

Digital pulse sensors

Incremental encoders, angle

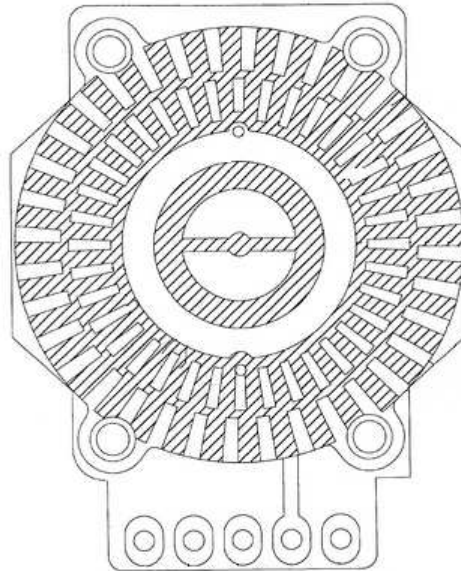


Luxury variant



Lowest Price

- **Mechanical sensing**



Contact pattern

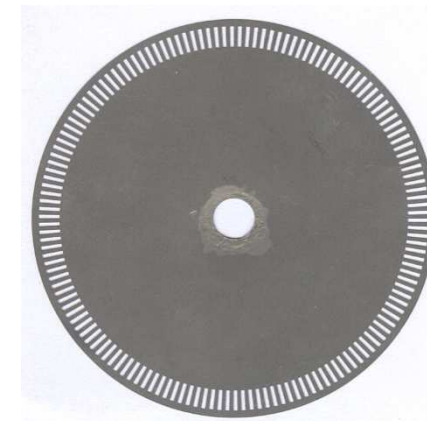
- **Optical sensing**



2 Optical fork coupler



2 spring contacts

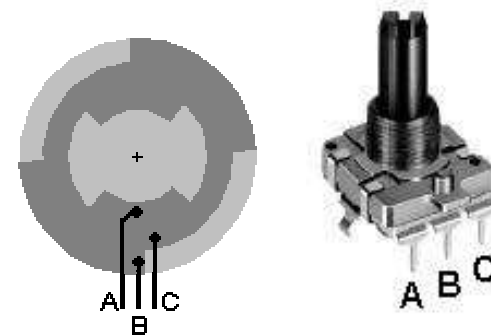


Hole/slit disc

Rotary Encoder

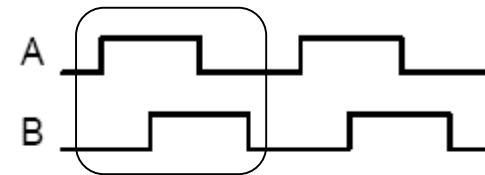
Rotary Encoders are often used as a digital angle sensor in the industry, but is now also used as the setting dials and knobs in consumer electronics (Jog up/down).

The latter types have mechanical contacts and mass-produced at low prices (there are encoders from about 20:-), so there is every reason to become familiar with this type of sensor.

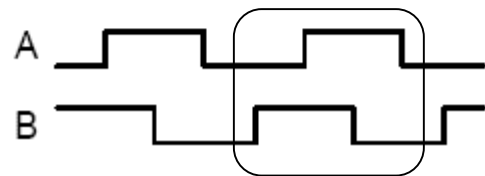


one "snap"

Rotation clockwise

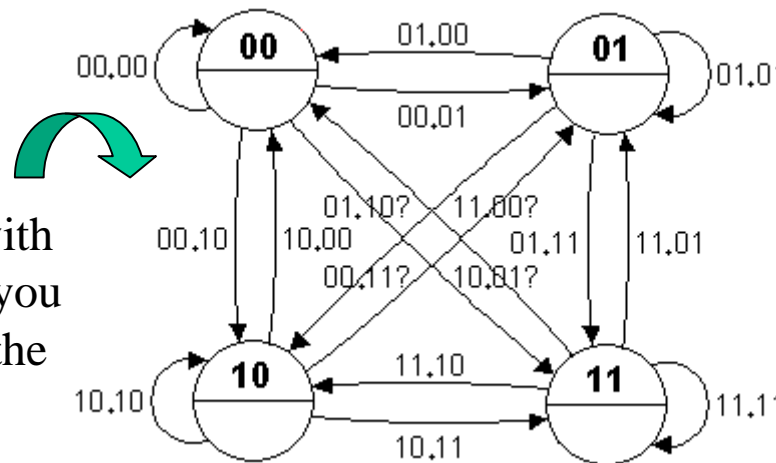


Rotation counter clockwise



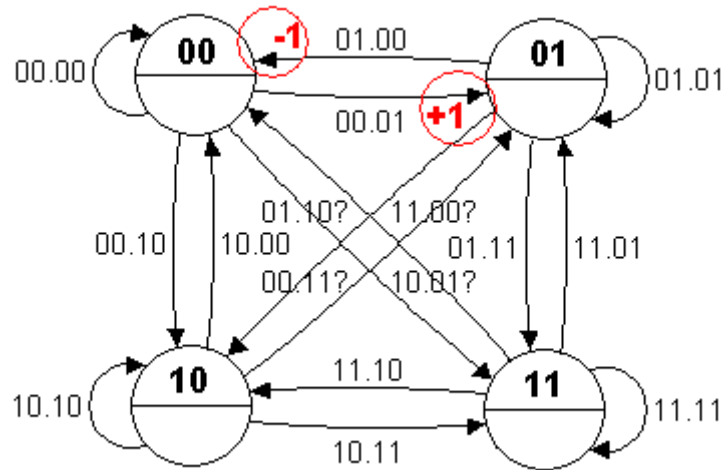
Gray-code:

... 00 01 11 10 ...



For each "snap" with the encoder shaft you move one turn in the state chart.

State chart

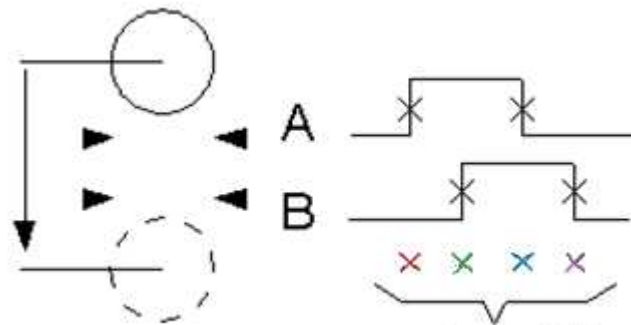
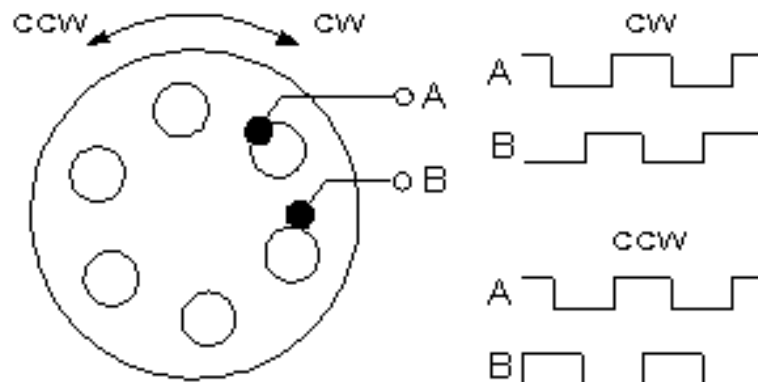


The sensor's four contact conditions can be plotted in a state diagram. Between the four states, there are a total of 16 different transitions (arrows in the diagram). The four diagonal transitions are actually "impossible" and can only occur by interference, or if you missed a reading.

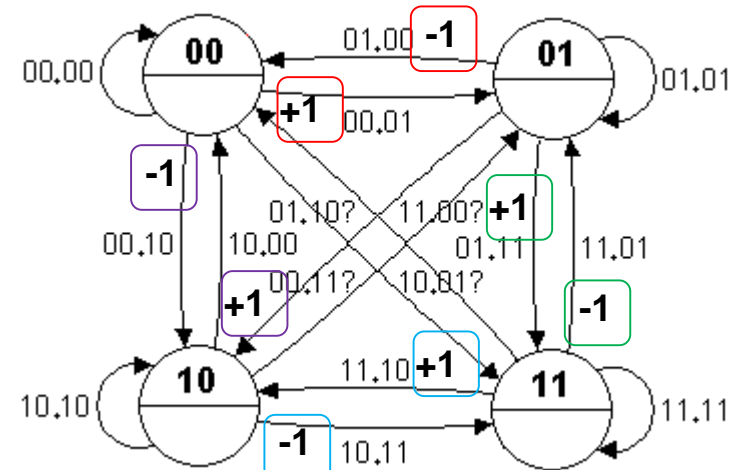
- One can count up the number of "snap" (+1) every time you moved from 00→01 in the state diagram, and down (-1) at 01→00.

(Digital interpolation)

- Four-fold higher resolution is possible.

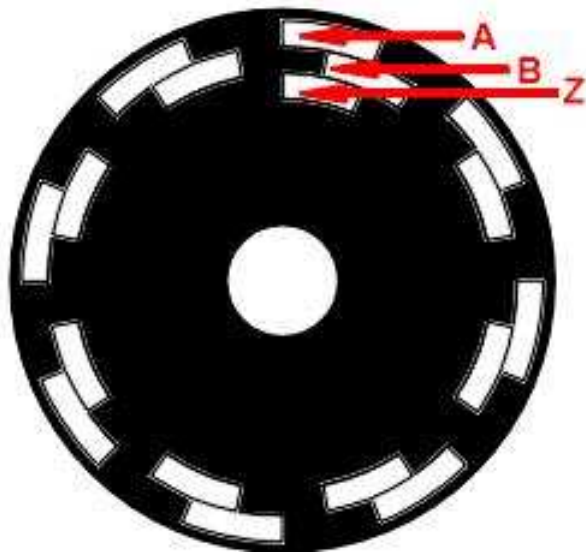


Four pulses for each hole



(Reference pulse)

Incremental encoder are based on counting and following all the changes. One then needs to know where you start from the beginning?



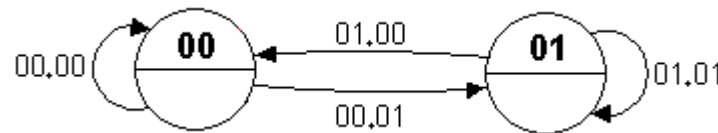
A third sensor Z produces a reference pulse once every turn.

William Sandqvist william@kth.se

(Binary constants)

The compiler Cc5x allows binary constants (not available in ANSI C). You may also arrange dots to indicate which bits belong together and form groups. Those dots have no meaning except to clarify the code.

ex. old→new
0b00.00 ⇒ 0
0b00.01 ⇒ 1
0b01.01 ⇒ 5
0b01.00 ⇒ 4

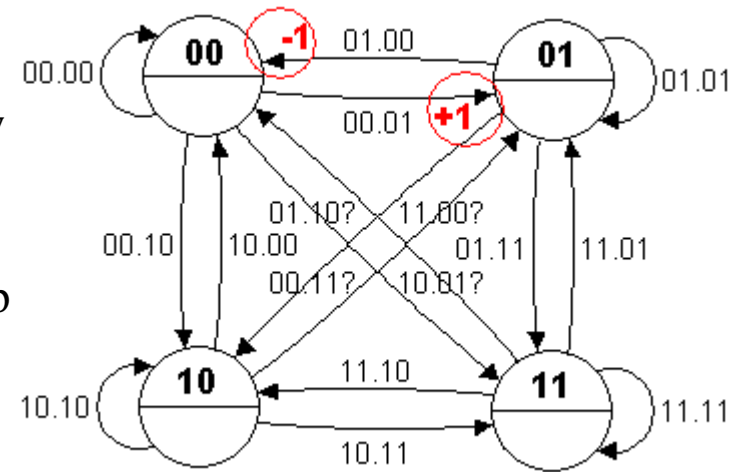


This is a way to you use the binary constants to denote the state transitions.

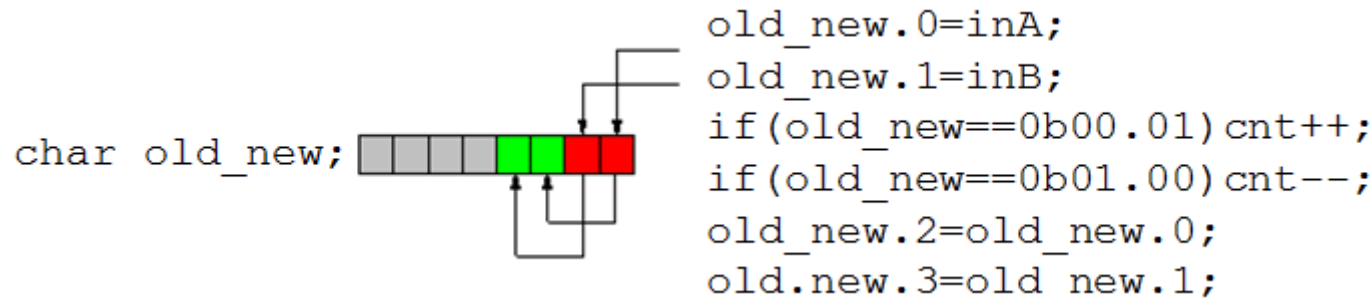
Count pulses

One stores the previous state in order to compare it with the current state. Each arrow in the state diagram consists of such a state pair **old.new**.

An easy way to read the sensor is to count up the position at the arrow **00.01** and down the position at the arrow **01.00**.



Even if the contact bounces "the net result" becomes correct, because you always have to go one way more than the other to change state.



```
old_new.0=inA;  
old_new.1=inB;  
if(old_new==0b00.01) cnt++;  
if(old_new==0b01.00) cnt--;  
old_new.2=old_new.0;  
old_new.3=old_new.1;
```



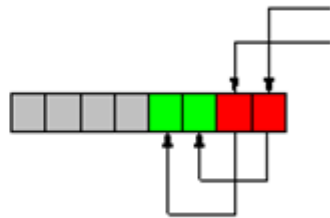
```

char old_new;
old_new.0=inA;
old_new.1=inB;
if(old_new==0b00.01) cnt++;
if(old_new==0b01.00) cnt--;
old_new.2=old_new.0;
old_new.3=old_new.1;

while(1)
{
  /* read encoder new value */
  old_new.0 = A;
  old_new.1 = B;
  /* compare with old value */
  if( old_new == 0b00.01 ) cnt--;
  if( old_new == 0b01.00 ) cnt++;
  /* replace old values with new values */
  old_new.2 = old_new.0;
  old_new.3 = old_new.1;

  /* this part takes long time!
  if(cnt != oldcnt) /* print value if changed ? */
    printf("Position: %d\r\n", cnt);
  oldcnt = cnt; /* update oldcnt
  */
}

```



Fast transition
6 μs



Slow
*/
Pulses during printf() are missed! */

William Sandqvist william@kth.se

Interrupt?

While the processor is printing position with `printf()` it can not **at the same time** read the encoder – then it can miss pulses!

”**Interrupt on change**”. PIC PORTs have the possibility to generate interrupt at changes. If, instead, it is the interrupt routine which read the encoder no pulses will be missed.

- `printf()` must now not use "bitbanging" - the interrupts would destroy the serial communication timing.
- `printf()` must use the standalone EUSART device that will not be disturbed by the interrupts.

Polling and Interrupt



Suppose you are sitting in a comfortable chair and reading a book. Suddenly You are interrupted by the phone ringing, You mark with a pencil where in the book you were and then you answer.



During the call the doorbell will ring and you tell the person you are speaking to in the phone to wait while you go to the door.

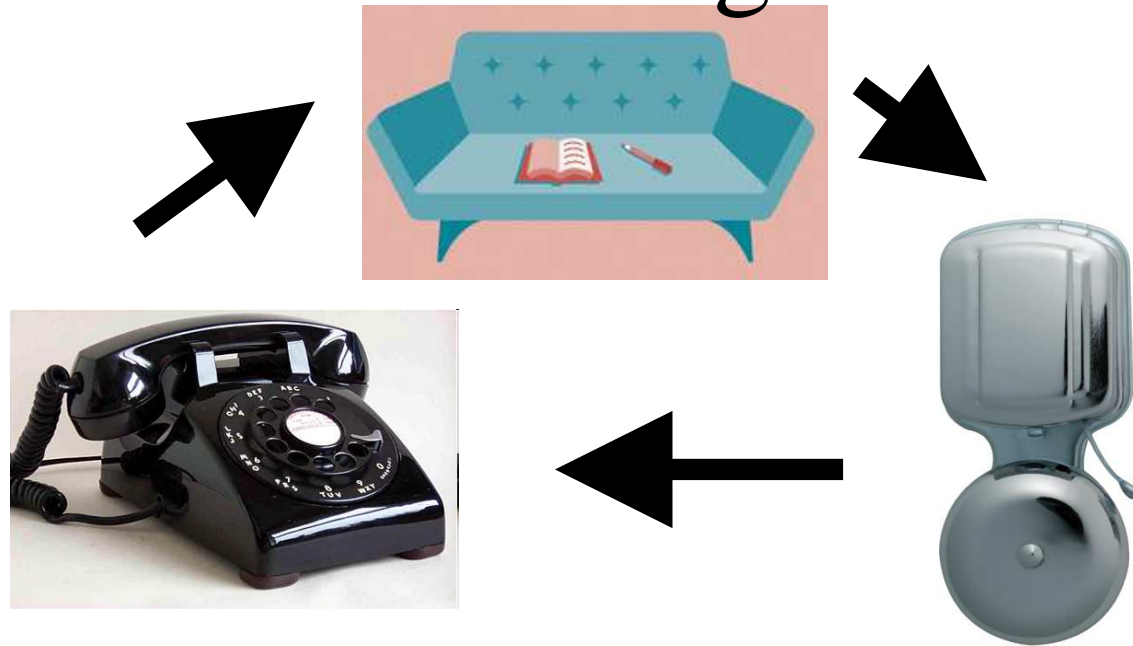


Interrupt

When you are finished with the matter at the door resume the call. When after a while have finished talking on the phone and finished the phone call you can return to the chair and continue to read the good book – at the pencil mark.



Polling



If there would be no interrupt mechanism one would be forced to rush around between the door– anyone there? – phone – anyone on the line? And the sofa.

This is called **polling**.

William Sandqvist william@kth.se

Interrupt mechanisms

Global and Local Enable



Don't you want to be disturbed, you can put on your earplugs - You have then made it impossible to interrupt, *disable interrupt*.

Remove the earplugs and you have re-enabled interrupt, *enable interrupt*. This is called **Global Enable**.

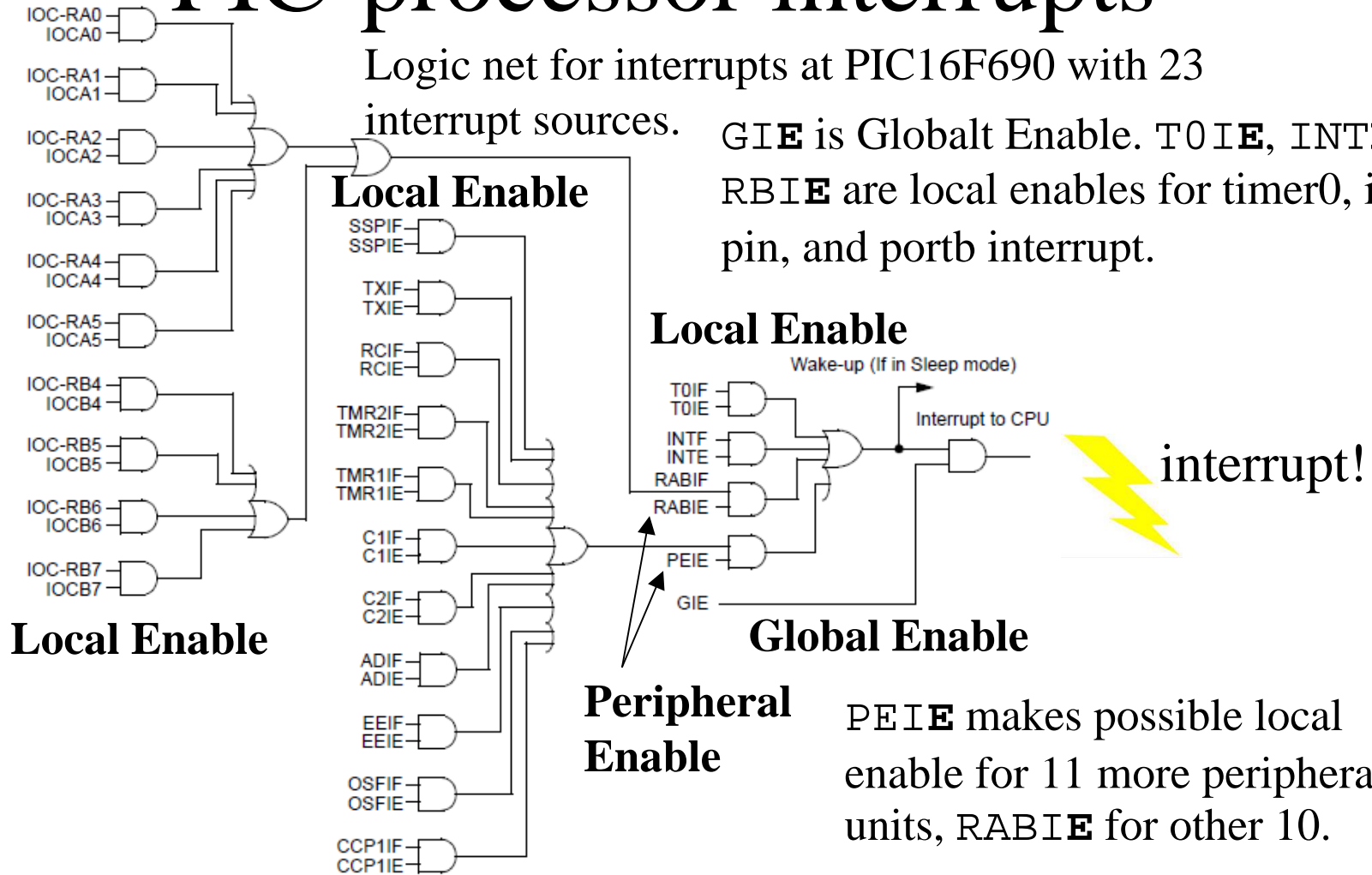
You also have the opportunity at the local level to enable/disable interrupt, **Local Enable**. You can for example disable phone by unplugging the jack. Then you still can hear the doorbell.



PIC-processor interrupts

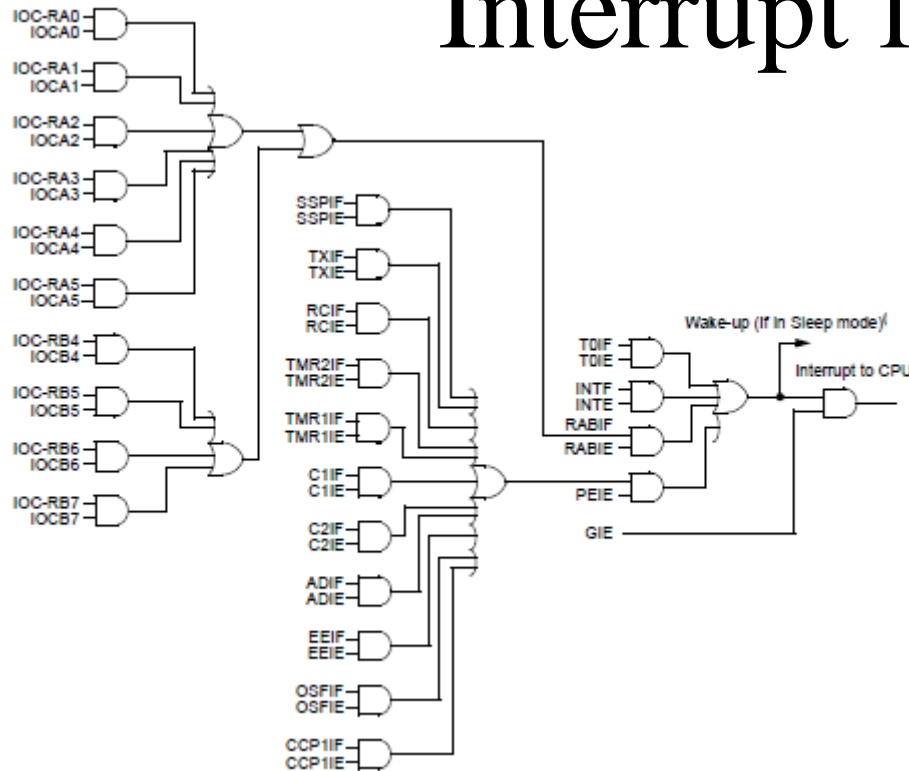
Logic net for interrupts at PIC16F690 with 23 interrupt sources.

GIE is Globalt Enable. **T0IE**, **INTE**, **RBIE** are local enables for timer0, int-pin, and portb interrupt.



PEIE makes possible local enable for 11 more peripheral units, **RBIE** for other 10.

Interrupt flags



TMR1IF TMR2IF
CCP1IF CMIF TXIF
RCIF EEIF T0IF
INTF RBIF are the
names on some of the
flags that indicates
different interrupt-
causes.

If there is a cause, *and* the source is local enable (and if it is a peripheral – it also is peripheral enable) *and* global enable is true – **then there will be an Interrupt!**

Interrupt flags

TABLE 14-6: SUMMARY OF INTERRUPT 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
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
PIE2	OSFIE	C2IE	C1IE	EEIE	—	—	—	—	0000 ----	0000 ----
PIR1	—	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	-000 0000	-000 0000
PIR2	OSFIF	C2IF	C1IF	EEIF	—	—	—	—	0000 ----	0000 ----

Interrupt routine

At interrupt the Interrupt routine is run. It's on a fixed place in the beginning of the program memory. Must be first.

`#pragma origin 4` Interrupt routine allways starts at address 4!

```
interrupt int_server( void )
```

```
{
```

```
int_save_registers
```

```
/* interrupt routine */
```

```
int_restore_registers
```

```
}
```

Macron to save the contents of registers. Otherwise the interrupt routine returns garbled results to the main program!

(PIC Manual. Part 14.4 Context saving during interrupt.)

Context saving

Interrupt! Context is important, page, rambank ...

• **Kungsgatan 4 – Stockholm**

• **Kungsgatan 4 – Avesta**



Cc5x saves the most important content – and warns if **more** could be needed to be saved.

(PIC Manual. Part 14.4 Context saving during interrupt.)

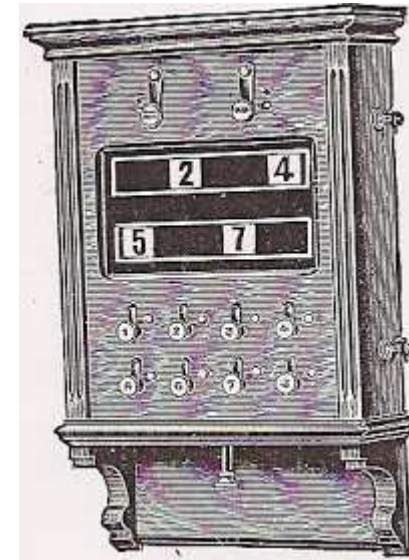
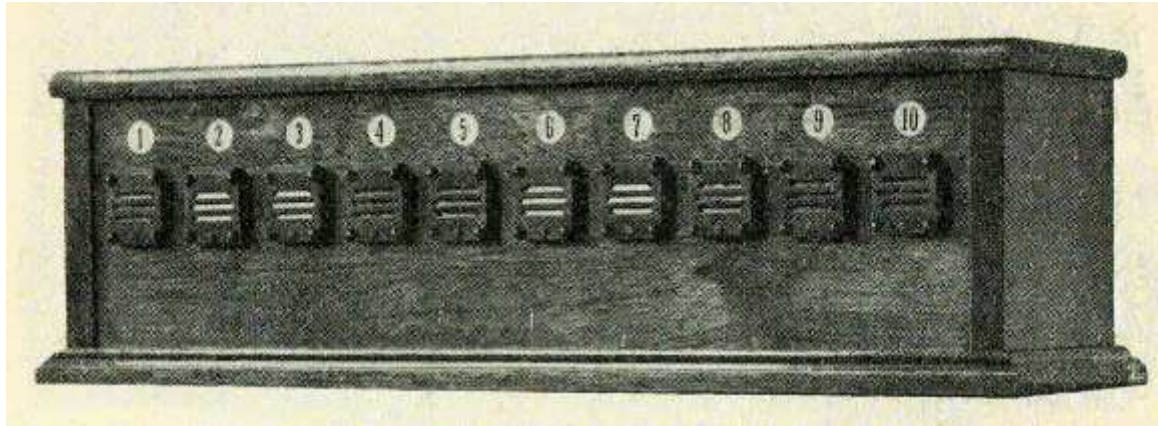
Reset interrupt flag

Interrupt flags indicate what caused the interrupt.

In the interrupt routine, check the flags and do what needs to be done.

The interruptflag that is 1 must be **reset** at the end of the interrupt routine - otherwise the interrupt continues forever!

Servants bell display



A "servants bell display", a elektromechanical signalling device which occurred in the early 1900's in luxury apartments. From push buttons in the different rooms one could call on the serving staff or the maid. The bell rang and the corresponding display indicated. When the mission was performed the servants pressed the button under the display to reset the indicator. - Is it perhaps from here Microchip got the idea for their interrupt mechanism?

William Sandqvist william@kth.se

RPG Interrupt program

```
char old_new; /* global to store transitions */
int cnt;      /* global to store RPG count   */
```

- *Interrupt routine must be first*

```
#pragma origin 4 /* only place for interrupt routine */
interrupt int_server( void )
```

```
{
```

```
    int_save_registers
```

```
    old_new.0 = PORTA.5;
```

```
    old_new.1 = PORTA.4;
```

```
    if( old_new == 0b00.01 ) cnt ++;
```

```
    if( old_new == 0b01.00 ) cnt --;
```

```
    old_new.2 = old_new.0;
```

```
    old_new.3 = old_new.1;
```

```
    RABIF = 0; /* Reset flag before leaving */
```

```
    int_restore_registers
```

```
}
```

*runs every time something
changes on porta*




to main program

main() -program

- *main() and other functions follow next*

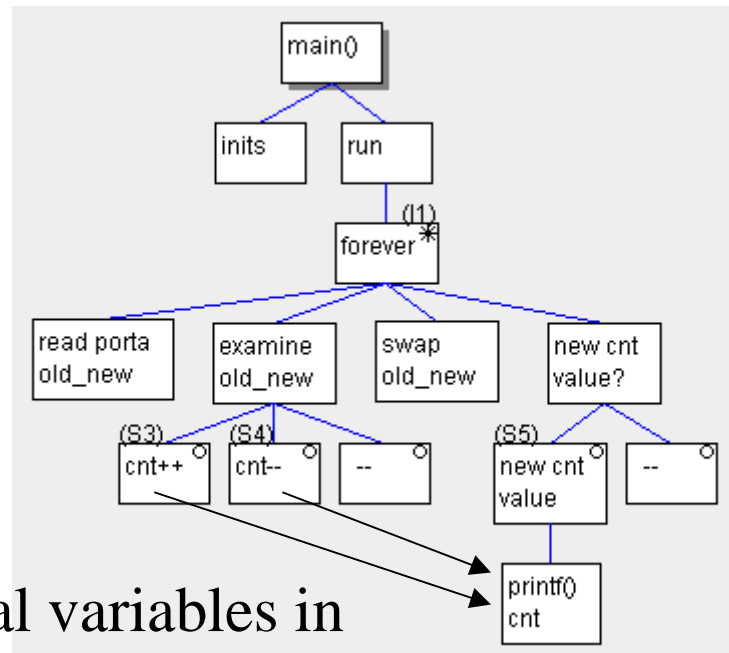
```
void main( void)
{
    init();          /* init ports          */      Interrupt on
    RABIE    = 1;    /* local enable    */      change porta
    GIE      = 1;    /* global enable   */
    initserial();   /* init serial unit */
    new_old = 0; cnt = 0;

    while(1)
    {
        printf("Position: %d\r\n", cnt );
        delay10(100); /* print RPG-count every second */
    }
}
```

from ISR


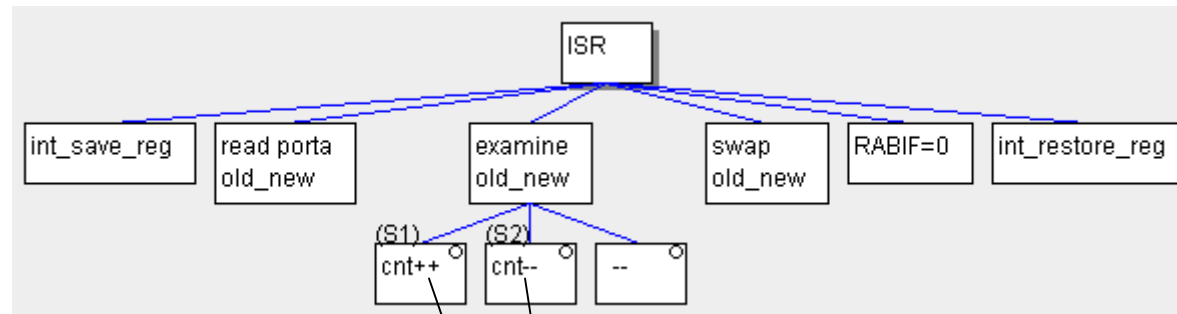
No pulse is missed in cnt, and the value is printed without trash every second!

RPG *without* interrupt

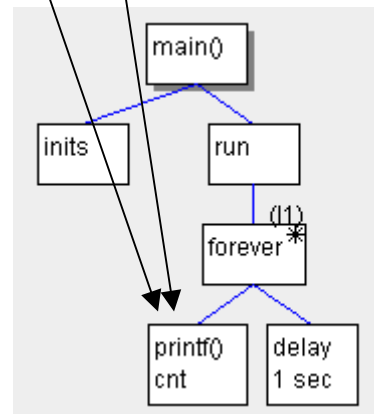


Local variables in
main().

RPG *with* interrupt

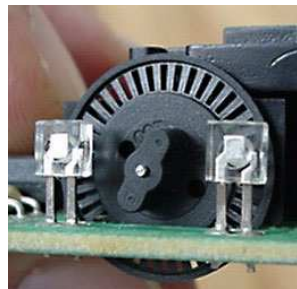
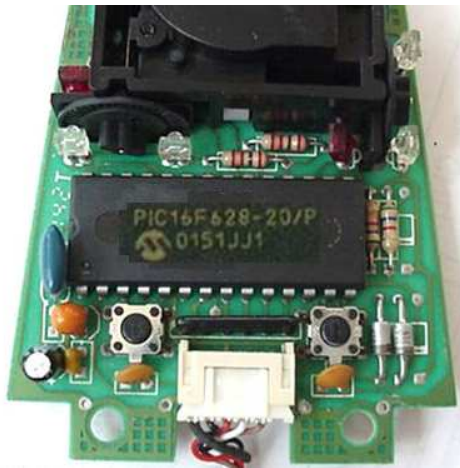


Global variables
outside main().

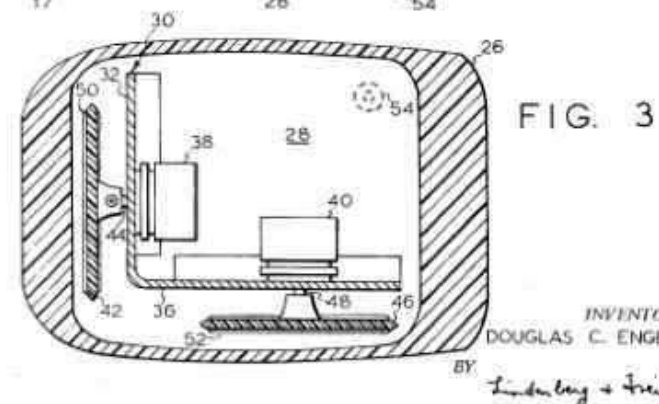
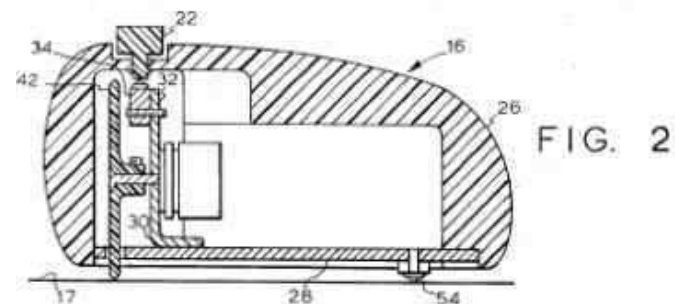
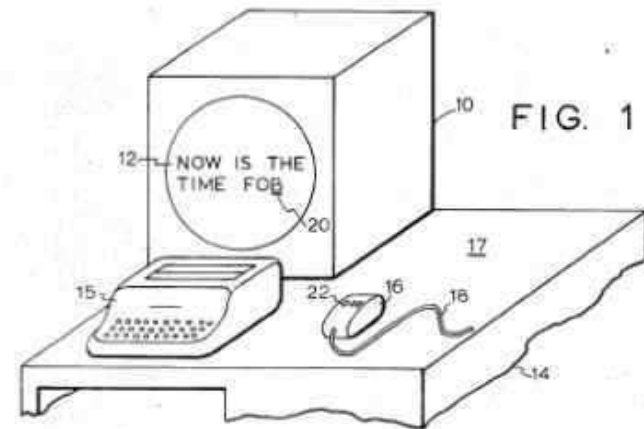


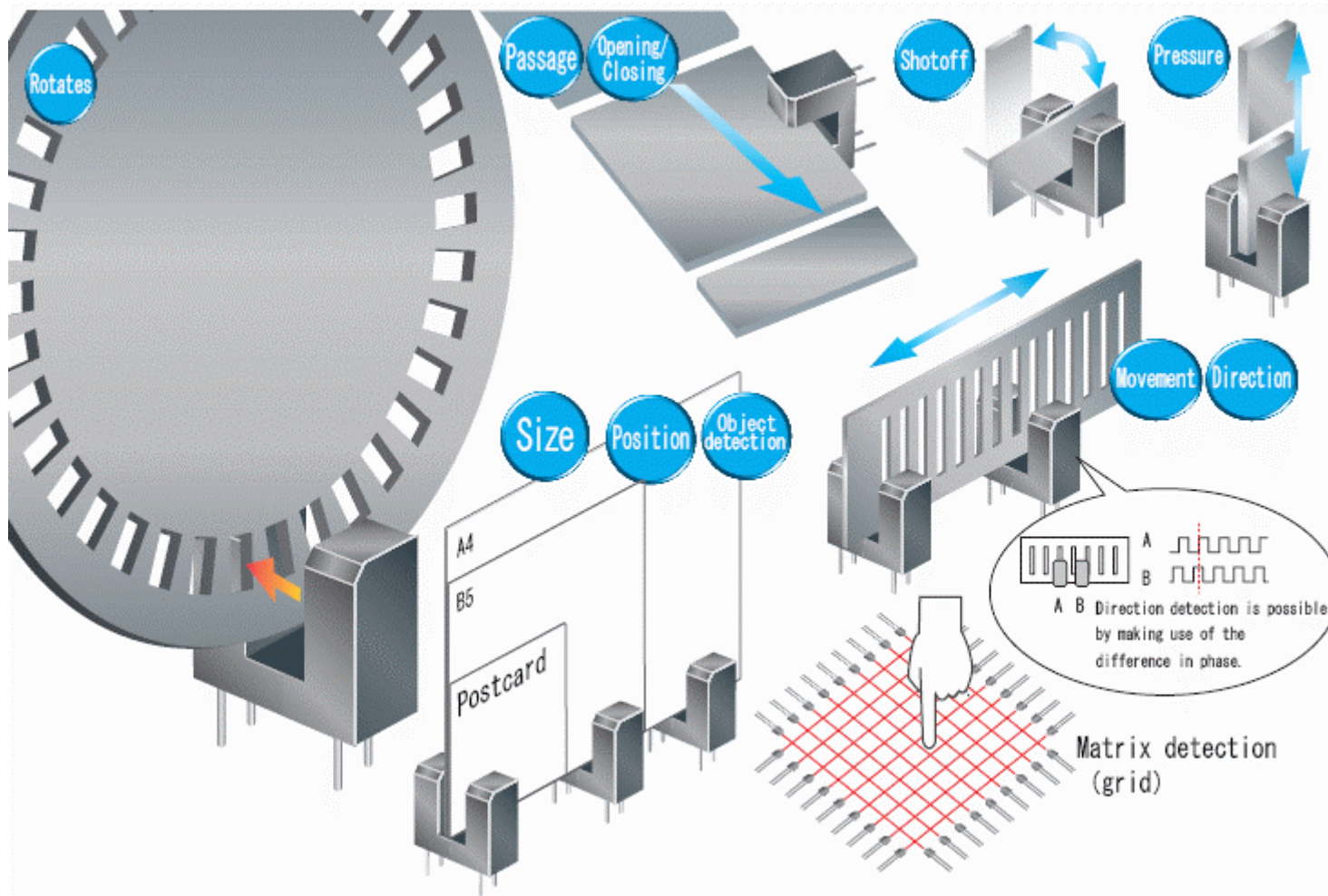
Computer Mouse.

A computer mouse contains two encoders, though nowadays optical. PIC16F690 has "Interrupt on change" for the four PORTB pins and 6 PORTA pins, which is sufficient to five encoders! - So it could very well be a PIC processor chip inside the mouse!



Nov. 17, 1970 D. C. ENGELBART 3,541,541
X-Y POSITION INDICATOR FOR A DISPLAY SYSTEM
Filed June 21, 1967 3 Sheets-Sheet 1





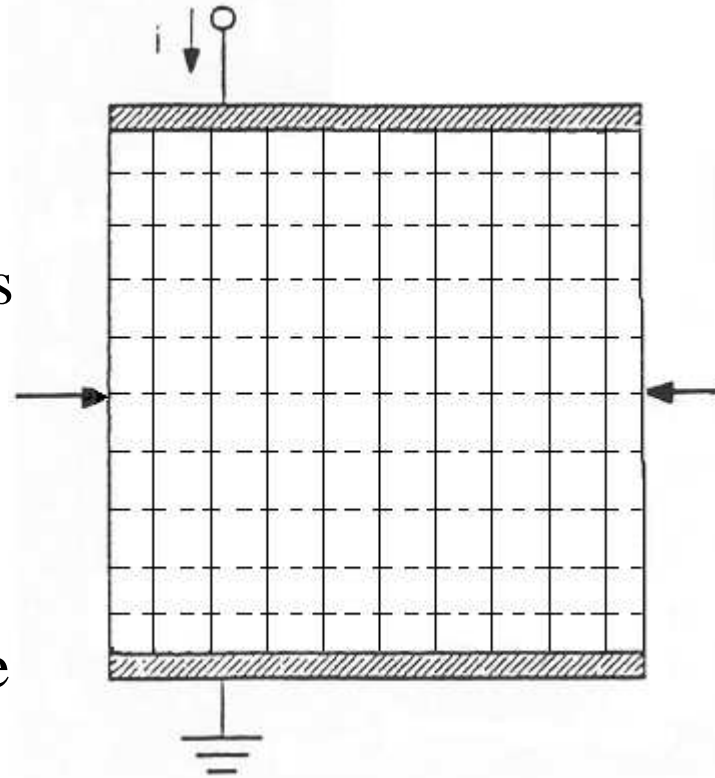
Other uses ...

William Sandqvist william@kth.se

William Sandqvist william@kth.se

Magnetic sensor

A plate traversed by a current between two of the sides. Current paths are parallel and charge distribution in the disc is homogeneous. The two electrodes (at the arrows) will be located along the same voltage line, and there will be no resultant potential difference between them.

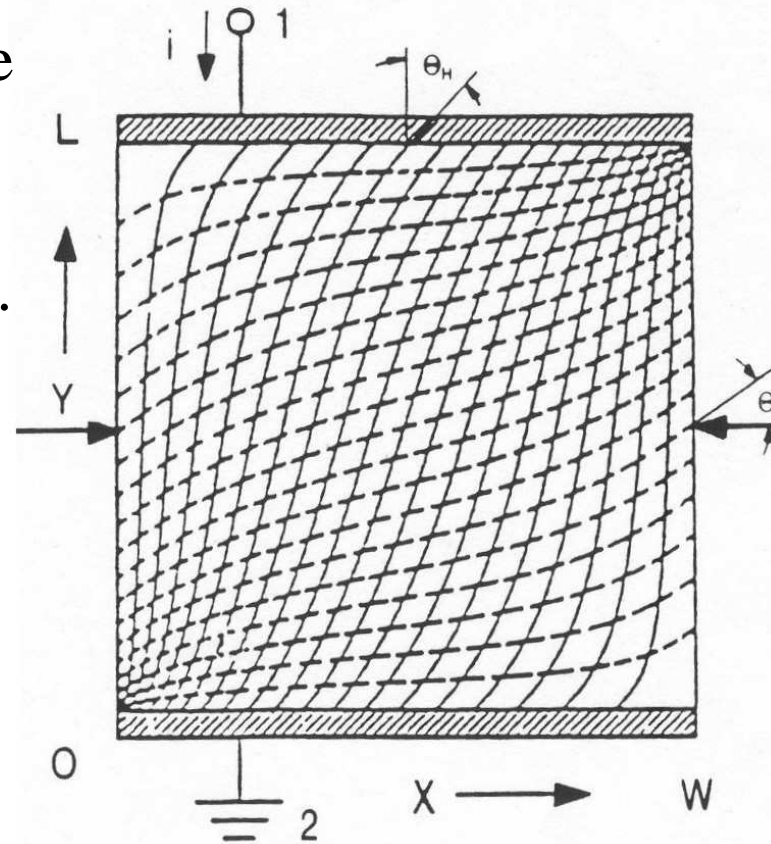


Magnetic sensor

Now a magnetic field forces the charge "out of off course". The current paths bend, and the charge distribution gets uneven.

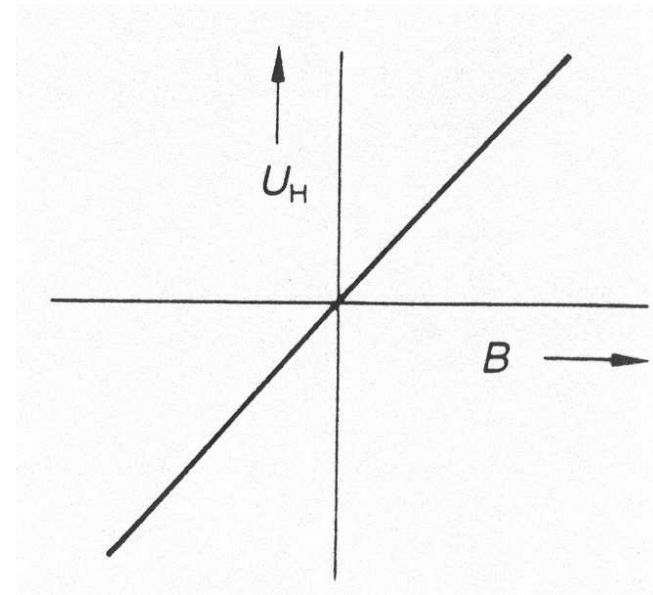
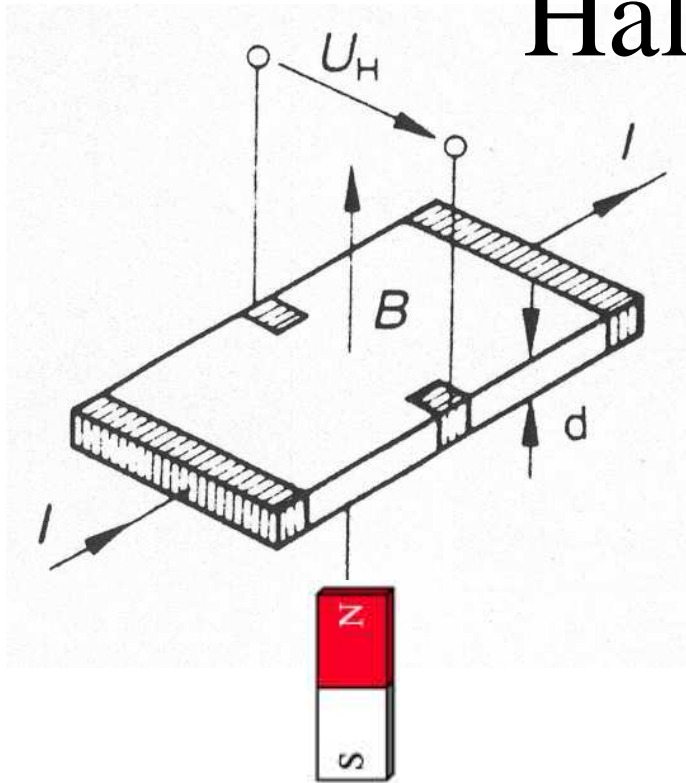


The two electrodes (at the arrows) is now at different voltage lines, and then a net voltage difference is produced.



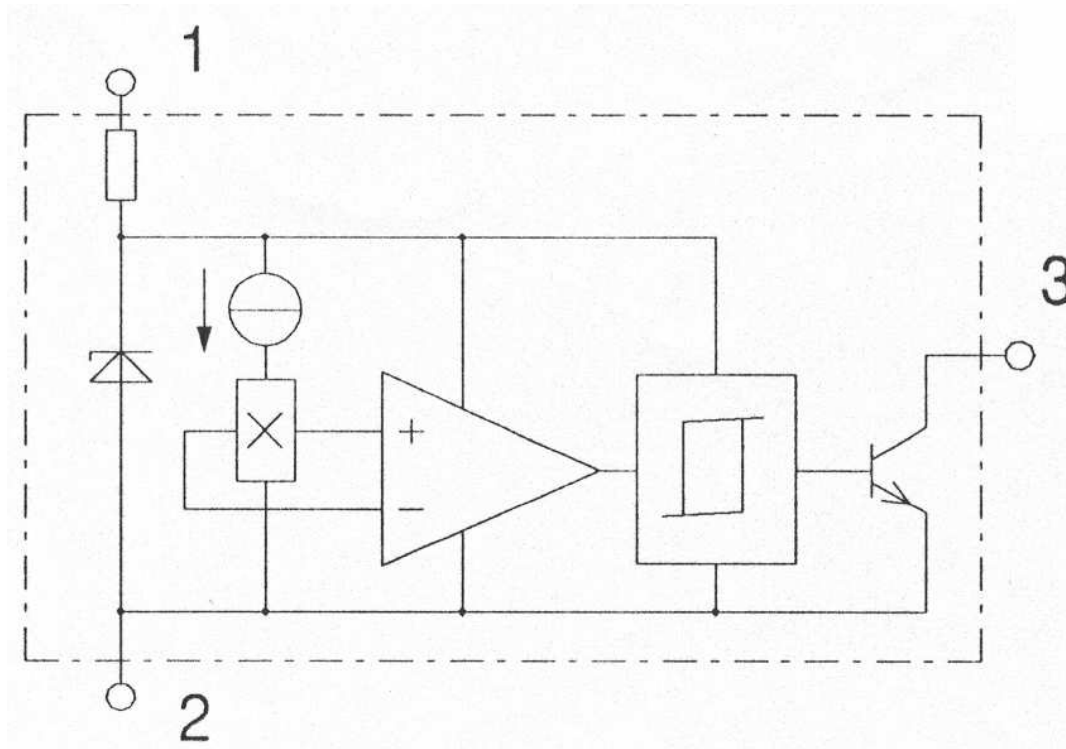
The stronger the magnetic field, the higher the voltage at the arrows!

Hall effect



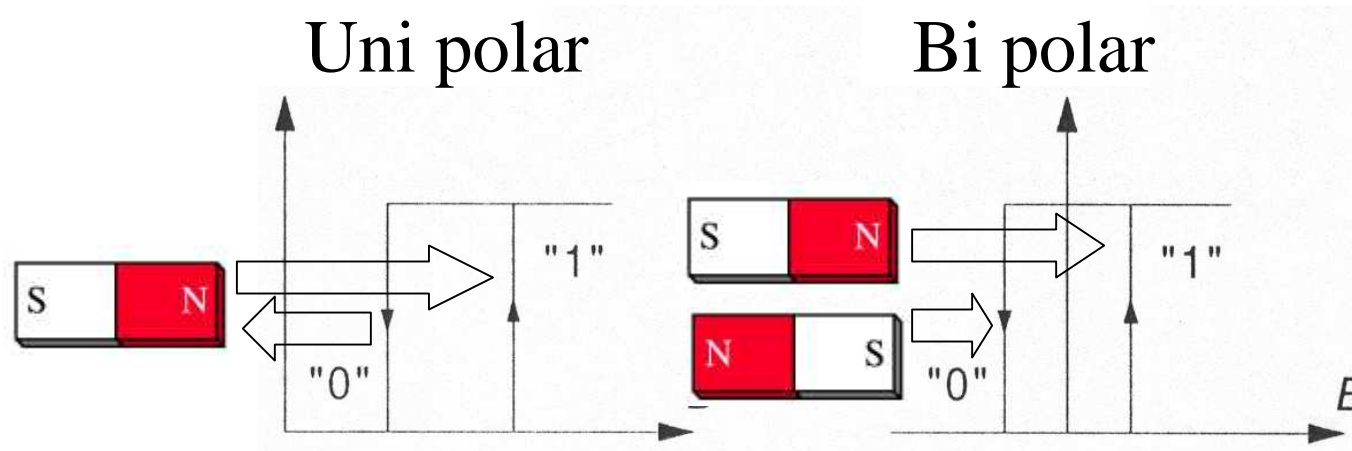
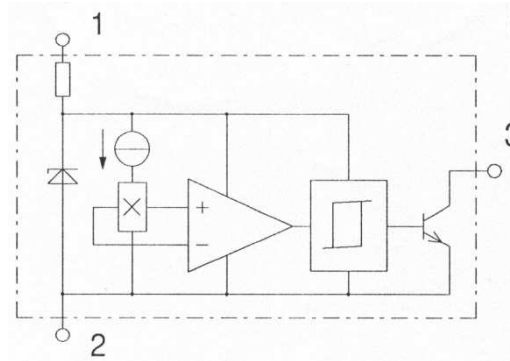
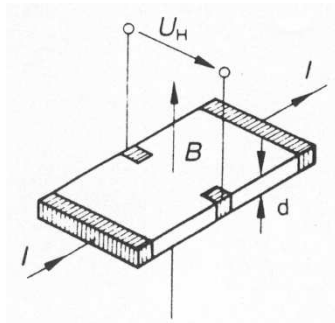
A weak Hall voltage U_H proportional to the magnetic field flow B , indicates the presence of the magnet.

Hall switch

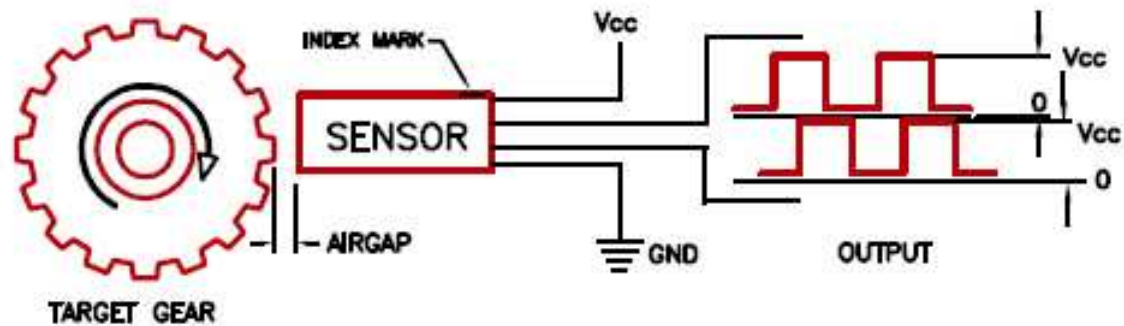
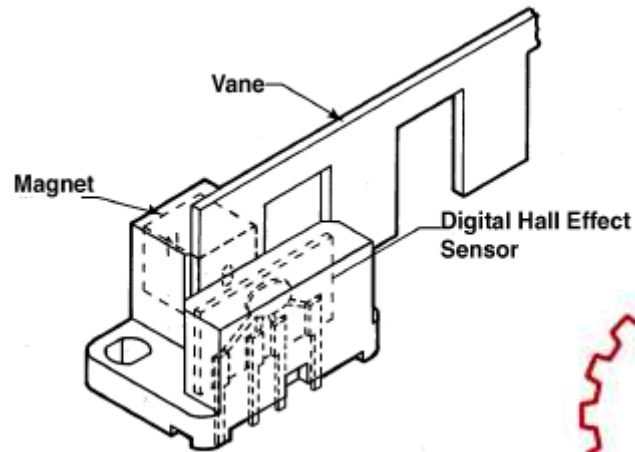
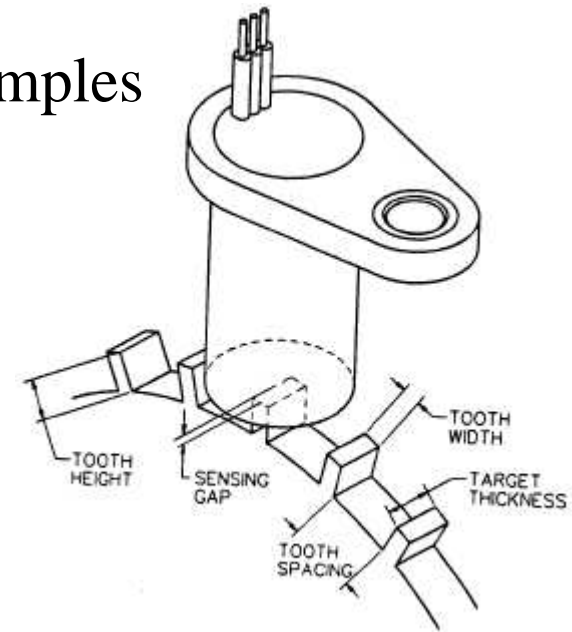


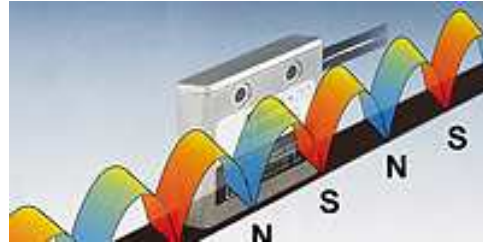
Current regulator, Hall element, amplifier, Schmitt-trigger, driver.

Hall sensors Unipolar/Bipolar



Some different application examples





Hall Switches over a magnetic tape - you'll see something like this in the lab. . .

William Sandqvist william@kth.se