Xtra-2 (Ver 2008-01-30)

Yttre signaler som påverkar exekveringen hos processorn CPU12 (Förenklad beskrivning)

Nedan beskrivs tre olika styrsignaler som på olika sätt styr exekveringen hos processorn CPU12. Signalerna kopplas in via tre olika ben ("pinnar") på microcontrollerns kapsel.

1. Reset' (Aktiv som nolla)
När en nolla utifrån läggs på Reset'-ingången upphör processorns arbete.
När sedan Reset'-ingången ges värdet 1 händer följande:

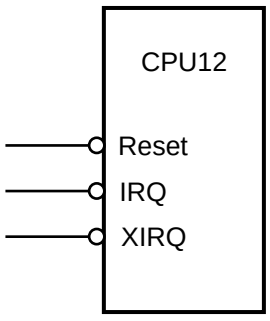
I-biten (I-flaggan), X-biten (X-flaggan), och S-biten (S-flaggan) i CC-registret ettställs. Processorn läser sedan innehållet i den sk "resetvektorn" på minnesadresserna FFFEH och FFFFH och placerar de båda 8-bitars dataorden som en 16-bitars adress i programräknaren (PC). Sedan hämtar processorn nästa instruktion från denna adress.

Genom att aktivera Reset'-signalen kan man alltså få processorn att starta om programexekveringen från den adress som finns lagrad i resetvektorn på adresserna FFFEH och FFFFH i minnet. Reset'-signalen aktiveras alltid när matningsspänningen till processorn slås på från början.

2. IRQ' (Interrupt request, Avbrottsbegäran) (Aktiv som nolla)
En yttre enhet kan få processorn att (oftast tillfälligt) byta program genom att lägga en nolla på insignalen IRQ'. Processorn exekverar då färdigt den instruktion den håller på med. Om avbrottssystemet är aktiverat, dvs om I-biten i flaggregistret (CC-registret) är nollställd, kommer processorn sedan att acceptera avbrottet . I annat fall kommer programexekveringen att fortsätta som vanligt.

Om avbrottsbegäran accepteras sparar processorn innehållen i de interna registren i ordningen PC, Y, X, A, B och CC på stacken. (Innehållet i PC är här adressen till nästa instruktion i det avbrutna programmet.) Därefter ettställs I-biten i CC-registret för att nya avbrott skall utestängas.

Processorn läser sedan innehållet i den sk "IRQ-vektorn" på minnesadresserna FFF2H och FFF3H. Detta innehåll, som är adressen till en avbrottsrutin, placeras i programräknaren. Därmed har processorn utfört ett hopp från det vanliga programmet till avbrottsrutinen.



När avbrottsrutinen börjar exekveras vet man att den händelse som orsakar avbrott har inträffat. Avbrottsrutinen skall därför utföra arbetet som är förknippat med avbrottshändelsen och sedan återvända till det vanliga (avbrutna) programmet med återställda registerinnehåll. Det finns en speciell instruktion, RTI (Return from interrupt), som återställer innehållen i samtliga processorregister genom att läsa de gamla registerinnehållen från stacken. Eftersom RTI även återställer innehållet i PC kommer exekveringen efter RTI att fortsätta i det program som tidigare blev avbrutet. En avbrottsrutin avslutas därför med instruktionen RTI.

Vid återhoppet från avbrottsrutinen är det viktigt att den aktiva signal som orsakade det pågående avbrottet har försvunnit, annars kommer ju ett nytt avbrott direkt att genereras av den "gamla" signalen efter återhoppet.

Lägg märke till att avbrottssystemet automatiskt aktiveras då de gamla registerinnehållen i processorn återställs vid återhopp från avbrottsrutinen eftersom det gamla CCR-innehållet, som hämtas från stacken, innehåller en nolla i I-biten.

3. XIRQ' (Non maskable interrupt request) (Aktiv som nolla)
XIRQ'-ingången har i princip samma funktion som IRQ'-ingången.

En sak som skiljer dem åt är att ett program inte kan utestänga XIRQ-avbrott genom att ettställa X-flaggan. Efter Reset är dock XIRQ-avbrott utestängt eftersom X-flaggan ettställs vid Reset. Vill man använda XIRQ-avbrott måste man först nollställa X-flaggan med någon av instruktionerna ANDCC #$BF, TFR A,CC, eller en RTI. Avbrottsvektorn för XIRQ-avbrott finns på adresserna FFF4H och FFF5H.

Nedan visas hur några olika vektorer är placerade i processorns adressrum.

FFF2H:	IRQ H
FFF3H:	IRQ L
FFF4H:	XIRQ H
FFF5H:	XIRQ L
FFF6H:	SWI H
FFF7H:	SWI L
FFFEH:	RESET H
FFFFH:	RESET L

SWI-vektorn ovan används av SWI-instruktionen som sparar processorns interna register och ettställer I-flaggan på samma sätt som vid ett avbrott. Den beskrivs i instruktionslistan för CPU12.

Xtra-2 (Ver 2008-01-30)

Exempel på avbrott med CPU12

Ett datorsystem med microcontrollern 68HCS12 skall användas för att köra ett huvudprogram.

Samtidigt som huvudprogrammet körs skall en variabel KNAPP (8 bitar) på minnesadressen 2800H ökas med ett varje gång en tryckknapp aktiveras.

Ökningen av variabeln KNAPP skall ske med hjälp av IRQ-avbrott genom att huvudprogrammet avbryts vid varje knapptryckning och en avbrottsrutin KNPINC då körs. Se figuren på nästa sida.

Avbrottsrutinen KNPINC skall ha startadressen 2400H i minnet.

Avbrottsvektorn för IRQ-avbrott antas finnas i RWM (läs- och skrivbart minne) på adresserna FFF2H och FFF3H.

Avbrottsrutinen på adressen KNPINC (2400H) blir:

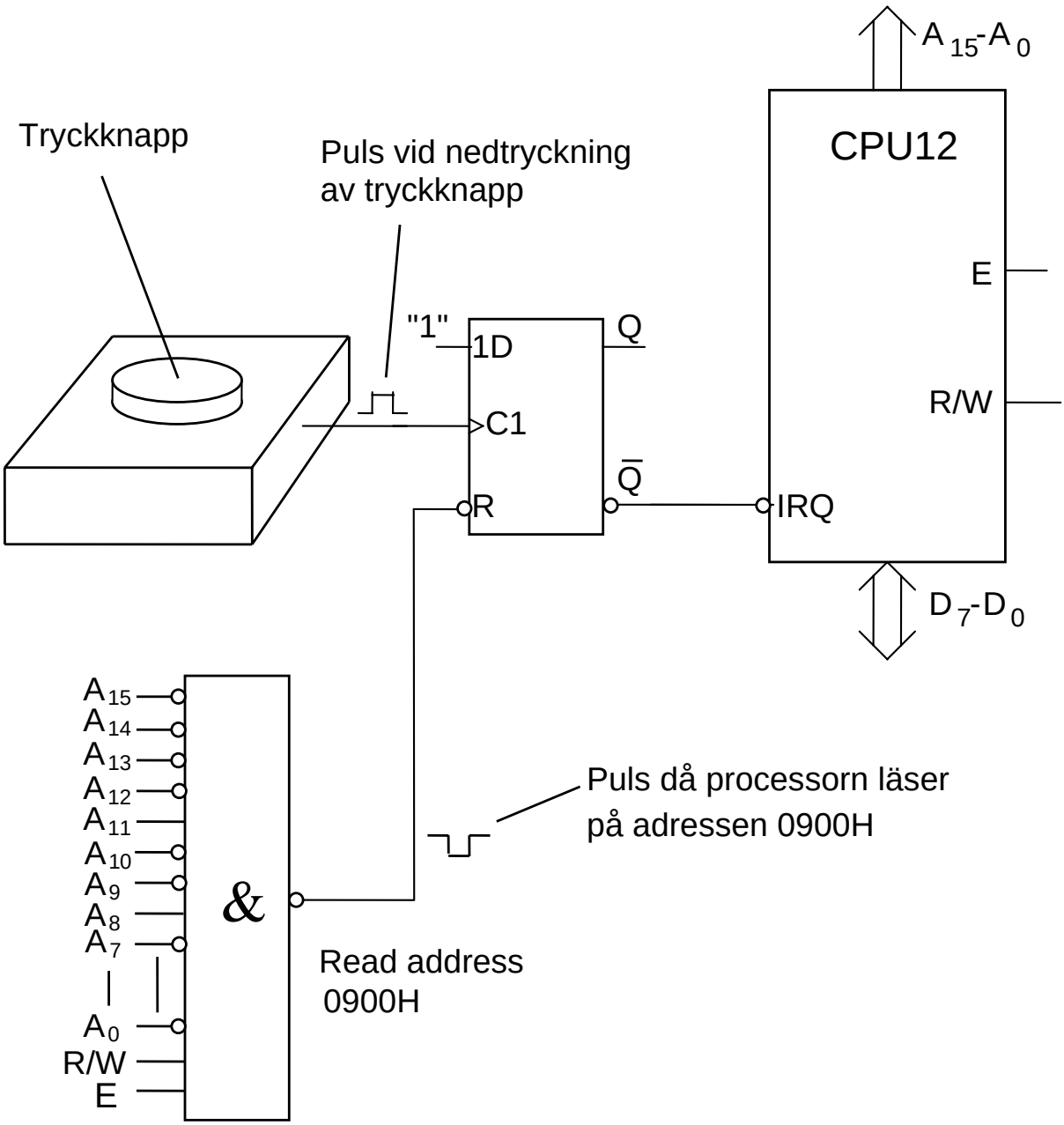
```
KNPINC  INC     $2800   Variabel KNAPP ökas med ett.
        TST     $0900   Nollställ avbrottsvippan.
        RTI             Återvänd till huvudprogrammet.
```

Nedan visas nödvändiga initieringar i huvudprogrammet för att avbrott på IRQ-ingången skall accepteras. Dessutom visas hur IRQ-vektorn ges sitt rätta värde KNPINC (2400H).

Huvudprogram:

```
MAINPG  LDS     #STACKADR   BOS definieras först.

        LDX     #KNPINC     Adr KNPINC = 2400H.
        STX     $FFF2       Ge IRQ-vektorn värdet KNPINC.

        TST     $0900       Nolla avbrottsvippan.

        CLR     $2800       Nolla variabeln KNAPP (2800H).

        CLI                 Aktivera avbrottssystemet (IRQ).
          .                 (Nolla I-biten i CC-registret.)
          .
```

Xtra-2 (Ver 2008-01-30)



Xtra-2 (Ver 2008-01-30)

In- och utmatning av data

En dator måste ha möjlighet att ta emot data från omgivningen och skicka ut data till omgivningen. Därför förses datorer med dataingångar (inportar) och datautgångar (utportar) som utgör gränssnittet mellan datorn och omvärlden.

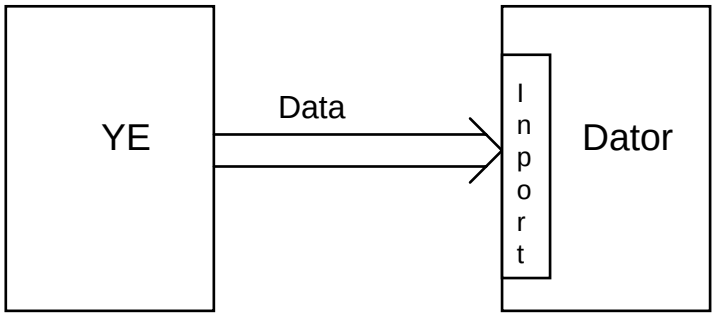
Från datorns synvinkel består omvärlden av olika yttre enheter som kan lämna data eller ta emot data.

All **inmatning** av data till en dator görs **från yttre enheter (YE)**.

All **utmatning** av data från en dator görs **till yttre enheter (YE)**.

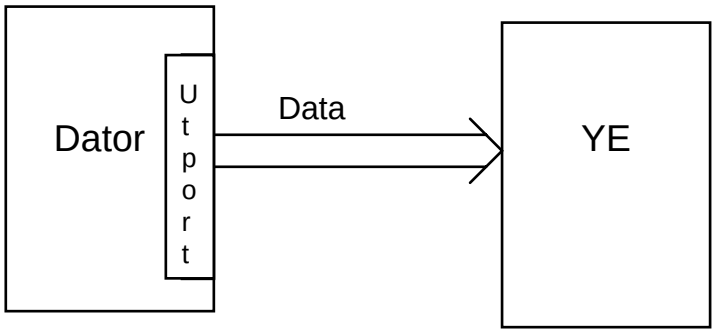
Vid **ovillkorlig in-** eller **utmatning** av data **bestämmer datorn själv in-** eller **utmatningshastigheten** utan medverkan av den yttre enheten.

Figuren nedan visar principen för ovillkorlig inmatning till en dator.



Vid ovillkorlig inmatning förutsätts att YE alltid har aktuell data tillgänglig för datorn.

Figuren nedan visar principen för ovillkorlig utmatning från en dator.

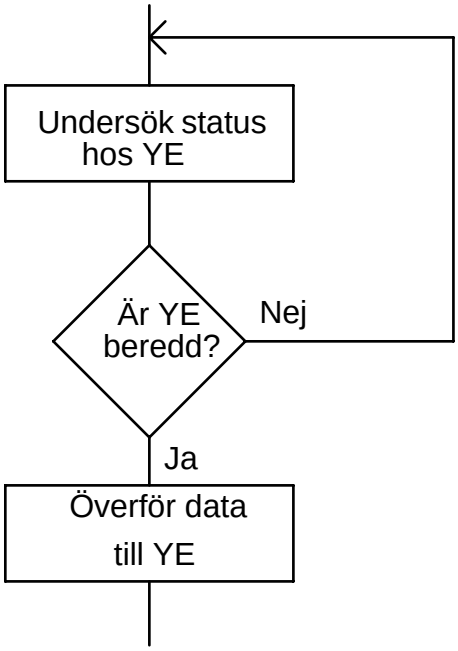


Vid ovillkorlig utmatning förutsätts att YE alltid kan ta emot data från datorn.

Vid **villkorlig** in- eller utmatning av data bestämmer den yttre enheten in- eller utmatningshastigheten genom att datorn på något sätt frågar YE om den är beredd att ta emot eller lämna data.

Datorn frågar YE om dess tillstånd genom att läsa statusinformation som YE alltid håller tillgänglig på en av datorns inportar. Av statusinformationen framgår om YE är klar för en dataöverföring. Principen för villkorlig dataöverföring framgår av flödesplanen nedan.

Xtra-2 (Ver 2008-01-30)



Statustest i vänteloop

Processorn ligger alltså i en programslinga och väntar på att YE skall bli klar för en dataöverföring. Eftersom YE ofta är en långsam enhet kommer processorn under en mycket stor del av tiden att befinna sig i vänteslingan och inte få så mycket "produktivt arbete utfört. Man säger att processorn är "busy-waiting".

I stället för att bara vänta på att YE skall bli klar kan processorn utföra nyttigt arbete om YE:s status kontrolleras på lämpliga ställen i programmet och överföringen utförs så snart status säger att YE är beredd.

En annan möjlighet att låta processorn arbeta med nyttigt arbete medan den yttre enheten gör sig klar är att låta YE signalera till processorn via avbrottssystemet att den är redo för en dataöverföring.

De olika varianterna av villkorlig överföring kan illustreras med följande exempel:

Du är hemma på ditt rum och behöver studera matte inför omtentan. Det är förmiddag och du väntar ett viktigt brev (CSN) med posten. Under väntetiden kan du bete dig på ett av följande sätt.

a) "busy-waiting"
Du står ivrigt väntande vid dörren och iakttar luckan till brevinkastet.

b) "upprepad statustest"
Du går till dörren en gång var femte minut för att se om brevet har kommit.

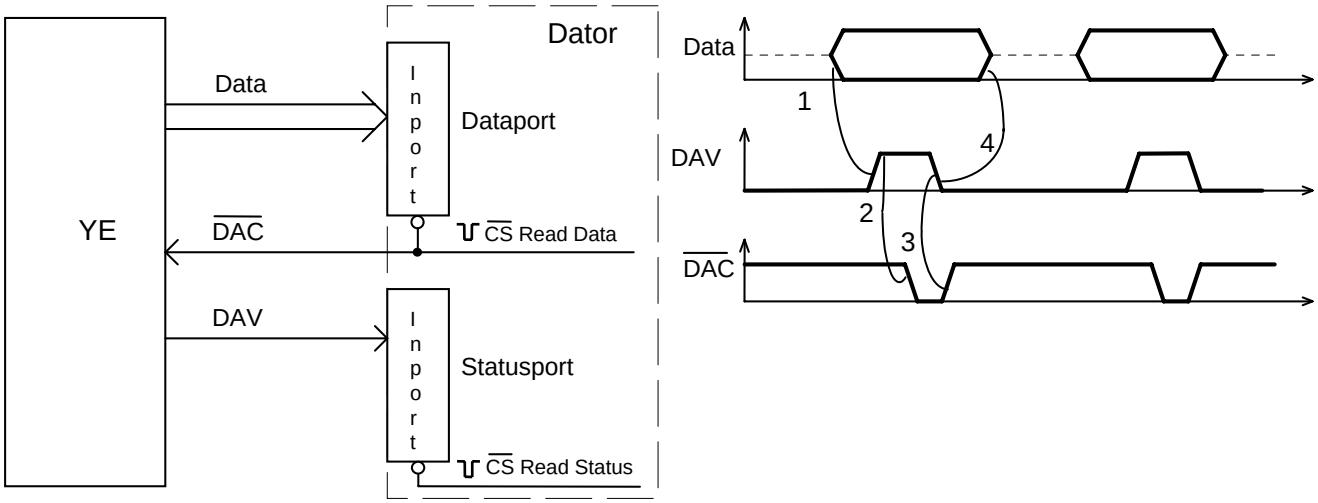
c) "avbrott"
Du har försett brevinkastet med en anordning som signalerar när posten kommer.

I de två första fallen ägnar du mycket tid åt att se om brevet kommit på bekostnad av studierna. I det sista fallet studerar du vidare tills du hör signalen att posten har kommit.

Nedan följer två tillämpningar på villkorlig överföring av data mellan yttre enhet och dator enligt principen "busy-wait".

Xtra-2 (Ver 2008-01-30)

Villkorlig inmatning av data till datorn från YE med "busy-wait".



Det signalmässiga samspelet 1-4 (markerat i diagrammet ovan) kallas "handskakning".

Exempel:
I den yttre enheten finns 32 databyte som skall överföras till datorn och placeras i dess minne på adresserna DATABUF till och med DATABUF + 31.
Yttre enhetens datautgång är kopplad till en av datorns inportar på adressen 800H. Statussignalen, DAV, är kopplad till bit noll (minst signifikant bit) på en annan av datorns inportar med adressen 801H.

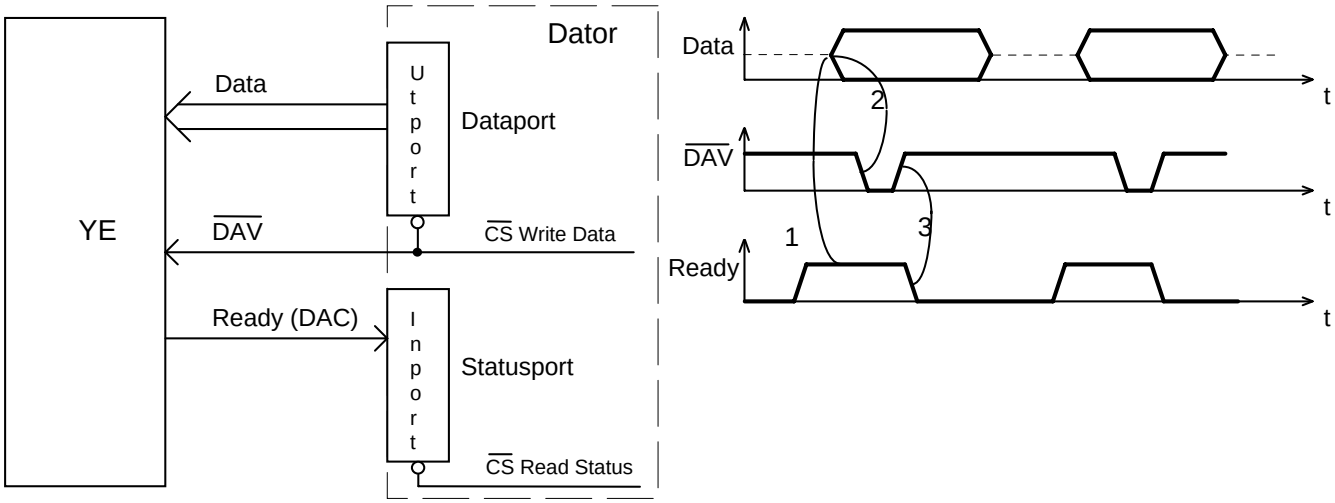
Den signal som verkställer läsningen av inporten på adress 800H (CS' Readadr 800H)
är också kopplad till den yttre enheten under namnet DAC'. När en negativ puls ( ) uanländer på
DAC' nollställer YE signalen DAV eftersom det aktuella dataordet då är läst av datorn.

Programavsnittet för inmatningen blir:

```
     LDAB      #$20       Byteräknare sätts till 20H = 32
     LDX       #DATABUF   Pekare till dataarean
WAIT LDAA      $0801      Läs status hos YE
     ANDA      #%00000001 Maska fram DAV-biten. (bit nr 0)
     BEQ       WAIT       Vänta tills DAV = 1
     LDAA      $0800      DAV = 1. Läs data från YE
     STAA      1,X+       Placera läst databyte i minnet och öka X
     DECB                 Räkna ner byteräknare med ett
     BNE       WAIT       Fortsätt med inläsning av nästa databyte
      .                   Samtliga databyte i YE lästa
      .
      .
```

Xtra-2 (Ver 2008-01-30)

Villkorlig utmatning av data från datorn till YE med "busy-wait".



Handskakningsförloppet är markerat med 1-3 i figuren ovan.

Exempel:
Datorn skall mata ut 48 databyte till den yttre enheten. De 48 dataorden är lagrade i datorns minne på adresserna DATABUF till och med DATABUF + 47.
En av datorns utportar med adressen 400H är kopplad till den yttre enhetens dataingång.
Statussignalen, Ready, från den yttre enheten är kopplad till bit noll (minst signifikant bit) på en av datorns inportar med adressen 600H.

Den signal som verkställer skrivningen av data på adress 400H (CS' Write adr 400H) är också kopplad till den yttre enheten under namnet DAV'. När en negativ puls ( ) kommer på DAV' nollställer YE signalen Ready för att ta hand om den databyte som just har kommit från datorn.

Programavsnittet för utmatningen blir:

```
     LDAB      #$30       Byteräknare sätts till 30H = 48
     LDX       #DATABUF   Pekare till dataarean
WAIT LDAA      $0600      Läs status hos YE
     ANDA      #%00000001 Maska fram Ready-biten. (bit nr 0)
     BEQ       WAIT       Vänta tills Ready = 1
     LDAA      1,X+       Ready =1. Läs data från minnet och öka X
     STAA      $0400      Skriv databyte till YE
     DECB                 Räkna ner byteräknare med ett
     BNE       WAIT       Fortsätt med utmatning av nästa databyte
      .                   Samtliga databyte i dataarean utmatade
      .
      .
```

Xtra-2 (Ver 2008-01-30)

Villkorlig inmatning av data till datorn från YE med användning av avbrott.

Exempel:
I den yttre enheten finns 32 databyte som skall överföras till datorn och placeras i dess minne på adresserna DATABUF till och med DATABUF + 31.

Hur YE är kopplad till processorn i detta fall framgår av figuren nedan.

Yttre enhetens datautgång är kopplad till en av datorns inportar på adressen 600H.

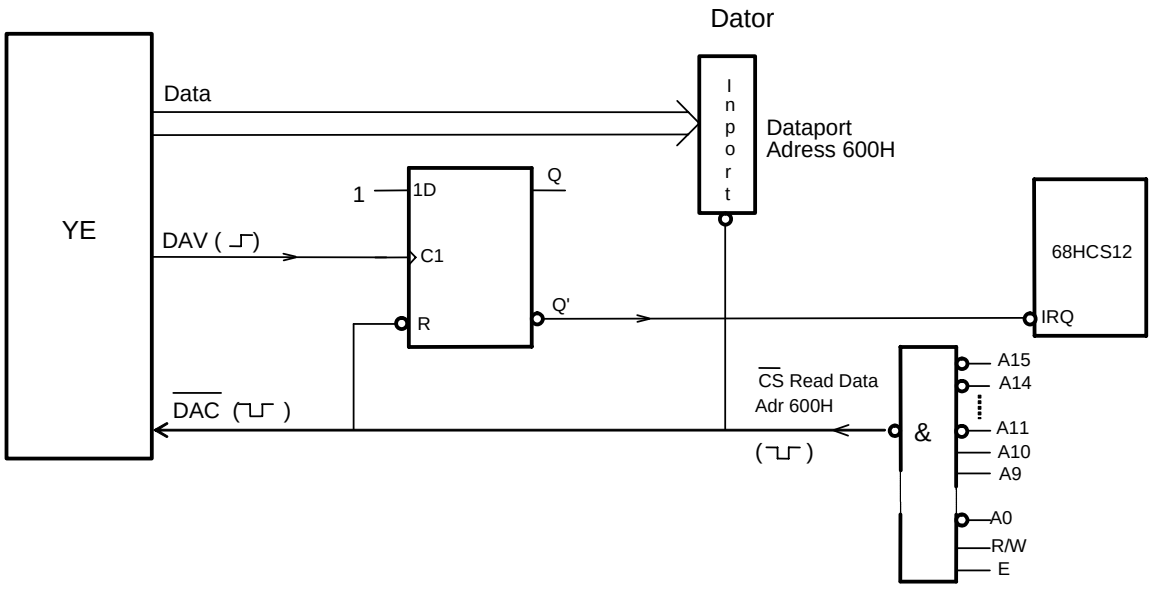
Statussignalen, DAV, är kopplad till klockingången på en flanktriggad D-vippa med ettställd D-ingång. En positiv flank på signalen DAV från YE ettställer därför D-vippan, dvs Q = 1 och Q' = 0. Eftersom vippans Q'-utgång är kopplad till processorns IRQ'-ingång får processorn då en avbrottsbegäran (IRQ' = 0).

Om processorns avbrottssystem är aktiverat (dvs I-biten i CC-registret = 0) kommer processorn att avbryta det pågående programmet, lagra samtliga registerinnehåll på stacken och sedan hoppa till en avbrottsrutin som pekas ut av IRQ-vektorn.

I avbrottsrutinen läser processorn dataordet från YE på adress 600H. CS-signalen som aktiverar inporten på adress 600H är även kopplad till D-vippans resetingång. Vid läsningen av data från YE kommer därför D-vippan att nollställas, dvs Q = 0 och Q' = 1. Eftersom Q' är kopplad till processorns IRQ'-ingång kommer avbrottsbegäran därför att upphöra.

Den signal som verkställer läsningen av inporten på adress 600H (CS' Read Adr 600H) är också kopplad till den yttre enheten under namnet DAC'. När en negativ puls ( ) anländer på DAC' nollställer YE signalen DAV eftersom det aktuella dataordet då är läst av datorn.

Programmet för inmatningen visas på nästa sida.



Xtra-2 (Ver 2008-01-30)

1. Huvudprogram

I huvudprogrammet initieras inmatningen av 32 byte från YE till minnet. IRQ-vektorn på adresserna FFF2H och FFF3H antas vara placerad i ett RWM. Stackpekaren antas vara initierad tidigare.

```
     LDX   #DATABUF   Startadress till dataarean
     STX   DATAPNT    Placera startadressen (16 bitar) för
                      dataarean på adresserna DATAPNT och
                      DATAPNT+1 i minnet

     LDAA  #32        Antal databyte som skall överföras
     STAA  BYTECNT    Placera byteräknare på adressen BYTECNT i
                      minnet

     CLR   INFLAGGA   Nollställ en flagga som visar att
                      inmatningen är klar. Denna flagga sätts
                      till FFH av avbrottsrutinen när samtliga
                      32 databyte är överförda

     LDX   #ARUT      Adressen till avbrottsrutinen
     STX   $FFF2      Placera adressen till avbrottsrutinen i
                      IRQ-vektorn. Denna antas här finnas i RWM.

     CLI              Aktivera avbrottssystemet genom att
                      nollställa I-biten i CC-registret

WAIT LDAA  INFLAGGA
     BEQ   WAIT       Vänta på att inmatningen skall bli klar
```

2. Avbrottsrutin

Hit kommer man varje gång YE signalerar att ett nytt dataord finns tillgängligt på inporten på adress 600H. Signaleringen sker genom att YE ger signalen DAV värdet ett. Detta triggar D-vippan som ger signalen IRQ' värdet noll. Dvs en avbrottsbegäran kommer till processorn.

```
ARUT LDAA $0600       Läs databyte från YE. Detta nollställer
                      också D-vippan som därmed tar bort
                      avbrottsbegäran.

     LDX  DATAPNT     Hämta datapekaren
     STAA 1,X+        Skriv databyten i dataarean och öka X

     STX  DATAPNT     Skriv tillbaka datapekaren (ökad med ett)

     DEC  BYTECNT     Minska antal återstående databytes med ett
     BNE  UT          Om vi inte är klara fortsätter vi med
                      nästa databyte när YE är färdig

     LDAA #$FF
     STAA INFLAGGA    Flagga till huvudprogrammet att samtliga
                      32 databyte är mottagna. Om vi vill kan vi
                      också stänga av avbrottssystemet här genom
                      att ettställa I-biten i det CC-värde som
                      finns i stacken

UT   RTI              Återvänd till huvudprogrammet
```