# Programmering av inbyggda system 2014/2015

CPU12 Reference Guide
Stencil: ”Assemblerprogrammering.pdf”

Ur innehållet:

- Räknarkretsar (”TIMERS”)
- Pulsbreddsmodulering (”PWM”)
- Analog-/Digital- omvandling (”AD”)
- Seriekommunikation (”SCI”)

# CRG, Clock Reset Generator

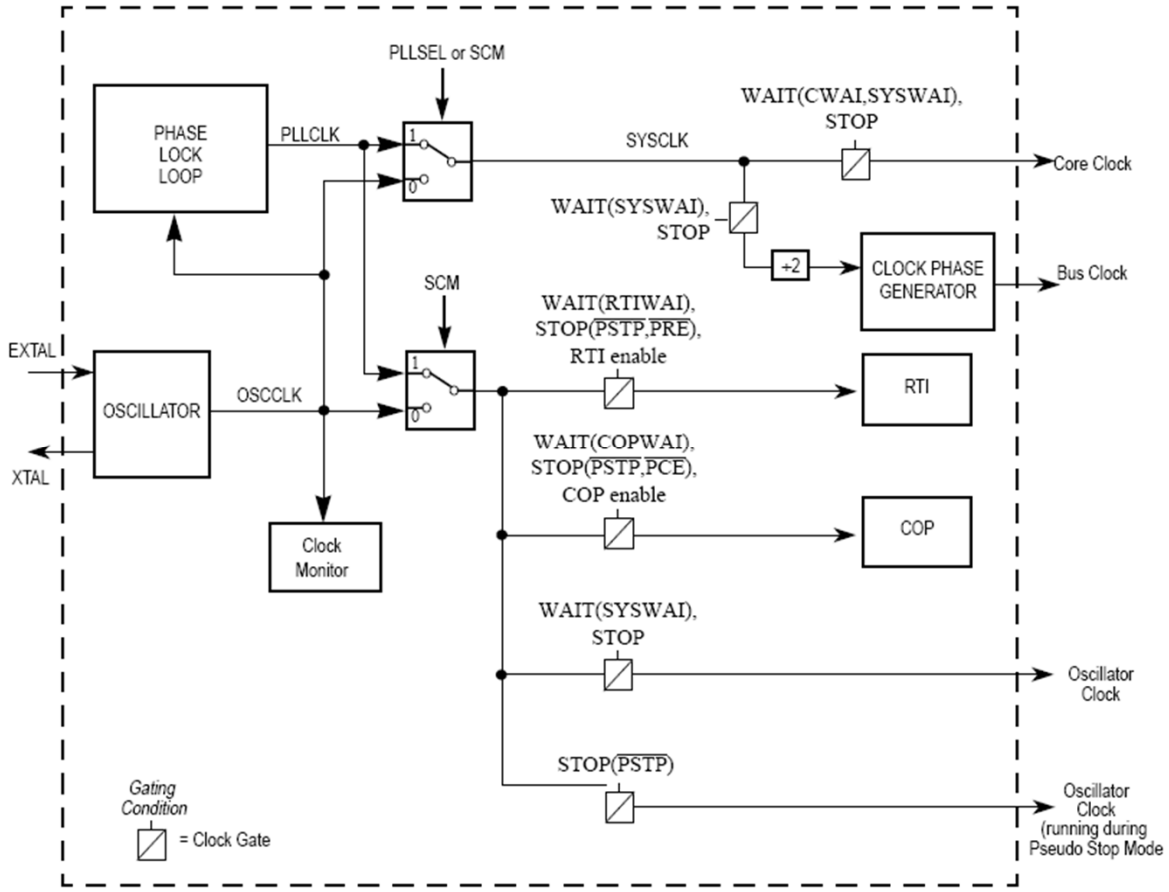
HCS12 har programmerbar arbetstakt. Kontrolleras från CRG-modul.

| Address Offset | Use | Access |
|---|---|---|
| $_00 | CRG Synthesizer Register (SYNR) | R/W |
| $_01 | CRG Reference Divider Register (REFDV) | R/W |
| $_02 | CRG Test Flags Register (CTFLG)[1] | R/W |
| $_03 | CRG Flags Register (CRGFLG) | R/W |
| $_04 | CRG Interrupt Enable Register (CRGINT) | R/W |
| $_05 | CRG Clock Select Register (CLKSEL) | R/W |
| $_06 | CRG PLL Control Register (PLLCTL) | R/W |
| $_07 | CRG RTI Control Register (RTICTL) | R/W |
| $_08 | CRG COP Control Register (COPCTL) | R/W |
| $_09 | CRG Force and Bypass Test Register (FORBYP)[2] | R/W |
| $_0A | CRG Test Control Register (CTCTL)[3] | R/W |
| $_0B | CRG COP Arm/Timer Reset (ARMCOP) | R/W |

NOTES:
1. CTFLG is intended for factory test purposes only.
2. FORBYP is intended for factory test purposes only.
3. CTCTL is intended for factory test purposes only.

$$PLLCLK = 2 \times OSCCLK \times \frac{(SYNR + 1)}{(REFDV + 1)}$$

BusClock (E) = PLLCLK/2



# EXEMPEL: Bestäm busfrekvens

Antag 8 MHz kristall.

PLLCLK får aldrig vara *mindre än* OSCCLK eftersom detta äventyrar stabilitetsvillkoren i oscillatorn.

PLLCLK/2 får aldrig vara *större än* nominella arbetsfrekvensen hos kretsen. För första generationens HCS12 innebär detta att PLLCLK/2 < 25 MHz.

$$50MHz > 2 \times 8MHz \times \frac{(SYNR+1)}{(REFDV+1)}$$

Sätt:
SYNR = 5 och REFDV = 1

$$2 \times 8MHz \times \frac{(5+1)}{(1+1)} = 2 \times 8 \times 3MHz = 48MHz$$

Basadress = $34

Algoritm:
1. Skriv nya värden till SYNR, REFDV.

2. Vänta tills kretsen ”låser” (LOCK=1)

3. Växla till PLL (sätt PPLSEL=1)

| | Clock Reset Generator (CRG) | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Offset | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Mnemonic | Namn |
| $34 | $00 | R | 0 | 0 | SYNR5 | SYN4 | SYN3 | SYN2 | SYN1 | SYN0 | SYNR | Synthesizer Register |
| | | W | | | | | | | | | | |
| $35 | $01 | R | 0 | 0 | 0 | 0 | REFDV3 | REFDV2 | REFDV1 | REFDV0 | REFDV | Reference Divide Register |
| | | W | | | | | | | | | | |
| $36 | $02 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | CTFLG | *)Test Flags Register |
| | | W | | | | | | | | | | |
| $37 | $03 | R | RTIF | PORF | LVRF | LOCKIF | LOCK | SCMIE | SCMIF | SCM | CRGFLG | Flags Register |
| | | W | | | | | | | | | | |
| $38 | $04 | R | RTIE | 0 | 0 | LOCKIE | 0 | 0 | SCMIE | 0 | CRGINT | Interrupt Enable Register |
| | | W | | | | | | | | | | |
| $39 | $05 | R | PLLSEL | PSTP | SYSWAI | ROAWAI | PLLWAI | CWAI | RTIWAI | COPWAI | CLKSEL | Clock Select Register |
| | | W | | | | | | | | | | |
| $3A | $06 | R | CME | PLLON | AUTO | AOQ | 0 | PRE | PCE | SCME | PLLCTL | PLL Control Register |
| | | W | | | | | | | | | | |
| $3B | $07 | R | 0 | RTR6 | RTR5 | RTR4 | RTR3 | RTR2 | RTR1 | RTR0 | RTICTL | RTI Control Register |
| | | W | | | | | | | | | | |
| $3C | $08 | R | WCOP | RSBCK | 0 | 0 | 0 | CR2 | CR1 | CR0 | COPCTL | COP Control Register |
| | | W | | | | | | | | | | |
| $3D | $09 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | FORBYP | *)Force and Bypass Test Register |
| | | W | | | | | | | | | | |
| $3E | $0A | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | CTCTL | *)Test Control Register |
| | | W | | | | | | | | | | |
| $3F | $0B | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ARMCOP | COP Arm/Timer Reset |
| | | W | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 | | |

| | Clock Reset Generator (CRG) | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Offset | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Mnemonic | Namn |
| $34 | $00 | R | 0 | 0 | **SYN5** | **SYN4** | **SYN3** | **SYN2** | **SYN1** | **SYN0** | SYNR | Synthesizer Register |
| | | W | | | | | | | | | | |
| $35 | $01 | R | 0 | 0 | 0 | 0 | **REFDV3** | **REFDV2** | **REFDV1** | **REFDV0** | REFDV | Reference Divide Register |
| | | W | | | | | | | | | | |
| $36 | $02 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | CTFLG | *)Test Flags Register |
| | | W | | | | | | | | | | |
| $37 | $03 | R | RTIF | PORF | LVRF | LOCKIF | **LOCK** | SCMIE | SCMIF | SCM | CRGFLG | Flags Register |
| | | W | | | | | | | | | | |
| $38 | $04 | R | RTIE | 0 | 0 | LOCKIE | 0 | 0 | SCMIE | 0 | CRGINT | Interrupt Enable Register |
| | | W | | | | | | | | | | |
| $39 | $05 | R | **PLLSEL** | PSTP | SYSWAI | ROAWAI | PLLWAI | CWAI | RTIWAI | COPWAI | CLKSEL | Clock Select Register |
| | | W | | | | | | | | | | |
| $3A | $06 | R | CME | PLLON | AUTO | AOQ | 0 | PRE | PCE | SCME | PLLCTL | PLL Control Register |
| | | W | | | | | | | | | | |
| $3B | $07 | R | 0 | RTR6 | RTR5 | RTR4 | RTR3 | RTR2 | RTR1 | RTR0 | RTICTL | RTI Control Register |
| | | W | | | | | | | | | | |
| $3C | $08 | R | WCOP | RSBCK | 0 | 0 | 0 | CR2 | CR1 | CR0 | COPCTL | COP Control Register |
| | | W | | | | | | | | | | |
| $3D | $09 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | FORBYP | *)Force and Bypass Test Register |
| | | W | | | | | | | | | | |
| $3E | $0A | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | CTCTL | *)Test Control Register |
| | | W | | | | | | | | | | |
| $3F | $0B | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ARMCOP | COP Arm/Timer Reset |
| | | W | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 | | |

# Beskriv modulens register som en **`struct`**

| Address Offset | Use | Access |
|---|---|---|
| $_00 | CRG Synthesizer Register (SYNR) | R/W |
| $_01 | CRG Reference Divider Register (REFDV) | R/W |
| $_02 | CRG Test Flags Register (CTFLG)[1] | R/W |
| $_03 | CRG Flags Register (CRGFLG) | R/W |
| $_04 | CRG Interrupt Enable Register (CRGINT) | R/W |
| $_05 | CRG Clock Select Register (CLKSEL) | R/W |
| $_06 | CRG PLL Control Register (PLLCTL) | R/W |
| $_07 | CRG RTI Control Register (RTICTL) | R/W |
| $_08 | CRG COP Control Register (COPCTL) | R/W |
| $_09 | CRG Force and Bypass Test Register (FORBYP)[2] | R/W |
| $_0A | CRG Test Control Register (CTCTL)[3] | R/W |
| $_0B | CRG COP Arm/Timer Reset (ARMCOP) | R/W |

NOTES:
1. CTFLG is intended for factory test purposes only.
2. FORBYP is intended for factory test purposes only.
3. CTCTL is intended for factory test purposes only.

```
/*
  crgDEFS.h
*/
typedef struct sCRG{
          volatile unsigned char synr;
          volatile unsigned char refdv;
          volatile unsigned char ctflg;
          volatile unsigned char crgflg;
}CRG;
```

**volatile**: värdet av symbolen (**synr**) kan ändras utanför programmets kontroll

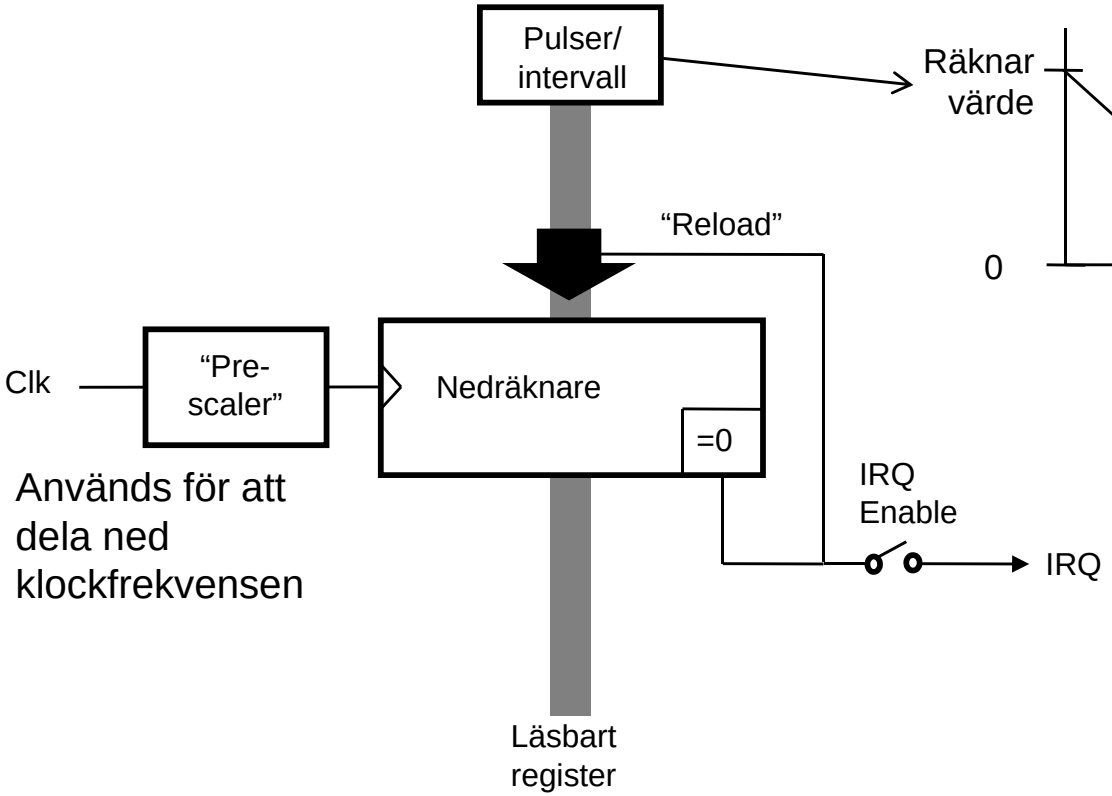
**unsigned**: för att inga implicita typkonverteringar ska sättas in av kompilatorn

**char**: därför att registrens storlek är 8 bitar

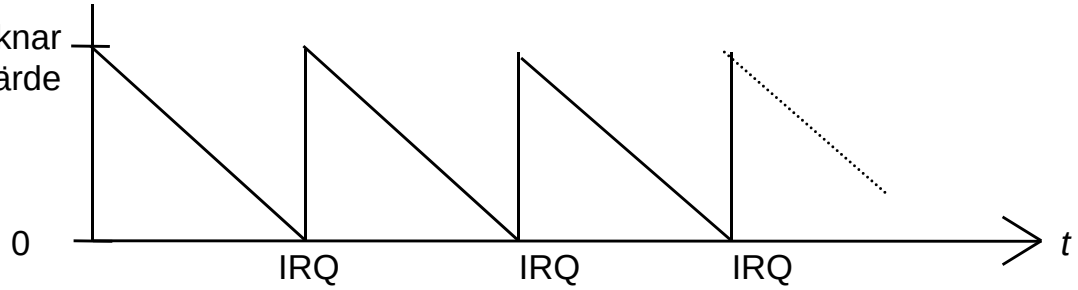
# ..programmering..

Implementera i assembler och ’C’
... Vi löser på tavlan...

# Räknarkrets (”timer”), principiell funktion




Används för att dela ned klockfrekvensen




Periodiska avbrott

Kan användas för att skapa en “REALTIDSKLOCKA”

# Realtidsklocka i HCS12

| Address Offset | Use | Access |
|---|---|---|
| $_00 | CRG Synthesizer Register (SYNR) | R/W |
| $_01 | CRG Reference Divider Register (REFDV) | R/W |
| $_02 | CRG Test Flags Register (CTFLG)[1] | R/W |
| $_03 | CRG Flags Register (CRGFLG) | R/W |
| $_04 | CRG Interrupt Enable Register (CRGINT) | R/W |
| $_05 | CRG Clock Select Register (CLKSEL) | R/W |
| $_06 | CRG PLL Control Register (PLLCTL) | R/W |
| $_07 | CRG RTI Control Register (RTICTL) | R/W |
| $_08 | CRG COP Control Register (COPCTL) | R/W |
| $_09 | CRG Force and Bypass Test Register (FORBYP)[2] | R/W |
| $_0A | CRG Test Control Register (CTCTL)[3] | R/W |
| $_0B | CRG COP Arm/Timer Reset (ARMCOP) | R/W |

NOTES:
1. CTFLG is intended for factory test purposes only.
2. FORBYP is intended for factory test purposes only.
3. CTCTL is intended for factory test purposes only.

Tre olika register används för realtidsklockan

# Realtidsklocka i HCS12, initiering

Algoritm, initiering

2. Aktivera avbrott från kretsen

1. Skriv tidbas för avbrottsintervall till RTICTL

| | Clock Reset Generator (CRG) | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Offset | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Mnemonic | Namn |
| $34 | $00 | R | 0 | 0 | SYN5 | SYN4 | SYN3 | SYN2 | SYN1 | SYN0 | SYNR | Synthesizer Register |
| | | W | | | | | | | | | | |
| $35 | $01 | R | 0 | 0 | 0 | 0 | REFDV3 | REFDV2 | REFDV1 | REFDV0 | REFDV | Reference Divide Register |
| | | W | | | | | | | | | | |
| $36 | $02 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | CTFLG | *)Test Flags Register |
| | | W | | | | | | | | | | |
| $37 | $03 | R | RTIF | PORF | LVRF | LOCKIF | LOCK | SCMIE | SCMIF | SCM | CRGFLG | Flags Register |
| | | W | | | | | | | | | | |
| $38 | $04 | R | RTIE | 0 | 0 | LOCKIE | 0 | 0 | SCMIE | 0 | CRGINT | Interrupt Enable Register |
| | | W | | | | | | | | | | |
| $39 | $05 | R | PLLSEL | PSTP | SYSWAI | ROAWAI | PLLWAI | CWAI | RTIWAI | COPWAI | CLKSEL | Clock Select Register |
| | | W | | | | | | | | | | |
| $3A | $06 | R | CME | PLLON | AUTO | AOQ | 0 | PRE | PCE | SCME | PLLCTL | PLL Control Register |
| | | W | | | | | | | | | | |
| $3B | $07 | R | 0 | RTR6 | RTR5 | RTR4 | RTR3 | RTR2 | RTR1 | RTR0 | RTICTL | RTI Control Register |
| | | W | | | | | | | | | | |
| $3C | $08 | R | WCOP | RSBCK | 0 | 0 | 0 | CR2 | CR1 | CR0 | COPCTL | COP Control Register |
| | | W | | | | | | | | | | |
| $3D | $09 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | FORBYP | *)Force and Bypass Test Register |
| | | W | | | | | | | | | | |
| $3E | $0A | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | CTCTL | *)Test Control Register |
| | | W | | | | | | | | | | |
| $3F | $0B | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ARMCOP | COP Arm/Timer Reset |
| | | W | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 | | |

# ”Prescaler” för räknarkretsen

$$\frac{OSCCLK}{RTR} = RTIfreq$$

| RTR [3:0] | RTR[6:4] | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 000 (OFF) | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
| 0000 | OFF | $2^{10}$ | $2^{11}$ | $2^{12}$ | $2^{13}$ | $2^{14}$ | $2^{15}$ | $2^{16}$ |
| 0001 | OFF | $2x2^{10}$ | $2x2^{11}$ | $2x2^{12}$ | $2x2^{13}$ | $2x2^{14}$ | $2x2^{15}$ | $2x2^{16}$ |
| 0010 | OFF | $3x2^{10}$ | $3x2^{11}$ | $3x2^{12}$ | $3x2^{13}$ | $3x2^{14}$ | $3x2^{15}$ | $3x2^{16}$ |
| 0011 | OFF | $4x2^{10}$ | $4x2^{11}$ | $4x2^{12}$ | $4x2^{13}$ | $4x2^{14}$ | $4x2^{15}$ | $4x2^{16}$ |
| 0100 | OFF | $5x2^{10}$ | $5x2^{11}$ | $5x2^{12}$ | $5x2^{13}$ | $5x2^{14}$ | $5x2^{15}$ | $5x2^{16}$ |
| 0101 | OFF | $6x2^{10}$ | $6x2^{11}$ | $6x2^{12}$ | $6x2^{13}$ | $6x2^{14}$ | $6x2^{15}$ | $6x2^{16}$ |
| 0110 | OFF | $7x2^{10}$ | $7x2^{11}$ | $7x2^{12}$ | $7x2^{13}$ | $7x2^{14}$ | $7x2^{15}$ | $7x2^{16}$ |
| 0111 | OFF | $8x2^{10}$ | $8x2^{11}$ | $8x2^{12}$ | $8x2^{13}$ | $8x2^{14}$ | $8x2^{15}$ | $8x2^{16}$ |
| 1000 | OFF | $9x2^{10}$ | $9x2^{11}$ | $9x2^{12}$ | $9x2^{13}$ | $9x2^{14}$ | $9x2^{15}$ | $9x2^{16}$ |
| 1001 | OFF | $10x2^{10}$ | $10x2^{11}$ | $10x2^{12}$ | $10x2^{13}$ | $10x2^{14}$ | $10x2^{15}$ | $10x2^{16}$ |
| 1010 | OFF | $11x2^{10}$ | $11x2^{11}$ | $11x2^{12}$ | $11x2^{13}$ | $11x2^{14}$ | $11x2^{15}$ | $11x2^{16}$ |
| 1011 | OFF | $12x2^{10}$ | $12x2^{11}$ | $12x2^{12}$ | $12x2^{13}$ | $12x2^{14}$ | $12x2^{15}$ | $12x2^{16}$ |
| 1100 | OFF | $13x2^{10}$ | $13x2^{11}$ | $13x2^{12}$ | $13x2^{13}$ | $13x2^{14}$ | $13x2^{15}$ | $13x2^{16}$ |
| 1101 | OFF | $14x2^{10}$ | $14x2^{11}$ | $14x2^{12}$ | $14x2^{13}$ | $14x2^{14}$ | $14x2^{15}$ | $14x2^{16}$ |
| 1110 | OFF | $15x2^{10}$ | $15x2^{11}$ | $15x2^{12}$ | $15x2^{13}$ | $15x2^{14}$ | $15x2^{15}$ | $15x2^{16}$ |
| 1111 | OFF | $16x2^{10}$ | $16x2^{11}$ | $16x2^{12}$ | $16x2^{13}$ | $16x2^{14}$ | $16x2^{15}$ | $16x2^{16}$ |

# Beräkning av tidbas

$$\frac{OSCCLK}{RTR} = RTIfreq \Rightarrow \frac{8\times10^6}{RTR} = \frac{1}{10^{-2}} \Rightarrow RTR = x\times2^y = 8\times10^4$$

(Se även exempel i ”Stencil 2”)
Den bästa approximationen har vi för
RTR = 100 1001 = $49, som medför: $10x2^{13} = 81920$
Eftersom detta värde är något större än det exakta, kommer vi att få en något längre periodtid, nämligen:

avbrottsfrekvens = $8\times10^6 / 81920 = 97.656$ Hz

vilket ger periodtiden:

0.01024 s = 10,24 ms.

Klockan kommer alltså att "gå för sakta" som en följd av detta systematiska fel.

# Realtidsklocka i HCS12, vid avbrott

Algoritm, kvittera avbrott

1. RTIF = 1

Clock Reset Generator (CRG)

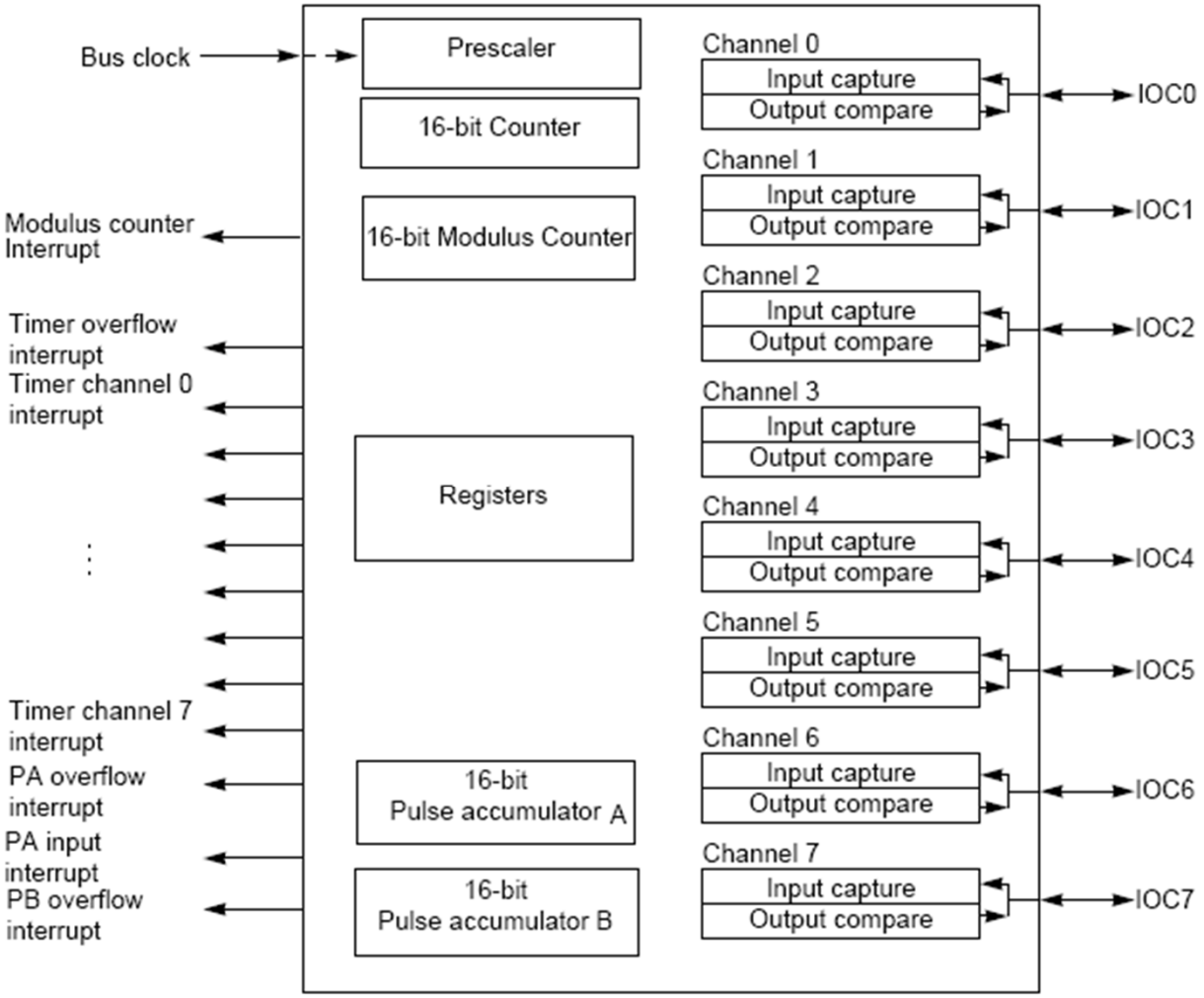
| Offset | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Mnemonic | Namn |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $34 $00 | R | 0 | 0 | SYN5 | SYN4 | SYN3 | SYN2 | SYN1 | SYN0 | SYNR | Synthesizer Register |
| | W | | | | | | | | | | |
| $35 $01 | R | 0 | 0 | 0 | 0 | REFDV3 | REFDV2 | REFDV1 | REFDV0 | REFDV | Reference Divide Register |
| | W | | | | | | | | | | |
| $36 $02 | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | CTFLG | *)Test Flags Register |
| | W | | | | | | | | | | |
| $37 $03 | R | RTIF | PORF | LVRF | LOCKIF | LOCK | SCMIE | SCMIF | SCM | CRGFLG | Flags Register |
| | W | | | | | | | | | | |
| $38 $04 | R | RTIE | 0 | 0 | LOCKIE | 0 | 0 | SCMIE | 0 | CRGINT | Interrupt Enable Register |
| | W | | | | | | | | | | |
| $39 $05 | R | PLLSEL | PSTP | SYSWAI | ROAWAI | PLLWAI | CWAI | RTIWAI | COPWAI | CLKSEL | Clock Select Register |
| | W | | | | | | | | | | |
| | R | | | | | 0 | | | | | |
| | W | | | ON | AUTO | AOQ | PRE | PCE | SCME | PLLCTL | PLL Control Register |
| | R | | | | | | | | | | |
| | W | | | 6 | RTR5 | RTR4 | RTR3 | RTR2 | RTR1 | RTR0 | RTICTL | RTI Control Register |
| | R | | | | 0 | 0 | 0 | | | | |
| | W | | | CK | | | | CR2 | CR1 | CR0 | COPCTL | COP Control Register |
| | R | | | | 0 | 0 | 0 | 0 | 0 | 0 | |
| | W | | | | | | | | | | FORBYP | *)Force and Bypass Test Register |
| | R | | | | 0 | 0 | 0 | 0 | 0 | 0 | |
| | W | | | | | | | | | | CTCTL | *)Test Control Register |
| | R | | | | 0 | 0 | 0 | 0 | 0 | 0 | |
| B | W | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 | ARMCOP | COP Arm/Timer Reset |

| Adress (hex) | Funktion |
|---|---|
| **FFF0** | **Real Time Interrupt** |
| FFEE | Enhanced Capture Timer channel |
| FFEC | Enhanced Capture Timer channel 1 |
| FFEA | Enhanced Capture Timer channel 2 |
| .... | .... |
| FF8E | Port P Interrupt |
| FF8C | PWM Emergency Shutdown |
| FF8A-FF80 | Reserverade |

# ..programmering..

Implementera i assembler och ’C’
... Vi löser på tavlan...

# Realtidsklocka med hög upplösning




”Enhanced Capture Timer” (ECT)

En maskincykels noggrannhet

EXEMPEL:
Arbetstakt= 24 MHz

PERIOD = 24 000

Intervall = 1 ms

Noggrannhet = 1/24 000 000 sek. $\approx 41{,}7 \times 10^{-9}$ sec.

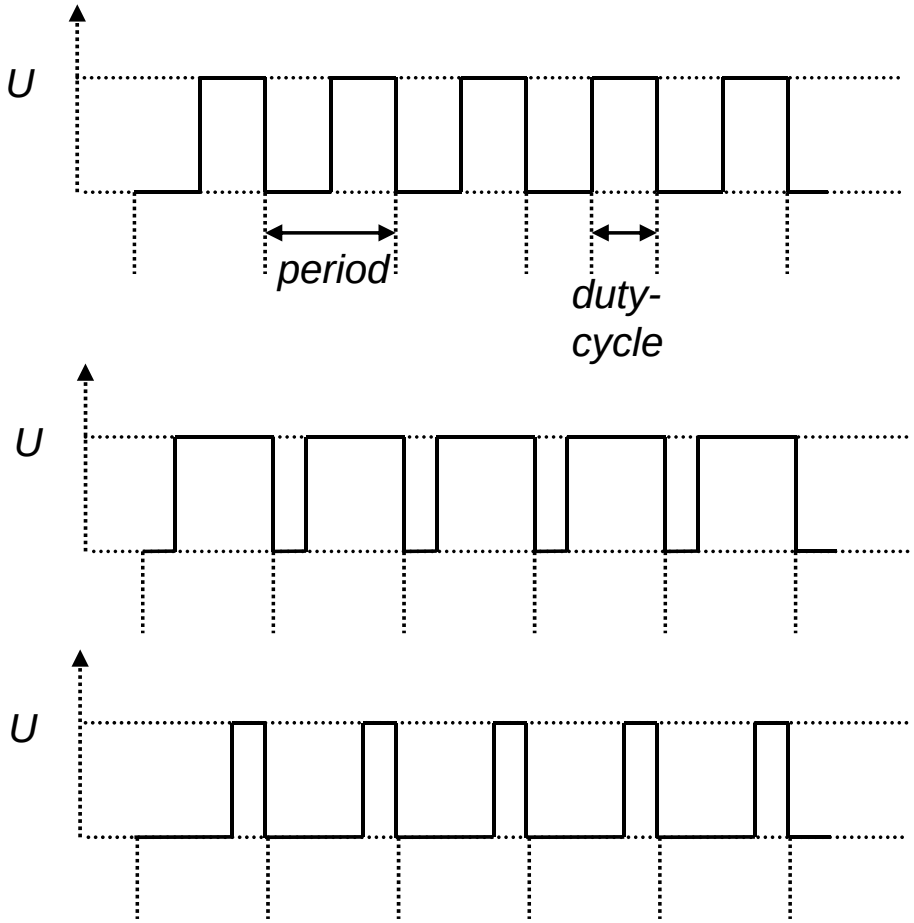
# Programexempel

```
TIOS     EQU     $40
TCNT     EQU     $44
TIE      EQU     $4C
TFLG1    EQU     $4E
TOC_0    EQU     $50

PERIOD   EQU     24000

Init:    MOVB    #1,TIOS  ; ch 0 är OC
         MOVB    #1,TIE   ; tillåt IRQ
         LDD     TCNT     ; aktuell cykel
         ADDD    #PERIOD  ; addera period
         STD     TOC_0    ; nästa avbrott
         RTS

         ORG     $FFEE
         FDB     TOCirq
```

```
TOCirq : MOVB    #1,TFLG1 ; kvittera
         LDD     TCNT     ; ny period
         ADDD    #PERIOD
         STD     TOC_0
         RTI
```

| Adress (hex) | Funktion |
|---|---|
| FFF0 | Real Time Interrupt |
| **FFEE** | **Enhanced Capture Timer channel 0** |
| FFEC | Enhanced Capture Timer channel 1 |
| FFEA | Enhanced Capture Timer channel 2 |
| .... | .... |
| FF8E | Port P Interrupt |
| FF8C | PWM Emergency Shutdown |
| FF8A-FF80 | Reserverade |

# Pulsbreddsmodulering (PWM)




$U_{out} = \frac{1}{2}U$

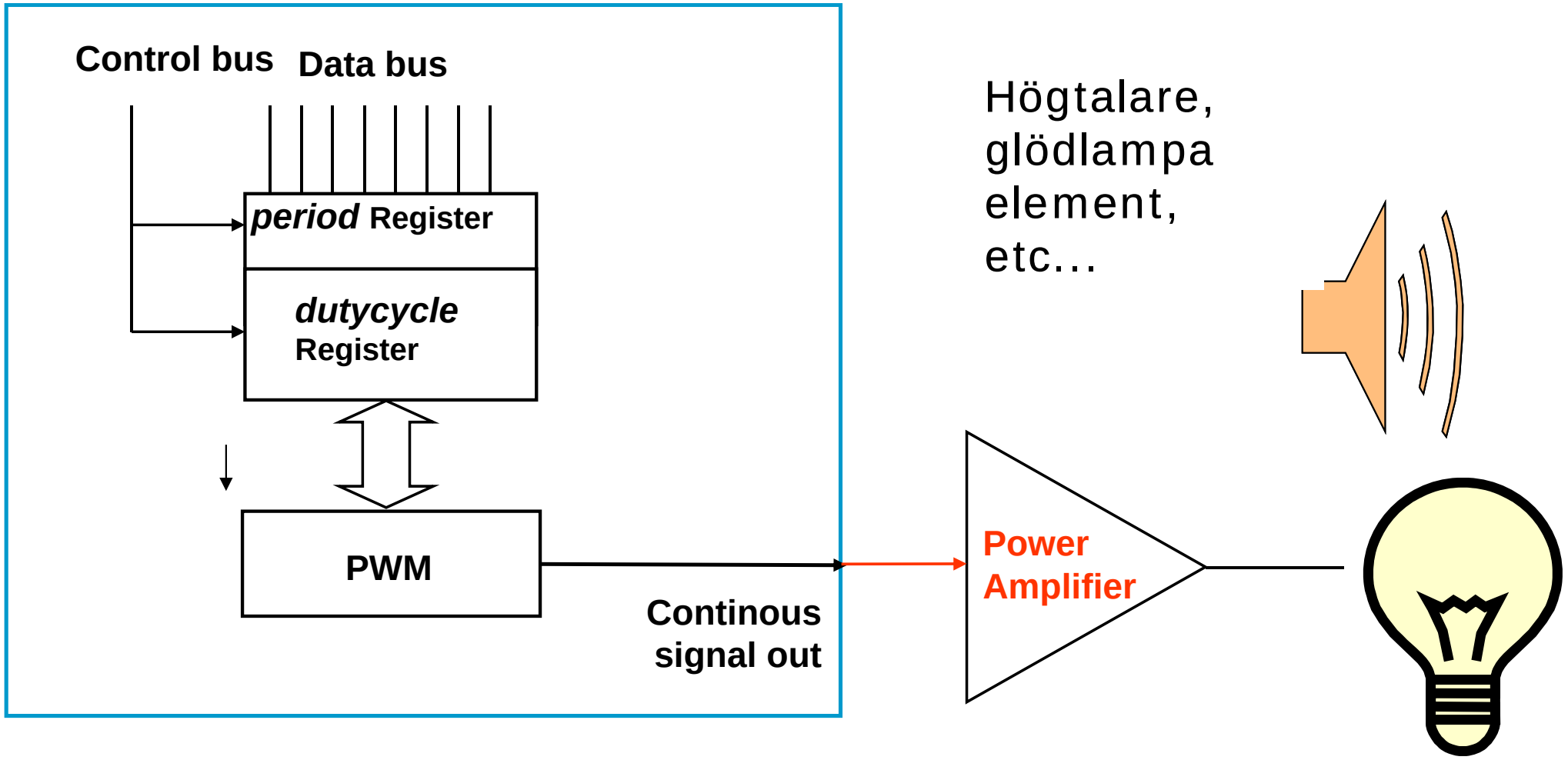
$U_{out} = \frac{3}{4}U$

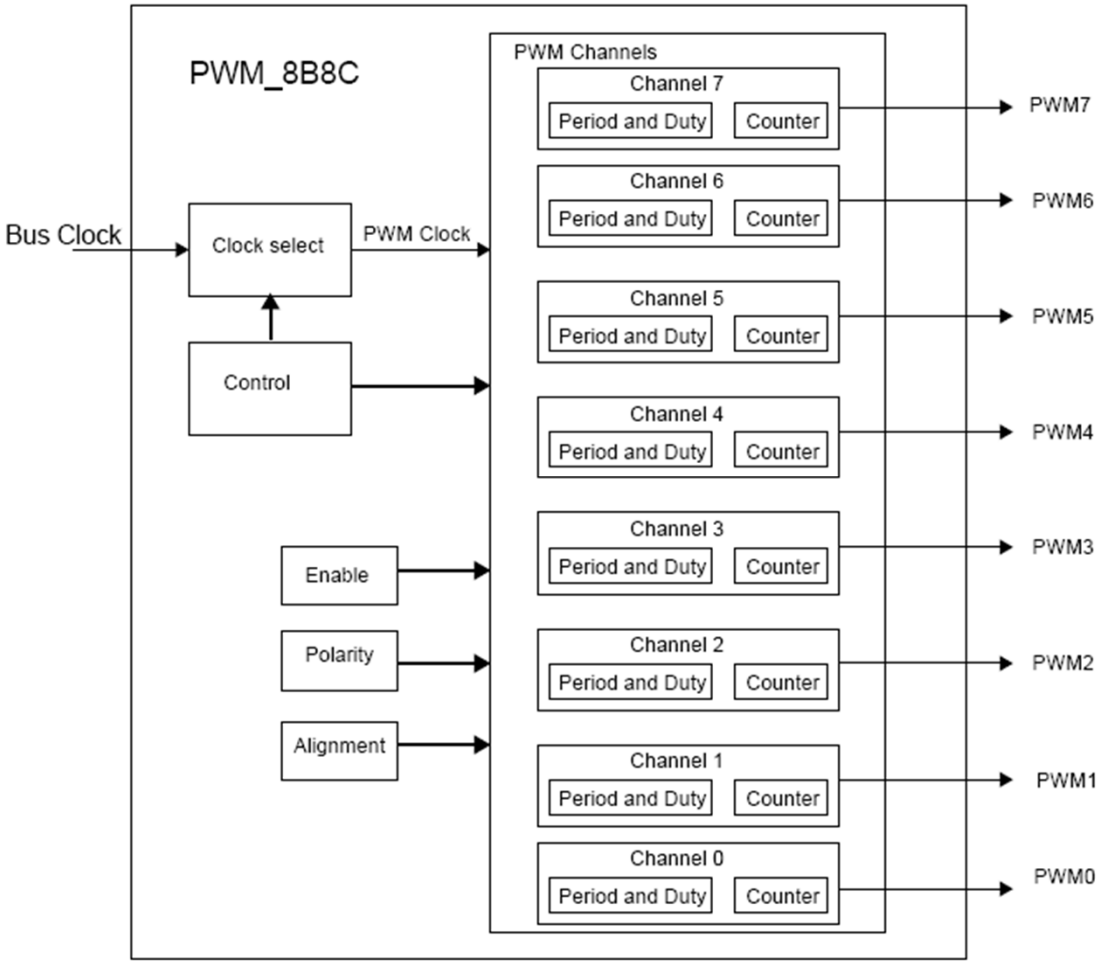
$U_{out} = \frac{1}{4}U$

$$U_{out} = \frac{dutycycle}{period}U$$

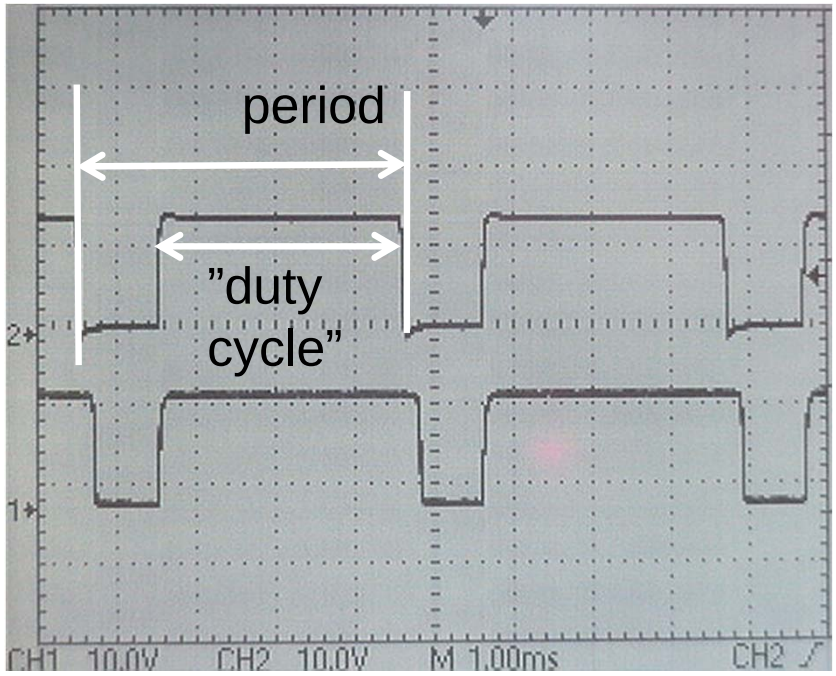
Period och ”duty-cycle” är programmerbart

# PWM-styrning






8 * 8 bitars
eller
4 * 16 bitars  räknare

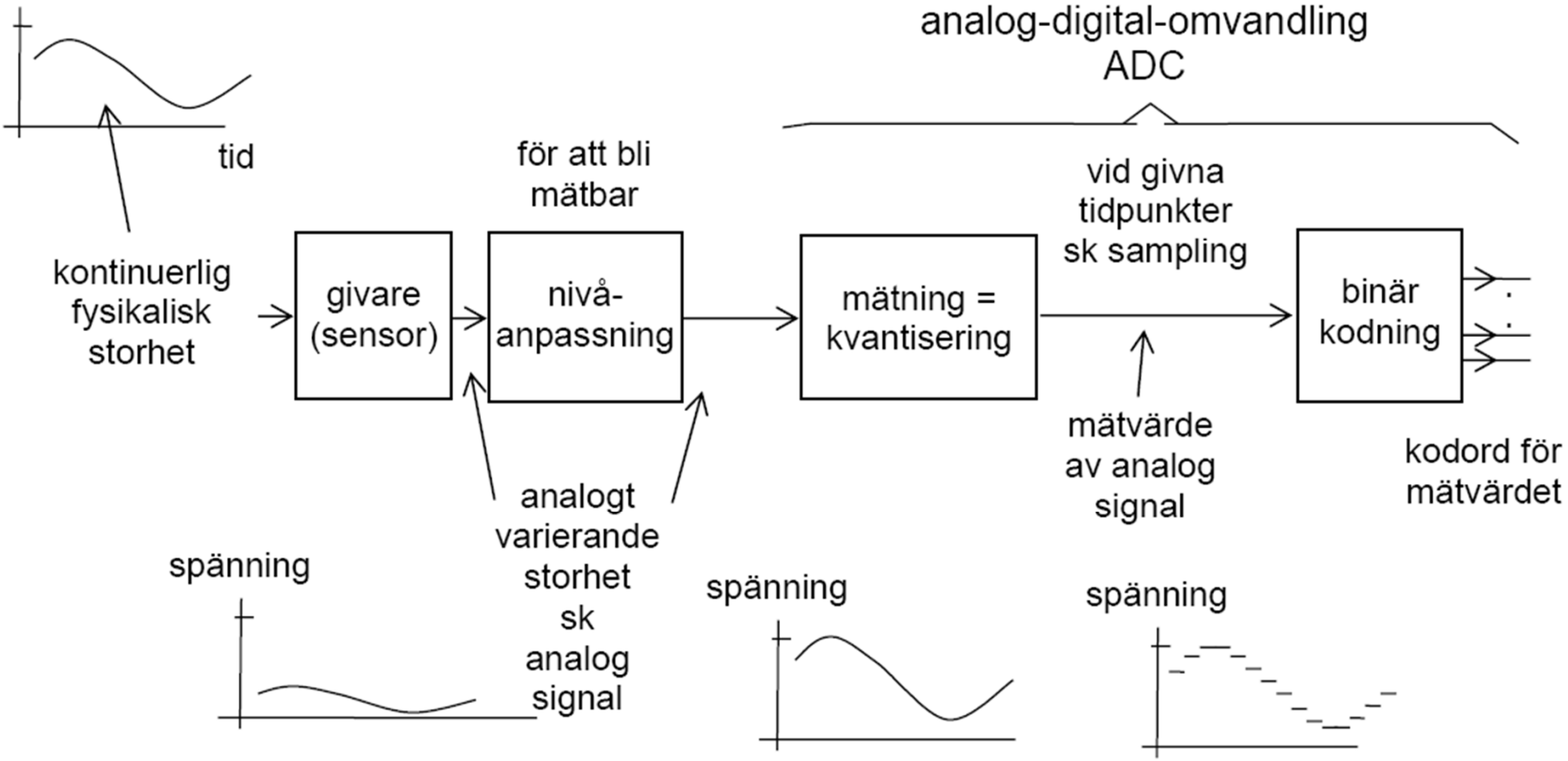


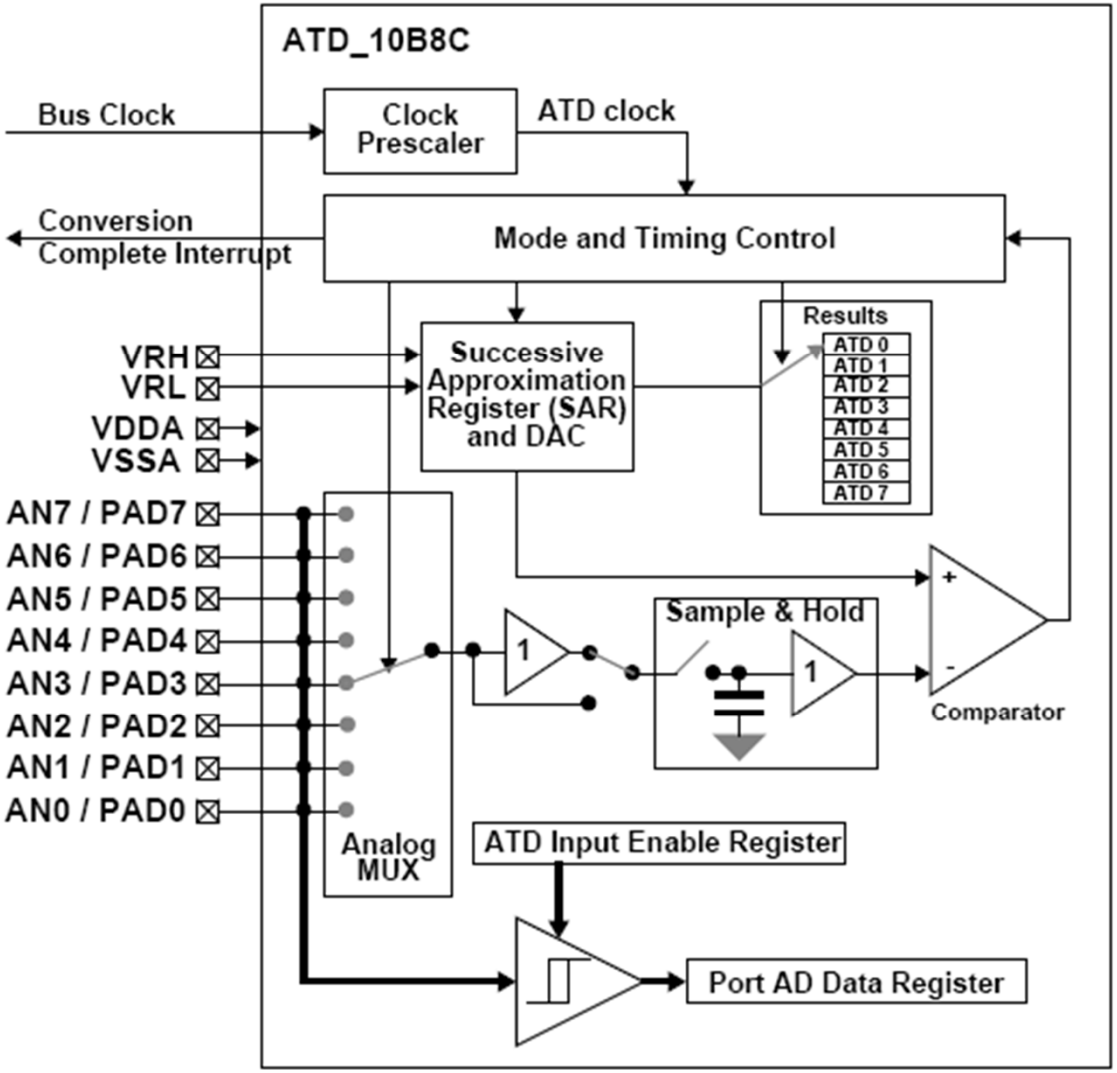
# Programexempel

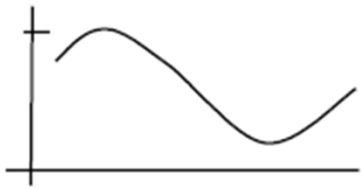
| Address | Use | Access |
|---|---|---|
| $_00 | PWM Enable Register (PWME) | R/W |
| $_01 | PWM Polarity Register (PWMPOL) | R/W |
| $_02 | PWM Clock Select Register (PWMCLK) | R/W |
| $_03 | PWM Prescale Clock Select Register (PWMPRCLK) | R/W |
| $_04 | PWM Center Align Enable Register (PWMCAE) | R/W |
| $_05 | PWM Control Register (PWMCTL) | R/W |
| $_06 | PWM Test Register (PWMTST)[1] | R/W |
| $_07 | PWM Prescale Counter Register (PWMPRSC)[2] | R/W |
| $_08 | PWM Scale A Register (PWMSCLA) | R/W |
| $_09 | PWM Scale B Register (PWMSCLB) | R/W |
| $_0A | PWM Scale A Counter Register (PWMSCNTA)[3] | R/W |
| $_0B | PWM Scale B Counter Register (PWMSCNTB)[4] | R/W |
| $_0C | PWM Channel 0 Counter Register (PWMCNT0) | R/W |
| $_0D | PWM Channel 1 Counter Register (PWMCNT1) | R/W |
| $_0E | PWM Channel 2 Counter Register (PWMCNT2) | R/W |
| $_0F | PWM Channel 3 Counter Register (PWMCNT3) | R/W |
| $_10 | PWM Channel 4 Counter Register (PWMCNT4) | R/W |
| $_11 | PWM Channel 5 Counter Register (PWMCNT5) | R/W |
| $_12 | PWM Channel 6 Counter Register (PWMCNT6) | R/W |
| $_13 | PWM Channel 7 Counter Register (PWMCNT7) | R/W |
| $_14 | PWM Channel 0 Period Register (PWMPER0) | R/W |
| $_15 | PWM Channel 1 Period Register (PWMPER1) | R/W |
| $_16 | PWM Channel 2 Period Register (PWMPER2) | R/W |
| $_17 | PWM Channel 3 Period Register (PWMPER3) | R/W |
| $_18 | PWM Channel 4 Period Register (PWMPER4) | R/W |
| $_19 | PWM Channel 5 Period Register (PWMPER5) | R/W |
| $_1A | PWM Channel 6 Period Register (PWMPER6) | R/W |
| $_1B | PWM Channel 7 Period Register (PWMPER7) | R/W |
| $_1C | PWM Channel 0 Duty Register (PWMDTY0) | R/W |
| $_1D | PWM Channel 1 Duty Register (PWMDTY1) | R/W |
| $_1E | PWM Channel 2 Duty Register (PWMDTY2) | R/W |
| $_1F | PWM Channel 3 Duty Register (PWMDTY3) | R/W |
| $_20 | PWM Channel 4 Duty Register (PWMDTY4) | R/W |
| $_21 | PWM Channel 5 Duty Register (PWMDTY5) | R/W |
| $_22 | PWM Channel 6 Duty Register (PWMDTY6) | R/W |
| $_23 | PWM Channel 7 Duty Register (PWMDTY7) | R/W |
| $_24 | PWM Shutdown Register (PWMSDN) | R/W |
| $_25 | Reserved | R |
| $_26 | Reserved | R |
| $_27 | Reserved | R |

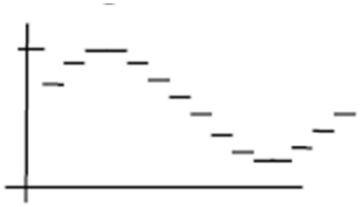
```
; PWM initiering
PWME            EQU     $A0
PWPOL           EQU     $A1
PWMPRCLK        EQU     $A3
PWMPER0         EQU     $B4
PWMDTY0         EQU     $BC

; låg nivå startar period
        CLR     PWMPOL

; c:a 4 ms periodtid
        MOVB    #$77,PWMPRCLK

; pwm kanal 0
        MOVB    #$FF,PWMPER0

; börja med 80% duty cycle..
        MOVB    #$D0,PWMDTY0

; aktivera kanal 0
        MOVB    #1,PWME
```

# Analog-/Digital- omvandling







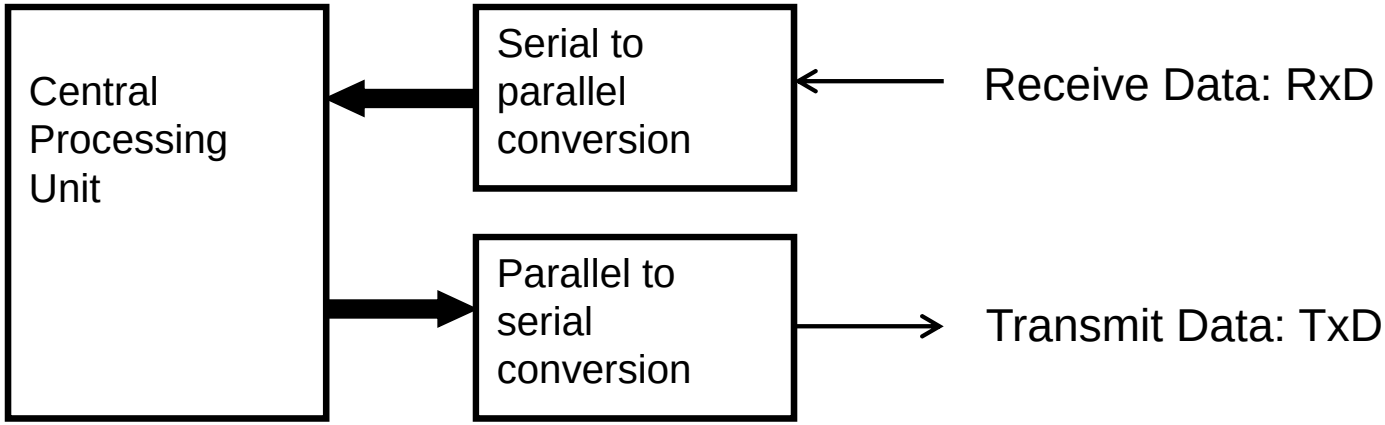
Multiplex
8 kanaler.

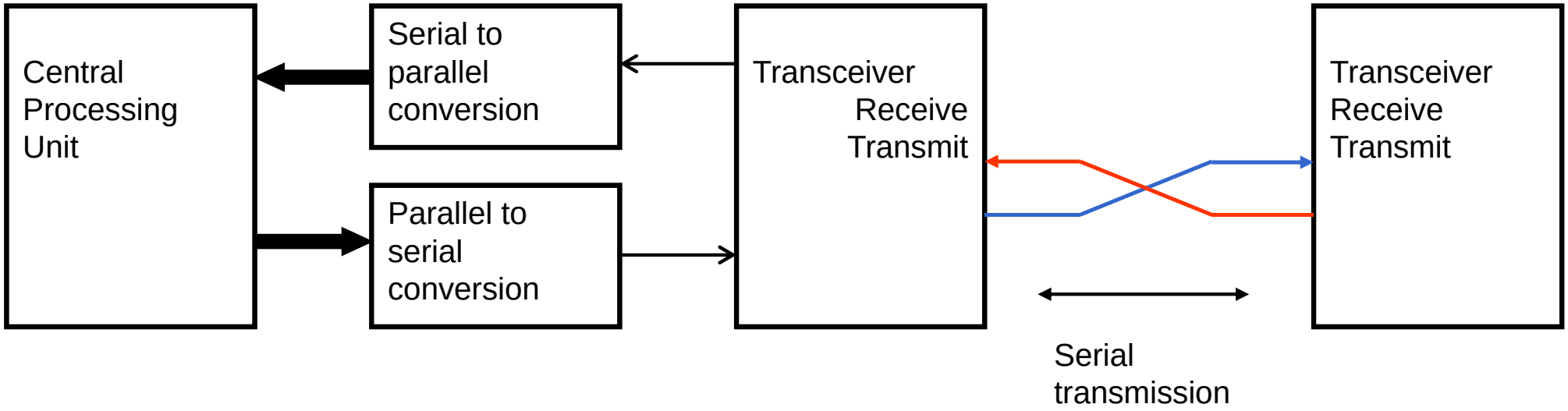
# Programexempel

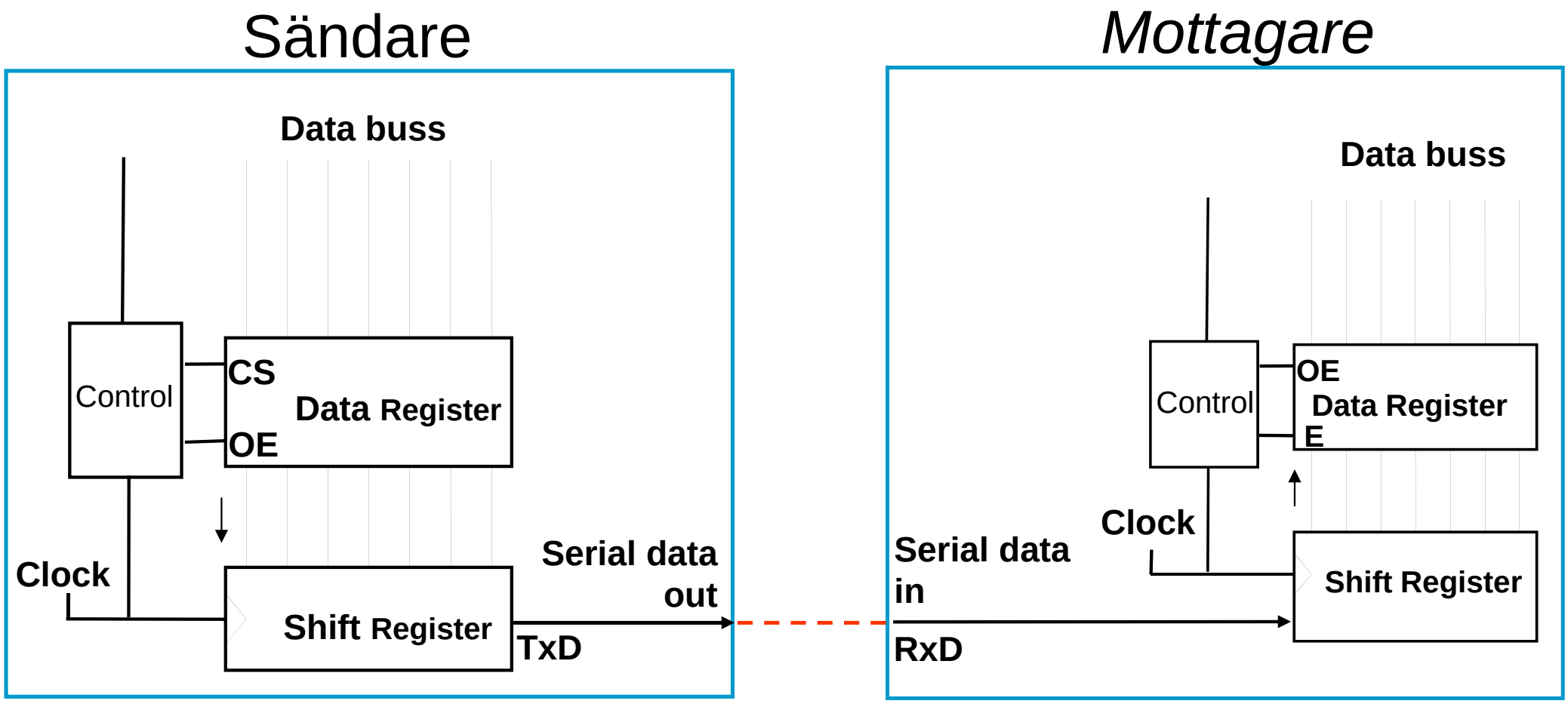
| Address Offset | Use | Access |
|---|---|---|
| $_00 | ATD Control Register 0 (ATDCTL0)[1] | R |
| $_01 | ATD Control Register 1 (ATDCTL1)[2] | R |
| $_02 | ATD Control Register 2 (ATDCTL2) | R/W |
| $_03 | ATD Control Register 3 (ATDCTL3) | R/W |
| $_04 | ATD Control Register 4 (ATDCTL4) | R/W |
| $_05 | ATD Control Register 5 (ATDCTL5) | R/W |
| $_06 | ATD Status Register 0 (ATDSTAT0) | R/W |
| $_07 | Unimplemented | |
| $_08 | ATD Test Register 0 (ATDTEST0)[3] | R |
| $_09 | ATD Test Register 1 (ATDTEST1) | R/W |
| $_0A | Unimplemented | |
| $_0B | ATD Status Register 1 (ATDSTAT1) | R |
| $_0C | Unimplemented | |
| $_0D | ATD Input Enable Register (ATDDIEN) | R/W |
| $_0E | Unimplemented | |
| $_0F | Port Data Register (PORTAD) | R |
| $_10, $_11 | ATD Result Register 0 (ATDDR0H, ATDDR0L) | R/W |
| $_12, $_13 | ATD Result Register 1 (ATDDR1H, ATDDR1L) | R/W |
| $_14, $_15 | ATD Result Register 2 (ATDDR2H, ATDDR2L) | R/W |
| $_16, $_17 | ATD Result Register 3 (ATDDR3H, ATDDR3L) | R/W |
| $_18, $_19 | ATD Result Register 4 (ATDDR4H, ATDDR4L) | R/W |
| $_1A, $_1B | ATD Result Register 5 (ATDDR5H, ATDDR5L) | R/W |
| $_1C, $_1D | ATD Result Register 6 (ATDDR6H, ATDDR6L) | R/W |
| $_1E, $_1F | ATD Result Register 7 (ATDDR7H, ATDDR7L) | R/W |

```
; AD initiering
; Högerjustera resultat, unipolärt
; kontinuerlig mode (scan), AD kanal 6

        MOVB    #$A6,ATDCTL5

; upplösning
        MOVB    #$E5,ATDCTL4

; en konverteringssekvens
        MOVB    #$40,ATDCTL3

; normal mode
        MOVB    #$C0,ATDCTL2

; Vänta tills omvandling klar
wAD:
        BRCLR   ATD0STAT0,#$80,wAD
; När resultat färdigt, läs
        LDAB    ATD0DR0L

...
```

# Seriekommunikation, SCI



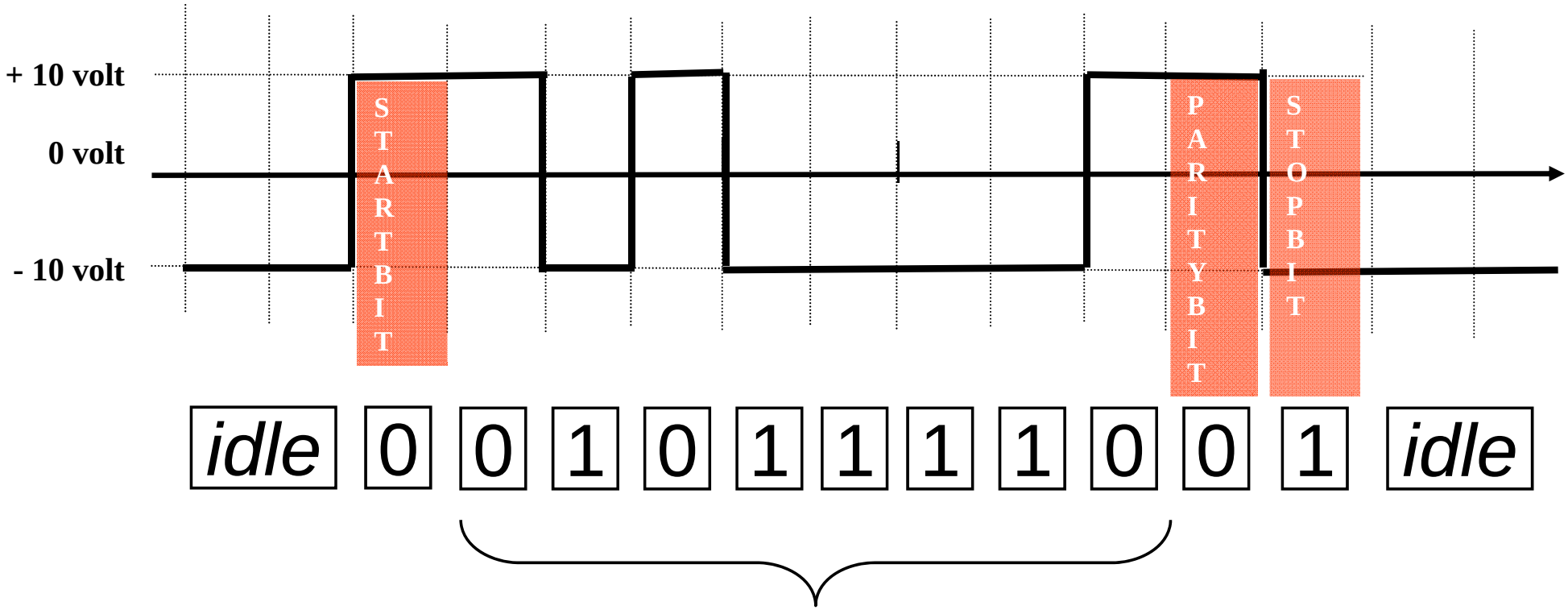






Sändare och mottagares klockor går i samma takt

# RS232 – överföring av tecknet ’z’

tecknet ”z” representeras av bitmönstret ”0111 1010” (ASCII-tecken).




’z’ – minst signifikanta bit först

# Initiering, ”busy-wait”

Basadress = $C8

Algoritm:
1. Initiera BAUDRATE

2. Aktivera
Transmitter
Receiver

Serial Communication Interface (SCI)

| Offset | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Mnemonic | Namn |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $00 | R | 0 | 0 | 0 | SBR12 | SBR11 | SBR10 | SBR9 | SBR8 | SCIBDH | Baud Rate Register High |
| | W | | | | | | | | | | |
| $01 | R | SBR7 | SBR6 | SBR5 | SBR4 | SBR3 | SBR2 | SBR1 | SBR0 | SCIBDL | Baud Rate Register Low |
| | W | | | | | | | | | | |
| $02 | R | LOOPS | SCISWAI | RSRC | M | WAKE | ILT | PE | PT | SCICR1 | Control Register 1 |
| | W | | | | | | | | | | |
| $03 | R | TIE | TCIE | RIE | ILIE | TE | RE | RWU | SBK | SCICR2 | Control Register 2 |
| | W | | | | | | | | | | |
| $04 | R | TDRE | TC | RDRF | IDLE | OR | NF | FE | PF | SCISR1 | Status Register 1 |
| | W | | | | | | | | | | |
| $05 | R | 0 | 0 | 0 | 0 | 0 | BRK13 | TXDIR | RAF | SCISR2 | Status Register 2 |
| | W | | | | | | | | | | |
| $06 | R | R8 | T8 | 0 | 0 | 0 | 0 | 0 | 0 | SCIDRH | Data Register High |
| | W | | | | | | | | | | |
| $07 | R | R7 | R6 | R5 | R4 | R3 | R2 | R1 | R0 | SCIDRL | Data Register Low |
| | W | T7 | T6 | T5 | T4 | T3 | T2 | T1 | T0 | | |

```
SCI0BD:         EQU     $C8     ; SCI 0 baudrate-register (16 bit).
SCI0CR2:        EQU     $CB     ; SCI 0 styr-register 2.
; Bitdefinitioner, styrregister
TE:             EQU     $08     ; Transmitter enable.
RE:             EQU     $04     ; Receiver enable.
```

# Skriv tecken via SCI

Algoritm:
TDRE =
(Transmit Data
Register
Empty)

1. Om TDRE=1
SCIDRL=tecken

| Serial Communication Interface (SCI) | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Offset | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Mnemonic | Namn |
| $00 | R | 0 | 0 | 0 | SBR12 | SBR11 | SBR10 | SBR9 | SBR8 | SCIBDH | Baud Rate Register High |
| | W | | | | | | | | | | |
| $01 | R | SBR7 | SBR6 | SBR5 | SBR4 | SBR3 | SBR2 | SBR1 | SBR0 | SCIBDL | Baud Rate Register Low |
| | W | | | | | | | | | | |
| $02 | R | LOOPS | SCISWAI | RSRC | M | WAKE | ILT | PE | PT | SCICR1 | Control Register 1 |
| | W | | | | | | | | | | |
| $03 | R | TIE | TCIE | RIE | ILIE | TE | RE | RWU | SBK | SCICR2 | Control Register 2 |
| | W | | | | | | | | | | |
| $04 | R | TDRE | TC | RDRF | IDLE | OR | NF | FE | PF | SCISR1 | Status Register 1 |
| | W | | | | | | | | | | |
| $05 | R | 0 | 0 | 0 | 0 | 0 | BRK13 | TXDIR | RAF | SCISR2 | Status Register 2 |
| | W | | | | | | | | | | |
| $06 | R | R8 | T8 | 0 | 0 | 0 | 0 | 0 | 0 | SCIDRH | Data Register High |
| | W | | | | | | | | | | |
| $07 | R | R7 | R6 | R5 | R4 | R3 | R2 | R1 | R0 | SCIDRL | Data Register Low |
| | W | T7 | T6 | T5 | T4 | T3 | T2 | T1 | T0 | | |

```
SCI0SR1:        EQU     $CC     ; SCI 0 status-register 1.
SCI0DRL:        EQU     $CF     ; SCI 0 data-register låg byte.
; Bitdefinitioner, statusregister
TDRE:           EQU     $80     ; Transmit data register empty status bit.
```

# Läs tecken från SCI

Algoritm:
RDRF =
(Receive Data
Register Full)

1. Om RDRF =1
tecken=SCIDRL

| Serial Communication Interface (SCI) | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Offset | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Mnemonic | Namn |
| $00 | R | 0 | 0 | 0 | SBR12 | SBR11 | SBR10 | SBR9 | SBR8 | SCIBDH | Baud Rate Register High |
| | W | | | | | | | | | | |
| $01 | R | SBR7 | SBR6 | SBR5 | SBR4 | SBR3 | SBR2 | SBR1 | SBR0 | SCIBDL | Baud Rate Register Low |
| | W | | | | | | | | | | |
| $02 | R | LOOPS | SCISWAI | RSRC | M | WAKE | ILT | PE | PT | SCICR1 | Control Register 1 |
| | W | | | | | | | | | | |
| $03 | R | TIE | TCIE | RIE | ILIE | TE | RE | RWU | SBK | SCICR2 | Control Register 2 |
| | W | | | | | | | | | | |
| $04 | R | TDRE | TC | RDRF | IDLE | OR | NF | FE | PF | SCISR1 | Status Register 1 |
| | W | | | | | | | | | | |
| $05 | R | 0 | 0 | 0 | 0 | 0 | BRK13 | TXDIR | RAF | SCISR2 | Status Register 2 |
| | W | | | | | | | | | | |
| $06 | R | R8 | T8 | 0 | 0 | 0 | 0 | 0 | 0 | SCIDRH | Data Register High |
| | W | | | | | | | | | | |
| $07 | R | R7 | R6 | R5 | R4 | R3 | R2 | R1 | R0 | SCIDRL | Data Register Low |
| | W | T7 | T6 | T5 | T4 | T3 | T2 | T1 | T0 | | |

```
SCI0SR1:        EQU     $CC     ; SCI 0 status-register 1.
SCI0DRL:        EQU     $CF     ; SCI 0 data-register låg byte.
; Bitdefinitioner, statusregister
RDRF:           EQU     $20     ; Receive data register full status bit.
```

# Bestämma Baudrate-värde

exempel: 9600 baud

PLLCLK=48 MHz -> E-klocka = 24 MHz

$$BR = \frac{PLLCLK / 2}{16 \times baudrate} \qquad baudrate = \frac{PLLCLK / 2}{16 \times BR}$$

| | | |
|---|---|---|
| 9 600 | $\frac{24 \times 10^6}{16 \times 9600} = 156{,}25$ | $\frac{24 \times 10^6}{16 \times 156} \approx 9615$ , $\frac{24 \times 10^6}{16 \times 157} \approx 9585$ |

```
Eclock:         EQU     24000000        ; 24 MHz
; BaudRate register värden, baseras på PLL-klocka
Baud9600:       EQU     (Eclock/(16*9600))
```

# ..programmering..

Implementera i assembler och ’C’
... Vi löser på tavlan...