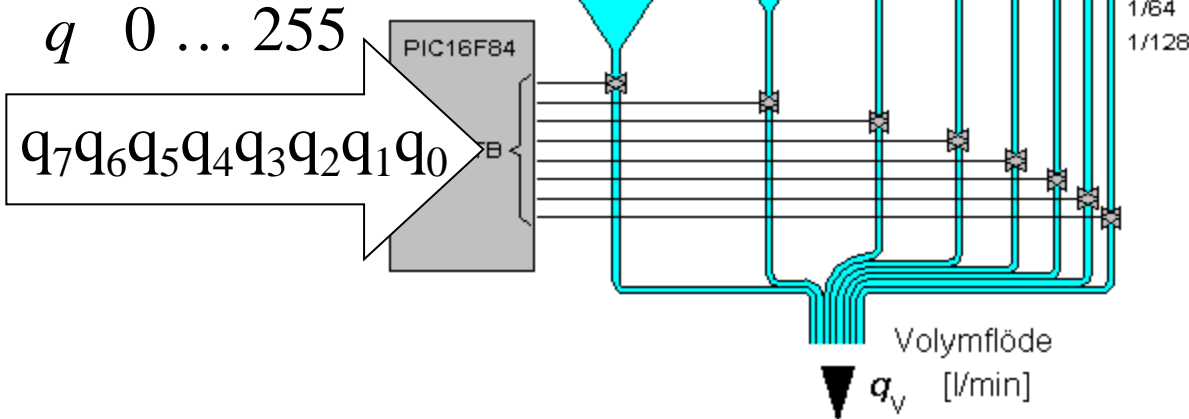


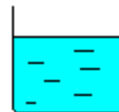
Digitalt eller Analogt

1 1/2 1/4 1/8 1/16 1/32 1/64 1/128
Binärkodade "trattar"

- digitalt:
 q 0 ... 255



- *Digital style*



- eller analogt?

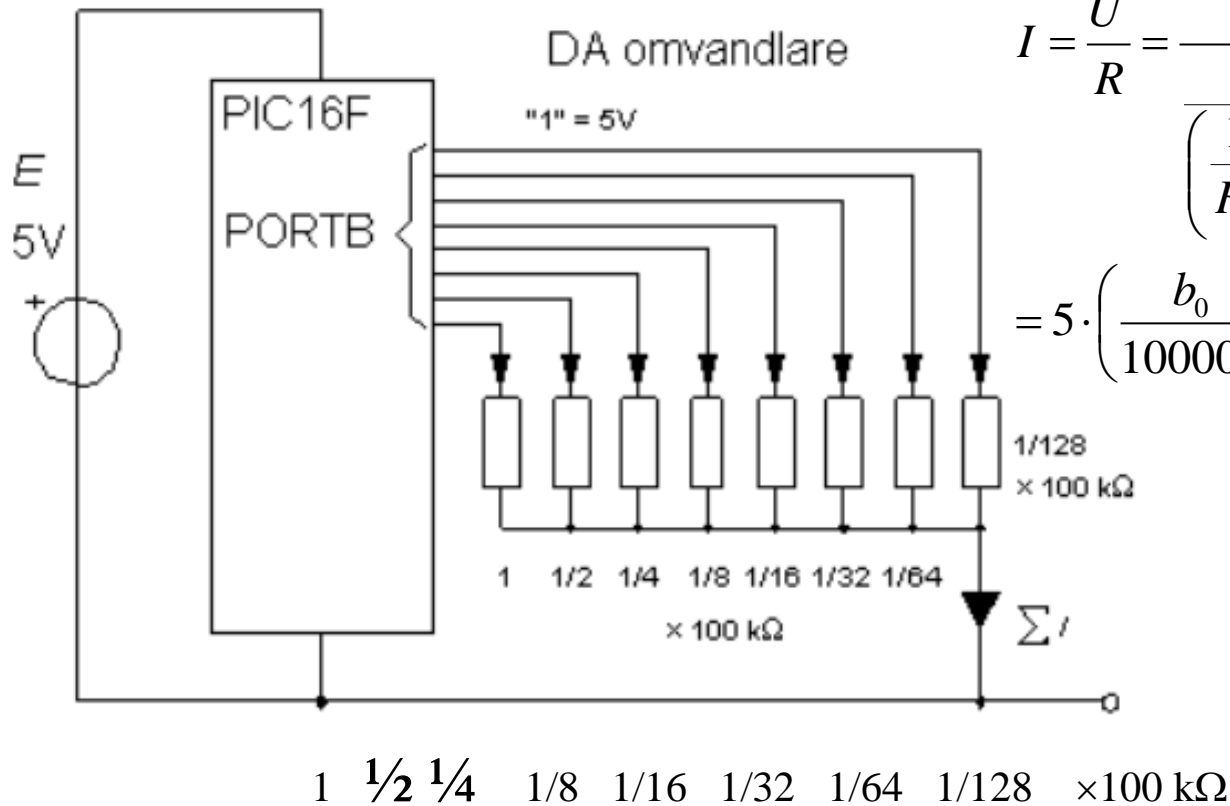


- *Old school*

Digital → Analog omvandlare?

$b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0$

- Binärkodade resistorvärden

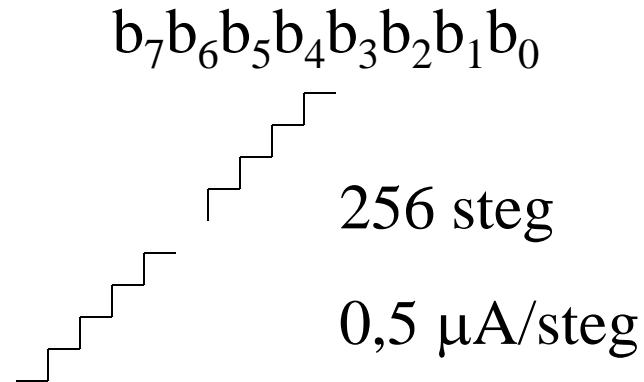
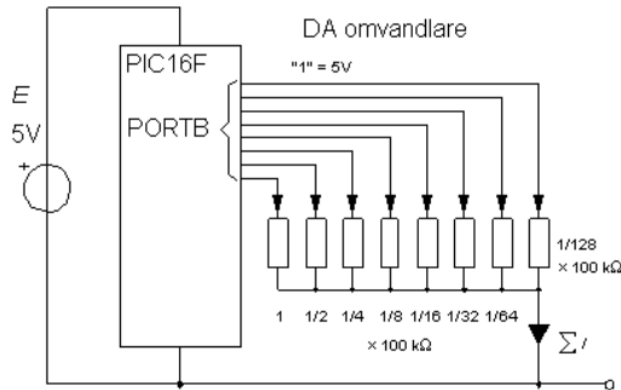


$$I = \frac{U}{R} = \frac{U}{\frac{1}{\left(\frac{1}{R_1} + \frac{1}{R_2} + \dots + \frac{1}{R_n}\right)}}$$

$$= 5 \cdot \left(\frac{b_0}{100000} + \frac{b_1}{50000} + \frac{b_2}{25000} + \dots + \frac{b_7}{781} \right)$$

Strömmen I är **summan** av strömmarna från de "binärkodade" resistorerna.

Digital → Analog omvandlare?



$$I = \frac{U}{R} = 5 \cdot \left(\frac{b_0}{100000} + \frac{b_1}{50000} + \frac{b_2}{25000} + \frac{b_3}{12500} + \frac{b_4}{6250} + \frac{b_5}{3125} + \frac{b_6}{1563} + \frac{b_7}{781} \right)$$

$$I_{\max} = 5 \cdot \left(\frac{1}{100000} + \frac{1}{50000} + \frac{1}{25000} + \frac{1}{12500} + \frac{1}{6250} + \frac{1}{3125} + \frac{1}{1563} + \frac{1}{781} \right) = 128 \mu\text{A}$$

Problem med toleranser

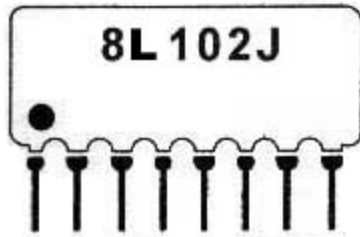
Binärkodade resistorer för 8 bitar.

Största resistorn *exakt* **100000 Ω** och minsta resistorn *exakt* **781 Ω** (helst 781,25)?

Det är svårt att tillverka så *olika* resistorer med snäva toleranser.

- Det finns en bättre lösning!

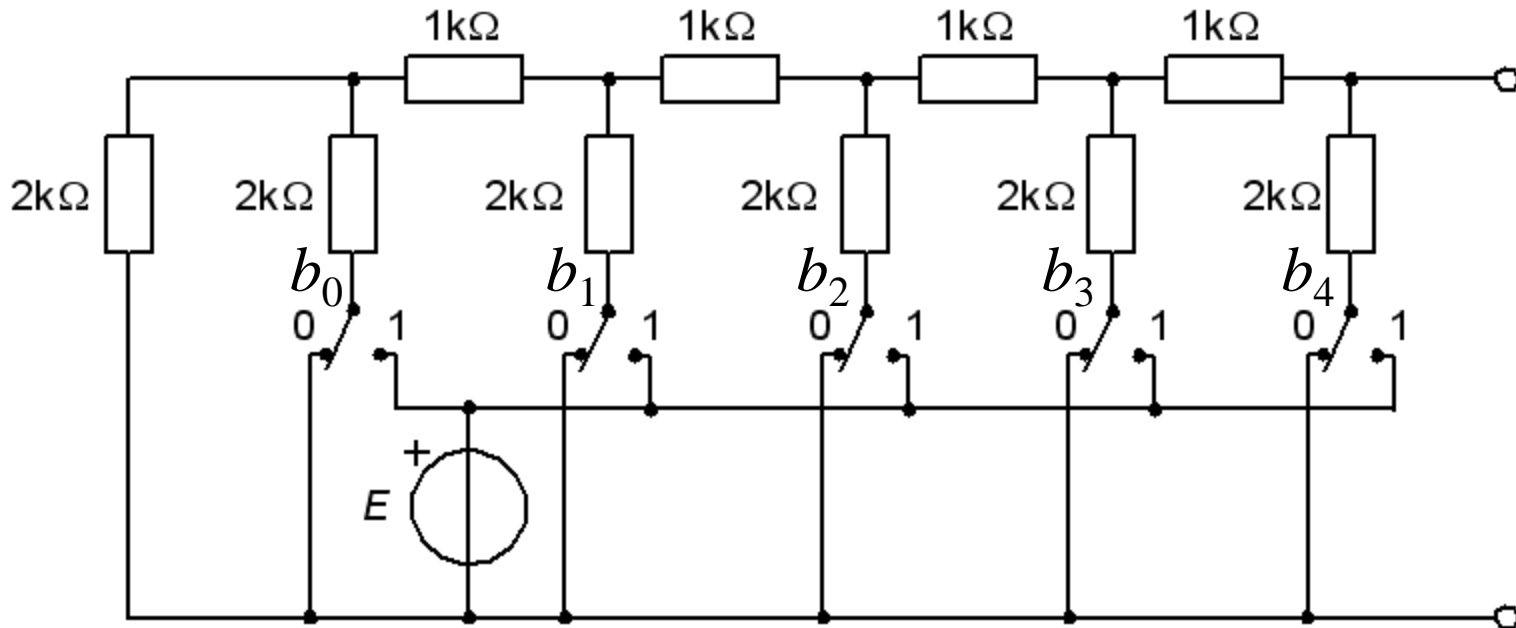
R2R-metoden.



R-2R-stegenät



$$b_4 b_3 b_2 b_1 b_0$$



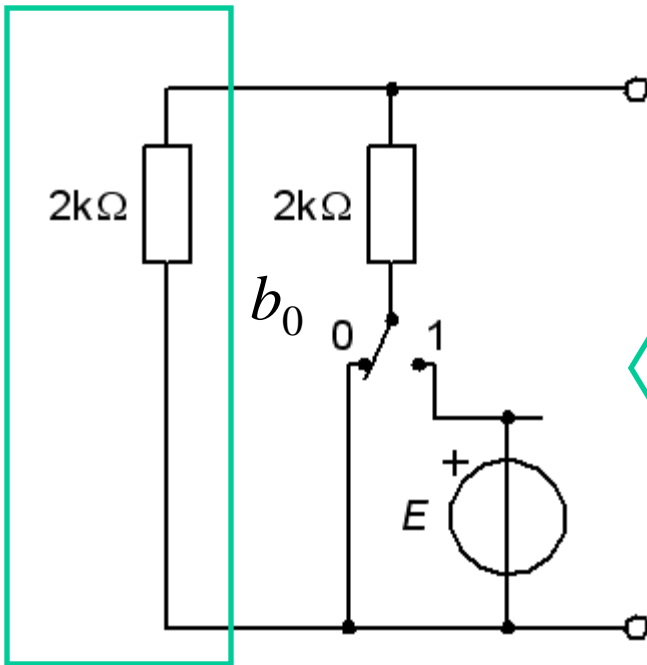
Bara *ett* motståndsvärde behöver tillverkas, R , och då är $R+R=2R$.
Man måste kunna tillverka många "lika" resistorer – det exakta värdet är inte längre det viktiga.

Tvåpolssatsens R_I

R2R-stegenätet är inte så lätt att förstå sig på ...

Upprepad användning av tvåpolssatsen och superpositionssatsen är vad som krävs.

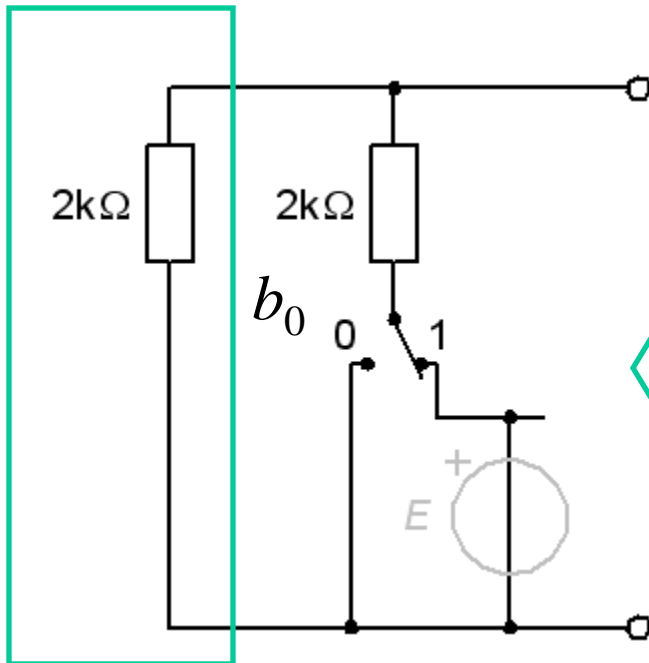
R-2R $b_0=0$ $R_I=?$



$R_I = ?$

$$R_I = \frac{2 \cdot 2}{2 + 2} = 1 \text{ k}\Omega$$

R-2R $b_0=1$ $R_I=?$

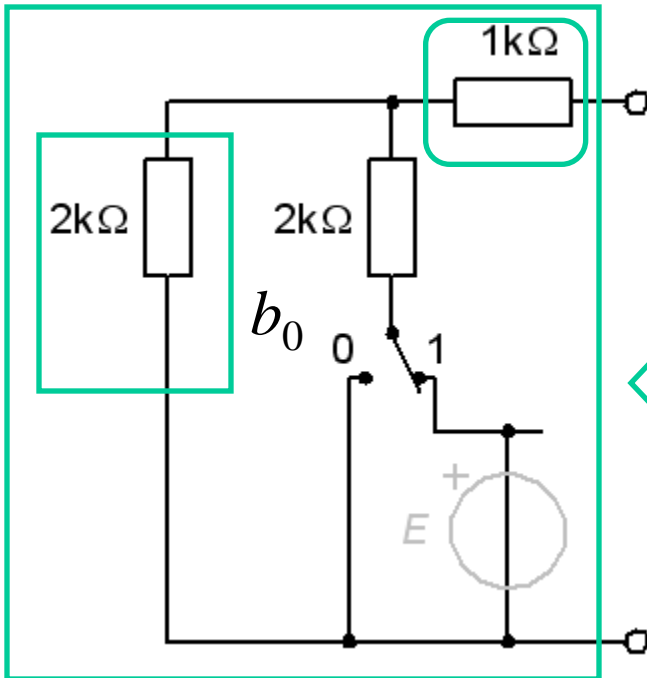


$R_I = ?$

$$R_I = \frac{2 \cdot 2}{2 + 2} = 1 \text{ k}\Omega$$

b_0 1 eller 0, samma inresistans!

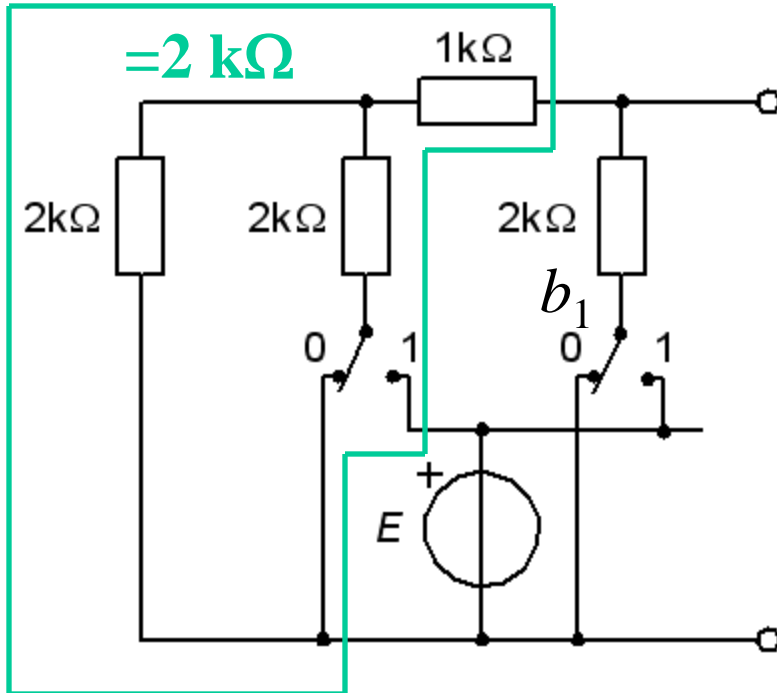
R-2R b_0 $R_I = ?$



$$R_I = 1 + 1 = 2 \text{ k}\Omega$$

Stegets totala resistans blir 2 kΩ.

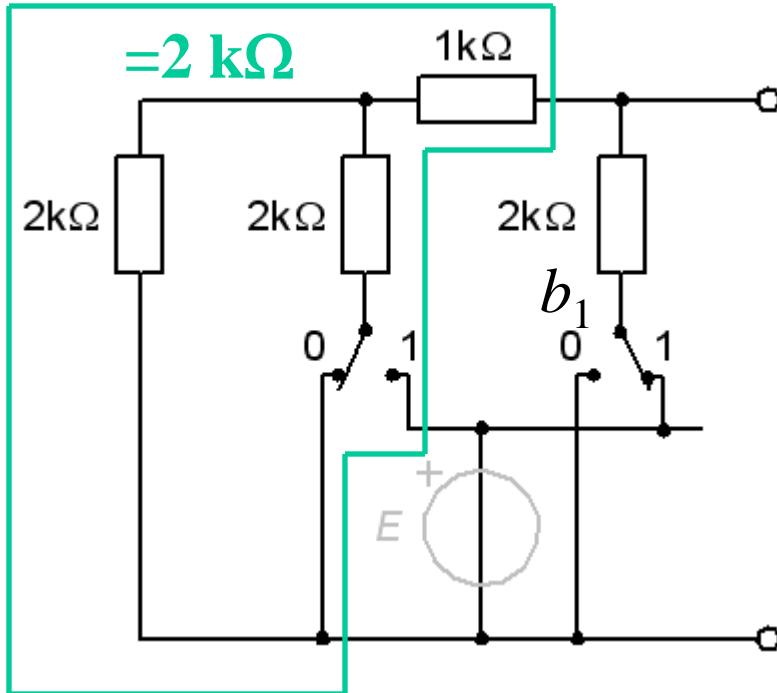
R-2R $b_1=0$ $R_I=?$



$R_I = ?$

$$R_I = \frac{2 \cdot 2}{2 + 2} = 1 k\Omega$$

R-2R $b_1=1$ $R_I=?$

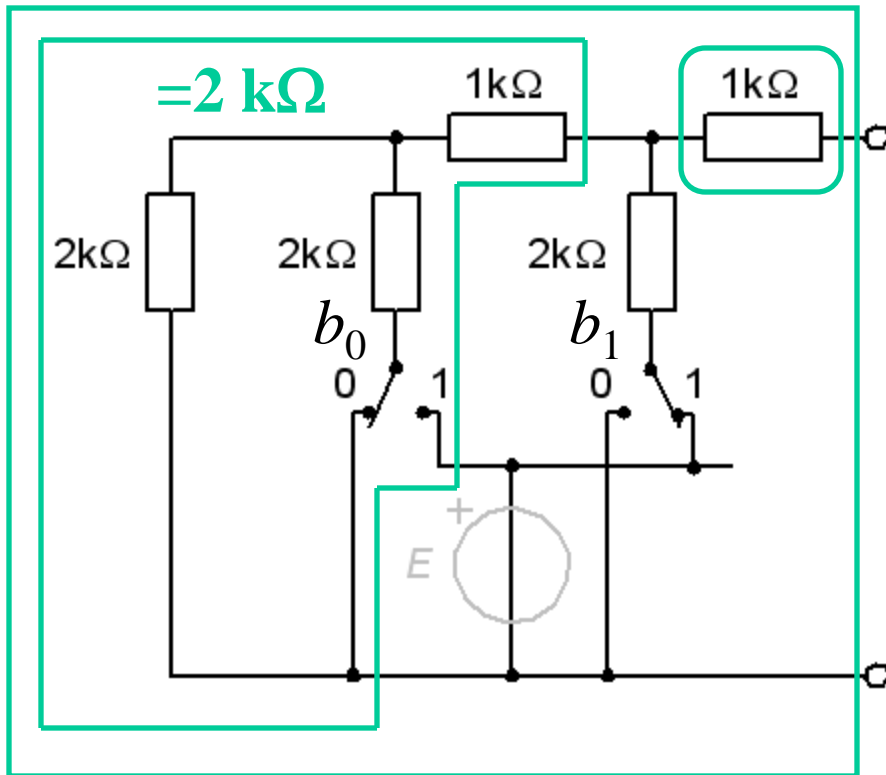


$R_I = ?$

$$R_I = \frac{2 \cdot 2}{2 + 2} = 1\text{ k}\Omega$$

b_1 1 eller 0, samma inresistans!

R-2R b_1 $R_I = ?$



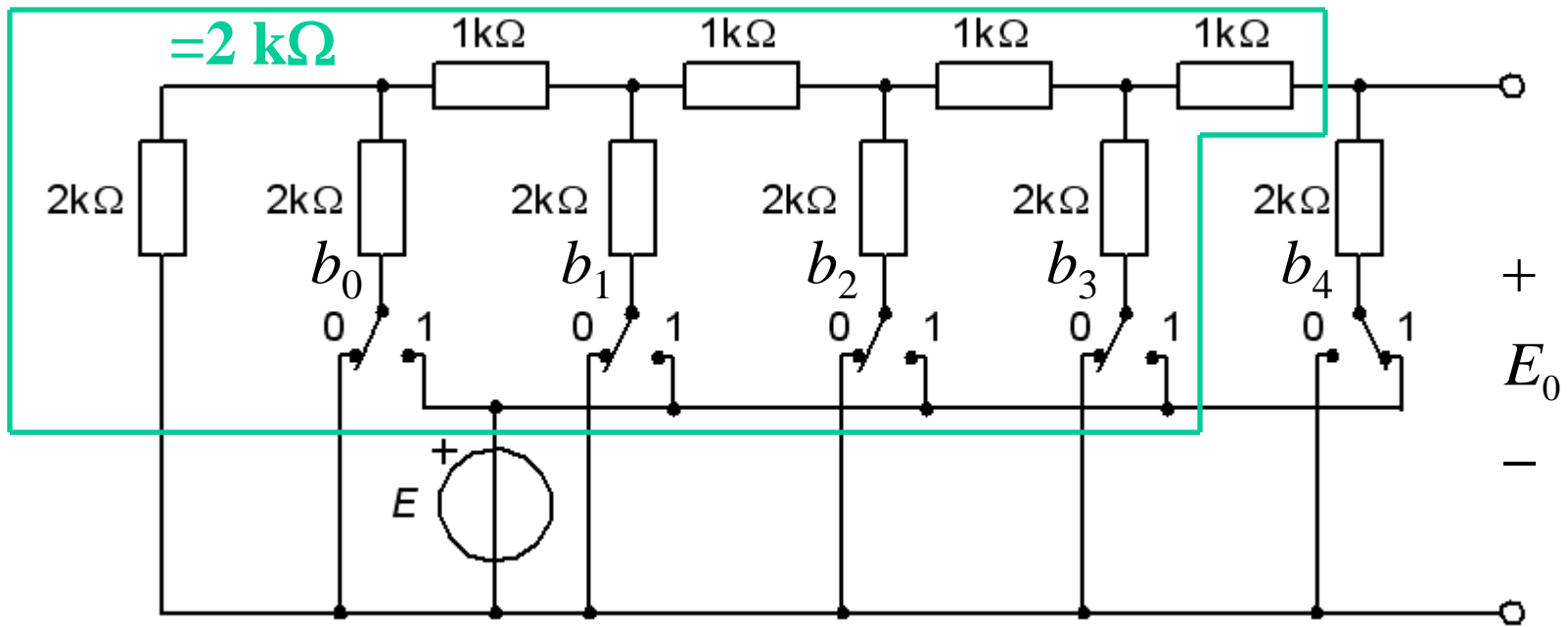
• Slutsats:

$$R_I = 1 + 1 = 2\text{ k}\Omega$$

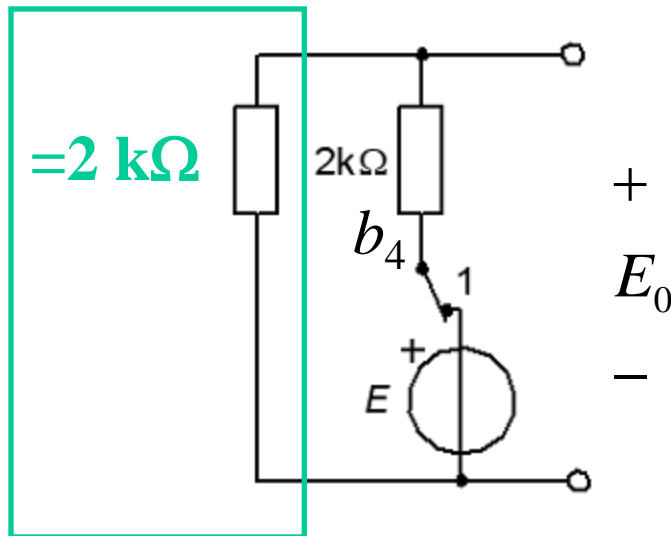
Oavsett 1 eller 0 blir totala inresistansen från de tidigare stegen *alltid* $2\text{ k}\Omega$

Tvåpolssatsens E_0

R-2R $b_4=1$ $E_0=?$



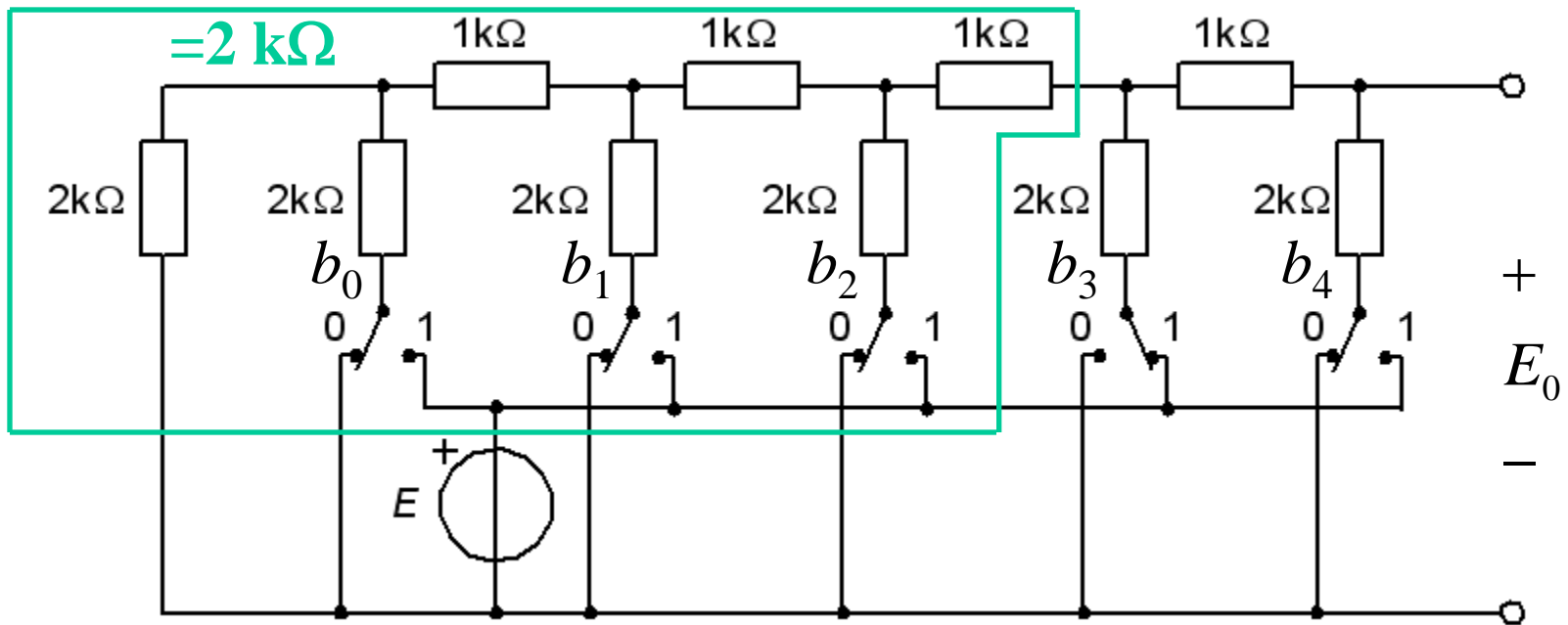
R-2R $b_4=1$ $E_0=?$



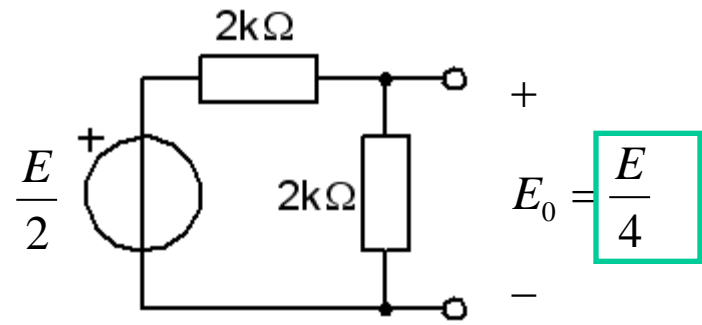
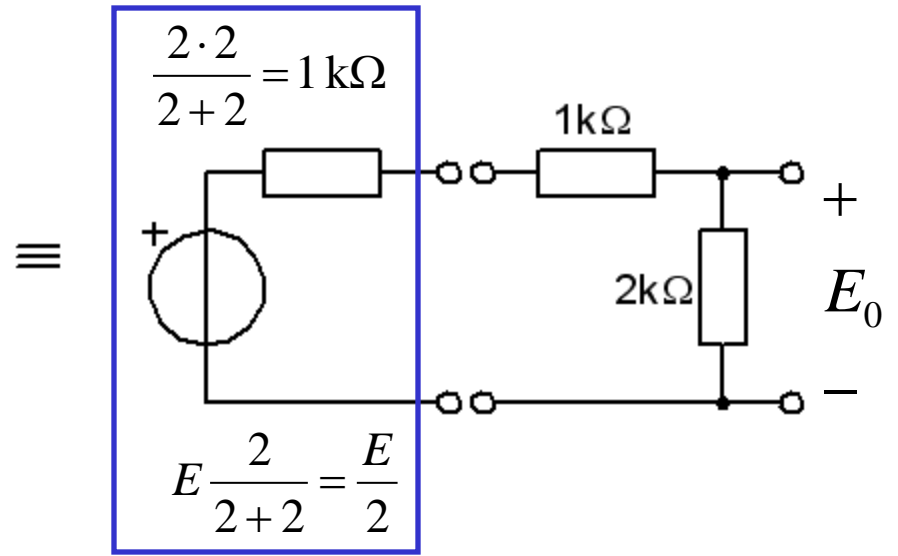
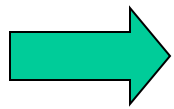
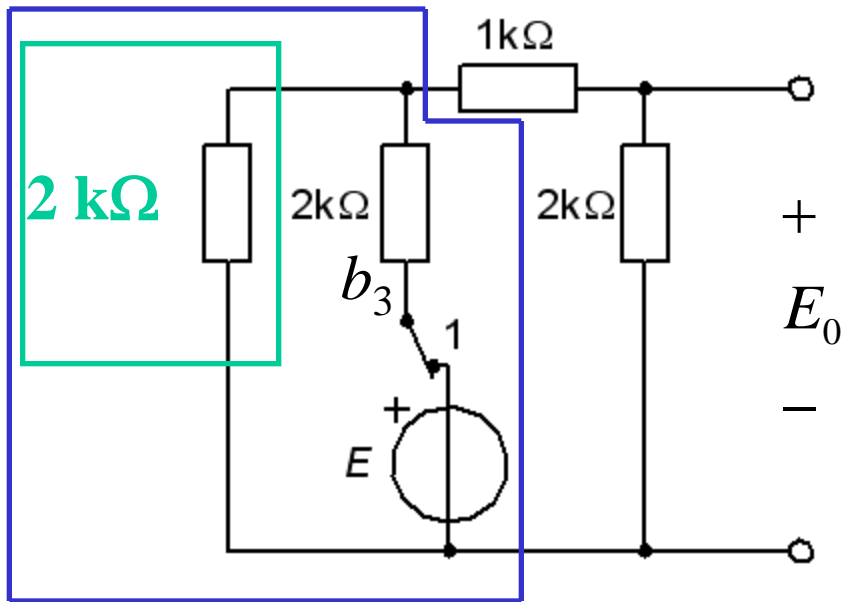
Spänningsdelare:

$$E_0 = E \frac{2}{2+2} = \frac{E}{2}$$

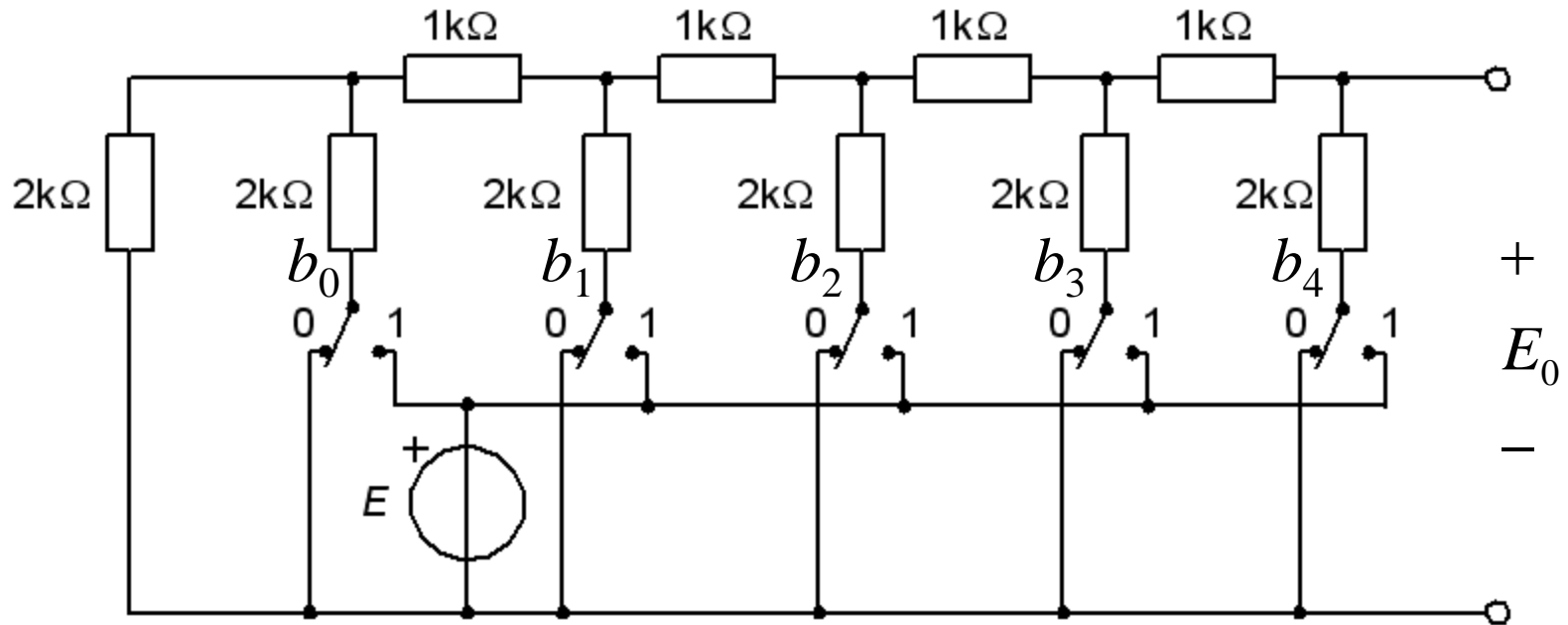
R-2R $b_3=1$ $E_0=?$



R-2R $b_3=1$ $E_0=?$



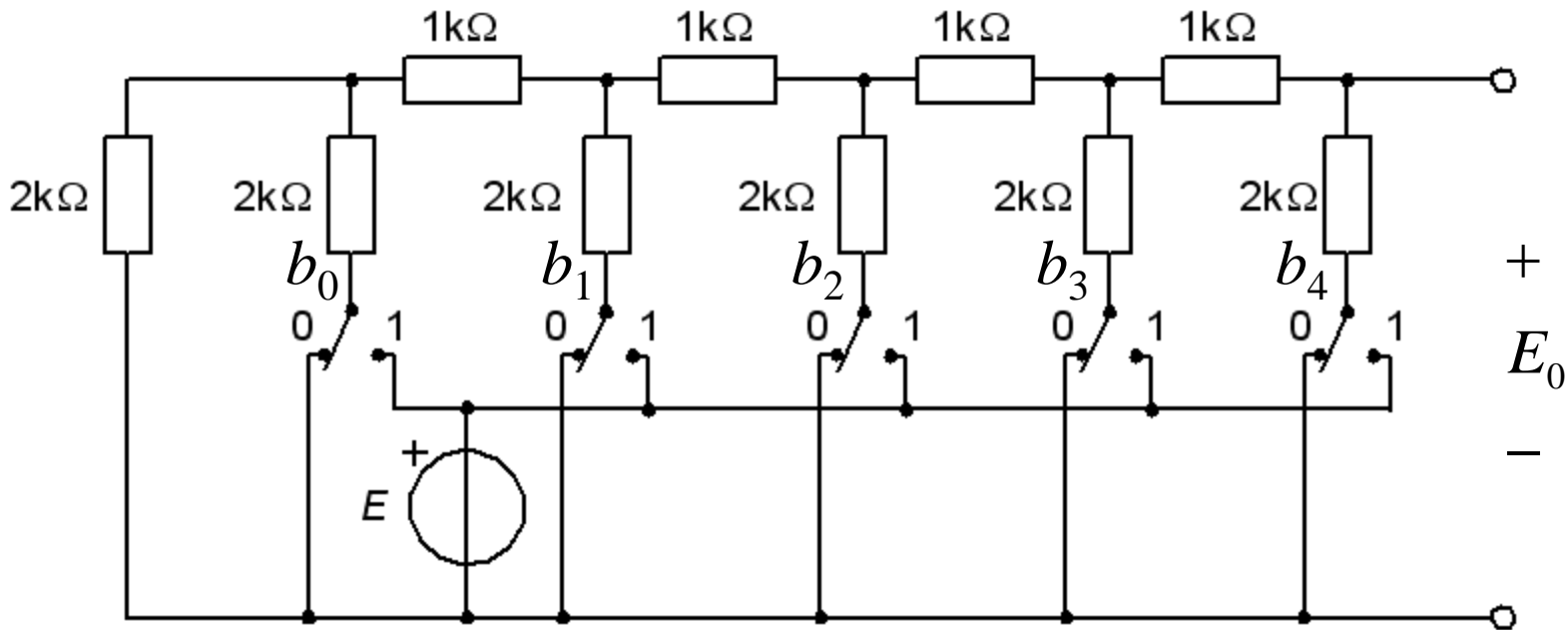
R-2R slutsats $E_0 \dots$



$$b_4 = 1 \Rightarrow \frac{E}{2} \quad b_3 = 1 \Rightarrow \frac{E}{4} \quad b_2 = 1 \Rightarrow \frac{E}{8} \quad \dots$$

- Rimlig gissning – eller hur?

R-2R superposition



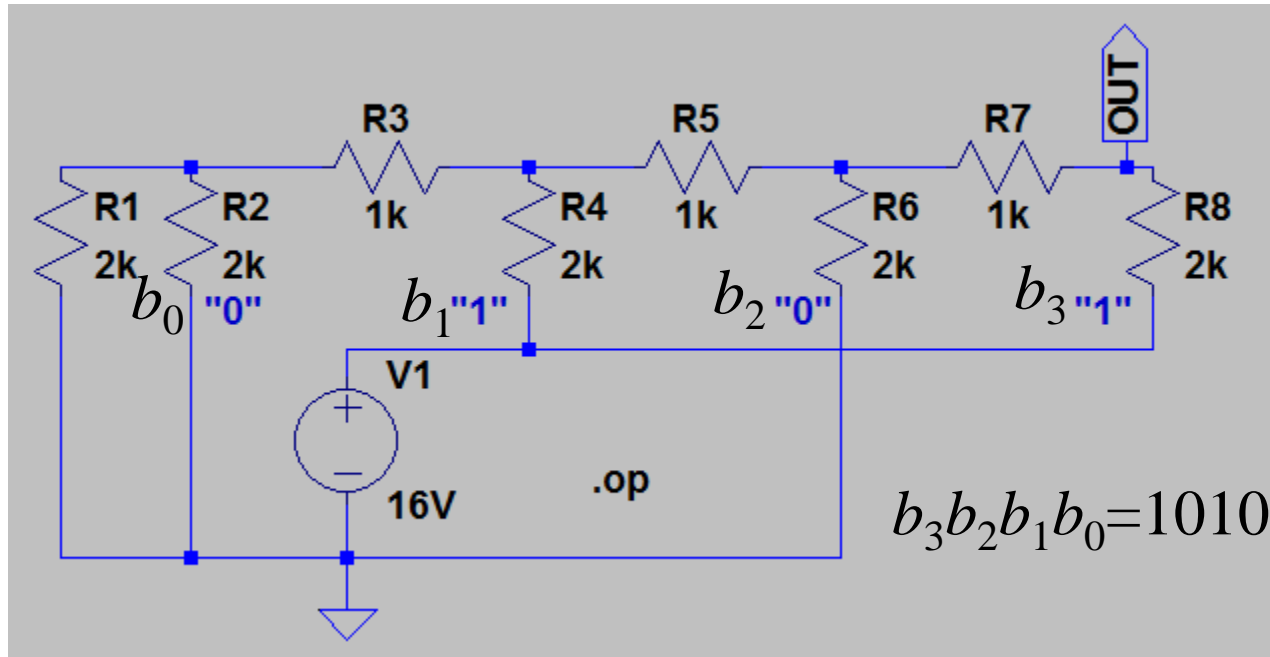
- Enligt **superpositionsprincipen** kan bidragen från $b_4 b_3 b_2 \dots b_0$ adderas var för sig:

$$b_4 b_3 b_2 b_1 b_0 \longrightarrow E_0 = E \cdot \left(\frac{b_4}{2} + \frac{b_3}{4} + \frac{b_2}{8} + \frac{b_1}{16} + \frac{b_0}{32} \right)$$

William Sandqvist william@kth.se

*Vi har en
DA-
omvandlare!*

R-2R simulering

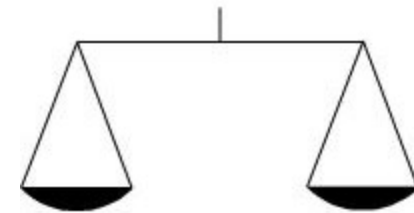
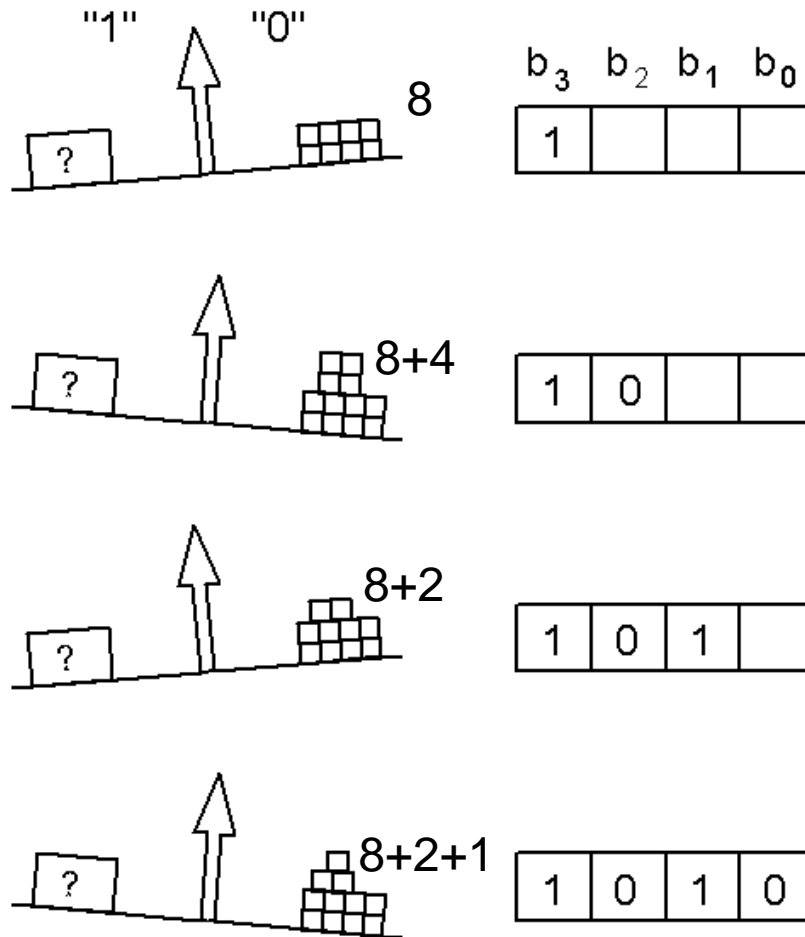


Vad tror Du OUT blir?

William Sandqvist william@kth.se

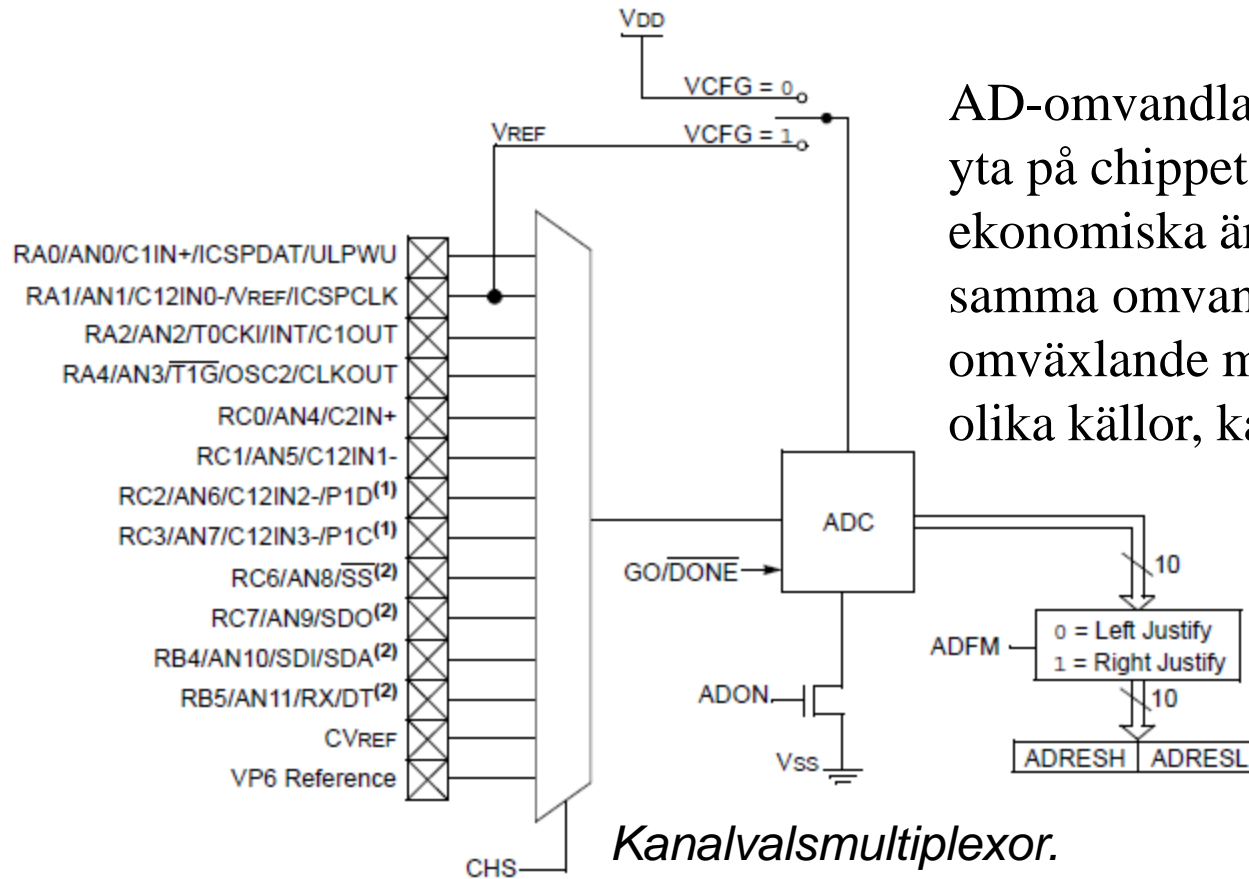
AD-omvandlare?

Binära vikter 8 4 2 1 ...



AD-omvandlingen tar tid. För varje bits ökad upplösning krävs jämförelse i ytterligare ett steg.

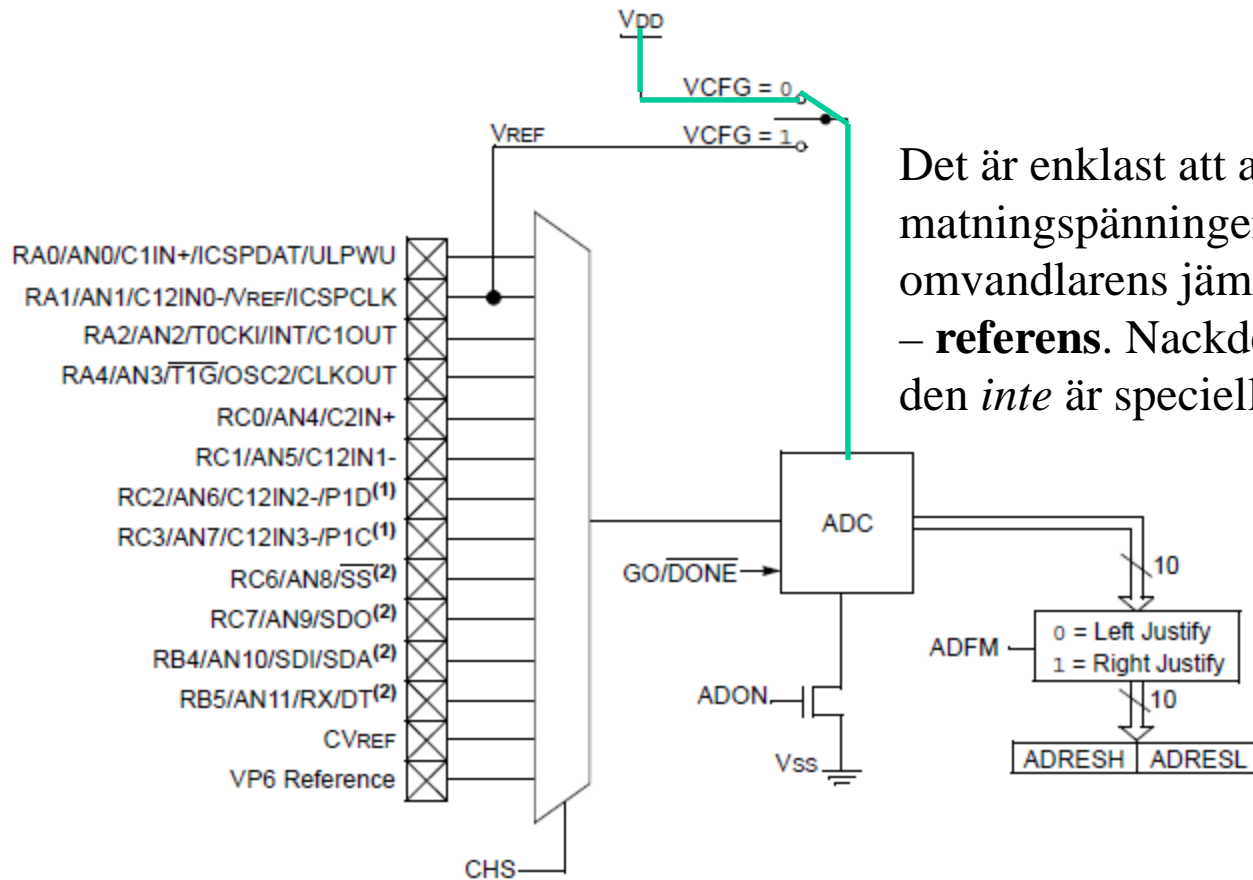
En AD-omvandlare, 14 kanaler



AD-omvandlaren upptar en stor yta på chippet – så det ekonomiska är att använda samma omvandlare till att omväxlande mäta upp till 14 olika källor, kanaler.

PIC-
processorn
har en 10-
bitars AD-
omvandlare

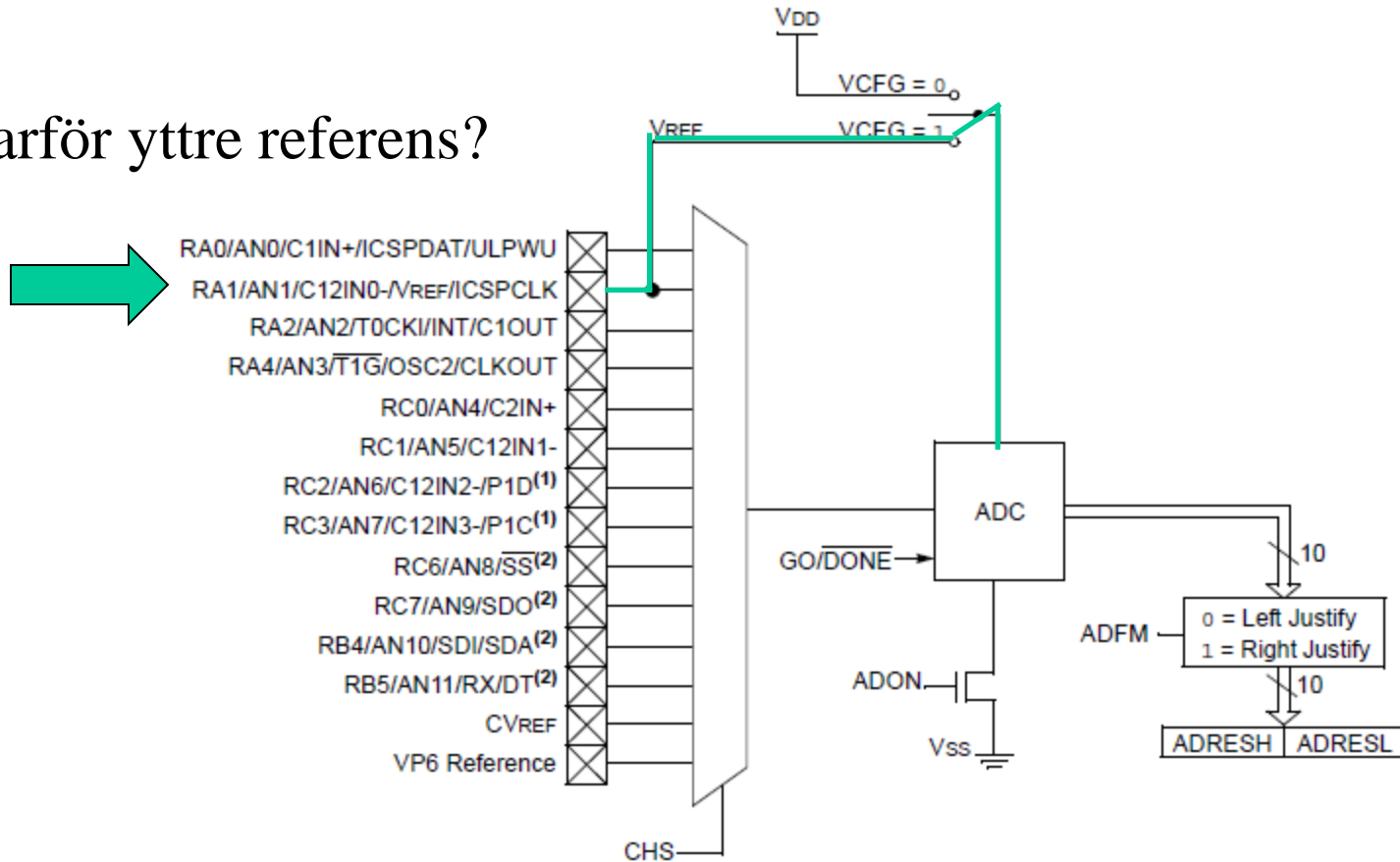
Matningspänningen som referens



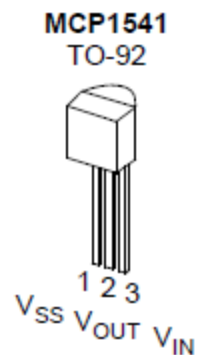
Det är enklast att använda matningspänningen som AD-omvandlarens jämförelsevärde – **referens**. Nackdelen är att den *inte* är speciellt noggrann.

Inre eller yttre referens?

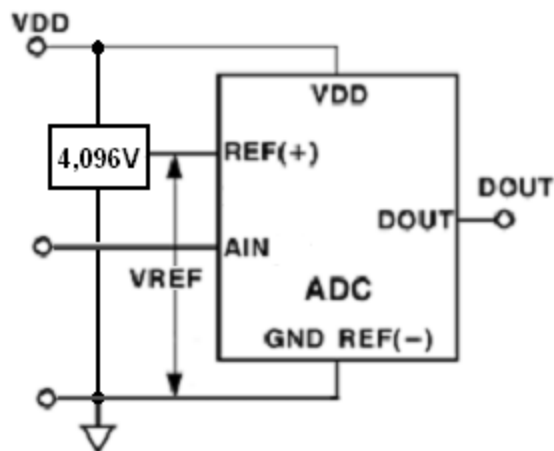
Varför yttre referens?



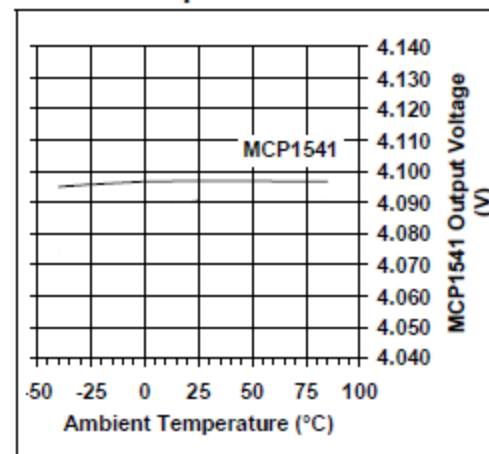
Stabiliserad referens 4,096 V



MCP1541



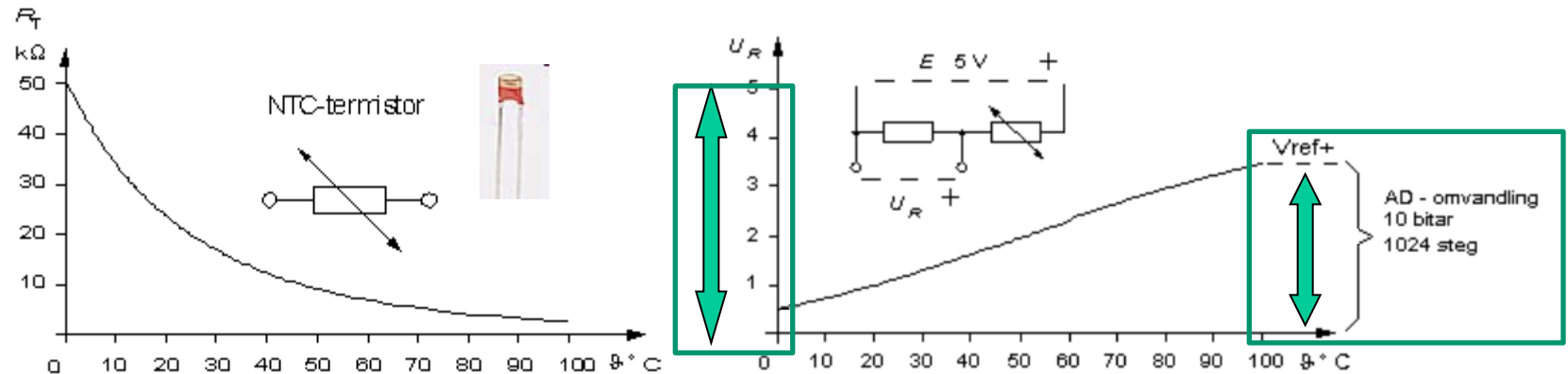
Temperature Drift



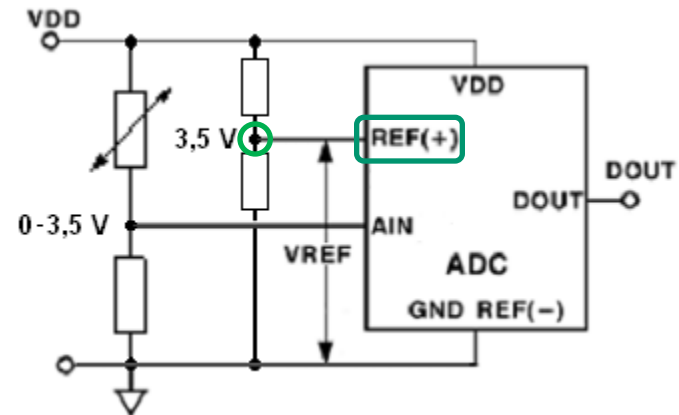
Om man köper en **stabiliserande referens** krets kan man kanske välja värdet 4,096 V ($4096=2^{12}$) som ger en 10-bitars AD-omvandlare exakta 4 mV-steg, *utan* att man behöver tillgripa skalning med multiplikationer och divisioner.

Anpassa mätområdet

En NTC-termistor har hög känslighet, men ett olinjärt temperatursamband.



Man linjäriserar med **en resistor** – och får då mätområdet 0 ...3,5 V. Om referensen då är 3,5 V i stället för 5 V så utnyttjar man *hela AD-omvandlarens område* till mätningen.



Eller så programmerar vi linjäriseringen

```
test5.c
1  /* test5.c float NTC linearization */
2  /* No hardware needed */
3  /* B Knudsen Cc5x C-compiler - not ANSI-C */
4
5  #include "16F690.h"
6  #include "math24f.h"
7  #include "math24lb.h"
8  #pragma config |= 0x00D4
9
10 void main( void)
11 {
12     unsigned long int R_T;
13     float T, temp1, temp2;
14     const float A0 = 123.456;
15     const float A1 = 345.678;
16     // T=1/(A0+A1*log(R_T))
17     temp1=log((float) R_T);
18     temp2=A1*temp1;
19     temp1=A0+temp2;
20     T=1/temp1;
21 }
22
```

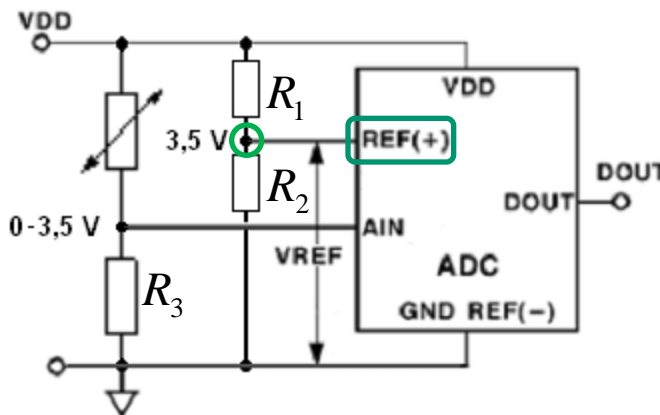
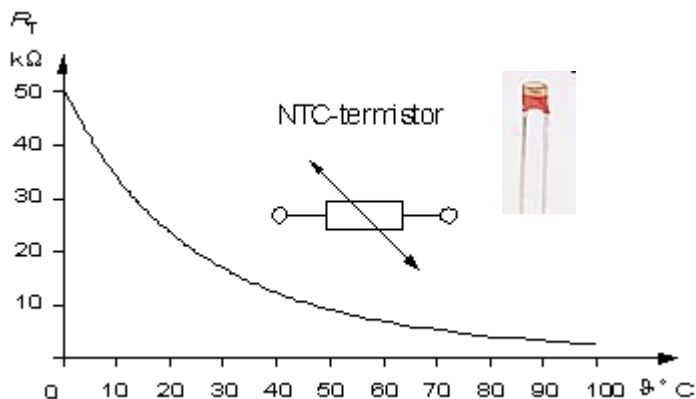
$$T = \frac{1}{a_0 + a_1 \cdot \ln(R_T)}$$



```
Console
NPP_SAVE: H:\PK2proj\Work\test5.c
CD: H:\PK2proj\npp
Current directory: H:\PK2proj\npp
CD: ..\Work\
Current directory: H:\PK2proj\Work
..\Cc5x\Cc5x.exe test5.c -a
Process started >>>
CC5X Version 3.4H, Copyright (c) B Knudsen Data, Norway 1992-2012
--> FREE edition, 8-16 bit int, 24 bit float, 32k code, reduced optim.
test5.c:
Chip = 16F690
RAM : -----
40h: =====, * *****
80h: -----
C0h: -----
100h: -----
140h: -----
RAM usage: 39 bytes (38 local), 217 bytes free
Optimizing - removed 69 instructions (-6 %)
File 'test5.asm'
File 'test5.occ'
File 'test5.hex'
Codepage 0 has 901 word(s) : 43 %
Codepage 1 has 0 word(s) : 0 %
Total of 901 code words (21 %)
* Estimated CODE SIZE of full optimization: 689 code words (-23 %)
<<< Process finished.
```

Det finns ett val ... ??

Det finns tydligen ett val mellan att köpa **tre resistorer** – eller att använda **900 instruktioner** programkod.



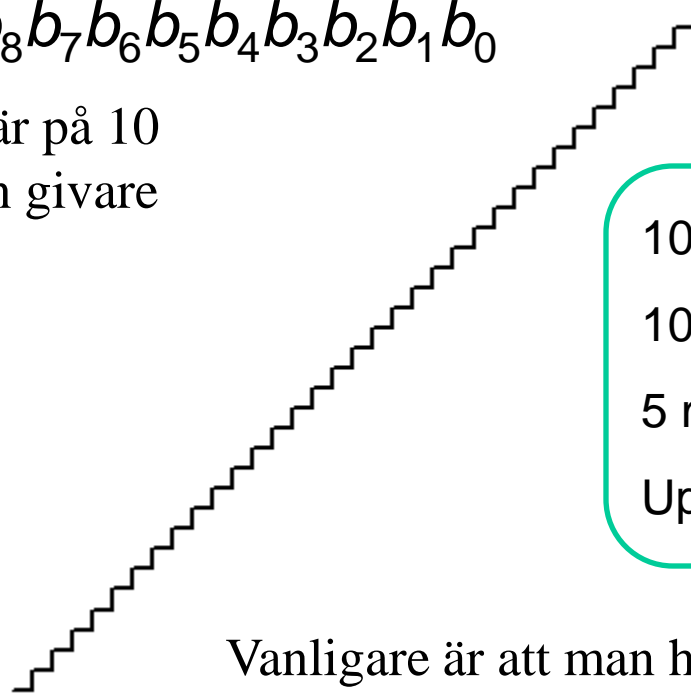
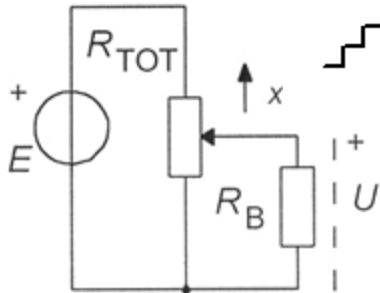
För den som känner till Inbyggd elektronik ...

Behövs 10 bitars upplösning?

$b_9 b_8 b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0$

AD-omvandlaren är på 10 bitar. Vad kostar en givare som har 10 bitars upplösning?

100 \$?



10 bitar

1024 steg

5 mV/steg

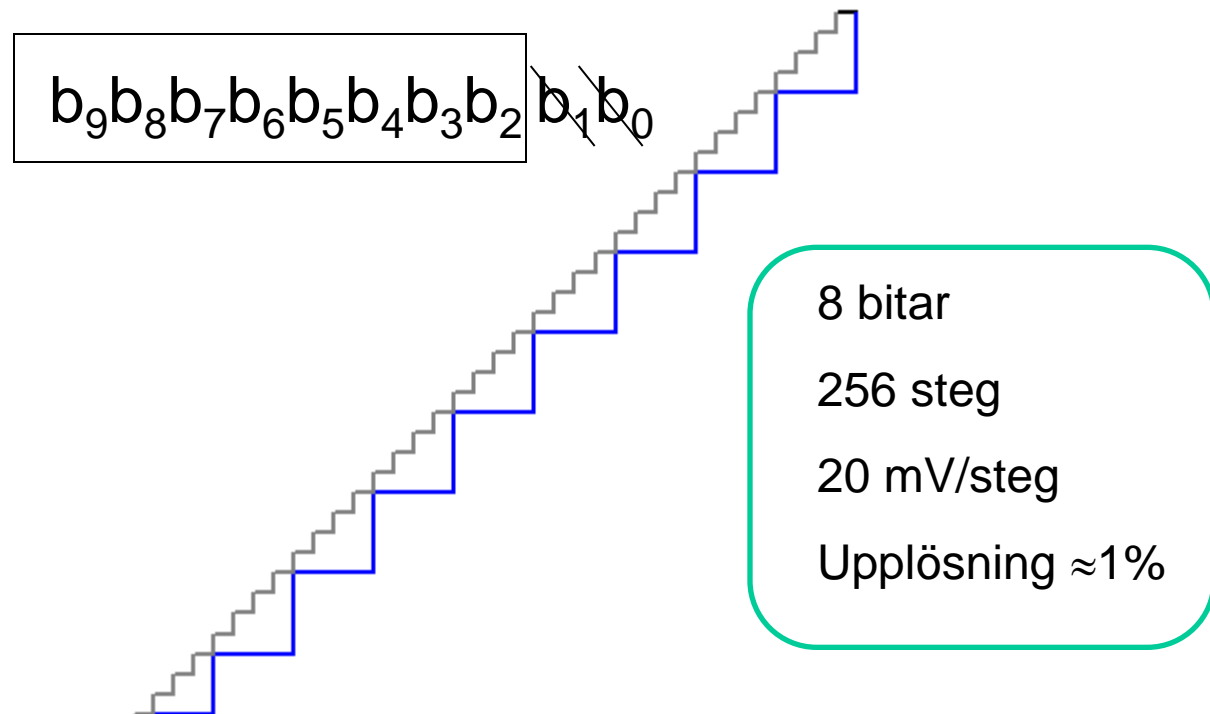
Upplösning $\approx 0,1\%$

Vanligare är att man har råd med en 8 bitars givare. (PICprocessorn själv kostar 2 \$).

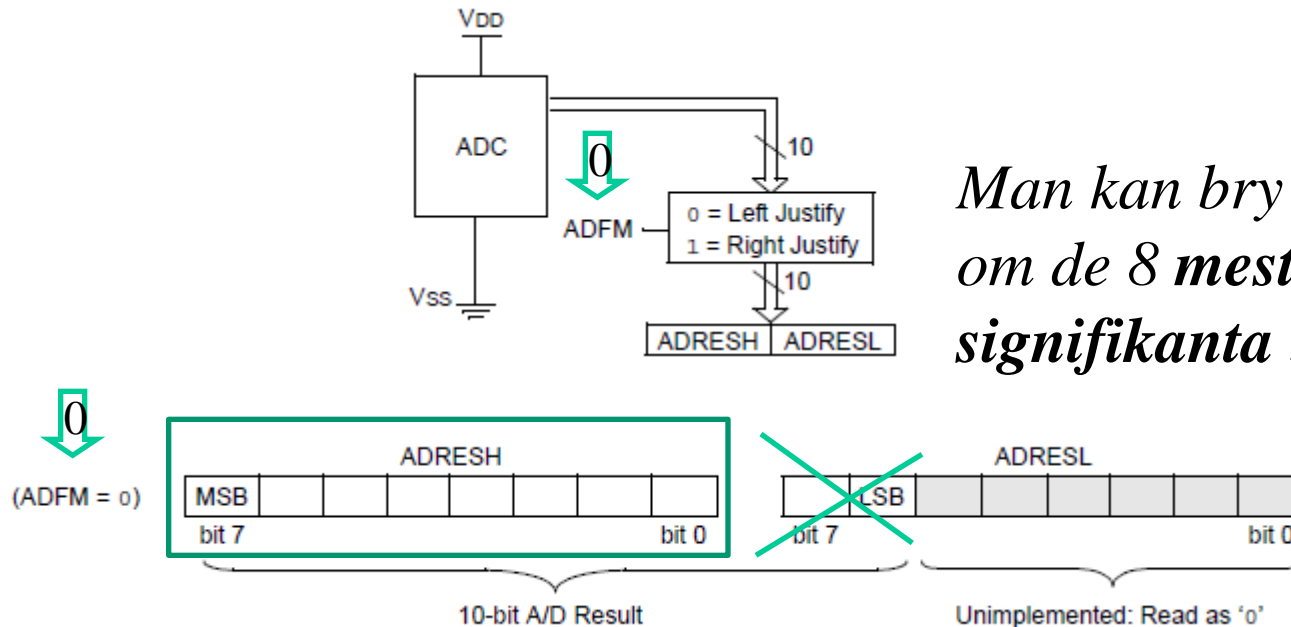
Bilden visar en resistiv givare för position som kan dra nytta av 10-bitars upplösning.

8-bitars program

Behöver man bara **8 bitars upplösning** kan man *strunta* i de två minst signifikanta bitarna och hantera resultatet som en Byte.



8-bitars program



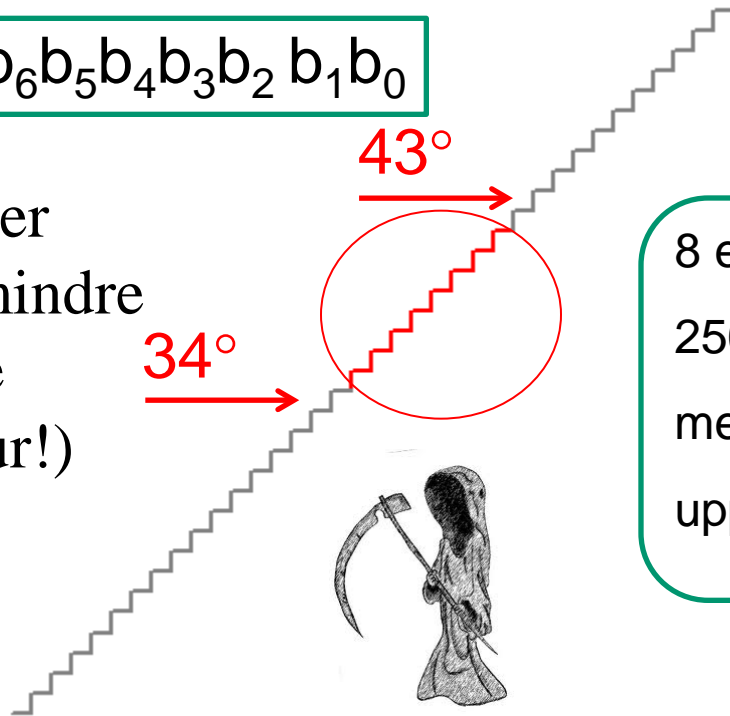
```
ADFM=0;  
char value;  
value = ADRESH; /* 8-bit measurement */
```

Undvik förstärkare

Behöver man bara **8 bitars upplösning** kan man *ändå* använda 10 bitars upplösning för att undvika att behöva förstärka givarsignalen. De två mest signifikanta bitarna blir **konstanta**. Man kan därför låta bli att läsa av dem.

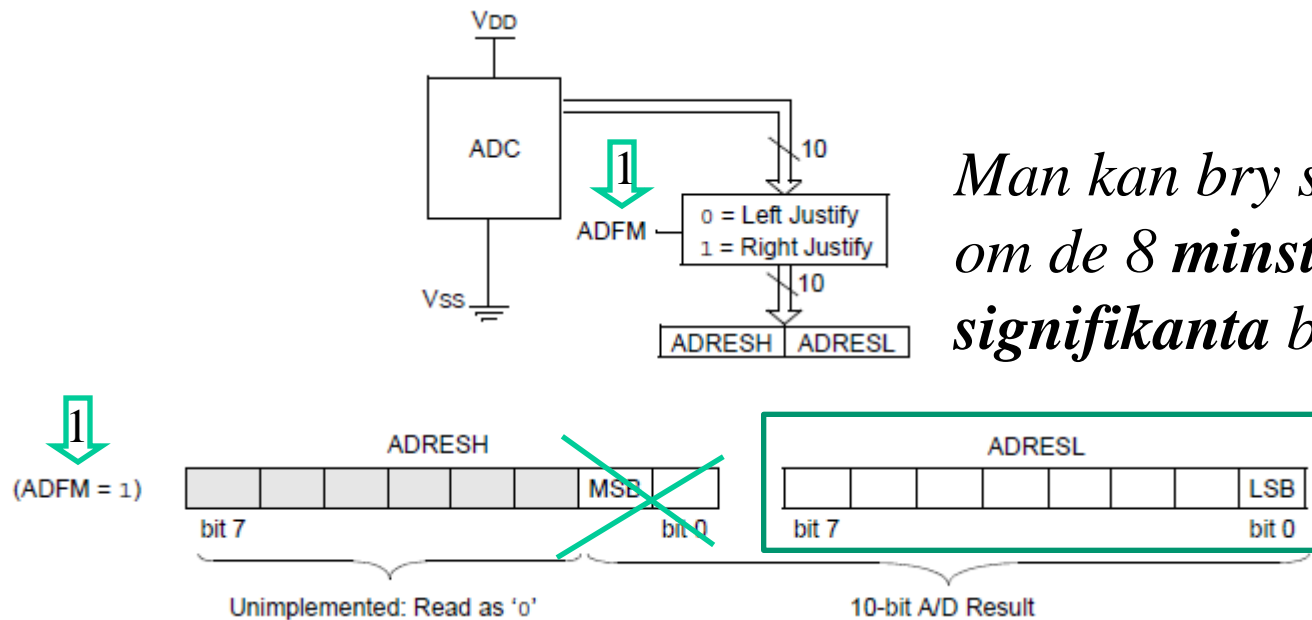
~~1~~ ~~0~~ $b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0$

En febertermometer
behöver bara ett mindre
temperaturområde
 $34^\circ \dots 43^\circ$ (eller hur!)



8 effektiva bitar,
256 effektiva steg
med 5 mV/steg
upplösning $\approx 1\%$

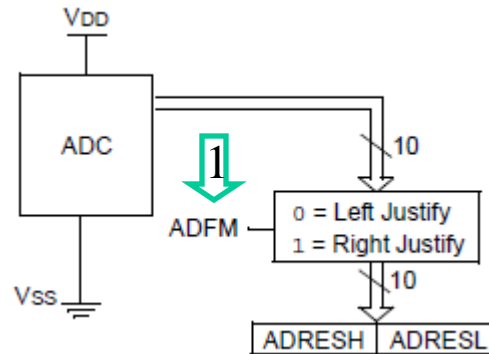
8-bitars program



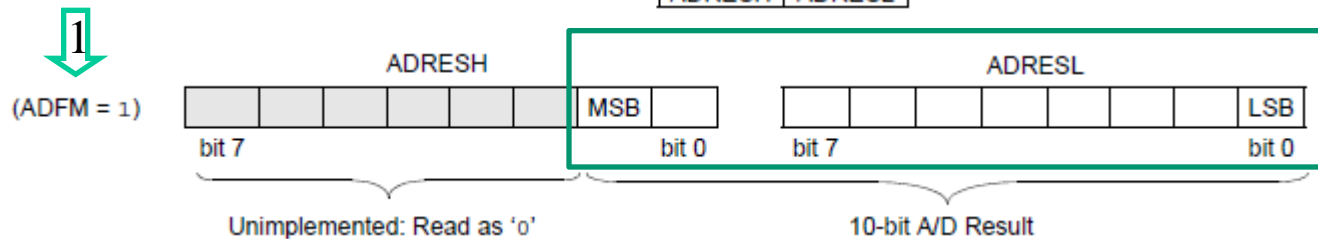
```
ADFM=1;  
char value;  
value = ADRESL; /* 8-bit measurement */
```

16-bitars program

10-bitar
1024 steg
5 mV/steg
upplösning $\approx 0,1\%$



När man verkligen behöver 10 bitar i en 16-bitars variabel.



ADFM=1;

unsigned long value;

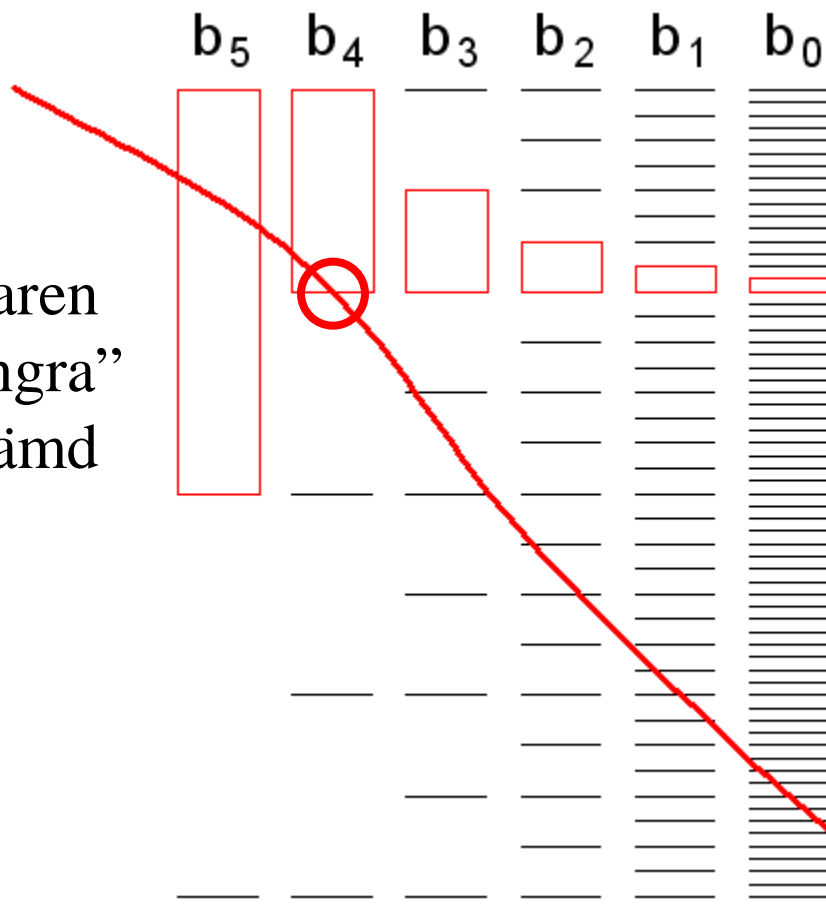
value = ADRESH * 256;

value += ADRESL; /* 10-bit measurement */

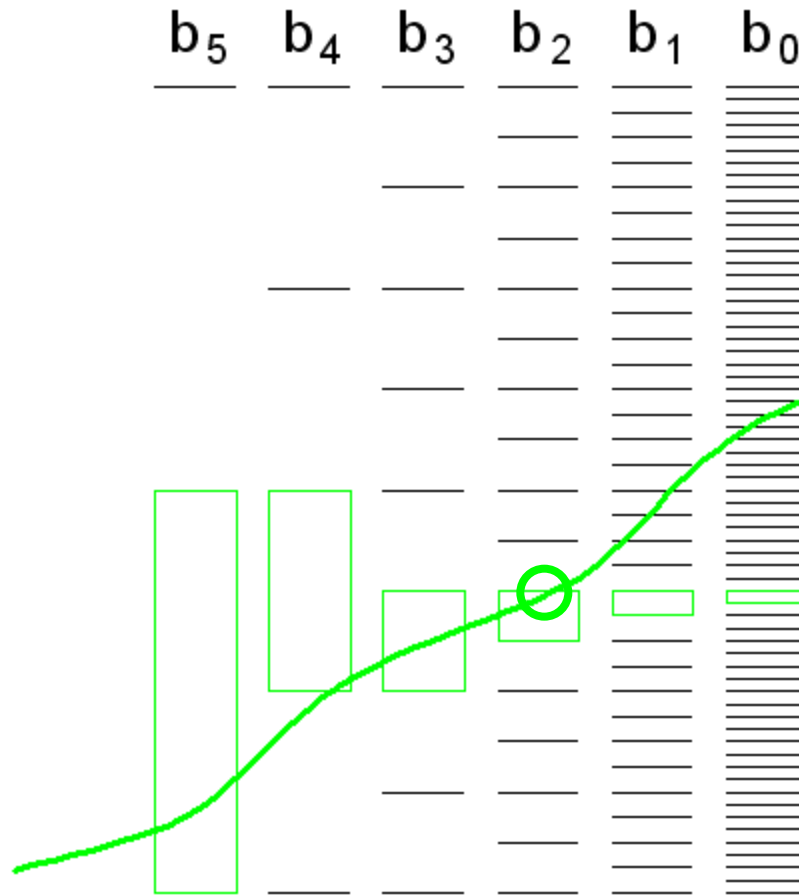
William Sandqvist william@kth.se

Vad händer om signalen ändrar sig under omvandlingen?

AD-omvandlaren kan aldrig "ångra" en redan bestämd bit!



Resultatet blir ett värde som förekommit under omvandlingen, men vid en obestämd tidpunkt!



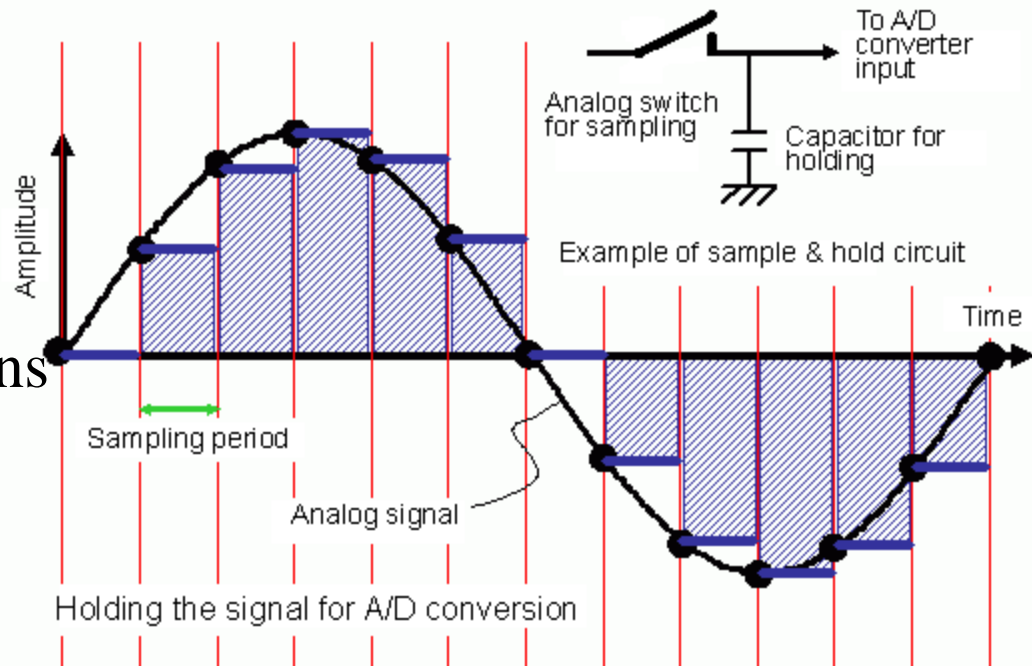
Problemet

Samplingtid-
punkten blir
obestämd!

Sample & Hold -krets

Lösningen är att den analoga signalen ”fryses” under omvandlingen.

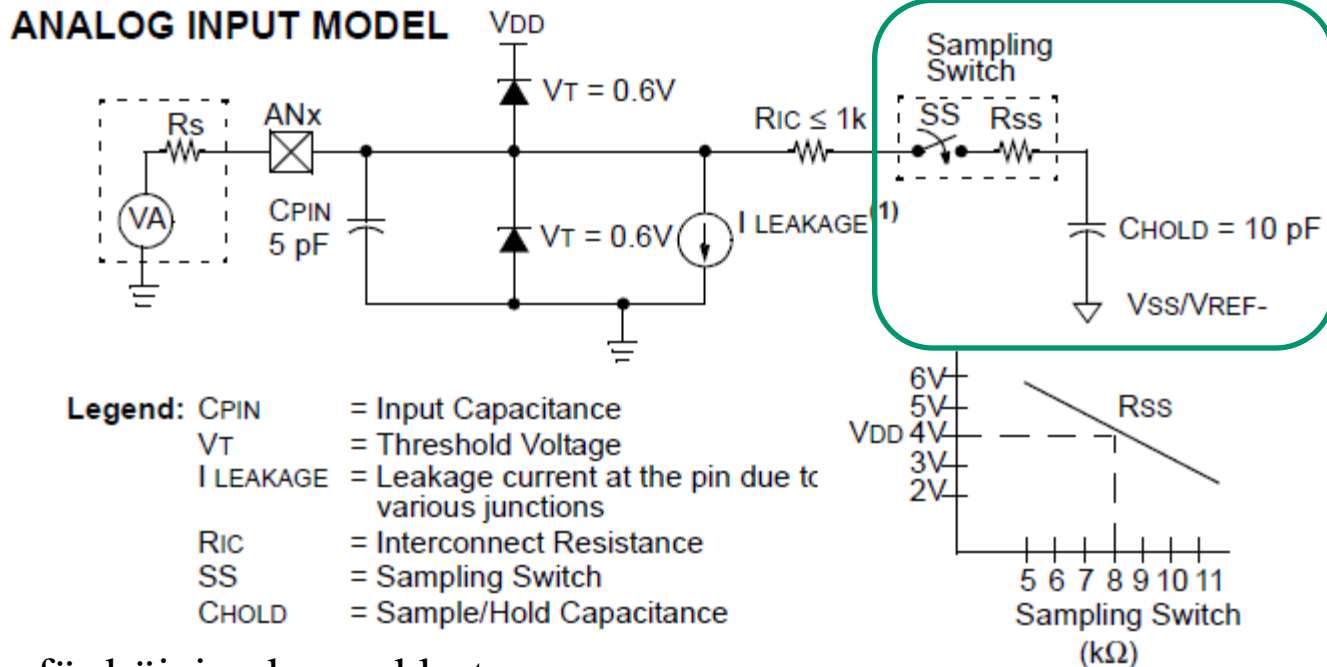
Vid AD-omvandlingens start tar en switch ett ”sample” av signalen och lagrar det i en kondensator.



PIC-processorernas samplingskondensator har kapacitansen $\approx 10\text{pF}$.

Acquisition time t_{ACQ}

Varje gång man har **valt/ändrat kanal** måste **samplingskondensatorn** C_{HOLD} hinna ladda upp sig till den analoga spänningen. Detta tar c:a 5 μ s.



En 5 μ s fördröjning kan enklast programmeras som:

```
nop2(); nop2(); nop(); /* 5 us 4 MHz clock */;
```

AD-klockpulser

AD-omvandlaren kan *högst* använda klockfrekvensen 250 kHz. Om PIC-procressorns klocka är **4 MHz** måste den först delas ned 16 gånger innan den kan användas som AD-klocka.

REGISTER 9-2: ADCON1: A/D CONTROL REGISTER 1

U-0	R/W-0	R/W-0	R/W-0	U-0	U-0	U-0	U-0
—	ADCS2	ADCS1	ADCS0	—	—	—	—
bit 7							bit 0

ADCS<2:0>: A/D Conversion Clock Select bits

000 = FOSC/2

001 = FOSC/8

010 = FOSC/32

x11 = FRC (clock derived from a dedicated internal oscillator = 500 kHz max)

100 = FOSC/4

101 = FOSC/16

110 = FOSC/64

ADCON1=0b0.101.0000;

$$1010000_2 = 80_{10} \quad T_{AD} = 4\mu\text{s} \quad f_{AD} = 500\text{kHz}$$

Starta AD och vänta på klar

REGISTER 9-1: ADCON0: A/D CONTROL REGISTER 0

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	VCFG	CHS3	CHS2	CHS1	CHS0	GO/DONE	ADON
bit 7							bit 0

Starta AD-
omvandling:

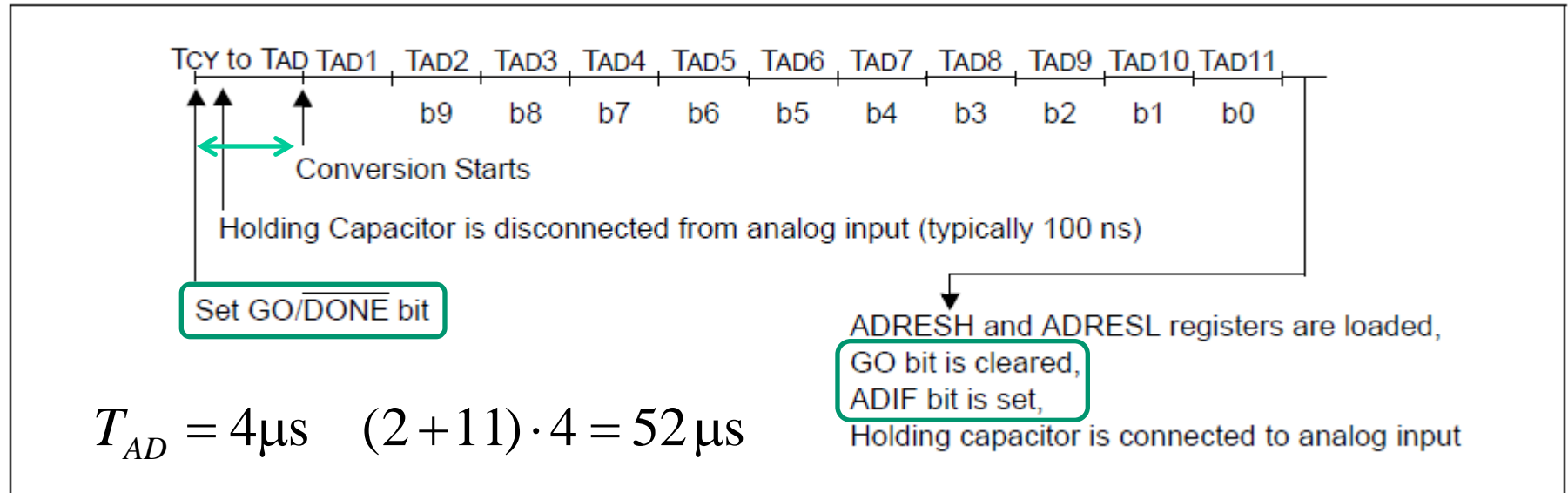
```
GO = 1;
```

Vänta på AD-
omvandling klar:

```
while(GO) ;
```

AD-omvandlingen tar tid

FIGURE 9-2: ANALOG-TO-DIGITAL CONVERSION T_{AD} CYCLES



Omvandlingen tar c:a 2+11 AD-klockpulser.

Om man bortser från att PIC-processorn måste göra något (?) med de AD-omvandlade värdena (som också tar tid), så blir den teoretiskt högsta samplingsfrekvensen:

en kanal

$$f_{S \max} = \frac{1}{52\mu\text{s}} = 19,2 \text{ kHz}$$

AD-omvandlingen tar tid

Om man omväxlande omvandlar två kanaler (stereo?) tillkommer omställning/inställningstiden för samplingskondensatorn $T_{ACQ} = 5 \mu\text{s}$.

två kanaler

$$f_{S_{\max}} = \frac{1}{52 + 5 + 52 + 5 [\mu\text{s}]} = 8,8 \text{ kHz}$$

PIC-processorn klarar de flesta industriella styrprocesser – men den är naturligtvis helt otillräcklig som ”signalprocessor” för tex. ljudeffekter!

Många inställningsmöjligheter!

TABLE 9-2: SUMMARY OF ASSOCIATED ADC REGISTERS

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other Resets
ADCON0	ADFM	VCFG	CHS3	CHS2	CHS1	CHS0	GO/DONE	ADON	0000 0000	0000 0000
ADCON1	—	ADCS2	ADCS1	ADCS0	—	—	—	—	-000 ----	-000 ----
ANSEL	ANS7	ANS6	ANS5	ANS4	ANS3	ANS2	ANS1	ANS0	1111 1111	1111 1111
ANSELH	—	—	—	—	ANS11	ANS10	ANS9	ANS8	---- 1111	---- 1111
ADRESH	A/D Result Register High Byte								xxxxx xxxxx	uuuu uuuu
ADRESL	A/D Result Register Low Byte								xxxxx xxxxx	uuuu uuuu
INTCON	GIE	PEIE	T0IE	INTE	RABIE	T0IF	INTF	RABIF	0000 000x	0000 000x
PIE1	—	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	-000 0000	-000 0000
PIR1	—	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	-000 0000	-000 0000
PORTA	—	—	RA5	RA4	RA3	RA2	RA1	RA0	--xx xxxxx	--uu uuuu
PORTB	RB7	RB6	RB5	RB4	—	—	—	—	xxxxx ----	uuuu ----
PORTC	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0	xxxxx xxxxx	uuuu uuuu
TRISA	—	—	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	--11 1111	--11 1111
TRISB	TRISB7	TRISB6	TRISB5	TRISB4	—	—	—	—	1111 ----	1111 ----
TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0	1111 1111	1111 1111

Legend: x = unknown, u = unchanged, — = unimplemented read as '0'. Shaded cells are not used for ADC module.

AD-omvandling – steg för steg

1. Configure Port:
 - Disable pin output driver (See TRIS register)
 - Configure pin as analog (See ANSEL register).
2. Configure the ADC module:
 - Select ADC conversion clock (ADCON1, ADCS<2:0>).
 - Configure voltage reference (ADCON0, VCFG).
 - Select ADC input channel (ADCON0, CHS<3:0>).
 - Select result format (ADCON0, ADFM).
 - Turn on ADC module (ADCON0, ADON)
3. Start conversion set the GO/DONE bit. (ADCON0, GO)
4. Wait for ADC conversion to complete, polling the GO/DONE bit. (ADCON0, GO)
5. Read ADC Result (ADRESH, ADRESL)

William Sandqvist william@kth.se