Avbrottshantering

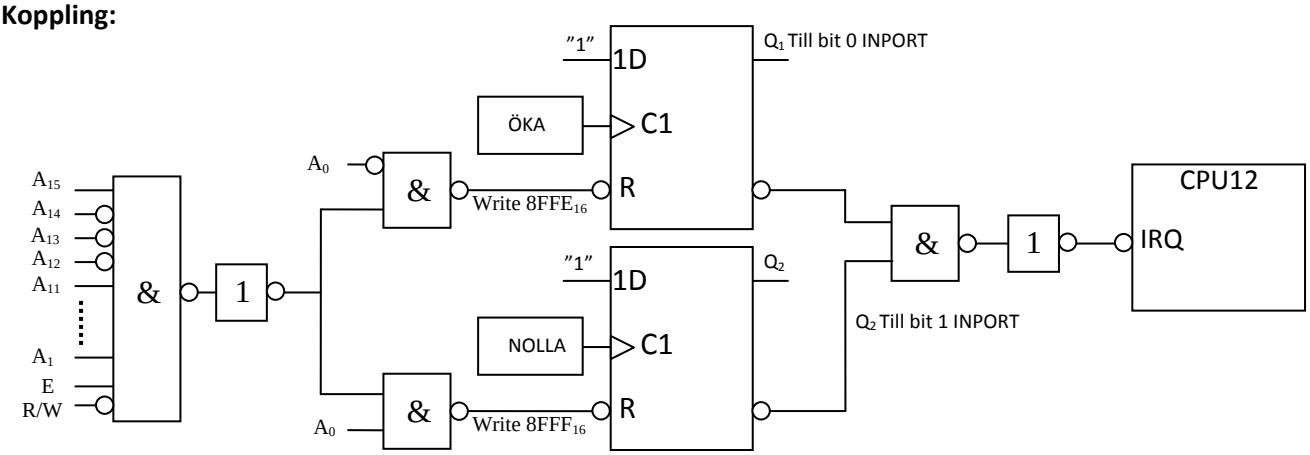
Övningsuppgifter

2013

Lösningsförslag
Uppgift 1-19

(Reservation för diverse fel!)

1. Koppling:



Program:

Avbrottsrutin (Interrupt handler)

```
IRQR    LDAA    INPORT          (INPORT = $900)
        BITA    #%00000001      Bit 0 är ansluten till ÖKA-vippan
        BNE     IRQINC
        BITA    #%00000010      Bit 1 är ansluten till NOLLA-vippan
        BNE     IRQCLR
IRQEX   RTI
```

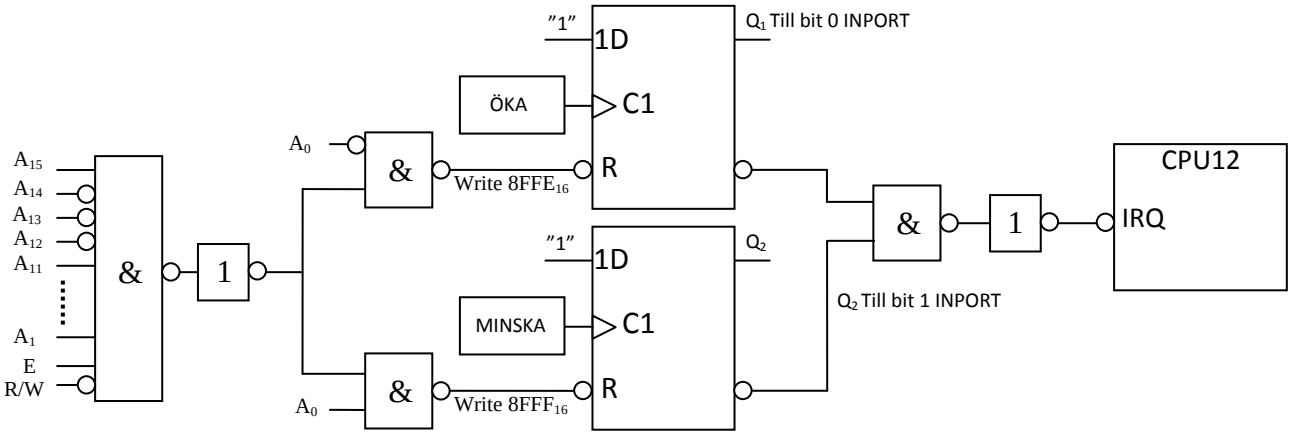
* **Avbrottsservicerutin för öka**

```
IRQINC  STAA    $8FFE           Nollställ avbrottsvippa nr 1
        LDAA    KNAPP
        CMPA    #255            Maxvärde?
        BEQ     IRQEX           Ja, skippa ökning
        INC     KNAPP
        BRA     IRQEX
```

* **Avbrottsservicerutin för nollställ**

```
IRQCLR  STAA    $8FFF           Nollställ avbrottsvippa nr 2
        CLR     KNAPP           Nollställ KNAPP
        BRA     IRQEX
```

2. Koppling:



Program:

Avbrottsrutin (Interrupt handler)

```
IRQR    LDAA    INPORT          (INPORT = $900)
        BITA    #%00000001      Bit 0 är ansluten till öka-vippan
        BNE     IRQINC
        BITA    #%00000010      Bit 1 är ansluten till minska-vippan
        BNE     IRQDEC
IRQEX   RTI
```

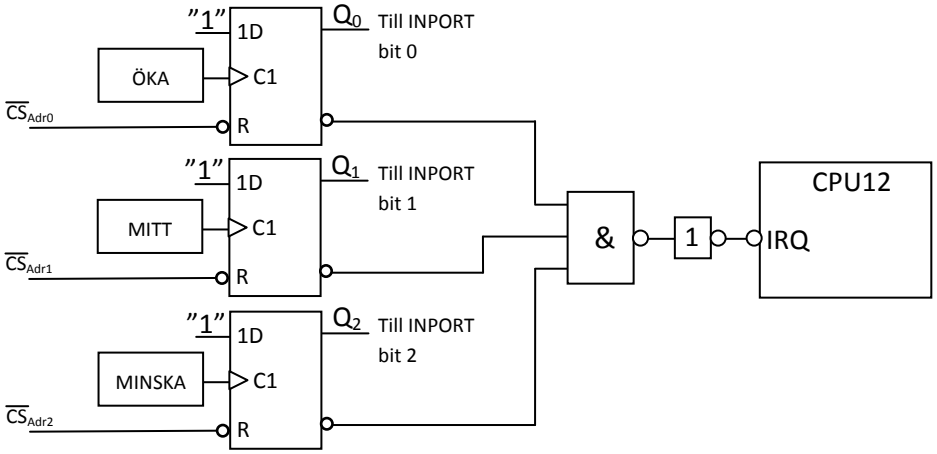
* **Avbrottsservicerutin för öka**

```
IRQINC  STAA    $8FFE           Nollställ avbrottsvippa nr 1
        LDAA    KNAPP
        CMPA    #255            Maxvärde?
        BEQ     IRQEX           Ja, skippa ökning
        INC     KNAPP
        BRA     IRQEX
```

* **Avbrottsservicerutin för minska**

```
IRQDEC  STAA    $8FFF           Nollställ avbrottsvippa nr 2
        TST     KNAPP           KNAPP = 0?
        BEQ     IRQEX           Ja, skippa minskning
        DEC     KNAPP
        BRA     IRQEX
```

3. Koppling:



Program:

Avbrottsrutin (Interrupt handler)

```
IRQR    LDAA    INPORT          (INPORT = $900)
        BITA    #%00000001      Bit 0 är ansluten till ÖKA-vippan
        BNE     IRQINC
        BITA    #%00000010      Bit 1 är ansluten till MITT-vippan
        BNE     IRQMID
        BITA    #%00000100      Bit 2 är ansluten till MINSKA-vippan
        BNE     IRQDEC
IRQEX   RTI
```

* **Avbrottsservicerutin för öka**

```
IRQINC  STAA    Adr0            Nollställ avbrottsvippa nr 0
        LDAA    KNAPP
        CMPA    #255            Maxvärde?
        BEQ     IRQEX           Ja, skippa ökning
        INC     KNAPP
        BRA     IRQEX
```

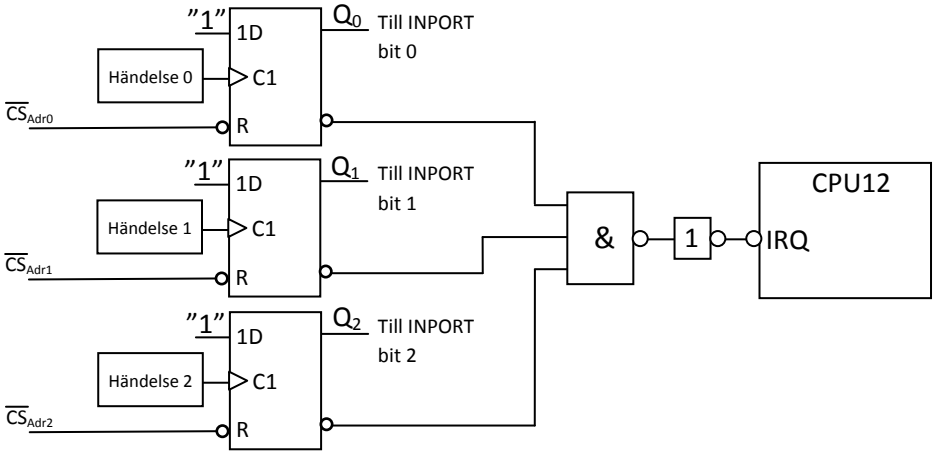
* **Avbrottsservicerutin för mittvärde**

```
IRQMID  STAA    Adr1            Nollställ avbrottsvippa nr 1
        MOVB    #128,KNAPP      Sätt KNAPP till mittvärde
        BEQ     IRQEX
```

* **Avbrottsservicerutin för minska**

```
IRQDEC  STAA    Adr2            Nollställ avbrottsvippa nr 2
        TST     KNAPP           KNAPP = 0?
        BEQ     IRQEX           Ja, skippa minskning
        DEC     KNAPP
        BRA     IRQEX
```

4. Koppling:



Program:

Avbrottsrutin (Interrupt handler)

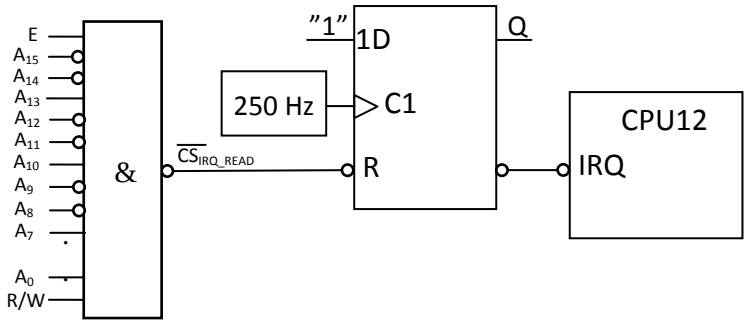
```
IRQR    LDAA    INPORT      (INPORT = $E0D0)
        BITA    #%00000001  Bit 0 är ansluten till Händelse 0-vippan
        BNE     IRQEV0
        BITA    #%00000010  Bit 1 är ansluten till Händelse 1-vippan
        BNE     IRQEV1
        BITA    #%00000100  Bit 2 är ansluten till Händelse 2-vippan
        BNE     IRQEV2
IRQEX   RTI
*               Avbrottsservicerutin för Händelse 0
IRQEV0  STAA    Adr0        Nollställ avbrottsvippa nr 0
        INC     EVCNT0      8-bitars räknare
        BRA     IRQEX
*               Avbrottsservicerutin för Händelse 1
IRQEV1  STAA    Adr1        Nollställ avbrottsvippa nr 1
        LDX     EVCNT1      16-bitars räknare
        INX
        STX     EVCNT1
        BEQ     IRQEX
*               Avbrottsservicerutin för Händelse 2
IRQEV2  STAA    Adr2        Nollställ avbrottsvippa nr 2
        LDX     EVCNT2+2    32-bitars räknare, låg del
        INX
        STX     EVCNT2+2
        BNE     IRQEX       Carry till hög del = 0

        LDX     EVCNT2      Carry till hög del = 1
        INX
        STX     EVCNT2
        BRA     IRQEX
```

Subrutin för initiering av variabler och avbrottssystem

```
INIRQ   CLR     Adr0        Nollställ avbrottsvippor
        CLR     Adr1
        CLR     Adr2

        CLR     EVCNT0      Nollställ variabler
        MOVW    #0,EVCNT1
        MOVW    #0,EVCNT2
        MOVW    #0,EVCNT2+2

        MOVW    #IRQR,$FFF2 Avbrottsvektor
        CLI                 Aktivera avbrottssystem
        RTS
```

EVCNT0 ($DC00_{16}$)	
EVCNT1	ms
	ls
EVCNT2	ms
EVCNT2+2	
	ls

5. Koppling:



Program:

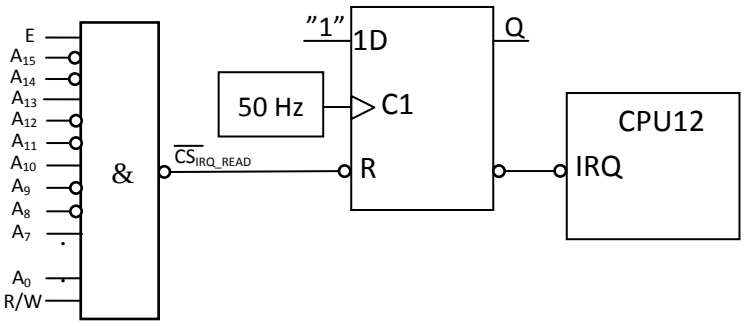
Avbrottsrutin

```
IRQR    TST     $24FF       Nollställ avbrottsvippan (T ex adr $24FF enligt figuren)
        DEC     IRQCNT      Avbrottsräknare
        BNE     IRQEX       Ej 1s, hoppa ut

        MOVB    #250,IRQCNT Ominitiera avbrottsräknare
        MOVB    SENSE,$DE00 Läs givaren och uppdatera värdet

IRQEX   RTI

IRQCNT  RMB     1
```

I**nitiering av variabler och avbrottssystem** (Stackpekaren antas initierad tidigare)

```
        TST     $24FF       Nollställ avbrottsvippan (T ex adr $24FF enligt figuren)
        MOVB    #250,IRQCNT Initiera avbrottsräknare för 1s
        MOVW    #IRQR,$FFF2 Avbrottsvektor
        CLI                 Aktivera avbrottssystem
        …
```

6. Koppling:



Program:

Avbrottsrutin

IRQR	TST	$24FF	Nollställ avbrottsvippan (T ex adr $24FF enligt figuren)
	DEC	IRQCNT	Avbrottsräknare
	BNE	IRQEX	Ej 1s, hoppa ut
	MOVB	#50,IRQCNT	Ominitiera avbrottsräknare
	MOVB	$600,$2800	Läs givaren och uppdatera värdet
IRQEX	RTI		
IRQCNT	RMB	1	

I**nitiering av variabler och avbrottssystem** (Stackpekaren antas initierad tidigare)

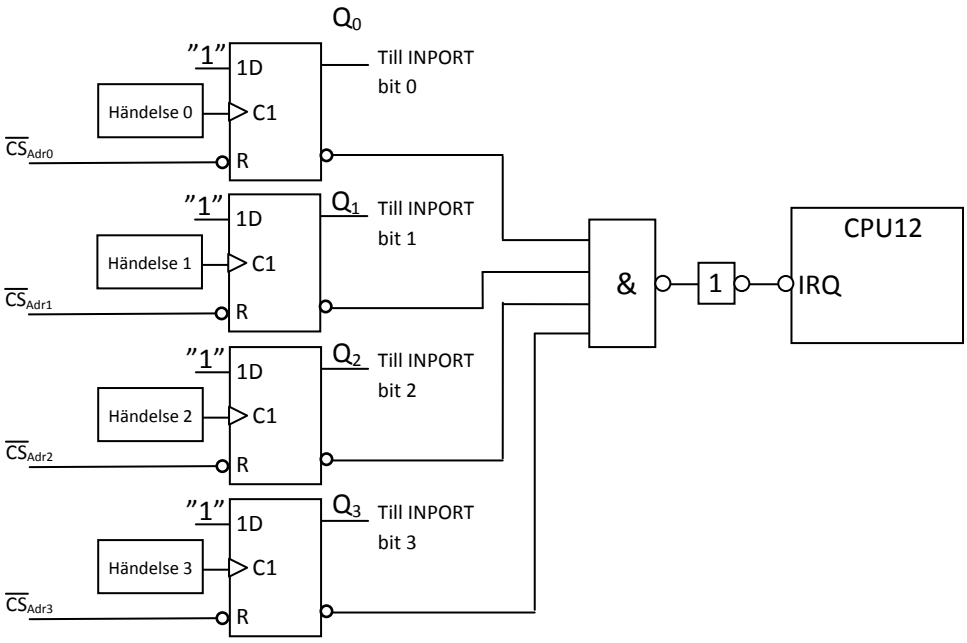
TST	$24FF	Nollställ avbrottsvippan (T ex adr $24FF enligt figuren)
MOVB	#50,IRQCNT	Initiera avbrottsräknare för 1s
MOVW	#IRQR,$FFF2	Avbrottsvektor
CLI		Aktivera avbrottssystem
…		

7. Antag att varje händelse genererar en puls.

Huvudprogrammet: (Stackpekaren antas vara initierad tidigare.)
Huvudprogrammet måste initiera avbrottssystemet, dvs. se till att hopp görs till avbrottsrutinens adress vid avbrottsförfrågan (IRQ) genom att skriva in adressen till avbrottsrutinen IRQ-vektorn. Eventuella variabler som kan behövas av servicerutinerna för de fyra olika avbrottskällorna måste också initieras. Avbrottsvipporna skall nollställas för att falska avbrott inte skall uppträda vid start.Som sista åtgärd skall avbrottsystemet aktiveras genom att I-flaggan i CC-registret nollställs.

Avbrottsrutinen:
Avbrottsrutinen måste först identifiera avbrottskällan genom att läsa av vilket Q_i-värde enligt figuren nedan som är aktivt och sedan hoppa till aktuell servicerutin. I servicerutinen skall de uppgifter utföras som är förknippade med det aktuella avbrottet. Före återhopp till avbrottsrutinen och senare återhopp till det avbrutna programmet med RTI måste också avbrottsvippan nollställas med en puls på vippans RESET-ingång genom att en läsning eller skrivning görs på aktuell adress enligt figuren nedan.

Koppling:



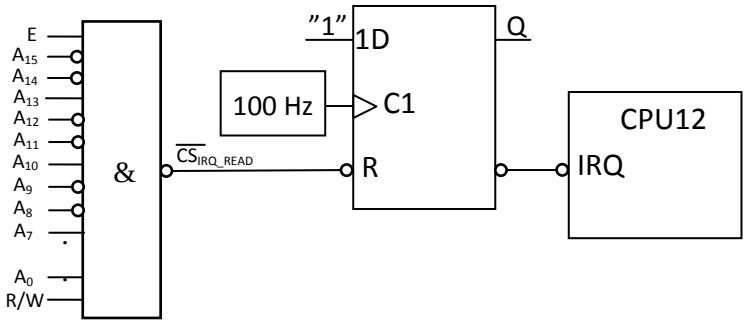
8. Antag att varje händelse genererar en puls.

Huvudprogrammet: (Stackpekaren antas vara initierad tidigare.)
Huvudprogrammet måste initiera avbrottssystemet för både XIRQ- och IRQ-avbrott, dvs. se till att hopp görs till respektive avbrottsrutins adress vid avbrottsförfrågan (XIRQ eller IRQ) genom att skriva in adressen till XIRQ-rutinen i XIRQ-vektorn ($FFF4) och adressen till IRQ-rutinen i IRQ-vektorn ($FFF2). Eventuella variabler som kan behövas av servicerutinerna för de olika avbrottskällorna måste också initieras. Avbrottsvipporna skall nollställas för att falska avbrott inte skall uppträda vid start. Som sista åtgärd skall avbrottsystemet aktiveras genom att X- och I-flaggan i CC-registret nollställs.

Avbrottsrutinen (IRQR):
Eftersom det bara finns en avbrottskälla för IRQ kan man i avbrottsrutinen IRQR direkt utföra de uppgifter som är förknippade med det aktuella avbrottet, inklusive eventuella ominitieringar av variabler som används av avbrottsrutinen. Före återhopp till det avbrutna programmet från IRQR-rutinen med RTI måste IRQ-avbrottsvippan nollställas med en CS'-puls på vippans RESET-ingång genom att en läsning eller skrivning görs på den adress man har valt för detta ändamål. I detta fall kan man t ex bilda CS'-signalen genom att avkoda läsning av IRQ-vektorn och då sker nollställningen av avbrottsvippan automatiskt då IRQ-avbrottet accepteras.

Om en begäran om XIRQ-avbrott kommer medan IRQ-avbrottet behandlas så avbryts IRQR och XIRQ-rutinen tar över. Återhopp görs sedan till IRQR med RTI i XIRQ-rutinen.

9. Koppling:



Program:

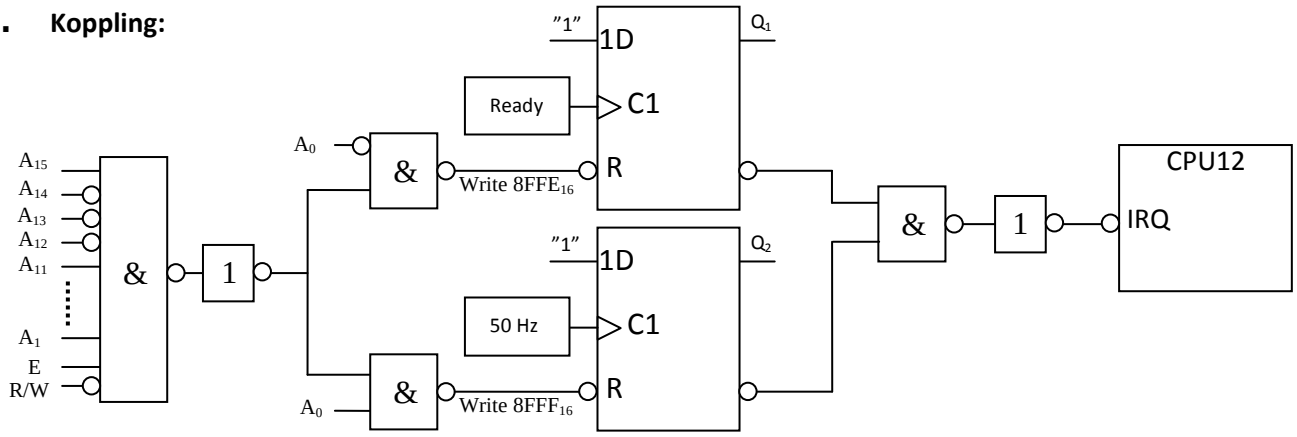
Avbrottsrutin

```
IRQR    TST     $24FF       Nollställ avbrottsvippan (T ex adr $24FF enligt figuren)
        LDX     IRQCNT      Hämta avbrottsräknare
        DEX
        STX     IRQCNT      Uppdatera avbrottsräknare
        BNE     IRQEX       Ej 1 min, hoppa ut
        MOVW    #6000,IRQCNT Ominitiera avbrottsräknare
        CLR     ADSTRT
        MOVB    #$80,ADSTRT Positiv flank på bit 7 startar AD-omvandling
IRLOOP  TST     ADSTAT      Bit 7 = 1?
        BPL     IRLOOP      Nej, vänta
        MOVB    PRESS,CURPR Ja. Läs givaren och uppdatera värdet
IRQEX   RTI
IRQCNT  RMB     2
```

I**nitiering av variabler och avbrottssystem** (Stackpekaren antas initierad tidigare)

```
        TST     $24FF       Nollställ avbrottsvippan (T ex adr $24FF enligt figuren)
        MOVW    #6000,IRQCNT Initiera avbrottsräknare för 1 min
        MOVW    #IRQR,$FFF2 Avbrottsvektor
        CLI                 Aktivera avbrottssystem
        …
```

10. **Koppling:**



Program:

Avbrottsrutin (Interrupt handler)

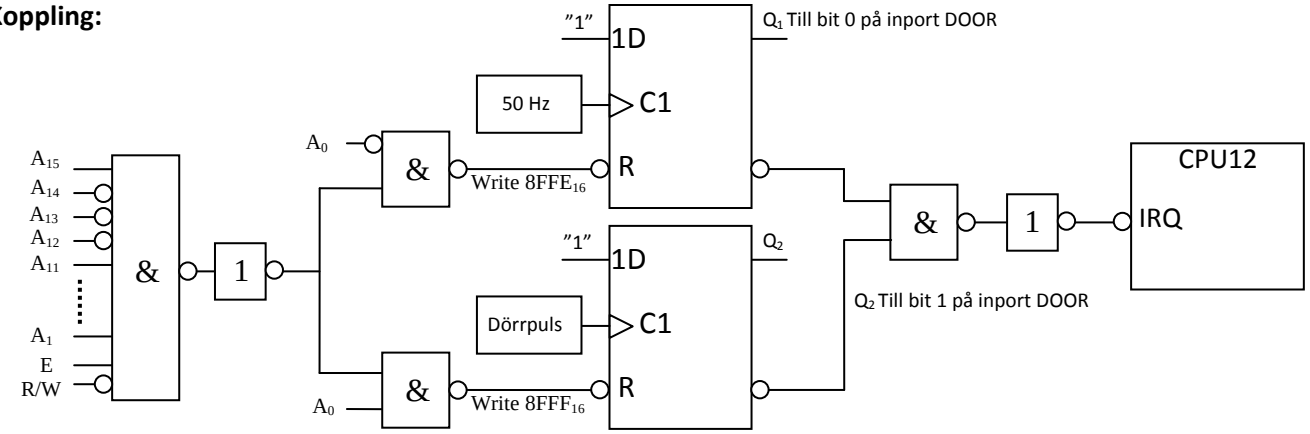
```
IRQR    LDAA    ADSTAT          Läs Ready-signal på bit 0
        BITA    #%00000001      Ready = 1?
        BNE     IRREAD          Ja, A/D-omvandling färdig
        CLR     $8FFF           Nej, alltså 50 Hz avbrott. Nollställ avbrottsvippa 2
        LDX     IRQCNT          50 Hz-avbrott, hämta avbrottsräknare
        DEX
        STX     IRQCNT          Uppdatera avbrottsräknare
        BNE     IRQEX           Ej 10 min, hoppa ut
        MOVW    #30000,IRQCNT   Ominitiera avbrottsräknare för 10 min och starta ADC
        MOVB    #1,ADCTRL       Negativ flank på bit 0 startar AD-omvandling
        CLR     ADCTRL          som ger nytt avbrott om ca. 100 mikrosek
IRQEX   RTI

IRREAD  CLR     $8FFE           Nollställ avbrottsvippa 1
        MOVB    LEVEL,NEWLEW    Läs nivå och uppdatera. Nollställ också Ready
        BRA     IRQEX
IRQCNT  RMB     2
```

I**nitiering av variabler och avbrottssystem** (Stackpekaren antas initierad tidigare)

```
        CLR     $8FFE           Nollställ avbrottsvipporna
        CLR     $8FFF

        MOVW    #30000,IRQCNT   Initiera avbrottsräknare för 10 min
        MOVW    #IRQR,$FFF2     Avbrottsvektor
        CLI                     Aktivera avbrottssystem
        …
```

11.

Koppling:



Program:

Avbrottsrutin (Interrupt handler)

```
IRQR    LDAA    DOOR            Läs avbrottsvippor
        BITA    #%00000001      Bit 0 är ansluten till 50 Hz-vippan
        BNE     IRQ50HZ
        BITA    #%00000010      Bit 1 är ansluten till Dörr-vippan
        BNE     IRQDR
IRQEX   RTI
```

* **Avbrottsservicerutin 50 Hz**

```
IRQ50HZ STAA    $8FFE           Nollställ 50Hz-vippan
        LDX     IRQCNT          50 Hz-avbrott, hämta avbrottsräknare
        DEX
        STX     IRQCNT          Uppdatera avbrottsräknare
        BNE     IRQEX           Ej 1 min, hoppa ut
        MOVW    #3000,IRQCNT    Ominitiera avbrottsräknare för 1 min
        MOVB    AIR,FLOW        Läs tryck och uppdatera
        BRA     IRQEX
```

* **Avbrottsservicerutin för DOOR**

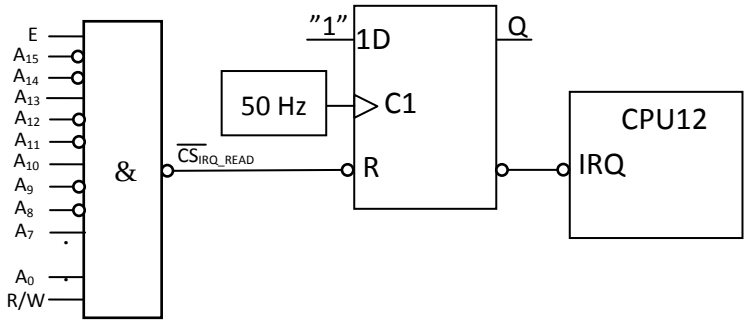
```
IRQDR   STAA    $8FFF           Nollställ DOOR-vippan
        MOVB    DOOR,POS        Läs dörrposition och uppdatera
        BRA     IRQEX
```

I**nitiering av variabler och avbrottssystem** (Stackpekaren antas initierad tidigare)

```
        CLR     $8FFE           Nollställ avbrottsvipporna
        CLR     $8FFF

        MOVW    #3000,IRQCNT    Initiera avbrottsräknare för 1 min
        MOVW    #IRQR,$FFF2     Avbrottsvektor
        CLI                     Aktivera avbrottssystem
        …

        ORG     $1FF0
IRQCNT  RMB     2
```

12.

Koppling:



Program:

Avbrottsrutin

```
IRQR    TST     $24FF       Nollställ avbrottsvippan (T ex adr $24FF enligt figuren)
        DEC     ICNT1       Avbrottsräknare för tiondels sekund
        BNE     CHK1S       Ej tiondels sekund, kolla om 1 sekund har gått

        MOVB    #5,ICNT1    Ominitiera avbrottsräknare för tiondels sekund
        MOVB    PORT1,GIV1  Läs givaren och uppdatera värdet

CHK1S   DEC     ICNT2       Avbrottsräknare för 1 sekund
        BNE     IRQEX       Ej 1 sekund, hoppa ut

        MOVB    #50,ICNT2   Ominitiera avbrottsräknare för 1 sekund
        MOVB    PORT2,GIV2  Läs givaren och uppdatera värdet

IRQEX   RTI
```

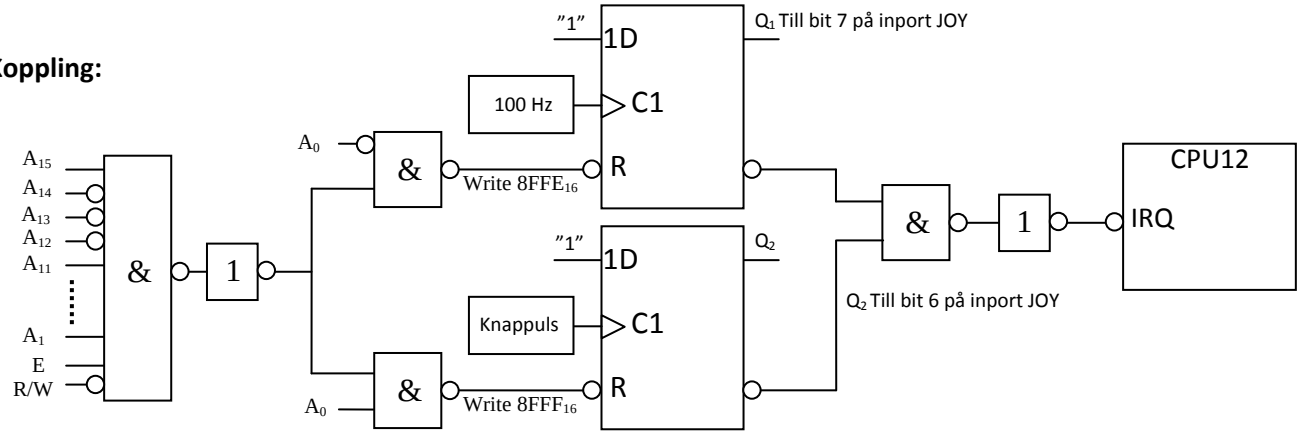
I**nitiering av variabler och avbrottssystem** (Stackpekaren antas initierad tidigare)

```
        TST     $24FF       Nollställ avbrottsvippan (T ex adr $24FF enligt figuren)
        MOVB    #5,ICNT1    Initiera avbrottsräknare för tiondels sekund
        MOVB    #50,ICNT2   Initiera avbrottsräknare för 1s

        MOVW    #IRQR,$FFF2 Avbrottsvektor
        CLI                 Aktivera avbrottssystem
        …
```

```
        ORG     $1FF0
ICNT1   RMB     1           Räknare för tiondels sekund
ICNT2   RMB     1           Räknare för en sekund
```

13.

Koppling:



Program:

Avbrottsrutin (Interrupt handler)

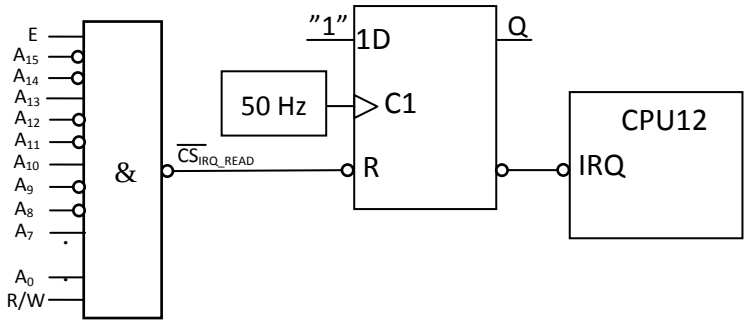
```
IRQR    LDAA    JOY             Läs avbrottsvippor
        BITA    #%10000000      Bit 7 är ansluten till 100 Hz-vippan
        BNE     INT100
        BITA    #%01000000      Bit 6 är ansluten till Knapp-vippan
        BNE     IRKNP
IRQEX   RTI

*       Avbrottsservicerutin 100 Hz

INT100  STAA    $8FFE           Nollställ 100Hz-vippan
        DEC     IRQCNT          100 Hz-avbrott, hämta avbrottsräknare
        BNE     IRQEX           Ej 20-dels sekund, hoppa ut
        MOVB    #5,IRQCNT       Ominitiera avbrottsräknare för 20-dels sekund
        MOVB    JOY,JSTICK      Läs av joysticksläge och uppdatera
        BRA     IRQEX

*       Avbrottsservicerutin för tryckknapp

IRKNP   STAA    $8FFF           Nollställ Knapp-vippan
        INC     KNAPP           Uppdatera antal knapptryckningar
        BRA     IRQEX
```

Initiering av variabler och avbrottssystem (Stackpekaren antas initierad tidigare)

```
        CLR     $8FFE           Nollställ avbrottsvipporna
        CLR     $8FFF

        MOVB    #5,IRQCNT       Initiera avbrottsräknare för 20-dels sekund
        CLR     KNAPP           Nollställ antal knapptryckningar
        MOVW    #IRQR,$FFF2     Avbrottsvektor
        CLI                     Aktivera avbrottssystem
        …

        ORG     $1FF0
IRQCNT  RMB     1
```

14.

Koppling:



Program:

Avbrottsrutin

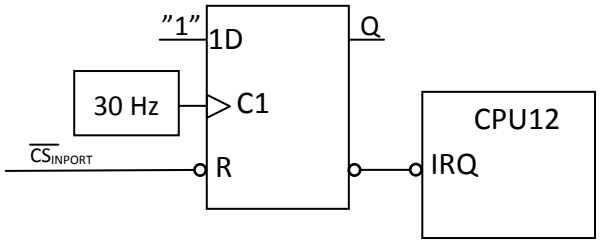
```
IRQR    TST     $24FF       Nollställ avbrottsvippan (T ex adr $24FF enligt figuren)
        DEC     IRQCNT      Avbrottsräknare
        BNE     IRQEX       Ej 1s, hoppa ut

        MOVB    #50,IRQCNT  Ominitiera avbrottsräknare
        MOVB    SENSE,$DE00 Läs givaren och uppdatera värdet

IRQEX   RTI

IRQCNT  RMB     1
```

I**nitiering av variabler och avbrottssystem** (Stackpekaren antas initierad tidigare)

```
        TST     $24FF       Nollställ avbrottsvippan (T ex adr $24FF enligt figuren)
        MOVB    #50,IRQCNT  Initiera avbrottsräknare för 1s
        MOVW    #IRQR,$FFF2 Avbrottsvektor
        CLI                 Aktivera avbrottssystem
        …
```

15.

Koppling:



Program:

Avbrottsrutin

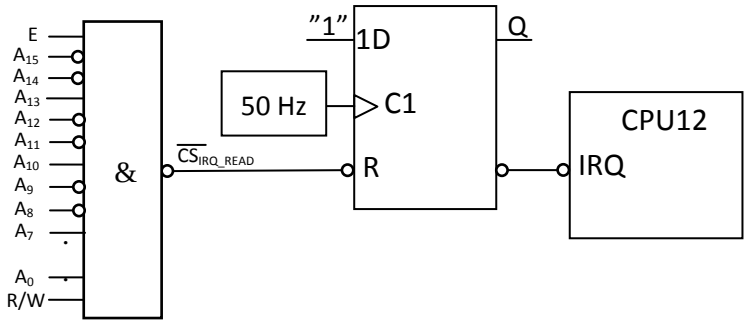
```
IRQR    MOVB    INPORT,INVAR    Uppdatera INVAR och nollställ avbrottsvippan
        DEC     COUNT           Avbrottsräknare för tiondels sekund
        BNE     IRQEX           Ej tiondels sekund. Hoppa ut
        MOVB    #3,COUNT        Ominitiera avbrottsräknare för tiondels sekund
        JSR     OUTPUT          Sköter utmatning till utport
IRQEX   RTI
```

I**nitiering av variabler och avbrottssystem** (Stackpekaren antas initierad tidigare)

```
        TST     INPORT          Nollställ avbrottsvippan
        MOVB    #3,COUNT        Initiera avbrottsräknare för tiondels sekund
        MOVW    #IRQR,$FFF2     Avbrottsvektor
        CLI                     Aktivera avbrottssystem
        …
```

16.

Koppling:



Program:

Avbrottsrutin

```
IRQR    TST     $24FF       Nollställ avbrottsvippan (T ex adr $24FF enligt figuren)
        DEC     COUNT       Avbrottsräknare
        BNE     IRQEX       Ej 1s, hoppa ut

        MOVB    #50,COUNT   Ominitiera avbrottsräknare

        LDAA    CLOCK+2     Hämta sekundvärdet
        ADDA    #1          En sekund till
        DAA                 Decimaljustera
        STAA    CLOCK +2    Uppdatera sekundvärdet
        CMPA    #$60        Hel minut?
        BNE     SHOW        Nej, visa tid
        CLR     CLOCK +2    Ja, nollställ sekundvärdet

        LDAA    CLOCK +1    Hämta minutvärdet
        ADDA    #1          En minut till
        DAA                 Decimaljustera
        STAA    CLOCK +1    Uppdatera minutvärdet
        CMPA    #$60        Hel timma?
        BNE     SHOW        Nej, visa tid
        CLR     CLOCK +1    Ja, nollställ minutvärdet

        LDAA    CLOCK       Hämta timvärdet
        ADDA    #1          En timma till
        DAA                 Decimaljustera
        STAA    CLOCK       Uppdatera timvärdet
        CMPA    #$24        Helt dygn?
        BNE     SHOW        Nej, visa tid
        CLR     CLOCK       Ja, nollställ timvärdet
SHOW    MOVB    CLOCK,TIM   Visa timvärdet
        MOVB    CLOCK+1,MIN Visa minutvärdet
        MOVB    CLOCK+2,SEK Visa sekundvärdet
IRQEX   RTI
```

I**nitiering av variabler och avbrottssystem** (Stackpekaren antas initierad tidigare)

```
        TST     $24FF       Nollställ avbrottsvippan (T ex adr $24FF enligt figuren)
        MOVB    #50,COUNT   Initiera avbrottsräknare för 1s

        CLR     CLOCK       Nollställ timvärdet
        CLR     CLOCK +1    Nollställ minutvärdet
        CLR     CLOCK +2    Nollställ sekundvärdet

        MOVW    #IRQR,$FFF2 Avbrottsvektor
        CLI                 Aktivera avbrottssystem
        …
```

17.

```
UTPORT  EQU     $3000           Utport för ASCII-tecken till skrivare
```

Subrutin för initiering av avbrottsstyrd utmatning av en textsträng

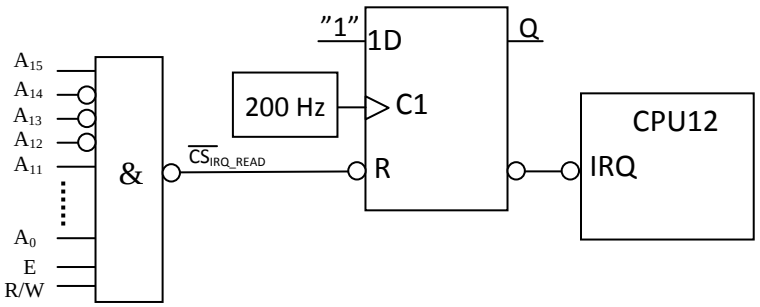
```
INISTR  CLR     UTPORT          Nollställ avbrottsvippan
        STX     STRPNT          Initiera global pekare till textsträng
        MOVW    #PRINTR,$FFF2   Avbrottsvektor
        CLI                     Aktivera avbrottssystem
        RTS
```

Avbrottsrutin för utmatning av textsträng till skrivare

```
PRINTR  LDX     STRPNT          Hämta pekare till textsträng
        LDAA    1.X+            Hämta tecken från textsträng
        STAA    UTPORT          Mata ut ASCII-tecken och nollställ avbrottsvippa
        BNE     PRIEX           Uthopp om ej strängslut

        MOVB    #$FF,RFLAG      Signalera klart till huvudprogram
        PULA                    Hämta CC-värde från stack
        ORAA    #$10            Sätt I = 1. (Stäng av avbrott)
        PSHA                    Skriv tillbaka CC-värde till stack

PRIEX   STX     STRPNT          Uppdatera pekare till textsträng
        RTI
```

18.

Koppling:

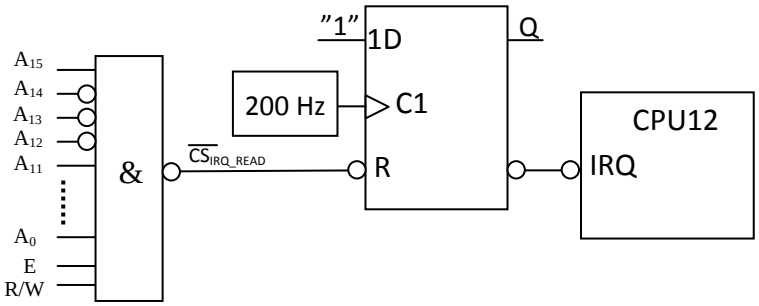


Program:

```
BOS0    EQU     $1FF0           BOS för program P0
BOS1    EQU     $2FF0           BOS för program P1
START0  EQU     $1000           Startadress  för program P0
START1  EQU     $2000           Startadress  för program P1
IRQRES  EQU     $8FFF           Läsadress för nollställning av IRQ-vippa
IRQVEC  EQU     $FFF2           Avbrottsvektorn
GLOBAL  EQU     $1FF0           (Denna krockar inte med BOS0 ovan!)

        ORG     START0
P0      MOVB    #$C0,BOS1-9     CCR till stack1
        MOVW    #START1,BOS1-2  Startadress för program P1 till stack1
        MOVW    #BOS1-9,SPSAVE  Stackpekarvärde efter avbrott till stack1

        LDS     #BOS0           Initiera stack för program 0
        TST     IRQRES          Nollställ avbrottsvippan
        MOVW    #IRQR,IRQVEC    Initiera IRQ-vektorn
        CLI                     Aktivera avbrottssystemet

P0LOOP  NOP                     Evighetsslinga i P0
        BRA     P0LOOP

        ORG     START1
P1      NOP                     Evighetsslinga i P1
        BRA     P1


IRQR    TST     IRQRES          Nollställ avbrottsvippan
        LDX     SPSAVE          Hämta "den andra" stackpekaren
        STS     SPSAVE          Spara den aktiva stackpekaren
        TFR     X,SP            Byt stackpekare
        RTI                     Återvänd till det andra programmet


        ORG     GLOBAL
SPSAVE  RMB     2               Plats för den passiva (andra) stackpekaren
```

19.

Koppling:



Program:

```
BOS0    EQU     $8000           BOS för program P0
BOS1    EQU     $C000           BOS för program P1
START   EQU     $1000           Startadress  för program PINIT (t ex)
START0  EQU     $4000           Startadress  för program P0
START1  EQU     $8000           Startadress  för program P1
IRQRES  EQU     $8FFF           Läsadress för nollställning av IRQ-vippa
IRQVEC  EQU     $FFF2           Avbrottsvektorn
GLOBAL  EQU     $2000

        ORG     START
PINIT   MOVB    #$C0,BOS1-9     CCR till stack1
        MOVW    #START1,BOS1-2  Startadress för program P1 till stack1
        MOVW    #BOS1-9,SPSAVE  Stackpekarvärde efter avbrott till stack1

        LDS     #BOS0           Initiera stack för program 0
        TST     IRQRES          Nollställ avbrottsvippan
        MOVB    #0,TOGGLE       För att välja varannat avbrott
        MOVW    #IRQR,IRQVEC    Initiera IRQ-vektorn
        CLI                     Aktivera avbrottssystemet
        LBRA    P0

        ORG     START0
P0      NOP                     Evighetsslinga i P0
        BRA     P0

        ORG     START1
P1      NOP                     Evighetsslinga i P1
        BRA     P1


IRQR    TST     IRQRES          Nollställ avbrottsvippan
        COM     TOGGLE          Endast processbyte vartannat avbrott
        BEQ     IRQEX           Ej processbyte

        LDX     SPSAVE          Processbyte, hämta "den andra" stackpekaren
        STS     SPSAVE          Spara den aktiva stackpekaren
        TFR     X,SP            Byt stackpekare
IRQEX   RTI                     Återvänd till det andra programmet


        ORG     GLOBAL
TOGGLE  RMB     1
SPSAVE  RMB     2               Plats för den passiva (andra) stackpekaren
```