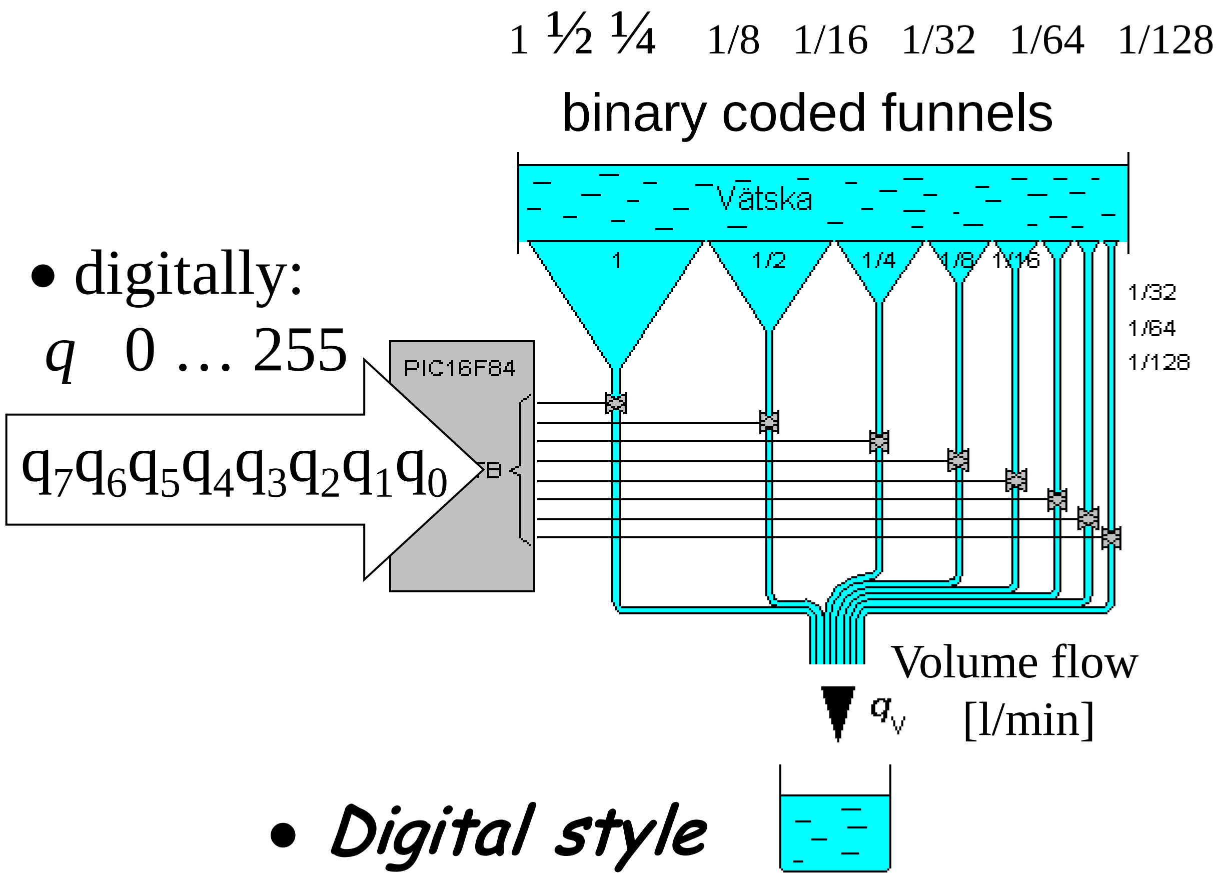
Digital or Analog

- digitally:

q 0 … 255



- *Digital style*

- or analog?

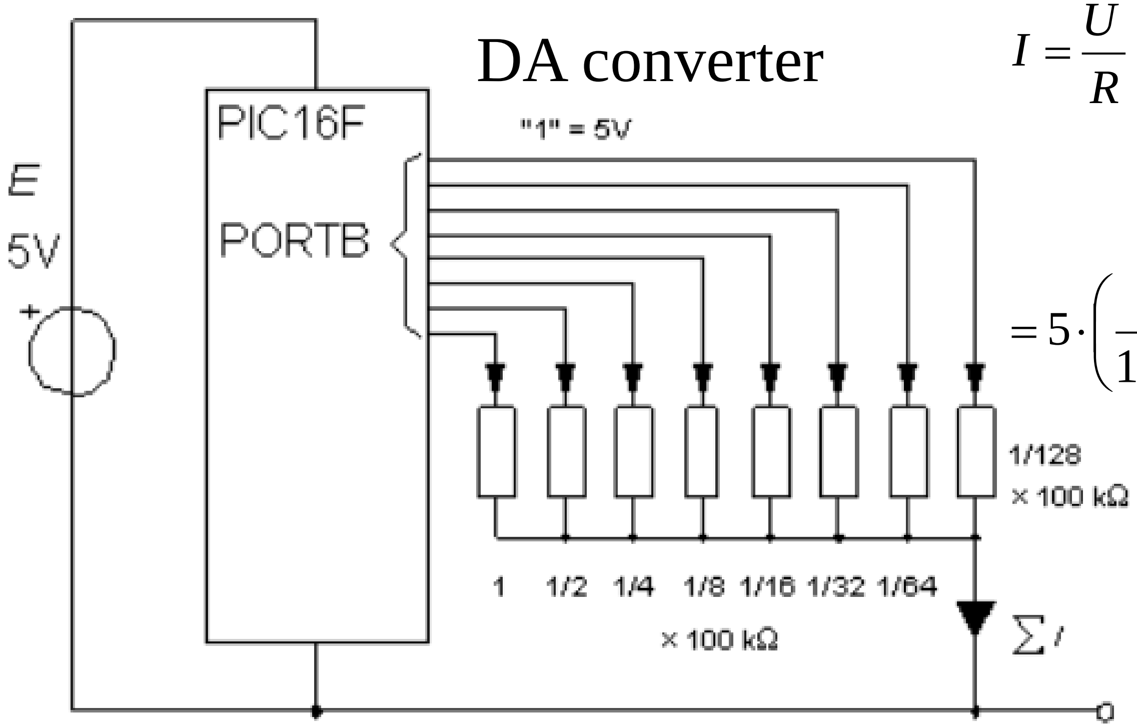
- *Old school*

Digital→Analog converter?

$b_7b_6b_5b_4b_3b_2b_1b_0$

- Binary coded resistor values

DA converter



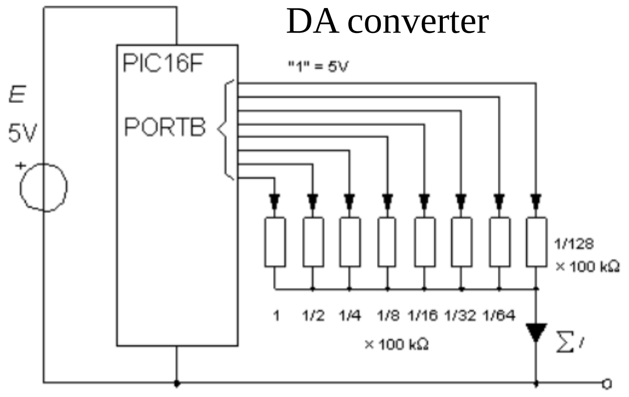
$$I = \frac{U}{R} = \frac{U}{\dfrac{1}{\left(\dfrac{1}{R_1}+\dfrac{1}{R_2}+\ldots+\dfrac{1}{R_n}\right)}}$$

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

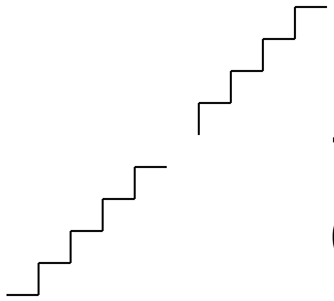
1 ½ ¼ 1/8 1/16 1/32 1/64 1/128 ×100 kΩ

The current ***I*** is the **sum** of the currents from the ”binary coded” resistors.

Digital→Analog converter?



$b_7b_6b_5b_4b_3b_2b_1b_0$



256 steps

0,5 μA/step

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

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

Problems with tolerances

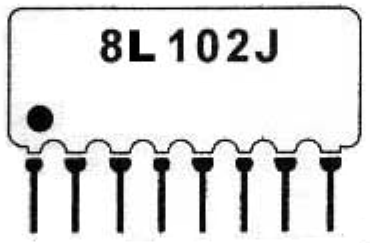
Binary coded resistors for 8 bit.

Biggest resistor *exactly* **100000 Ω** and smallest resistor *exactly* **781 Ω** (preferably 781,25)?

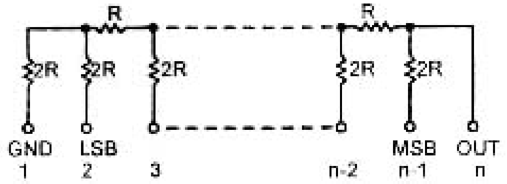
It is difficult to manufacture such various resistors with tight tolerances.

- There is a better solution!

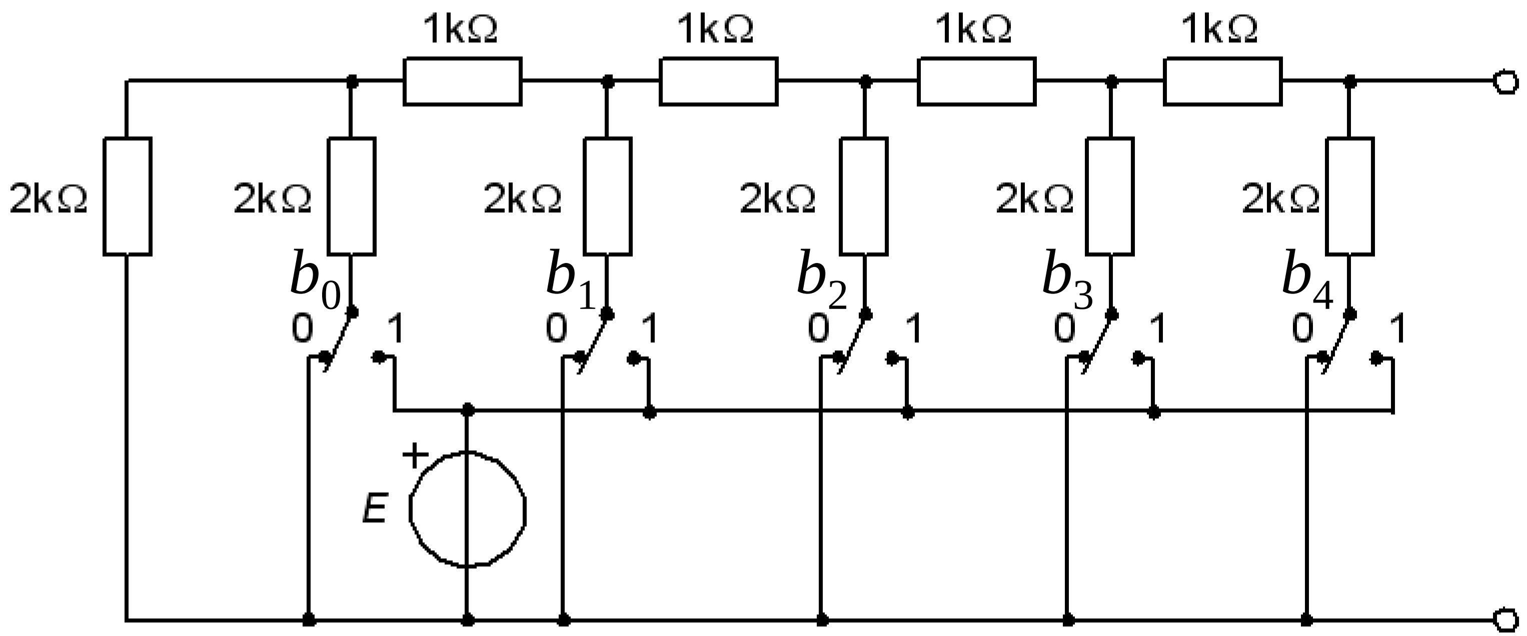
R2R- method.



R-2R-ladder



$b_4b_3b_2b_1b_0$



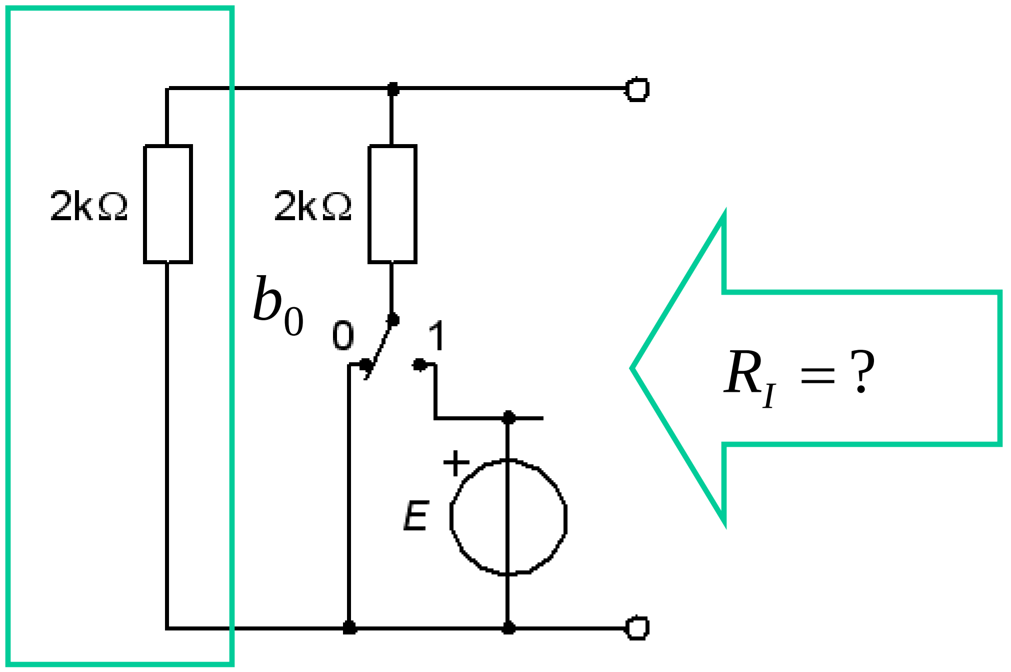
Just one resistance value needs to be manufactured, *R*, and then *R*+*R* =2*R*. One must be able to produce many "equal" resistors - the exact value is no longer important.

Two-terminal equivalent R_I

The R2R-ladder is not that easy to understand …

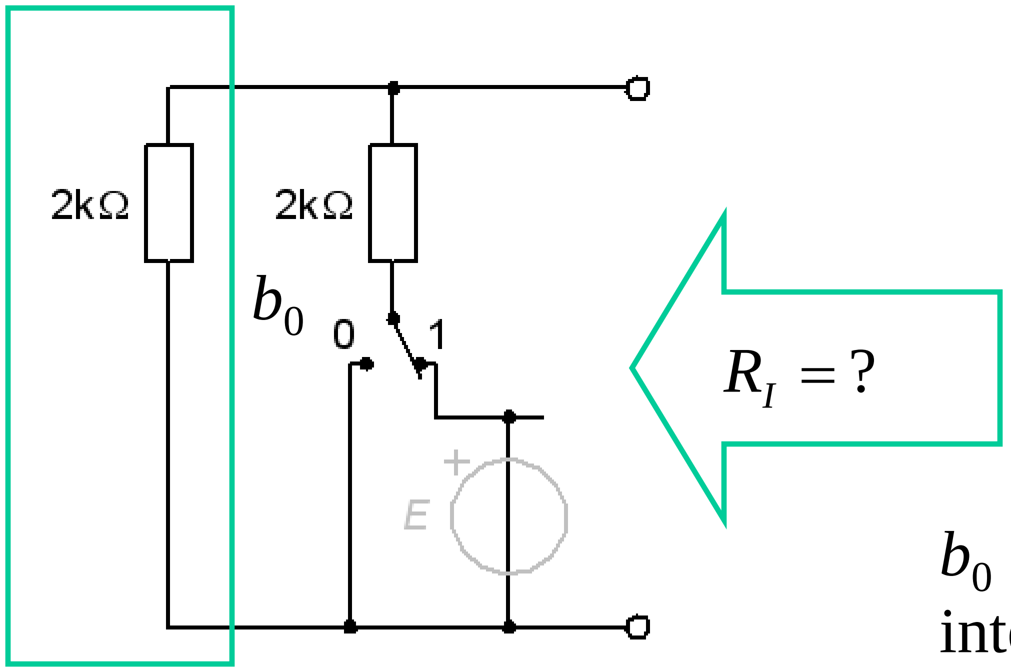
Repeated use of two-terminal equivalents and the superposition rule is what is required.

R-2R b_0=0 R_I =?



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

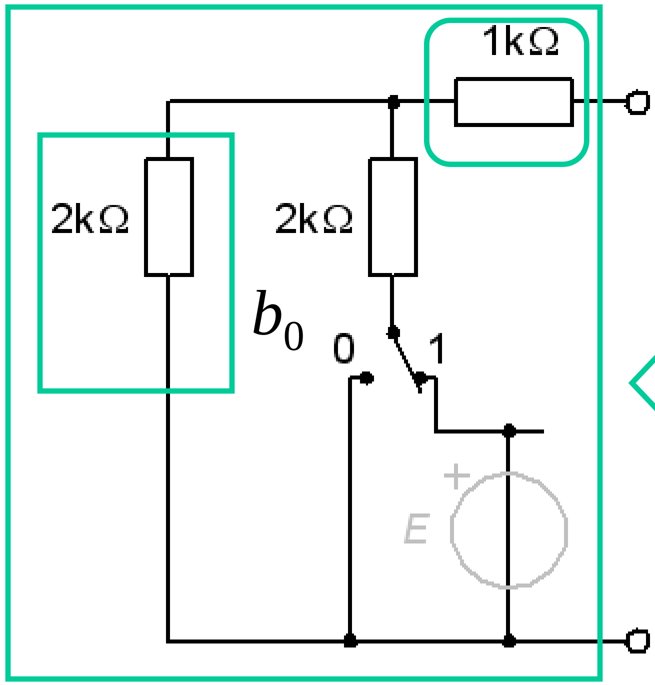
R-2R b_0=1 R_I =?



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

b_0 1 or 0, same internal resistance!

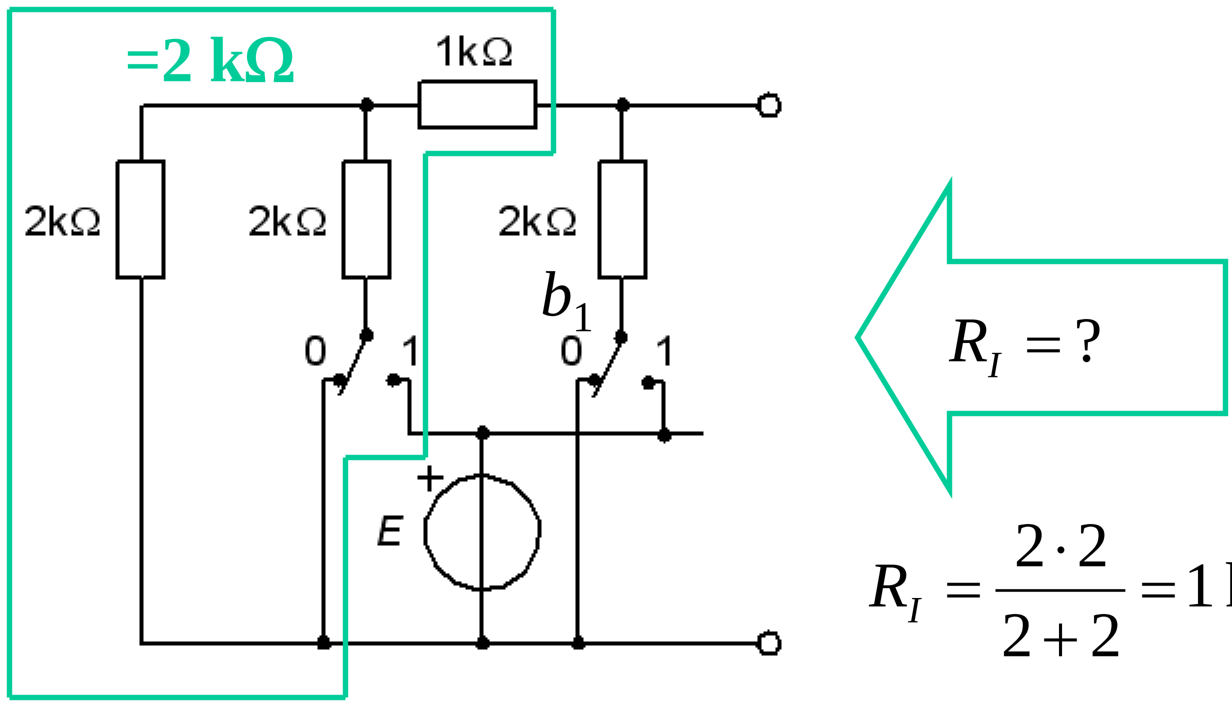
R-2R b_0 $R_\mathrm{I} = ?$



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

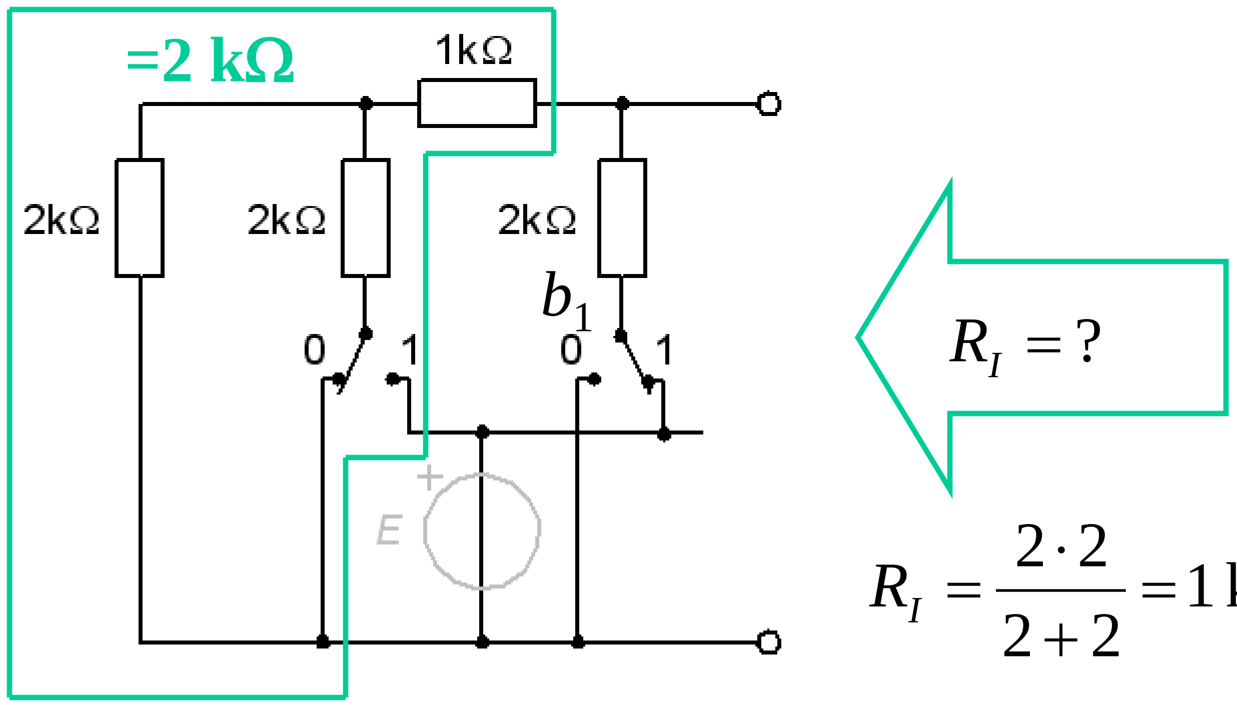
The total resistance of one stage becomes 2 kΩ.

R-2R b_1=0 R_I =?



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

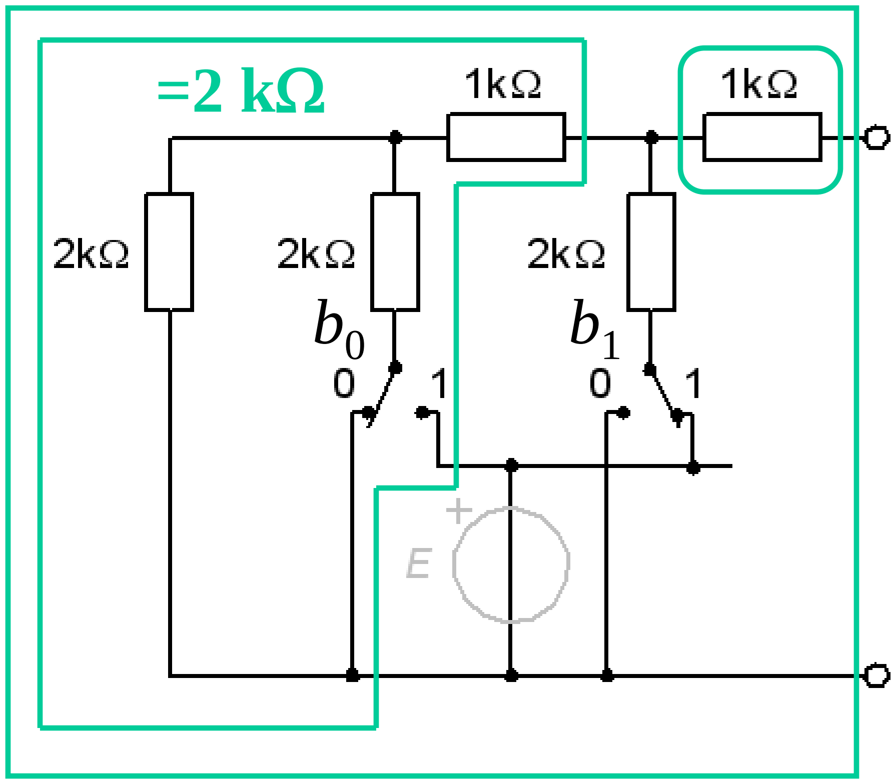
R-2R b_1=1 R_I =?



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

b_1 1 or 0, same internal resistance!

R-2R b_1 $R_I = ?$



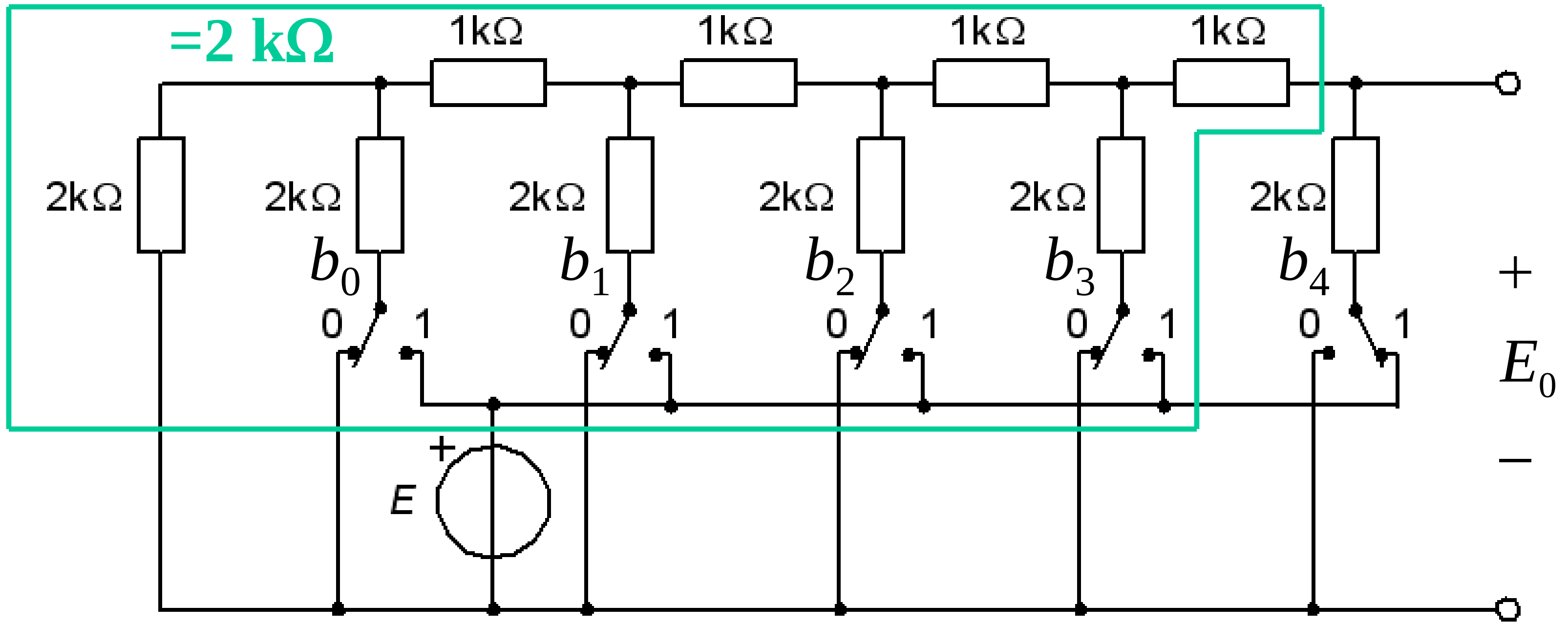
- Conclusion:

$R_I = 1 + 1 = 2$ kΩ

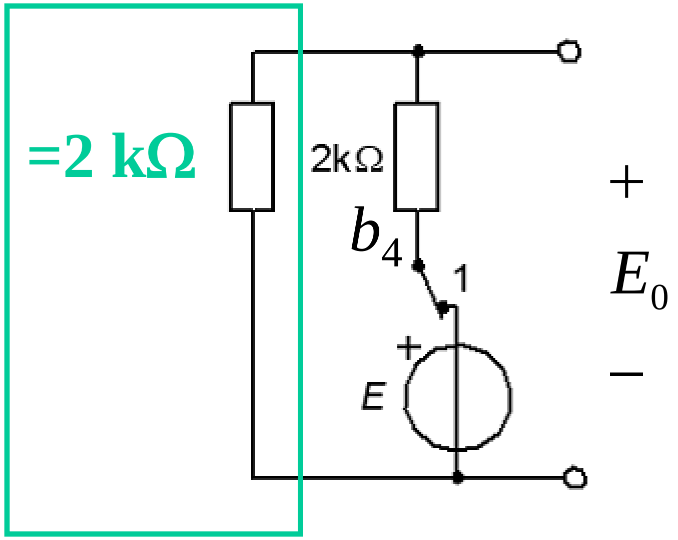
Regardless 1's or 0's is the total input resistance from the previous stages always 2 kΩ

Two-terminal equivalent E_0

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


=2 kΩ
1kΩ
1kΩ
1kΩ
1kΩ
2kΩ
2kΩ
2kΩ
2kΩ
2kΩ
2kΩ
b_0
b_1
b_2
b_3
b_4
0
1
0
1
0
1
0
1
0
1
+
E_0
−
+
E

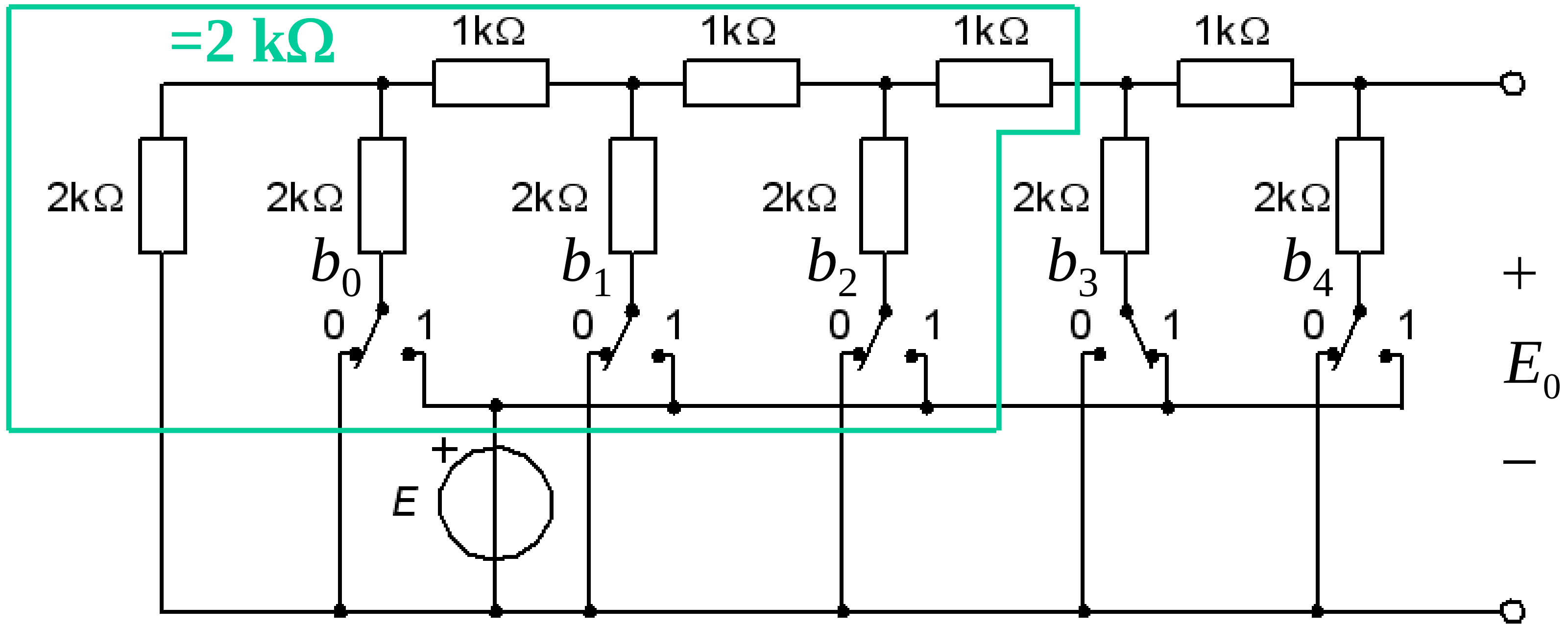
R-2R b_4=1 E_0=?



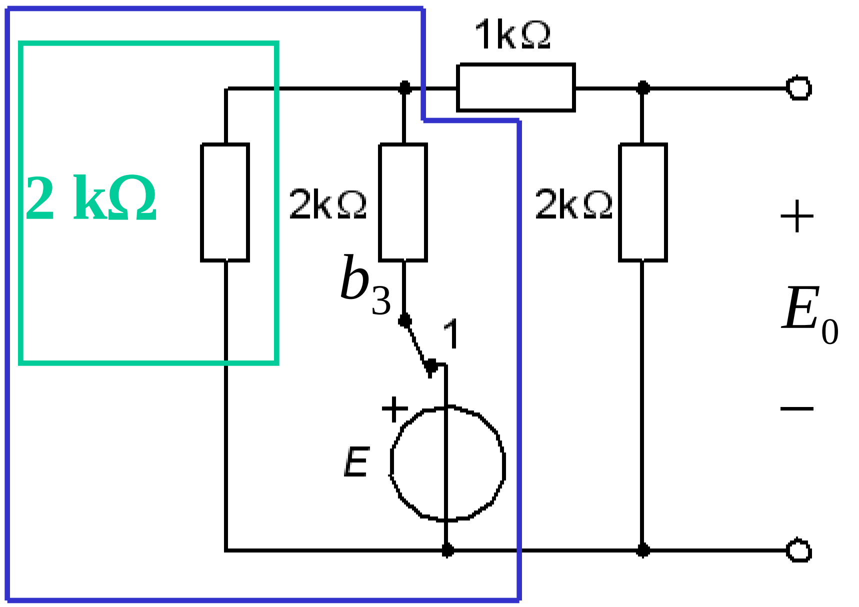
Volage divider:

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

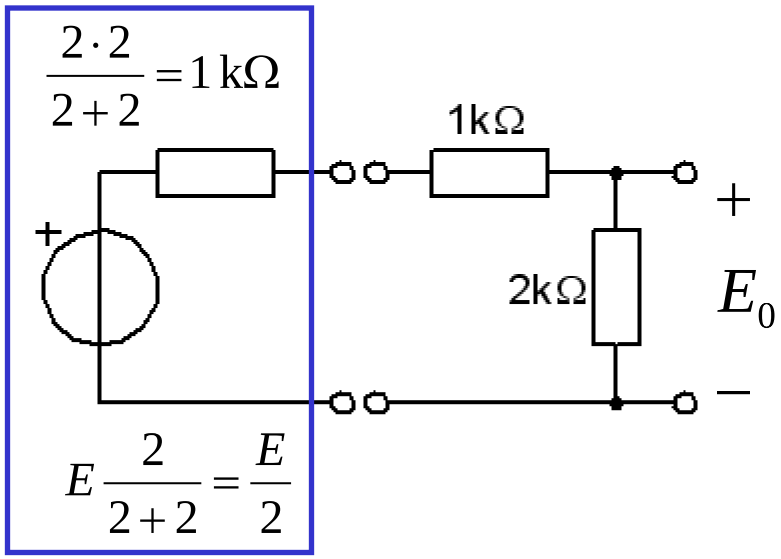
R-2R b_3=1 E_0=?



R-2R b_3=1 E_0=?

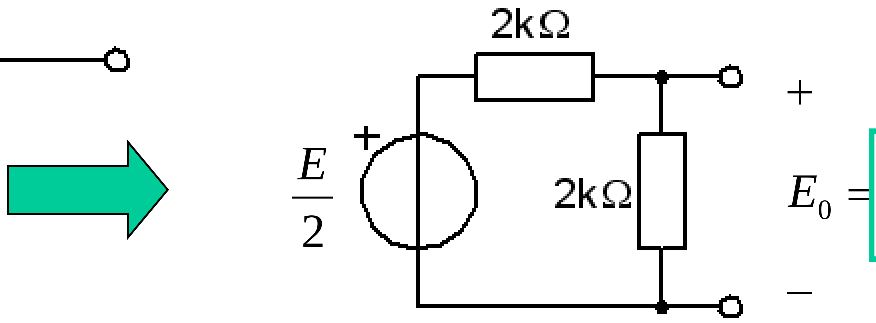


$\equiv$



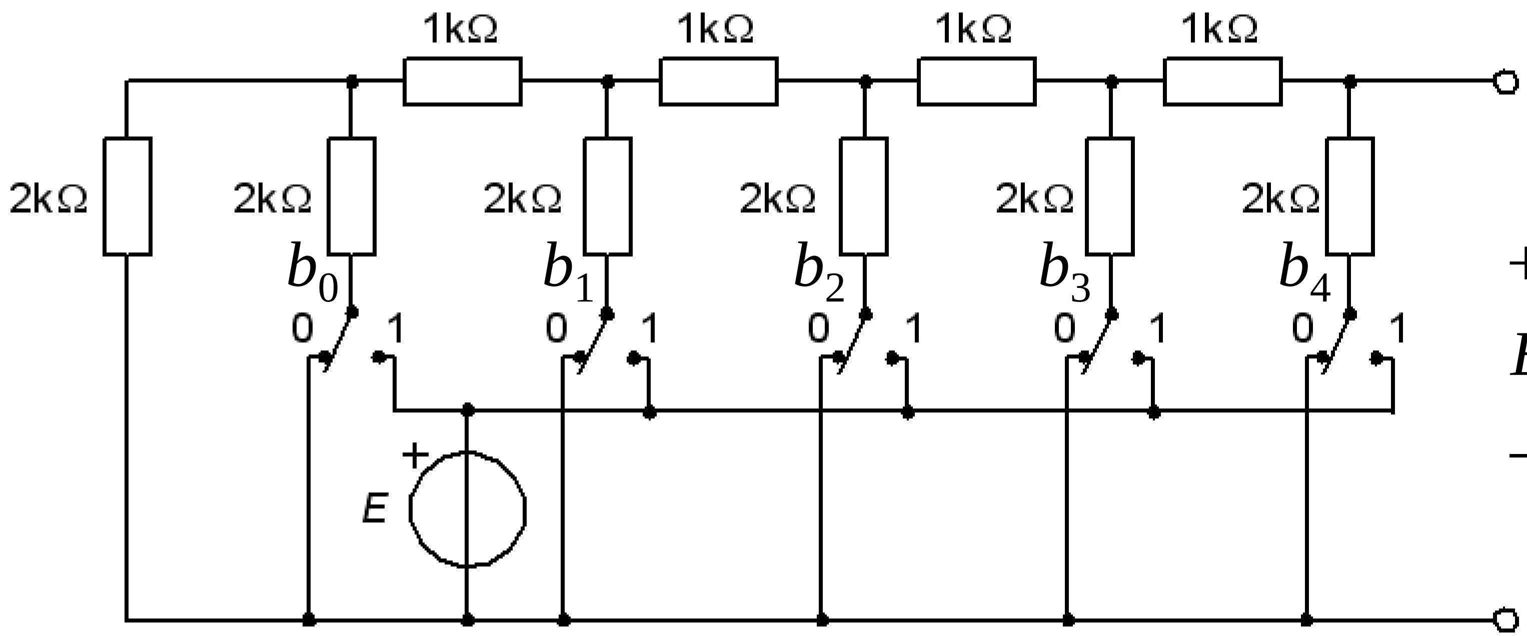
$$\frac{2\cdot 2}{2+2}=1\,\text{k}\Omega$$

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



$$E_0=\boxed{\frac{E}{4}}$$

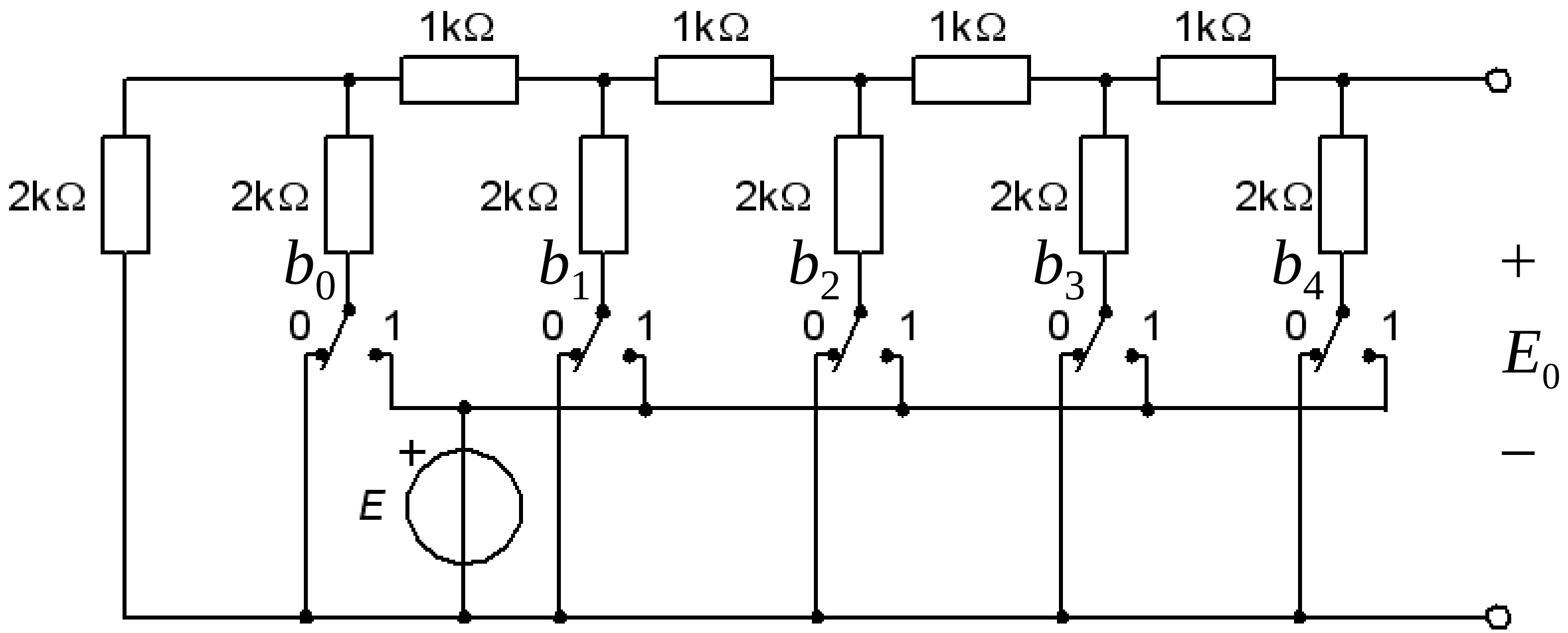
R-2R conclusion E_0 …



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

- Reasonable guess – is it not?

R-2R superposition

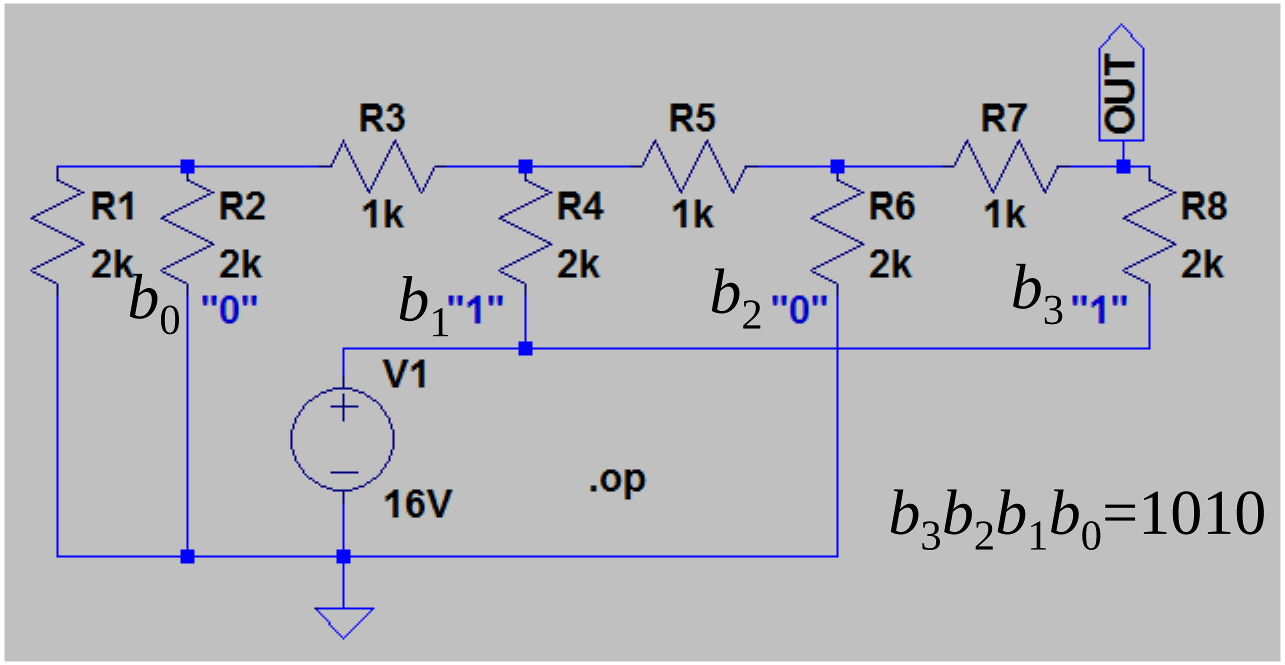


- According to the **superposition** principle, the contributions of $b_4\ b_3\ b_2\ \ldots\ b_0$ can be added individually:

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

We have a DA-converter!

R-2R simulation

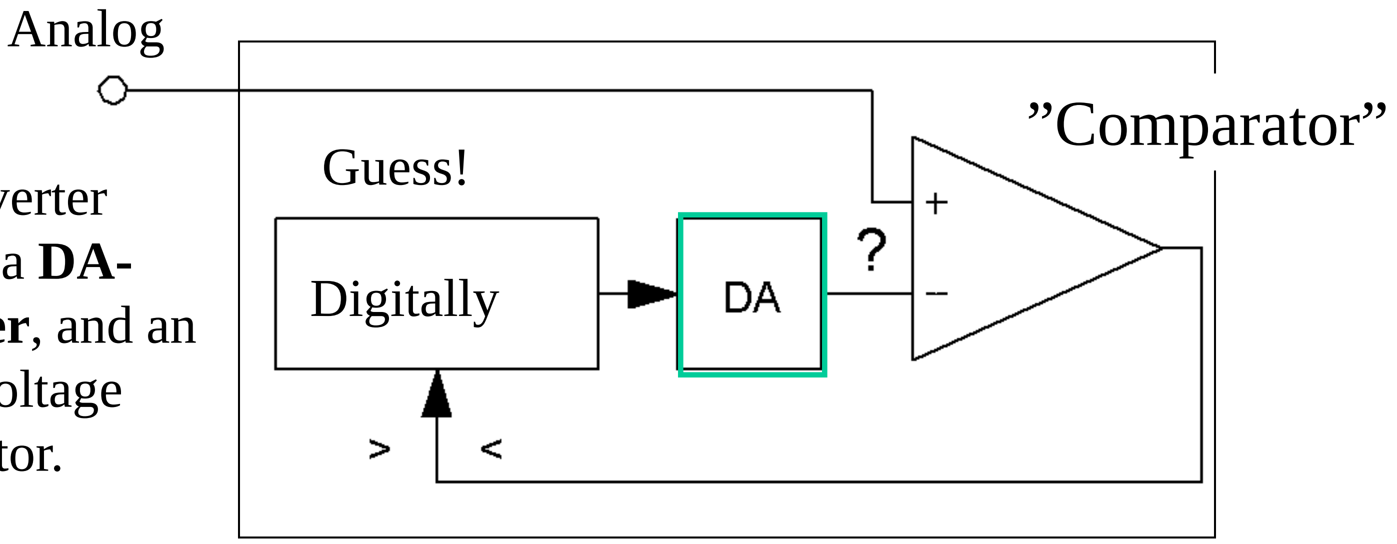


Guess value of voltage OUT?

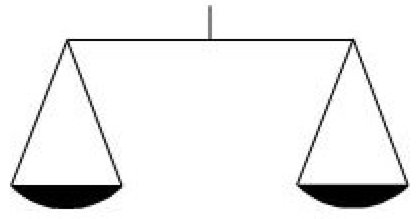
AD-converter?

Successive approximations

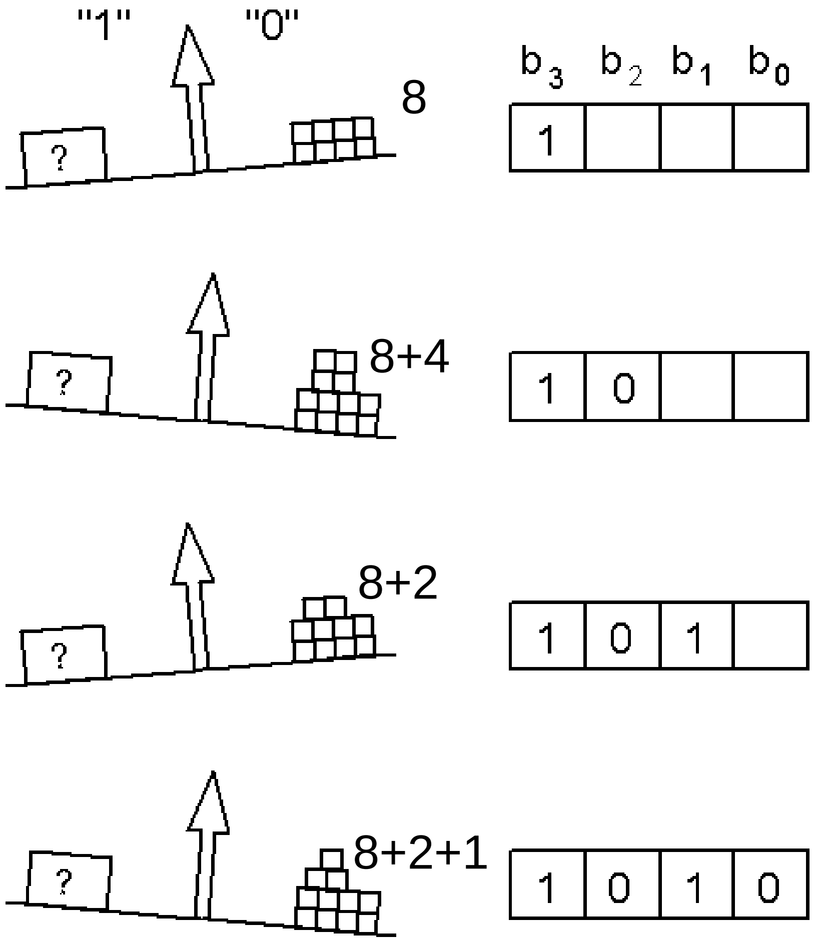
AD-converter contains a **DA-converter**, and an analog voltage comparator.

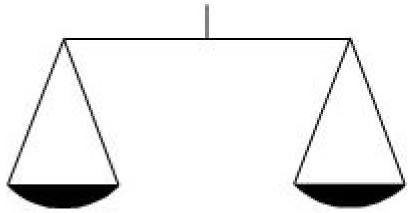


AD conversion according to the method of successive approximations is comparable to weigh an unknown mass with binary weights on a balance. We try step by step to adding binary ”weight” if ”<” or remove ” weight” if ”>”.



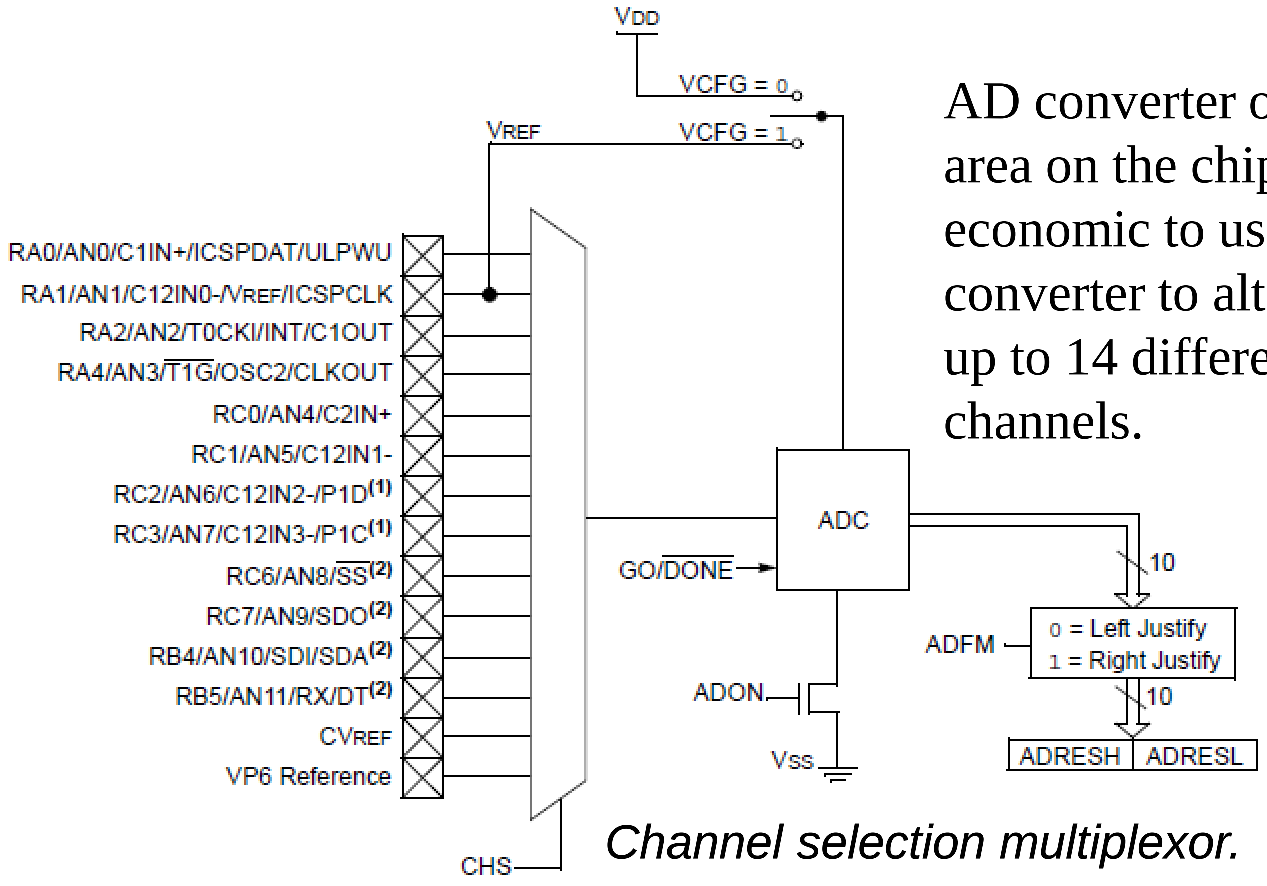
Binary weights 8 4 2 1 …





AD conversion takes time. For each bit higher resolution a comparison one step further is required.

An AD-converter, 14 channels

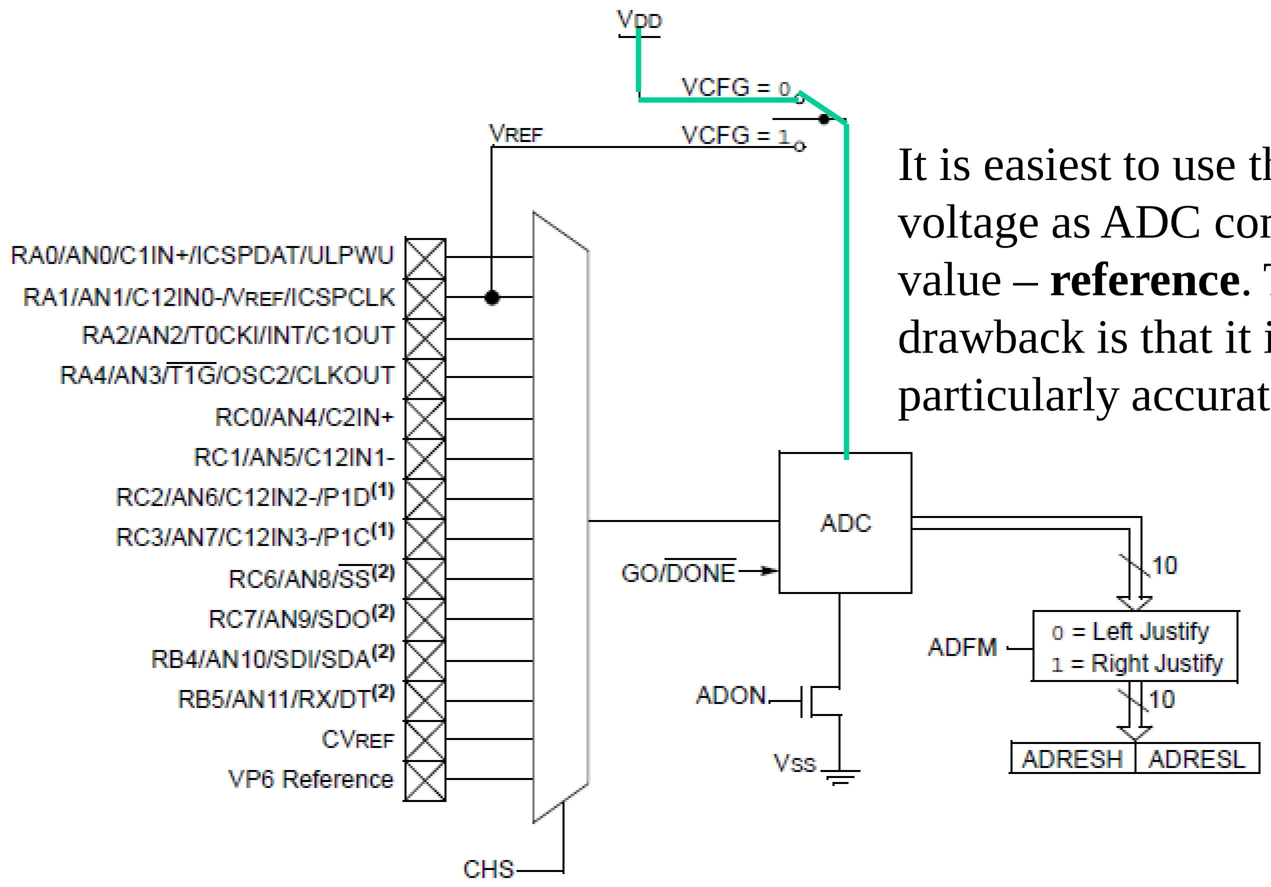


Channel selection multiplexor.

AD converter occupies a large area on the chip - so it is economic to use the same converter to alternately measure up to 14 different sources, channels.

The PIC-processor has a 10-bits AD-converter

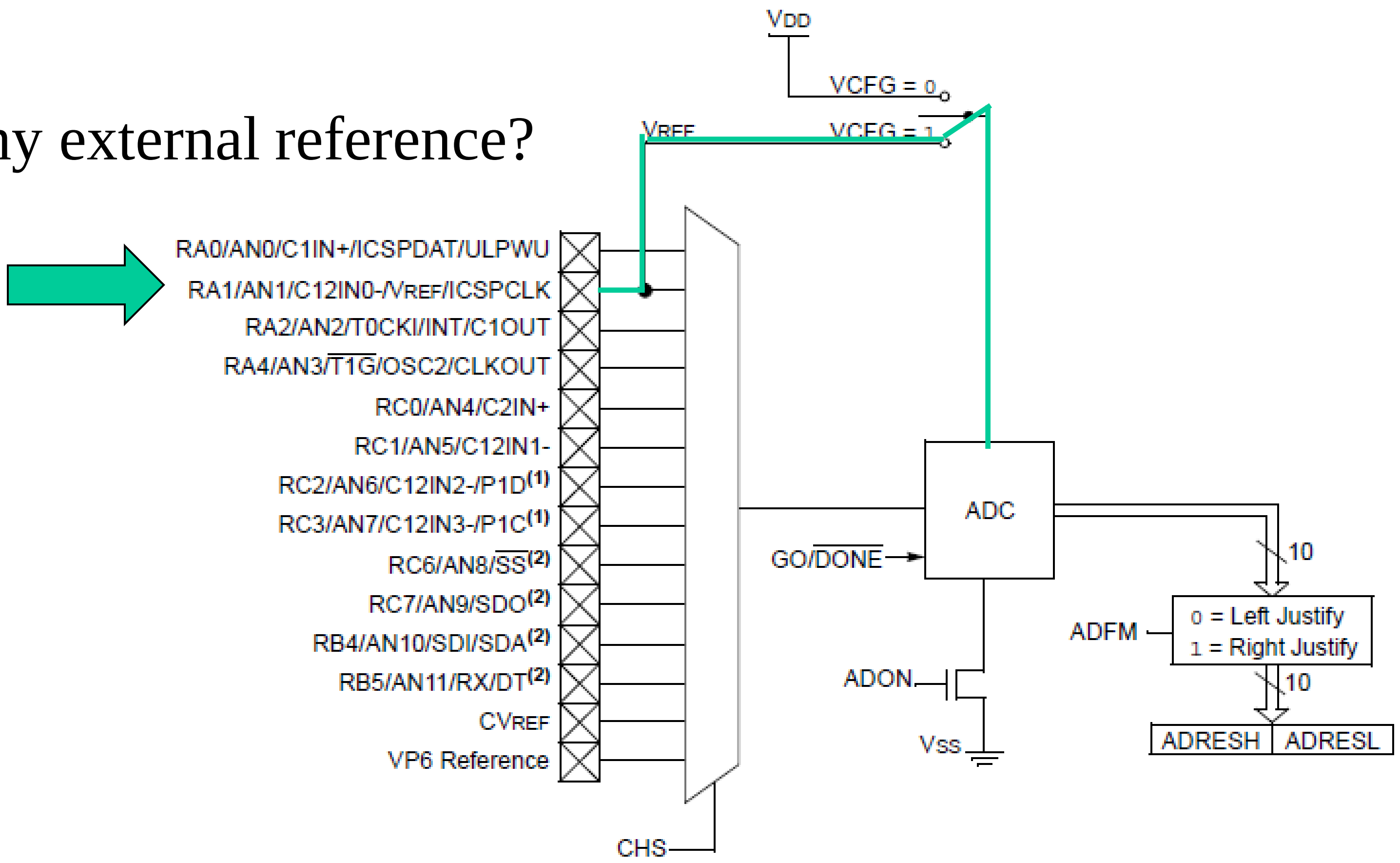
Supply voltage as reference



It is easiest to use the supply voltage as ADC comparison value – **reference**. The drawback is that it is not particularly accurate.

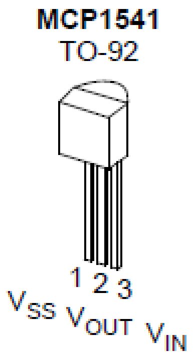
Internal or external reference?

Why external reference?

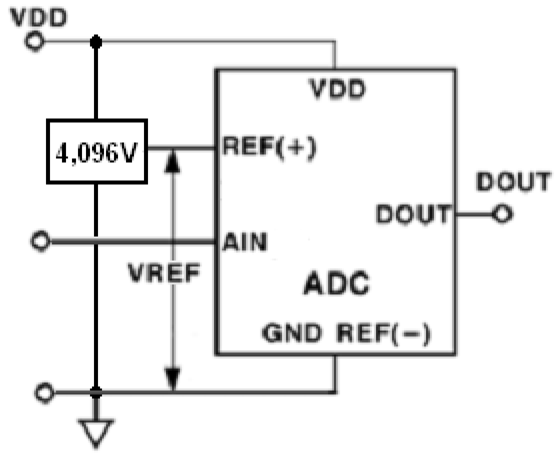


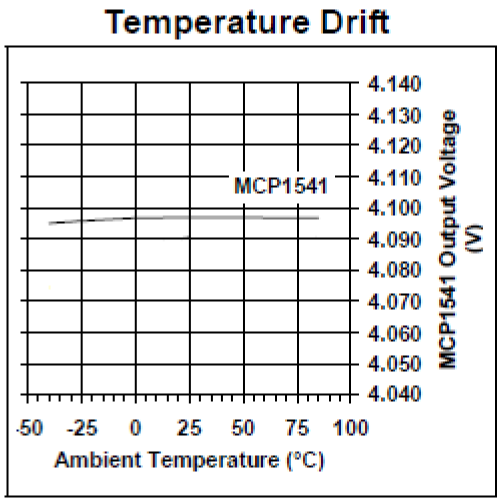
William Sandqvist william@kth.se

Stabilized reference 4,096 V



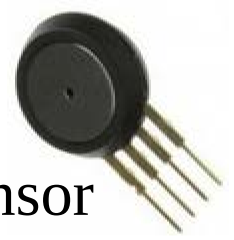
MCP1541

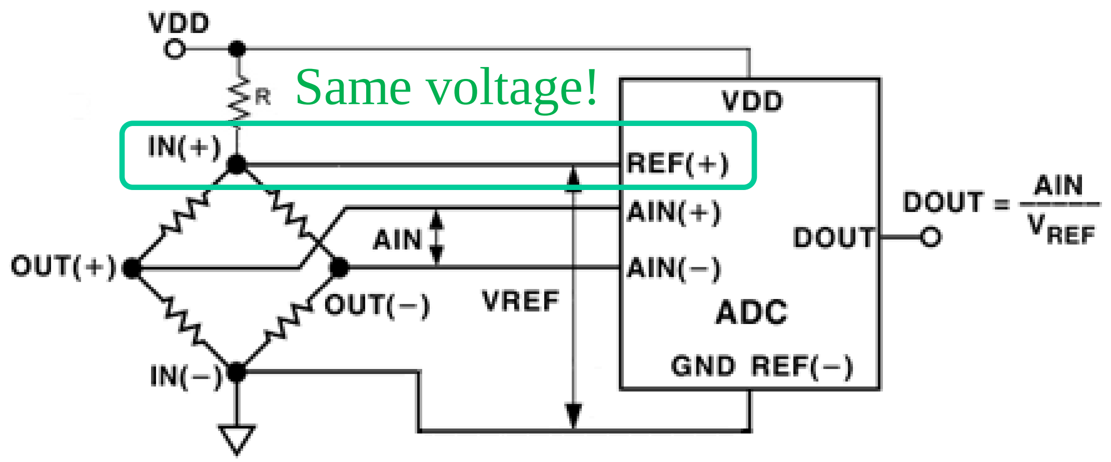




If you buy a stabilized reference circuit you can perhaps choose the value 4,096 V ($4096=2^{12}$) which gives a 10-bit AD-converter exact 4 mV-steps, *without* the need to scale the measured result with multiplications and divisions.

Ratiometric connection

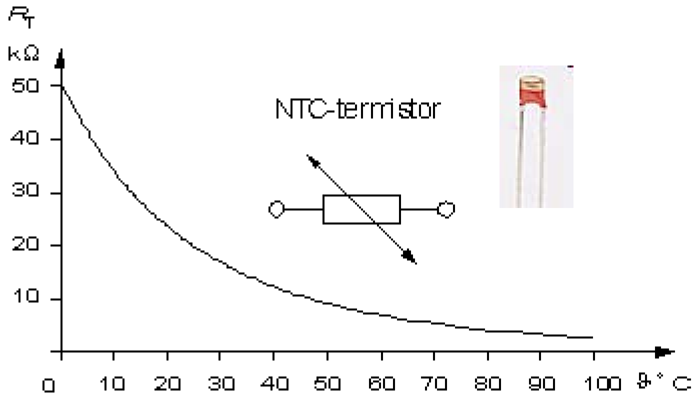

sensor

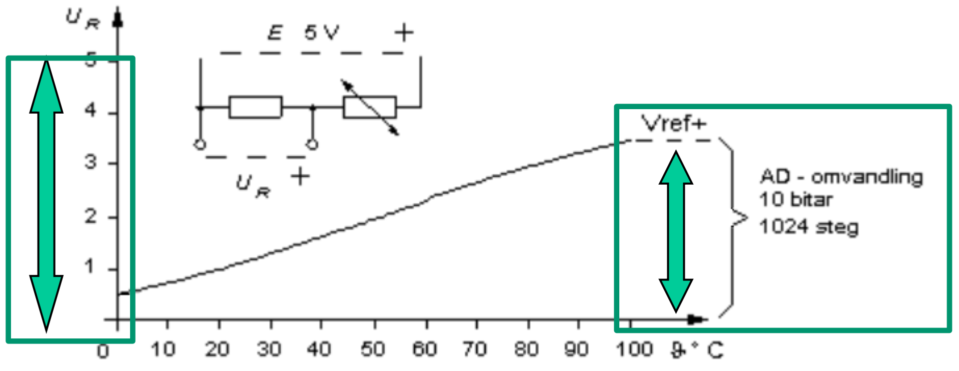


If a sensor measurement value depends on it's supply voltage, you can either have stabilized supply (expensive) – or easier, use so-called ratiometric connection. If the sensor supply voltage and AD converter reference voltage **are the same**, then changes in this voltage will be the same for both, and the AD converted measured value will remain intact!

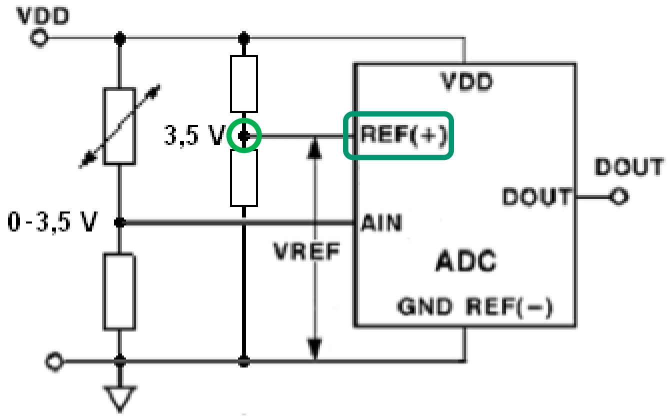
Adapt measuring range

An NTC thermistor has a high sensitivity but a non-linear temperature relationship.

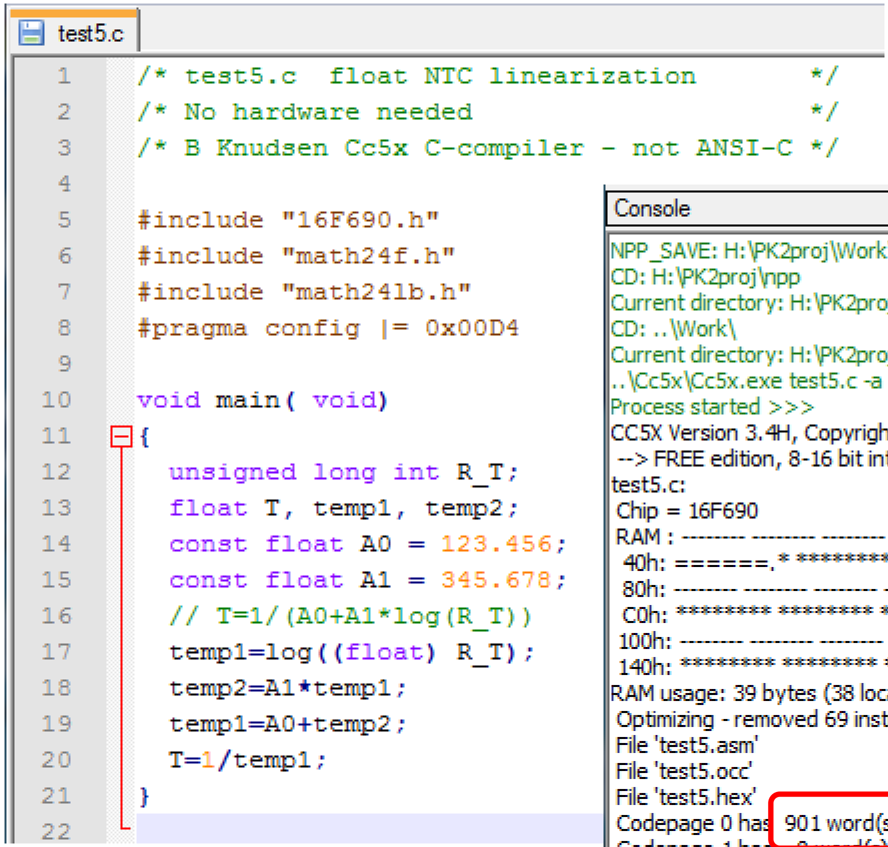




We can linearize with a resistor – and then get the measuring range 0 …3,5 V. If the reference is 3,5 V instead of 5 V then one utilizes the entire ADC range for the measurement.



Or we program the linearization



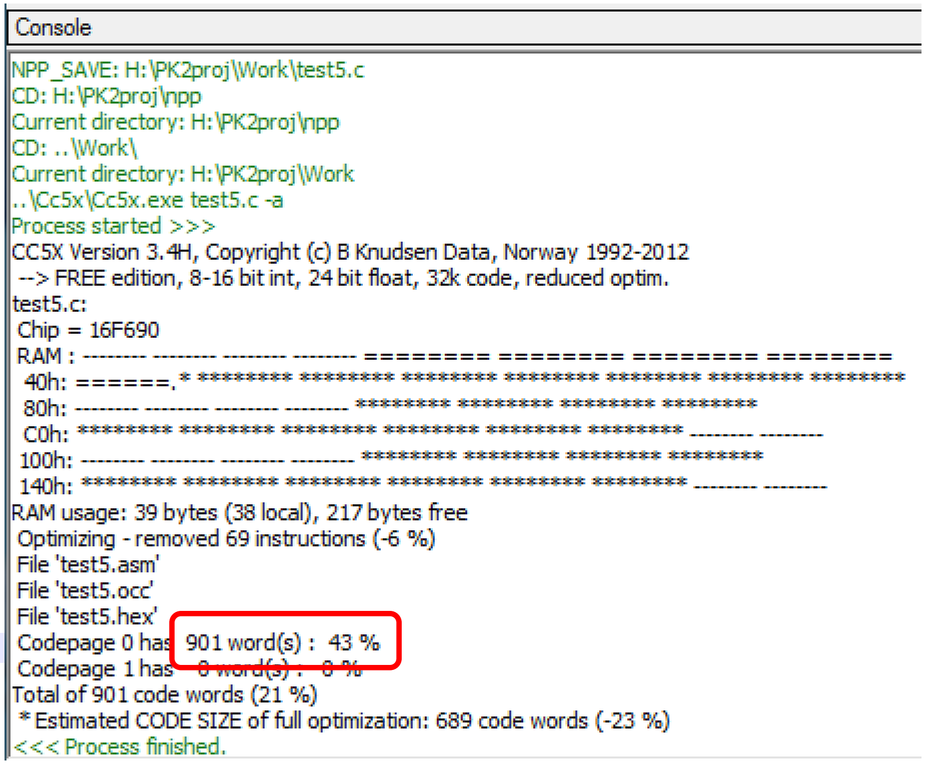
```
test5.c
/* test5.c  float NTC linearization          */
/* No hardware needed                        */
/* B Knudsen Cc5x C-compiler - not ANSI-C */

#include "16F690.h"
#include "math24f.h"
#include "math24lb.h"
#pragma config |= 0x00D4

void main( void)
{
  unsigned long int R_T;
  float T, temp1, temp2;
  const float A0 = 123.456;
  const float A1 = 345.678;
  // T=1/(A0+A1*log(R_T))
  temp1=log((float) R_T);
  temp2=A1*temp1;
  temp1=A0+temp2;
  T=1/temp1;
}

```

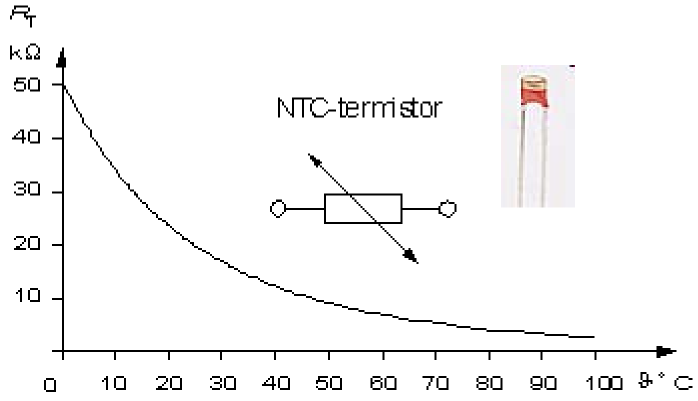
$$T = \frac{1}{a_0 + a_1 \cdot \ln\left(R_T\right)}$$

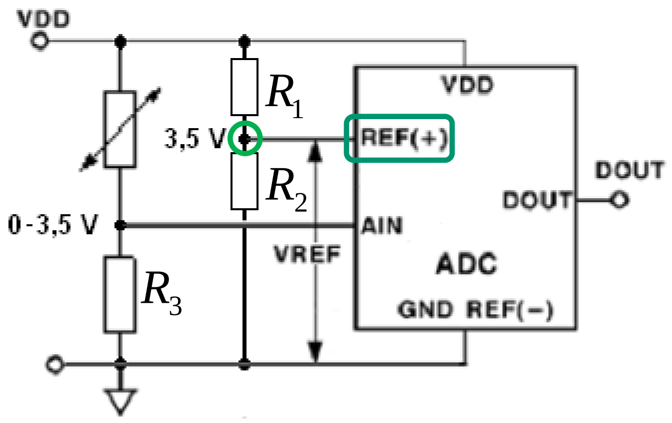


```
Console
NPP_SAVE: H:\PK2proj\Work\test5.c
CD: H:\PK2proj\npp
Current directory: H:\PK2proj\npp
CD: ..\Work\
Current directory: H:\PK2proj\Work
..\Cc5x\Cc5x.exe test5.c -a
Process started >>>
CC5X Version 3.4H, Copyright (c) B Knudsen Data, Norway 1992-2012
 --> FREE edition, 8-16 bit int, 24 bit 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.
```

There exists a choice … ??

There is apparently a choice between buying **three resistors** – or to use an **900 instruction** program.



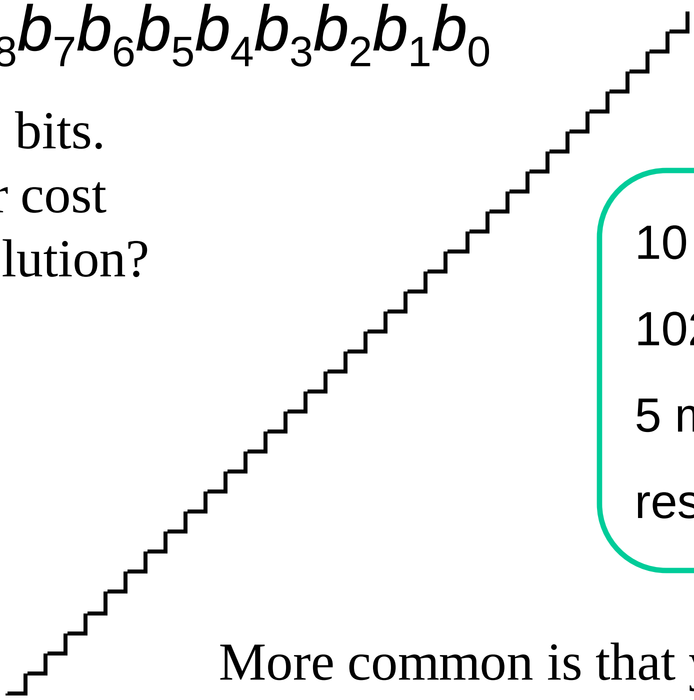


Price for one 1 kr

For those of you who knows about Embedded Electronics …

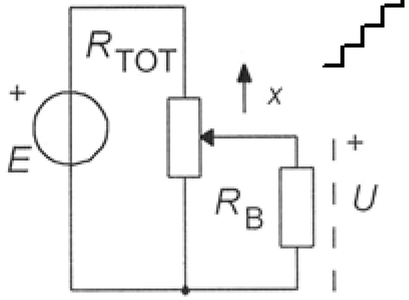
Are 10-bit resolution required?

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



AD converter is 10 bits.
What does a sensor cost
that has 10-bit resolution?

100 $?



10 bits

1024 steps

5 mV/step

resolution ≈ 0,1%

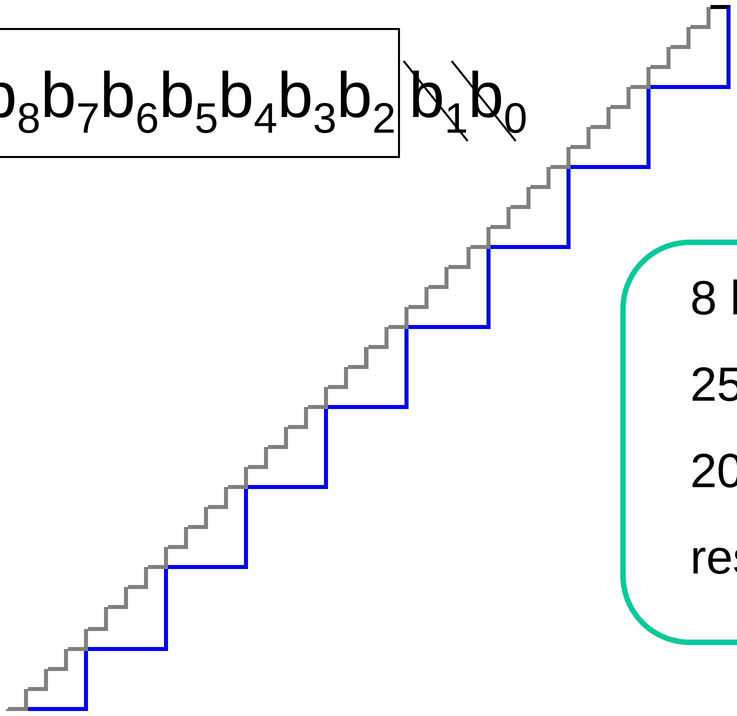
More common is that you can afford an 8-bit sensor. (PIC processor itself costs 2 $).

The picture shows a resistive position sensor that can take advantage of 10-bit resolution.

8-bit program

If one need only 8 bit resolution one can ignore the two least significant bits and handle the result as a byte.

$\boxed{b_9 b_8 b_7 b_6 b_5 b_4 b_3 b_2}\, \cancel{b_1}\cancel{b_0}$



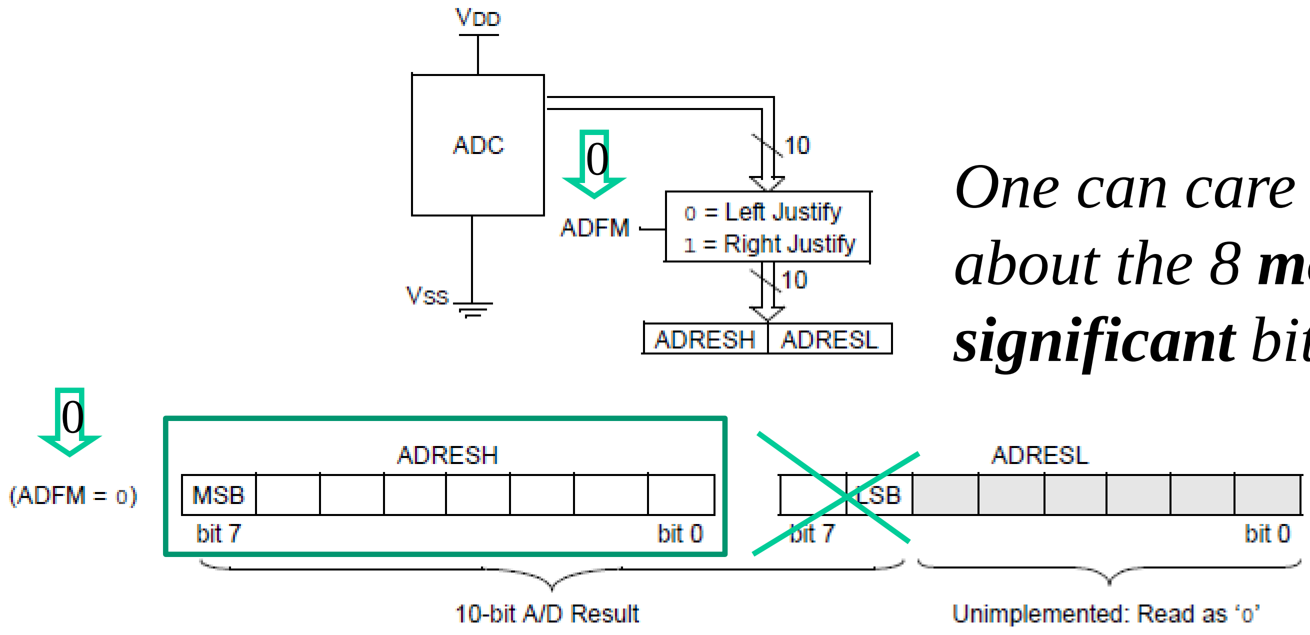
8 bit

256 steps

20 mV/step

resolution ≈1%

8-bit program



One can care only about the 8 ***most significant*** *bits!*

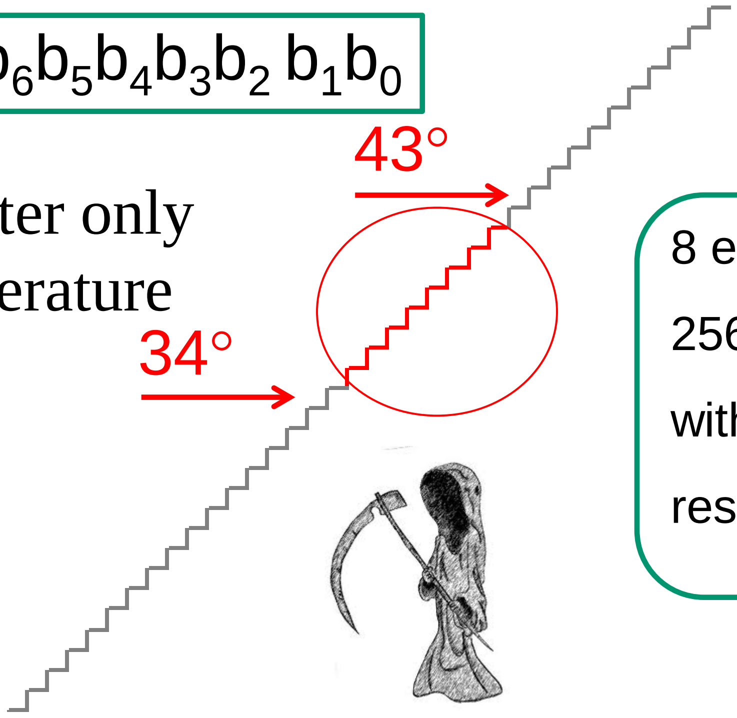
```
ADFM=0;
char value;
value = ADRESH;  /* 8-bit measurement */
```

Avoid amplifier

If one only need 8 bit resolution one can still use the 10-bit resolution to avoid the need to amplify the sensor signal, the two most significant bits becomes constant. One can therefore ignore to read them.

$\cancel{1}\,\cancel{0}\,\boxed{b_7b_6b_5b_4b_3b_2\,b_1b_0}$

A fever thermometer only need a small temperature range 34°…43° (right?)



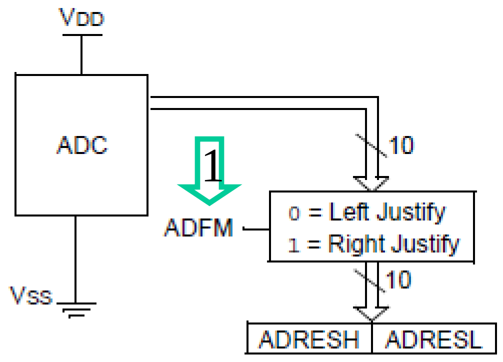
8 effective bits,

256 effective steps

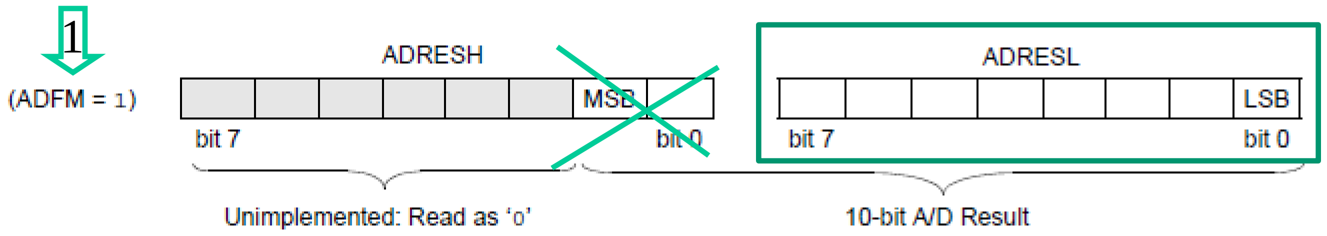
with 5 mV/step

resolution ≈1%

8-bit program (10 bit)



One can care only about the 8 ***least significant*** *bits that change!*



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

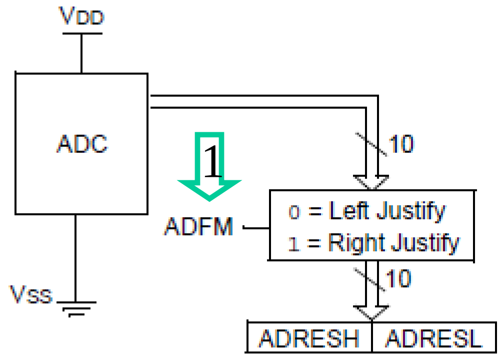
16-bit program

10-bits

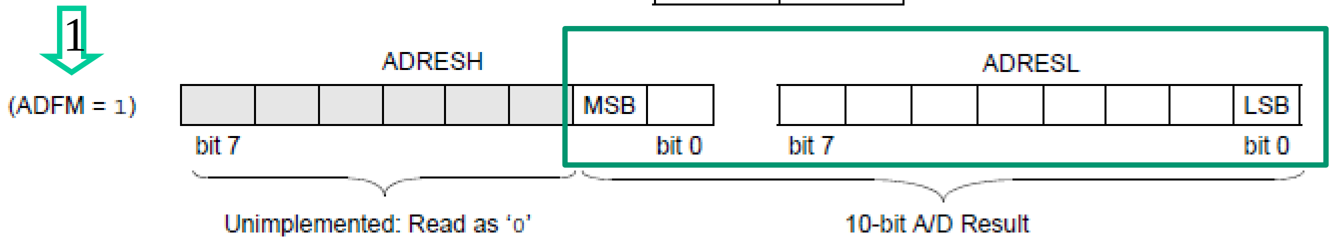
1024 steps

5 mV/step

resolution ≈ 0,1%



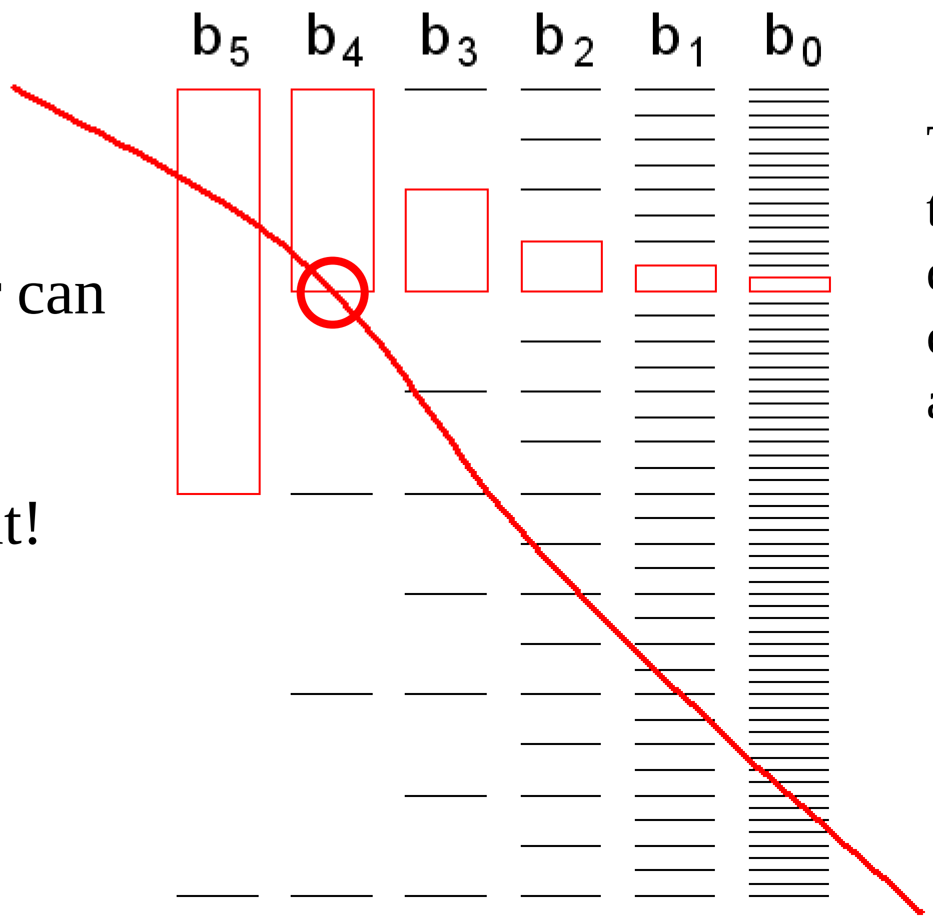
When you really need 10 bits in a 16-bit variable.



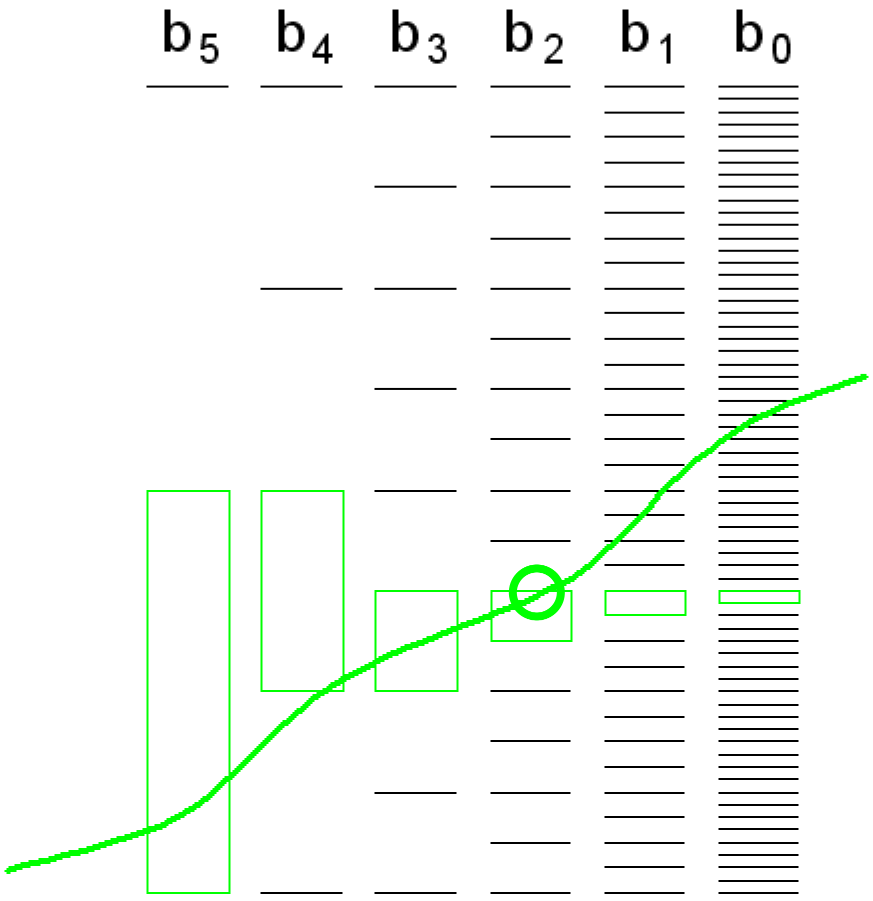
```
ADFM=1;
unsigned long value;
value =  ADRESH * 256;
value += ADRESL; /* 10-bit measurement */
```

What happens if the signal changes during conversion?

AD converter can not "undo" a previously determined bit!



The result is a value that has occurred during the conversion, but at an unspecified time!



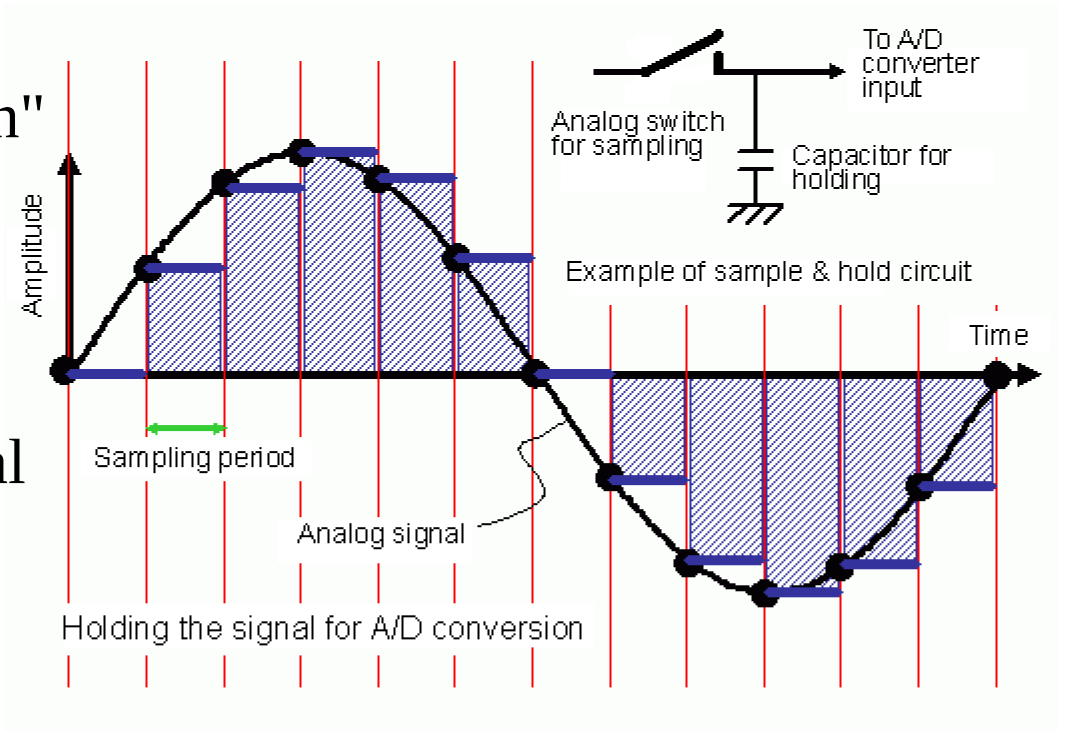
The problem

The time for the sampling is indeterminate!

Sample & Hold - circuit

The solution is that the analog signal is "frozen" during the conversion.

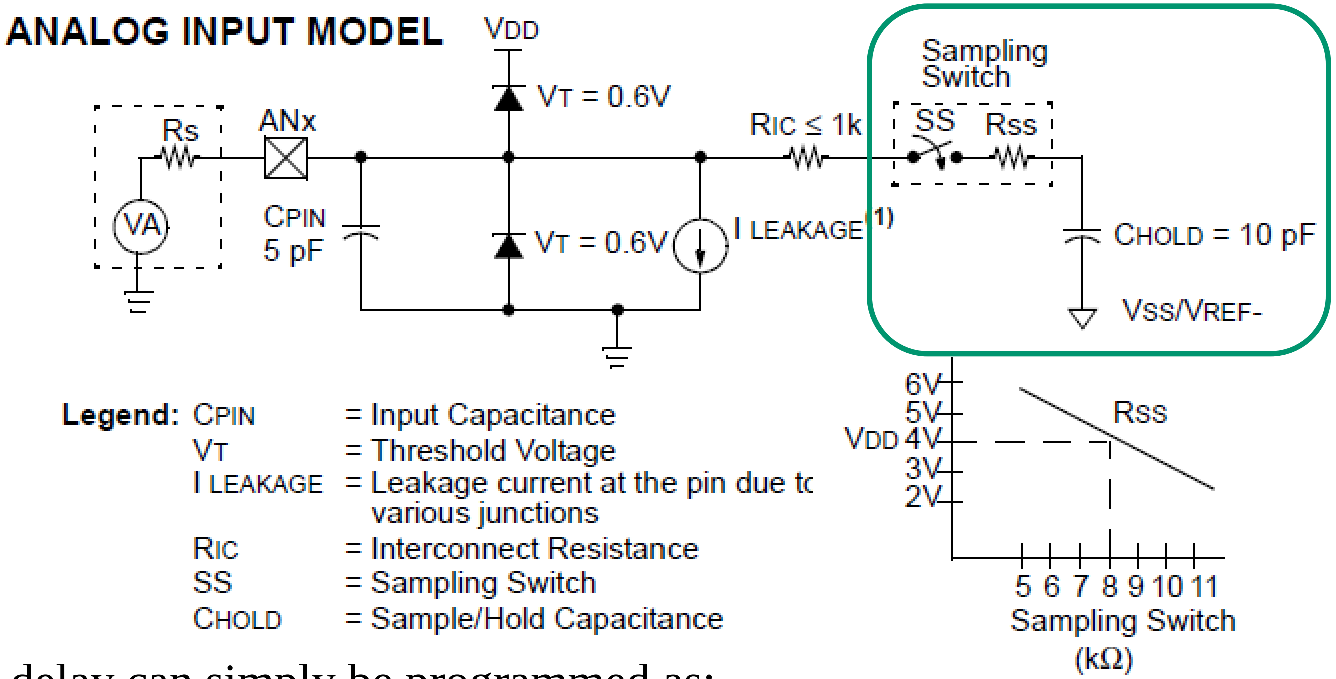
At the AD conversion start a switch is taking a "sample" of the signal and stores it in a capacitor.


To A/D converter input
Analog switch for sampling
Capacitor for holding
Amplitude
Example of sample & hold circuit
Time
Sampling period
Analog signal
Holding the signal for A/D conversion

The course PIC processor, the sampling capacitor has capacitance ≈10pF.

Acquisition time t_{ACQ}

Every time one has **chosen/changed channel** the **sampling capacitor** C_{HOLD} must have time to recharge to the analog voltage. This will take about 5 µs.



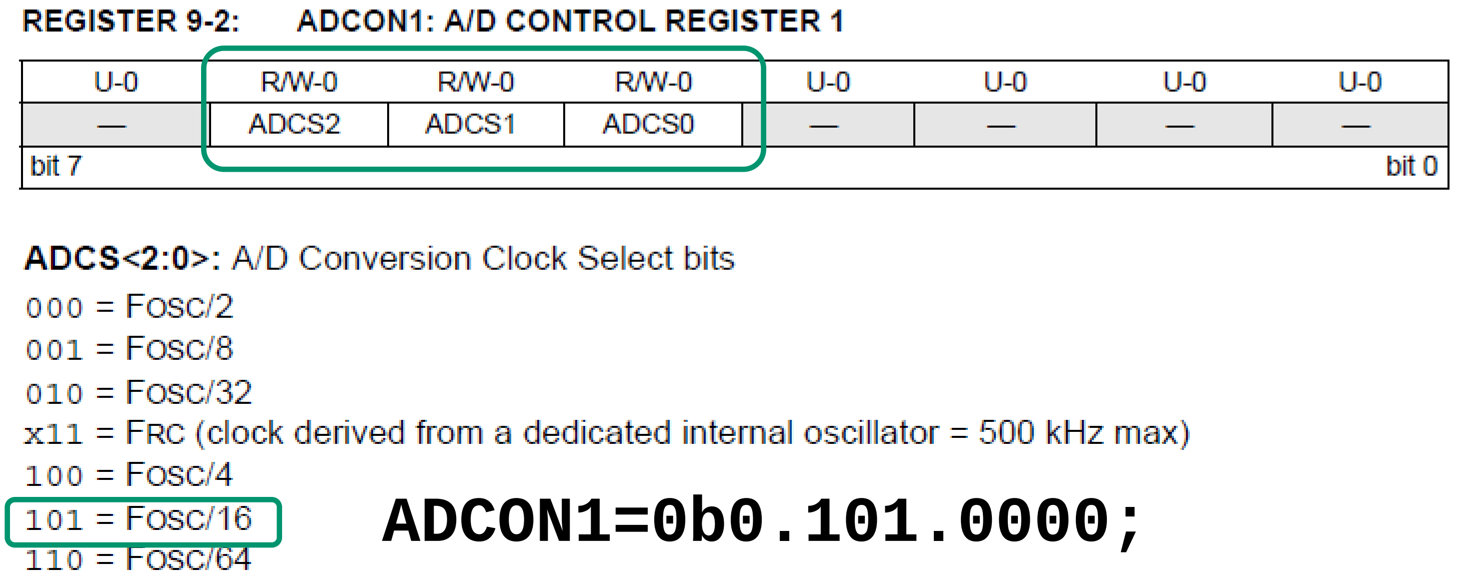
Legend:	CPIN	= Input Capacitance
	VT	= Threshold Voltage
	I LEAKAGE	= Leakage current at the pin due tc various junctions
	RIC	= Interconnect Resistance
	SS	= Sampling Switch
	CHOLD	= Sample/Hold Capacitance

A 5 µs delay can simply be programmed as:

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

AD-clock pulses

AD-converter can use a maximum clock frequency of 250 kHz. If the PIC processorn clock is **4 MHz** this must firs be divided 16 times before it can be used as AD-clock. This frequency divider is provided.



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

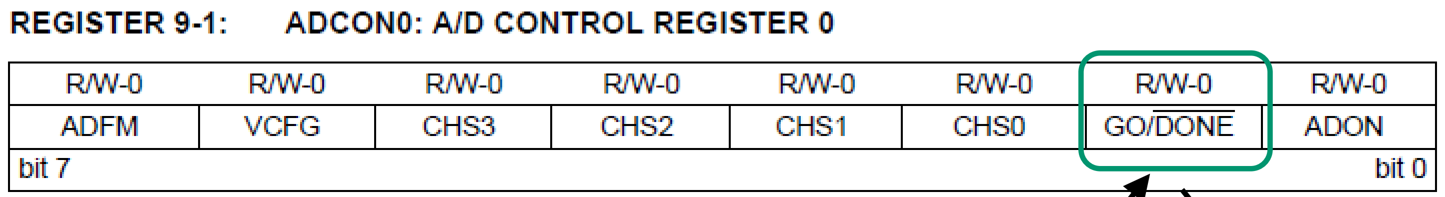
ADCS<2:0>: A/D Conversion Clock Select bits

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;`

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

Start AD and wait for done



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/$\overline{\text{DONE}}$	ADON
bit 7							bit 0

Start AD-conversion:

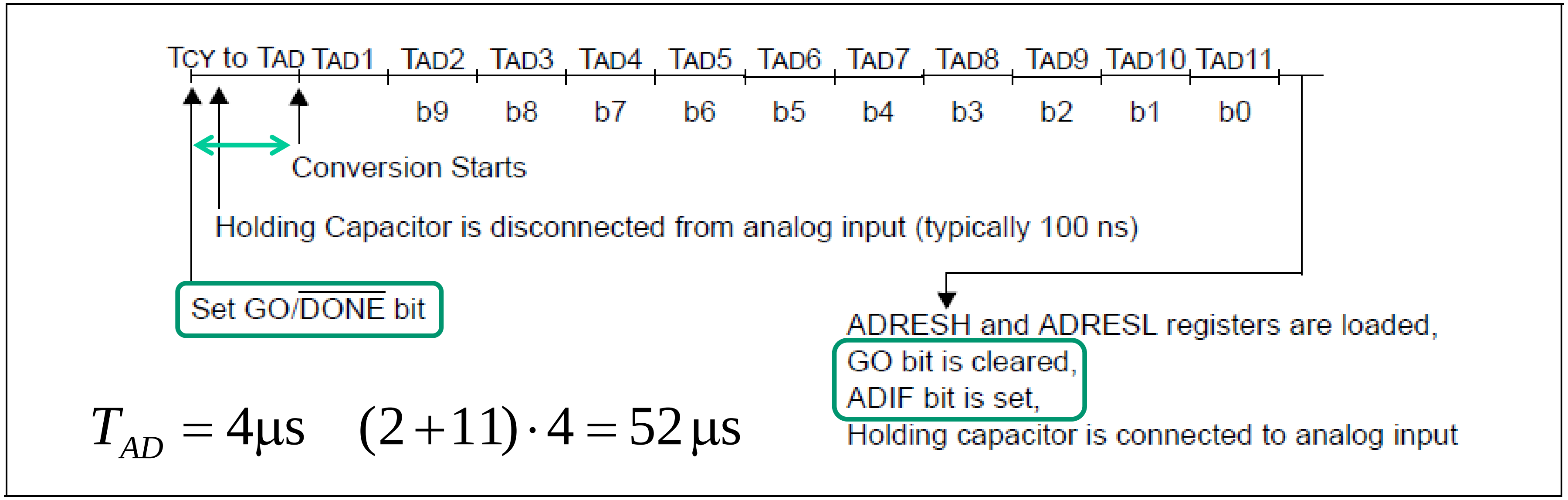
`GO = 1;`

Wait for AD-conversion done:

`while(GO) ;`

AD-conversion takes time

FIGURE 9-2: ANALOG-TO-DIGITAL CONVERSION TAD CYCLES


TCY to TAD
TAD1
TAD2
TAD3
TAD4
TAD5
TAD6
TAD7
TAD8
TAD9
TAD10
TAD11
b9
b8
b7
b6
b5
b4
b3
b2
b1
b0
Conversion Starts
Holding Capacitor is disconnected from analog input (typically 100 ns)
Set GO/DONE bit
ADRESH and ADRESL registers are loaded,
GO bit is cleared,
ADIF bit is set,
Holding capacitor is connected to analog input

$$T_{AD} = 4\mu\text{s} \quad (2+11)\cdot 4 = 52\,\mu\text{s}$$

The conversion takes approximately 2 + 11 AD clock pulses. If one ignores the fact that the PIC processor must do something (?) With the AD-converted value (which also takes time), then the theoretically maximum sampling rate becomes:

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

AD-conversion takes time

If one converts alternating two channels (stereo?) then there will also be the setting time of sampling capacitor T_{ACQ} = 5 μs.

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

The PIC processor can handle most industrial control processes - but it is of course totally inadequate as a "signal processor" for sound effects!

Many setup possibilities

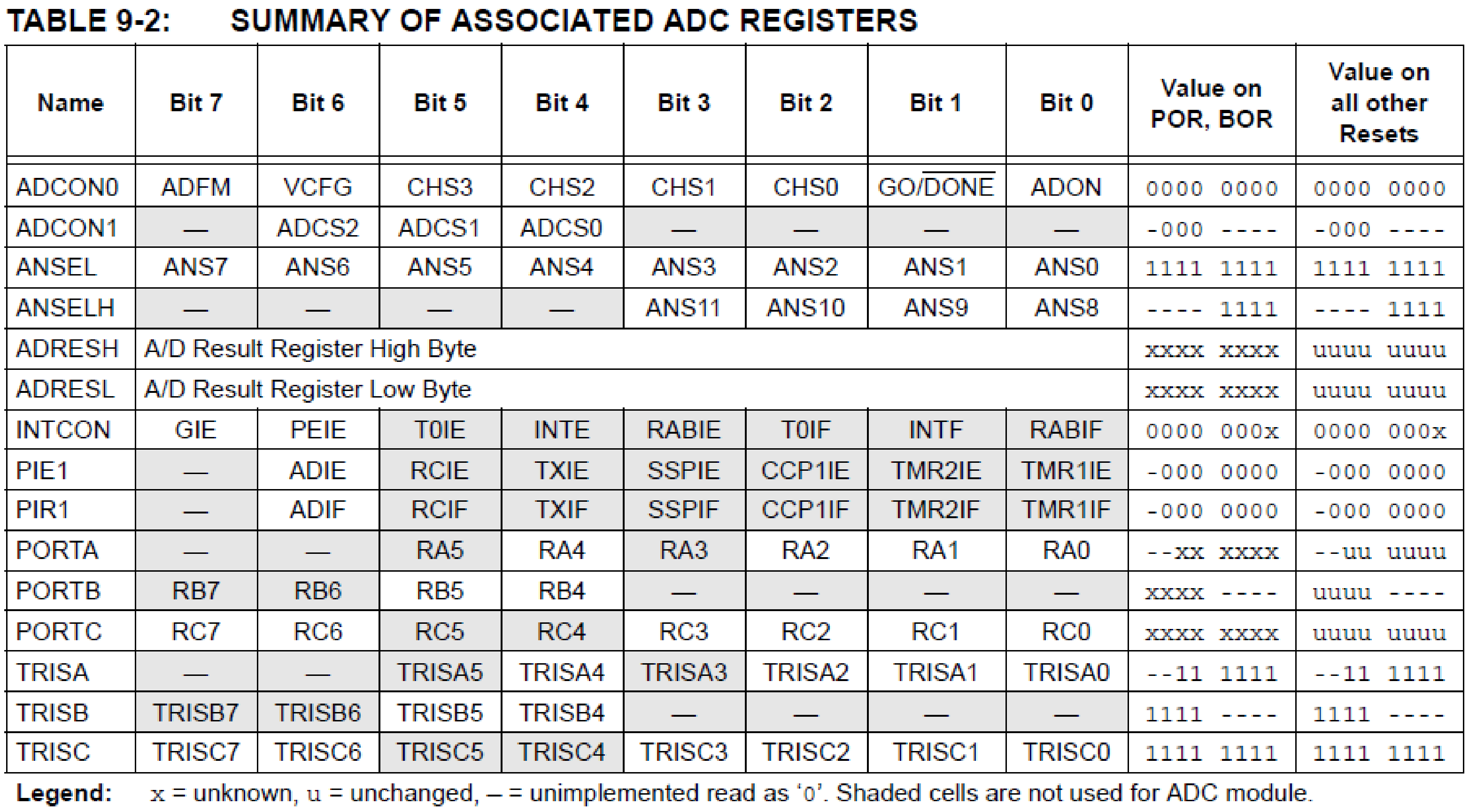


TABLE 9-2: SUMMARY OF ASSOCIATED ADC REGISTERS

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/$\overline{\text{DONE}}$	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								xxxx xxxx	uuuu uuuu
ADRESL	A/D Result Register Low Byte								xxxx xxxx	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 xxxx	--uu uuuu
PORTB	RB7	RB6	RB5	RB4	—	—	—	—	xxxx ----	uuuu ----
PORTC	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0	xxxx xxxx	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-conversion – step by step

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)