

DAT 015 – Maskinorienterad Programmering 2011/2012

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.

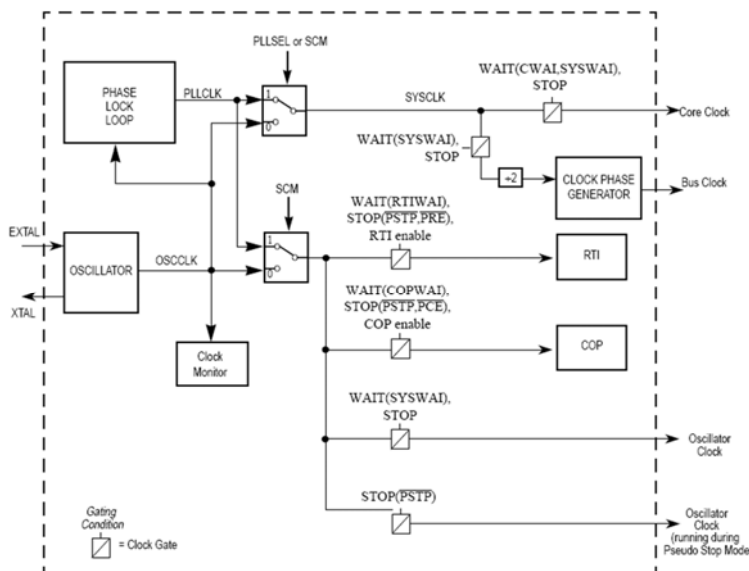
Address Offset	Use	Access
\$_00	CRG Synthesizer Register (SYNR)	R/W
\$_01	CRG Reference Divider Register (REFDV)	R/W
\$_02	CRG Test Flags Register (CTFLG) ¹	R/W
\$_03	CRG Flags Register (CRGFLG)	R/W
\$_04	CRG Interrupt Enable Register (CRGIN1)	R/W
\$_05	CRG Clock Select Register (CLKSEL)	R/W
\$_06	CRG 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.

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

$$BusClock (E) = PLLCLK / 2$$



EXEMPEL: Bestäm busfrekvens

Antag 8 MHz kristall.

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

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

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

Sätt:

$$\text{SYNR} = 5 \text{ och } \text{REFDV} = 1$$

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

Basadress = \$34

Algoritm:

1. Skriv nya värden till SYN_R, REF_{DV}.

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

3. Växla till PLL (sätt PPLSEL=1)

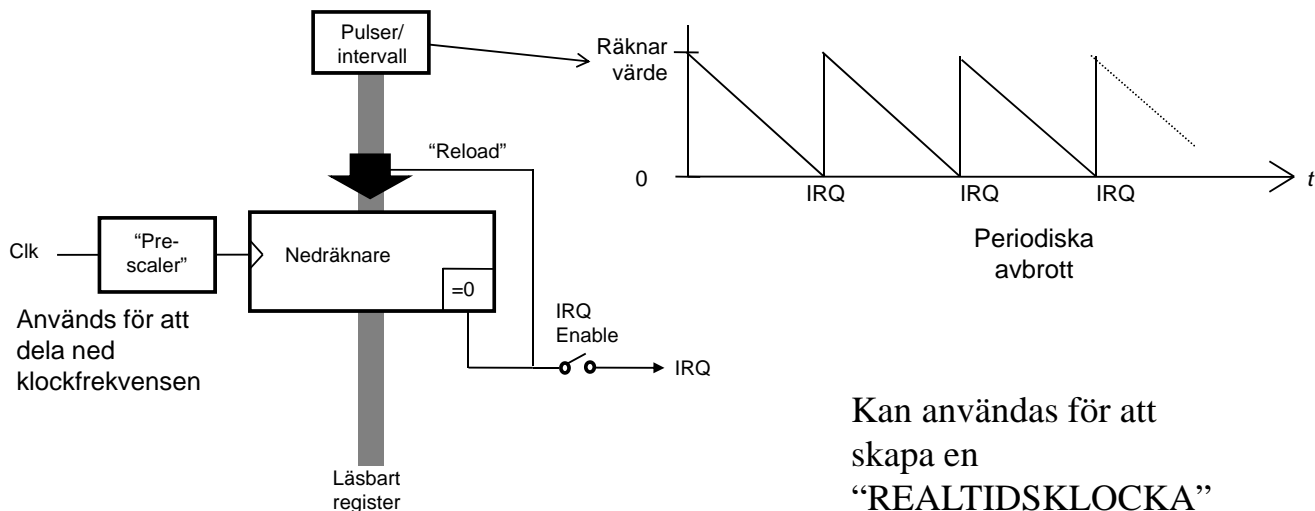
Clock Reset Generator (CRG)													
Offset		7	6	5	4	3	2	1	0	Mnemonic	Namn		
\$34	\$0	R	0	0	SYN5	SYN4	SYN3	SYN2	SYN1	SYN0	SYNR	Synthesizer Register	
	0	W											
\$35	\$0	R	0	0	0	0	REFDV	REFDV	REFDV	REFDV	REFDV	Reference Divide Register	
	1	W					3	2	1	0			
\$36	\$0	R	0	0	0	0	0	0	0	0	CTFLG	*)Test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

..programmering..

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

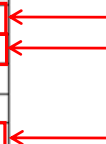
Räknarkrets ("timer"), principiell funktion



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

Tre olika register används för realtidsklockan



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.

Realtidsklocka i HCS12, initiering

Algorithm, 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	\$0	R	0	0	SYN5	SYN4	SYN3	SYN2	SYN1	SYN0	SYNR	Synthesizer Register	
	W												
\$35	\$0	R	0	0	0	0	REFDV	REFDV	REFDV	REFDV	REFDV	Reference Divide Register	
	W						3	2	1	0			
\$36	\$0	R	0	0	0	0	0	0	0	0	CTFLG	*)Test Flags Register	
	W												
\$37	\$0	R	RTIF	PORF	LVRF	LOCKIF	LOCK	SCMIE	SCMIF	SCM	CRGFLG	Flags Register	
	W												
\$38	\$0	R	RTIE	0	0	LOCKIE	0	0	SCMIE	0	CRGINT	Interrupt Enable Register	
	W												
\$39	\$0	R	PLLSEL	PSTP	SYSWAI	ROAWAI	PLLWAI	CWAI	RTIWAI	COPWAI	CLKSEL	Clock Select Register	
	W												
\$3A	\$0	R	CME	PLLON	AUTO	AOQ	0	PRE	PCE	SCME	PLLCTL	PLL Control Register	
	W												
\$3B	\$0	R	0	RTR6	RTR5	RTR4	RTR3	RTR2	RTR1	RTR0	RTICTL	RTI Control Register	
	W												
\$3C	\$0	R	WCOP	RSBCK	0	0	0	CR2	CR1	CR0	COPCTL	COP Control Register	
	W												
\$3D	\$0	R	0	0	0	0	0	0	0	0	FORBYP	*)Force and Bypass Test Register	
	W												
\$3E	\$0	R	0	0	0	0	0	0	0	0	CTCTL	*)Test Control Register	
	W												
\$3F	\$0	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

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

(Se även exempel i "Stencil 2")

Den bästa approximationen har vi för

$RTR = 100\ 1001 = \$49$, som medför: $10 \times 2^{13} = 81920$

Eftersom detta värde är något större än det exakta, kommer vi att få en något längre periodtid, nämligen:

avbrottsfrekvens = $8 \times 10^6 / 81920 = 97.656\text{ Hz}$

vilket ger periodtiden:

$0.01024\text{ s} = 10,24\text{ ms}$.

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

Realtidsklocka i HCS12, vid avbrott

Algorithm, kvittera avbrott

1. RTIF = 1

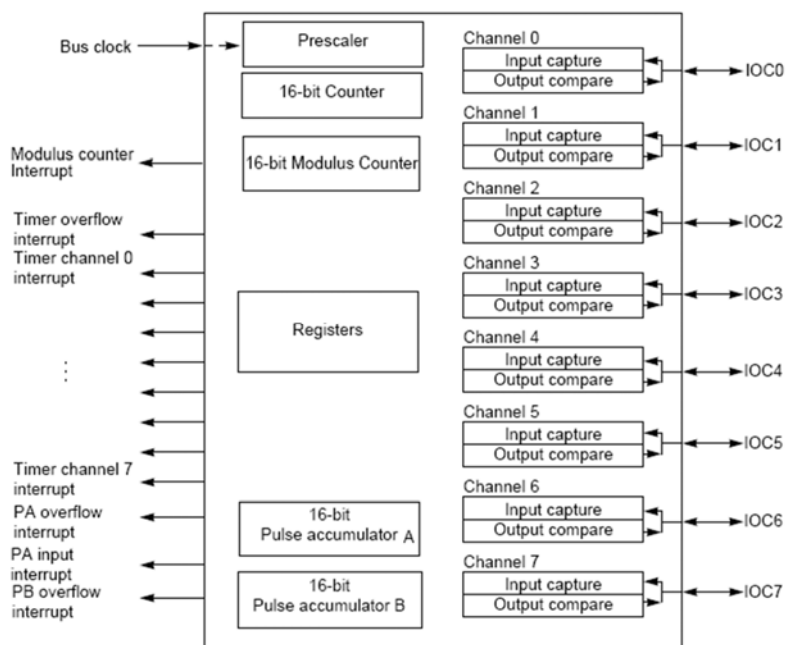
Clock Reset Generator (CRG)											
Offset	7	6	5	4	3	2	1	0	Mnemonic	Namn	
\$34	\$0 R	0	0	SYN5	SYN4	SYN3	SYN2	SYN1	SYN0	SYNR	Synthesizer Register
	0 W										
\$35	\$0 R	0	0	0	0	REFDV	REFDV	REFDV	REFDV	REFDV	Reference Divide Register
	1 W					3	2	1	0		
\$36	\$0 R	0	0	0	0	0	0	0	0	CTFLG	*)Test Flags Register
	2 W										
\$37	\$0 R	RTIF	PORF	LVRF	LOCKIF	LOCK	SCMIE	SCMIF	SCM	CRGFLG	Flags Register
	3 W										
\$38	\$0 R	RTIE	0	0	LOCKIE	0	0	SCMIE	0	CRGINT	Interrupt Enable Register
	4 W										
\$39	\$0 R	PLLSEL	PSTP	SYSWAI	ROAWAI	PLLWAI	CWAI	RTIWAI	COPWAI	CLKSEL	Clock Select Register
	5 W										
\$40	\$0 R	0	0	0	0	0	PRE	PCE	SCME	PLLCTL	PLL Control Register
	0 W										
\$41	\$0 R	0	0	0	0	0	0	0	0	RTICTL	RTI Control Register
	6 W										
\$42	\$0 R	0	0	0	0	0	0	0	0	COPCTL	COP Control Register
	7 W										
\$43	\$0 R	0	0	0	0	0	0	0	0	FORBYP	*)Force and Bypass Test Register
	8 W										
\$44	\$0 R	0	0	0	0	0	0	0	0	CTCTL	*)Test Control Register
	9 W										
\$45	\$0 R	0	0	0	0	0	0	0	0	ARMCOP	COP Arm/Timer Reset
	10 W										

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

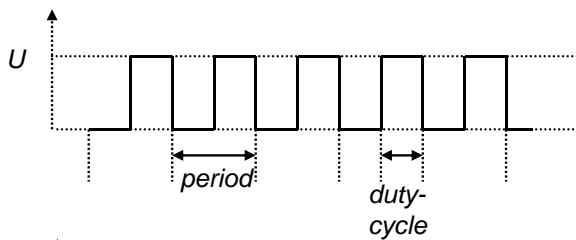
        ORG    $FFEE
        FDB    TOCirq
    
```

```

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

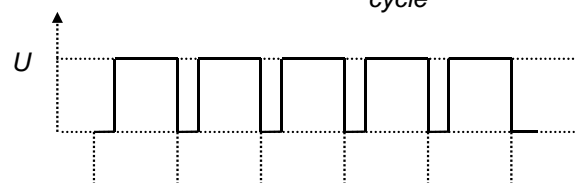
Adress (hex)	Funktion
FFF0	Real Time Interrupt
FFEE	Enhanced Capture Timer channel 0
FFEC	Enhanced Capture Timer channel 1
FFEA	Enhanced Capture Timer channel 2
....
FF8E	Port P Interrupt
FF8C	PWM Emergency Shutdown
FF8A-FF80	Reserverade

Pulsbreddsmodulering (PWM)



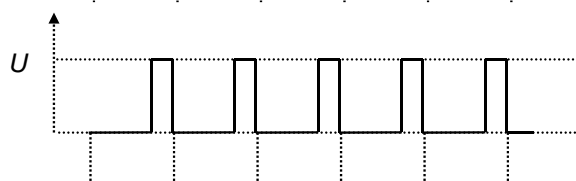
$$U_{out} = \frac{1}{2}U$$

$$U_{out} = \frac{\text{duty cycle}}{\text{period}}U$$



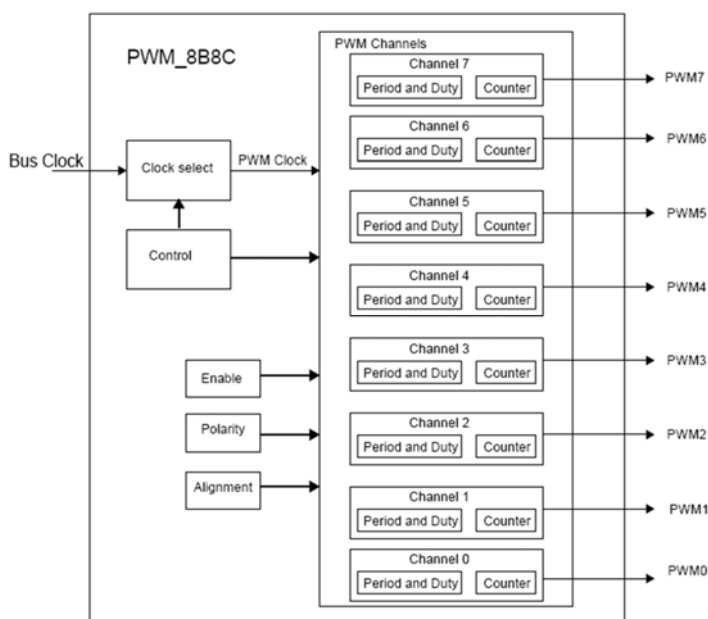
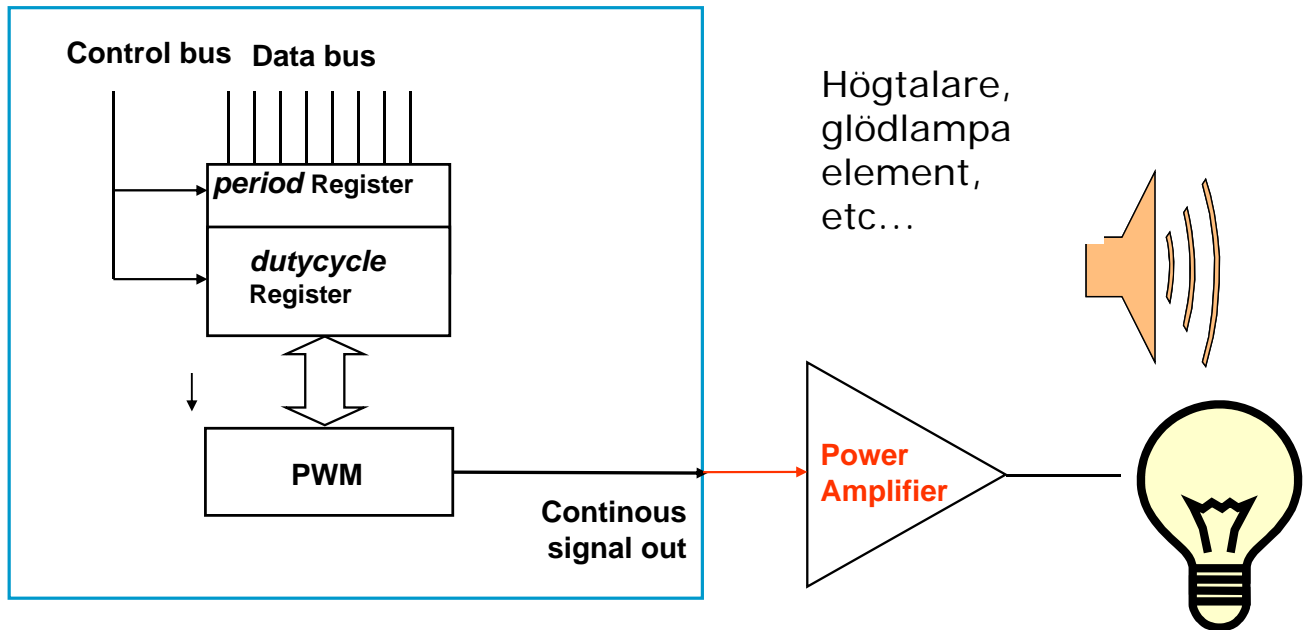
$$U_{out} = \frac{3}{4}U$$

Period och "duty-cycle" är programmerbart

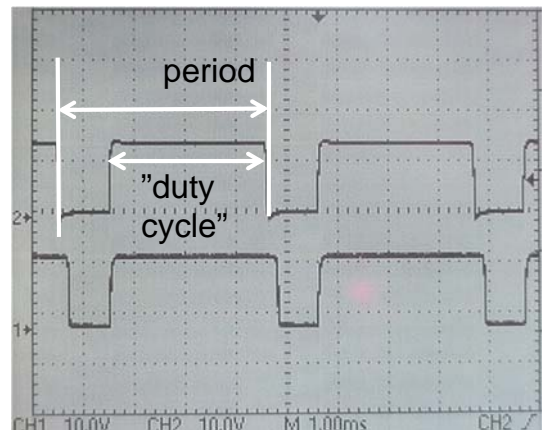


$$U_{out} = \frac{1}{4}U$$

PWM-styrning



8 * 8 bitars
eller
4 * 16 bitars räknare



Programexempel

Address	Use	Access
\$_00	PWM Enable Register (PWME)	RW
\$_01	PWM Polarity Register (PWPOL)	RW
\$_02	PWM Clock Select Register (PWPCLK)	RW
\$_03	PWM Prescale Clock Select Register (PWPCLCK)	RW
\$_04	PWM Center Align Enable Register (PWCMAE)	RW
\$_05	PWM Control Register (PWCNTL)	RW
\$_06	PWM Test Register (PWTST) ¹	RW
\$_07	PWM Prescale Counter Register (PWPSCR) ²	RW
\$_08	PWM Scale A Register (PWSCLA)	RW
\$_09	PWM Scale B Register (PWSCLB)	RW
\$_0A	PWM Scale A Counter Register (PWSCNTA) ³	RW
\$_0B	PWM Scale B Counter Register (PWSCNTB) ⁴	RW
\$_0C	PWM Channel 0 Counter Register (PWCNT0)	RW
\$_0D	PWM Channel 1 Counter Register (PWCNT1)	RW
\$_0E	PWM Channel 2 Counter Register (PWCNT2)	RW
\$_0F	PWM Channel 3 Counter Register (PWCNT3)	RW
\$_10	PWM Channel 4 Counter Register (PWCNT4)	RW
\$_11	PWM Channel 5 Counter Register (PWCNT5)	RW
\$_12	PWM Channel 6 Counter Register (PWCNT6)	RW
\$_13	PWM Channel 7 Counter Register (PWCNT7)	RW
\$_14	PWM Channel 0 Period Register (PWPERR0)	RW
\$_15	PWM Channel 1 Period Register (PWPERR1)	RW
\$_16	PWM Channel 2 Period Register (PWPERR2)	RW
\$_17	PWM Channel 3 Period Register (PWPERR3)	RW
\$_18	PWM Channel 4 Period Register (PWPERR4)	RW
\$_19	PWM Channel 5 Period Register (PWPERR5)	RW
\$_1A	PWM Channel 6 Period Register (PWPERR6)	RW
\$_1B	PWM Channel 7 Period Register (PWPERR7)	RW
\$_1C	PWM Channel 0 Duty Register (PWDY0)	RW
\$_1D	PWM Channel 1 Duty Register (PWDY1)	RW
\$_1E	PWM Channel 2 Duty Register (PWDY2)	RW
\$_1F	PWM Channel 3 Duty Register (PWDY3)	RW
\$_20	PWM Channel 4 Duty Register (PWDY4)	RW
\$_21	PWM Channel 5 Duty Register (PWDY5)	RW
\$_22	PWM Channel 6 Duty Register (PWDY6)	RW
\$_23	PWM Channel 7 Duty Register (PWDY7)	RW
\$_24	PWM Shutdown Register (PWMSDN)	RW
\$_25	Reserved	R
\$_26	Reserved	R
\$_27	Reserved	R

```

; PWM initiering
PWME      EQU    $A0
PWPOL     EQU    $A1
PWPCLCK   EQU    $A3
PWPPER0   EQU    $B4
PWMDTY0   EQU    $BC

; låg nivå startar period
      CLR    PWPOL

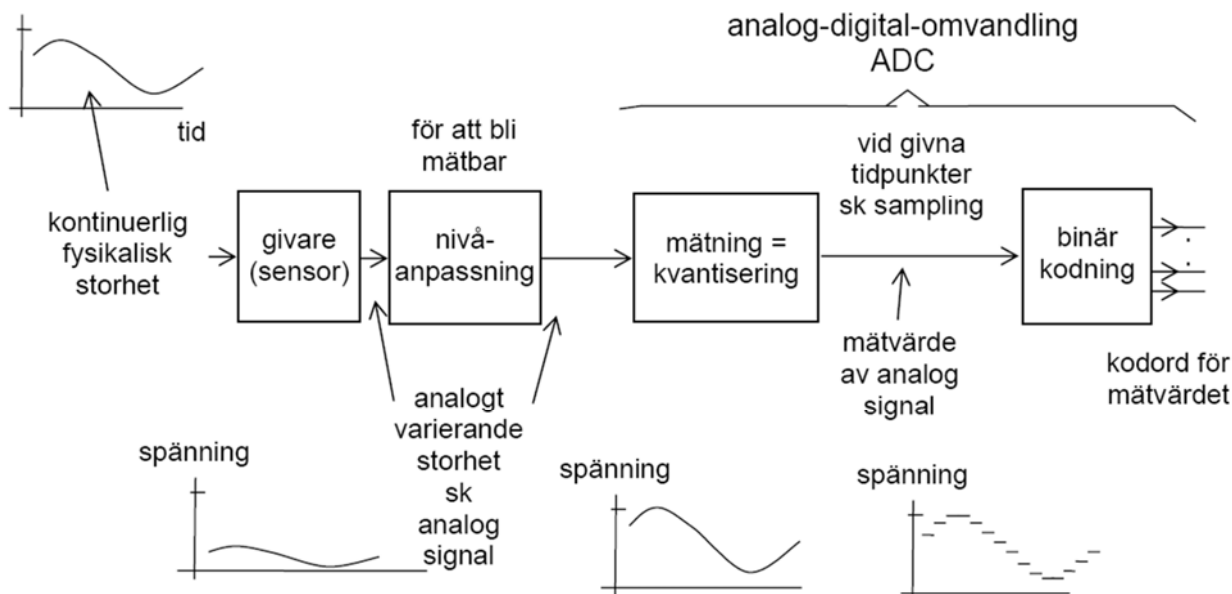
; c:a 4 ms periodtid
      MOVB  #$77, PWPCLCK

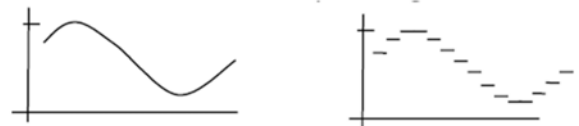
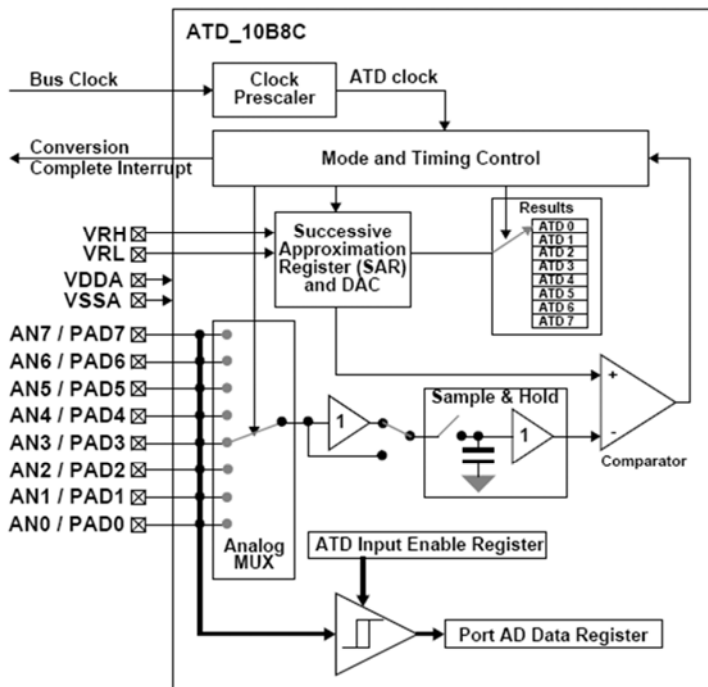
; pwm kanal 0
      MOVB  #$FF, PWPPER0

; 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

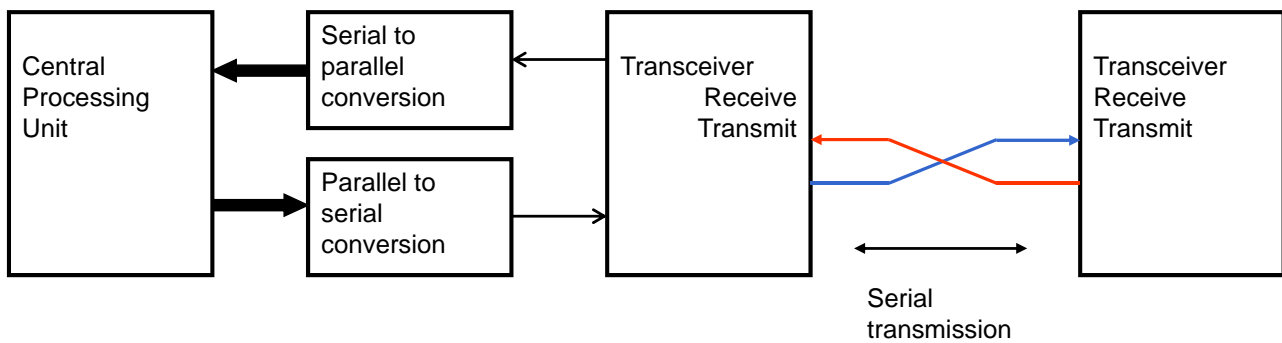
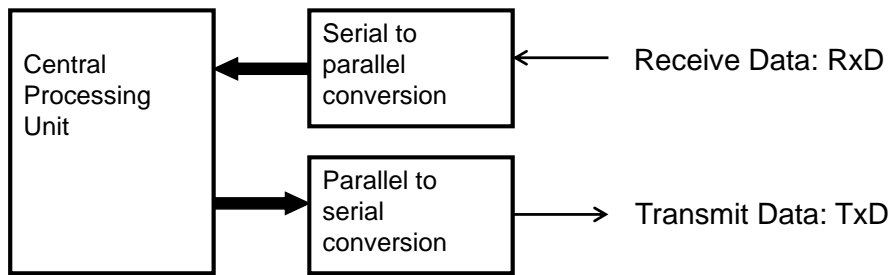
; normal mode
        MOVB    #C0,ATDCTL2

; Vänta tills omvandling klar
wAD:
        BRCLR   ATDSTAT0,#80,wAD

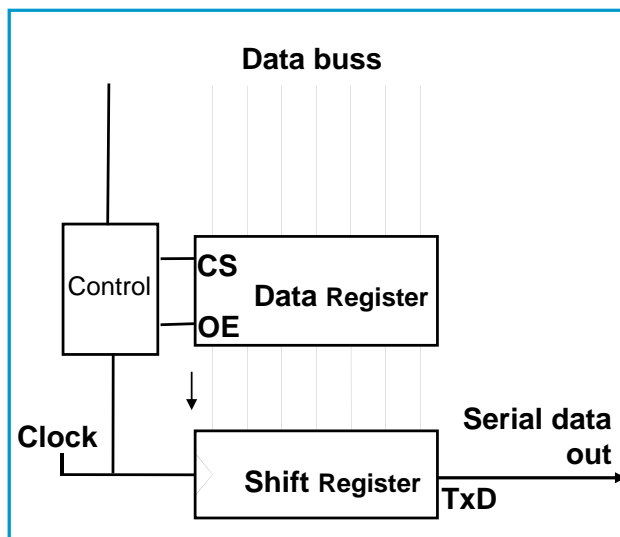
; När resultat färdigt, läs
        LDAB    ATD0DR0L

...
    
```

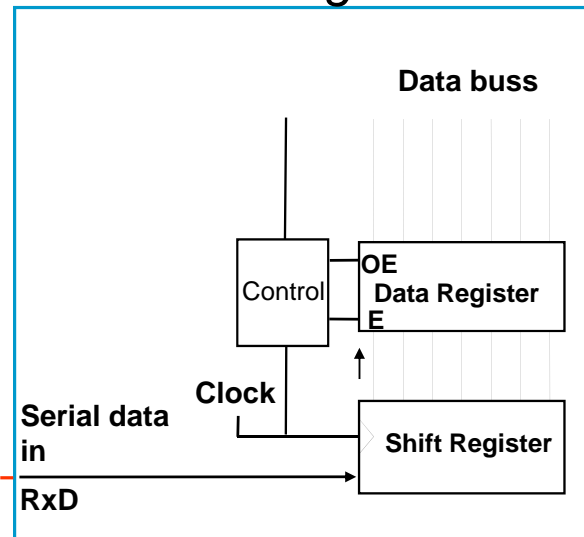
Seriekommunikation, SCI



Sändare



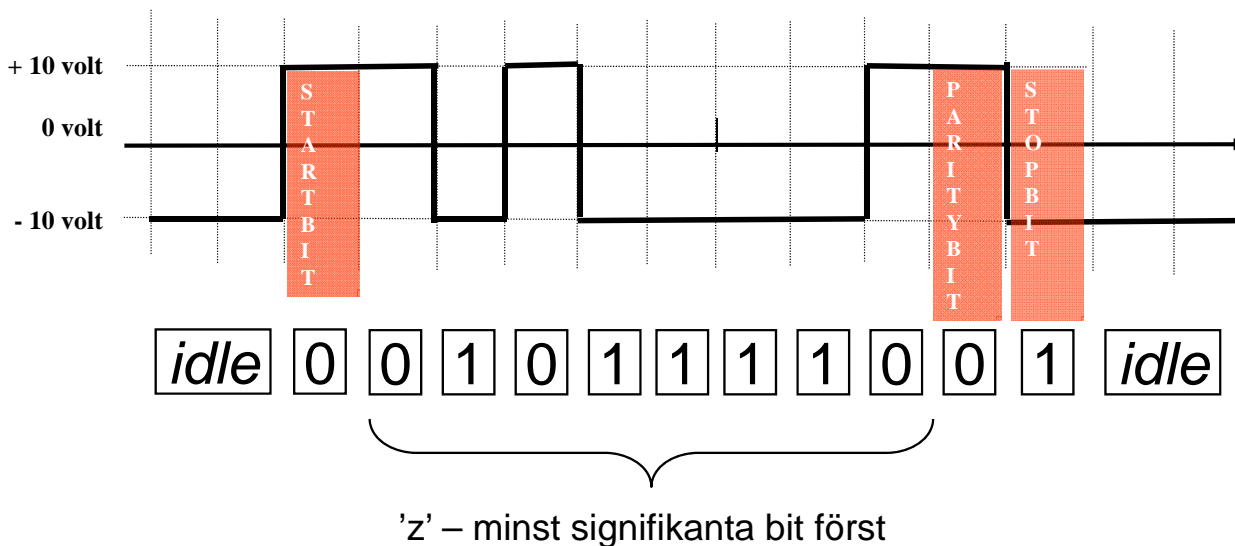
Mottagare



Sändare och mottagares klockor går i samma takt

RS232 – överföring av tecknet 'z'

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



Initiering, "busy-wait"

Basadress = \$C8

Algorithm:
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

Algorithm:
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		

```

SCIOSR1:      EQU      $CC      ; SCI 0 status-register 1.
SCIODRL:      EQU      $CF      ; SCI 0 data-register låg byte.
; Bitdefinitioner, statusregister
TDRE:         EQU      $80      ; Transmit data register empty status bit.
    
```

Läs tecken från SCI

Algorithm:
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		

```

SCIOSR1:      EQU      $CC      ; SCI 0 status-register 1.
SCIODRL:      EQU      $CF      ; SCI 0 data-register låg byte.
; Bitdefinitioner, statusregister
RDRF:         EQU      $20      ; Receive data register full status bit.
    
```

Bestämna Baudrate-värde

exempel: 9600 baud

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

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

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

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...