CTS signalen.

DCD "Data Carrier Detect". (Ingång aktiv låg)
Funktionerna är samma som för CTS signalen med skillnaden att den kan användas för att göra mottagning av data möjlig.

RTS "Request To Send". (Utgång, aktiv låg)
I asynkronmod blir RTS hög när sändningsbufferten är tom, och låg när data har laddats i bufferten. Kan kontrolleras av CPU:n via ett register.

$\overline{\text{DTR/REQ}}$ "Data Terminal Ready/Request". (Utgång, aktiv låg)
följer det värde som DTR-biten är programmerad till. Kan också användas som "Request"-signal vid DMA-hantering.

$\overline{\text{INTACK}}$ "Interrupt Acknowledge". (Ingång aktiv låg)
Indikerar att CPU:n accepterat en interruptbegäran.

RTxC "Receive/Transmit Clocks". (Ingång aktiv låg)
Mottagningsklocka men kan programmeras för användning som sändningsklocka, insignal till baudrategeneratorn samt som klocka till PLL:en. Kan också användas tillsammans med SYNC för att bilda ingång för en kristall till oscillator.

TRxC "Transit/Receive Clock". (In-eller utgång, aktiv låg) Sändningsklocka men kan som med RTxC också programmeras till liknande funktioner.

SYNC "Synkronization" (Ingång/Utgång, aktiv låg)
Används som in- eller utgång eller som del i kristalloscillatorn. Vid asynkronmottagning fungerar den likartat med CTS och DCD under det att den vid synkron kommunikation används för synkronisering.

RxD "Recive Data" (Ingång, aktiv hög)
Denna signal innehåller mottagna data.

TxD "Transmit Data" (Utgång, aktiv hög)
Denna signal innehåller sänddata.

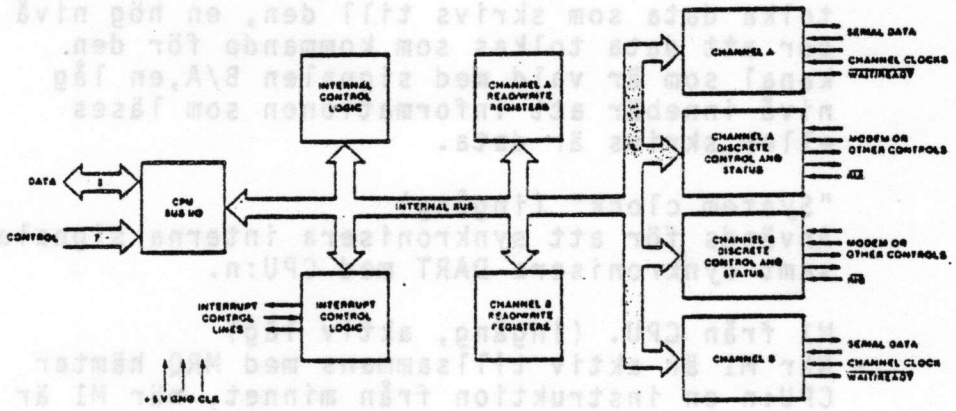
5.2.6 DART

DART (Dual Asynkron Reciever Transmitter) är en seriell kommunikationskrets med två progarammerbara kanaler och som är konstruerade för att klara ett brett område av seriell kommunikation i ett mikrodatorsystem. Grundfunktionerna är att omvandla data från pararellell till serieform eller från serie till parallelldata. Några av funktionerna i DART är följande:

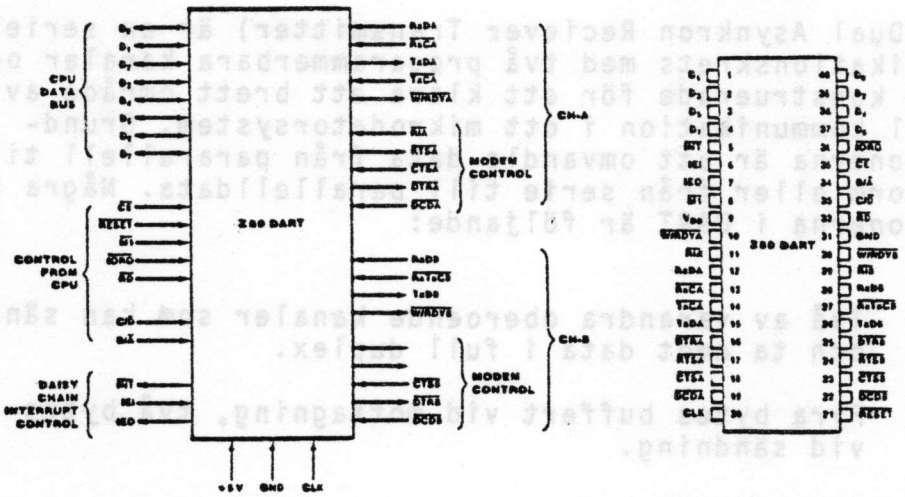
- Två av varandra oberoende kanaler som kan sända och ta emot data i full duplex.
- Fyra bytes buffert vid mottagning, två bytes vid sändning.

Vid asynkron överföring:

- 5,6,7,eller 8 bit/tecken
- 1,1 1/2 eller 2 stoppbitar
- Jämn, udda eller ingen paritet
- Paritetsfel, överskrivningsdetektering
- Klockfrekvensen kan vara bithastigheten x1,x16,x32 och x64
- Separata modemkontrollsignaler för båda kanalerna
- Modemstatus kan övervakas



Block Diagram



In- och utsignaler

DO-D7 Databuss. (dubbelriktad, tri-state)
Bussen används för att överföra data och kommando ord mellan CPU och DART.

CE "Chip Enable" (ingång, aktiv låg)
Aktiverar DART för att ta emot data eller kommando under en skrivoperation eller lägga ut data på bussen under en läsoperation.

B/A Kanalval A eller B (ingång, hög väljer kanal B). Med den här ingången bestämmer man vilken kanal man ska kommunicera med under en skriv eller läsoperation.

C/D "Control/Data" (ingång, hög = kontroll)
Med den här ingången bestäms hur DART:en ska tolka data som skrivs till den, en hög nivå gör att data tolkas som kommando för den kanal som är vald med signalen B/A,en låg nivå innebär att informationen som läses eller skrivs är data.

CLK "System clock" (ingång)
Används för att synkronisera interna signaler samt synkronisera DART med CPU:n.

M1 M1 från CPU. (ingång, aktiv låg)
När M1 är aktiv tillsammans med MRQ hämtar CPU:n en instruktion från minnet, när M1 är aktiv tillsammans med IORQ indikerar det ett godkännande på en interruptbegäran. DART:n kommer då att placera en interruptvektor på databussen om någon av kanalerna har aktiverat INT signalen.

IORQ "Input/Output Request from CPU". (ingång, aktiv låg)
IORQ tillsammans med CE och RD signalerna överför data eller kontrollord mellan CPU och DART:n. När CE, IORQ och RD aktiva kommer data/kontrollord att läsas från adresserad kanal. När CE, IORQ är aktiva är RD inaktiv kommer data/kontrollord att skrivas till DART.
Om IORQ och M1 är aktiva samtidigt innebär det att godkännade på en avbrottsbegäran från CPU, och om kretsen har genererat en INT signal ska dem lägga ut en interruptvektor på bussen.

RD "Read from CPU". (ingång, aktiv låg)
Aktiverar tillsammans med CE, IORQ en läsoperation eller om RD är hög en skriv-operation.

IEI "Interrupt Enable In" (ingång,aktiv låg)
Den här signalen används för att i ett system inordna olika enheter i en kedja med olika prioritet för att begära interrupt. En hög nivå talar om för kretsen att ingen annan enhet med högre prioritet har avgett interrupt till CPU:n, om den varit låg hade kretsen inte aktiverat sin INT signal utan väntat tills IEI signalen blivit hög.

IEO "Interrupt Enable Out". (Utgång, aktiv låg)
IEO signalen kommer att aktiveras om någon av kanalerna i DART har avgett interrupt och fått en godkännande på densamma eller om IEO ingången har blivit aktiv vilket indikerar att en enhet med högre prioritet ligger i interruptrutin.

RESET "Reset" (Ingång, aktiv låg)
En aktiv RESET inaktiverar både sändning och mottagning, lägger TxD signalerna och modem-signalerna på hög nivå. Registren i DART:n måste sedan omprogrammeras.

INT "Interrupt Request". (Utgång, aktiv låg)
När DART har programmerats för att avge interrupt, kommer en interruptbegäran till CPU:n eller READY signal vid DMA överföring.

W/RDY "Wait/Ready". (Utgång)
Kan programmeras för att fungera antingen som en "WAIT"-signal för synkronisering med CPU:n eller "READY"-signal vid DMA-överföring

CTS "Clear To Send". (Ingång, aktiv låg)
Kan programmeras för "Auto enable" för sändning av data vilket innebär att CTS signalen måste vara låg för att data ska sändas på TxD.

Nivån på CTS kan läsas av via ett register. DART:n kan programmeras så att den avger interrupt när ett omslag sker på CTS signalen.

DCD "Data Carrier Detect". (Ingång aktiv låg)
Funktionerna är samma som för CTS signalen med skillnaden att den kan användas för att göra mottagning av data möjlig.

RxD "Receive Data".(Ingång)
Ingång för seriedata.

TxD "Transmit Data". (Utgång)
Utgång för seriedata.

RxC "Receiver Clock" (Ingång)
I asynkronmod kan mottagningsklockan vara 1,16,32, eller 64 gånger överföringshastigheten av data. RxD ingången avkänns under den positiva flanken på RxC signalen.

TxC "Transmit Clock" (Ingång)
I asynkronmod kan sändningsklockan vara 1,16,32 eller 64 gånger överföringshastigheten av data. Klockan delas ned intern i kretsen och multipeln måste vara samma för både RxC och TxC. TxD utgången växlar data på den negativa flanken på TxC.

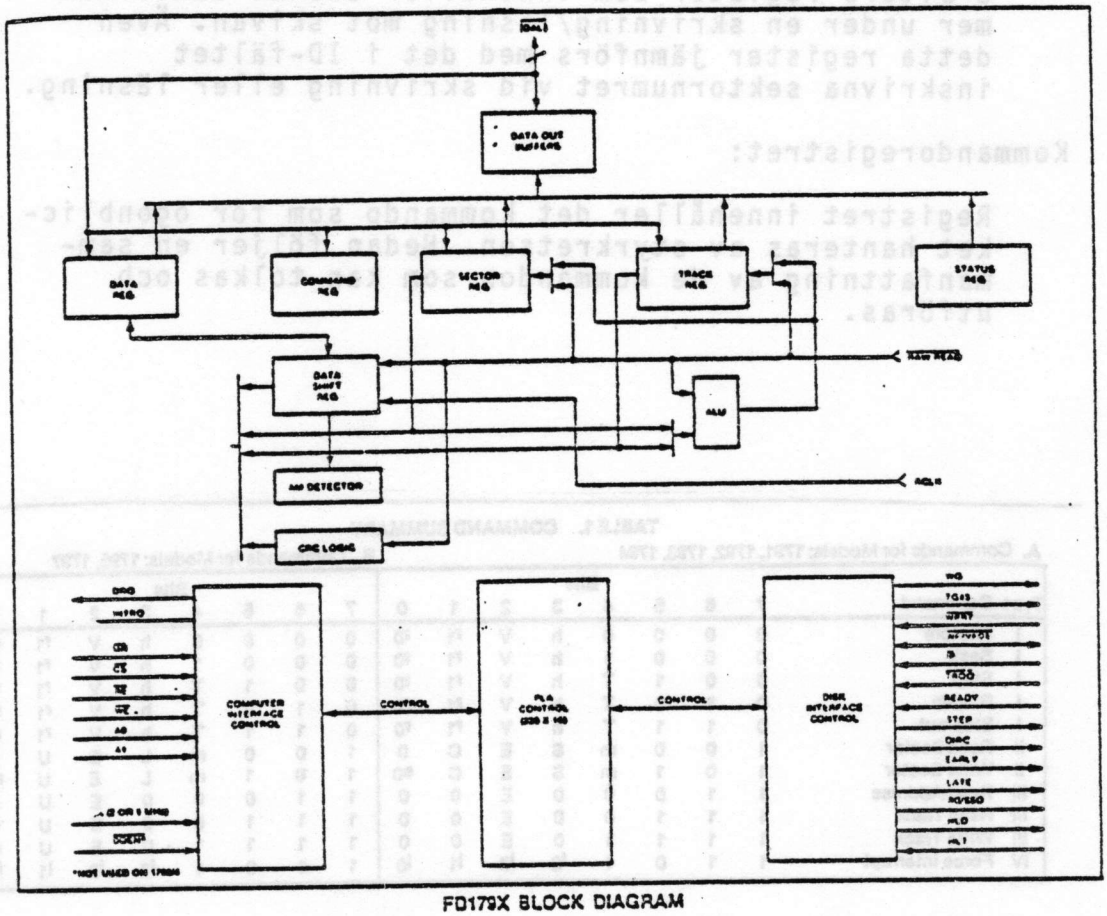
RTS "Request To Send" (Utgång, aktiv låg)
I asynkronmod blir RTS hög när sändningsbufferten är tom, och låg när data har laddats i bufferten. Kan kontrolleras av CPU:n via ett register.

DTR "Data Terminal Ready". (Utgång, aktiv låg)
Kontrolleras av CPU:n via ett register.

RI Ring Indikator. (Ingång, aktiv låg)
I asynkronmod fungerar den liknande som CTS och DCD och kan läsas av CPU:n via ett register.

5.2.7 FD 1797

FD 1797 är en generell periferikrets för styrning av flexskiveenheter. Den sitter som länk mellan CPU:ns bussystem varifrån den får kommandon om vad som ska läsas eller skrivas och omvandlar dess till styrsignaler för flexskiveenheterna. Kommunikationen med CPU:n sker via ett antal register som talar om vilket spår och sektor som data ska skrivas till eller läsas från samt kommandosregister där CPU:n talar om vad som ska utföras och dataregister där data på väg till eller från flexskivan mellanlagras. Här finns också ett statusregister där CPU:n kan hämta uppgifter om flexskivestatus samt om något fel uppstått under läsningen/-skrivningen.



FD179X BLOCK DIAGRAM

Dataskiftregister:

8-bitars register som samlar ihop seriedata från dataingången "$\overline{\text{RAW READ}}$" vid läsning från flexskivan samt lämnar data till den seriella utgången $\overline{\text{WD}}$ vid skrivning.

Dataregister:

Fungerar som buffertregister för data som ska transporteras till eller från skriftregistert vid skrivning eller läsning. Då ett sökkommando utförs innehåller det adressen till det önskade spåret.

Spårregister:

8-bitars register som innehåller spårnumret till aktuellt spår. Registret uppdateras för varje spårförflyttning som flexskiveenhetens skriv/läshuvud gör. Innehållet i registret jämnförs vid skrivning eller läsning med det spårnummer som finns inskrivet i flexskivans olika ID-fält.

Sektorregister:

8-bitars register som innehåller önskat sektornummer under en skrivning/läsning mot skivan. Även detta register jämnförs med det i ID-fältet inskrivna sektornumret vid skrivning eller läsning.

Kommandoregistret:

Registret innehåller det kommando som för ögonblicket hanteras av styrkretsen. Nedan följer en sammanfattning av de kommandon som kan tolkas och utföras.

TABLE 1. COMMAND SUMMARY

A. Commands for Models: 1791, 1792, 1793, 1794

Type	Command	Bits 7	6	5	4	3	2	1	0
I	Restore	0	0	0	0	h	V	r_1	r_0
I	Seek	0	0	0	1	h	V	r_1	r_0
I	Step	0	0	1	T	h	V	r_1	r_0
I	Step-In	0	1	0	T	h	V	r_1	r_0
I	Step-out	0	1	1	T	h	V	r_1	r_0
II	Read Sector	1	0	0	m	S	E	C	0
II	Write Sector	1	0	1	m	S	E	C	a_0
III	Read Address	1	1	0	0	0	E	0	0
III	Read Track	1	1	1	0	0	E	0	0
III	Write Track	1	1	1	1	0	E	0	0
IV	Force Interrupt	1	1	0	1	I_3	I_2	I_1	I_0

B. Commands for Models: 1795, 1797

Type	Command	Bits 7	6	5	4	3	2	1	0
I	Restore	0	0	0	0	h	V	r_1	r_0
I	Seek	0	0	0	1	h	V	r_1	r_0
I	Step	0	0	1	T	h	V	r_1	r_0
I	Step-In	0	1	0	T	h	V	r_1	r_0
I	Step-out	0	1	1	T	h	V	r_1	r_0
II	Read Sector	1	0	0	m	L	E	U	0
II	Write Sector	1	0	1	m	L	E	U	a_0
III	Read Address	1	1	0	0	0	E	U	0
III	Read Track	1	1	1	0	0	E	U	0
III	Write Track	1	1	1	1	0	E	U	0
IV	Force Interrupt	1	1	0	1	I_3	I_2	I_1	I_0

FLAG SUMMARY

TABLE 2. FLAG SUMMARY

Command Type	Bit No(s)		Description
I	0, 1	r_1 r_0 = Stepping Motor Rate See Table 3 for Rate Summary	
I	2	V = Track Number Verify Flag	V = 0, No verify V = 1, Verify on destination track
I	3	h = Head Load Flag	h = 1, Load head at beginning h = 0, Unload head at beginning
I	4	T = Track Update Flag	T = 0, No update T = 1, Update track register
II	0	a_0 = Data Address Mark	a_0 = 0, FB (DAM) a_0 = 1, F8 (deleted DAM)
II	1	C = Side Compare Flag	C = 0, Disable side compare C = 1, Enable side compare
II & III	1	U = Update SSO	U = 0, Update SSO to 0 U = 1, Update SSO to 1
II & III	2	E = 15 MS Delay	E = 0, No 15 MS delay E = 1, 15 MS delay
II	3	S = Side Compare Flag	S = 0, Compare for side 0 S = 1, Compare for side 1
II	3	L = Sector Length Flag	LSB's Sector Length in ID Field 00: L = 0: 256; L = 1: 128 01: L = 0: 512; L = 1: 256 10: L = 0: 1024; L = 1: 512 11: L = 0: 128; L = 1: 1024
II	4	m = Multiple Record Flag	m = 0, Single record m = 1, Multiple records
IV	0-3	I_x = Interrupt Condition Flags I_0 = 1 Not Ready To Ready Transition I_1 = 1 Ready To Not Ready Transition I_2 = 1 Index Pulse I_3 = 1 Immediate Interrupt, Requires A Reset I_3-I_0 = 0 Terminate With No Interrupt (INTRQ)	

*NOTE: See Type IV Command Description for further information.

Statusregister:

8-bitarsregister som innehåller flexskivestatus. Registrets funktion varieras beroende på vilket kommando som utförs. Nedan följer en sammanställning över registerfunktionen.

STATUS REGISTER SUMMARY

BIT	ALL TYPE I COMMANDS	READ ADDRESS	READ SECTOR	READ TRACK	WRITE SECTOR	WRITE TRACK
S7	NOT READY	NOT READY	NOT READY	NOT READY	NOT READY	NOT READY
S6	WRITE PROTECT	0	0	0	WRITE PROTECT	WRITE PROTECT
S5	HEAD LOADED	0	RECORD TYPE	0	WRITE FAULT	WRITE FAULT
S4	SEEK ERROR	RNF	RNF	0	RNF	0
S3	CRC ERROR	CRC ERROR	CRC ERROR	0	CRC ERROR	0
S2	TRACK 0	LOST DATA	LOST DATA	LOST DATA	LOST DATA	LOST DATA
S1	INDEX PULSE	DRQ	DRQ	DRQ	DRQ	DRQ
S0	BUSY	BUSY	BUSY	BUSY	BUSY	BUSY

STATUS FOR TYPE I COMMANDS

BIT NAME	MEANING
S7 NOT READY	This bit when set indicates the drive is not ready. When reset it indicates that the drive is ready. This bit is an inverted copy of the Ready input and logically 'ored' with MR.
S6 PROTECTED	When set, indicates Write Protect is activated. This bit is an inverted copy of $\overline{\text{WRPT}}$ input.
S5 HEAD LOADED	When set, it indicates the head is loaded and engaged. This bit is a logical "and" of HLD and HLT signals.
S4 SEEK ERROR	When set, the desired track was not verified. This bit is reset to 0 when updated.
S3 CRC ERROR	CRC encountered in ID field.
S2 TRACK 00	When set, indicates Read/Write head is positioned to Track 0. This bit is an inverted copy of the $\overline{\text{TR00}}$ input.
S1 INDEX	When set, indicates index mark detected from drive. This bit is an inverted copy of the $\overline{\text{IP}}$ input.
S0 BUSY	When set command is in progress. When reset no command is in progress.

CRC-logik:

Beräknar och kontrollerar en sektors checksumma. Vid läsning från flexskivan kontrolleras den inskrivna checksumman med den genom dataflödet beräknade. Vid skrivning beräknas checksumman och skrivs sedan till den aktuella sektorn.

ALU-logik:

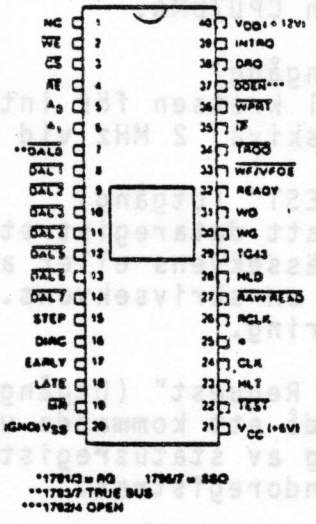
Används som seriell komparator samt upp- och nedräknare. Den används för att uppdatera register och för att jämnföra det på flexskivan inskrivna ID - fältet.

"Timing" och kontroll:

Alla kontrollsignaler från styrkretsen till både CPU och flexskiveinterface samt intern "timing" genereras av denna logik.

AM-detektor:

Kontrollerar adressmärken från ID, data och index under en skriv- eller lässekvens.



PIN DESIGNATION

$\overline{MR}$ "MASTER RESET" (Ingång, aktiv låg)
Vid låg nivå nollställs 1797:an och värdet 03H laddas i kommandoregistret. "Not Ready-biten" i statusregistret nollställs och då signalen återgår till hög nivå efter reset utförs ett RESTORE-kommando och värdet 01H laddas i sektorregistret.

VSS Jordanslutning.

VCC +5V matningspänning.

VDD +12V matningsspänning.

$\overline{WE}$ "Write Enable" (Ingång, aktiv låg)
Skrivstrob från CPU:n, talar om att databussen innehåller aktivt data som ska skrivas till kretsen.

$\overline{CS}$ "Chip Select" (Ingång aktiv låg)
Aktiverar kretsen för CPU-kommunikation.

$\overline{RE}$ "Read Enable" (Ingång, aktiv låg)
Lässtrob från CPU, får kretsen att lägga ut data på databussen till CPU:n.

A0-A1 "Register Select Lines" (Ingångar)
Väljer register vid sändning/mottagning enligt följande tabell.

$\overline{CS}$	A1	A0	$\overline{RE}$	$\overline{WE}$
0	0	0	Statusregister	Kommandoregister
0	0	1	Spårregister	Spårregister
0	1	0	Sektorregister	Sektorregister
0	1	1	Dataregister	Dataregister

DAL0-7 "Data Access Lines" (dubbelriktad)
8-bitars databuss för överföring av data mellan 1797:an och CPU/DMA.

CLK "CLOCK" (Ingång)
Klocka till kretsen för intern timing, 1MHZ vid 130mm flexskiva, 2 MHz vid 200mm flexskiva.

DRQ "DATA REQUEST" (Utgång)
Indikerar att dataregistret innehåller data under en lässekvens eller att dataregistret är tomt under en skrivsekvens. Används normalt vid DMA-överföring.

INTRQ "Interrupt Request" (Utgång)
Aktiveras då ett kommando utförs och återställs vid läsning av statusregistret eller skrivning till kommandoregistret.

STEP "STEP" (Utgång)
Ger en puls för varje steg man önskar förflytta flexskiveenhetens skriv/läs-huvud.

DIRC "Direction" (Utgång)
Anger stegningsriktningen vid förflyttning av flexskiveenhetens skriv/läs-huvud.

EARLY "EARLY" (Utgång)
Används tillsammans med "LATE" för att ange förkompenseringen vid skrivning på flexskivan.

LATE "LATE" (Utgång)
Se "EARLY".

HLT "Head Load Timing" (Ingång, aktiv låg)
Används för att indikera att flexskiveenhetens skriv/läs-huvud är klart för skrivning/läsning.

SSO "Side Select Output" (Utgång)
Anger vilket av de två skriv/läs-huvudena som ska aktiveras vid dubbelsidiga flexskiveenheter.

RCLK "Read Clock" (Ingång)
Klocka från dataseparatorn som bildats av inkommande data.

$\overline{\text{RAW READ}}$ (Ingång)
Seriedata från flexskiveenheten.

HLD "Head Load" (Utgång)
Styr när skriv/läs-huvudet ska läggas an mot flexskivan.

TG 43 "Tracks Greater Than 43" (Utgång)
Talar om för flexskiveenheten att något spår över 44 adresseras.

WG "Write Gate" (Utgång)
Aktiveras före skivning av data till flexskiveenheten.

WD "Write Data" (Utgång)
Seriedata som ska skrivas till flexskivan.

READY "READY" (Ingång)
"Ready"-signal från flexskiveenheten som indikerar att den är klar för en skriv-eller lässekvens.

$\overline{\text{WF/VFOE}}$ "Write Fault/VFO Enable" (Dubbelriktad)
Fungerar som ingång då WG är aktiv och kan då känna av om något fel inträffar vid skrivning till flexskivan. Då WG är låg fungerar den som enable till en extern dataseparator.

$\overline{\text{TR00}}$ "Track 00" (Ingång)
Indikerar att skriv/läs-huvudet har sin position över spår 0.

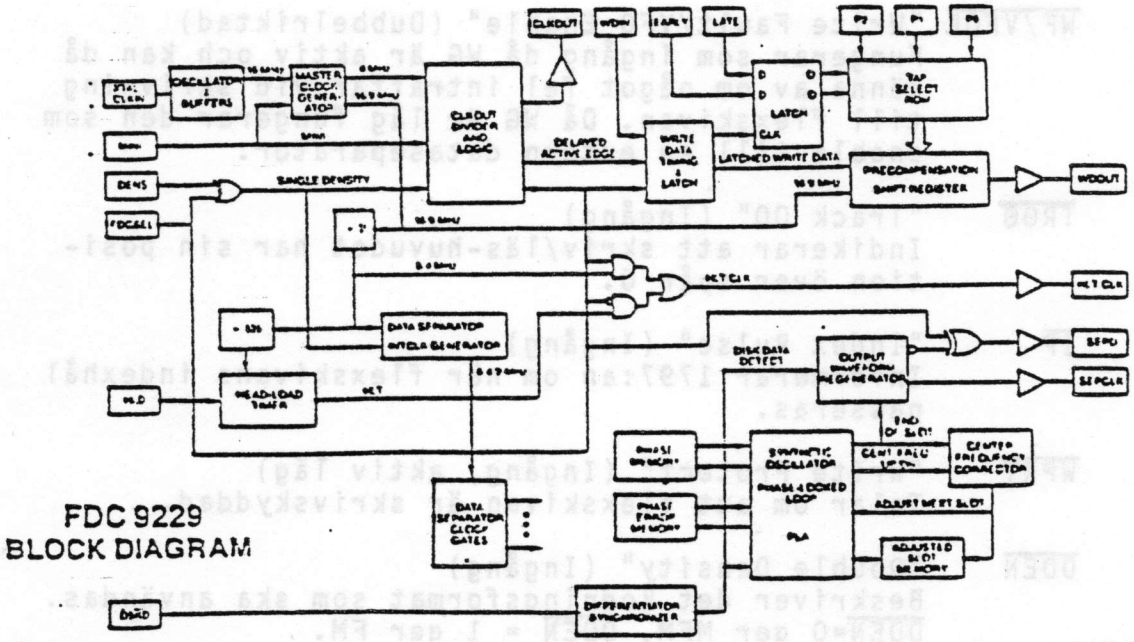
$\overline{\text{IP}}$ "Index Pulse" (Ingång)
Informerar 1797:an om när flexskivans indexhål passeras.

$\overline{\text{WPRT}}$ "Write Protect" (Ingång, aktiv låg)
Talar om att flexskivan är skrivskyddad.

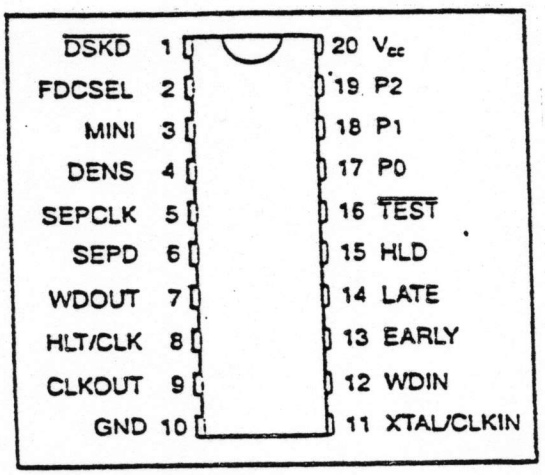
$\overline{\text{DDEN}}$ "Double Density" (Ingång)
Beskriver det kodningsformat som ska användas. $\overline{\text{DDEN}}$=0 ger MFM, $\overline{\text{DDEN}}$ = 1 ger FM.

5.2.8 FDC 9229 B

Dataseparatorn används tillsammans med styrkretsen FD1797 för att forma flexskiveinterface. Dess huvuduppgift är att separera data och klocka ut den bitström som levereras av flexskiveenheten vid en läsning från flexskivan. Den ska också bearbeta data som ska skrivas till flexskivan så att de får rätt skrivkompensering samt leverera klocka till kontrollkretsen beroende på flexskivetyp.



Blockschema



Signaler

$\overline{\text{DSKD}}$ "Disk Data" (Ingång)
Seriella data från flexskiveenheten.

FDCSEL "Floppy Disk Control Select" (Ingång)
Ställer om dataseparatorn beroende på vilken styrkrets som används. (I ABC 1600 = 0 = FDC1797)

MINI "Minifloppy" (Ingång)
Talar om vilken flexskivetyp som används, 130mm eller 200mm. Påverkan tillsammans med DENS-signalen klockdelningen till dataseparingen samt klockfrekvensen ut till FD1797.

DENS "Density" (Ingång)
Talar om vilket kodningsformat som används, FM eller MHF. Används tillsammans med MINI för att påverka klockdelningen till dataseparatorn.

SEPCLK "Separate Clock" (Utgång)
Klocka som styrs av dataströmmen från indata. Separeras från data i dataseparatorn.

SEPD "Separate Data" (Utgång)
Data som separerats ur flexskivedata, levereras till styrkretsen.

WDOUT "Write Data Out" (Utgång)
Data som skrives till flexskivan. Dessa data är nu behandlade med aktuell förkompensering.

HLT "Head Load Timing" (Utgång)
Aktiveras av HLD men först efter den fördröjningstid 40/80ms som ges av kombinationen DENS/MINI.

CLKOUT "Clock Out" (Utgång)
Klocka till styrkretsen FD1797.

CLKIN "Clock In" (Ingång)
Klocka från systemet som delas i FDC9229B för att generera klocka ut och till att användas internt.

WDIN "Write Data In" (Ingång)
Data från FD1797 som efter det att skrivkompenseringen lagts på, sänds till flexskivan.

EARLY (Ingång)
Ger tillsammans med LATE utgångläget för skrivkompensering.

LATE (Ingång) Se EARLY

HLD "Head Load" (Ingång)
Styr HLT efter viss fördröjning.

P0-2 "Precompensation" (Ingångar)
Kombinationen av dessa tre signaler avgör efter vilken tidsfaktor skrivkompenseringen ska ske.

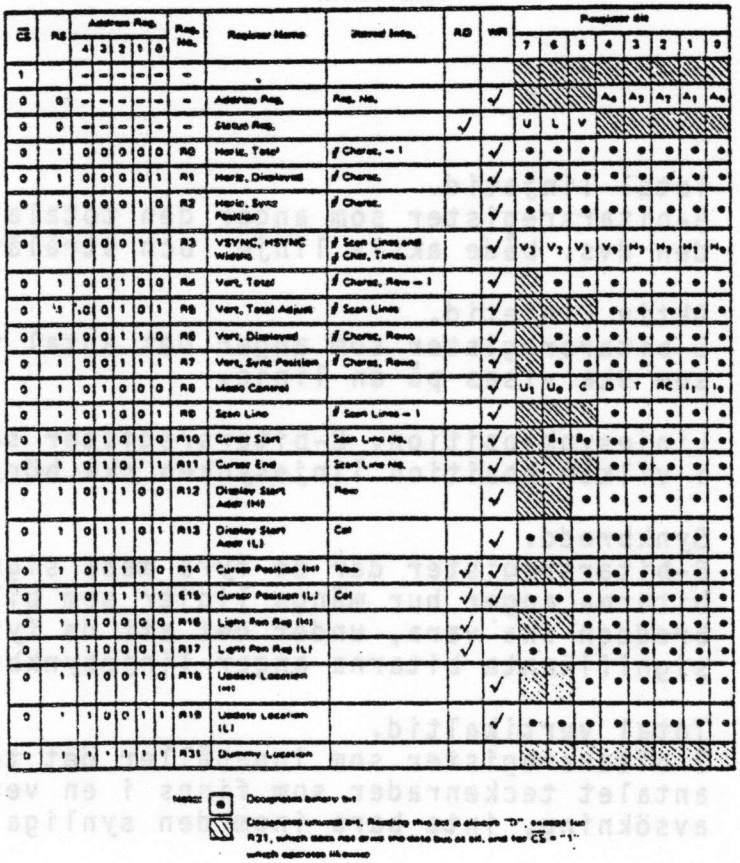
Tabell skrivkompensering

MINI	P2	P1	P0	PRECOMP VALUE
0	0	0	0	0 ns
0	0	0	1	62.5 ns
0	0	1	0	125 ns
0	0	1	1	187.5 ns
0	1	0	0	250 ns
0	1	0	1	250 ns
0	1	1	0	312.5 ns
0	1	1	1	312.5 ns

MINI	P2	P1	P0	PRECOMP VALUE
1	0	0	0	0 ns
1	0	0	1	125 ns
1	0	1	0	250 ns
1	0	1	1	375 ns
1	1	0	0	500 ns
1	1	0	1	500 ns
1	1	1	0	625 ns
1	1	1	1	625 ns

5.2.9 CRTC 6845 E

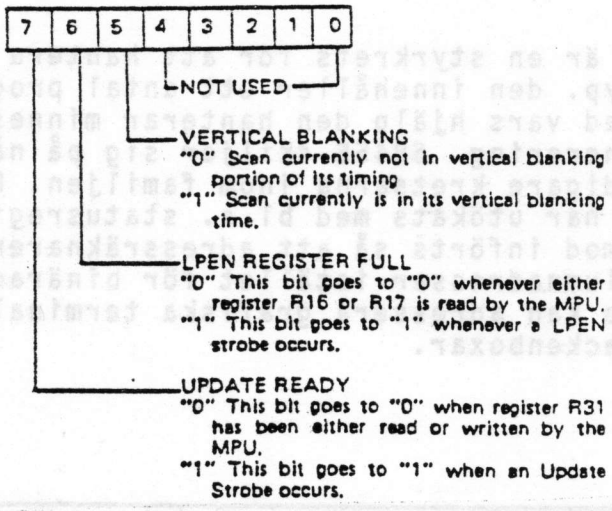
CRTC 6845E är en styrkrets för att hantera bildskärmar av rastertyp. den innehåller ett antal programmerbara register med vars hjälp den hanterar minnesadressering och synkgenerering. 6845E skiljer sig på några punkter från de tidigare kretsarna inom familjen. Registeruppsättningen har utökats med bl.a. statusregister, samt att en ny mod införts så att adressräknaren kan ange rad och kolumnadresser istället för binäradress så att man lättare kan adressera grafiska terminaler utan tanke på teckenboxar.



Register:

AR Adressregister.
5-bitarsregister som innehåller adressen till det register som ska uppdateras. Adressregistret skrivs då både RS och $\overline{CS}$ har låg nivå under det att då RS går hög används adressregistrets innehåll för att peka ut aktuellt register.

SR Statusregister.
3-bitarsregister för statusindikering enl. följande:



R0 Total linjetid.
8-bitarsregister som anger den totala linjetiden dvs. både aktiv linje och stråtergång.

R1 Aktiv Linjetid.
8-bitarsregister som anger det antal tecken som ska visas på en linje.

R2 Linjesynkposition. 8-bitarsregister som anger i vilken position linjesynken ska börja.

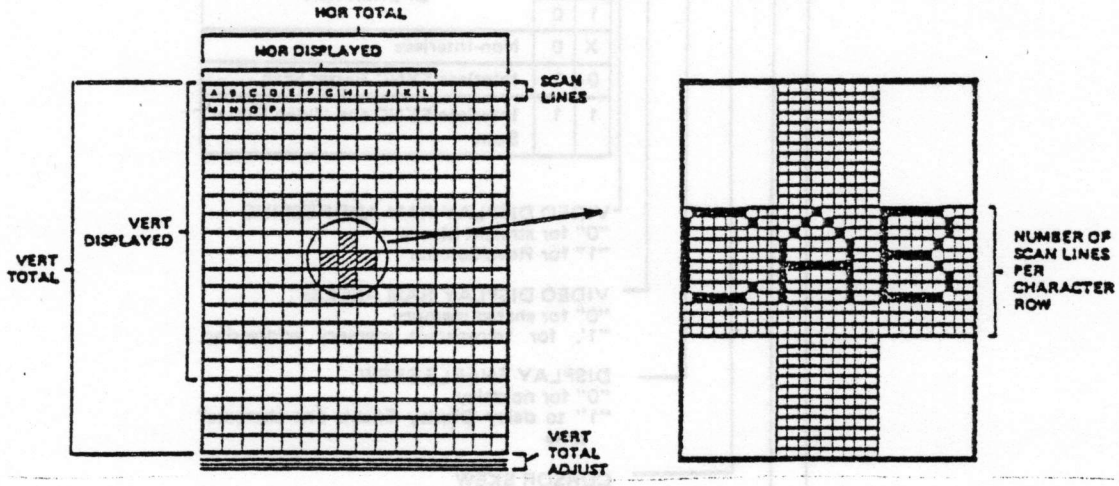
R3 Synkbredd.
8-bitarsregister där de fyra mest signifikanta bitarna anger hur många linjer som bildsynkbredden ska vara, under det att de fyra minst signifikanta bitarna anger linjesynkbredden.

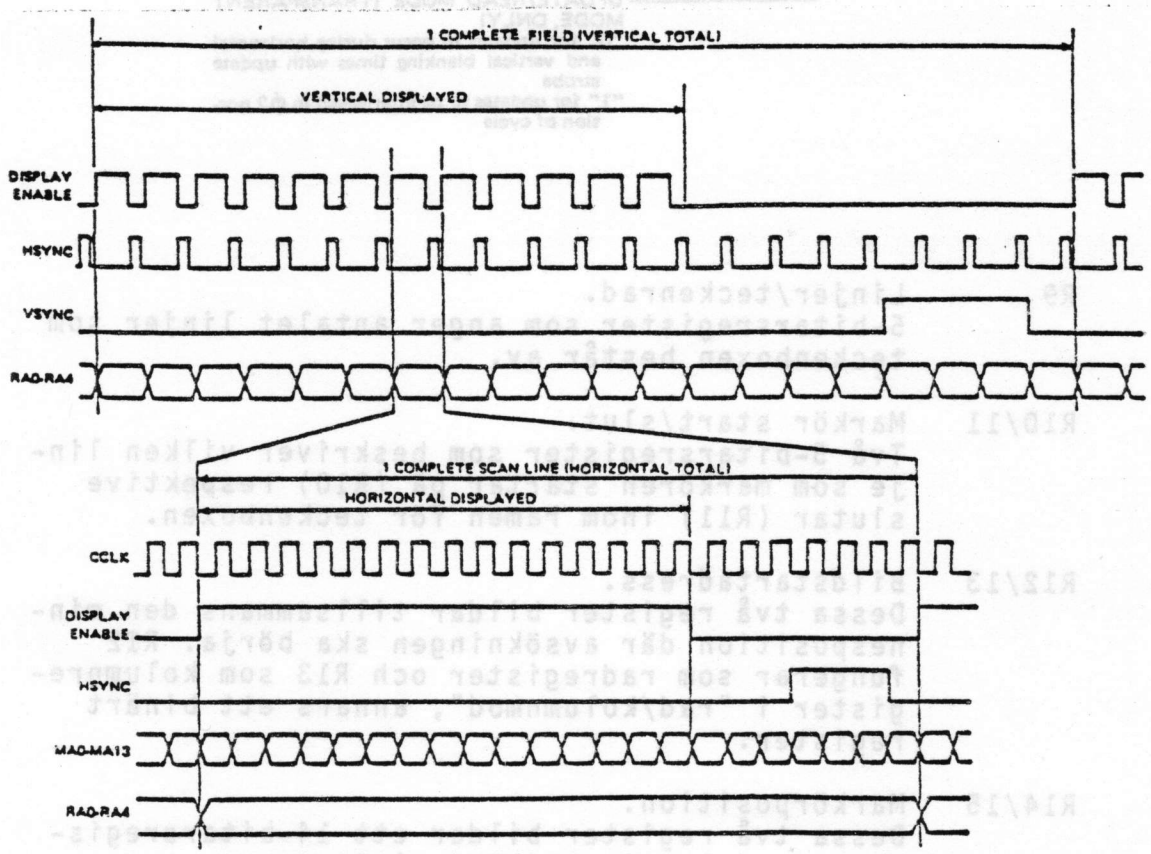
R4 Total vertikaltid.
7-bitarsregister som innehåller det totala antalet teckenrader som finns i en vertikal avsökning, inte bara inom den synliga delen.

R5 Justering av vertikal totaltid.
5-bitars register som innehåller det antal linjer som ska adderas till teckenraderna i R4 för att få den totala tiden för varje bildväxling.

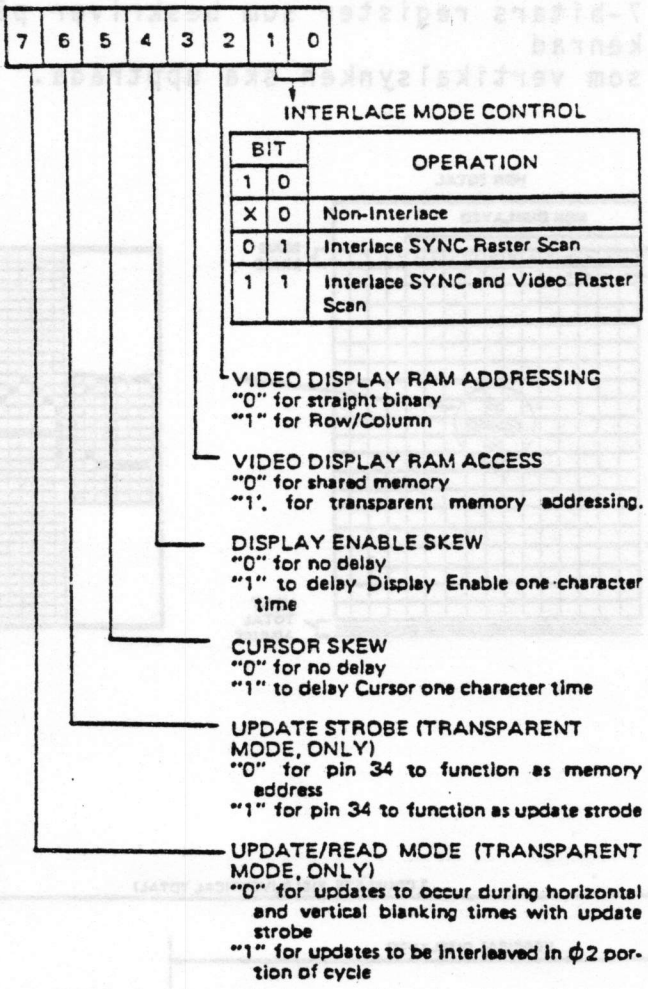
R6 Synlig bildyta.
7-bitars register som talar om det antal teckenrader som ska visas på bildskärmen.

R7 Vertikal synkposition.
7-bitars register som beskriver på vilken teckenrad
som vertikalsynken ska uppträda.





R8 "Mode Control".
Registerinformation enl. följande.



R9 Linjer/teckenrad.
5-bitarsregister som anger antalet linjer som teckenboxen består av.

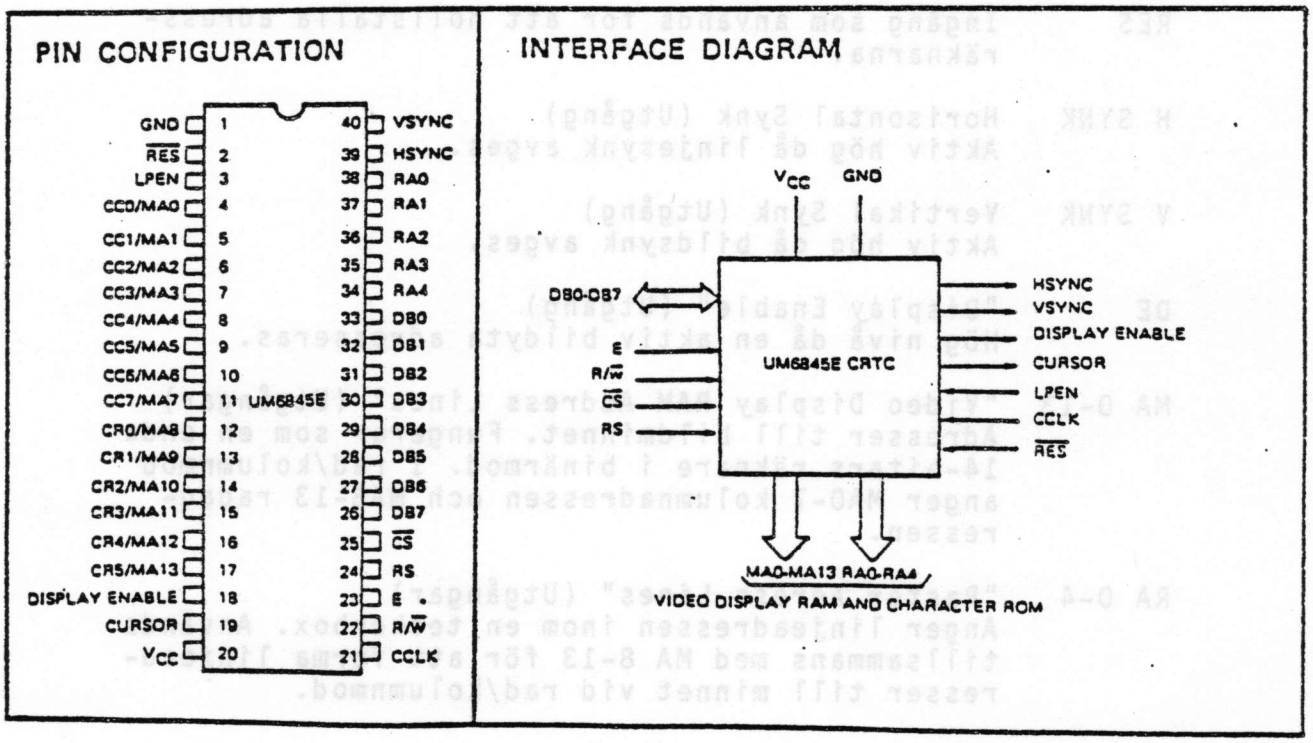
R10/11 Markör start/slut.
Två 5-bitarsregister som beskriver vilken linje som markören startar på (R10) respektive slutar (R11) inom ramen för teckenboxen.

R12/13 Bildstartadress.
Dessa två register bildar tillsammans den minnesposition där avsökningen ska börja. R12 fungerar som radregister och R13 som kolumnregister i "rad/kolumnmod", annars ett binärt register.

R14/15 Markörposition.
Dessa två register bildar ett 14-bitarsregister som talar om markörpositionen.

R16/17 Ljuspenna.
Dessa två 14-bitarsregister visar ljuspennans position.

R18/19 Aktuell position.
Dessa två register bildar ett 14-bitarsregister som pekar ut nästa position som ska läsas ur minnet.



In/Utgångar:

E "Enable" (Ingång)
Används för att synkronisera läsning/skrivning mellan CPU och 6845E.

R/$\overline{W}$ "Read/Write" (Ingång)
Läs/skrivstrob från CPU:n som anger datöverföringens riktning.

$\overline{CS}$ "Chip Select" (Ingång)
Aktiveras av CPU:n då den vill skriva eller läsa i något register.

RS "Register Select" (Ingång)
Vid låg nivå kan CPU:n skriva i adressregistret för att ange det register som ska uppdateras. Vid läsning kommer statusregistrets innehåll att finnas tillgängligt på databussen. Då RS har hög nivå skrivs data till det tidigare valda registret.

DB 0-7 "Data Bus" (Dubbelriktad)
8-bitars dubbelriktad databuss för att transportera data till och från registren.

CURSOR (Utgång) Aktiv hög då minnesadress överens - stämmer med marköradressen i R14/15.

LPEN Ingång från ljuspenna.

CCLK "Character Clock" (Ingång)
Teckenklocka för att intern "timing" och minnesadressering.

$\overline{\text{RES}}$ Ingång som används för att nollställa adressräknarna.

H SYNK Horisontal Synk (Utgång)
Aktiv hög då linjesynk avges.

V SYNK Vertikal Synk (Utgång)
Aktiv hög då bildsynk avges.

DE "Display Enable" (Utgång)
Hög nivå då en aktiv bildyta adresseras.

MA 0-13 "Video Display RAM Address Lines" (Utgångar)
Adresser till bildminnet. Fungerar som en enda 14-bitars räknare i binärmod. I rad/kolumnmod anger MA0-7 kolumnadressen och MA8-13 radadressen.

RA 0-4 "Raster Adress Lines" (Utgångar)
Anger linjeadressen inom en teckenbox. Används tillsammans med MA 8-13 för att forma linjeadresser till minnet vid rad/kolumnmod.

6 Skivminnet i ABC 1600

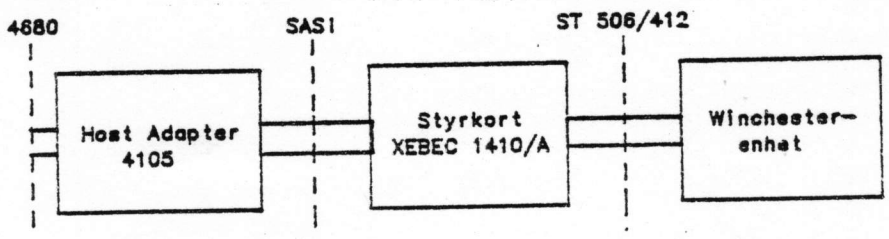
För att få tillgång till en stor mängd data finns ett skivminne av winchstertyp inbyggt i ABC 1600. Skivminnet är av fast typ dvs. man kan inte byta ut dess skivor. Data och program måste därför kopieras till en flexskiva eller magnetband då man önskar en säkerhetskopia eller i omvänd riktning tillföra systemet nya programvaror eller data.

Winchesterminnen har avsevärt högre tillförlitlighet än exv. flexskiveminnen, de är också snabbare och har högre lagringskapacitet. I UNIX-baserade datorsystem som ABC 1600 är det en nödvänighet att ha tillgång till ett snabbt skivminne med god lagringskapacitet för att samtidigt kunna använda flera programvaror utan att därför byta media.

Skivminnnet innehåller vid leverans de filer som behövs för att initiera systemet samt operativsystemet ABCenix med sina "utilities". Efter detta lägger man på de användarorienterade programvarorna som gör systemet till CAD-system, ordbehandlare eller liknande som gör ABC 1600 till ett användbart system.

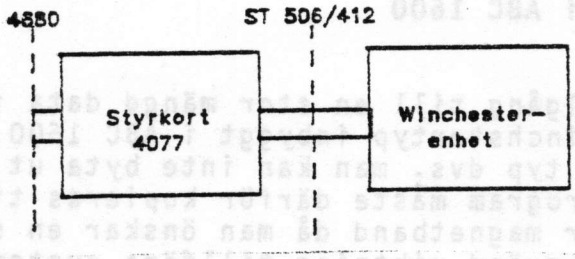
Winchesterminnet ansluts till ABC 1600 via expansionsplatsen BUS2. Då expansionsplatserna är avsedda att hantera många olika typer av interface och styrkort passar de inte direkt till winchsterenheten, där själva lagrinsmediat sitter, utan deras signaler måste först omvandlas för att kommunikationen med winchesterenheten ska fungera. Detta sker i ett s.k. interface vars främsta uppgift är att signalmässigt koppla samman de två portarna samt att mellanlagra data som transporteras till eller från skivminnet.

I ABC 1600 kan två olika lösningar på denna anpassning förekomma:



Alternativ 1

Alternativ 1 innebär att anpassningen består av två moduler, en s.k. "SASI-interface" som utför en signalomvandling mellan expansionsbuss BUS2 och själva styrkortet som hanterar mellanlagring av data samt signalomvandlingen till själva winchesterenheten.



Alternativ 2

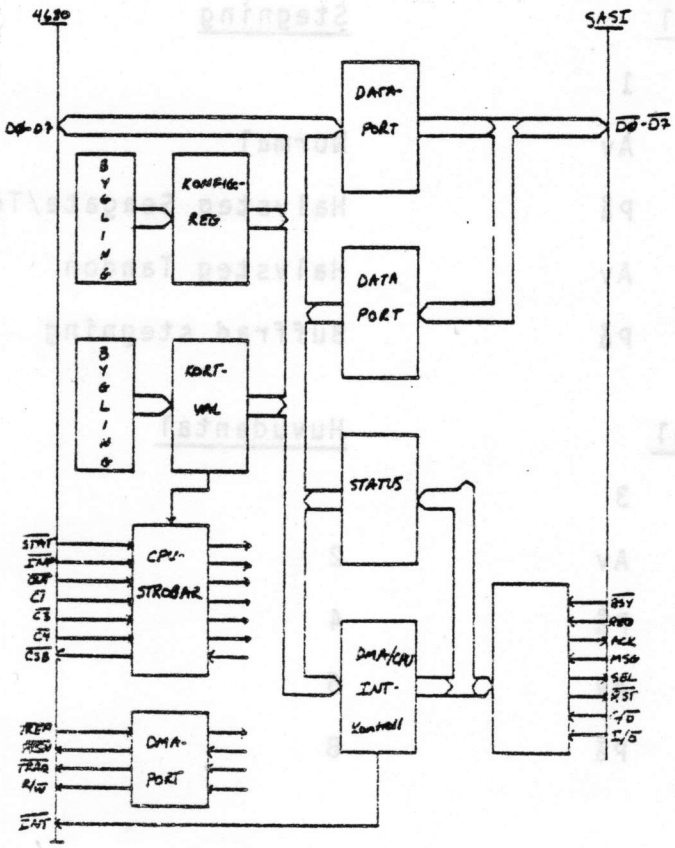
I alternativ 2 har dessa båda moduler ersatts av en enda som hanterar båda kortens uppgifter.

Båda alternativen och deras ingående enheter ska nu beskrivas med början på alternativ 1.

6.1 SASI-INTERFACE 4105

SASI-interfacet som placeras i själva expansionsutrymmet för BUS2, omvandlar datorsystemets I/O-signaler till SASI-snitt som idag används av många typer av styrkort. Den kan genom sina handskakningssignaler användas vid både CPU- och DMA-överföring. Dess huvudblock är följande:

Dataportar:	För transport av data och kommando till, samt data och status från styrkortet.
Konfigureringsregister:	Talar om skivminnets egenskaper och liknande parametrar.
Statusregister:	Talar om vissa handskakningssignalers läge.
DMA/CPU-register:	Sköter omkopplingen mellan DMA/CPU.
Kortvalsregister:	Anger kortadress.
Handskakning SASI:	Signalering till/från styrkortet.
Handskakning 4680:	Strobar och signaler för datorkommunikationen.



I likhet med alla andra I/O-kort av 4680-typ måste SASI-interface 4105 adresseras med hjälp av kortvals - stroben CS samtidigt som kortadressen läggs ut på databussen. I datakomparatorn 2E jämförs databussens kortadress med det på byglingsfältet 5E inställda värdet. Är adresserna lika aktiveras komparatorns utgång och denna öppnar porten 1B och en del av porten 1A så att CPU:ns I/O-strobar släpps fram samtidigt som kortet signalerar CSB tillbaka till bussinterfacet.

Med hjälp av I/O-strobarna kan nu CPU:n styra och läsa SASIinterfacets register. Genom att aktivera stroben STAT (läsning) kan statusregistret 2D läsas, INP ger konfigurationsregistrets data utom då styrkortets BSY-signal är aktiv (låg), då CPU:n med denna strob läser data från styrkortet.

OBS!
Vid senare versioner av operativsystem ligger konfigurationsdata på skivminnet, vilket gör att detta register saknar betydelse, men för ordningens skull redovisar vi byglarnas funktion.

Konfigurationsregister data

Bygel nr	Funktion
1-2	Stegningsoptioner
3-4	Antal huvuden
5-8	Skivminnestyp

Bygel 2	Bygel 1	Stegning
Av	Av	Normal
Av	På	Halvsteg Seagate/Texas
På	Av	Halvsteg Tandon
På	På	Buffrad stegning

Bygel 4	Bygel 3	Huvudantal
Av	Av	2
Av	På	4
På	Av	6
På	På	8

Bygel				Skivminnestyp
8	7	6	5	
AV	Av	Av	Av	Seagate ST506 (5 Mbyte)
Av	Av	Av	På	RODIME RO 100
Av	Av	På	Av	SHUGART SA 600
Av	Av	På	På	Seagate ST 412 (10 Mbyte)

När CPU:n aktiverar någon av strobarna C1,C3,C4 skrivs data till SASI-interfacet. Med hjälp av C1 aktiveras SEL-signalen till styrkortet via D-vippan 2A och NAND-grinden 5A. SEL-signalen används för att välja rätt styrkort då flera är anlslutna till samma SASI-interface genom att en adress samtidigt avges på data-bussen.

För att adressen ska nå styrkortet måste dataregistret 3D och dataporten 5D aktiveras. 3D aktiveras av OUT-stroben som alltid avges då någon av strobarna C1-C4 ges, under det att 5D styrs av I/O-signalen från styr-kortet. I/O-signalen indikerar om styrkortet förväntar sig en skrivning eller läsning och står i initialskedet alltid hög dvs. data väntas från SASI-interfacet.

Stroben C3 grindas med resetsignalen RST och används för att nollställa dataregistret 3D, DMA-registret 3E samt sänder den resetsignal till styrkortet. Enda skillnaden mellan RST och C3 är att RST även deaktiv-erar kortvalet.

C4 klockar DMA-registret 3E vars utgångar styr om över-föringen ska ske med hjälp av DMA eller om CPU:n han-terar den själv. Registrets innehåll talar också om hur interruptsignaleringen ska gå till. Indata till regist-ret kommer från databussens bitar D5-D7 där D5 och D7 styr interruptsignalen och D6, DMA/CPU-mod. Sätts utgång $\overline{Q2}$ låg öppnas porten 1A så att DMA:ns handskak-ningssignaler släpps fram.

Med hjälp av bygel S1 kan man välja om TREN-signalen ska fördröjas eller ej för att erhålla rätt "timing". Då SASI-interfacet används i ABC 1600 ska denna alltid vara byglad utan fördröjning dvs i läge "b".

I detta skede har redan styrkortet och aktuell DMA programmerats för respektive uppgifter av CPU:n och signaleringen mot CPU och DMA fortsätter som tidigare beskrivits under avsnittet DMA i tekniska beskrivning-en.

Registret styr också SASI-interfacets grindlogik så att CPU:ns eller DMA:ns signaler kan påverka styrkortet. Grindlogiken ser också till att signalanpassningen mot styrkortet blir den riktiga. SASI-snittet består förutom databuss av åtta kontroll och handskakningssignaler som SASI-interfacet måste hantera för att kommunikationen ska fungera. Vi har tidigare omtalat SEL-signalen, med vilken rätt styrkort adresseras samt RSTsignalen som används för nollställning av styrkortets register.

I/O-signalen från styrkortet indikerar riktningen av dataflödet som kortet för tillfället är programmerat att hantera och används på SASI-interfacet för att ställa om dataportarnas riktning samt att generara R/$\overline{W}$-signal till bussinterfacet i DMA-mod.

$\overline{C}$/D-signalen från styrkortet indikerar om informationen på databussen är kommando eller data. Låg nivå anger vid skrivning till styrkortet att databussen innehåller ett kommando till styrkortet under det att det vid läsning indikerar att databussen innehåller statusinformation från styrkortet. Då $\overline{C}$/D är hög innebär detta att data överförs på bussen. $\overline{C}$/D-signalen känns också av i SASI-interfacet för att indikera olika lägen tillsammans med andra SASI-signaler. I kommandoläge ger den tillsammans med I/O-signalen i läsläge, interrupt till CPU:n för att idikera att styrkortet sänder "errorstatus". I datamod genererar den interrupt då data överförs i CPU-mod eller $\overline{TRRQ}$ tillsammans med REQ i DMA - mod.

BUSY-signalen går låg som svar på en SEL-signal med rätt adress från SASI-interfacet. Signalen indikerar till SASI-interfacet att styrkortet är klart för överföring.

BUSY aktiverar också dataportarna 1D eller 3C tillsammans med CPU/DMA-strobarna beroende på läget samt återställer selektsignalen vid D-vippen 2A.

MSG-signalen från styrkortet indikerar att aktuellt kommando har utförts och sänder samtidigt meddelandedata till SASI-interfacet.

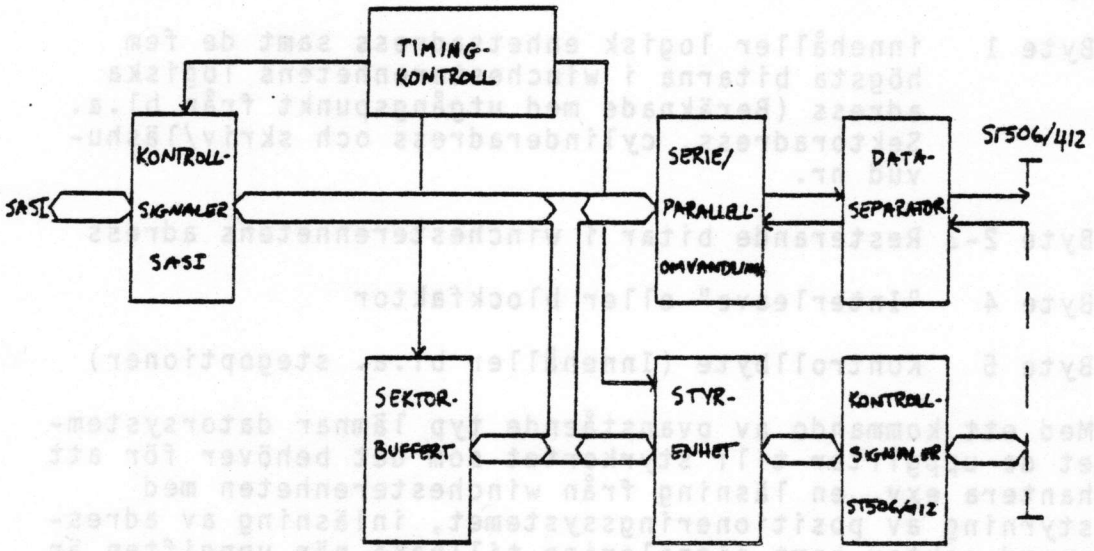
REQ-signalen sätts låg av styrkortet för alla data som överförs mellan SASI-interfacet och styrkortet. Signalen ger också indirekt upphov till ACK-signalen genom att öppna D-vippen 2A så att den kan klockas av CPU:ns eller DMA:ns strobsignaler. ACK-signalen genereras av SASI-interfacet som svar på REQ-signalen och indikerar att interfacet är klart att sända eller ta emot data.

6.2 Styrkort

Mellan SASI-interfacet 4105 och winchsterenheten sitter styrkortet. Detta kan var av olika typ varför vi här bara beskriver dess funktion helt allmänt då en mer noggrann studie bara gäller för en enda typ. De styrkort som används i ABC 1600 är normalt av typ XEBEX 1410 eller XEBEX 1410A men även andra kan förekomma.

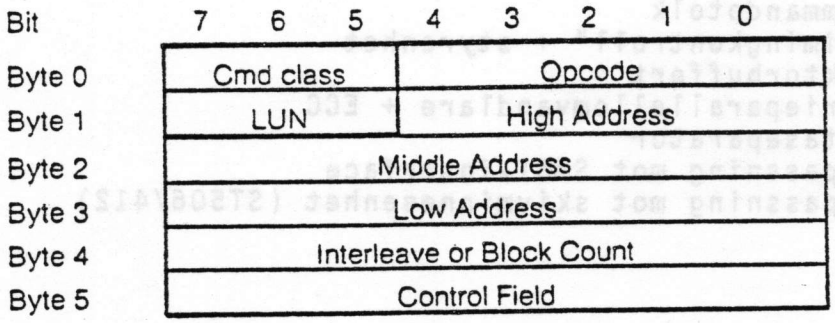
Styrkortets huvudblock är:

- Kommandotolk
- "Timingkontroll" + styrenhet
- Sektorbuffert
- Serieparallellomvandlare + ECC
- Dataseparator
- Anpassning mot SASI-interface
- Anpassning mot skivminnesenhet (ST506/412)



Styrkortets uppgift är att handha winchesterenheten genom att omvandla de styrkommandon som kommer från datorsystemet till signaler som kopplas till winchesterenhetens standardinterface ST506/412.

Det ska också kunna lagra datainnehållet i en hel sektor (256/512 bytes) samt att med hjälp av en kontrollkod ECC som skrivs in samtidigt med data, avhjälpa fel som kan uppträda vid läsning från winchesterenhetens media, s.k. "Error Correction". Från datasystemet får styrenheten via SASI-interfacet ett kommando bestående av sex bytes.



Kommandosekvens

Byte 0 innehåller kommandoklass samt operationskod

Byte 1 innehåller logisk enhetsadress samt de fem högsta bitarna i winchesterenhetens logiska adress (Beräknade med utgångspunkt från bl.a. Sektoradress, cylinderadress och skriv/läshuvud nr.

Byte 2-3 Resterande bitar i winchesterenhetens adress

Byte 4 "Interleave" eller blockfaktor

Byte 5 Kontrollbyte (Innehåller bl.a. stegoptioner)

Med ett kommando av ovanstående typ lämnar datorsystemet de uppgifter till styrkortet som det behöver för att hantera exv. en läsning från winchesterenheten med styrning av positioneringssystemet, inläsning av adresserad sektor samt signalering tillbaka när uppgiften är utförd.

I sektorbufferten mellanlagras data som ska överföras till winchesterenheten eller som just lästs från den för att undvika att någon av enheterna skivminne eller dator inte hinner med. Serie/parallellomvandlaren är reversibel och omvandlar dels parallelldata från sektorbufferten till seriedata till winchesterenheten via dataseparatorn eller omvänt. Här adderas också den s k felkorrektionskoden (ECC) till data som skrivs till winchesterenheten samt kontrollerar densamma vid läsning från enheten. Med hjälp av denna kod kan läsfel från mediat rättas och återställas i sitt ursprungliga skick.

Datorseparatorn får data vid skrivning från serie/parallellomvandlaren i serieform enligt NRZ-kod som konverteras till MFM-kod för att få en s.k. självklockande kod. När data sedan läses från winchesterenheten separeras data och klocka så att en seriell dataström enligt NRZ-format erhålls samt en dataklocka vars ursprung är det MFM-kodade data som levereras från winchesterenheten.

Den enhet som styr intern och extern tidskontroll samt kontrollerar dataflödet kallas "timingkontroll". Den hanterar styrsignaler och handskakning med SASI-interfacet, sektorbuffert och serie/parallellomvandlare. För kommunikation med datorsystem och skivminne finns de två interfacen enligt SASI respektive ST506/412. Av dessa har SASI-interfacet redan behandlats.

Skivminnesinterfacet ST506/412 har blivit någon form av industristandard och idag använder de flesta skivminnesleverantörer detta snitt. Snittet har stora likheter med det som sedan länge varit standard för flexskiveenheter men arbetar med snabbare seriedata (5 Mbit/s =Winchester resp. 250Kbit/s för miniflexskiveenheter), samt att varje enhet har sitt eget kablage för dataöverföring. Här finns också fler signaler för att välja skriv/läshuvud. Vilka signaler som förekommer är också beroende på styrkortets och winchesterenhetens egenskaper och möjligheter.

Signal Pin	Ground Return	Signal Name
2	1	Reduced Write Current
4	3	Head Select 2^2
6	5	Write Select Gate
8	7	Seek Complete
10	9	Track 00
12	11	Write Fault
14	13	Head Select 2^0
16	15	Reserved
18	17	Head Select 2^1
20	19	Index
22	21	Ready
24	23	Step
26	25	Drive Select 1
28	27	Drive Select 2
30	29	Reserved
32	31	Reserved
34	33	Direction In

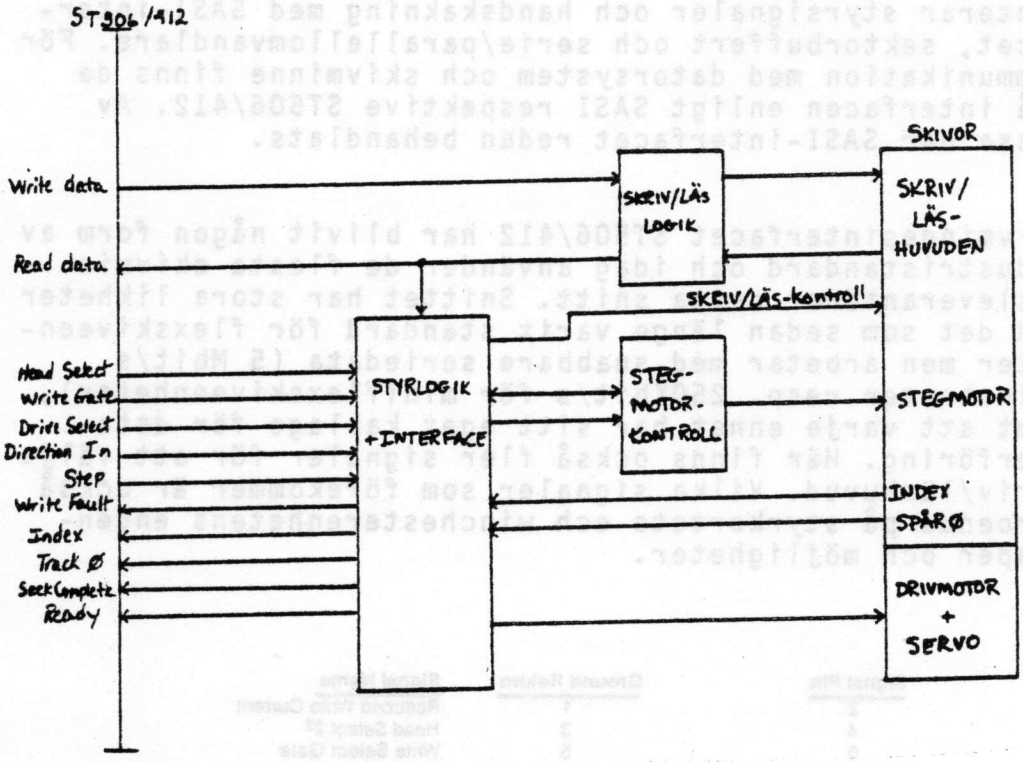
CONNECTORS J2 and J3, DATA SIGNALS, PIN ASSIGNMENTS

Signal Pin	Ground Return	Signal Name
1	2	Drive Selected
5	6	Spare
7	8	Reserved
—	—	Spares, pins 9 and 10
11	12	Ground (GND)
13		MFM Write Data+
14		MFM Write Data-
15	16	Ground (GND)
17		MFM Read Data+
18		MFM Read Data-
19	20	Ground (GND)

Signalsnitt Winchesterenhet

6.3 Winchesterenhet

När det gäller Winchesterenhet kan ABC 1600 även här vara bestyckade med olika typer/fabrikat. Exempel på skivminnestyper är BASF6188 (13Mbyte), BASF 6188R (20 Mbyte) och NEC D5126 (20Mbyte). Här liksom i fallet med styrkortet skiljer sig konstruktionerna avsevärt varför vi endast beskriver grundprinciperna för winchesterenheten.



Blockschema Winchesterenhet

Winchesterenhetens huvuddelar är följande:

- Skivor med magnetmaterial
- Skriv/läshuvuden + förstärkare
- Drivmotor + servo
- Styrmotor eller "voice coil"
- Styrlogik + interface med styrkort

Antalet skivor i en winchesterenhet kan variera från 1 till 8 st beroende på skivminnestyp och normalt finns här endast en eller två skivor. De är hårda (ej böjliga som flexskivor) och sitter inneslutna i ett dammfritt rum. Detta utrymme får under inga omständigheter öppnas då detta omedelbart förstör skivor och huvuden. Måste de av någon anledning öppnas ska detta ske i ett s.k. dammfritt rum som normalt bara finns på tillverkningsstället samt någon enstaka specialiserad serviceinrättning.

Skivorna är belagda med ett tunt skikt av finkorningt magnetiskt material där data skrivs och läses av huvudet som svävar fritt över ytan. Skivans rotationshastighet gör att en luftkudde bildas mellan skivan och huvudet så att det ej ligger an mot skivan som vid flexskivor. Detta gör att slitaget ska bli minimalt och tillförlitligheten ökar.

Skriv/läshuvudena sitter också inne i den dammfria kapseln och är dubbelt så många som antalet skivor då man utnyttjar skivans båda sidor. Till skriv/läshuvudet skickas data i seriell form från styrkortet som via en spole och en smal spalt i huvudet ger flödesändringar i skivans magnetiska skikt.

Vid läsning ger de små flödesändringarna upphov till strömändring i huvudets spole som därefter detekteras, förstärkes och signalomvandlas så att dessa kan registreras som seriella data av styrkortet.

Drivmotorn, som normalt driver skivorna direkt utan någon mellankoppling i form av remmar eller band, ger skivorna en hastighet av c:a 3600 varv/minut. Med hjälp av ett servo hålls denna hastighet konstant så att hastighetsvariationer inte kan äventyra lässäkerheten. Då spänningen slås till tar det mellan 10-15 sekunder innan motorn ställt in sig på rätt hastighet.

För att ge skriv/läshuvudena rätt position i förhållande till det inskrivna spåret på skivorna finns någon form av positioneringssystem. Vid denna typ av winchesterminnen är stegmotorer vanligast men även andra typer förekommer. Stegmotorn styrs av signalerna "STEP" och "DIRECTION" från styrkortet, där "STEP" ger en puls för varje spår som skriv/läshuvudet ska förflytta sig och "DIRECTION" anger förflyttnigens riktning.

De flesta skivminnen kan idag hanteras med s.k. "buffered step", vilket innebär att stegpulserna i snabb följd påförs winchesterenheten, mycket snabbare än stegmotorn kan arbeta. Där räknas de under tiden in i ett register i winchesterenheten så att styrkortet kan ägna sig åt andra uppgifter än att vänta på en långsam stegningsfrekvens. Stegmotorn får också en chans att stega så snabbt som möjligt utan hänsyn till en stegtid som är tilltagen med god marginal. Detta gör att man snabbare kommer åt data på skivminnet.

Stegningskontrollen som ofta är processorstyrd kan normalt också finjustera huvudets läge så att maximal signalstryka erhålles. Då rätt position uppnåtts, signalerar winchesterenheten genom att aktivera signalen "SEEK COMPLETE" till styrkortet.

Av de signaler som vi inte tidigare studerat märks de fyra "DRIVE SELECT"-ledningarna med vars hjälp man kan ansluta upp till fyra skivminnen på samma kontrollledningar. På winchesterenheten finns ett byglingsfält där dess adress anges och som motsvarar någon av de fyra ledningarna.

"HEAD SELECT" avkodas binärt på winchesterenheten och de tre ledningarna ger möjlighet till åtta huvuden. Även signalen "Reduced Write Current" kan ibland användas som selektledning och antalet adressbara huvuden ökar då till 16. Denna signal är normalt till för att reducera skrivströmmen på de inre spåren men detta hanterar numera de flesta skivminnen själva varför signalen blivit mer eller mindre överflödig.

"INDEX" och "TRACK0" kommer från sensorer som hör till skivorna. "INDEX" anger att ett nytt varv just ska inledas under det att "TRACK0" indikerar att skriv/läshuvudena står över spår 0.

"WRITE FAULT" indikerar att något fel uppstått vid skrivning till skivan och "WRITE GATE" talar om att data kommer att skrivas till skivan.

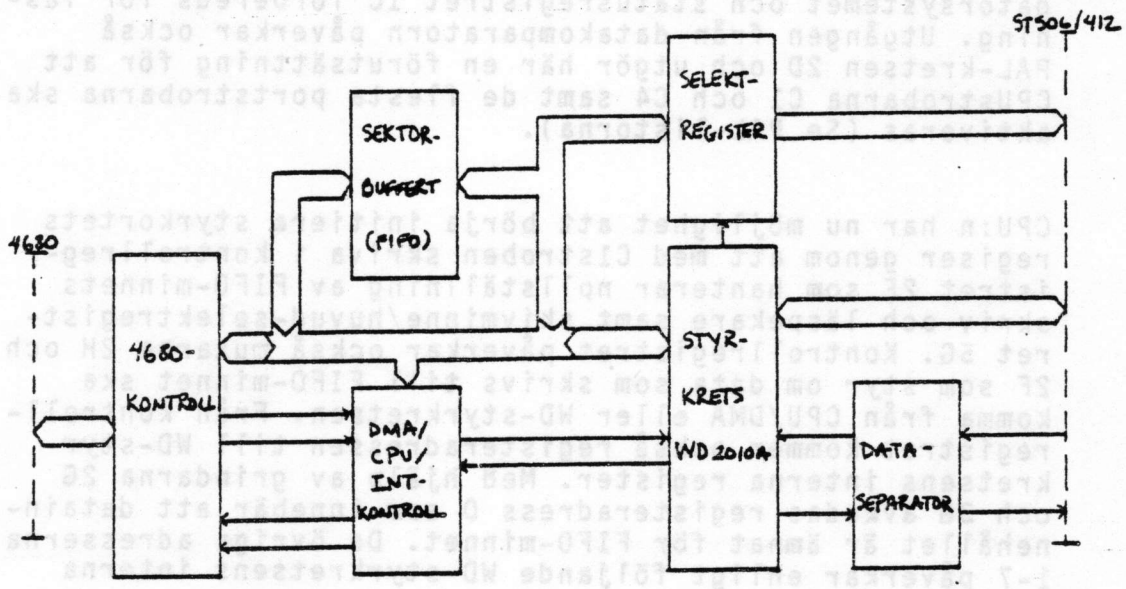
"READY" slutligen indikerar att winchesterenheten är klar att ta emot styrsignaler.

6.4 Winchesterstyrkort 4077

Winchesterenheten i ABC 1600 kan som tidigare visats styras enligt ett annat alternativ. Här ersätts SASI-interfacet 4105 och styrkortet med en enda modul, styrkortet 4077. Den kombinerar de båda funktionerna och innebär att man inte går via SASI-snittet utan omvandlar signalerna direkt från 4680 till ST506/412. Kortet som är av europaformat (100*160mm) placeras i expansionsutrymmet BUS2 d.v.s. den position som SASI-interfacet hade tidigare. Det lilla formatet har åstadkommits med hjälp av den intelligenta styrkretsen samt att grindlogiken samlats i PAL-kretsar.

4077-kortets huvuddelar är följande:

- Styrkrets WD 2010A
- Dataseparator WD 10C 20A
- Sektorbuffert
- Anpassning mot datorsystem (4680)
- Anpassning mot skivminne (ST 506/412)



Blockschema

Styrkortets huvuduppgift är att mellanlagra de data som transporteras mellan datorsystemet och winchesterenheten samt att med ledning av de kommandon som kommer från systemet genera lämpliga strobar för att hantera winchesterenheten.

Den centrala delen av styrkortet utgöres av styrkretsen WD 2010A som tolkar de kommandon som kommer från CPU:n, hanterar parallell/serieomvandling av data samt de flesta strobarna mot winchesterenheten.

Till sin hjälp har den en dataseparator som påverkar seriedata till/från winchesterenheten så att omvandlig till den självklockande koden MFM samt skrivkompensering utförs vid skrivning till winchesterenheten och separering av data och klocka vid läsning från densamma. Styrkretsen har också tillgång till sektorbufferten som i detta fall utgöres av ett s.k. FIFO - minne, med vars hjälp data kan mellanlagras vid överföring.

Till sektorbufferten hör också portkretsar som kan styra flödet på databussen samt en kontrollenhet som styr portarnas riktning. Här finns också kontrol - logik för CPU/DMA-omkoppling samt interrupthantering och register för status, skivminne/huvud-selekt och registerkontroll.

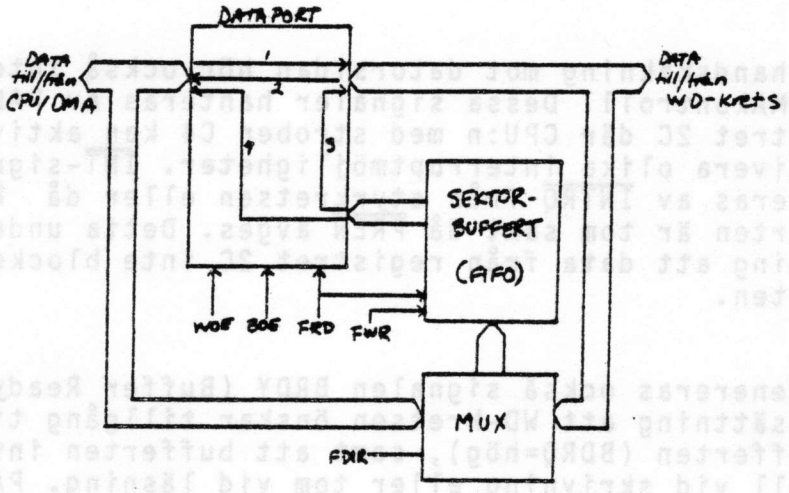
Vid dataöverföring till/från winchesterenheten aktiveras styrkortet via kortvalsstroben $\overline{CS}$ från CPU:n samtidigt som kortadressen läggs ut på databussen.

Via portkretsen 1E jämförs adressen i datakomparator 3H med den på byglingsfältet SW1 inställda och då dessa stämmer överens aktiveras CSB-stroben tillbaka till datorsystemet och statusregistret 1C förbereds för läsning. Utgången från datakomparatorn påverkar också PAL-kretsen 2D och utgör här en förutsättning för att CPUstrobarna C1 och C4 samt de flesta portstrobarna ska aktiveras (Se PAL-listorna).

CPU:n har nu möjlighet att börja initiera styrkortets regiser genom att med C1stroben skriva i kontrollregistret 2E som hanterar nollställning av FIFO-minnets skriv-och läspekare samt skivminne/huvud-selektregistret 5G. Kontrollregistret påverkar också muxarna 2H och 2F som styr om data som skrivs till FIFO-minnet ska komma från CPU/DMA eller WD-styrkretsen. Från kontrollregistret kommer också registeradressen till WD-styrkretsens interna register. Med hjälp av grindarna 2G och 3G avkodas registeradress 0 som innebär att datainnehållet är ämnat för FIFO-minnet. De övriga adresserna 1-7 påverkar enligt följande WD-styrkretsens interna register:

Adress				
A2	A1	A0	Läsning	Skrivning
0	0	0	Ej aktiv	
0	0	1	Errorregister	Cylinder för start av skrivkompensering
0	1	0	Antal sektorer	
0	1	1	Sektornummer	
1	0	0	Lägsta cylindernummer	
1	0	1	Högsta cylindernummer	
1	1	0	Skivminne/huvud-register	
1	1	1	Statusregister	Kommandoregister

Dataflödet till/från FIFO och WD-styrkretsen kopplas via de två PAL-kretsarna 1F och 1H. Med hjälp av de fyra styrstrobarna BOE, $\overline{\text{BOE}}$ (Bus Output Enable) WOE (Winchester Output Enable) och $\overline{\text{FRD}}$ (FIFO Read) kan data kopplas i önskad riktning.



Dataflödet FIFO+DATAPORT

Då CPU eller DMA skriver till FIFO-minnet sker datta genom muxarna 2F och 2H genom att $\overline{\text{FDIR}}$ har låg nivå samt att stroben $\overline{\text{FRD}}$ aktiveras.

När CPU:n skriver data till WD-styrkretsen direkt (kommando eller registerdata) aktiveras $\overline{\text{WOE}}$, vilket gör denna port till en utgång för data, samt WD-kretsens skristrob $\overline{\text{WWR}}$. Data från WD-kretsens status- och error-register läses till CPU:n genom att $\overline{\text{BOE}}$ satts till låg nivå vilket gör dataporten mot CPU:n till en aktiv utgång samtidigt som skrivstroben $\overline{\text{WRD}}$ aktiveras.

Båda dessa transaktioner förutsätter att stroben $\overline{BCS}$ från WD-styrkretsen ligger hög, vilket innebär att styrkretsen inte själv driver strobarna.

Då $\overline{BOE}$ och $\overline{FRD}$ är aktiva kan CPU/DMA läsa data från FIFO-minnet och då kopplas FIFO-minnets utgångsdata via PAL-kretsarna till CPU:ns databuss.

WD-styrkretsen måste också ha tillgång till FIFO-minnet ch förutsättningen för detta är att stroben BCS är aktiv. Detta innebär att WD-kretsen själv driver skriv/lässtrobarna och kan nu via PAL 2D även styra FIFO-minnets strobar.

Då WD-kretsen läser från FIFO aktiveras $\overline{FRD}$ och $\overline{WOE}$, vilket gör att data finns tillgängligt för styrketsen och vid skrivning till FIFO sker detta genom att $\overline{FWR}$ aktiveras samtidigt som data levereras via muxarna. I detta läge är FDIR-signalen hög. Genom PAL-listorna får du mer upplysning om de funktioner som styr PAL-kretsarna.

Via dataporten kan CPU:n också skriva i registret SDH (Select Drive Head) som ligger utanför själva WD-kretsen. WD-kretsen lagrar upp de registerdata som sänds men kan p.g.a. brist på anslutningsben inte själva styra funktionerna. Det blir således registret 5G som får styra skivminneselekt och huvudselekt men laddning av registret sker via stroben SDHLE (Select Drive Head Latch Enable) som avges från styrkretsen.

Till handskakning mot datorsidan hör också interrupt och DMAkontroll. Dessa signaler hanteras av PAL 1B och registret 2C där CPU:n med stroben C4 kan aktivera/deaktivera olika interruptmöjligheter. $\overline{INT}$-signalen kan aktiveras av $\overline{INTRQ}$ från styrkretsen eller då FIFO-bufferten är tom samt då $\overline{PREN}$ avges. Detta under förutsättning att data från registret 2C inte blockerar möjligheten.

Här genereras också signalen BRDY (Buffer Ready) under förutsättning att WD-kretsen önskar tillgång till sektorbufferten (BDRQ=hög), samt att bufferten inte redan är full vid skrivning eller tom vid läsning. PAL 1B levererar också $\overline{TRRQ}$-signal till bussinterfacet samt skriv/lässtrob i DMA-mod.

Interfacet mot winchesterenheten hanteras av WD-styrkretsen med utgångspunkt från innehållet i de interna registren och kontrollsignalerna INDX (Indexpulse), WFLT (Write Fault), RDY (Ready), TOO (Track OO), SKC (Seek Complete) samt utsignalerna STEP och DIR (Direction). Selektsignalerna hanteras som vi tidigare sett av SDH-registret 5G.

När det gäller hanteringen av seriedata till eller från skivminnet, bearbetas dessa av dataseparatorn U 101. Dess uppgift är att omvandla dataströmmen till winchesterenheten så att en självklockande kod (MFM) bildas. Som utgångspunkt för detta används seriedata enligt NRZ-kod som kommer från styrkretsen på WD-ledningen samt skrivklockan WC. För att också ge data till winchesterenheten rätt skrivkompensering avkänns signalerna EY (Early) och LE (Late) så att rätt fördröjning av data kan uppnås.

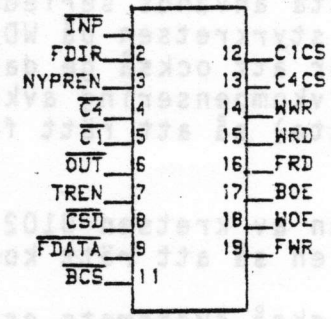
Fördröjningstiden får man av kretsen U102 som i fasta steg kan fördröja signalen så att rätt kompensering erhålls.
I dataseparatorn finns också systemets oscillator som via en 10 MHz kristall efter delning levererar 5 MHz-klocka till styrkretsen Här finns också en PLL med en spänningsstyrd oscillator vars styrande element bl.a. utgörs av de externa kapacitansdioderna CR01 och CR02. Med hjälp av denna PLL återskapas klockan från det data som läses från skivminnet. Efter dataseparering levereras data och klocka till styrkretsen via signalerna RC (Read Clock) och RD (Read Data).

Till 4077-kortet kan två winchesterenheter anslutas. Kontrollsignalerna till/från dessa går via en gemensam 34-polig kabel under det att data överförs på separata 20-poliga kablage. Data överförs balanserat, vilket innebär att två ledningar åtgår för skrivdata resp. läsdata. Detta innebär också att signalerna måste anpassas till detta då de lämnas eller tas emot från winchesterenheten. Dessa anpassningar sker i ingångskretsen 5A och utgångskretsen 5C.

6.4.1 PAL 2D

Uppgift: Styrning av FIFO-minne

Typ: PAL 16L8A-2



PAL 2D

Insignaler

$\overline{INP}$	INP-strob från 4680
FDIR	Anger inenhet WD/CPU till FIFO
MYPREN	Skapar förutsättningar för PREN från register 2C
$\overline{C4}$	Utstrob C4 från 4680
$\overline{C1}$	Utstrob C1 från 4680
$\overline{OUT}$	OUT-strob från 4680
TREN	"Transfer Enable" från DMA-kontroll
$\overline{CSD}$	Samma strob som CSB, anger att kortet är uppvalt
$\overline{FDATA}$	Från NAND-grind 2G, anger att data på bussen är avsedd för FIFO
$\overline{BCS}$	Från WD-krets, avsedd att ge åtkomst till sektorbuffert

Utgångar

$\overline{WOE} = \overline{BCS} * \overline{FDIR} * \overline{WRD} * \overline{FRD} * + \quad (1)$
$BCS * FDATA * \overline{WWR} \quad (2)$

Utgången WOE aktiverar utgångarna på PAL-kretsarna 1F och 1H så att dataporten mot WD-kretsen blir aktiverad som utgång. Detta sker i två lägen, dels då WD-kretsen läser data från FIFO (1), dels då CPU:n skriver i WD-kretsens register (2).

$\overline{BOE} = \overline{FDATA} * FDIR * \overline{FRD} * \overline{INP} * \overline{CSD} + \quad (1)$

$MYPREN * FDIR * \overline{FRD} * \overline{TREN} + \quad (2)$

FDATA * BCS * $\overline{WRD}$ * $\overline{INP}$ * $\overline{CSD}$ (3)

Utgången BOE aktiverar utgångarna på PAL-kretsarna 1F och 1H så att dataporten mot CPU/DMA blir aktiverad som utgång. Detta händer när CPU:n läser från FIFO (1), DMA:n läser från FIFO (2) samt då CPU:n läser status eller errorkod från WD-kretsen.

$\overline{WWR}$ = FDATA * BCS * $\overline{UTP}$ * $\overline{CSD}$

Skrivstrob till WD-kresten då CPU:n skriver i dess register. Denna signal kan också drivas av WD-kretsen och då är denna utgång högimpediv.

$\overline{WRD}$ = FDATA * BCS * $\overline{INP}$ * $\overline{CSD}$

CPU:n skriver till WD-krets. I övrigt se $\overline{WWR}$.

$\overline{FWR}$ = $\overline{FDATA}$ * $\overline{FDIR}$ * $\overline{UTP}$ * $\overline{CSD}$ + (1)

MYPREN * $\overline{FDIR}$ * $\overline{TREN}$ + (2)

$\overline{BCS}$ * FDIR * $\overline{WWR}$ (3)

Skrivstrob till FIFO-minne. Aktiv då CPU (1), DMA (2) eller WD (3) skriver data till FIFO.

$\overline{FRD}$ = $\overline{FDATA}$ * FDIR * $\overline{INP}$ * $\overline{CSD}$ + (1)

MYPREN * FDIR * $\overline{TREN}$ + (2)

$\overline{BCS}$ * $\overline{FDIR}$ * $\overline{WRD}$ + (3)

Lässtrob till FIFO-minne. Aktiv då CPU (1), DMA (2) eller WD (3) läser data från FIFO.

$\overline{C1CS}$ = $\overline{CSD}$ * $\overline{C1}$

Samma som $\overline{C1}$-strob från 4680 då kortet är uppvalt.

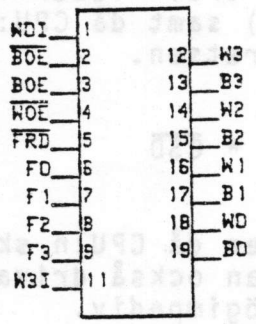
$\overline{C4CS}$ = $\overline{CSD}$ * $\overline{C4}$

Samma som C4-strob från 4680 då kortet är uppvalt.

6.4.2 PAL 1F och 1H

Uppgift: Dataport till och från 4680-buss samt från FIFO till WD-styrkrets.

Typ: PAL 16L8A-2



PAL 1F OCH 1H

Insignaler

WO1 = WO

$\overline{BOE}$ = Aktiverar datautgången till 4680

BOE = Aktiverar datautgången till 4680

$\overline{WOE}$ = Aktiverar datautgången till WD

$\overline{FRD}$ = Lässtrob till FIFO

F0 = Databit 0 eller 4 från FIFO

F1 = Databit 1 eller 5 från FIFO

F2 = " 2 " 6 " "

F3 = " 3 " 7 " "

W3I = W3

Utgångar

De två PAL-kretsarna bildar tillsammans en port som styr dataflödet mellan CPU/DMA, FIFO och WD-styrkretsen. Med hjälp av signalerna BOE, $\overline{BOE}$, $\overline{WOE}$ samt $\overline{FRD}$ kan utgångarna WO-W8 resp BO-B8 göras aktiva. I övriga fall fungerar de som ingångar.

BOE och $\overline{BOE}$ aktiverar utgångarna mot 4680-bussen, WOE-utgångarna mot WD-styrkretsen under det att FRD styr om data ska hämtas från FIFO-minnet.

Nedan följer två exempel som beskriver när respektive utgång aktiveras.

$\overline{Wx} = \overline{FRD} * \overline{Fx} +$ (1) (Aktiveras av $\overline{WOE}$)
$FRD * \overline{Bx}$ (2)

Utgång Wx står för alla W-utgångar och aktiveras då

WD-kretsen läser data från FIFO (1) samt då CPU skriver till WDregistren (2).

$\overline{Bx} = \overline{FRD} * \overline{Fx} +$ (1)

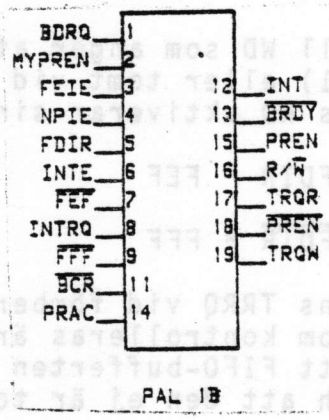
$FRD * \overline{Wx}$ (2)

Utgång Bx står för alla datautgångar till 4680 och aktiveras då CPU/DMA läser från FIFO (1) samt då CPU:n läser från WD-registren.

6.4.3 PAL 1B

Uppgift: Att hantera interrupt samt DMA-signalering till 4680-bussen.

Typ: PAL 16L8A-2



Insignaler

BDRQ = Begäran från WD att få tillgång till FIFO eller DMA

MYPREN = Skapar förutsättningar för PREN, från register 2C.

FEIE = Ger möjlighet till interrupt från FIFO

NPIE = Ger möjlighet till interrupt vid PREN

FDIR = Anger inenhet WD/CPU till FIFO

INTE = Ger möjlighet till interruptbegäran från WD och FIFO

$\overline{FEF}$ = Anger att FIFO-minnet är tomt

INTRQ = Interruptbegäran från WD

$\overline{FFF}$ = Anger att FIFO-minnet är fullt

$\overline{BCR}$ = "Buffer Counter Reset" från WD

PRAC = Svar på en PREN-signal, från Bussinterface

Utgångar

INT = INTE * INTRQ + (1)

INTE * $\overline{FEF}$ * FEIE + (2)

NPIE * PREN (3)

Interruptsignal till CPU:n. Aktiv vid interrupt från WD om ej blockerad av INTE (1), vid tom FIFO-buffert om ej blockerad av INTE eller FEIE (2) samt då PREN avges om ej blockerad av NPIE. Med hjälp av register 2C kan således CPU:n bestämma vad som ska tillåtas generera ett interrupt.

$\overline{BRDY}$ = BDRQ * $\overline{FFF}$ * $\overline{FDIR}$ + (1)

BDRQ * FEF * FDIR + (2)

$\overline{BRDY}$ * BCR (3)

Anger "Buffert Ready" till WD som anger att FIFO-minnet är fullt vid skrivning (1) eller tomt vid läsning (2). (3) håller signalen tills WD aktiverar sin BCR-utgång.

$\overline{TRQW}$ = MYPREN * PRAC * FDIR * FEF

$\overline{TRQR}$ = MYPREW * PRAC * $\overline{FDIR}$ * FFF

Signalerna ger tillsammans TRRQ vid förberedelser till en DMA-överföring. Det som kontrolleras är att PREN och PRAC signalerats, samt att FIFO-bufferten inte redan är full vid en skrivning och att den ej är tom vid en läsning.

R/$\overline{W}$ = MYPREN * FDIR

Signalen ger skriv/lässtrob till bussinterfacet då PREN avges.

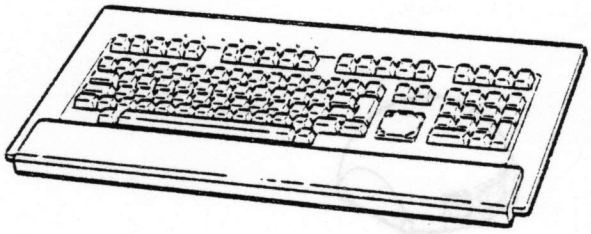
$\overline{PREN}$ = MYPREN

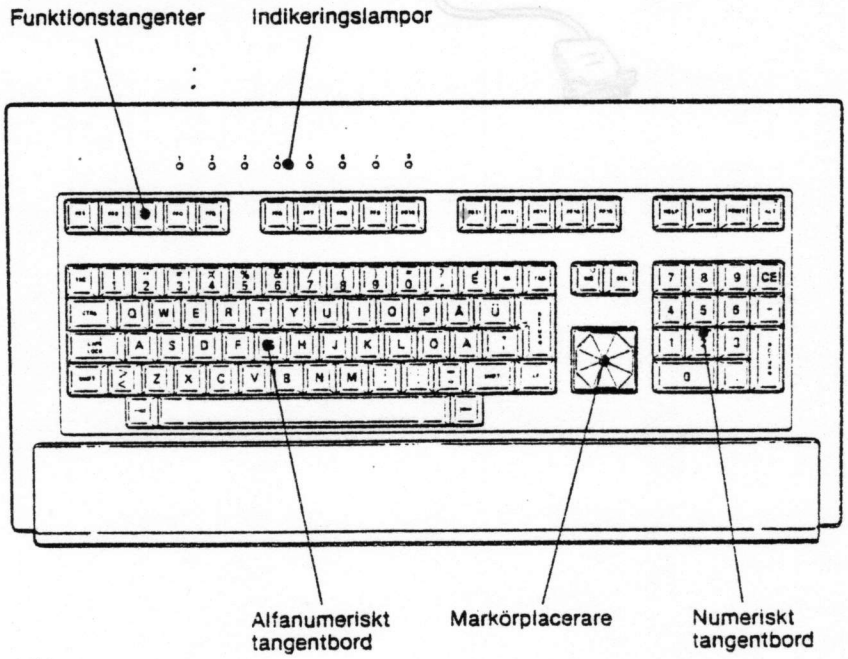
PREN = $\overline{PREN}$

PREN-signaler som avges dels till bussinterfacet, dels till statusregistret.

7 Tangentbord

ABC 99 är ett generellt tangentbord som genom sin uppbyggnad kan användas till en mängd skiftande applikationer. Det är avsett att användas tillsammans med datorsystem ABC 1600 men kan även anslutas till ABC 806. Tangentbordet har hög tillförlitlighet på grund av de kapacitiva switcharna, är av lågprofiltyp och motsvarar med sin utformning högt ställda krav på ergonomi och kvalitet.

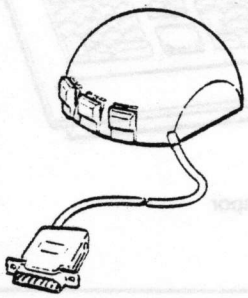




Förutom den alfanumeriska delen, som motsvarar svensk standard, finns också en separat numerisk del för sifferinmatning, placerad till höger. Det finns också 19 funktionstangenter varav 15 med generell beteckning för olika applikationer. Med markörplaceraren kan man styra markören på bildskärmen i åtta olika riktningar för att snabbt nå önskad position.

Tangentbordet har också sk fältflyttningstangenter placerade på var sin sida av mellanslagstangenten för att enklare kunna hoppa mellan fält vid inmatning i formulär. På ovansidan finns också åtta st lysdioder som kan användas för att indikera kommunikationsstatus vid användning som terminal.

Till ABC 99 kan också anslutas en "MUS" genom vilken menyval och markeringar enkelt kan göras men den kan också användas som "Penna" vid framställning av ritningar och symboler med hjälp av grafikeditorer. Kommunikationen med tangentbordet är dubbelriktad dvs tecken kan inte bara sändas från tangentbordet till datorsystemet utan data kan också sändas till tangentbordet för att sätta vissa parametrar och på detta sätt förändra egenskaperna.



7.1 Tekniska egenskaper

De viktigaste tekniska egenskaperna är följande:

- Inbyggd högtalare som kan leverera signal av 300 ms längd då ASCII-kod 7 sänds till tangentbordet. Ljudgeneratorn lämnar också en kort puls (5 ms valbart 20 ms) vid nedtryckning av någon tangent, vilket ger en känsla av kvittens. Ljudet kan med kommando helt stängas av om så önskas.

- Kommunikation sker med 8 bitars seriell kod full duplex, med två stoppbitar vid sändning och en stoppbit vid mottagning. Kommunikationshastigheten är ca 8000 bit/sek. Till datorsystemet levereras en klocka som är 16 * kommunikationshastigheten, som sedan delas och används av systemet som mottagningsklocka.

- Automatisk repetering av tecken då en tangent varit nedtryckt mer än 500 ms. Repetitionsfrekvensen är sedan 60 ms. Funktionen kan stängas av med kommando. Tangent nr 16, 17 18, 19 och 35 är undantagna från automatisk repetering även i normalfallet.

- Programmässig och hårdvarumässig "watchdog" övervakar så att inget oförutsätt inträffar och återstartar tangentbordet till normaläge efter ett spänningsbortfall eller programfel.

- "N key rollover" som innebär att då flera tangenter samtidigt är aktiverade, läses de i den ordning de trycktes. Dvs då man skriver händer det lätt att man trycker på en tangent innan föregående släppts upp kommer detta trots allt att resultera i korrekta koder ut.

- INS, ALT och CAPS LOCK-tangenterna har inbyggda lysdioder som indikerar att de är aktiverade. Funktionerna kan även aktiveras genom kommando.

- Inbyggd resetswitch som initierar tangentbordet samt sänder reset till datorsystemet (Obs! Påverkar endast tangentbordet då ABC 1600 används).

- Inbyggd spänningsstabilisator omvandlar inkommande spänningsmatning (+8-+12V/800mA) till +5V).

- UP/DOWN-mod, innebär att tangentbordet sänder en unik kod för varje tangentposition. Genererar en kod vid tryckning och samma kod +80H då man släpper tangenten. Varken automatisk repetering eller skiftfunktioner genereras i detta läge. Styrs med kommando, avkodningen sker i datorsystemet.

- Kommandosekvenserna som kan användas för att styra tangentbordets funktioner följer.

7.2 Kommandokoder

KOD (Binär)		(Hex)	FUNKTION
0000	0111	07	Ljudsignal avges från TB
0000	0001	01	Omöjliggör alarmfunktion
1000	0001	81	Möjliggör "
0000	0010	02	Stänger av kort tangentklick
1000	0010	82	Sätter på " "
0000	0100	04	Stänger av långt tangentklick
1000	0100	84	Sätter på " "
0000	0101	05	Stänger av automatisk repetering
1000	0101	85	Sätter på " "
1000	0110	86	Aktiverar UP/DOWN-mod
0000	0110	06	Aktiverar ASCII-mod (Normal)
0000	1000	08	Återställer CAPS LOCK-tangent
1000	1000	88	Aktiverar CAPS-LOCK-funktionen
0000	1001	09	Släcker INS-tangenten
1000	1001	89	Tänder INS-tangenten
0000	1010	0A	Släcker ALT-tangenten
1000	1010	8A	Tänder ALT-tangenten
0001	1000	18	Tangentbordet svarar med en identifikationskod (se kap.7.8)
0011	1001	39	Skriv data till "musens" positionsregister
0011	1010	3A	Skriv data till "musens" skalningsregister
0011	1011	3B	Skriv data till "musens gränsregister
0011	1100	3C	Läs data från "mus"
0011	1110	3E	Läs data kontinuerligt från "mus"
0011	1111	3F	Stäng av kontinuerlig läsning från "mus"
0000	0000	00	Tänd lysdiod 1

1000	0000	80	Släck	"	"
0001	0000	10	Tänd	"	2
1001	0000	90	Släck	"	2
0010	0000	20	Tänd	"	3
1010	0000	A0	Släck	"	3
0011	0000	30	Tänd	"	4
1011	0000	B0	Släck	"	4
0100	0000	40	Tänd	"	5
1100	0000	C0	Släck	"	5
0101	0000	50	Tänd	"	6
1101	0000	D0	Släck	"	6
0110	0000	60	Tänd	"	7
1110	0000	E0	Släck	"	7
0111	0000	70	Tänd	"	8
1111	0000	F0	Släck	"	8

7.3 Hantering av "musen"

Musen levererar data till tangentbordet med hjälp av fyra seriella signaler som avkodas av tangentbordet så att riktning i X- och Y-led kan avkodas. Den lämnar dessutom tre signaler från knapparna på "musen" och deras status kan läsas av från tangentbordet. I tangentbordet tas data från musen omhand och lagras i två register, ett i X-led och ett i Y-led. Här finns också ett antal register som jämförs med inkommande data, så att max- och min värden, skalfaktorer och uppräkningsfaktorer kan sättas redan i tangentbordet. Detta gör att datasystemet får data från musen serverat på önskat sätt och behöver inte ta upp tid med omräkning.

Alla register är 12 bitar långa och delas i två bytes på 6 bitar var innan de skickas till datorsystemet. Innan data sänds till systemet adderas ett offsetvärde på 20H till varje byte så att värdet kommer att ligga mellan 20H och 5FH, vilket gör att alla värden ligger inom det område som innehåller skrivbara tecken. Detta för att undvika att värden som ESC och liknande sänds till datorsystemet.

Nedan följer en beskrivning av de kommandosekvenser som skickas till tangentbordet för att sätta musens register, samt registrens initialvärden vid start.

Register	Initialvärde (HEX)
X-position	000
Y-position	000
X-min	000
X-max	FFF
Y-min	000
Y-max	FFF
X-skalfaktor	001
Y-skalfaktor	001
X-uppräkning	001
Y-uppräkning	001

Skrivning av "mus"-register:

Programmering av position: (lb = låg byte, hb = hög byte)

0011 1001, X-pos hb, X-pos lb, Y-pos hb, Y-pos lb (5 bytes)

Programmering av skalfaktor och uppräkning:

0011 1010, X-skal hb, X-skal lb, Y-skal hb, Y-skal lb, X-uppr. hb, X-uppr. lb, Y-uppr. hb, Y-uppr. lb (9 bytes).

Programmering av gränsvärden:

0011 1011, X-min hb, X-min lb, X-max hb, X-max lb, Y-min hb, Y-min lb, Y-max hb, Y-max lb (9 bytes).

Läsning av "mus":

Beroende av vilket kommando som sänds till tangentbordet kan man läsa av registren en (1) gång eller också kontinuerligt så att data sänds i en enda ström tills stoppkommando ges. I den kontinuerliga moden sänds dock data endast om något register ändrat värde.

Då koden 0011 1100 (3C H) sänds till tangentbordet svarar det med att sända tillbaka följande sekvens en gång.

1001 0000, X-pos hb, X-pos lb, Y-pos hb, Y-pos lb, knappstatus (6 bytes).

Sänds koden 0011 1110 (3E H) till tangentbordet svarar det med följande sekvens varje gång registren ändrar sig.

1001 0001, X-pos hb, X-pos lb, Y-pos hb, Y-pos lb (5 bytes) eller då knappstatus ändrat sig:

1001 0010, knappstatus (2 bytes). (Första gången sänds bägge).

Den kontinuerliga moden stoppas då 0011 1111 (3F H) sänds till tangentbordet.

Avkodningen av tangentbordet fortsätter som vanligt och dessa koder kan sändas mellan varje sekvens men bryter den ej.

I följande tabell kan de koder som tangentbordet sänder till datorsystemet studeras. Tangentnumret refererar till de angivna i kap. 7.7. Studera även kap. 7.8 för ytterligare information om teckentyp. UP/ DOWN-koden motsvarar en tryckning av resp knapp. Då knappen släpps upp adderas detta värde med 80H då koden sänds till datorsystemet.

7.4 Tangentkoder

Tangentkoder, Svensk version

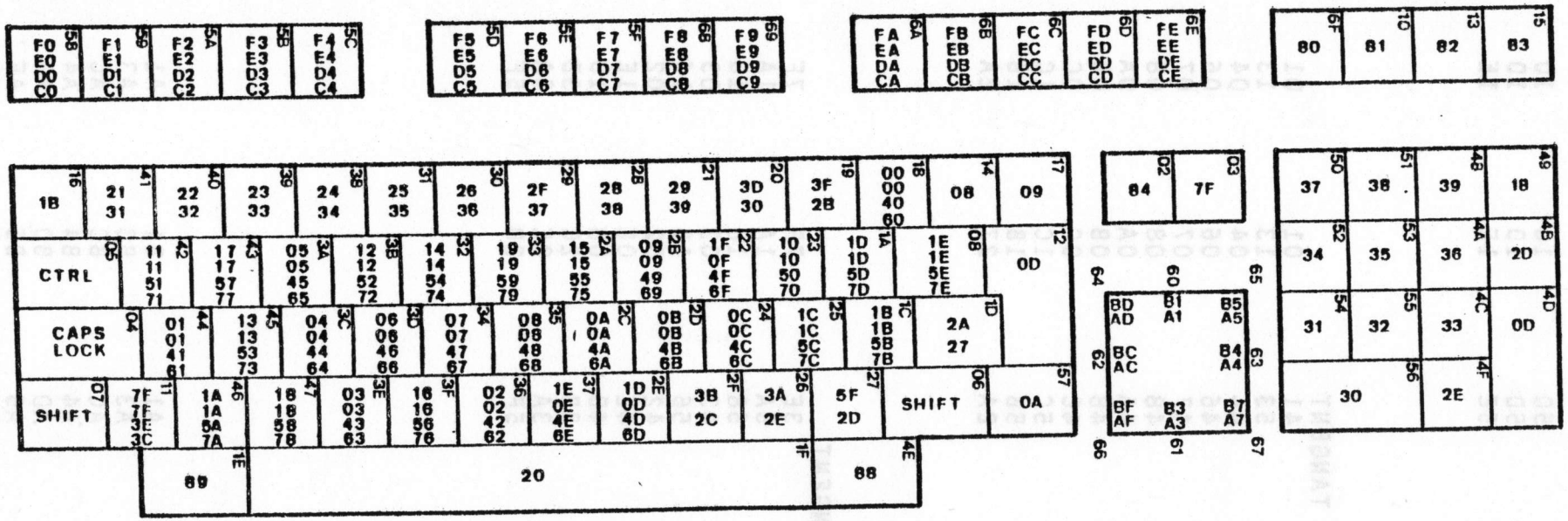
TANGENT NR.	OSKIFTAD HEX	SKIFTAD HEX	CONTROL HEX	CTRL/SK HEX	UP/DOWN HEX	CAPS LOCK PÅVERKAR
1	C0	D0	E0	F0	58	
2	C1	D1	E1	F1	59	
3	C2	D2	E2	F2	5A	
4	C3	D3	E3	F3	5B	
5	C4	D4	E4	F4	5C	
6	C5	D5	E5	F5	5D	
7	C6	D6	E6	F6	5E	
8	C7	D7	E7	F7	5F	
9	C8	D8	E8	F8	68	
10	C9	D9	E8	F8	69	
11	CA	DA	EA	FA	6A	
12	CB	DB	EB	FB	6B	
13	CC	DC	EC	FC	6C	
14	CD	DD	ED	FD	6D	
15	CE	DE	EE	FE	6E	
16	80				6F	
17	81				10	
18	82				13	
19	83				15	
20	1B				16	
21	31	21	31	21	41	
22	32	22	32	22	40	
23	33	32	33	32	39	
24	34	24	34	24	38	
25	35	25	35	25	31	
26	36	26	36	26	30	
27	37	2F	37	2F	29	
28	38	28	38	28	28	
29	39	29	39	29	21	
30	30	3D	30	3D	20	
31	2B	3F	2B	3F	19	
32	60	40	00	00	18	
33	08				14	
34	09				17	
35	84				02	
36	7F				03	
37	37				50	
38	38				51	
39	39				48	
40	18				49	
41	CTRL TANGENT				05	
42	71	51	11	11	42	*
43	77	57	17	17	43	*
44	65	45	05	05	3A	*
45	72	52	12	12	3B	*
46	74	54	14	14	32	*
47	79	59	19	19	33	*
48	75	55	15	15	2A	*
49	69	49	09	09	2B	*
50	6F	4F	0F	1F	22	*

TANGENT NR.	OSKIFTAD HEX	SKIFTAD HEX	CONTROL HEX	CTRL/SK HEX	UP/DOWN HEX	CAPS LOCK PÅVERKAR
51	70	50	10	10	23	*
52	7D	5D	1D	1D	1A	*
53	7E	5E	1E	1E	08	*
54	0D				12	
55	34				52	
56	35				53	
57	36				4A	
58	2D				4R	
59	CAPS LOCK TANGENT				04	
60	61	41	01	01	44	*
61	73	53	13	13	45	*
62	64	44	04	04	30	*
63	66	46	06	06	3D	*
64	67	47	07	07	34	*
65	68	48	08	08	35	*
66	6A	4A	0A	0A	20	*
67	6B	4B	0B	0B	2D	*
68	6C	4C	0C	0C	24	*
69	7C	5C	1C	1C	25	*
70	7B	5B	1B	1B	1C	*
71	27	2A	27	2A	1D	
72						
73	31				54	
74	32				55	
75	33				4C	
76	0D				4D	
77	SKIFT TANGENT				07	
78	3C	3E	7F	7F	11	
79	7A	5A	1A	1A	46	*
80	78	58	18	18	47	*
81	63	43	03	03	3E	*
82	76	56	16	16	3F	*
83	62	42	02	02	36	*
84	6E	4E	0E	1E	37	*
85	6D	4D	0D	1D	2F	*
86	2C	3B	2C	3B	2E	
87	2E	3A	2E	3A	26	
88	2D	5F	2D	5F	27	
89	SKIFT				06	
90	08				57	
91	30				56	
92	2E				4F	
93	89				1F	
94	20				1F	
95	88				4E	

72

Upp	A1	B1	A1	B1	60
Ned	A3	B3	A3	B3	61
Vänster	AC	BC	AC	BC	62
Höger	A4	B4	A4	B4	63
Upp-vänster	AD	BD	AD	BD	64
Upp-höger	A5	B5	AF	BF	66
Ned-höger	A7	B7	A7	B7	67

ABC99 SWE·FIN



7.5 Kort beskrivning av elektronik

Elektroniken i ABC 99 är mycket enkelt uppbyggd och består i pricip av de två mikroprocessorerna Z2 och Z5 samt tangentavkodaren Z1. Arbetsuppgifterna mellan de två processorerna är så fördelade att Z2 hanterar avkodningen av tangenterna samt all kommunikation med datorsystemet under det att Z5 hanterar all behandling av "musen".

Programvaran som styr processorerna samt tabellerna för teckenavkodning finns i PROM-kretsarna Z3 och Z6. Från processorernas muxade adress/databuss separeras signalerna i registren Z4 resp Z7 så att adress till PROM-arna kan ges samtidigt som data läses. Till huvudprocessorns kombinerade data/adressbuss är också lysdiodregistret Z8 kopplat, där varje diod individuellt kan tändas eller släckas. Med hjälp av utgångarna P13-P15 aktiveras dioderna till INS, ALT oh CAPS LOCK-tangenterna och utgång P16 levererar ljudsignal till högtalarna. Bygeln Z14 indikerar vilken landsvariant som gäller och dess läge erhålls som en del av identifieringskoden.

Kommunikationen med datorsystemet sker via kontakt J1. Tangentbordet får kraftmatning stift 1 och 2 som omvandlas till +5V i den linjära spänningsstabilisatorn 7805. Detta minskar risken för spänningsfall och därigenom störningsrisken pga långa kablage. Genom stift 8 i J1 levereras tangentbordet baudrateklocka till DART:en och via stift 7, seriedata. På stift 9 kommer data från datorsystemet och via stift 11 talar tangentbordet om att data kommer att sändas med hjälp av den så kallade KEY-DOWN-stroben. Den sista signalen är reset och kan både sändas eller tas mot. Observera att detta ej gäller ABC 1600. Från resetknappen J4 påverkas dels stift 10 och dels monovipporna Z9 som ger resetsignal till tangentbordets huvudprocessor. Monovipporna kommer också att reagera för även kortvariga spänningsbortfall vilket gör att tangentbordet automatiskt återstartar i dessa lägen.

Avkodningen av tangentmatrisen hanteras av Z1 som sänder ut en puls på någon av utgångarna X0-X15 samtidigt som resultatet registreras på ingångarna Y0-Y7.

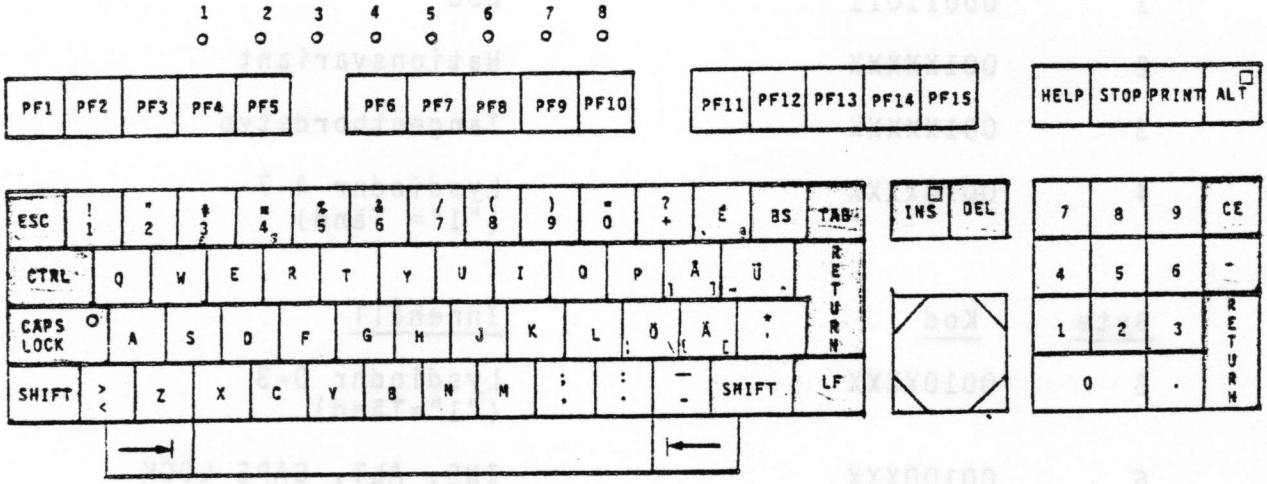
Då en tangentposition är nedtryckt sänds dessa data till processorn som nu kan leta i tabellen efter den kod som ska sändas beroende på position och landsvariant.

Slavprocessorn Z5 hanterar "mus"-interfacet via kontakt J3. "Musen" lämnar fyra seriella signaler (stift 2-5), två i X-led och två i Y-led som beskriver riktning och hastighet i förflyttningen.

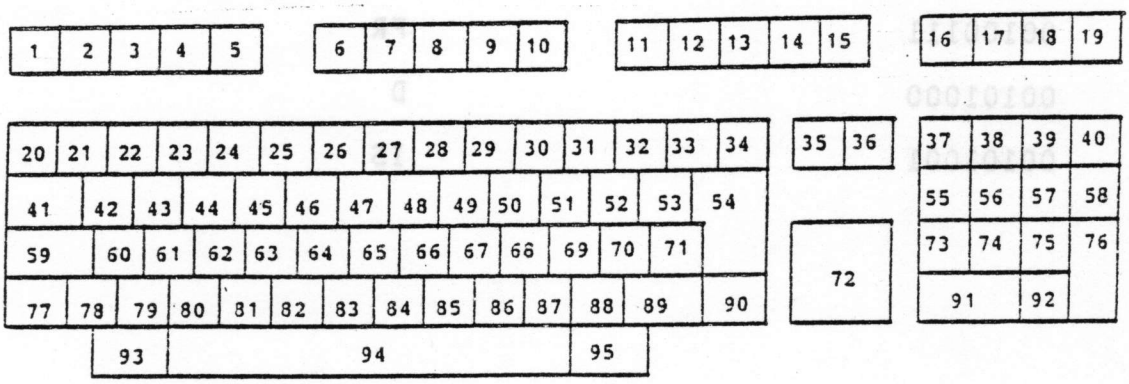
Processorn uppdaterar de register som tidigare beskrivits så att data hela tiden finns tillgängliga och kan läsas ut om kommando och läsning sker. Data från slavprocessorn transporteras via huvudprocessorn där sändningen sedan sker till datorsystemet. Via stift 6-8 kan även status på musens knappar registreras, samt via stift 14 om "musens" är ansluten.

Via kontakt J2 kan också ett extra numeriskt tangentbord anslutas som nu också hanteras av slavprocessorn.

7.6 Tangentbordslayout



7.7 Tangentnummer



7.8 Identifikationskod

Då datorsystemet begär identifikation genom att sända koden 18 H, svarar tangentbordet med att sända tillbaka följande kod omfattande sju bytes:

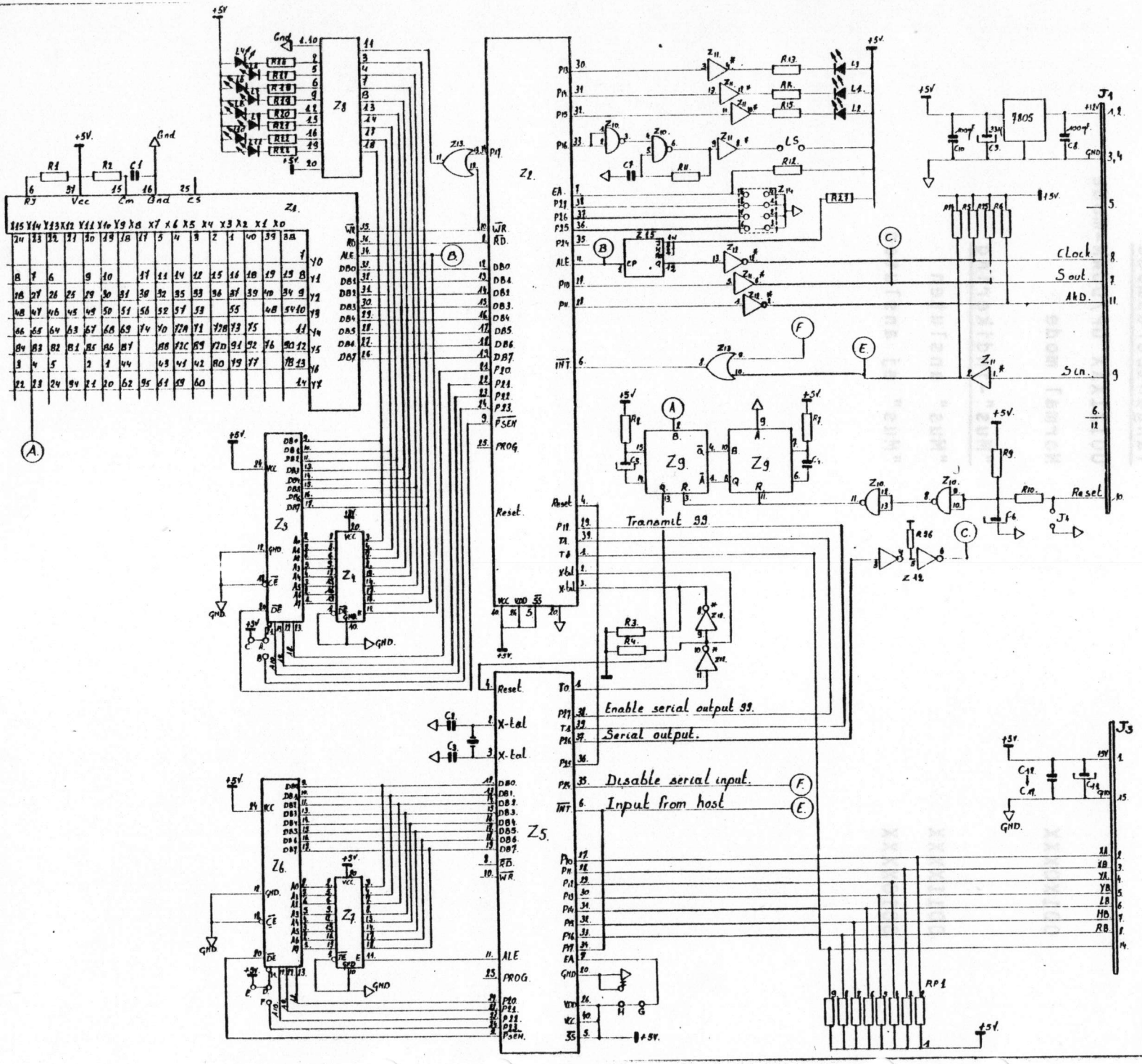
Byte	Kod	Innehåll
1	00011011	ESC
2	001XXXXX	Nationsvariant
3	001XXXXX	Tangentbordstyp
4	0010XXXX	Lysdiodnr 4-7 ("1"= Tänd)

Byte	Kod	Innehåll
5	0010XXXX	Lysdiodnr 0-3 ("1"=Tänd)
6	00100XXX	INS, ALT, CAPS LOCK ("1"=Tänd)
7	00100000	Reserverad

Byte 2	Nation
00100001	S
00100010	N
00100011	DK
00100100	USA
00100101	GB
00100110	ES
00100111	FR
00101000	D
00101001	IS

Byte 3	Tangentbordstyp
001XX001	ABC 55
001XX010	ABC 77
001XX011	ABC 99
	Tangentbordsmode
	001X1XXX UP/DOWN-mode
001X0XXX	Normal mode
	"Mus"-indikering
0011XXXX	"Mus" ansluten
0010XXXX	"Mus" ej ansluten

7.9 Kretsschema



GENERAL INSTRUMENT Computer Product's Division C.P. Clare International N.V.	ckc 9337 A42. Luxor. P.C.B.: 106143-64-1.

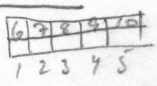
7.10 Datoranslutning

Stift	Funktion
1	+12V DC
2	+12V DC
3	Jord
4	Jord
5	Förväxlingsskydd
6	Ej ansluten
7	Data, sändning
8	Baudrate klocka
9	Data, mottagning
10	Reset
11	"Key down"
12	NC

"Mus"-anslutning

Amiga

Apollomus



Stift		Funktion	
1	7	5V/50 mA	10 (vit)
2	2	XA	
3	4	XB	
4	1	YA	
5	3	YB	
6	6	Vänster knapp	6 (svart)
7	5	Mittknapp	7 (blå)
8	9	Höger knapp	2 (röd)
9-13	—	NC	
14	—	"Mus" ansluten	
15	8	Jord	1 (tjock svart) 3 (rosa)

8 Reservdelslista

ABC 1600 art.nr. 230 8401-16/17

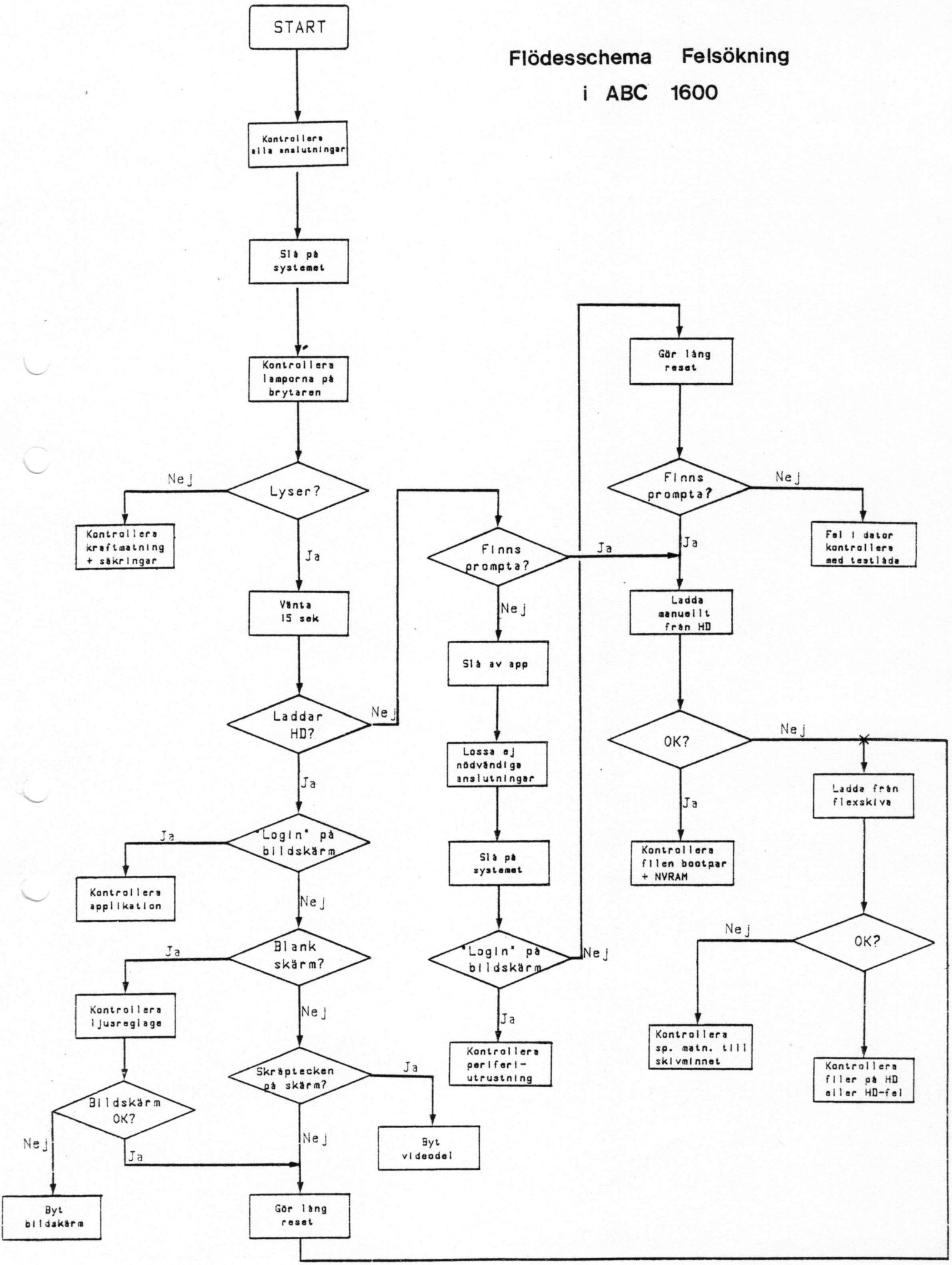
Lock - botten	40 98440-16
Hylsdon 15 pol	43 60365-01
Din-kont 7 pol	43 60397-02
Hylsdon 25pol D-sub	43 60663-01
Stiftdon 5pol amp	43 60728-01
Kont.hus 5pol	43 60729-01
Hylsdon 15pol D-sub	43 60843-01
Hylslist 15pol vinkl.	43 60844-01
Hylslist 40pol vinkl.	43 60845-01
Hylslist 70pol	43 60846-01
Kabelstam	43 71620-01
Kabel contr.-floppy	43 71634-01
Kabel host-w.contr.	43 71635-01
Videokabel mon. -dator	43 71813-01
Videokabel intern	43 71814-01
Videok. intern med skärm	43 71814-10
Videokabel 15- pol	43 71815-01
Lucka	44 20876-01
Lock KPL	44 70041-01
Botten KPL	44 70042-01
Front bearbetat	44 70055-01
Busskort	55 21081-01
Kommunikationskort	55 21082-01
Processokort ABC 1600	55 21100-02
Videokort X-35	55 21101-01
SASI interface	55 21111-01
KK reset	55 21147-01
Balkong CPU ABC 1600	55 21225-01
Floppy disk drive	55 31067-02
Nätdel rovsing	55 50897-01
Winch.drive BASF 6188 (variant 16)	55 31068-01
Contr. XEBEC S1410 A (variant 16)	55 50932-02
Winch.drive NEC D5126 (variant 17)	55 31070-01
Contr. ACB 4000 (variant 17)	55 50038-01
Strömställare	56 00057-01
Tryckknapp	56 40133-01
Beröringsskydd strömställare	56 95104-01
Nätfilter med säkring	59 20008-01
Digital delay line	59 60080-01
Fläkt Papst 4112 GXL	60 10011-01
22R 5% 1/3W	61 29247-01
47R 5% 1/3W	61 29249-01
100R 5% 1/3W	61 29250-01
220R 5% 1/3W	61 29252-01
270R 5% 1/3W	61 29253-01
470R 5% 1/3W	61 29256-01
680R 5% 1/3W	61 29257-01
1K 5% 1/3W	61 29258-01
2.2K 5% 1/3W	61 29260-01
2.7K 5% 1/3W	61 29261-01

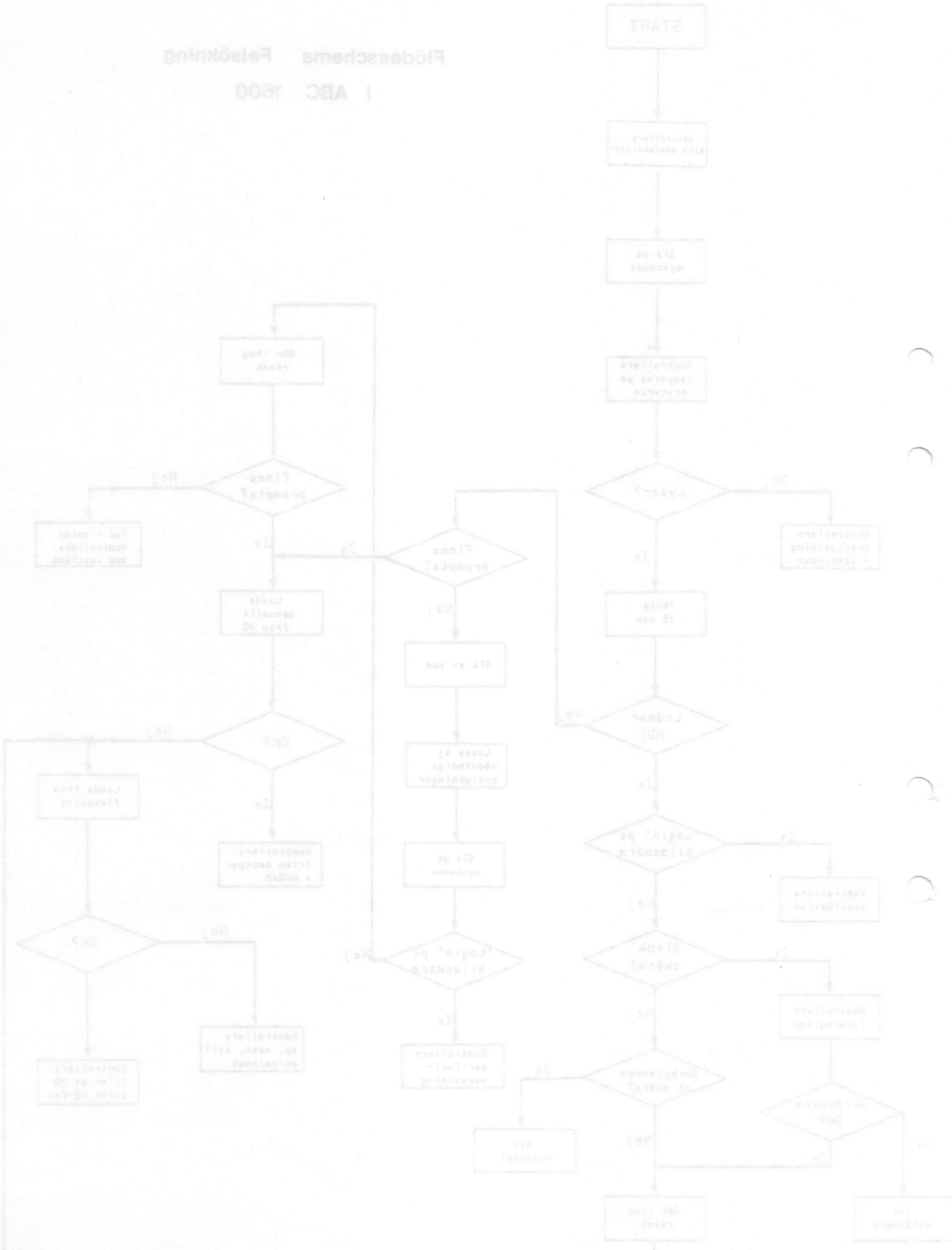
10K 5% 1/3W	61 29265-01
47K 5% 1/3W	61 29272-01
100K 5% 1/3W	61 29274-01
1.2K 5% 1/3W	61 29290-01
12R 5% 1/3W	61 29442-01
10R 5% 1/3W	61 29671-01
15R 5% 1/3W	61 29672-01
VDR	61 30010-01
SIL 1K X 9	61 90040-01
SIL 4.7K X 9	61 90041-01
DIL 47R X 8	61 90044-01
SIL 470R X 9	61 90049-01
SIL 1KX8	61 90051-01
SIP 220/330 X 6	61 90053-01
DIL 22R X 8	61 90059-01
DIL 330R X 8	61 90065-01
220PF KER	62 00014-01
100NF KER	62 00039-01
33PF KER	62 00047-01
4.7NF KER	62 00052-01
470PF	62 00055-01
10NF	62 21342-01
100NF KER	62 21871-01
10UF TANTAL	62 50021-01
47UF TANTAL	62 50151-01
TRIMKOND 2.5 - 27 PF	62 80016-01
1N 4148	63 08824-01
BC 547 B	63 10011-01
BC 557 B	63 10057-01
CQY 40 LED	63 40136-01
BZX 83 C3V0	63 40139-01
BZX 83 C3V6	63 40229-01
IC-HÅLLARE 8PIN	63 80009-01
IC-HÅLLARE 14PIN	63 80023-01
IC-HÅLLARE 16PIN	63 80024-01
IC-HÅLLARE 18PIN	63 80025-01
IC-HÅLLARE 20PIN	63 80026-01
IC-HÅLLARE 24PIN	63 80027-01
IC-HÅLLARE 40PIN	63 80029-01
IC-HÅLLARE 28PIN	63 80040-01
IC-HÅLLARE 48PIN	63 80045-01
X-TAL 32768 HZ	63 90053-01
OSC 64 MHZ	63 90067-01
LM 2903	64 20017-01
SN 7406	64 40000-01
SN 74 LS 257	64 40046-01
SN 74 LS 273	64 40048-01
SN 74 LS 283	64 40049-01
SN 74 LS 375	64 40050-01
SN 74 LS 393	64 40051-01
SN 74 ALS 138	64 40052-01
SN 7474	64 40071-01
SN 74 LS 125A	64 40083-01
SN 74 LS 367A	64 40084-01
SN 74 S 240	64 40095-01
SN 74 LS 51	64 40098-01
SN 74 ALS 112	64 40099-01
SN 74 LS 174	64 40110-01
SN 74 ALS 02	64 40155-01
SN 74 ALS 32	64 40156-01

SN 74 ALS 74	64 40157-01
SN 74 ALS 175	64 40159-01
SN 74 ALS 10	64 40162-01
SN 74 ALS 00	64 40183-01
SN 74 ALS 04	64 40184-01
SN 74 ALS 27	64 40185-01
SN 74 ALS 157	64 40187-01
SN 74 ALS 158	64 40188-01
SN 74 ALS 163	64 40189-01
MEM E 050-16	64 40198-01
F 74 F 02	64 40199-01
F 74 F 189	64 40200-01
SN 74 ALS 1240	64 40205-01
SN 74 LS 259	64 40206-01
8T 26A	64 40227-01
F 74 F 00	64 40228-01
F 74 F 112	64 40229-01
F 74 F 280	64 40230-01
F 74 F 352	64 40231-01
F 74 F 353	64 40232-01
F 74 F 64	64 40233-01
F 74 F 74	64 40234-01
HEF 4737 B	64 40235-01
SN 74 ALS 08	64 40237-01
SN 74 ALS 139	64 40239-01
SN 74 ALS 174	64 40240-01
SN 74 ALS 241	64 40242-01
SN 74 ALS 244	64 40243-01
SN 74 ALS 244-1	64 40244-01
SN 74 ALS 245-1	64 40245-01
SN 74 ALS 37	64 40246-01
SN 74 ALS 51	64 40247-01
SN 74 LS 19	64 40249-01
SN 74 LS 219 A	64 40250-01
SN 74 LS 490	64 40252-01
SN 74 S 133	64 40254-01
SN 74 S 260	64 40255-01
SN 74 ALS 137	64 40257-01
SN 74 ALS 245	64 40259-01
SN 74 ALS 30	64 40260-01
SN 74 ALS 373	64 40261-01
SN 74 ALS 169	64 40265-01
SN 74 ALS 253	64 40266-01
SN 74 ALS 374	64 40267-01
F 74 F 08	64 40268-01
F 74 F 11	64 40269-01
F 74 F 32	64 40270-01
F 74 F 86	64 40271-01
F 74 F 163	64 40272-01
F 74 F 175	64 40273-01
F 74 F 194	64 40274-01
F 74 F 244	64 40275-01
F 74 F 350	64 40277-01
F 74 F 374	64 40278-01
F 74 F 374 FAIRCHILD	64 40278-02
F 74 F 521	64 40279-01
SN 74 S 38	64 40288-01
F 74 F 20	64 40289-01
F 74 F 157	64 40290-01
F 74 F 534	64 40291-01

7649 64 90361-01	64 60002-01
IMS 2600-10	64 60003-01
TMM 41256-15	64 60004-01
PAL16 L8A 64 90349-02	64 70014-01
SY 68 45 E	64 80001-01
MC 1488 P	64 90043-01
MC 1489 AP	64 90044-01
7603 64 90358-01	64 90067-01
Z80 A DART	64 90103-01
2732 A-3 64 90363-01	64 90137-04
PAL16 R4 64 90354-01	64 90185-01
PAL16 L8 64 90360-01	64 90223-01
27 128-2 64 90356-05	64 90298-01
MC 68008	64 90300-01
PAL16R6 64 90352-02	64 90301-01
Z 8530 SCC	64 90302-01
Z 8536 CIO	64 90303-01
Z 80 A DMA	64 90304-01
FDC 9229 BT	64 90305-01
2149H-2	64 90307-01
FD 1797-02	64 90309-01
NMC 9306	64 90310-01
NI-CD BATTERY 2.4V	65 50566-01
SÄKRINGSHÅLLARE	65 80006-01
SÄKRING 1A TRÖG	65 82669-01
SÄKRING 4A TRÖG	65 83214-01

Flödesschema Felsökning i ABC 1600





10 Testlåda

Testlåda för felsökning av ABC 1600 finns att köpa ifrån Luxor AB.

Testlådans användning

Testlådan är avsedd för service-verkstäder och andra felsökare framtagen för att lokalisera svåra fel på processorkortet som gör att maskinen är "helt död" eller har mycket svårt att starta.

Lådan har 2 stk. sjusegment siffror, 8+1 lysdioder och 2 stk. omkopplare samt tre stk. flatkablar mot testobjekt. Om lådan hittar ett allvarligt fel visas felstypen på siffrorna och närmare detaljer på lysdioderna. Med testlådans bruksanvisnings hjälp kan man tolka dessa signaler och lättare lokalisera felet.

Om så mycket av kortet fungerar så att CPU:en får kontakt med en terminal, körs resten av testningen mer i klartext och meny-val.

Utrustning

1 Testlådan med dessa flatkablar.

2 En terminal som kan köra med 9600 Baud och skicka ut en baudrate-klocka. T. ex. en ABC 802 med ändrad DIP-switch.

3 En terminalkabel mellan CPU-kortet och terminalen.

4 En kraftenhet som ger +5V, +12V och -12V. Kabeln från datorenhetens kraftenhet är en aning kort men kan räcka.

5 För mer fullständig test bör även flexskiveenheten och SASI-interfacet med winchester anslutas.