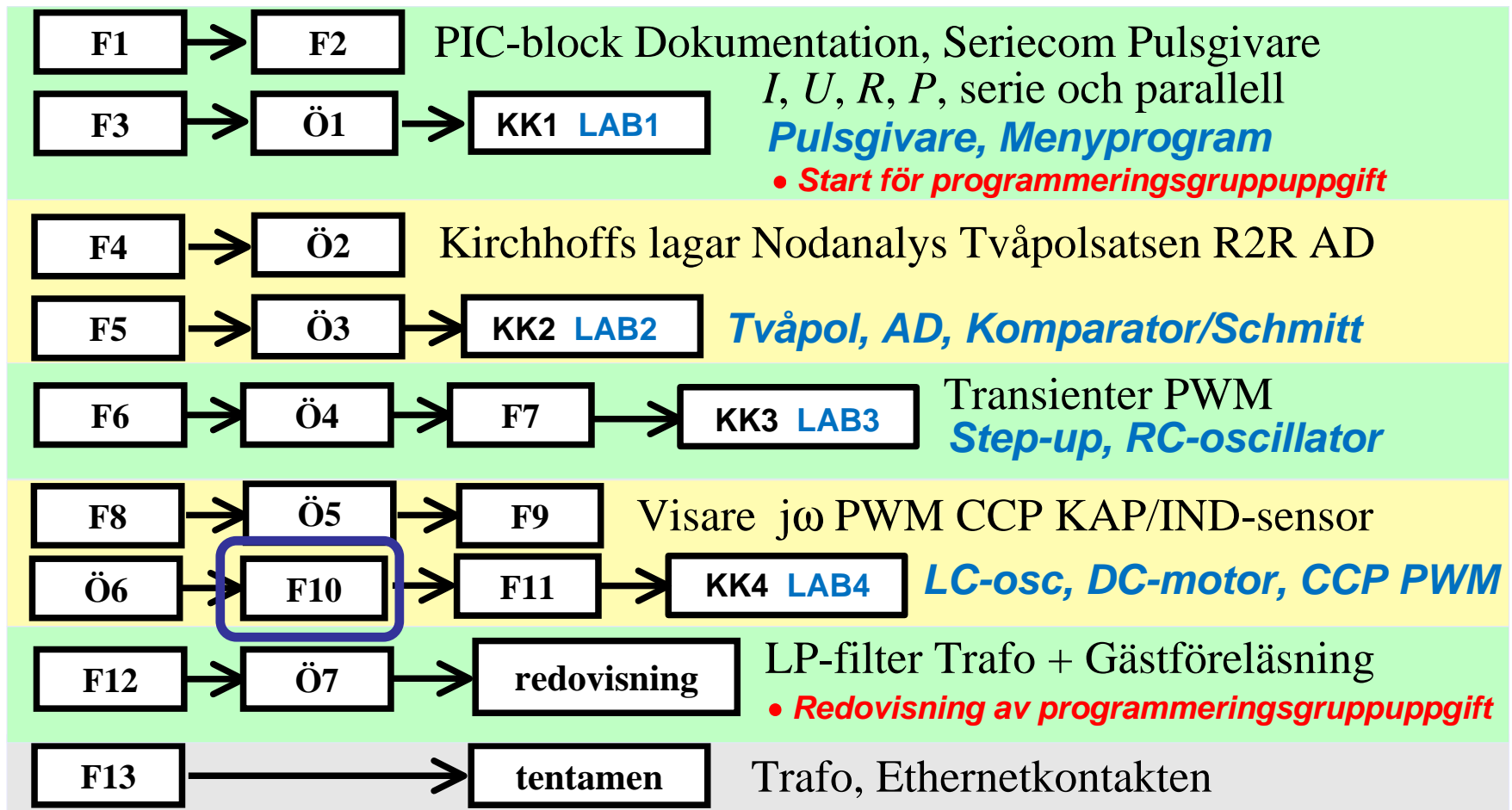
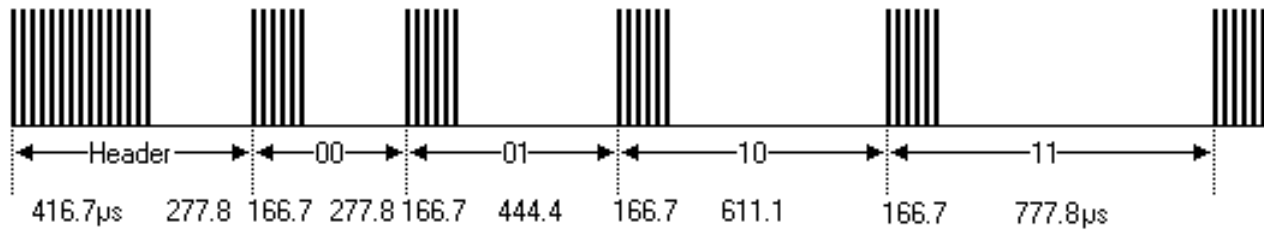
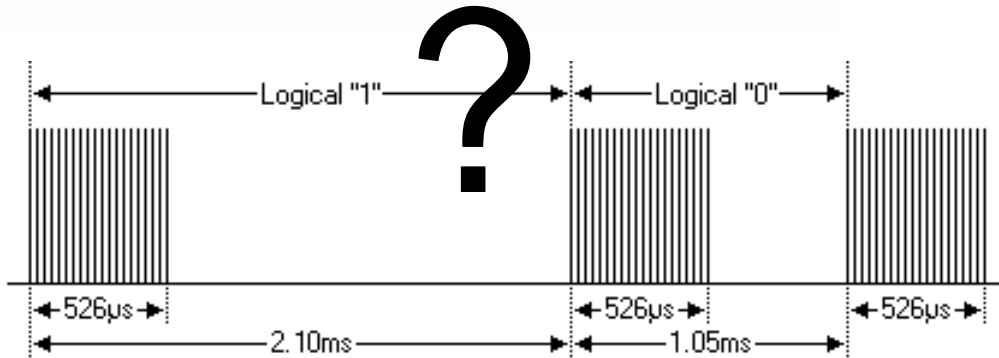
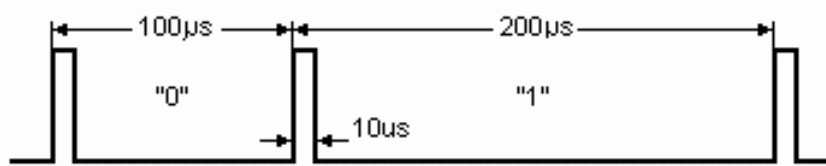


# IE1206 Inbyggd Elektronik



# Hur mäter man pulser?



*Att mäta olika digitala pulser är en av PIC-processorns huvuduppgifter*

William Sandqvist [william@kth.se](mailto:william@kth.se)

- *Pulser från otaliga sensorer*

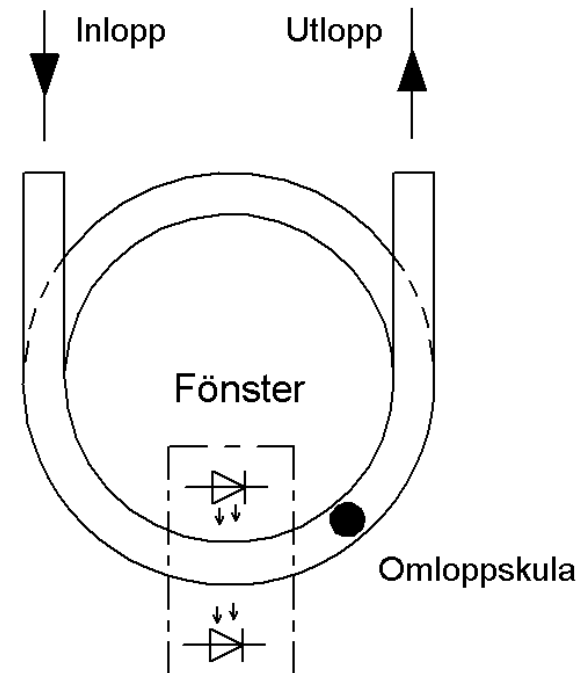


Otaliga givare har som utstorhet någon form av digitala pulser: antal, tid, periodtid, frekvens, dutycycle ...

*Här följer några exempel:*

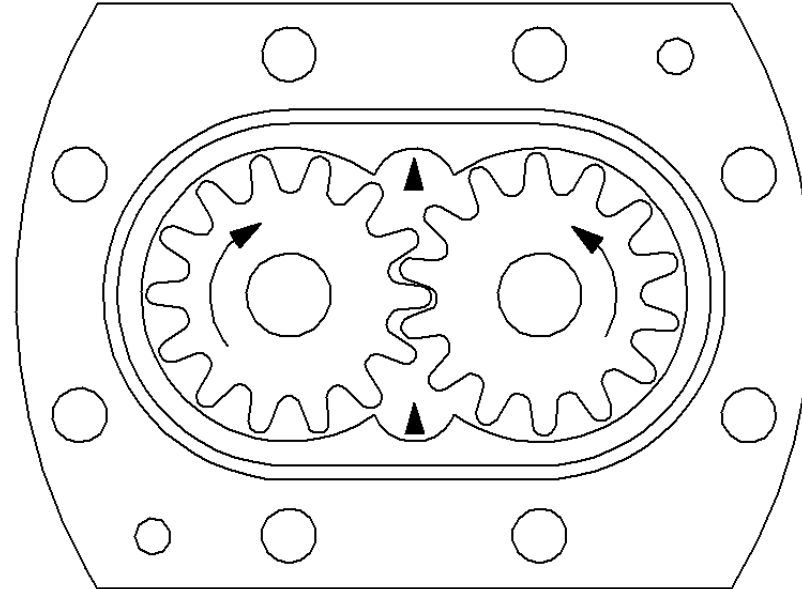
**Medströmsmätare.** Omloppskulan strömmar med vätskan och passerar fotodioden varje varv.

Givaren används som bensinmätare, **antalet** pulser från fotodioden summeras som förbrukat bränsle.

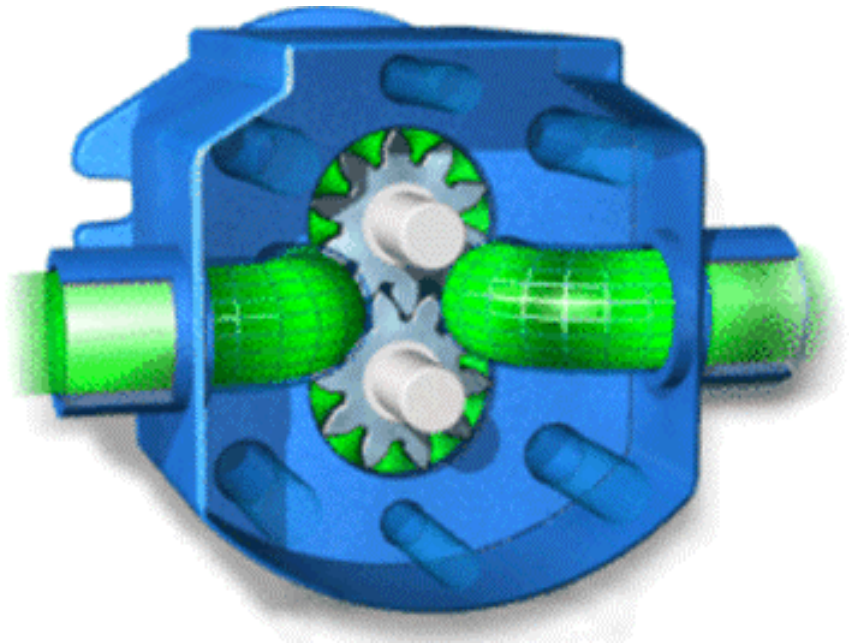




*ex. Antal*

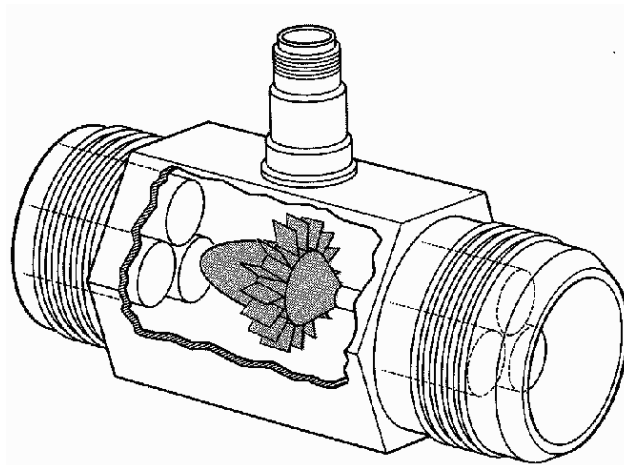


**Kugghjulsmätare.** Vätska förflyttas i "kuggluckorna". Inget läckage, kan mäta mycket *små* vätskemängder (upplösningen är volymen i en kugglucka). Används som bensinmätare på bensinstationerna. **Antalet** vridningsvarv är ett mått på vätskemängden.



William Sandqvist [william@kth.se](mailto:william@kth.se)

# Propeller och turbin -mätare

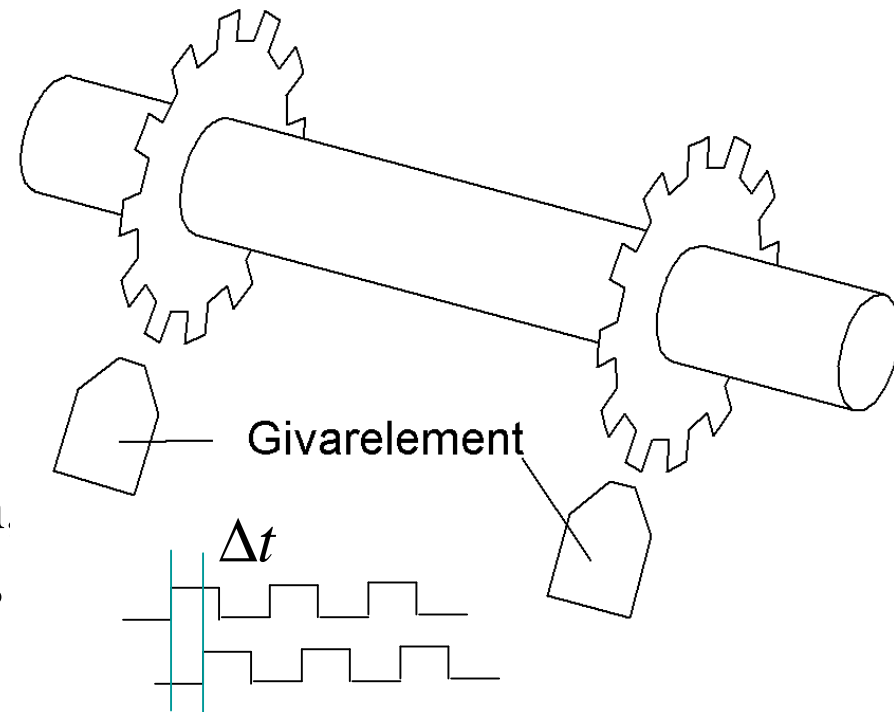


**Pulsfrekvensen** är proportionell mot strömningshastigheten.

## *ex. Pulstid*

### **Momentmätare.**

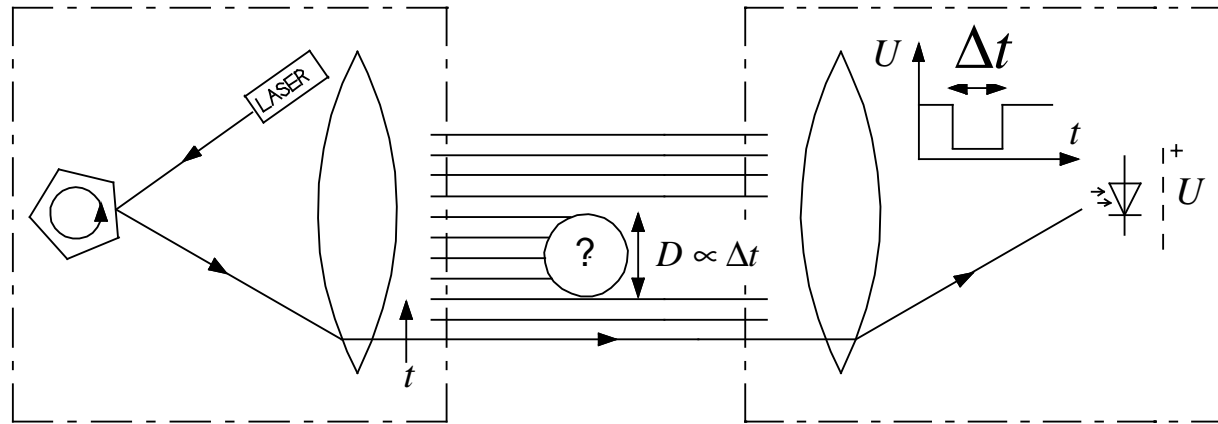
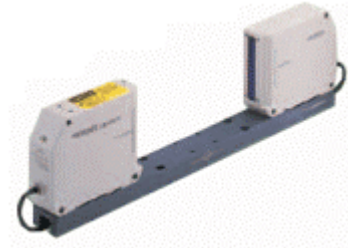
Då ett vridmoment överförs med en roterande axel vrids (skjuvas) denna så att kugg-hjulen vrids i förhållande till varandra. Det blir en mätbar *tids-skillnad* mellan pulserna från givar-elementen som känner av kuggtopparnas passage.



Vridmomentet kan räknas fram utifrån denna tids-skillnad med kännedom om axelns vridstyvhet.



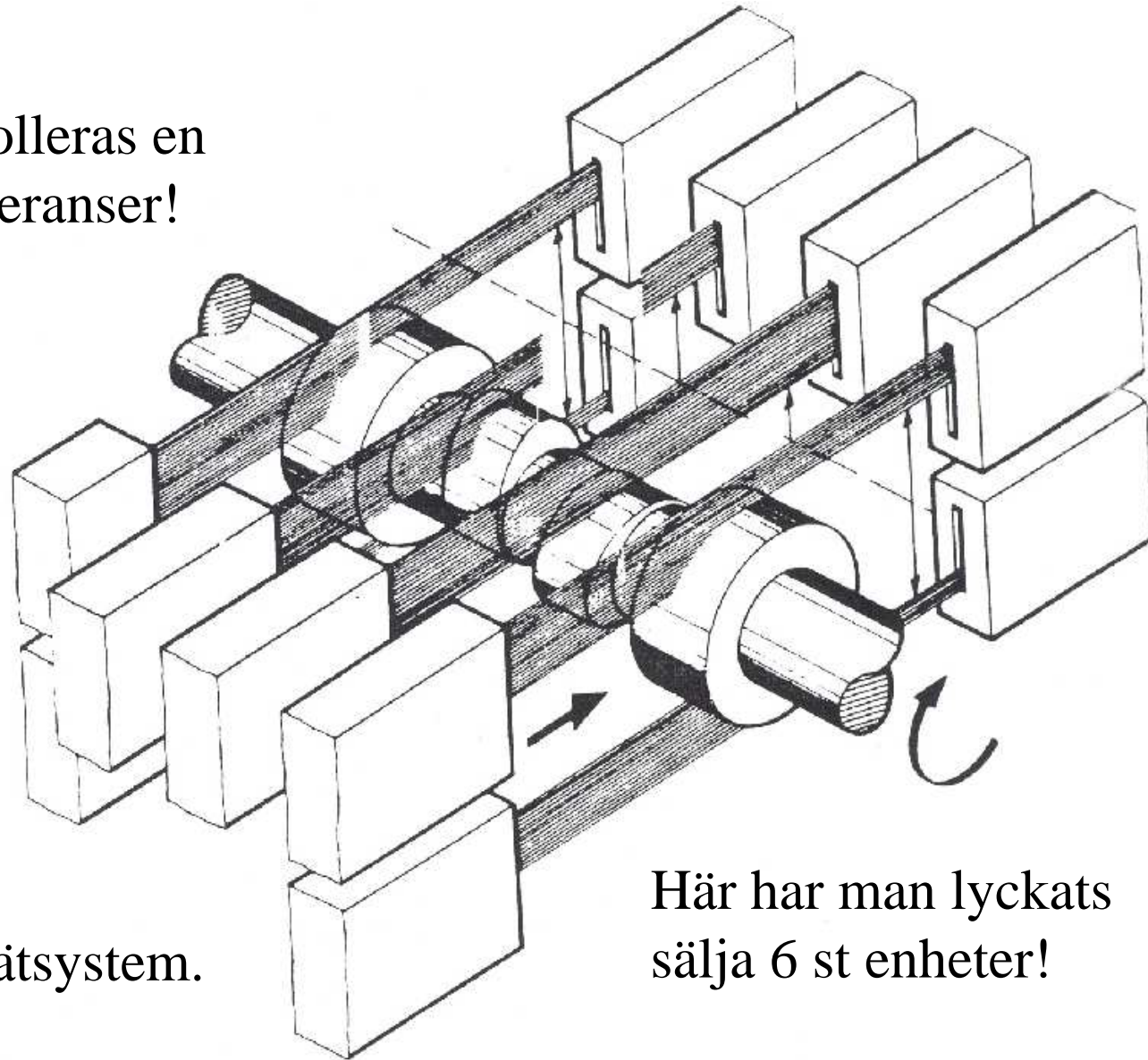
# ex. Pulstid



**Laser Scan Micrometer.** Mätobjektets diameter skuggar laserljuset. En upplösning om  $1 \mu\text{m}$  är möjlig.

$$D \propto \Delta t$$

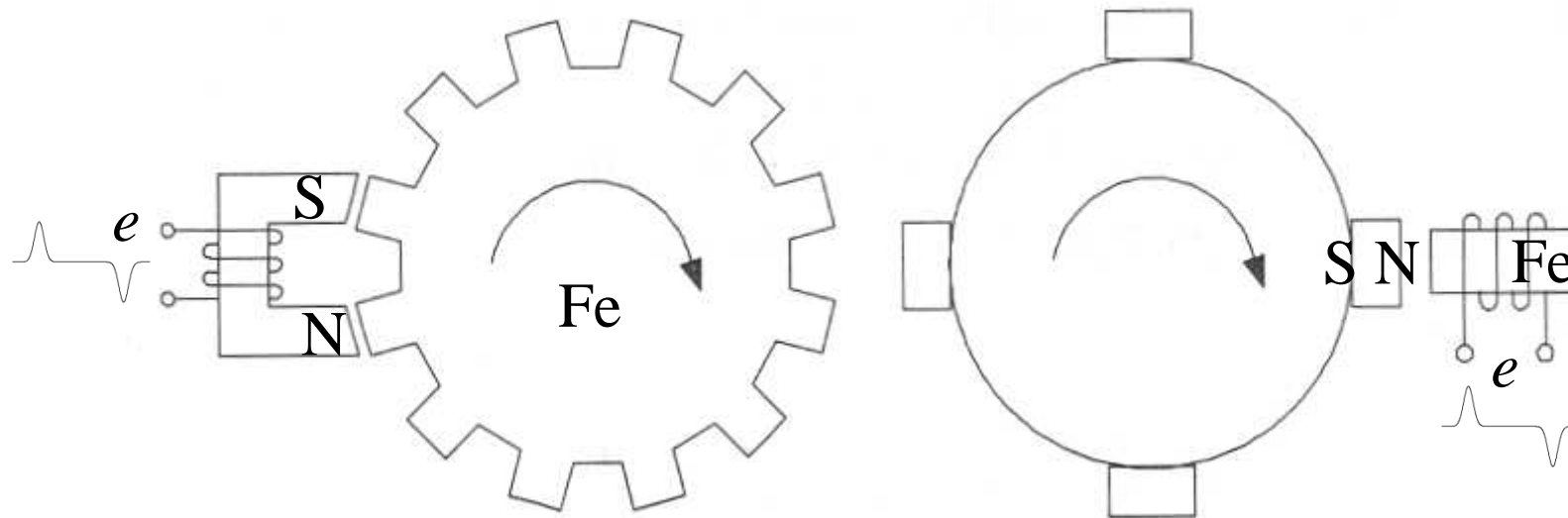
Så här kontrolleras en kamaxels toleranser!



Försäljarens  
önskedröm:  
Datoriserat mätsystem.

Här har man lyckats  
sälja 6 st enheter!

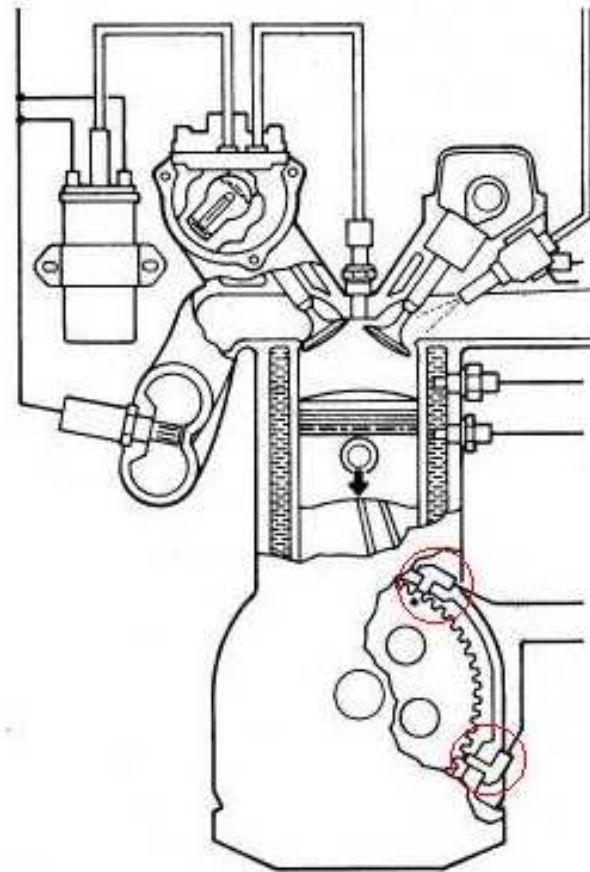
# Induktiv pulsgivare



Vissa krav ställs på de magnetiska egenskaperna.

$$e \propto \frac{\Delta\Phi}{\Delta t}$$

# Styrning av förbränningsmotor



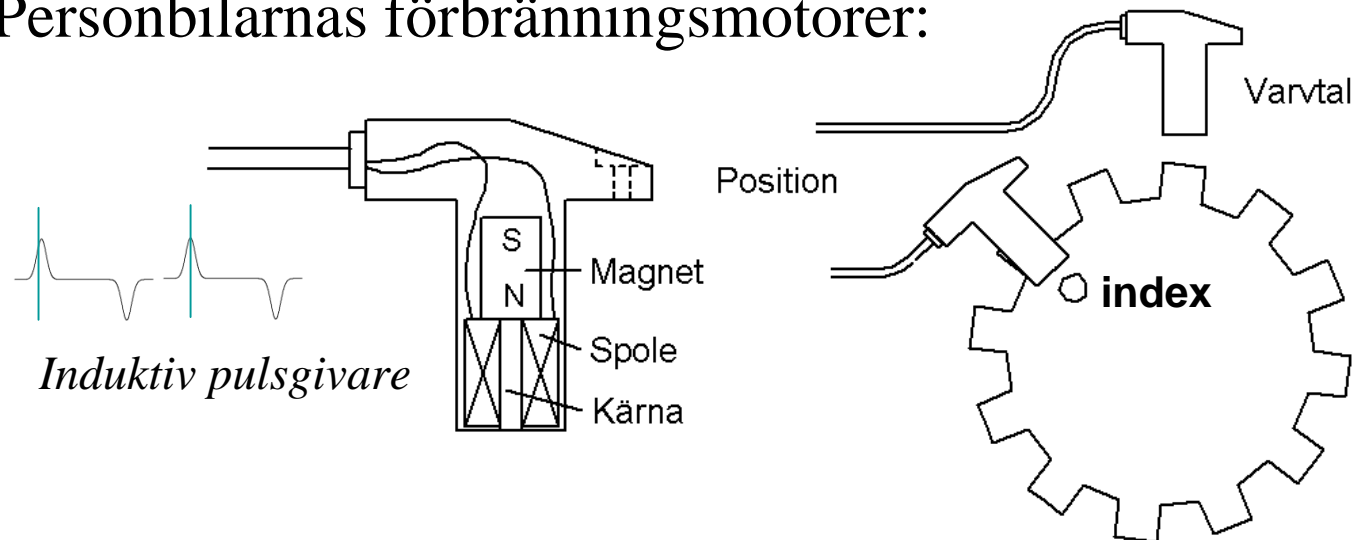
*Induktiv pulsgivare*



*Induktiv pulsgivare*

# *ex. Pulstid, antal*

Personbilarnas förbränningsmotorer:

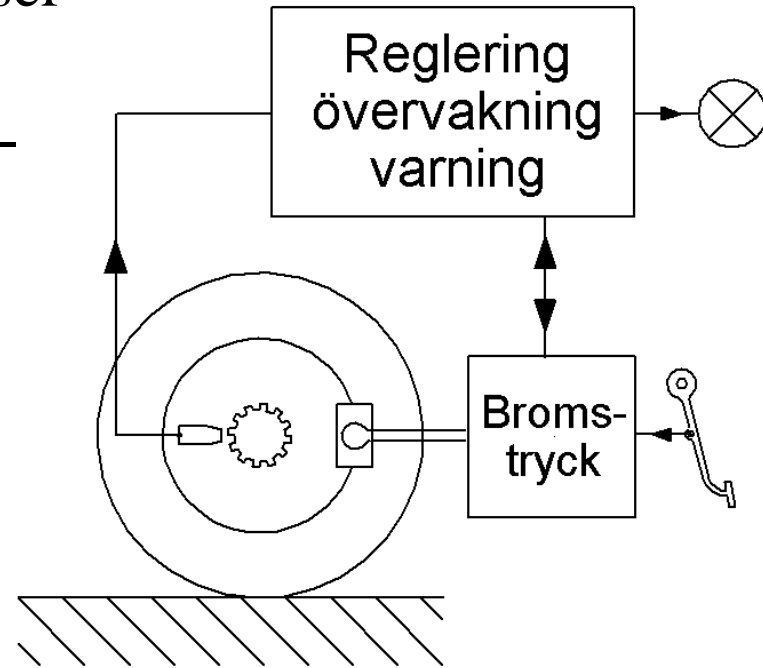
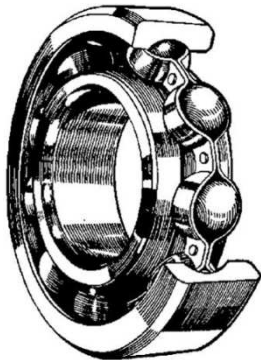


Varvtal och vinkel mäts mot ett kugghjul ("startkransen") med en induktiv pulsgivare. Givaren ger en **puls** för varje kuggtopp. Varvtalet räknas ut från **pulstiden** mellan två kuggtoppar.

Ett "indexmärke" markerar vinkeln  $0^\circ$ .  
(Alternativt kan en kugge "saknas" vid  $0^\circ$ ).

# ex. *Låg pulsfrekvens*

**ABS-bromsar.** När hjulet ”låser sig”, släpper det greppet mot marken. Detta upptäcker ABS-systemet och ”minskar” då på bromstrycket.



En pulsgivare integrerad i hjullagret ger en pulsfrekvens som är proportionell mot hjulets varvtal. ”Låst” hjul innebär **låg pulsfrekvens**.

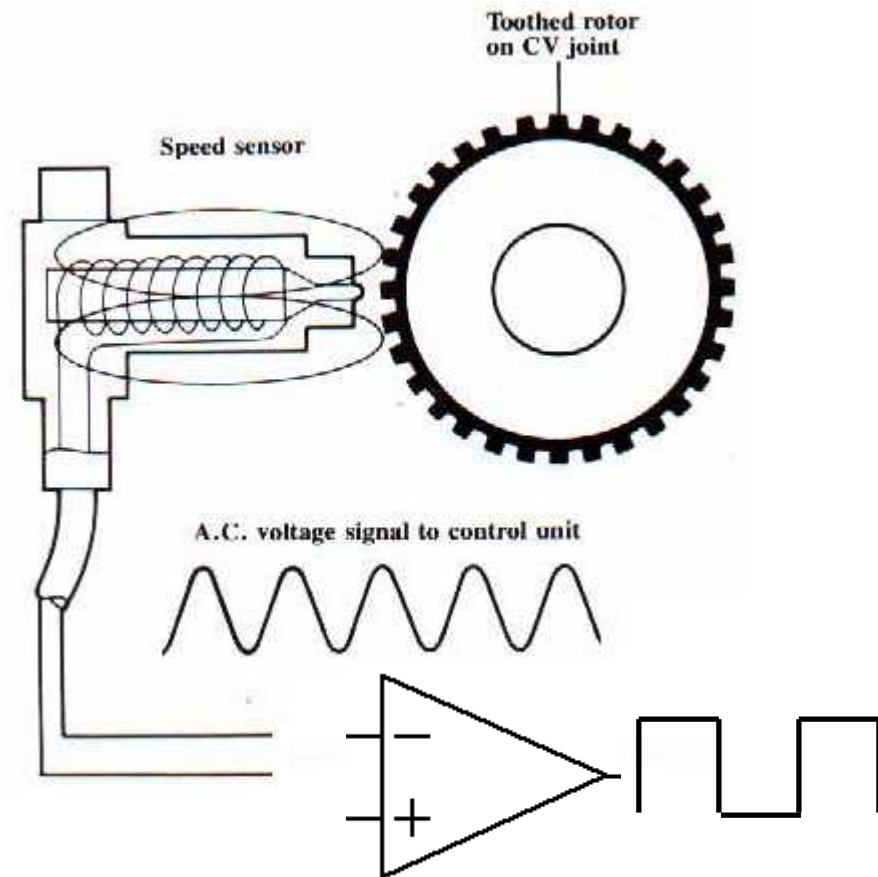
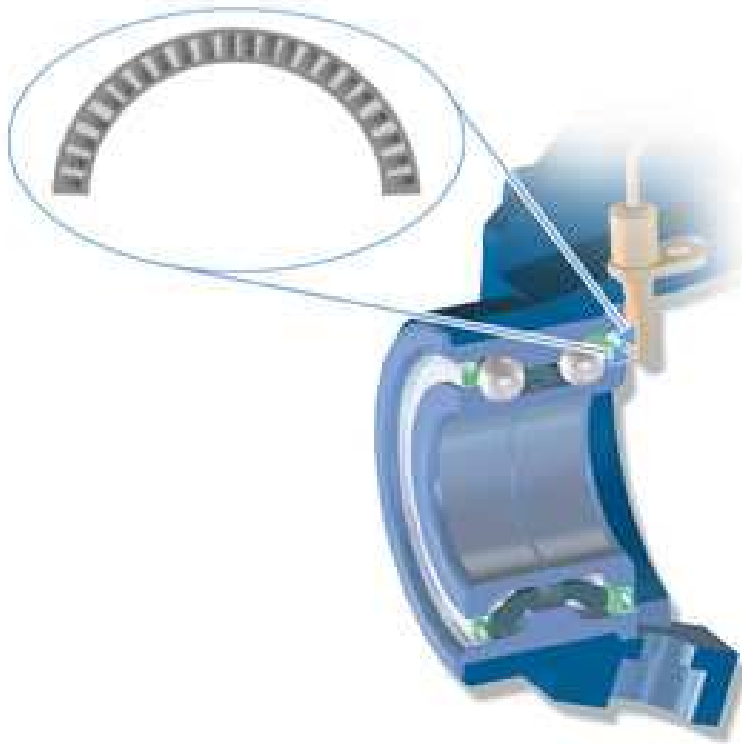
# Givare integreras numera ofta i *rena* verkstadsprodukter



Hjullagerenhet med integrerad ABS-sensor. SKF.

# Induktiv ABS-givare (spole)

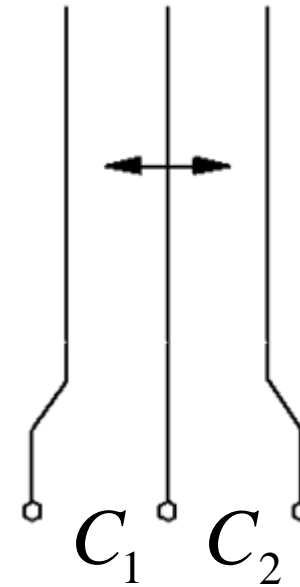
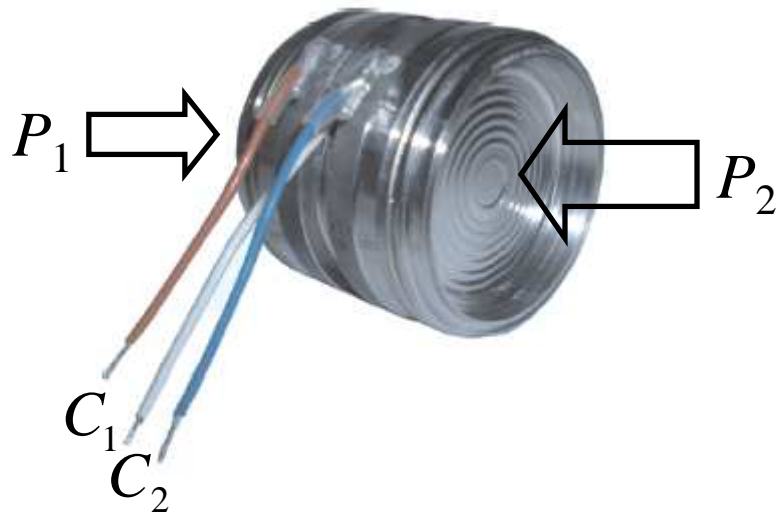
Det tandade metallhjulet är **inbakat** i kullagrets plast-tätning!  
(ex. SKF)





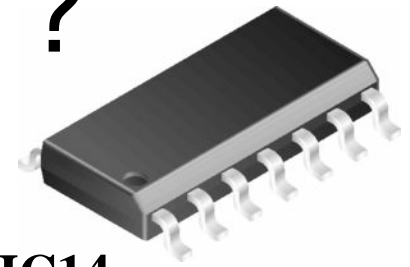
William Sandqvist [william@kth.se](mailto:william@kth.se)

# *f* Kapacitiv tryckgivare



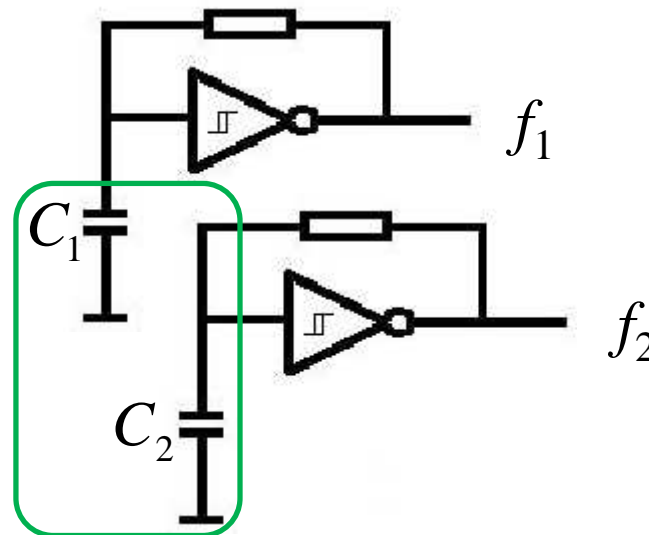
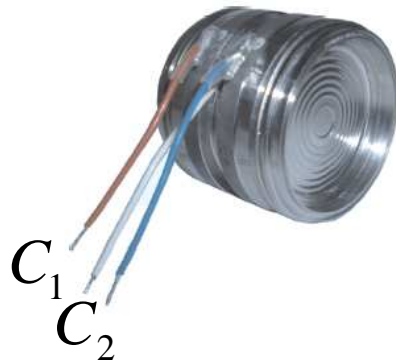
*Differentialkondensator för tryckskillnad*

# Enkel mätutrustning ?



74HC14

Sex CMOS Schmitt-trigger inverter



$$\frac{f_1}{f_2}$$

Två oscillatorer byggs *nära* differentialkondensatorerna. Frekvenserna  $f_1$  och  $f_2$  mäts. Bildar man **kvoten** mellan frekvenserna får man en storhet där allt som påverkat givarna *lika* undertrycks (= kan förkortas bort).

William Sandqvist [william@kth.se](mailto:william@kth.se)

# Noggrann mätning av $f$

Mätning av frekvens kan ske mycket noggrant.  
Mycket noggrannare än andra mätningar.

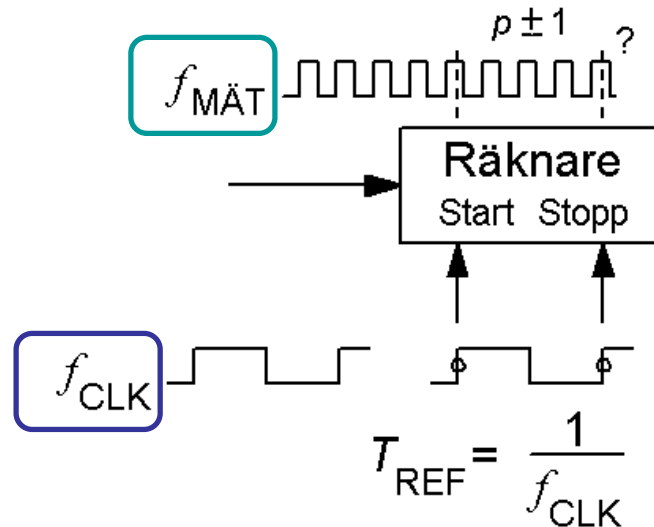
Sensorerna avger pulser av mycket varierande utseende och frekvenser – det finns inte en enda mätmetod som kan täcka alla mätfall.

PIC-processorn har **tre** olika **TIMER**'s och en CCP-enhet för detta. Processorklockan kan genereras med åtta olika metoder.

# Frekvensmätning

Hög  
frekvens

$$f_{\text{MÄT}} > f_{\text{CLK}}$$



Kvantisering.

*Räknaren räknar bara  
"hela" pulser.*

$$f = \frac{p \pm 1}{T_{\text{REF}}}$$

$$f = (p \pm 1) \cdot f_{\text{CLK}}$$

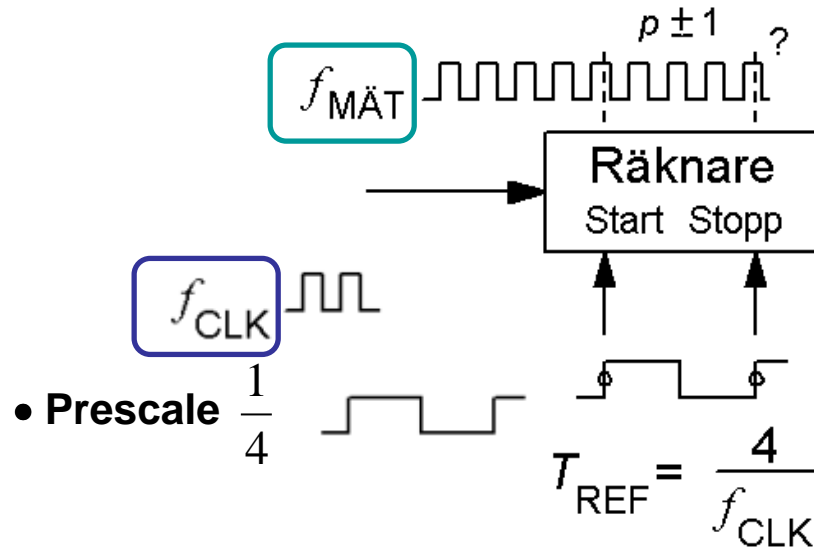
- **Direkt frekvensmätning**

Antalet positiva flanker  $p$  under *en* period av  $T_{\text{REF}}$  räknas ( $T_{\text{REF}} = 1/f_{\text{CLK}}$ ).

Hög mätfrekvens  $f_{\text{MÄT}}$  tillsammans med *lång* mättid  $T_{\text{REF}}$  minimerar inverkan av **kvantiseringsfelet**.

# Frekvensmätning

Lägre  
frekvens  
 $f_{\text{MÄT}} > \frac{1}{4} f_{\text{CLK}}$



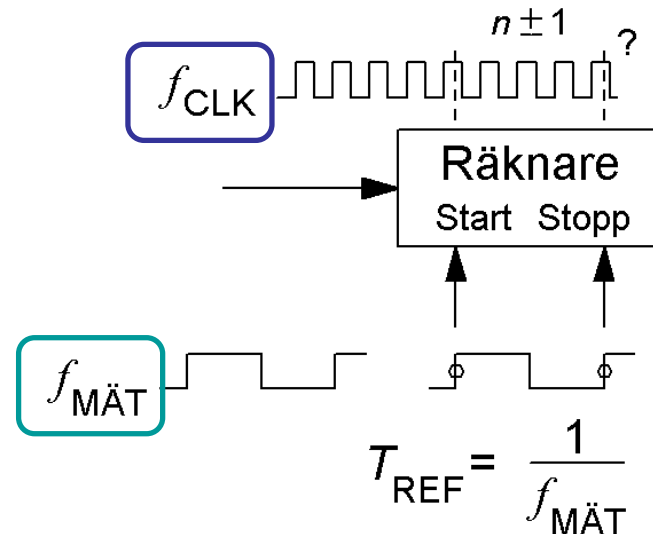
$$T_{\text{REF}} = \frac{1}{\frac{1}{4} \cdot f_{\text{CLK}}} = \frac{4}{f_{\text{CLK}}}$$
$$f = \frac{p \pm 1}{T_{\text{REF}}} = \frac{(p \pm 1) \cdot f_{\text{CLK}}}{4}$$

*Lägre* mätfrekvens kräver att mättiden förlängs genom att man delar ned referens-frekvensen  $f_{\text{CLK}}$  med en **prescaler**.

# Periodtidmätning

Låg  
frekvens

$$f_{\text{MÄT}} < f_{\text{CLK}}$$



$$f = \frac{f_{\text{CLK}}}{(n \pm 1)}$$

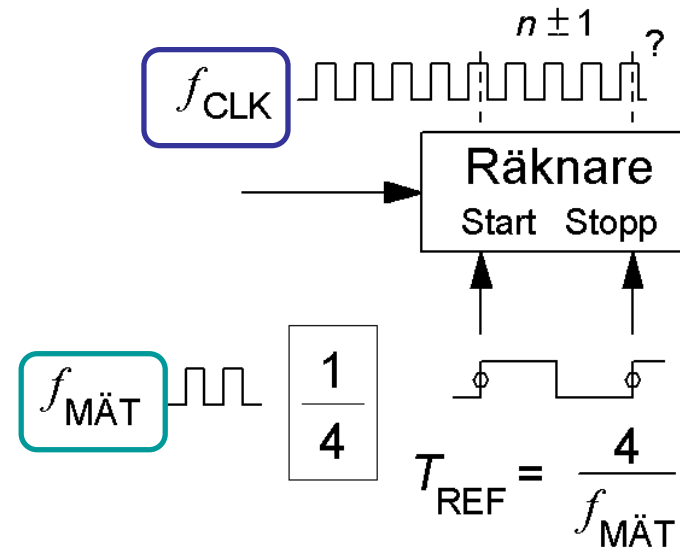
Alternativt kan man vid låg mätfrekvens göra en **periodtidmätning**. Detta är indirekt frekvensmätning. Mätfrekvensen får man genom att *invertera* räknevärdet.

Under en period av mätsignalen räknar man  $n$  klockpulser.



# Multiperiodtidmätning

Högre  
frekvens  
 $\frac{1}{4} f_{\text{MÄT}} < f_{\text{CLK}}$



$$f = k \cdot \frac{f_{\text{CLK}}}{(n \pm 1)}$$

Högre mätfrekvenser kan mätas med **multiperiodtidmätning**. Mätfrekvensen delas då ned med en faktor  $k$  innan mätningen (registrera var 4:e eller var 16:e flank).

- PIC-processorn är förberedd för alla dessa olika mätmetoder. (Och många fler ... )

William Sandqvist [william@kth.se](mailto:william@kth.se)

# Klockfrekvensens noggrannhet

Förutom kvantiseringsfel, dvs. att man bara räknar hela pulser, kommer man alltid att ha ett **relativt fel** som är lika stort som referensfrekvensens fel.

**Ex.** Armbandsur kräver kristall. Kristaller har typiskt fel  $\Delta f \pm 20$  ppM (parts per million).  $f = 4 \text{ MHz} \pm 80 \text{ Hz}$ .

Önskemål: uret får inte dra sig mer än 10 sek/månad.  
 $10\text{s}/(30[\text{dgr}] \cdot 24[\text{tim}] \cdot 60[\text{min}] \cdot 60[\text{sek}]) = 25 \text{ ppM}$ .



# Klockfrekvensens noggrannhet

**Ex.** Tidtagarur som ska användas till ett 800 m lopp.

( 2 minuters total mättid räcker troligen )

Önskemål: upplösning 0,01 sek.

$1/(2[\text{min}] \cdot 60[\text{sek}] \cdot 100) = 1 \text{ ‰}$ .



En RC-oscillator har typiskt 5% fel, om den inte trimmas.

( R 1%, men C sällan bättre än 5% )

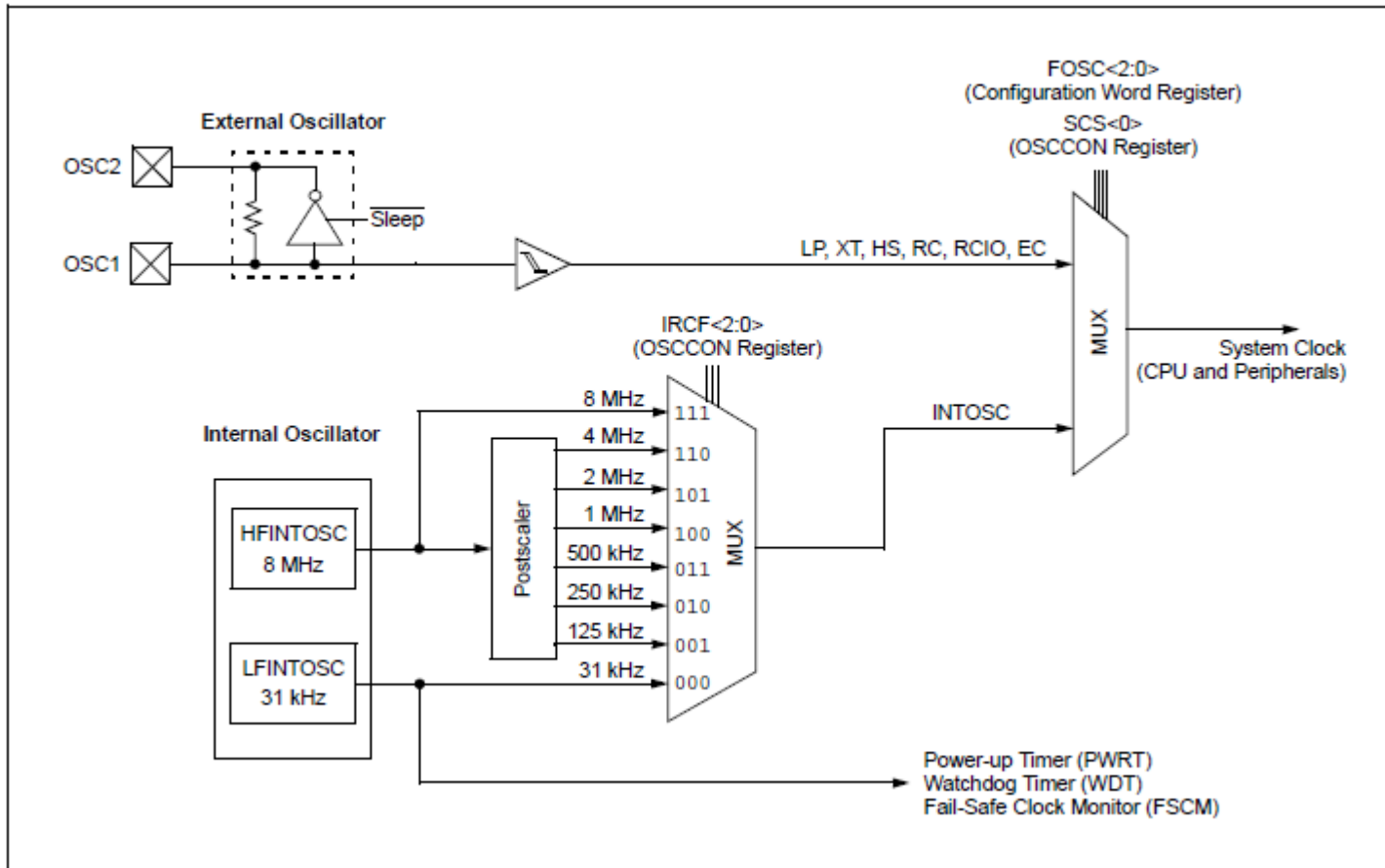
PIC16F690-processorns inre RC-oscillator är **fabrikstrimmad** till  $\pm 1\%$ .

*Detta räcker inte ... men den går att fintrimma!*



# PIC-processorns klockmodul

FIGURE 3-1: SIMPLIFIED PIC® MCU CLOCK SOURCE BLOCK DIAGRAM

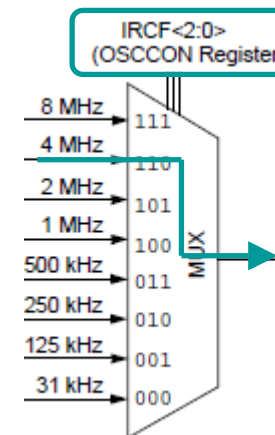


# PIC-processorns klockmodul

REGISTER 3-1: OSCCON: OSCILLATOR CONTROL REGISTER

U-0	R/W-1	R/W-1	R/W-0	R-1	R-0	R-0	R/W-0
—	IRCF2	IRCF1	IRCF0	OSTS <sup>(1)</sup>	HTS	LTS	SCS
bit 7							bit 0

- Vid laborationerna använder vi standardinställningen, default, 4 MHz – som gör det enkelt att beräkna exekveringstid.



REGISTER 3-2: OSCTUNE: OSCILLATOR TUNING REGISTER

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	TUN4	TUN3	TUN2	TUN1	TUN0
bit 7							bit 0

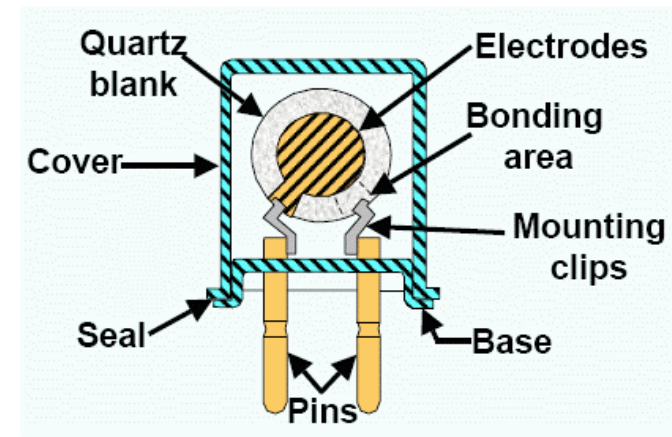
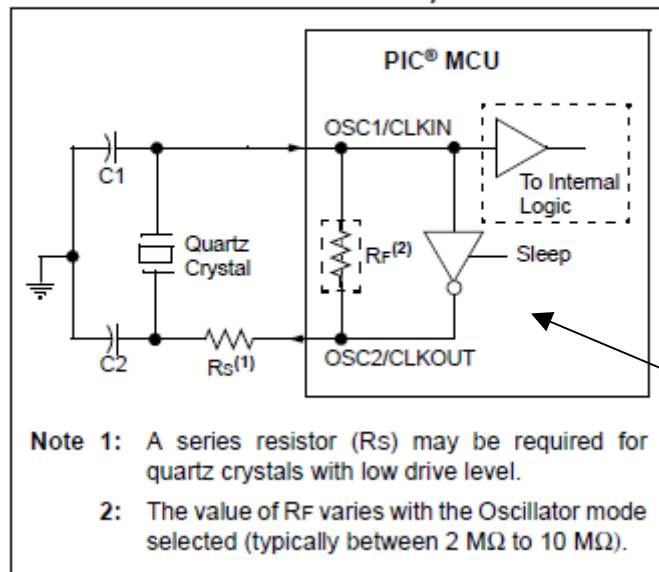
10000(min) – 00000 (fabrikstrim) – 01111(max)

- Om man har möjlighet att ”trimma själv” så kan den fabrikstrimmade frekvensen justeras i  $\pm 16$  små steg till  $\approx \pm 0,5\%$ . *Nu tillräckligt för tidtagaruret!*

# Yttre kristall



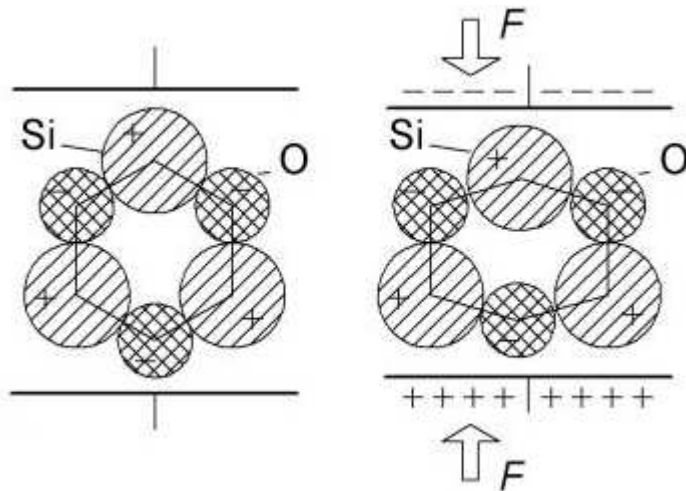
FIGURE 3-3: QUARTZ CRYSTAL OPERATION (LP, XT OR HS MODE)



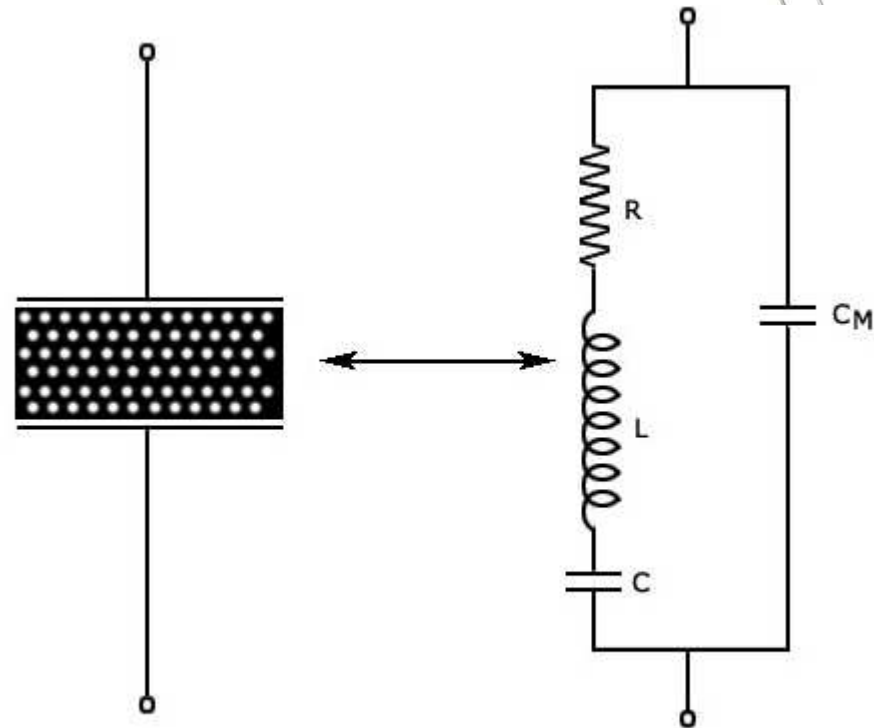
Samma slags krets som till kursens LC-oscillatorlab.

PIC-processorerne kan använda yttre kristall. C1 och C2 kan man ”skolka ifrån” på kopplingsdäck, men de är nödvändiga på ett mönsterkort.

# Piezoelektrisk kristall



Tillför man ström (laddning) till en ”bergkristall” komprimeras den, när den sedan ”fjädrar” tillbaka avger den ström.



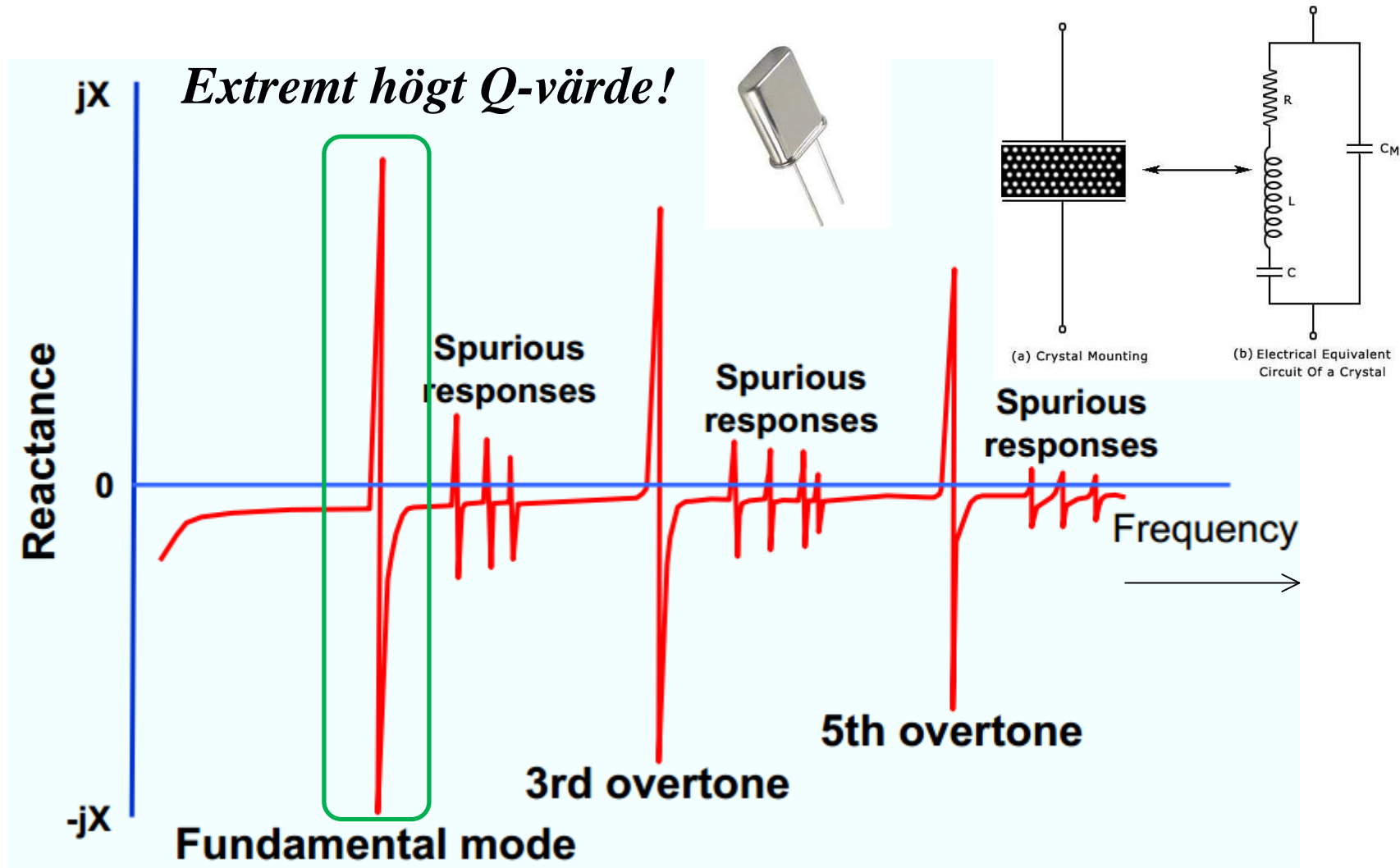
(a) Crystal Mounting

(b) Electrical Equivalent Circuit Of a Crystal

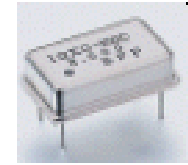
- Elektriskt kan kristallen jämföras med en resonanskrets – med **extremt** högt Q-värde.



# Piezoelektrisk kristall



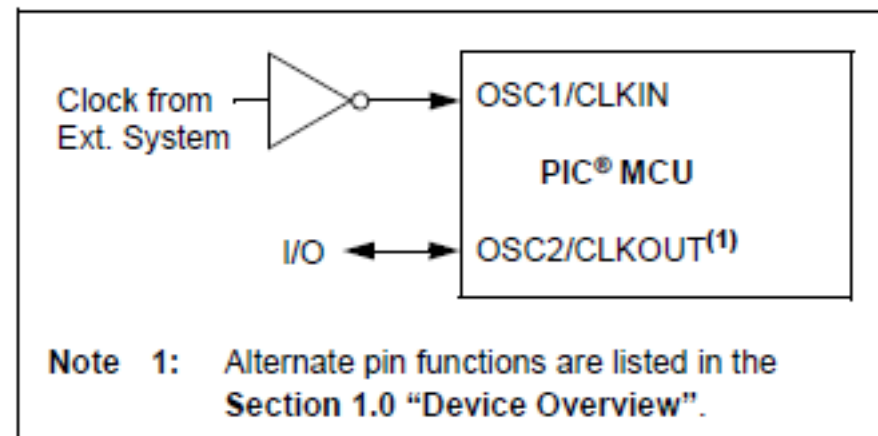
# Yttre klockfrekvenssignal



PIC-processorererna kan använda *yttre* klockfrekvenssignal.

Har man tillgång till en **exakt frekvens** kan PIC-processorn bli lika exakt. ( Bilden visar en sådan extern klockmodul med oscillator och kristall ”allt i ett” ).

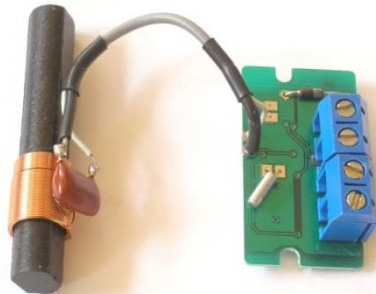
**FIGURE 3-2: EXTERNAL CLOCK (EC) MODE OPERATION**



# Atom-normal?

Radiokontrollerade klockor från tex. Claes Ohlsson & co är **låsta** till en atomnormal i tyskland.

*Det kan faktiskt gå att få tag på extremt noggrann referensfrekvens till lågt pris!*



En sådan klockmodul ger en puls per sekund (med undantag för sekund nr 60). En så kallad PPS-signal.

# Låg klockfrekvens $RC$

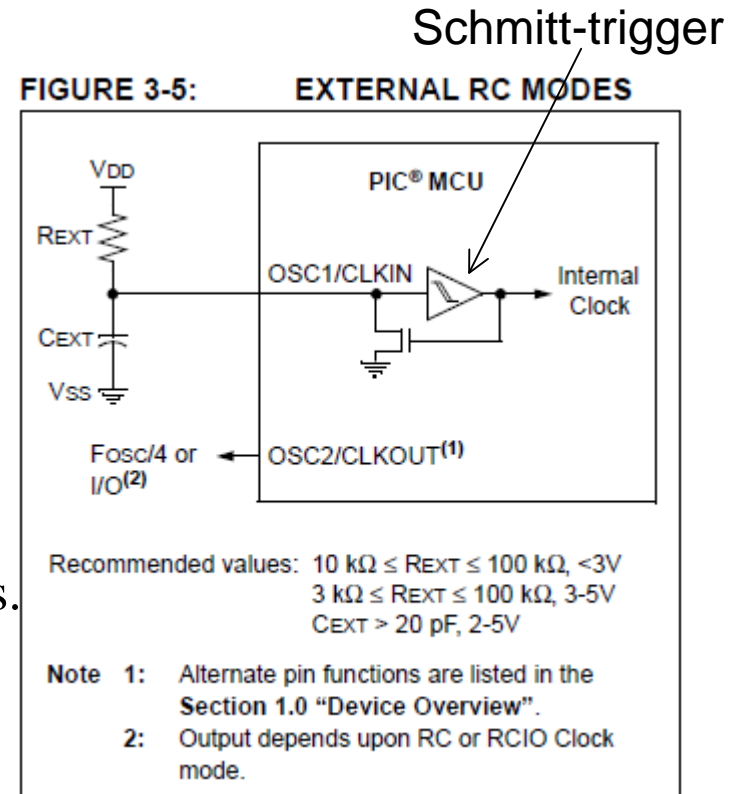
När frekvensnoggrannheten inte har så stor betydelse

– yttre RC-krets.

Datainsamling av ett mätvärde om dagen kräver ingen hög klockfrekvens. Man kan sedan ändra/höja klockfrekvensen i programmet när processorn ska avrapportera!

- Ju lägre klockfrekvens desto lägre strömförbrukning, och mindre risk för att PIC-processorn avger störningar

William Sandqvist william@kth.se

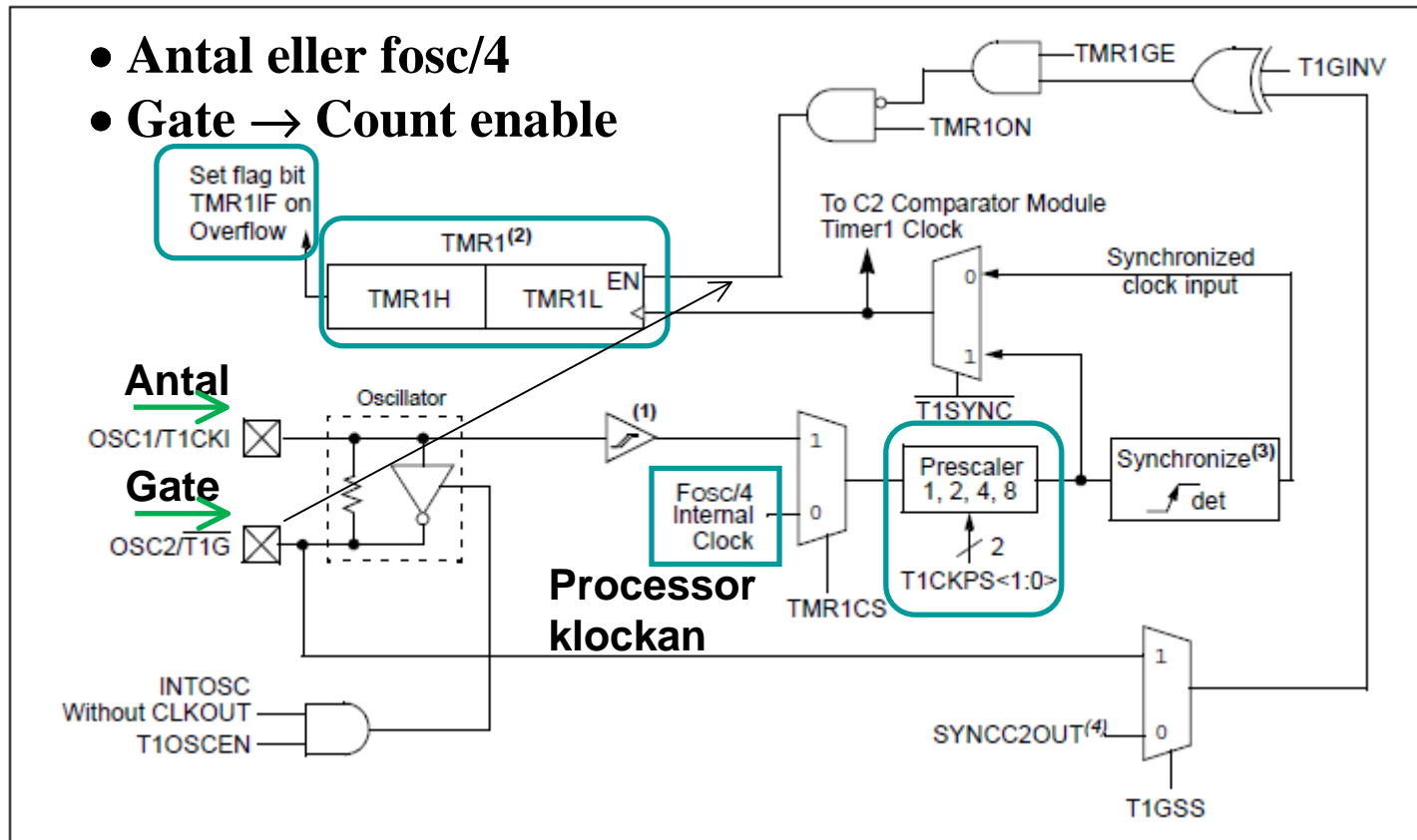


William Sandqvist [william@kth.se](mailto:william@kth.se)



# PIC 16F690 Timer1

FIGURE 6-1: TIMER1 BLOCK DIAGRAM



# PIC 16F690 Timer1

Timer1 är en 16-bitars timer/counter. Man når den genom två 8-bitars-register **TMR1H** och **TMR1L**. En flagga **TMR1IF** sätts om timern räknat runt. Måste 0-ställas ifall man vill veta om detta sker igen.

Timer1 kan använda en egen oscillator – för 32768 Hz tidtagarkristaller, eller processorklockan. Timer 1 har sedan en **Prescaler** { 1:1,1:2,1:4,1:8 }.

REGISTER 6-1: T1CON: TIMER 1 CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
T1GINV <sup>(1)</sup>	TMR1GE <sup>(2)</sup>	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{T1SYNC}$	TMR1CS	TMR1ON
bit 7		prescaler					bit 0
0	0			0	1	0	1

Inställning vid frekvesmättningslab.



# Hur läser man från en 16-bit "free running" Timer1?

Timer1 är en 16-bitarsräknare. Den måste läsas av som två 8-bitarstal, de 8 mest signifikanta bitarna **TMR1H** och de 8 minst signifikanta bitarna **TMR1L**. Detta kan vara ett problem eftersom Timern kan "slå runt" mellan avläsningarna av 8-bits talen. Följande kod visar det säkra sättet:

```
long unsigned int time; char TEMPH; char TEMPL;
TEMPH = TMR1H; TEMPL = TMR1L;
if (TEMPH == TMR1H) // Timer1 not rolled over = good value
{
    time = TEMPH*256;           OK direkt
    time += TEMPL;
}
else // Timer1 rolled over - no new rollover for some time
    // lots of time to read new good values
{
    time = TMR1H*256;
    time += TMR1L;           OK nu
}
```

# Hur skriver man till en 16-bit "free running" Timer1?

Det kan också vara problematiskt att *skriva* till en 16-bitarsräknare eftersom det måste ske som två 8-bitarstal. Detta är det säkra sättet:

```
TMR1L = 0; // clear low byte = no rollover for some time
TMR1H = 12345/256; // high byte of constant 12345
TMR1L = 12345%256; // low byte of constant 12345
```

Talet **12345** ryms i 16 bitar. Med heltalsdivision / och modulo % delas en konstant enkelt upp i två 8-bitarsdelar. Ett annat sätt är att använda hexadecimala konstanter:

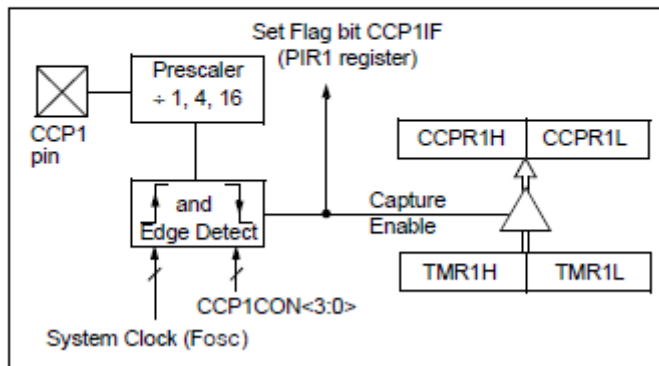
$$12345_{10} = 3039_{16} \quad \text{TMR1H} = 0x30 \quad \text{TMR1L} = 0x39$$

# CCP synkroniseringsregister

## ECCP-enheten, Enhanced Capture/Compare/(PWM)

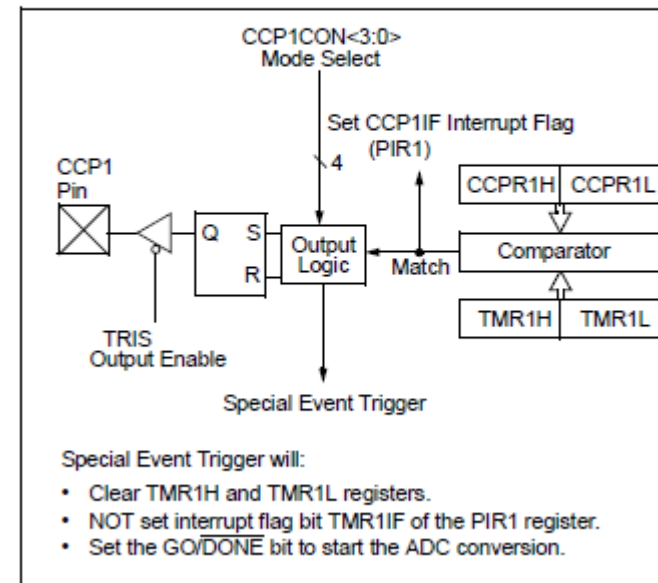
- Man kan undvika att skriva till och läsa från Timer1-registren – det finns *synkroniserade register* i ECCP-enheten för detta!

FIGURE 11-1: CAPTURE MODE OPERATION BLOCK DIAGRAM

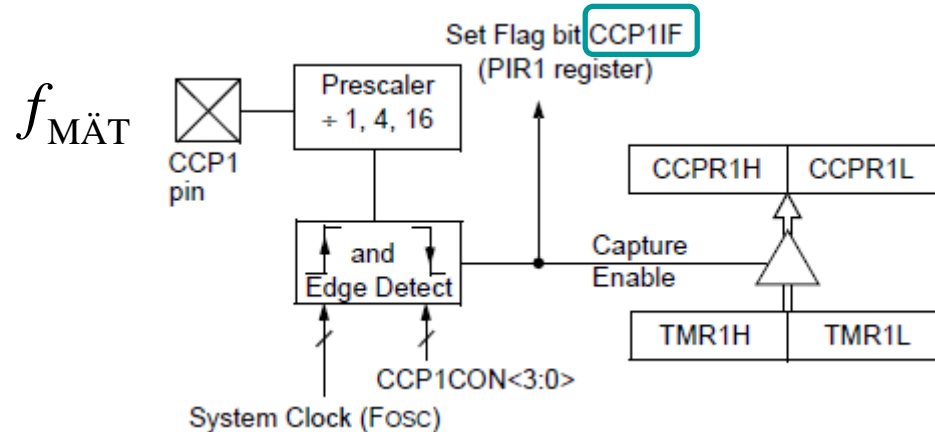


- **CCPR1H och CCPR1L**

FIGURE 11-2: COMPARE MODE OPERATION BLOCK DIAGRAM



# ECCP Capture modes

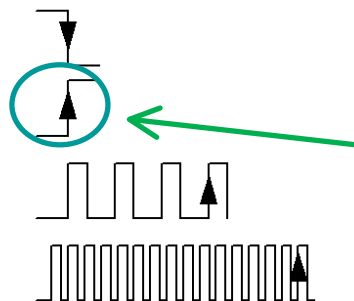


När Capture-händelsen inträffar så kopieras Timer1 direkt över till **CCPR1H** och **CCPR1L** där de kan läsas i "lugn och ro". Biten **CCP1IF** signalerar detta. Den måste vi sedan nollställa.

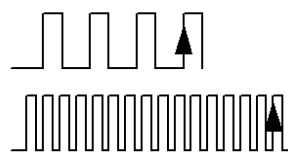
REGISTER 11-1: CCP1CON: ENHANCED CCP1 CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
P1M1	P1M0	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0
bit 7				bit 0			
-	-	-	-	0	1	0	1

Periodtid  
mätning



Multiperiod  
tid mätning



**CCP1M<3:0>**: ECCP Mode Select bits

- 0000 = Capture/Compare/PWM off (resets ECCP module)
- 0100 = Capture mode, every falling edge
- 0101 = Capture mode, every rising edge
- 0110 = Capture mode, every 4th rising edge
- 0111 = Capture mode, every 16th rising edge

William Sandqvist [william@kth.se](mailto:william@kth.se)

# Setup Timer1

Timer1, så snabbt som möjligt:

```
// Setup TIMER1
```

```
/*
```

```
0.x.xx.x.x.x.x TMR1 gate not invert
```

```
x.0.xx.x.x.x.x TMR1 gate not enable
```

```
x.x.00.x.x.x.x Prescale 1:1
```

```
x.x.xx.0.x.x.x TMR1-oscillator is shut off
```

```
x.x.xx.x.1.x.x no input clock-synchronization
```

```
x.x.xx.x.x.0.x Use internal clock f_osc/4
```

```
x.x.xx.x.x.x.1 TIMER1 is ON
```

```
*/
```

```
T1CON = 0b0.0.00.0.1.0.1 ;
```

Tydlig kommentar som visar hur **T1CON** värdet tagits fram.

# Setup ECCP

CCP1, fånga tiden för positiva flanker:

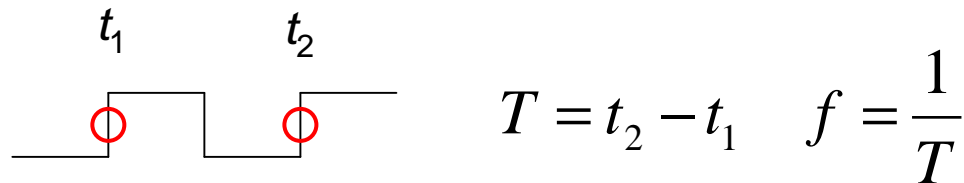
```
// Setup CCP1
/*
  00.00.xxxx -- --
  xx.xx.0101 Capture each positive edge
*/

CCP1CON = 0b00.00.0101 ;
```

# Vänta på flankerna

```
unsigned long T, f, t1, t2; 16-bitarstal
```

```
CCP1IF = 0 ; // reset the flag
while (CCP1IF == 0 ) ; // wait for capture
t1 = CCPR1H*256;
t1 += CCPR1L;
CCP1IF = 0 ; // reset the flag
while (CCP1IF == 0 ) ; // wait for next capture
t2 = CCPR1H*256;
t2 += CCPR1L;
```



```
T = t2 - t1; // calculate period
f = 1000000U/T; // calculate frequency
```

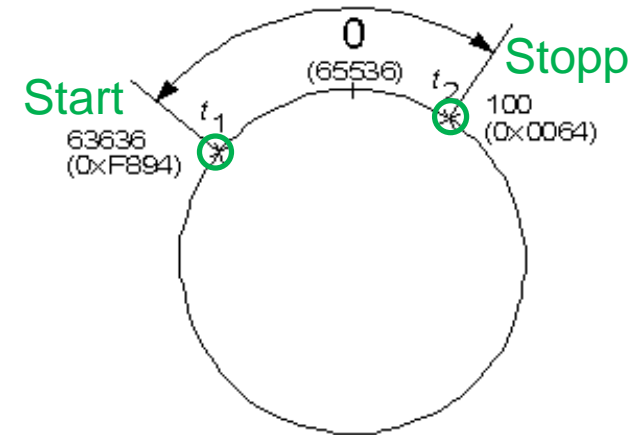


# t2 - t1

`unsigned long T, f, t1, t2;`

- Vad händer om  $t1 > t2$  (100-63636)?

Skillnaden  $t2-t1$  räknas **modulo  $2^{16}$**  så antalet tick mellan  $t1$  och  $t2$  blir därför alltid det korrekta värdet ”varvet runt”!



$$(100 - 63636) \bmod (2^{16}) = 2000$$

$$f = 1000000U/T$$

```
unsigned long T, f, t1, t2; /* long is max 65535 */
```

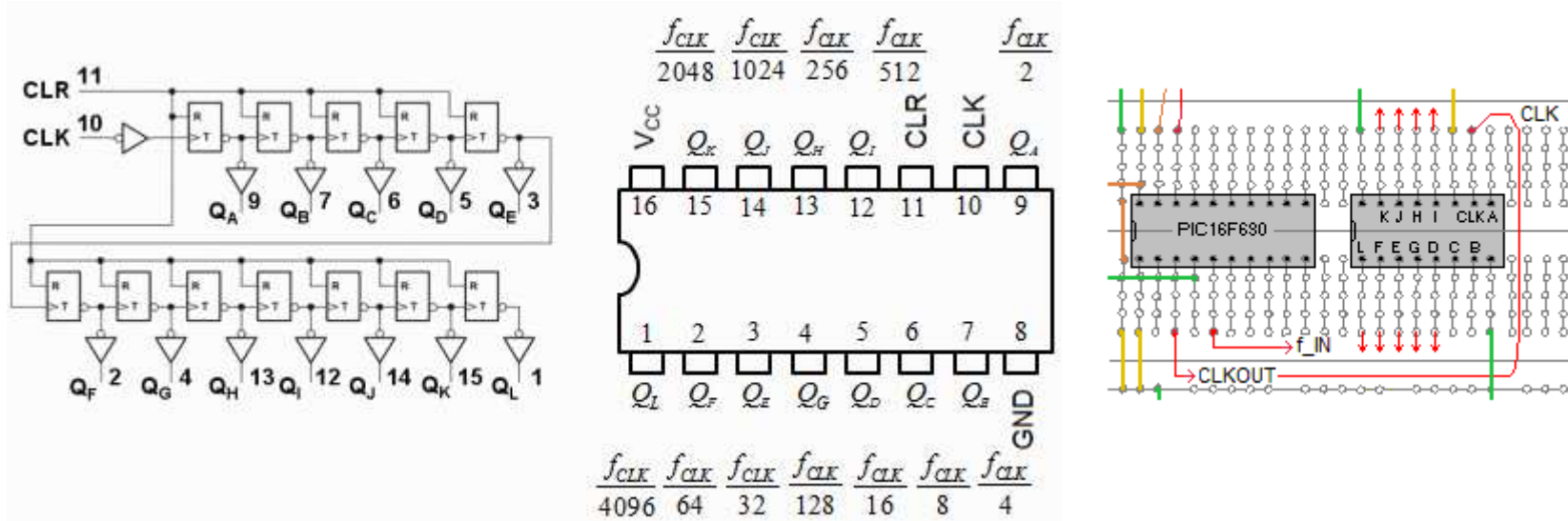
```
f = 1000000U/T;
```

- **Skalfaktorn** mellan  $f$  och  $T$  är 1000000. Timer1 klockas med 1 MHz.
- Om  $T=1$  ( $T=1\pm 1$ ) är mätfrekvensen 1 MHz.  $f > 65535$ , ryms ej.
- Om  $T=10$  ( $T=10\pm 1$ ) är mätfrekvensen 100 kHz.  $f > 65535$ , ryms ej.
- Om  $T=100$  ( $T=100\pm 1$ ) är mätfrekvensen **10 kHz**.  $f < 65535$ , ryms.
- Om  $T=1000$  ( $T=1000\pm 1$ ) är mätfrekvensen **1 kHz**.  $f < 65535$ , ryms.
- Om  $T=10000$  ( $T=10000\pm 1$ ) är mätfrekvensen **100 Hz**.  $f < 65535$ , ryms.
- Om  $T > 65535$  TMR1 slår runt kan bli vad som helst  $f = ?$

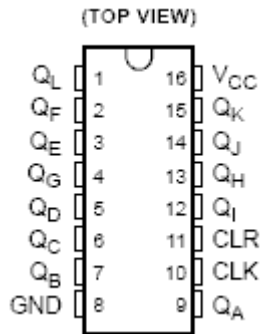
William Sandqvist [william@kth.se](mailto:william@kth.se)

# Frekvensmätning lab

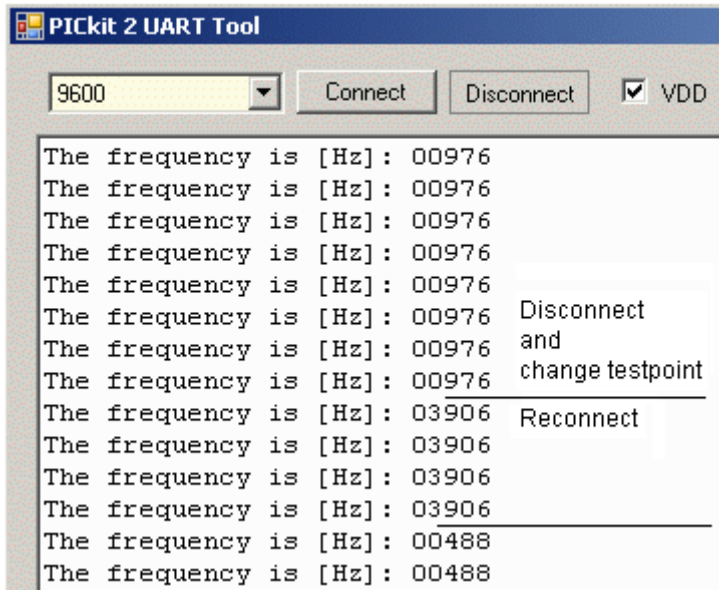
PIC16F690 kan koppla ut processorklockan  $f_{OSC} / 4 = 1 \text{ MHz}$  på pinnen **CLKOUT**. Med det billiga frekvensdelar-chippet 74HC4040 får vi tillgång till 12 olika frekvenser att mäta på!



# Frekvensmätning lab



Varför blir mätvärdena så otroligt exakta?  
Har Du fått tag på en *superbra* PIC16F690?



74HC4040 $f_{CLK} = 1 \text{ MHz}$			
PIN	freq [Hz]	Uppmätt [Hz]	Kommentar
1	$10^6/2^{12} = 244,1$		
2	$10^6/2^6 = 15625$		
3	$10^6/2^5 = 31250$		
4	$10^6/2^7 = 7812,5$		
5	$10^6/2^4 = 62500$		
6	$10^6/2^3 = 125000$		
7	$10^6/2^2 = 250000$		
9	$10^6/2 = 500000$		
12	$10^6/2^9 = 1953,1$		
13	$10^6/2^8 = 3906,3$	3906	
14	$10^6/2^{10} = 976,6$	976	
15	$10^6/2^{11} = 488,3$	488	



*Något verkar skumt ...*

William Sandqvist [william@kth.se](mailto:william@kth.se)