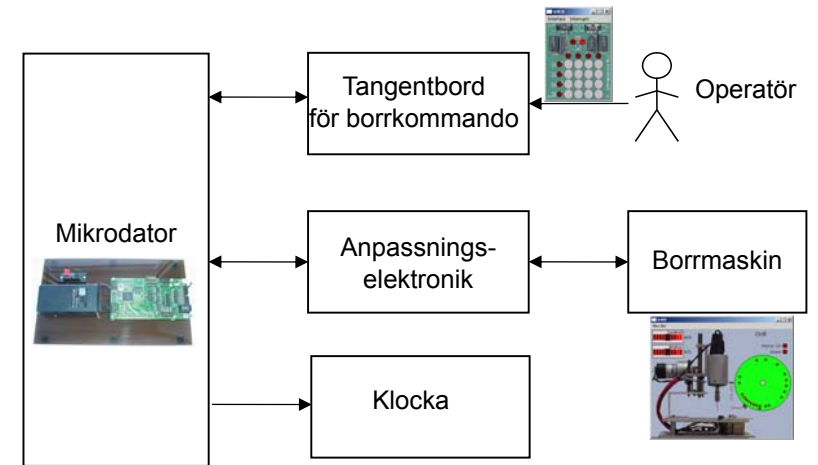


Maskinorienterad Programmering 2012/2013

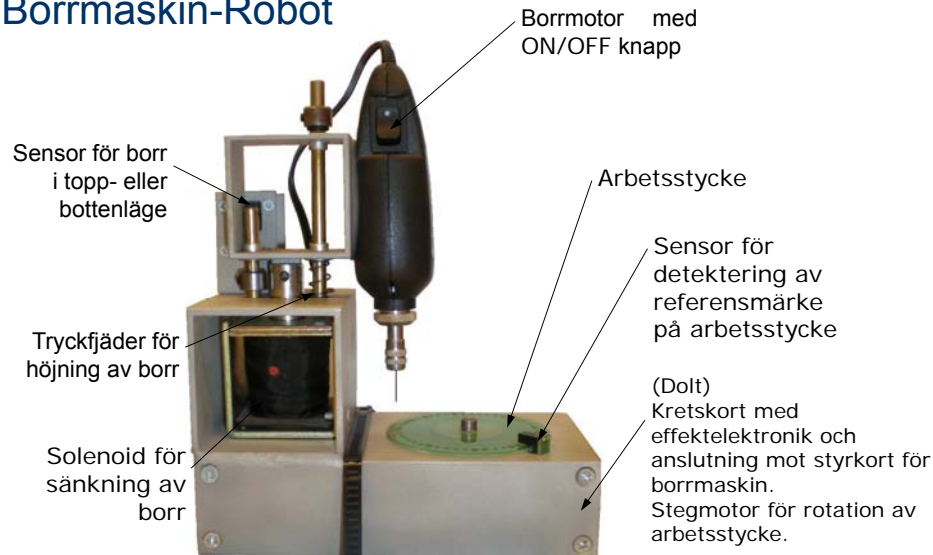
Genomgång av laborationer:
"Programutveckling i assembler"

Arbetsbok för MC12, kapitel 4

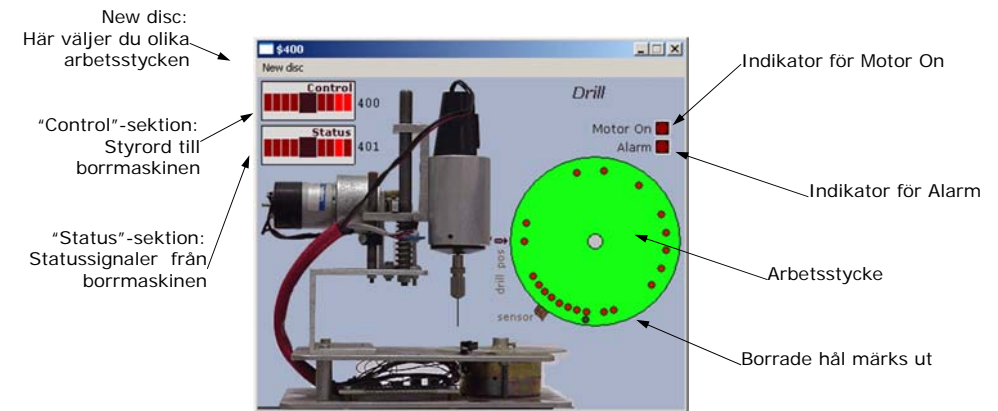
Laborationsmoment 2 - En Borrautomat



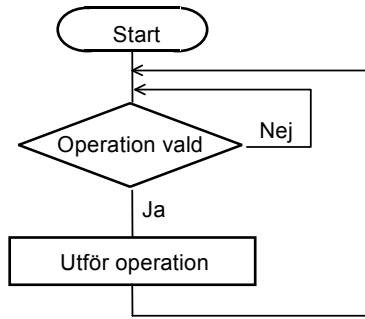
Borrmaskin-Robot



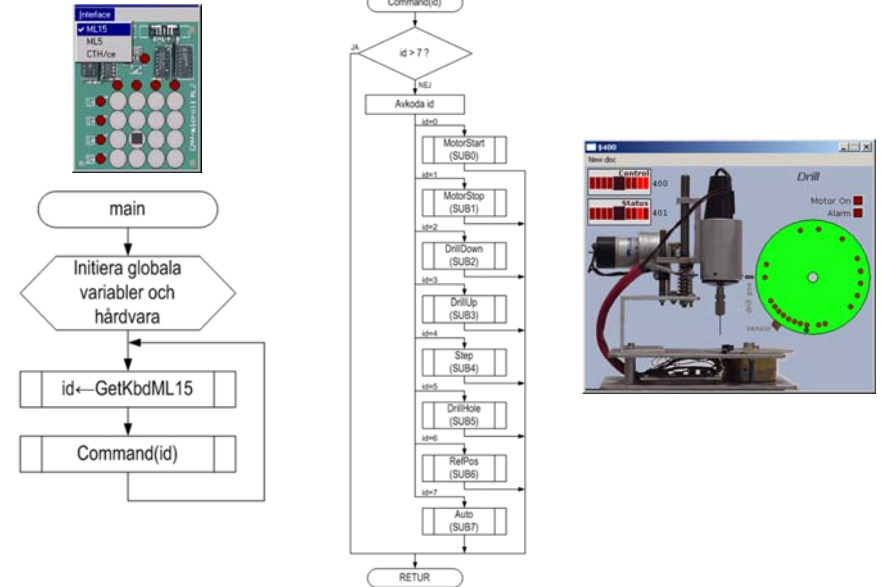
Simulatoren för borrmaskinen



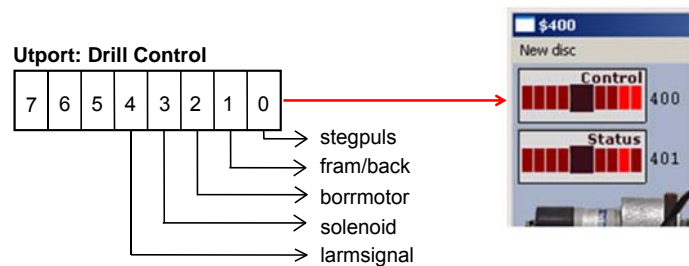
Specifikation



- starta bormotorn
- stoppa bormotorn
- sänk borret
- höj borret
- vrid arbetsstycket ett steg
- vrid arbetsstycket till referenspositionen
- borra ett hål
- borra hål längs cirkeln enligt ett bestämt mönster.



Styrdord till bormaskinen



- Bit 4 = 1: Larm på
- Bit 3 = 1: Borret sänks
- Bit 2 = 1: Bormotorn roterar
- Bit 1 = 1: Medurs vridning
- Bit 0: Pos flank Stegpuls

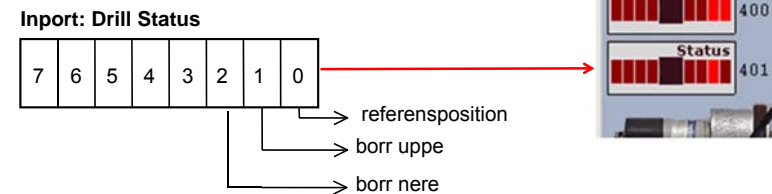
Logiknivå: "Aktiv hög"

Att göra "RESET" på bormaskinen således:

```

LDAA #0           ; Passiva signaler
STAA $400
---
```

Statusord från bormaskinen

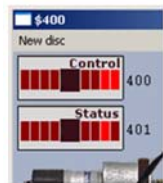


- Bit 2 = 1: Borr i bottenläge
- Bit 1 = 1: Borr i toppläge
- Bit 0 = 1: Referensposition

Logiknivå: "Aktiv hög"

Anm:
Statusporten
ansluts till adress
\$600 i
laborations-
systemet

Testförfarande



Uppgift 71

```
DipSwitch EQU $600
HexDisplay EQU $700
DrillControl EQU $0400
DrillStatus EQU $0401
```

```
Loop LDAA DipSwitch ; Läs strömbrytare
     STAA DrillControl ; Ge styrord
     LDAB DrillStatus ; Läs status
     BRA Loop
```

Villkorlig assemblering ger korrekta portadresser

```
; Definiera macro 'SIM' för test i simulator
#define SIM

DipSwitch EQU $600 ; Dip Switch Input
DrillControl EQU $0400 ; Drill Control Output
#ifdef SIM
DrillStatus EQU $0401 ; Drill Status Input
#else
DrillStatus EQU $0600 ; Drill Status Input
#endif
```

Anm: "Dip Switch Input" och bormaskin kan inte användas samtidigt i laborationssystemet (MC12).

Använd USE-direktivet

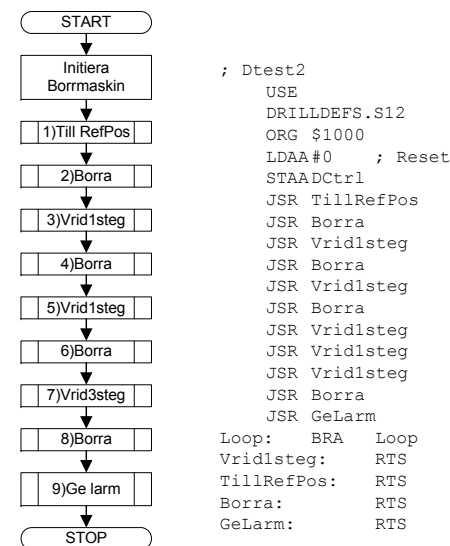
```
; Labdefs.S12
DipSwitch EQU $600 ; Dip Switch Input
DrillControl EQU $0400 ; Drill Control Output
#ifdef SIM
DrillStatus EQU $0401 ; Drill Status Input
#else
DrillStatus EQU $0600 ; Drill Status Input
#endif
```

```
; Drilltest1.s12
#define SIM
USE "DRILLDEFS.S12"

Loop LDAA DipSwitch ; Läs strömbrytare
     STAA DrillControl ; Ge styrord
     LDAB DrillStatus ; Läs status
     BRA Loop
```

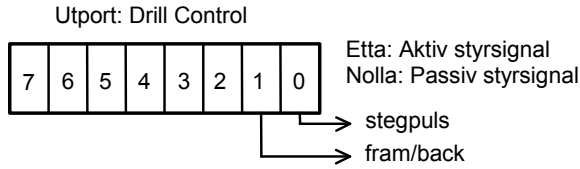
Inledande uppgift med bormaskinen

- 1) Arbetsstycket vrids till referensposition.
- 2) Hål borras
- 3) Arbetsstycket vrids *medurs* ett steg
- 4) Hål borras
- 5) Arbetsstycket vrids *medurs* ett steg
- 6) Hål borras
- 7) Arbetsstycket vrids *medurs* tre steg
- 8) Hål borras
- 9) En larmsignal ges som indikation på att uppgiften är klar.

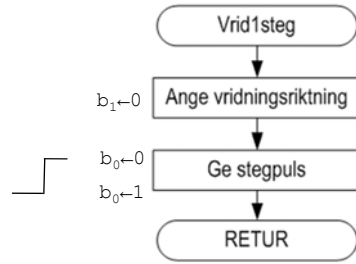


Att vrida arbetsstycket

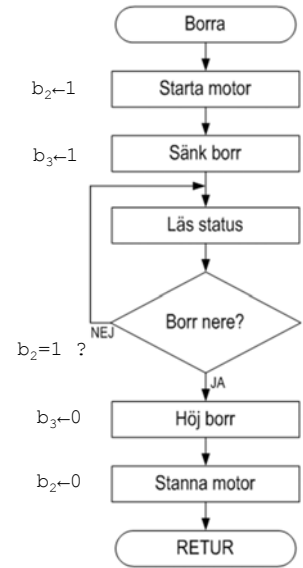
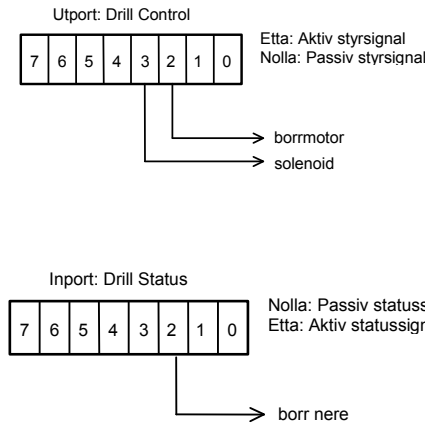
Uppgift 77



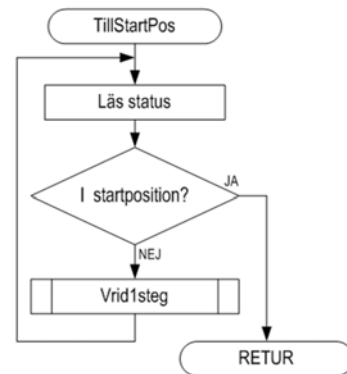
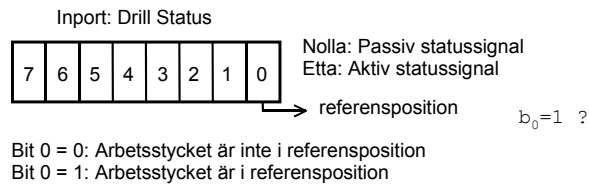
Bit 0 = 0 → 1 (Pos puls): Arbetsstycket vrids
Bit 1 = 0: Medurs vridningsriktning
Bit 1 = 1: Moturs vridningsriktning



Att borra ett hål



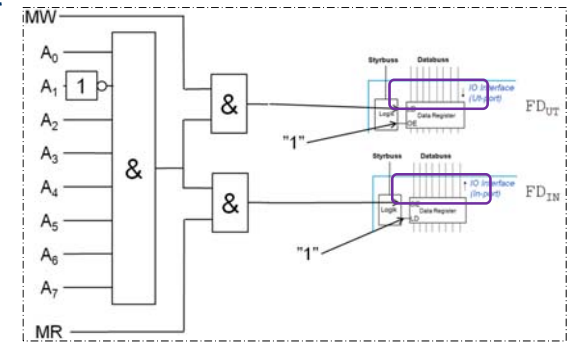
Att vrida arbetsstycket till referenspositionen



Att bara ändra en bit i taget

- * Läs nuvarande styrorrd
LDAA DrillControl
- * Nollställ lämplig bit
ANDA #xx
- * Skriv nytt styrorrd
STAA DrillControl

;sekvensen är funktionellt
;likvärdig med:
BCLR #~DrillControl



Fungerar inte här ty porten är "icke läsbar" utport...

Kopia av styrordet

Variabel DCSshadow ska hela tiden ha samma värde som DrillControl hade haft om porten varit läsbar...

För att nollställa en bit används nu:

```
LDAA    DCSshadow
ANDA    #~xx
STAA    DCSshadow
STAA    DrillControl
```

för att ettställa en bit används:

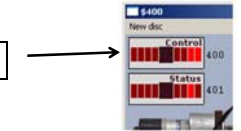
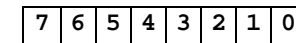
```
LDAA    DCSshadow
ORAA    #xx
STAA    DCSshadow
STAA    DrillControl
```

```
DCSshadow    RMB    1
```

Subrutiner för att manipulera styrregistret Outone och Outzero

```
* Subrutin Outone. Läser kopian av
* bormaskinens styrord på adress
* DCSshadow. Ettställer en av bitarna och
* skriver det nya styrordet till
* utporten DrillControl samt tillbaka till
* kopian DCSshadow.
* Biten som nollställs ges av innehållet
* i B-registret (0-7) vid anrop.
* Om (B) > 7 utförs ingenting.
* Anrop:          LDAB    #bitnummer
*                JSR     Outone
* Utdata:         Inga
* Registerpåverkan: Ingen
* Anropade subrutiner: Inga
```

"bitnummer" = 0..7

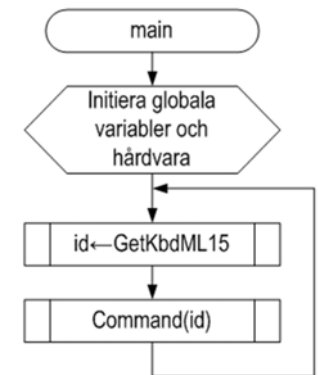


Fördröjningar i mekaniska delar

- Starta bormotorn
(vänta tills den är uppe i varv, ca 1 sekund)
- Vrid arbetsstycket ett steg
(vänta tills det har vridits till rätt position, ca 200 ms)
- Lyft borret
(vänta tills borret har kommit ovanför arbetsstycket, ca 300ms)
- etc

Bormaskinrobot

tangent kod	Operation	subrutin
0	starta bormotorn	MotorStart
1	stoppa bormotorn	MotorStop
2	sänk borret	DrillDown
3	höj borret	DrillUp
4	rotera arbetsstycket medurs ett steg	Step
5	borra ett hål	DrillHole
6	stega arbetsstycket till referensposition	RefPos
7	borra hål längs cirkeln enligt mönster	DoAuto



Rutinen Command

Uppgift 88

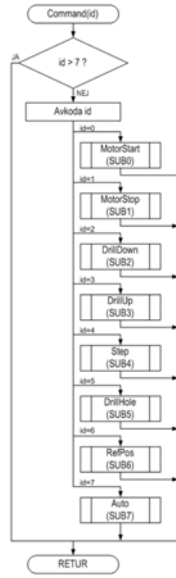
```

Command:
* giltigt värde?
  CMPB    #7
  BHI    CommandExit
* hopp tabellens basadress
  LDH    #JUMPTAB
* offset är 2 bytes per adress

  ASLB
* hämta subrutinens startadress
  LDH    B,X
* utför subrutin
  JSR    ,X
* återvänd från kommandorutin
CommandExit:
  RTS
  
```

```

*****
* Tabell med subrutinadresser
JUMPTAB  FDB    SUB0,SUB1,SUB2,SUB3
         FDB    SUB4,SUB5,SUB6,SUB7
*****
* subrutiner för test, byts senare
* ut mot START, STOP, DOWN etc
SUB0     MOVW #0,ParOut
         RTS
SUB1     MOVW #1,ParOut
         RTS
SUB2     MOVW #2,ParOut
         RTS
SUB3     MOVW #3,ParOut
         RTS
SUB4     MOVW #4,ParOut
         RTS
SUB5     MOVW #5,ParOut
         RTS
SUB6     MOVW #6,ParOut
         RTS
SUB7     MOVW #7,ParOut
         RTS
  
```



Filen Main.s12

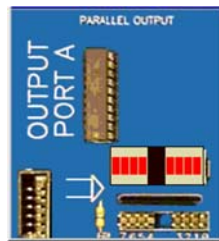
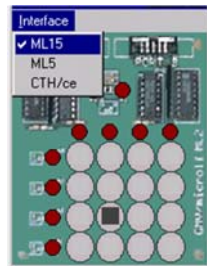
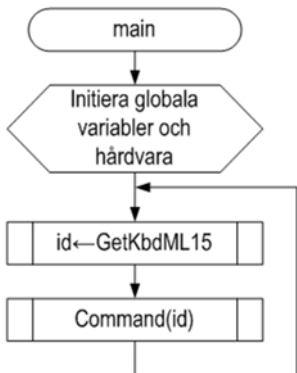
STRUKTUR

1. Inkludera definitionsfil
2. Initiera bormaskin
3. Huvudprogram
4. Subrutinen COMMAND
5. Inkludera fil (filer) med ytterligare subrutiner.
6. Plats för variabler

```

; Definitioner
USE      Labdefs.s12
ORG      $1000
; *****
; Huvudprogram
; Invänta vald operation
main_loop:
  JSR    GetKbdML15
; Utför vald operation
  JSR    Command
  BRA    main_loop
; *****
Command:
; *****
; Placera alla variabler här
DCShadow RMB    1
  
```

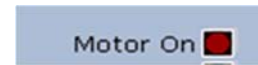
Att testa filen Main.s12



```

SUB0     MOVW #0,ParOut
         RTS
SUB1     MOVW #1,ParOut
         RTS
SUB2     MOVW #2,ParOut
         RTS
SUB3     MOVW #3,ParOut
         RTS
SUB4     MOVW #4,ParOut
         RTS
SUB5     MOVW #5,ParOut
         RTS
SUB6     MOVW #6,ParOut
         RTS
SUB7     MOVW #7,ParOut
         RTS
  
```

Rutiner MotorStart och MotorStop



```

; SUBRUTIN MotorStart.
; Subrutinen startar
; bormotorn väntar därefter i 1 sekund
; före återhopp så att borret uppnår
; rätt hastighet.
*
* Anrop:                               JSR    MotorStart
*
* Indata:                               Inga
* Utdata:                               Inga
* Registerpåverkan:                    Ingen
* Anropade subrutiner:                 Outone
*
*                                     DELAY
  
```

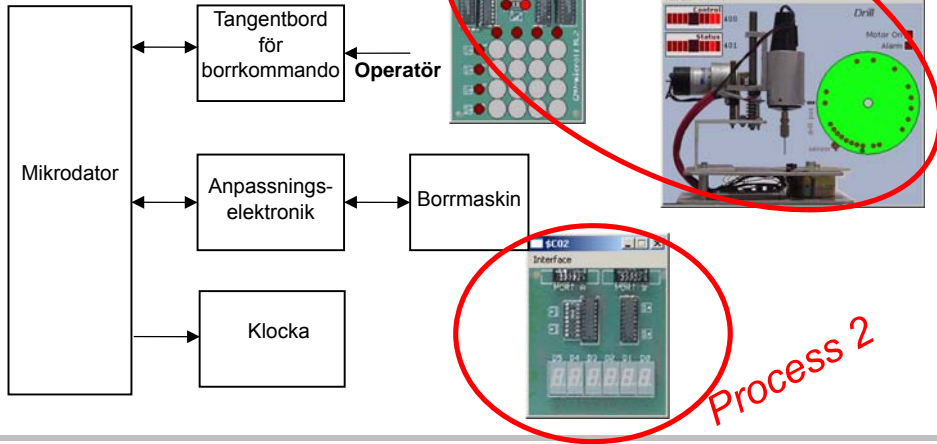
```

* SUBRUTIN STOP. Subrutinen stannar
* bormotorn.
*
* Anrop:                               JSR    MotorStop
* Indata:                               Inga
* Utdata:                               Inga
* Registerpåverkan:                    Ingen
* Anropade subrutiner:                 Outone
  
```

```

Ändringar i huvudprogram...
JUMPTAB  FDB    MotorStart, MotorStop
         FDB    SUB2,SUB3
         FDB    SUB4,SUB5,SUB6,SUB7
*****
* subrutiner för test, byts senare
* ut mot START, STOP, DOWN etc
MotorStart .....
         RTS
MotorStop .....
         RTS
SUB2     MOVW #2,ParOut
         RTS
.....
  
```

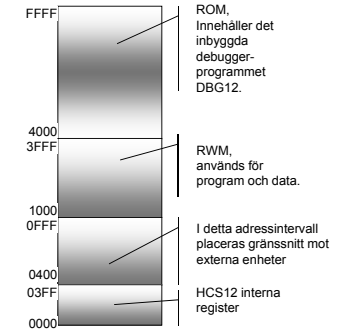

Laborationsmoment 2 Pseudoparallell exekvering



Applikation för avbrott (IRQ)

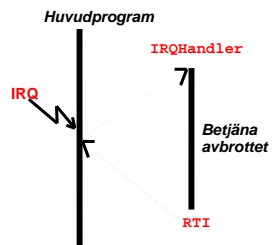
```
;Initieringssekvens
    ORG    XXXX
    ...
; nollställ I-flagga
    ANDCC #\$FE
    JSR    _main
    ...
```

```
    ORG    \$FFF2
    FDB    irq_service_routine
; Avbrotts hanterare
irq_service_routine:
    ...
    RTI
```



I laborationssystemet (MC12) kan vi INTE placera avbrottsvektorena på deras rätta platser (konflikt med DBG12)
I stället placeras dom i RWM

MC12 (DBG12) och avbrott



Vektor ROM	Funktion
FFFE	RESET, Startvektor
FFFC	Clock Monitor Fail, JMP [3FFC]
FFFA	COP Watchdog Timeout, JMP [3FFA]
FFF8	Illegal Op Code, JMP [3FF8]
FFF6	SWI, JMP [3FF6]
FFF4	XIRQ, JMP [3FF4]
FFF2	IRQ, JMP [3FF2]
FF8C	Enhetsspecifika vektorer
FFF0	

Allmänt

```
    ORG    \$FFF2
    FDB    irq_service_routine
; Avbrotts hanterare
irq_service_routine:
    ...
    RTI
```

Men i MC12 och simulator...

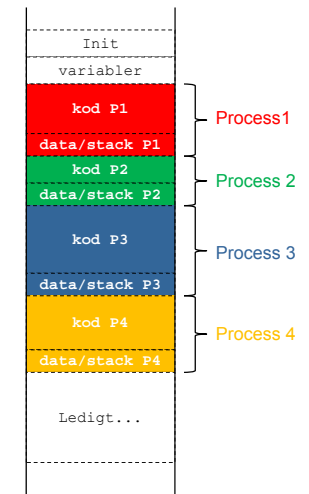
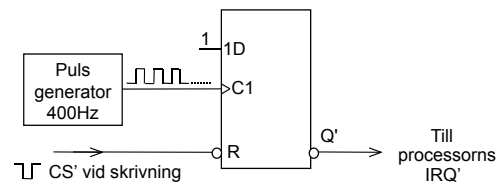
```
    ORG    \$3FF2
    FDB    irq_service_routine
; Avbrotts hanterare
irq_service_routine:
    ...
    RTI
```

Processbyte

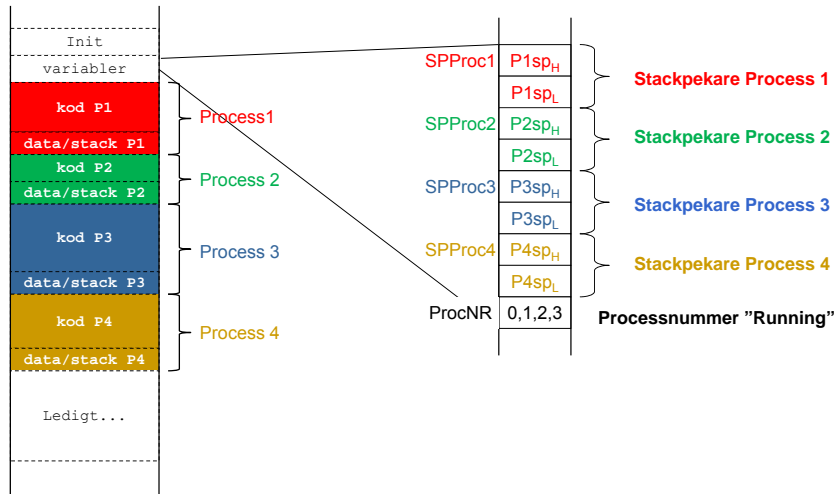
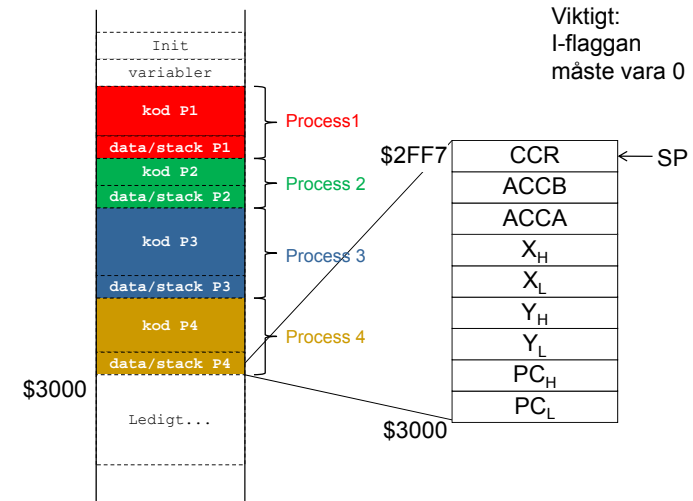
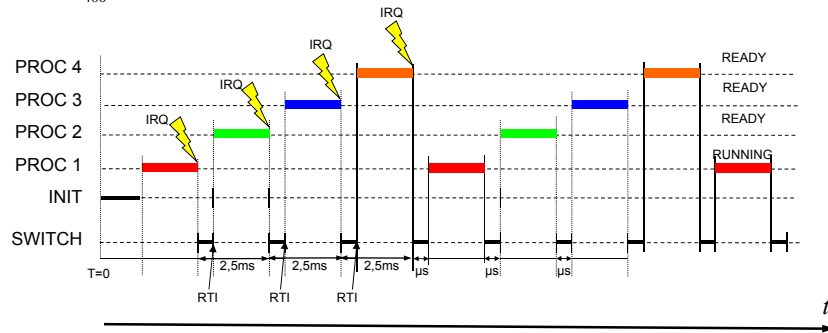
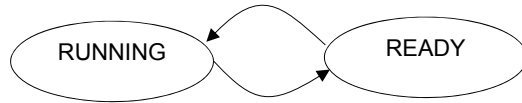
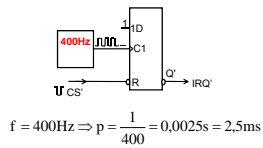
En processor
– flera program
– körs "samtidigt" (pseudoparallellt)

HDW krav: En avbrottskälla som ger regelbundna avbrott (Ex Timer)

SW krav: En avbrottsrutin (SWITCH) som växlar process



Processtillstånd

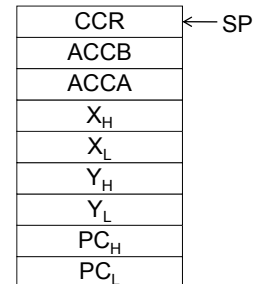


Initial stack för process och "processbyte":

```

    ORG    TopOfStack-9
    Istack:FCB    $C0    ; Initialt CCR
                FCB    0    ; Initialt B
                FCB    0    ; Initialt A
                FDB    0    ; Initialt X
                FDB    0    ; Initialt Y
                FDB    Start ; Initialt PC

    ORG    Code
    LDD    #Istack
    STD    SPProc
    
```



```

    IRQHandler:
    ; Spara "Running" stackpekare
        STS    ...
    ; Välj ny "Running"
        LDS    ...
    ; Återstarta
        RTI
    
```