INDUSTRIAL INFORMATICS

and EMBEDDED SYSTEMS

Francesco Leporati
Tel. 0382 985678
e-mail: francesco.leporati@unipv.it

COURSE SECTIONS:

- DIGITAL INTERFACING
- INDUSTRIAL COMMUNICATIONS & FIELD BUSES
- EMBEDDED ARM MICROPROCESSORS

THE EXAM

<u>Typically</u>: 1) a code fragment in ARM assembly language to be interpreted

2) an exercise on the evaluation of the frequency of the pulses provided by a suitable sensor, considering a microprocessor whose bits are employed for different purposes (driving a motor, acquisition of a physical a magnitude, generating a signal, PWM, PFM, …)

<u>As the case is</u>: 1) questions on the different parts of the course to establish if mechanisms have been really captured

2) exercises on the communication protocol part

MASTER THESIS (finalised to job positions)

Companies that propose thesis:

MARELLI MOTORSPORTS (Corbetta Milano): video data logging, telemetry

TEMIS (Corbetta Milano): automotive, satellites, data acquisition and communication

BDSOUND (Assago): sound engineering, audio signals acquisition through microcontrollers (Cortex M4, ST, …)

AZCOM (Rozzano): wireless (and not) safe communications through sw control (DSP and FPGA processors)

FACILITY LIFE: cybersercurity

ST MICROELECTRONICS: embedded processors

MASTER THESIS (in laboratory)

DSP, GPU, FPGA technologies applied to:

Brain cerebellar simulation

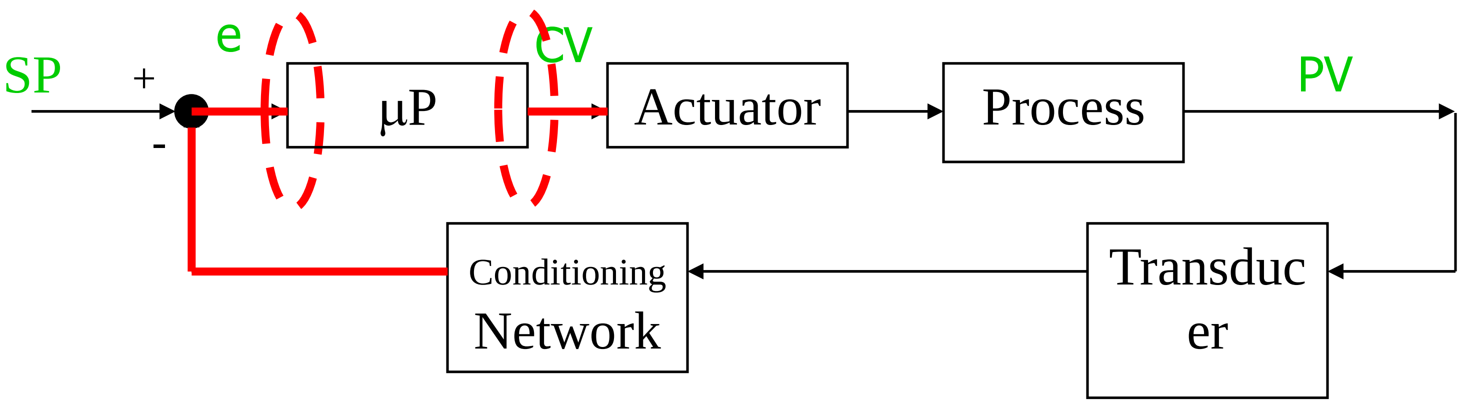
Hyperspectral image processing for high accuracy cancer detection

Glaucoma identification in ophtalmic imaging

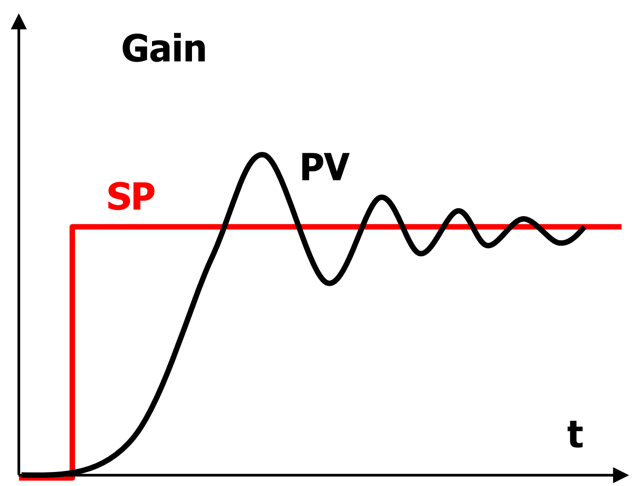
Automotive applications

Signal processing through embedded microprocessors

FOCUS OF THE COURSE



- **Control algorithm** = the controller operating principle
- **Process** = the problem. Know the transfer function
- **Minimize** the gap between SP and PV (and dependency on environmental variables)
- **Oscillations** due to the velocity of the answer to SP variations ⇒ minimize the **transient**
- **Overshootings** ⇒ limit the amplitude
- **Stability** (no infinite or diverging oscillations)
- **Delays**



COURSE SECTIONS

- **DIGITAL INTERFACING**

- **INDUSTRIAL COMMUNICATIONS & FIELD BUSES**

- **EMBEDDED ARM MICROPROCESSORS**

DIGITAL INTERFACING

Interfacing = interactions between active operative devices (i. e. CPU) and/or passive ones (i. e. memories, sensors, ports)

The course is mainly (but not only) focused on the interactions between **calculator** and **external world** (every thing not directly manageable by the CPU through buses)

Several kinds of **in-homogeneities**:

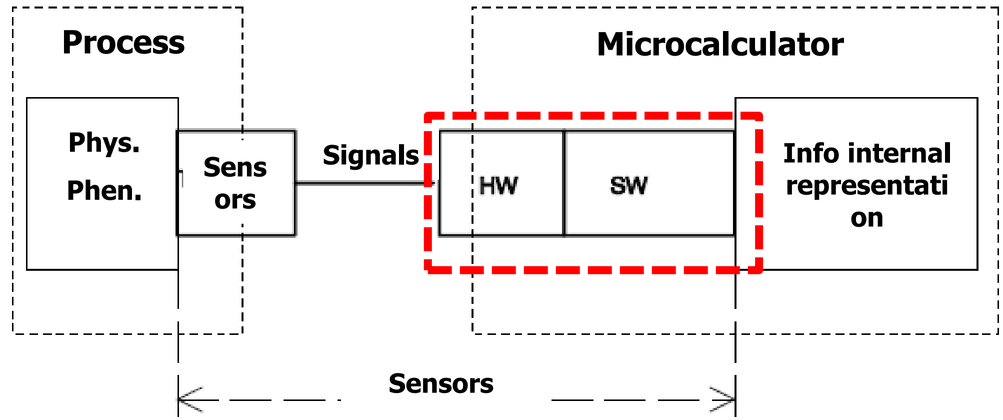
- *Physical*: the external world is characterized by physical parameters that can be represented only through electric entities with suitable amplitudes (transducers)
- *Formal:* codification (BCD/Gray), representation (**pulse count or frequency measurement**), signal processing/management (**PWM or PFM modulation**)
- *Temporal*: how external phenomena are synchronized with signal acquisition/emission? (**timers**, **interrupt**/polling/DMA, **real time** protocols)
- *Spatial*: computers and controlled devices are not physically adjacent (**transmission**)
- *Attitudinal*: Human Computer Interaction

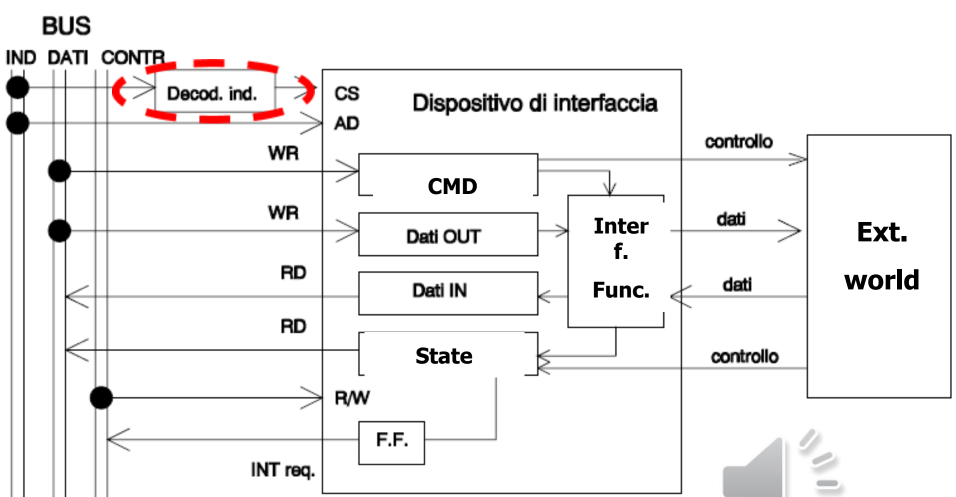
A suitable **chain** of interacting devices allows to overtake these in-homogeneities

DIGITAL INTERFACING

Typical computer-external world interfacing

Interface design = identification of a chain of components able to wholly satisfy the requirements of data transformation and communication

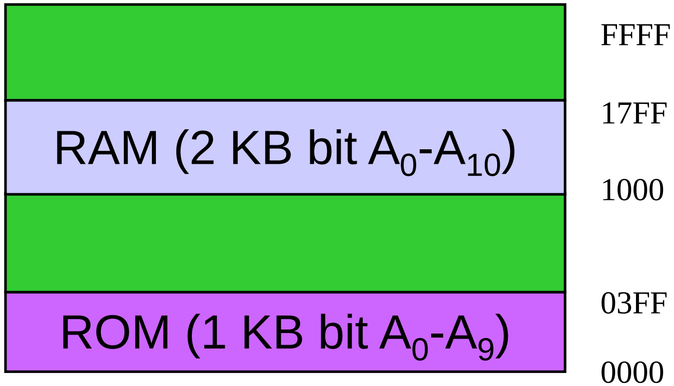




A general interface digital circuit

- I/O buffer (*Fifo*)
- *State registers* for significant situations (errors, overrun, interrupt …)
- *Command registers* to setup mode functions and modify state bits
- *Various functions*: DMA, ser/par transfers

ADDRESSING ON THE BASIS OF A MEMORY MAP



μP with 16 bit address bus and 8 data bus

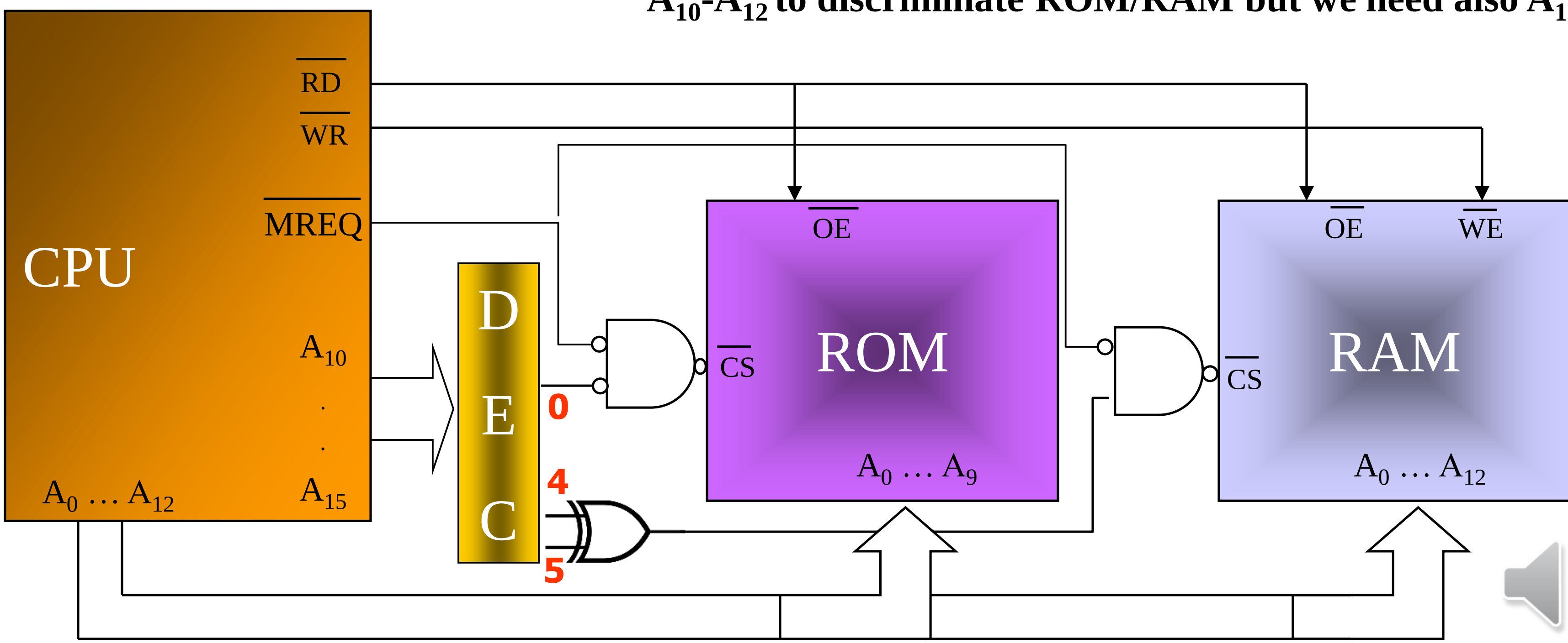
End_RAM 0001 0111 1111 1111

Beg_RAM 0001 0000 0000 0000

End_ROM 0000 0011 1111 1111

Beg_ROM 0000 0000 0000 0000

A_{10}-A_{12} to discriminate ROM/RAM but we need also A_{13}-A_{15}



ADDRESSING ON THE BASIS OF A MEMORY MAP

1) A single address space (MEMORY MAPPED I/O)

lw $reg1, off($reg) $reg= port (mem) address

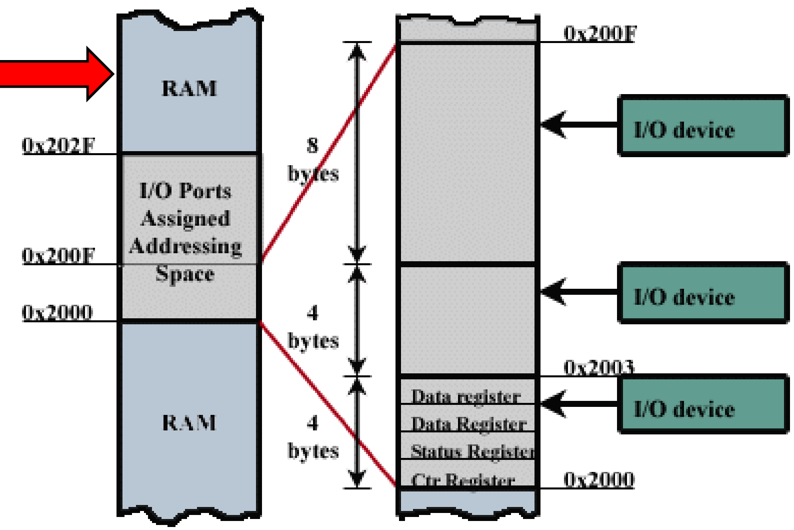
ex. ARM, MIPS

2) Double address space (CUSTOM INSTRUCTIONS)

lw $reg1, off($reg) $reg=memory address

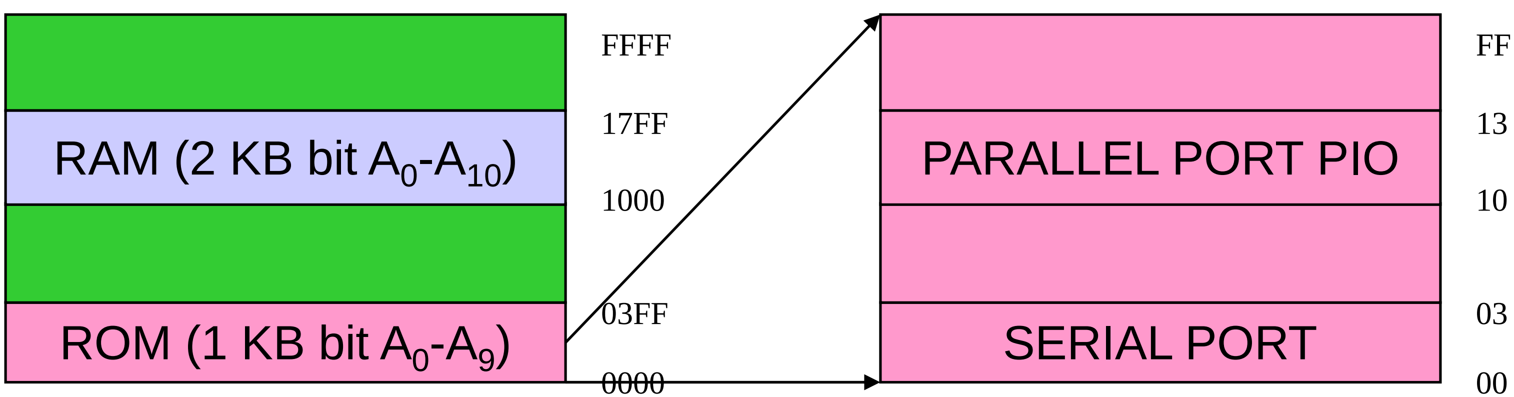
in $dest, ($reg) $reg=port address

ex. INTEL

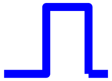


Control signals to distinguish between memory and I/O access (MREQ, IORQ)

addressable space devoted to data ***addressable (duplicated) space devoted to peripherals***



DIGITAL INTERFACING

On off signals 

- They represent physical entities acquired through single bits of a microprocessor port
- What is important is the value of the bit, not its variation
- 8-16-32 bit groups acquisition (bytes, words)
- Let's **test** the Kth bit

Esempio in C
```
var_bool = 0;
if (immagine & mask_K)
    var_bool = 1;
```
Variante più compatta:
```
var_bool = (immagine & mask_K) && TRUE;
```

- And now two assembly routines (the right one is optimized maybe by the compiler)

Esempio in Assembler 8086 (versione intuitiva)
```
        MOV     AL, [IMMAGINE]
        AND     AL, MASK_K
        MOV     [VAR_BOOL], AL
        JZ      SKIP
        MOV     [VAR_BOOL], 0FFH
SKIP:
        ..
```

Esempio in Assembler 8086 (versione più efficiente da meditare)
```
        MOV     AL, [IMMAGINE]
        AND     AL, MASK_K
        SUB     AL, MASK_K
        SBB     AL,AL                   ; SuBtract with Borrow
        CPL     AL
        MOV     [VAR_BOOL], AL
        .....  seguito ......
```

On off signals

•Let’s **set** the Kth bit

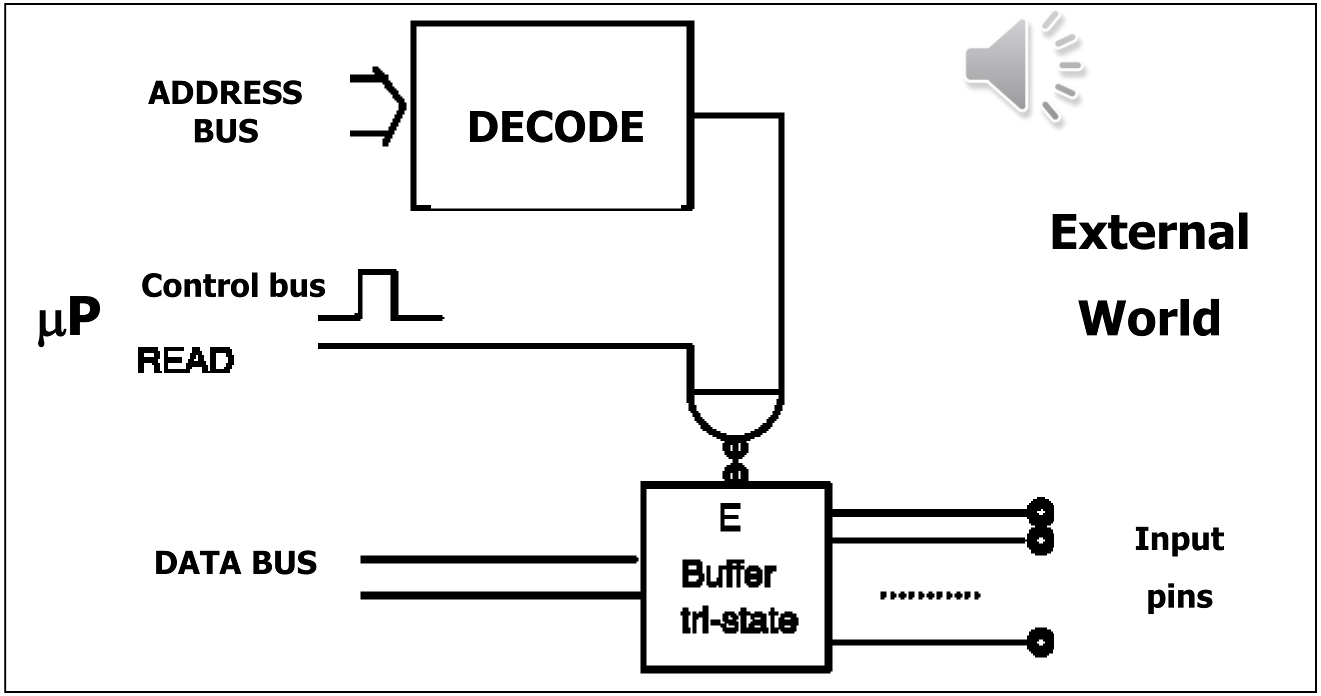
```
Esempio in C
if (var_bool)
   immagine |= mask_K;        // OR con mask
else
   immagine &= ~ mask_K; // AND con COMPLEMENTO di mask
```

```
Esempio in Assembler
     MOV  AL, MASK_K         ; maschera
     AND  AL, [VAR_BOOL]     ; isola bit in posizione K
     MOV  AH, AL             ; salva in AH
     MOV  AL, MASK_K
     CPL  AL                 ; maschera complementata
     AND  AL, [IMMAGINE]     ; azzera bit in posizione K
     OR   AL, AH             ; inserisce nuovo bit in posizione K
     MOV  [IMMAGINE], AL     ; aggiorna variabile immagine
```

DIGITAL INTERFACING

On off signals: "state" acquisition

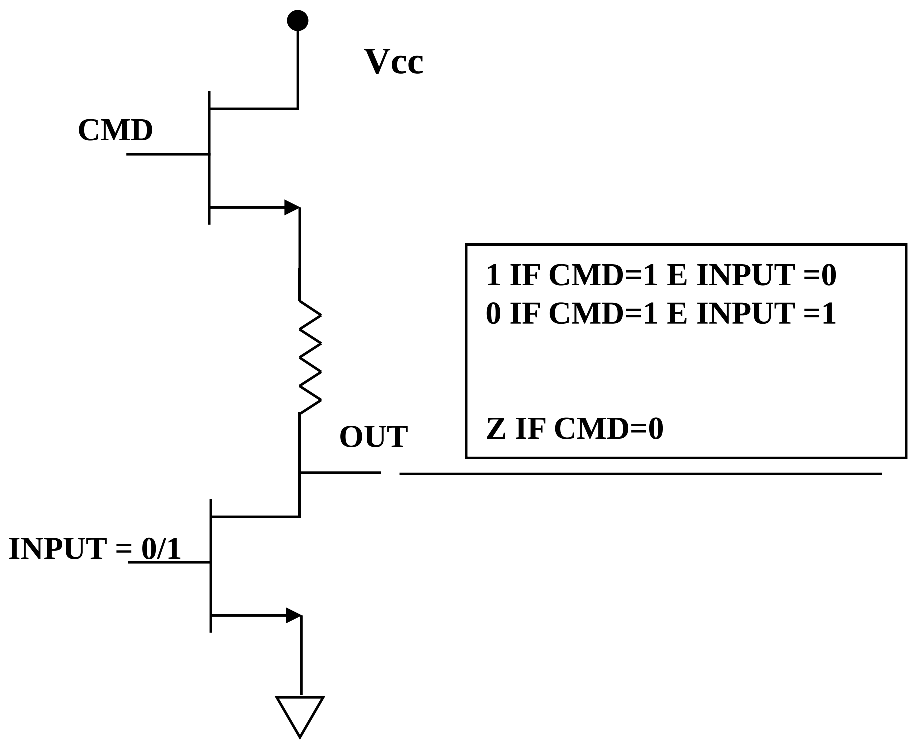
- The value of a bit is read to establish the level (*state*) of a line related to a physical entity: use of a *three state buffer*



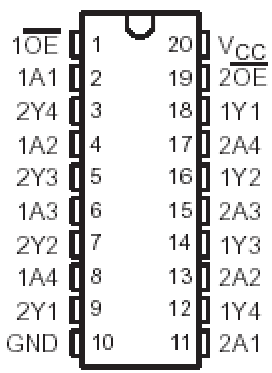
DIGITAL INTERFACING

On off signals

- three state buffers to de-couple the line with respect other devices



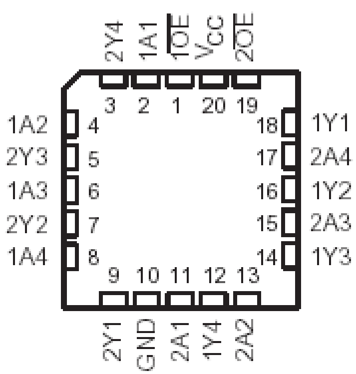
COTS buffer components (Texas Instr. SN 74AC244)



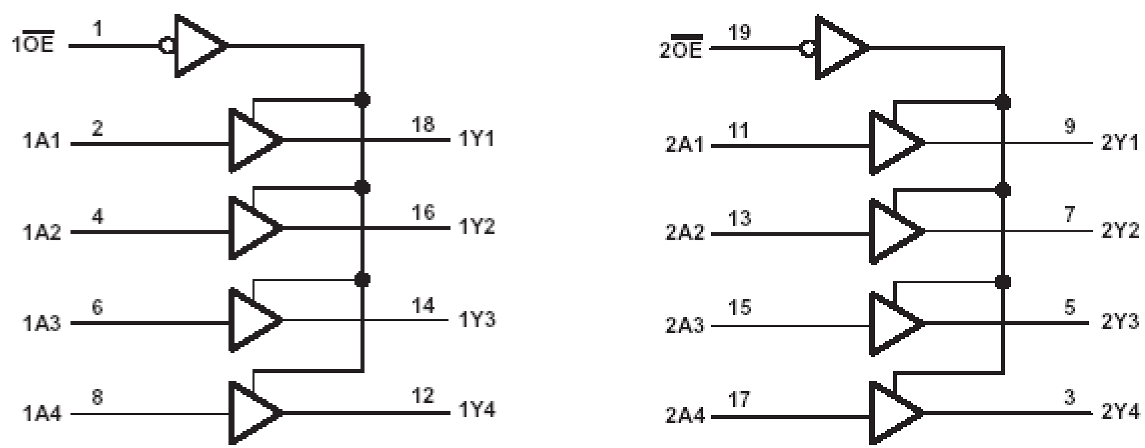
FUNCTION TABLE
(each buffer)

INPUTS		OUTPUT
$\overline{OE}$	A	Y
L	H	H
L	L	L
H	X	Z

SN54AC244 . . . FK PACKAGE
(TOP VIEW)



logic diagram (positive logic)



DIGITAL INTERFACING

On off signals: event triggering

•The value of a bit is read just to establish if a commutation has occurred (event) of a line related to a physical situation, by comparing it with the previous value

•2 possible approaches:

–**SW triggering through polling**

>A microprocessor port is continuously sampled. The software compares the new value with the old one and identifies the eventual commutation

-triggering through interrupt

>The signal carrying the "event" is connected to a microprocessor port's pin

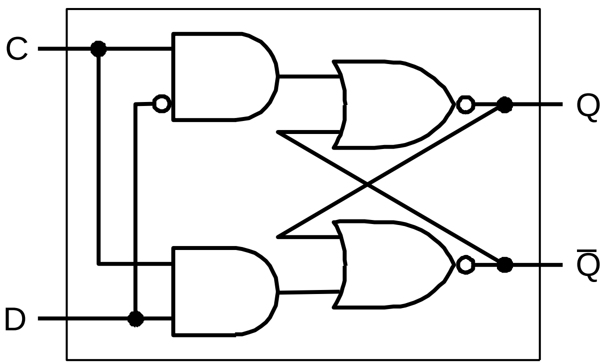
>The single commutation edge is determined through pins edge triggering (falling/raising edges)

>Both the edges are acquired through suitable electronic circuits sensible to edge or to levels. Where is the difference?

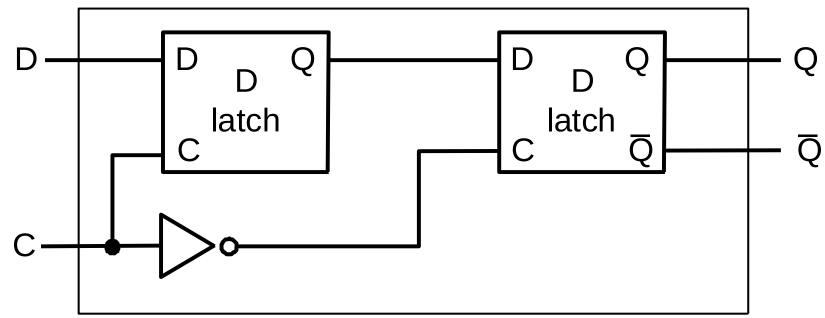
DIGITAL INTERFACING

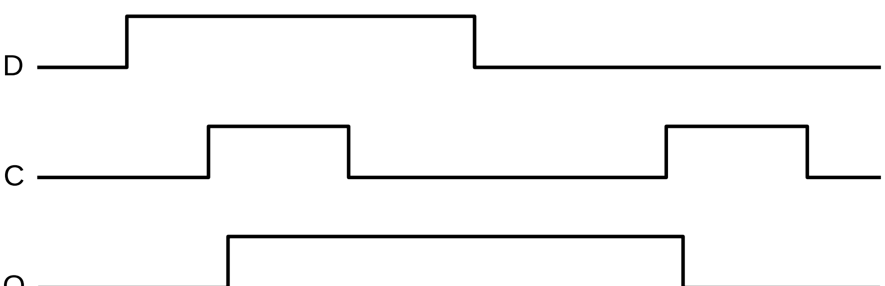
On off signals: "event" acquisition through interrupt

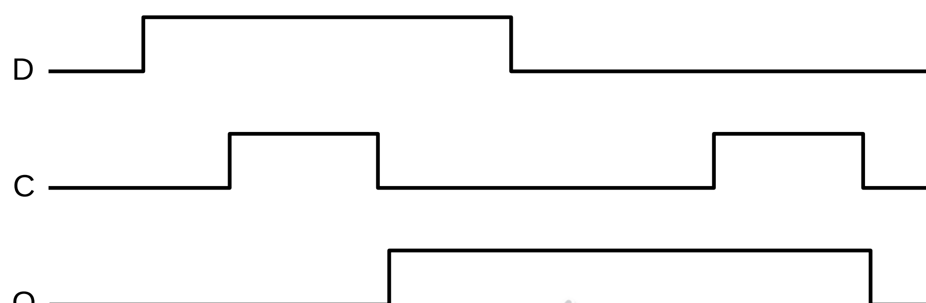
Latch (sensible to the level)



FF (sensible to the edge)



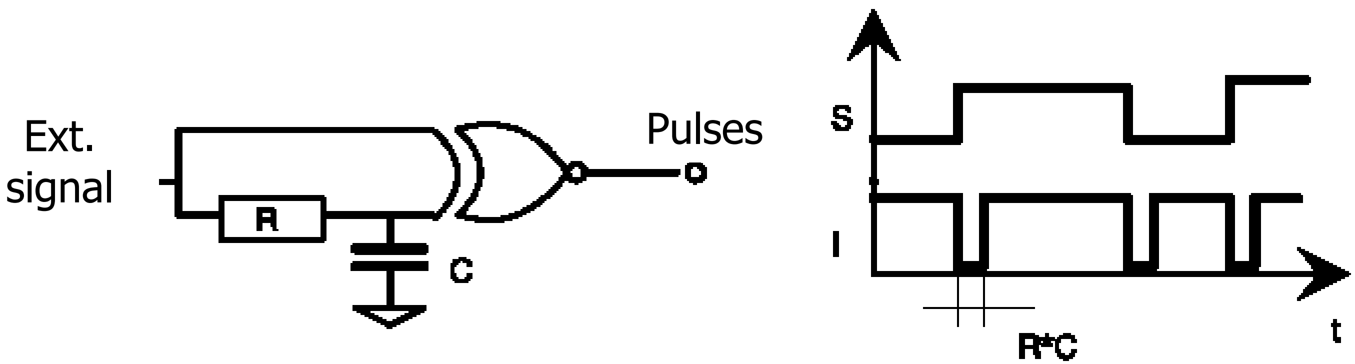




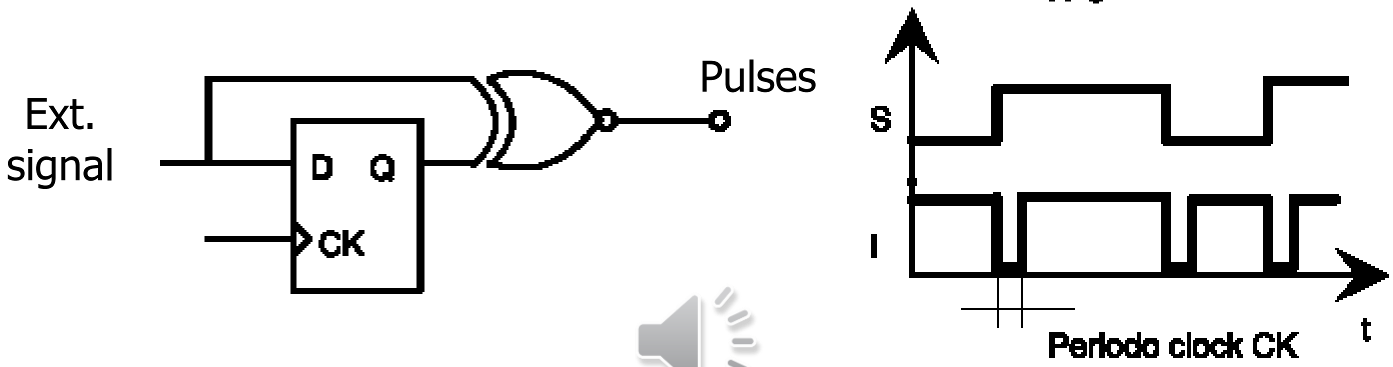
DIGITAL INTERFACING

On off signals

• a bit value is read to establish the possible commutation using *one-edge sensible circuits.* Below a circuit able to transform commutation of **both** the edges in a single direction



asynchronous



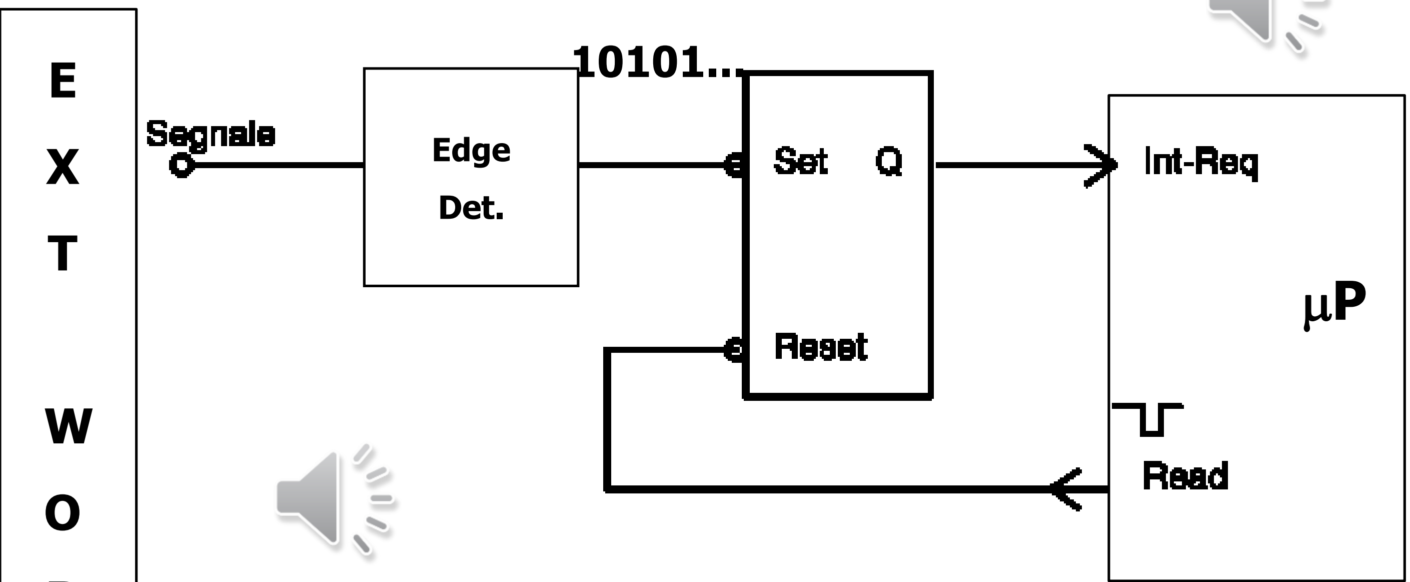
synchronous

$T_{ck} < T_s/2$ if duty cycle 50%

DIGITAL INTERFACING

On off signals

a bit value is read to establish the possible commutation using *level sensible circuits* (a FFJK, or a SR, is required to keep the input fixed)



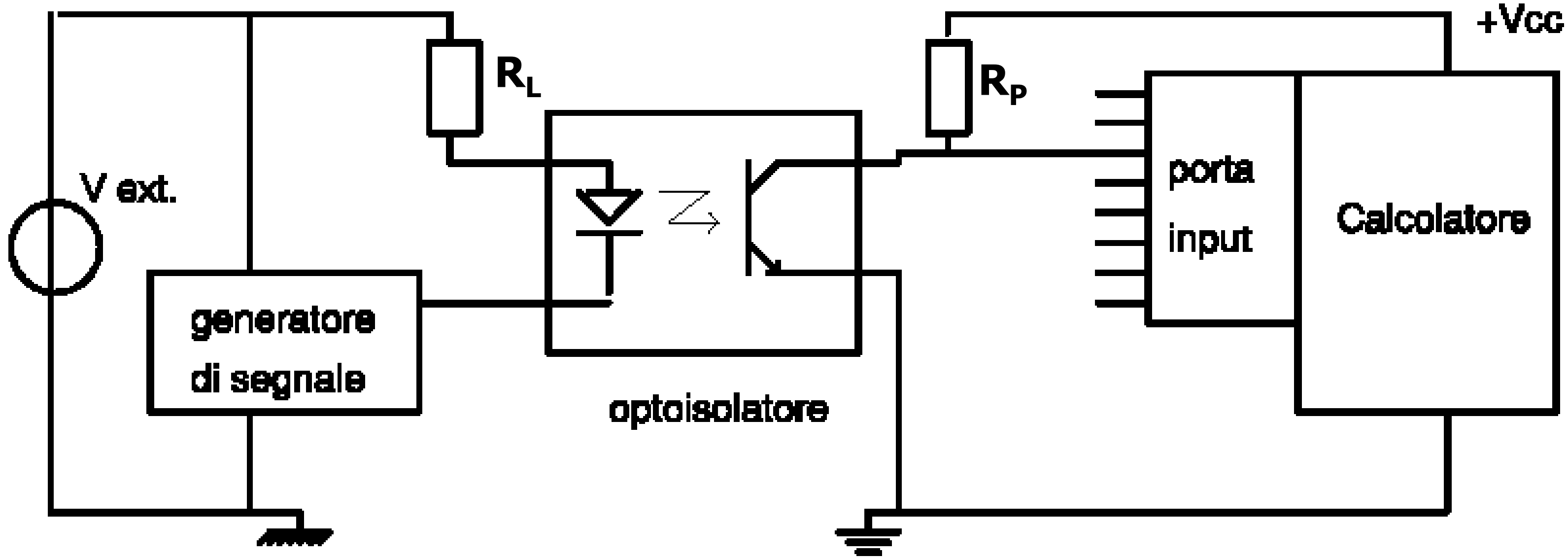
FFJK/SR	
00	U
10	1
01	0
11	$\bar{U}$/NP

The interrupt routine must be faster than pulse frequency

DIGITAL INTERFACING

Physical signal adjustment: opto-couplers

- Mandatory if grounds are different or if possible over-voltages
- R_L to determine necessary current for LED power up
- R_P pull-up (saturation vs "steep" edges: what possible values?)

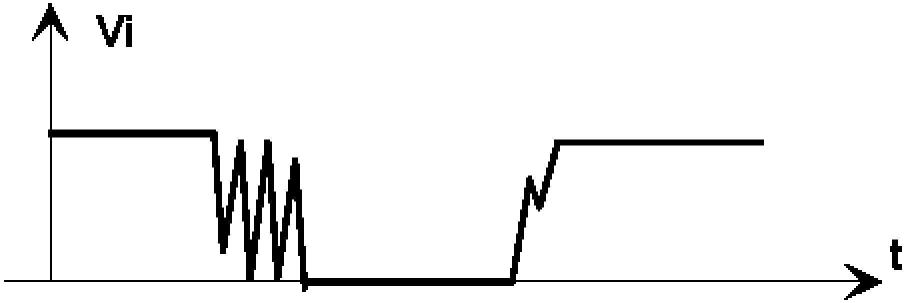


DIGITAL INTERFACING: FILTERING

On off signals

•This kind of signals are produced by relays, electronic switches, buttons, limit-switches

•Possible *spurious* states irrelevant if the signal level is required but misleading if we are interested in the commutation (i. e. an associated *event*)



•*electric* vs *algorithmic* filtering

–*electric*: required if interrupt o counters are used (⇒ RC + Schmitt trigger)

–*algorithmic*: necessary if the signal is sampled. The sampling will turn out in a series of values among which some are significant others not (spurious). The choice among them is driven by the information that the signal carries ⇒ definition of a suitable *transient death time*

DIGITAL INTERFACING

Fine corsa a leva, stelo, pulsante

Lever, stem, push-button,

Fine corsa ad albero

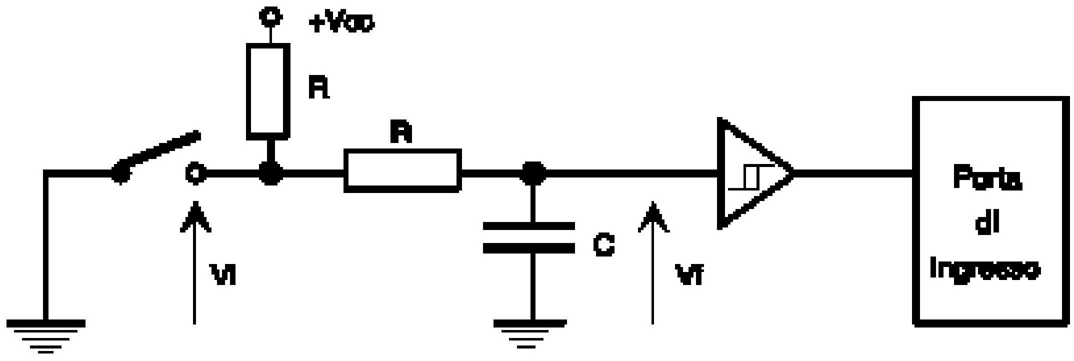
tree limit switches

DIGITAL INTERFACING

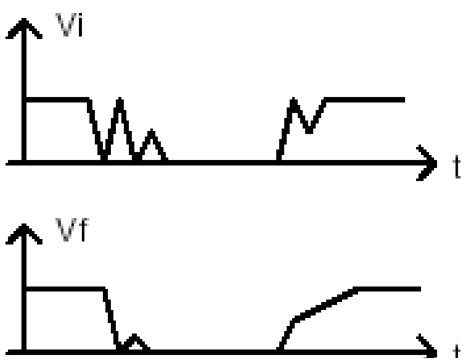
Electric filter through RC circuit + Schmitt trigger

Choose carefully the values of R and C, since they influence the circuit time constant and thus the reliability of the microprocessor acquisition

Slow capacitor charge/discharge could imply a slow crossing in the uncertainty range, bringing an oscillating behaviour. Therefore a Schmitt trigger could be suitable.



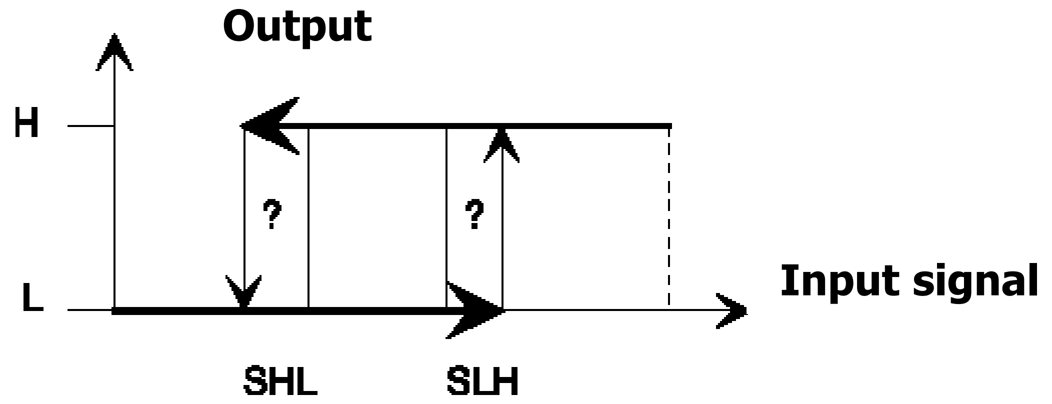
Simple filtering circuit for input signal to be read by a μP

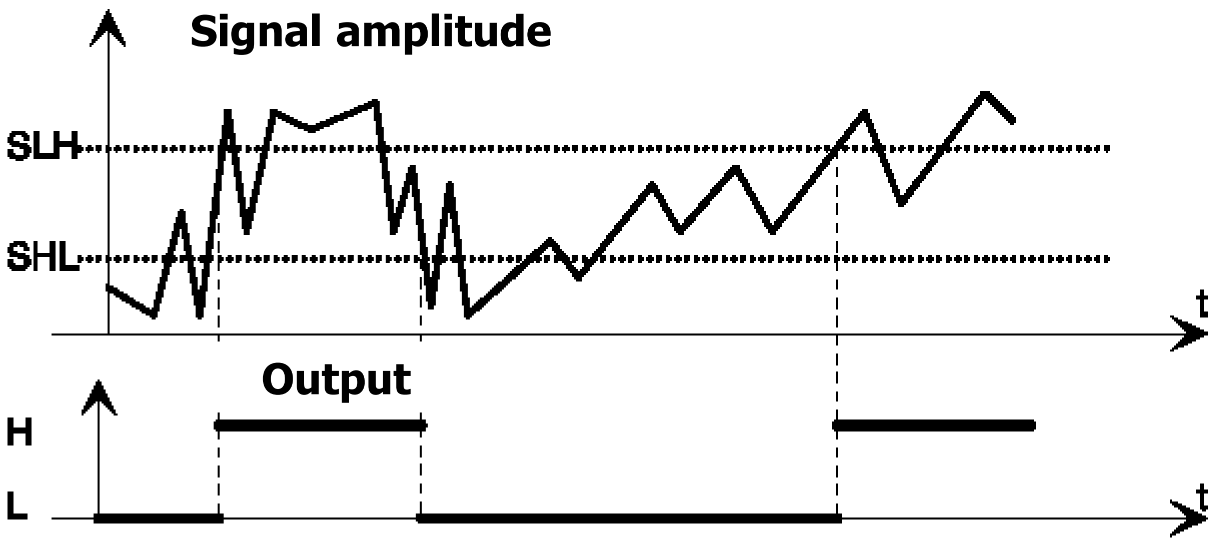


Input waveforms before and after filtering

Schmitt Trigger (Filter)

- Its characteristic is an hysteresis
- It depends on the behavior of a magnitude (memory effect): different output according to the direction of variation
- Useful in case of slow spurious signals and hw detection of a commutation
- The input signal is "squared" and is more suitably distinguished by the port electronic circuits

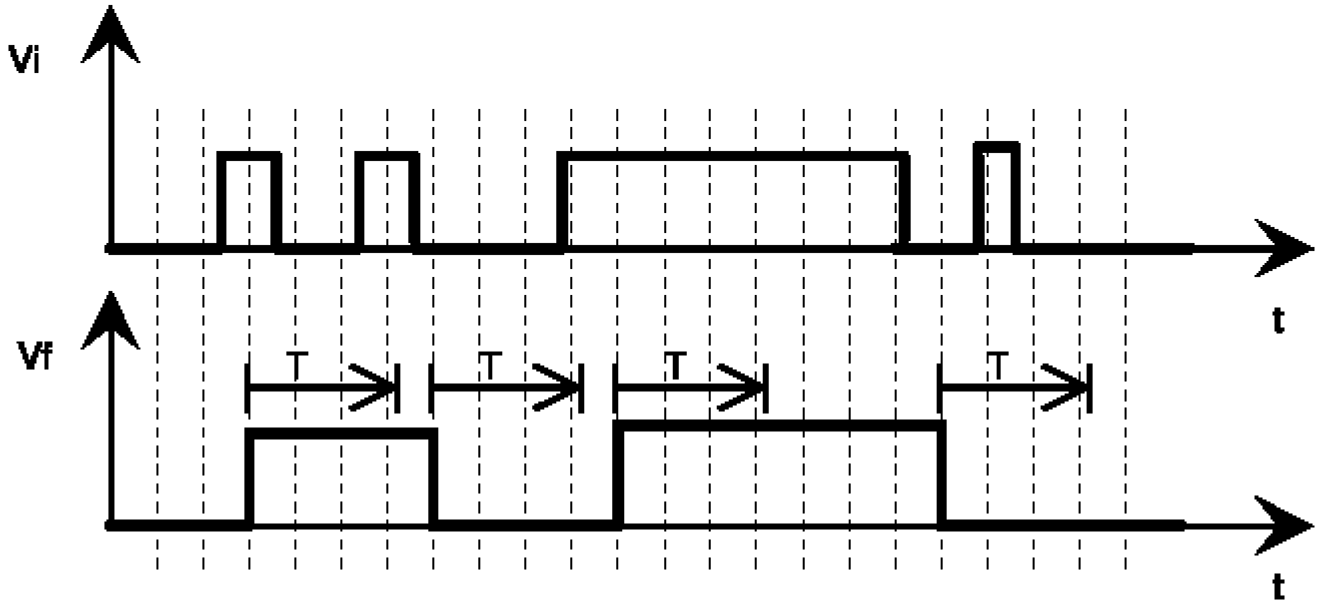




DIGITAL INTERFACING

Algorithmic filtering: first edge detection

- The first variation is considered as really significant and accepted
- Further variations are ignored during the time interval T (*transition death or dead time*)
- The algorithm does not work with isolated spurious signals
- Spurious signals follow a real significant event and run out within the T interval. Ok in case of mechanical contacts

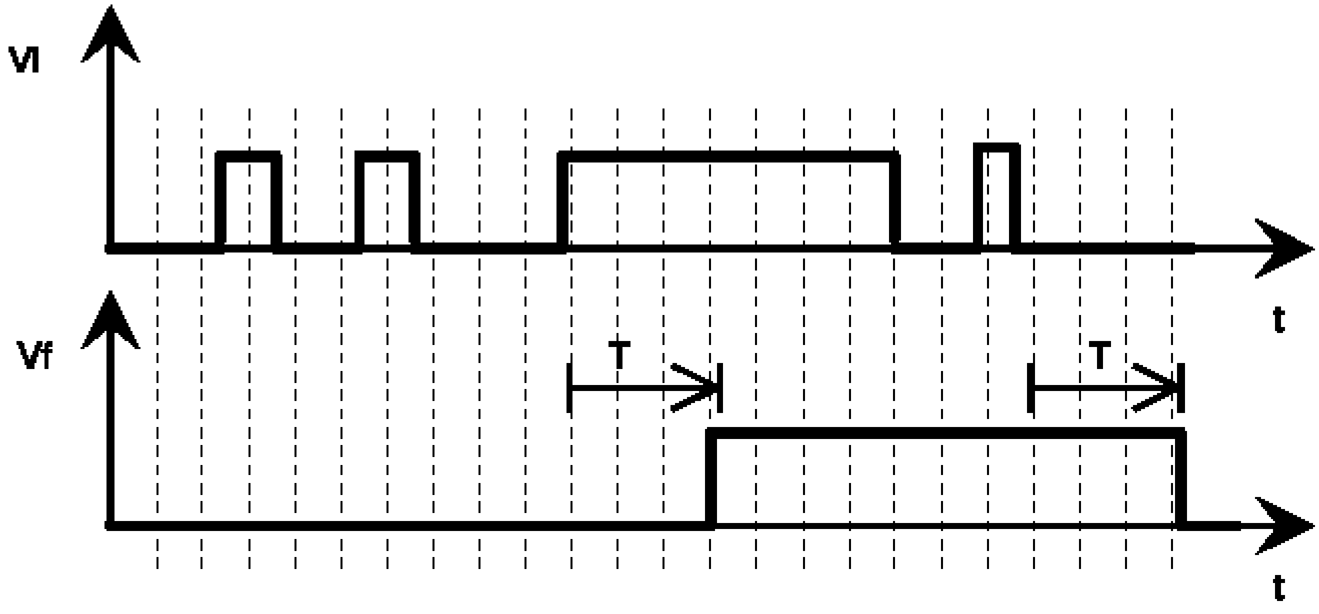


First edge detection, a simple assembly code

```
; Routine da chiamare ciclicamente ad ogni periodo di campionamento.
; Riporta in AL il valore filtrato
ACQ_BIT_N:
        CMP     [TRANSITORIO],0
        JE      REGIME
;qui in transitorio - conta il tempo trascorso
        DEC     [TRANSITORIO]
        JMP     ESCI
;qui transitorio esaurito - pronto a sentire event. commutaz.
REGIME:
        IN      AL,PORTA_IN             ;legge la porta
        AND     AL,MASK_N               ;isola il bit considerato
        CMP     AL,[PRECEDENTE]         ;confronta con valore preced.
        JE      ESCI
;qui rilevata commutazione
        MOV     [PRECEDENTE],AL         ;aggiorna il valore
        MOV     [TRANSITORIO],T_MORTO   ;rilancia il transitorio
ESCI:
        MOV     AL,[PRECEDENTE]         ;riporta il valore valido
        RET
```

Algorithmic filtering: expired transient detection

- The first variation is not significant
- The corresponding state is accepted as valid only if unchanged within T time interval
- The algorithm works well with isolated spurious signals
- A low pass filtering effect (delay)



DIGITAL INTERFACING

Exhausted transient detection, a simple assembly code

```
; Routine da chiamare ciclicamente ad ogni periodo di campionamento.
; Riporta in AL il valore di ingresso filtrato
;----- RILIEVO VALIDO A FINE TRANSITORIO----
ACQ_BIT_N:
        IN      AL, PORTA_IN                ;legge la porta
        AND     AL, MASK_N                  ;isola il bit considerato
        CMP     AL, [PRECEDENTE]            ;confronta con valore preced.
        JE      STABILE
;qui rilevata commutazione
        MOV     [TRANSITORIO], T_MORTO      ;rilancia il transitorio
        MOV     [PRECEDENTE], AL            ;aggiorna il valore
        JMP     ESCI
STABILE:
        DEC     [TRANSITORIO]               ;conta il tempo trascorso
        JNZ     ESCI
;qui terminato il transitorio - accetta la commutazione
        MOV     AL, [PRECEDENTE]            ;prende il valore stabile
        MOV     [VALIDO], AL                ;e lo considera valido
ESCI:
        MOV     AL, [VALIDO]                ;valore da riportare
        RET
```

DIGITAL INTERFACING

On off signals emission

- **The output on a μP port is carried out through latches (permanent values)**

- **Output of initial setup values or after a reset (0 if possible)**

- **Or simply those values that will manipulate the process through the actuator**

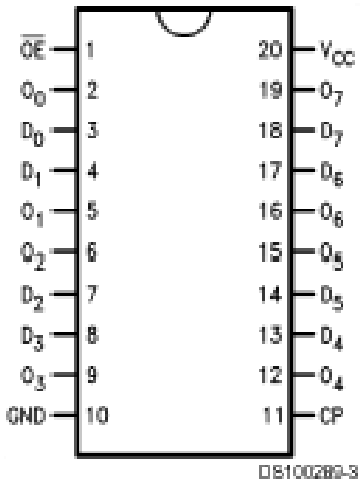
- **Output through "image" variables (it is not possible to output single bits)**

- **Image variables processed through AND/OR/XOR masks**

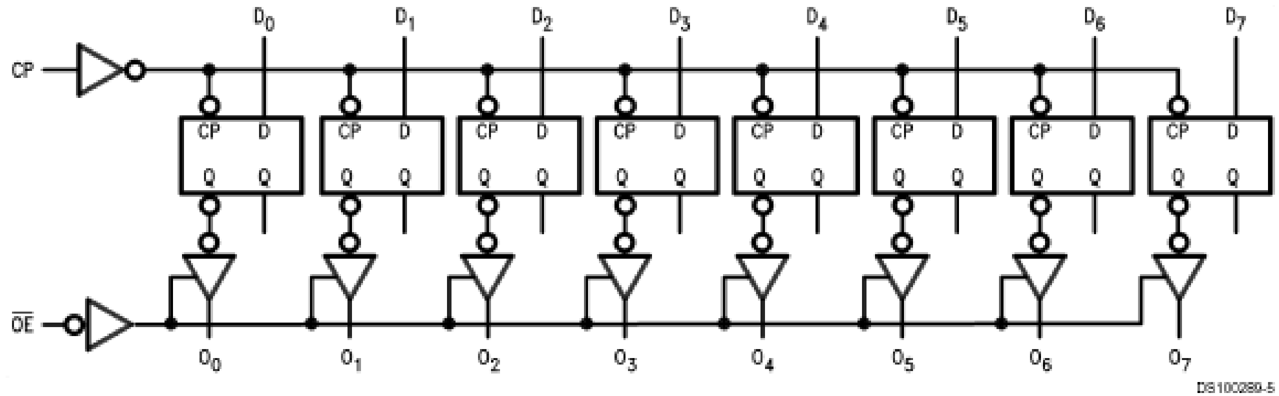
- **the National Instruments Latch 54ACT374 component**

DIGITAL INTERFACING

Pin Assignment for DIP and Flatpak

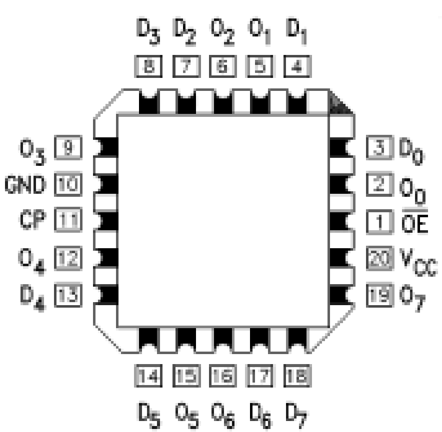


Logic Diagram



Please note that this diagram is provided only for the understanding of logic operations and should not be used to estimate propagation delays.

Pin Assignment for LCC



Truth Table

Inputs			Outputs
D_n	CP	$\overline{OE}$	O_n
H	⌐	L	H
L	⌐	L	L
X	X	H	Z

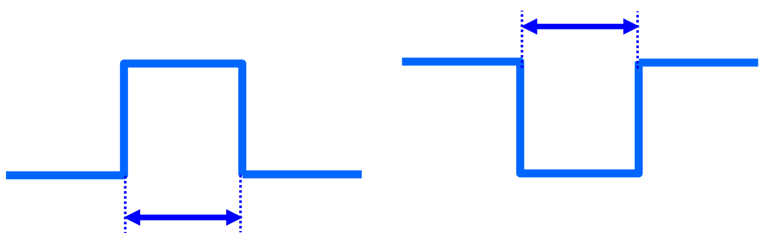
H = HIGH Voltage Level
L = LOW Voltage Level
X = Immaterial
Z = High Impedance
⌐ = LOW-to-HIGH Transition

Latch 54ACT374

National Instruments

DIGITAL INTERFACING

Pulses



Binary digital signal commutations to which a single event is associated considered as significant

Sometimes both the edges are useful since associated to the beginning and the end of a event ⇒interval measurement

Pulses can be acquired for:

- **Temporization**
- **Counting a suitable number of moving objects**
- **Calculating object position**
- **Calculating object velocity**

Pulses can be emitted for:

- **PFM device driving**
- **PWM device driving**
- **Power supply to an electric load through partialization of the signal**

DIGITAL INTERFACING

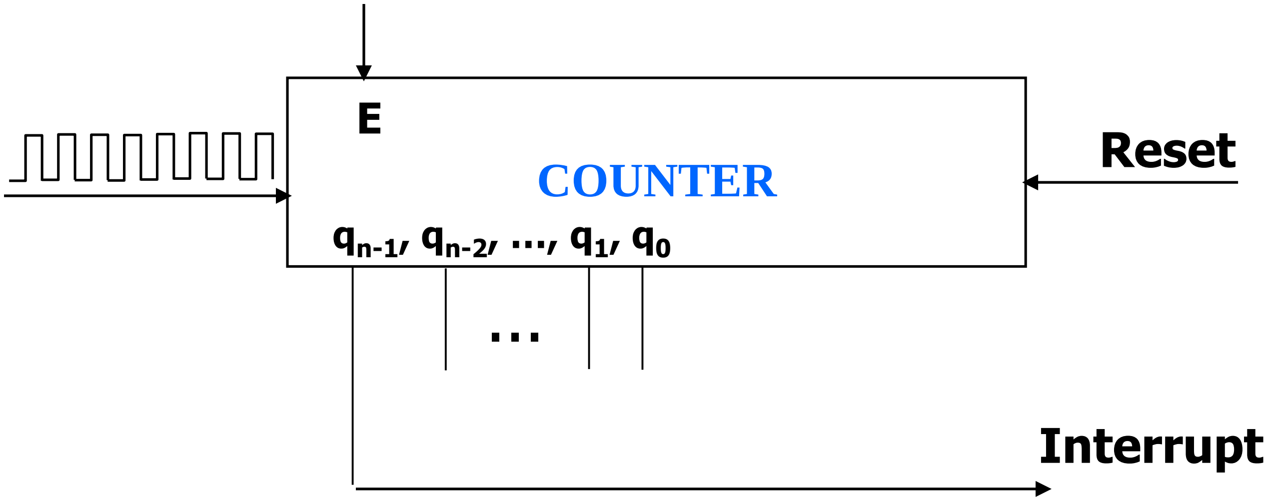
Pulse acquisition for counting (object passage, complete rounds …)

- **LED + phototransistor to reveal the object passage**
- **Total primary counter must be upgraded (i. e. program variable)**
- **Max reachable value => type definition => unsigned long (32 bit)**
- **Non volatile counting (eventually secondary counters in powered buffer memories / files. Trade-off between freshness and CPU overload)**
- **Manual vs automatic reset operations**
- **Variables must be stored through atomic saving operations in memory**

DIGITAL INTERFACING

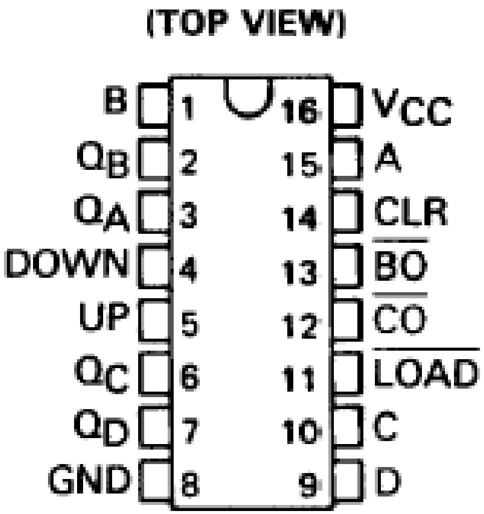
Hw primary counter

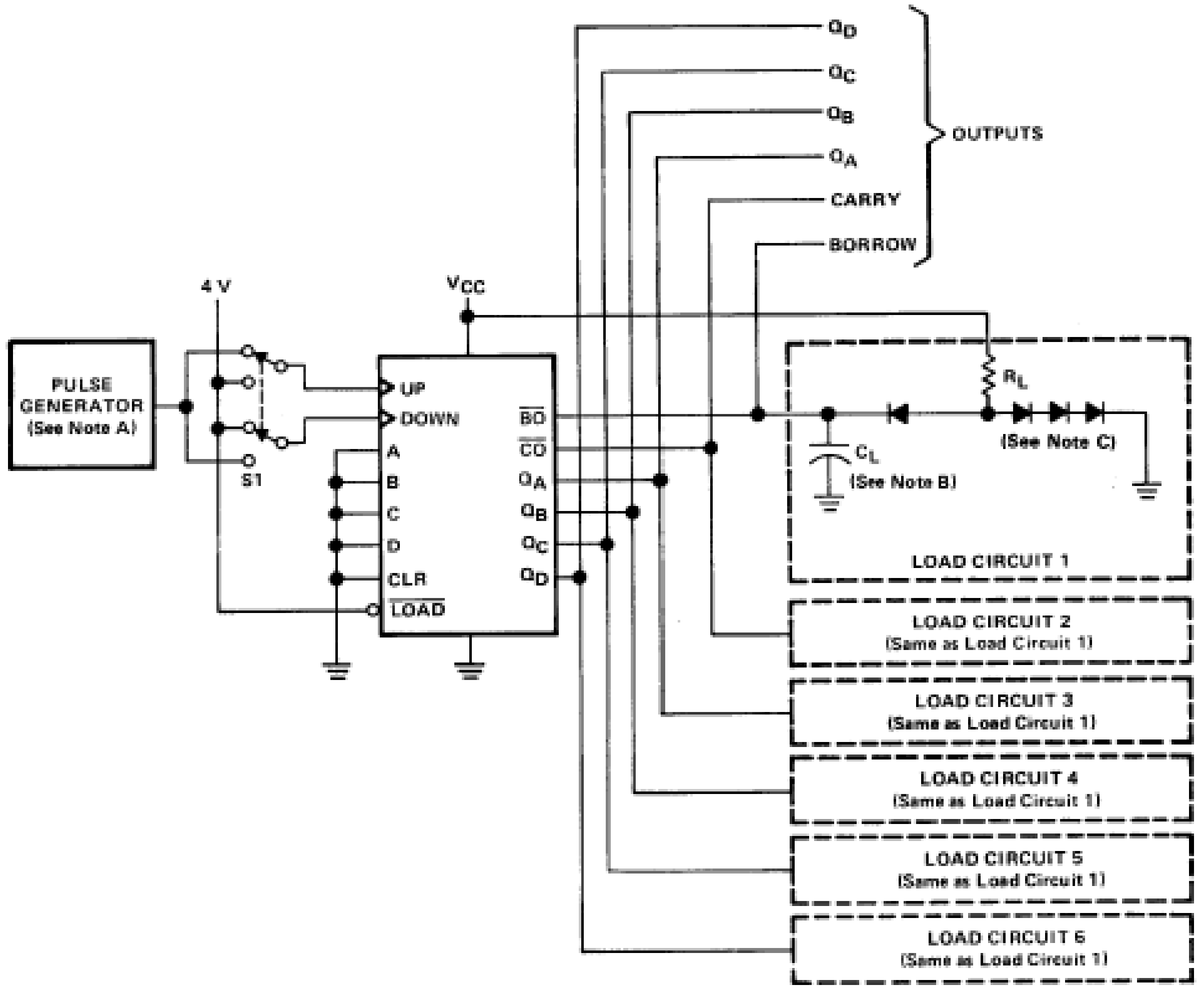
- Less sw elaboration charge onto the CPU
- Mandatory if pulse frequency not acquirable through polling from the μP port (it requires few assembly instructions)
- If the counter is connected with a μP port we read a state information
- If the counter is connected with an interrupt line (triggers the interrupt) we read an event information



DIGITAL INTERFACING

Hw primary counter: a commercial component Texas SN74193





- **Clear pin to reset**
- **Up/Down pins for setting bidirectional inputs**
- **Load pin to values preset**
- **A, B, C, D preset inputs**
- **Borrow/carry pins**

DIGITAL INTERFACING

External (primary) hw counter providing a "status" information

N bit hw counter

- **μprocessor port connection**
- **Acquisition routine periodic activation T_c**

where $T_c \leq (2^N-1)*T_{imp}$

Input from the port (present counter value)

 # pulses between two inputs (V_{new} - V_{old}

added to a total (secondary) counter (carry management)

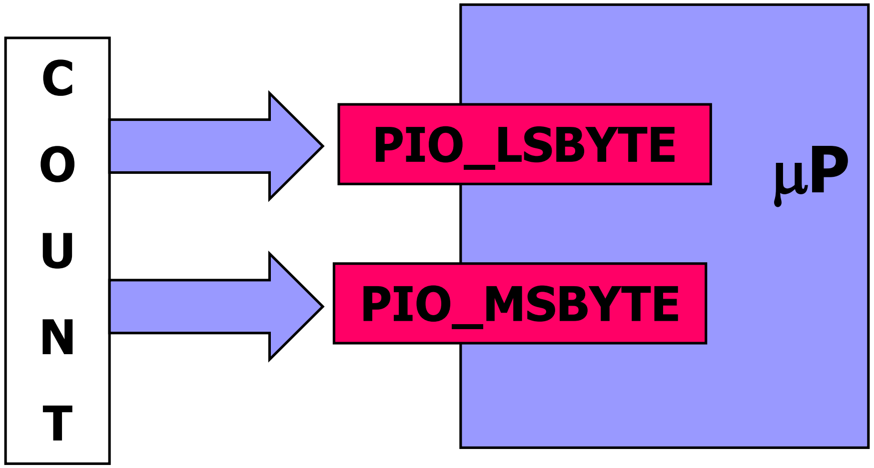
```
AGGIORNA_CONTATORE:
        SUB  AX,AX
        IN   AL,PORTA_CONT
        SUB  AL,[PRECEDENTE]
        ADD  [PRECEDENTE],AL
        ADD  [CONTATORE],AX
        ADC  [CONTATORE + 2],0
        RET
```

DIGITAL INTERFACING

External (primary) hw counter providing a “status” information

If n° of bits > than readable by a single instruction

- **Bytes from the counter are read with successive read phases**
- **Auxiliary register with the same width of the counter to be read in two phases**




```
AGGIORNA_CONTATORE:
            IN      AL,P_MSBYTE
RIPETI:
            MOV     BH,AL
            IN      AL,P_LSBYTE
            MOV     BL,AL
            IN      AL,P_MSBYTE
            CMP     AL,BH
            JNE     RIPETI
;qui BX = valore valido
            MOV     AX,BX
            SUB     AX,[PRECEDENTE]
            ADD     [PRECEDENTE],AX
            ADD     [CONTATORE],AX
            ADC     [CONTATORE + 2],0
```

DIGITAL INTERFACING

External (primary) hw counter providing an "event" information

- **Interrupt line connected to a single counter bit or to the carry bit**
- **Used as "pre-scaler" (frequency division)**

$$F_{out} = F_{in} / 2^N \text{ (if carry bit connected)}$$

$$F_{out} = F_{in} / 2^k \text{ (if } k^{th} \text{ bit connected)}$$

- **Memory SW counter upgrading (if high frequencies)**
- **Least significant count directly read from the counter**

SW counter	Direct acquisition

DIGITAL INTERFACING

Pulses to evaluate an object position

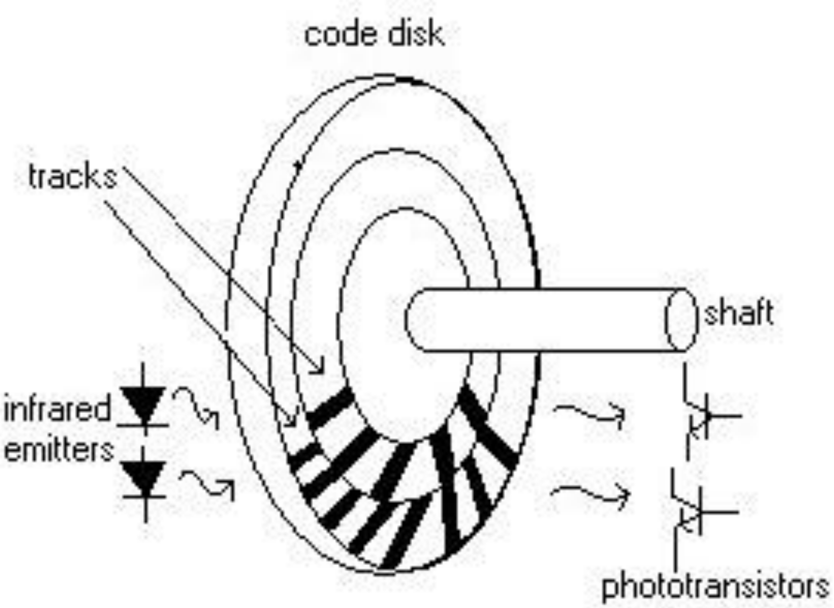
- **Pulses corresponds to incremental movements. Where is the “zero” point?**
  - **Is it an un-surmountable extreme? Unsigned numbers**
  - **Is it an arbitrary position? Signed numbers**

- **The incremental encoder**

A wheel made up by windows alternatively transparent and opaque to a light beam emitted by a LED source

The light is received by a phototransistor

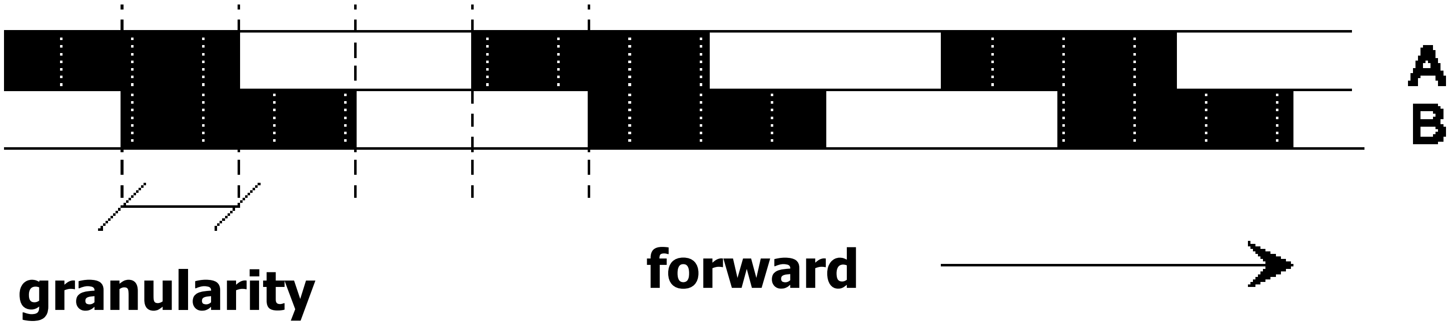
Each window provides a binary information (i. e. a bit)



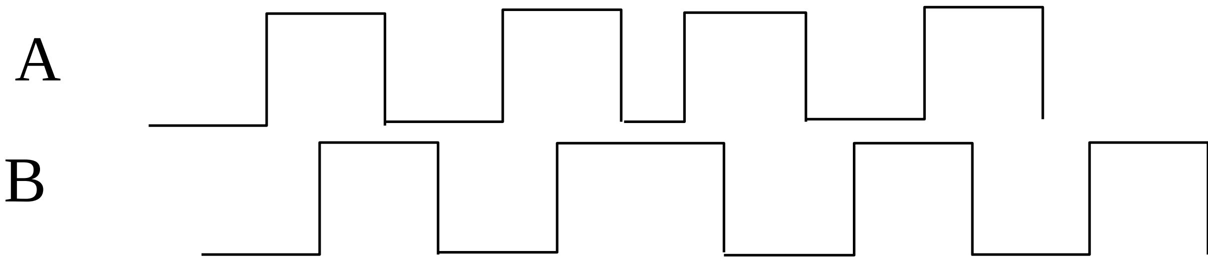
DIGITAL INTERFACING

Incremental Encoder

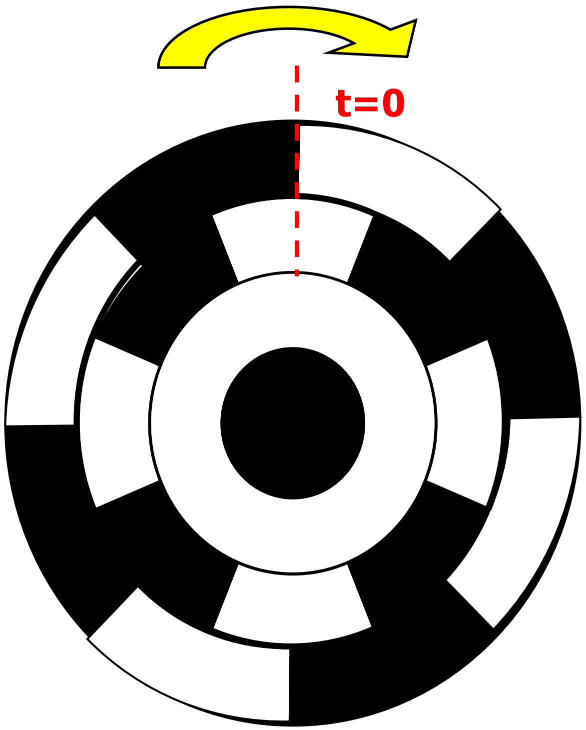
- **Two concentric crowns, 1/4 period out of phase**
- **Two couples of LED/transistor (1 light, 0 dark).**
- **Minimum measurable angle = half window**



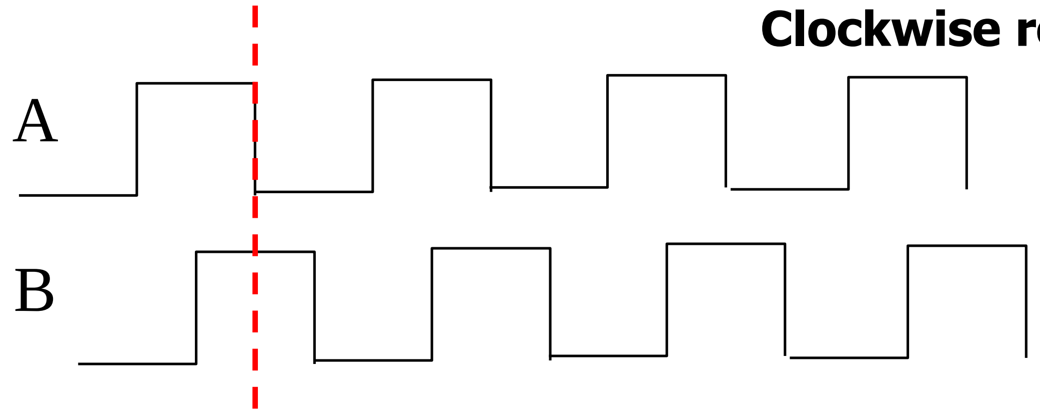
The two acquired signals (pulse trains) as they appear if the direction is reversed



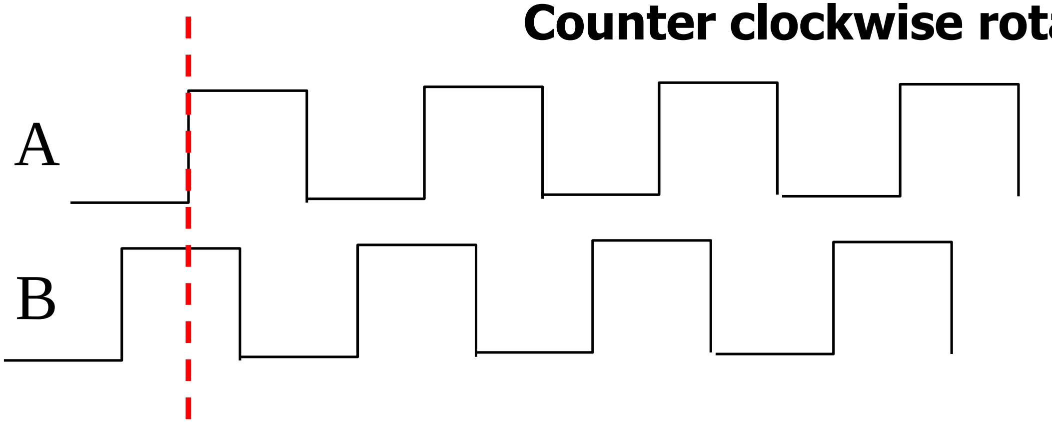
Incremental encoder



Clockwise rotation



Counter clockwise rotation



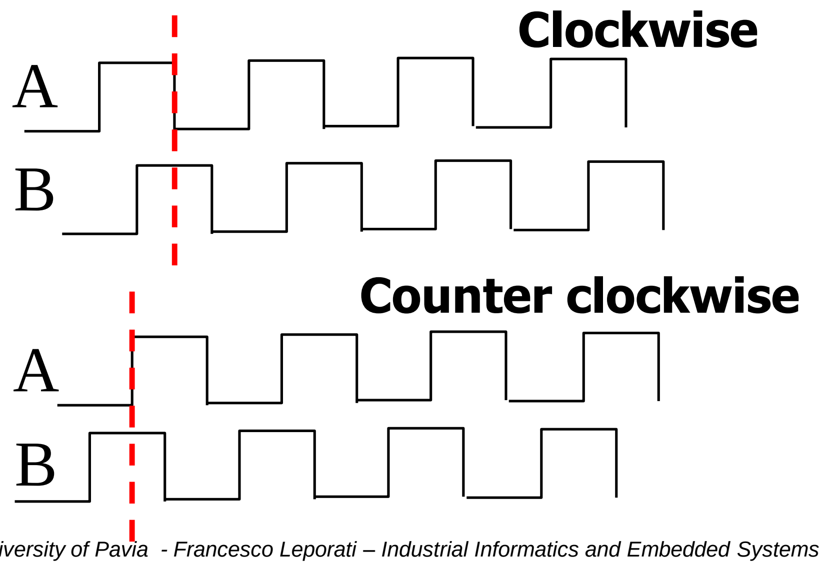
DIGITAL INTERFACING

Counting specs

- definition of “zero” (natural/integer counting variable)
- maximum counting vs granularity (size)

Counting

- edge detection
- movement direction
- primary counter upgrading (then eventually upgrade the secondary one)
- 2 possible approaches: sw/hw

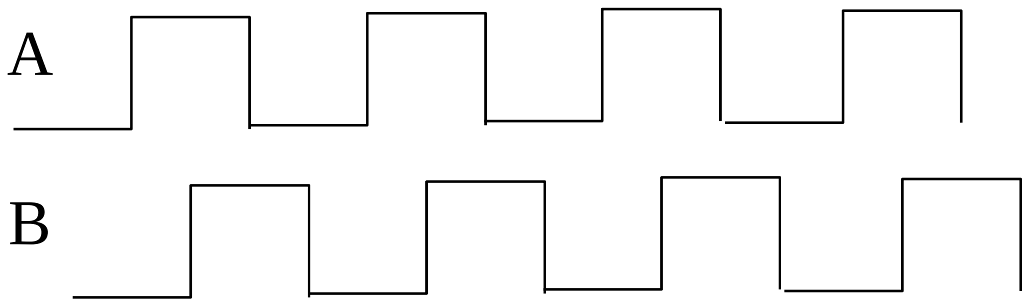


A	B	C
P	0	+1
P	1	−1
N	0	−1
N	1	+1
0	P	−1
1	P	+1
0	N	+1
1	N	−1

VELOCITY CALCULATION THROUGH AN ENCODER – sw approach

- **A and B: 2 pulse trains connected to two pins of the μP port**
- **Commutations detection**
- **Movement direction identification: forward (+) or backward (-)**
- **Counter variables (primary and eventually secondary)**
- **algorithm (hypothesis clockwise rotation considered as positive)**

```
if B switched
        if B=A count++
        else count--
else if A switched
        if A=B count --
        else count ++
end if
```



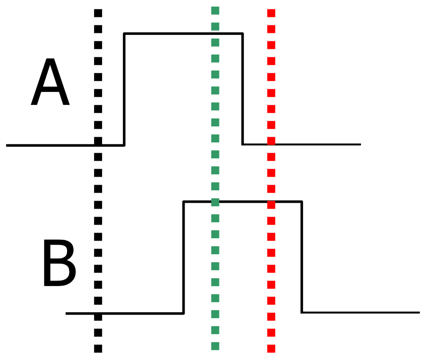
DIGITAL INTERFACING

An example of sw management of a bi-directional incremental encoder

```
AGGIORNA_POS_ENCODER:
        IN      AL,P_ENC
        AND     AL,3                    ;maschera i bit utili
        MOV     BL,AL
        SHR     BL,1                    ;BL.0 = B (porta segnale B in bit 0
        XCHG    AL,[PRECEDENTE]
        XOR     AL,[PRECEDENTE]
        JZ      ESCI
;qui rilevata una commutazione
        AND     AL,1
        MOV     CL,AL                   ;CL.0 = 1 se commutato A
        MOV     AL,[PRECEDENTE]
        AND     AL,1                    ;AL.0 = A
        XOR     AL,BL
        XOR     AL,CL
        JNZ     DECREMENTA
;qui si deve incrementare
        ADD     [CONTATORE],1
        JMP     ESCI
DECREMENTA:
        SUB     [CONTATORE],1
ESCI:
        RET
```

AL = A ⊕ B	CL = Anew ⊕ Aold	AL ⊕CL	ACTION	notes	rotation
0	0	0	+1	B↑ Anew=B	CW
1	0	1	-1	B↑ Anew≠B	CCW
0	1	1	-1	A↑ Anew=B	CCW
1	1	0	+1	A↑ Anew≠B	CW

Routine activation frequency?



We have to distinguish between:

case ok B(t-1)≠B(t) with A=1, (cw)

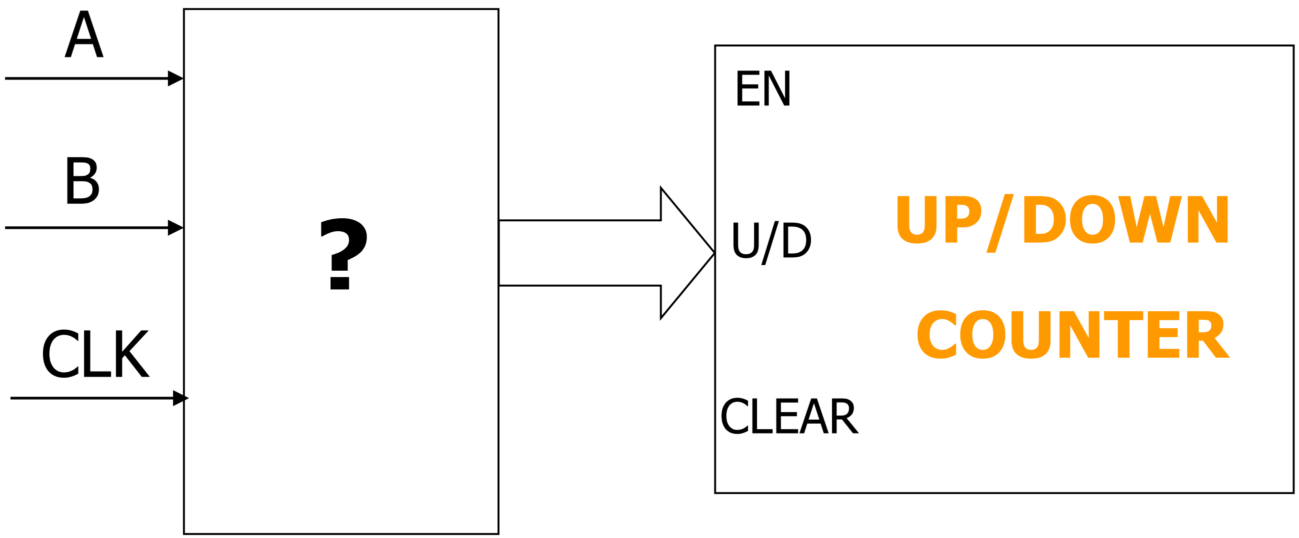
case ko B(t-1)≠B(t) with A=0 (ccw)

F_{clk}= 2*F_s

This on both the A e B signals

Fclk= 4*Fs

Hw management of a bi-directional incremental encoder



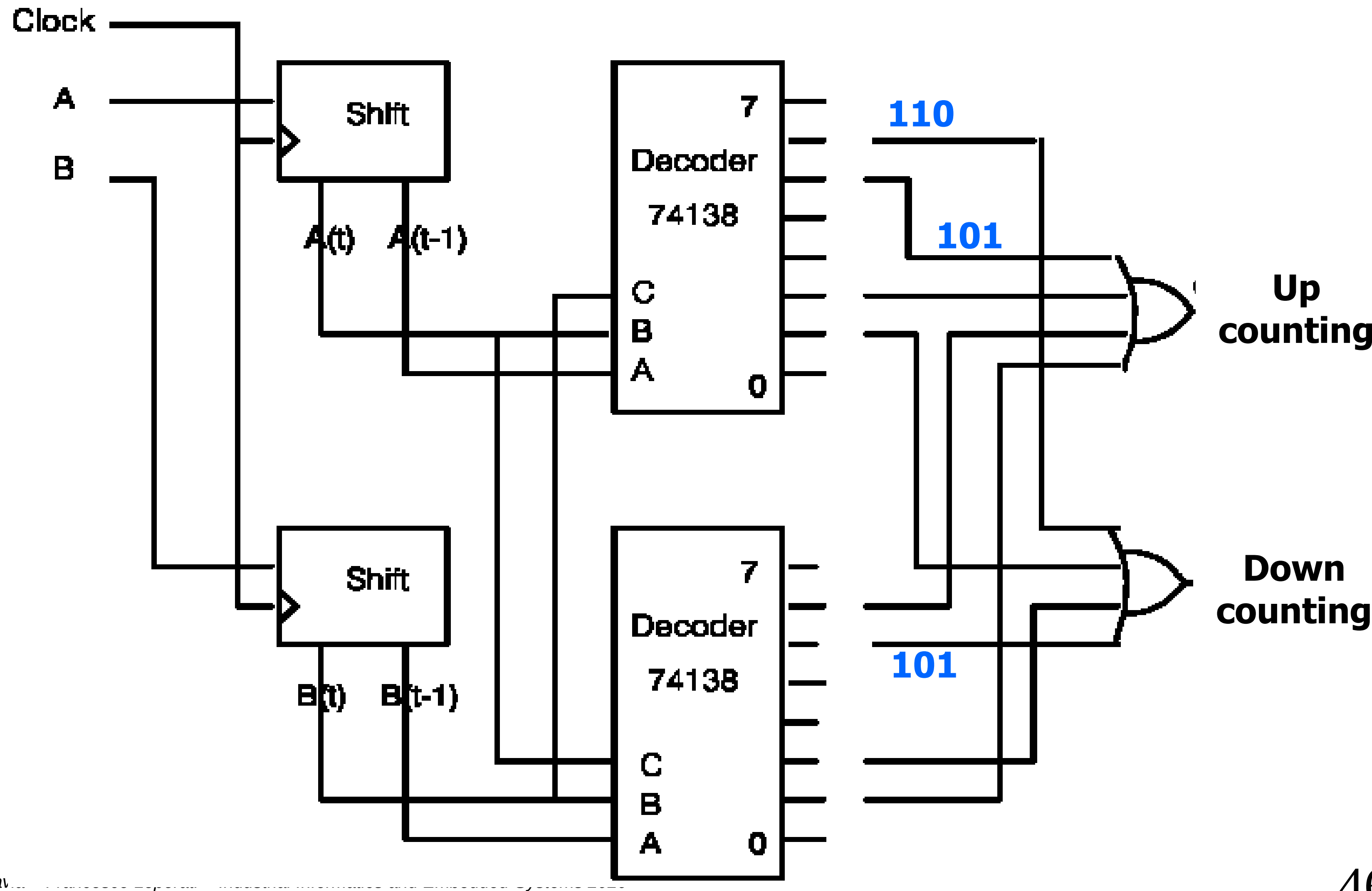
Activation frequency?

The hw logic must distinguish among commutations of the same signal (A o B); however, within the present commutation and the next one it must also recognise if the other signal is 0 or 1.

$$F_{clk} = 8 * F_s$$

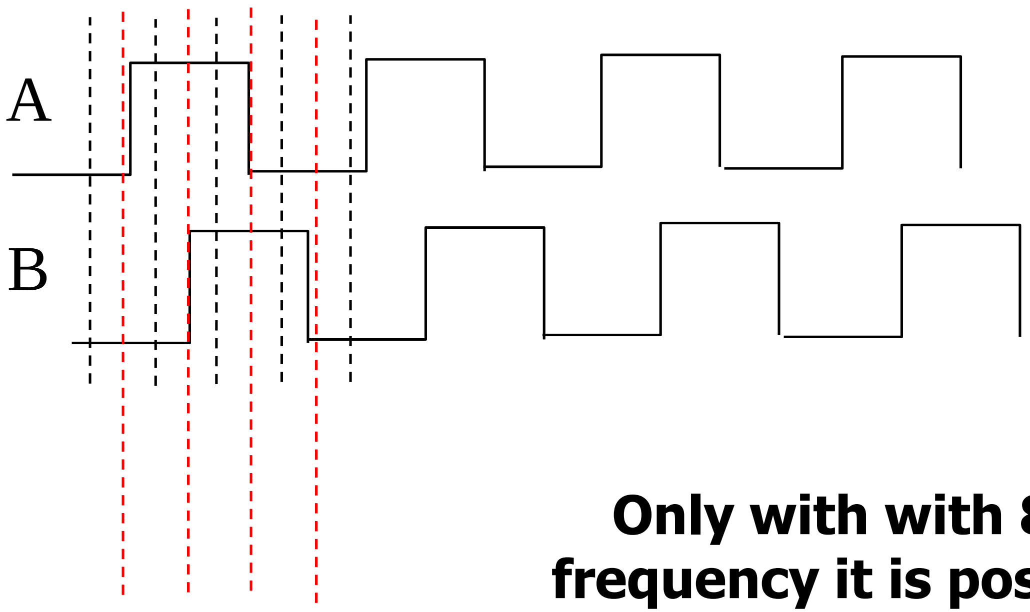
DIGITAL INTERFACING

Hw management of a bi-directional incremental encoder



DIGITAL INTERFACING

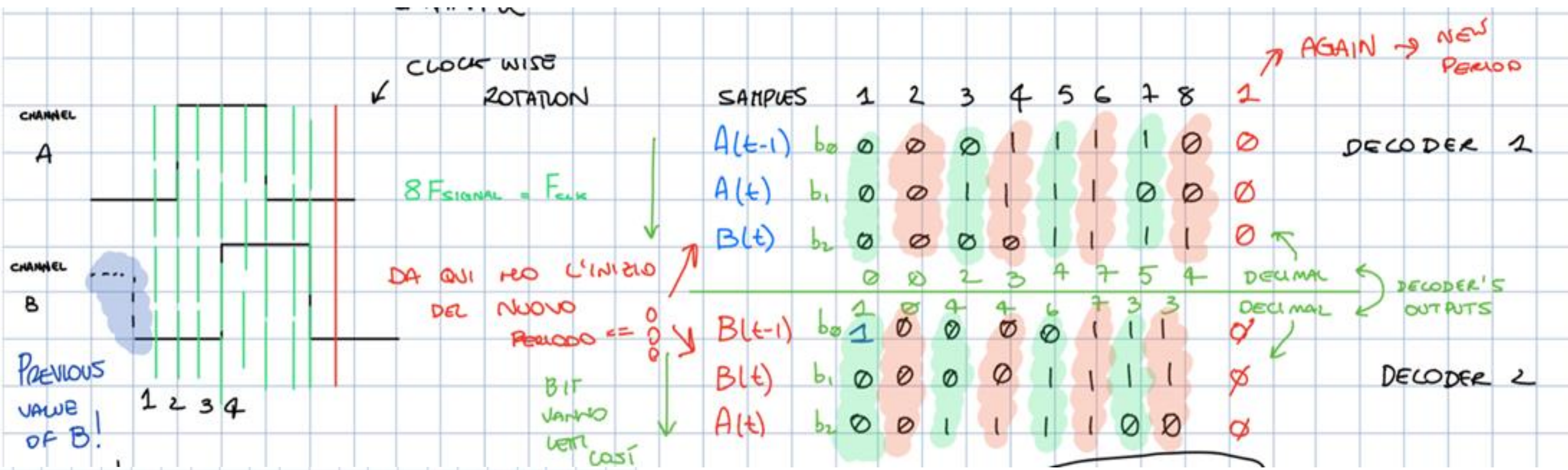
Hw management of a bi-directional incremental encoder

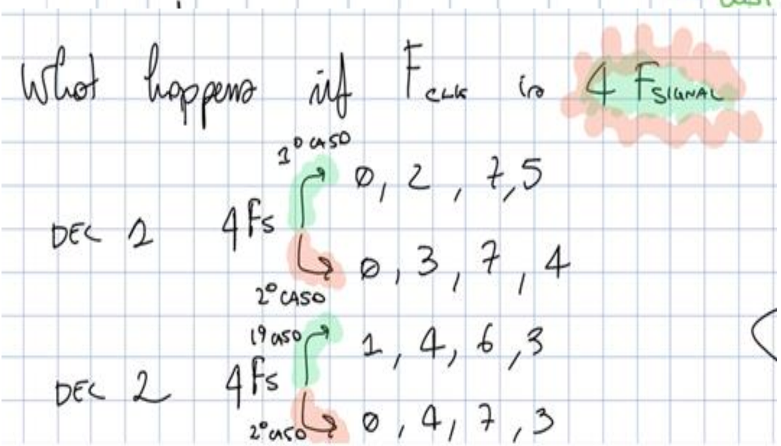


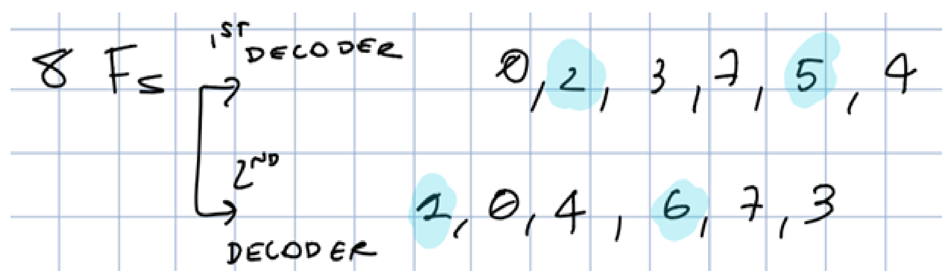
Only with with 8F_s sampling frequency it is possible to reveal useful commutations and the turnover among useful configurations and "sleep" states

Hw management of a bi-directional incremental encoder

CW rotation

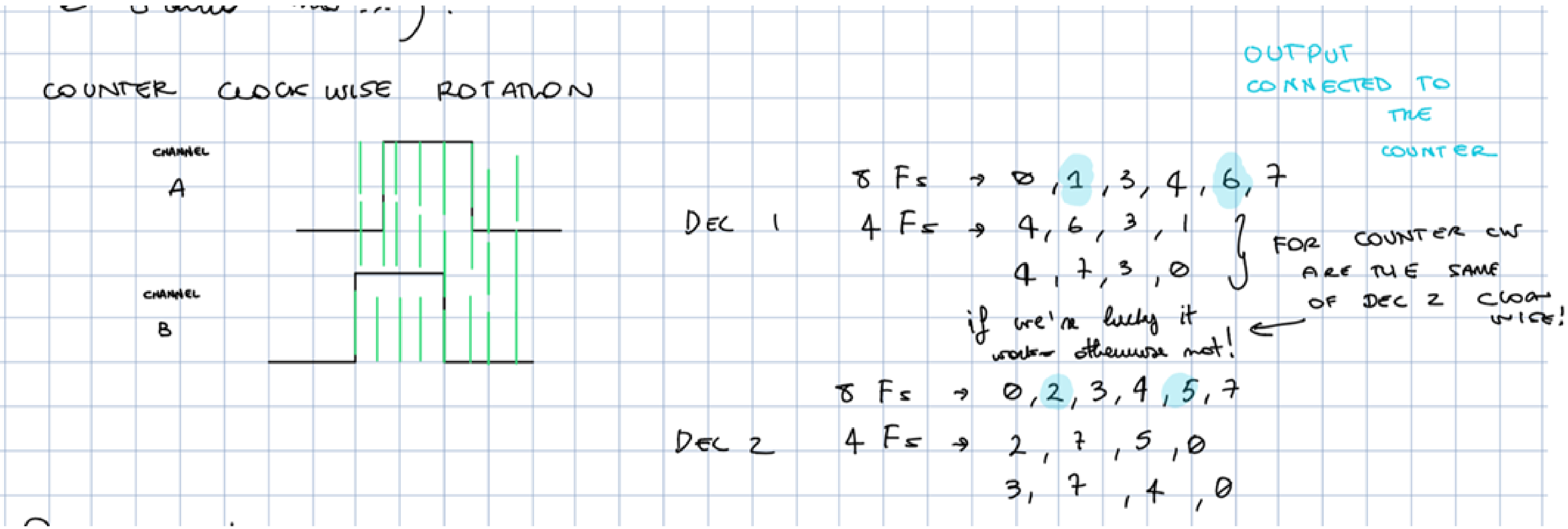




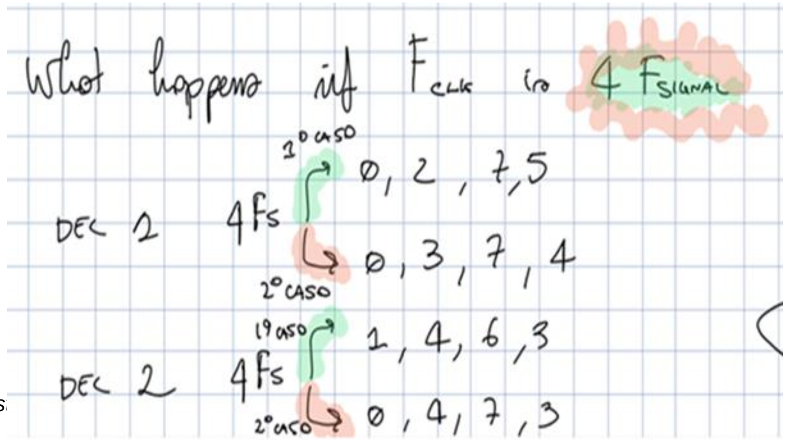


Hw management of a bi-directional incremental encoder

CCW rotation



CW rotation



DIGITAL INTERFACING

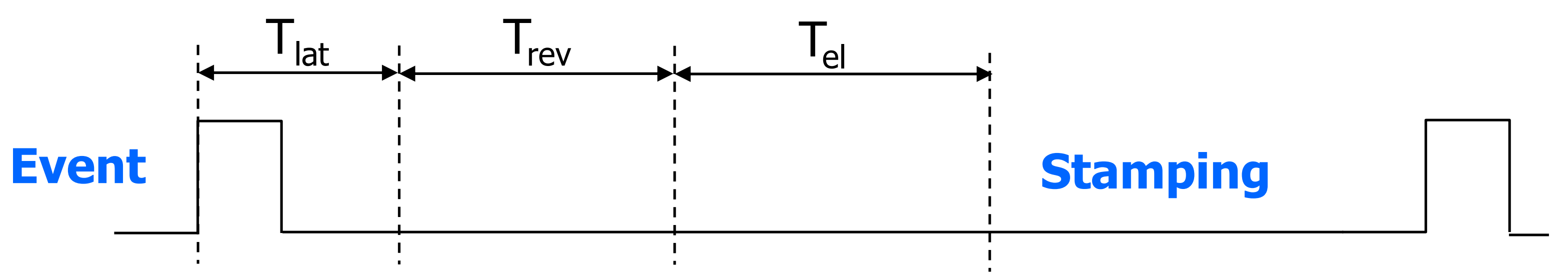
Pulses for timing

Time-stamping means to associate events and times into which they happen

Errors due to the model describing time evolution within the machine (TIG – temporal granularity)

Delays due to timer acquisition sw (T_{el}) and to event recognition (T_{rev})

Latencies = time interval between the event and its perception (T_{lat}). Non deterministic, if they are due to resources charged with task perception and in that moment devoted to other activities (priority). Typical in interrupt service and in processes activation



DIGITAL INTERFACING

Pulses for timing (maximum priority routine)

Events internal to the machine (i. e. switch of a control unit signal)

$$t\text{-}TIG+T_{el} < T_{st} < t+T_{el}$$

Events external to the machine (i. e. switch of a signal read through a port)

$$t\text{-}TIG+T_{rev}+T_{el} < T_{st} < t+T_{lat.max}+T_{el}+T_{rev}$$

Measurement of an interval between external events (constants components neutralise each other, deterministic ones do not disappear)

$$-TIG-T_{lat.max} < (T_{st1}-T_{st2}) < TIG+T_{lat.max}$$

What is TIG? It depends on the machine time representation model.

Let us open a parenthesis …

TIME CHARACTERISATION

- **Is the time an event container (box) or the events cause the course of the time?**

- **During life we follow the first approach, in computer science we are close to the second one**

- CLOCKS **= phenomena that generate regular events (ticks) used to quantify the time passing**

- **We need a model**

TIME AS IT IS REPRESENTED WITHIN A CPU

- **Hp: bijective correspondence among time moments and real numbers**
- **Continuity, limit, derivative, integral existence ecc.**
- **Time as linear, monotonic, discrete**
- **Clock = time events generator with domain N^+**
- **Quantization = duration of the phenomenon taken as clock, usually very negligible with respect events that must be measured**

TIMING MODEL

- **LOCAL TIME UNIT (UTL) = absolute time interval corresponding to 1 in the measurement unit of the clock**

- **LOCAL TIME BASE (BTL) = multiple of the UTL, representing the period of the cyclic phenomenon assumed as clock**

- **Let' suppose that in a PC the real time clock is 55 msec: the time is represented in msec (UTL), but it is increased every 55 (BTL)**

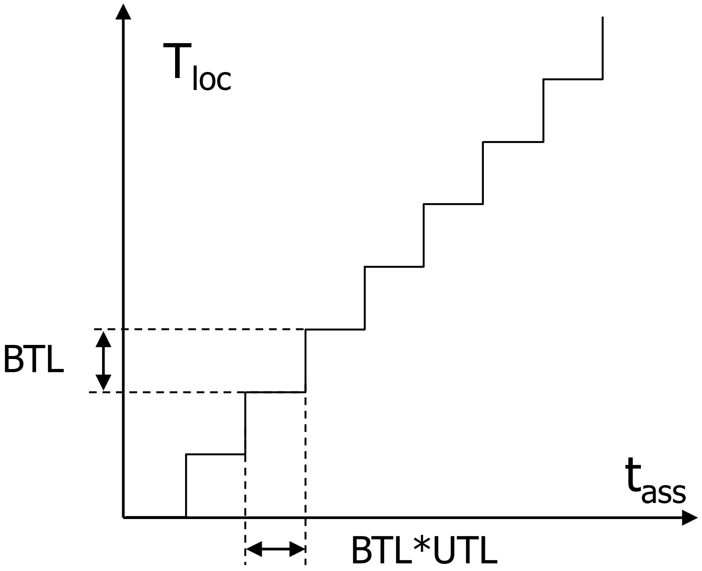
- **UTL = resolution, BTL = granularity**

- **If BTL=1, resolution = granularity**

- **If T = clock time and t = absolute time,**

$$T = n * BTL$$

$$n * BTL * UTL \leq t \leq (n+1) * BTL * UTL$$



TIME MODEL: PRECISION ERRORS

- **They are due to how the phenomenon (assumed as clock) differs from the ideal behavior**

- **PRECISION**

  A clock is precise if t(tick(n)) - t(tick(n-1)) = BTL*utl + e(n) with e(n)=0

  – **Systematic error ⇒ the phenomenon taken as clock features a not correct frequency**

  – **Null mean errors ⇒ not stable frequency (jitter)**

  – **precision measurement = BTL*utl /mean_err = (10^4-10^6)**

- **CORRECTNESS**

  A clock is correct if abs(T-t/UTL) < BTL, that is the local time T is synchronised with the absolute time (within the quantization error BTL)

  –**A precise clock if is correct in t, is correct in t'>t**

  –**Sistematic unaccuracy (mean_err≠0) ⇒ no correctness**

  –**If intervals are measured, systematic errors do not exert any influence**

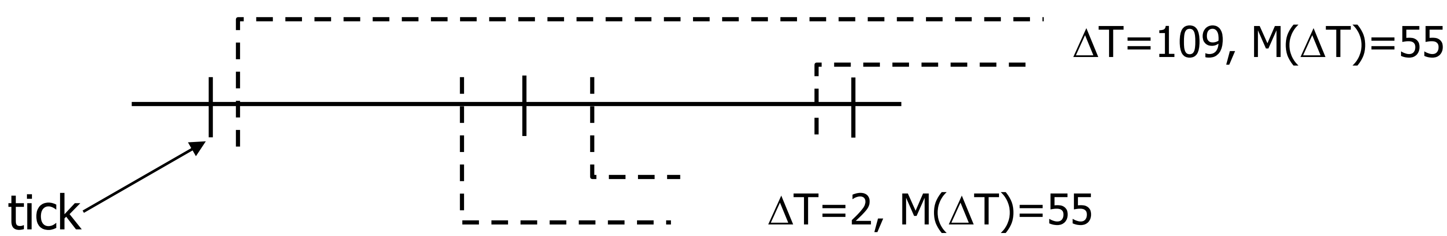
TIME MODEL: QUANTIZATION ERRORS

- **They are caused by the representation of the time**

- **ORDER: it is not possible to establish an order between 2 events that happen within 2 consecutive ticks (i. e. within the same BTL)**

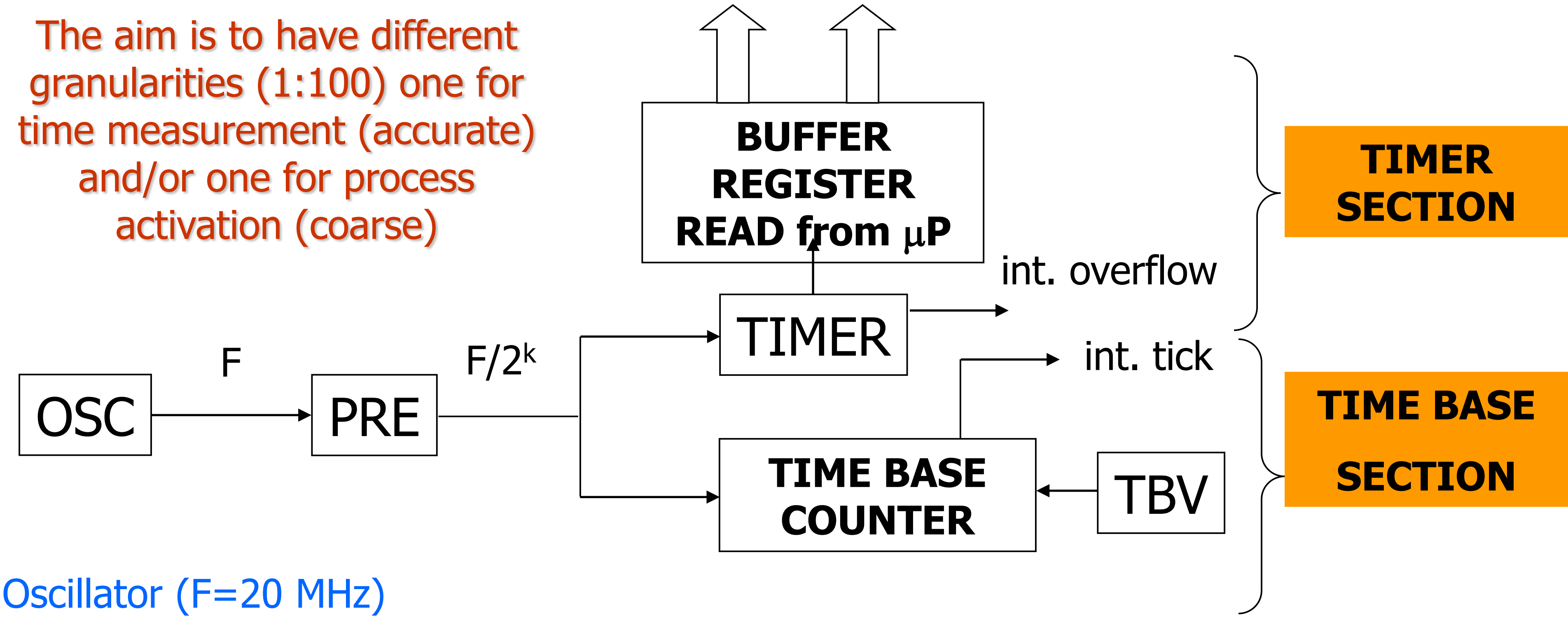
- **CONTEMPORANEITY: if the time interval between two events is $\Delta T<BTL$, the computer considers them as simultaneous (contemporaneous)**

- **MEASUREMENT:**
  - **2 events, whose real time distance is ΔT, are considered as within**
    $N*BTL < M(\Delta T) < (N+1)*BTL \qquad N=floor(\Delta T/BTL)$
  - **events, whose time distance is N*BTL, can correspond to a real time interval ΔT:** $(N-1)*BTL*UTL < \Delta T < (N+1)*BTL*UTL$



TYPICAL TIMING SYSTEMS (TIMERS)

The aim is to have different granularities (1:100) one for time measurement (accurate) and/or one for process activation (coarse)



Oscillator (F=20 MHz)

Prescaler (frequency division, 2^k=1-256)

TIMER = N bit binary cyclic counter (16-24)

TIME BASE COUNTER = 1 tick when the set period (TBV) ends + restoration TBV

Decrement counters

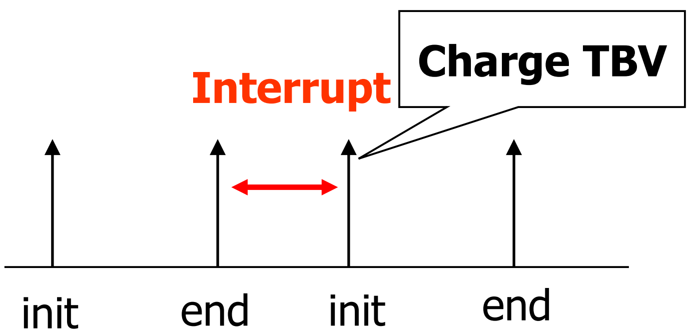
TYPICAL TIMING SYSTEMS (TIMERS)

• *TIMER section*

TIMER RESOLUTION (count units) = $2^k/F$=UTL

TIMER PERIOD = $2^N * 2^k/F$

BTL= 2^N (on the overflow)



$(COUNT+TBV) \bmod 2^M = COUNT+TBV-2^M$

$= 2^M - x + TBV - 2^M = TBV - x$

to include counts lost due

to interrupt latency

• *TIME BASE section*

TIME BASE COUNTER RESOLUTION = $2^k/F$=UTL

TIME BASE COUNTER PERIOD = TBV $* 2^k/F$

BTL=TBV

TBV RESTORE

• Set TBV automatically or …

• Explicit setup within the sw routine that manages tick interrupt but …

• $(CONTEGGIO+TBV) \bmod 2^M$

SOME IDEAS

F=1 MHz, Prescaler=1 ⇒ period 1μsec, if N=16 ⇒ 65000 μsec

F=1 MHz, Prescaler=256 ⇒ period 256μsec, if N=16 ⇒ 16 sec

F=20 MHz, Prescaler=256 ⇒ period 12.5μsec, if N=24 ⇒ 200 sec

Process activation	⇒ TIME BASE COUNTER (TBC)
Time stamping (date and time)	⇒ tick TBC
	⇒ overflow TIMER
Interval measurement	⇒ TIMER

NB. If long intervals are measured we need a sw variable onto which to store the counter values. This implies a partial un alignment between the total sw counter value and the hw register (containing something like a «fractional part» of the overall value).

DIGITAL INTERFACING

Pulses are acquired to calculate the velocity (i. e. frequency) of an object

Angular velocity or (scale factor) linear velocity can be determined

<u>Problem data</u>

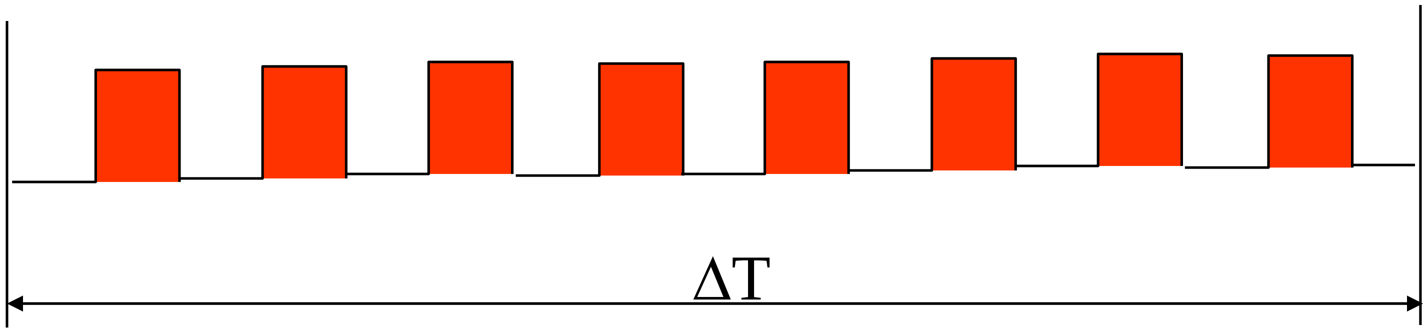
- **Encoder granularity = (round or meter/pulse)$^{-1}$ => quantization effect**
- **Maximum and minimum frequency to be measured**
- **Pulse (usually given by a sensor) duration**

<u>Specs</u>

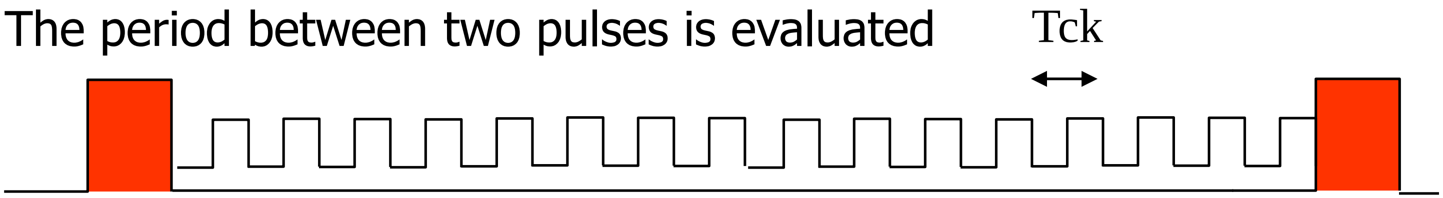
- **Measurement production frequency (acceptable depending on the particular phenomenon to be measured)**
- **Delay between the acquisition instant and the moment to which it is related**
- **Errors (relative vs. absolute, related to measurement value or to the full scale value, Erm% vs. Erf%)**

VELOCITY CALCULATION

Pulses are counted during a known time interval



The period between two pulses is evaluated



What is measured is an "average velocity"

Pulse acquisition techniques

•**Software**

Cyclic (Tc period) pulse acquisition

If time for reading negligible and Tg granularity of the used clock (BTL* UTL)

Error on the event detection: $-Tg<E<Tc$

•**Interrupt**

The signal carrying the pulse is connected to the interrupt pin of a microprocessor

$T_{lat.max}$ = max. latency of the interrupt service routine

Te = execution time of the interrupt service routine

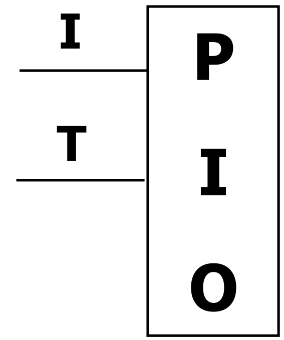
Error on the event detection: $-Tg+Te<E<T_{lat.max}+Te$

•**Hardware**

Dedicated circuits with working times negligible with respect to event duration

Error on the event detection: $-Tg<E<0$

Pulse count during time interval: I-SW, T-SW



•**Sw routine executed with period Tc:**

- Input port read
- if positive (raising up) edge I   count++;
- if positive edge T   vel=(enc. granularity)*counter_val/ΔT; count=0;

•**Constraints**

Tc< minimum pulse duration $\Rightarrow$ f Tc< 1/2 (f = pulse frequency)

f_P (measure production freq.) = 1/ΔT; f_{min} (minimum detectable frequency) > 1/ΔT

•**Errors**

DTM= real (effective) count interval; n_p=real n° of pulses DTM

f = real signal frequency = n_p /DTM; f_m = measured frequency = n (counter value) /ΔT

Quantization err. on pulse identification [-1 … 1] $\Rightarrow$ $n_p - 1 < n < n_p + 1$

Time err. on ΔT estimation [-Tc … +Tc] $\Rightarrow$ $\Delta T - Tc < DTM < \Delta T + Tc$

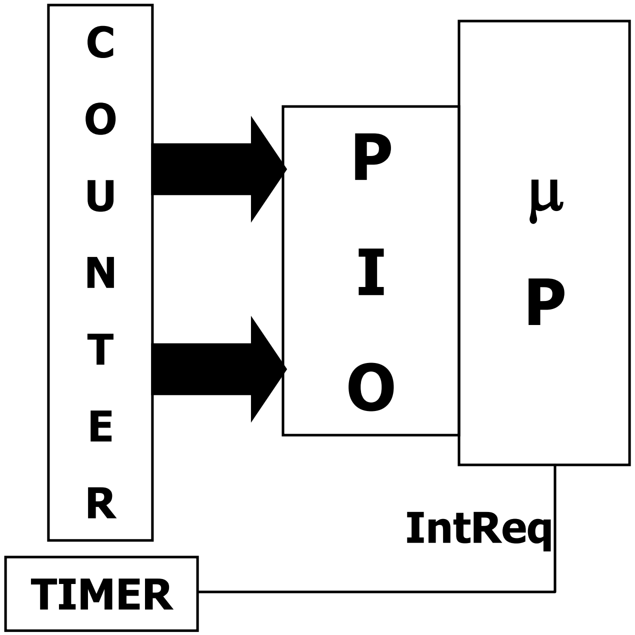
$$f(\Delta T - Tc) - 1/\Delta T < f_m < f(\Delta T + Tc) + 1/\Delta T$$

$Ea_{sup} = (1+fTc)/\Delta T = -Ea_{inf}$ $Er_{sup} = Ea/_{fmin} = -Er_{inf}$

Pulse count during time interval: I-HW, T-int

•**N bit counter connected to a microprocessor port, read during interrupt routine at the beginning and at the end of ΔT. Steps of the interrupt service routine:**

- interrupt disable and context saving
- count = present counter value – previous value;
- freq = count/ ΔT;
- previous value = present counter value;
- context restore and interrupt enable



•**Constraints**

$f_{max}\,\Delta T < 2^N$ (f=frequency of pulses); $f_P = 1/\Delta T$; $f_{min} > 1/\Delta T$

•**Errors**

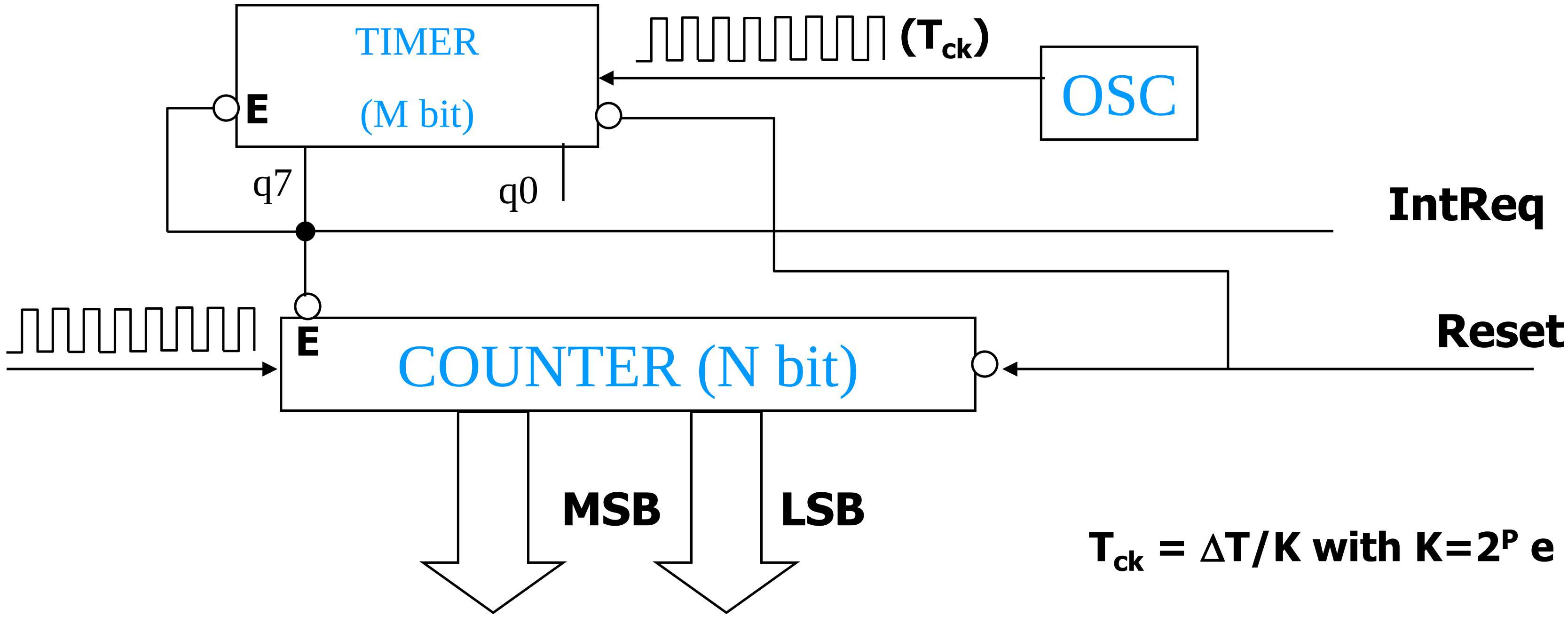
Quantization error on pulses identification [-1 … 1]

Time error on ΔT estimation $[-T_{Imax} \dots +T_{Imax}]$ T_{Imax}= max. interrupt latency

$$f(\Delta T - T_{Imax}) - 1/\Delta T < f_m < f(\Delta T + T_{Imax}) + 1/\Delta T$$

$Ea_{sup} = (1 + f\,T_{Imax})/\Delta T = -Ea_{inf}$ $Er_{sup} = Ea/_{fmin} = -Er_{inf}$

Pulse count during time interval: I-HW, T-HW



T_{ck} = ΔT/K with K=2^P e P<M

•Constraints

f_{max} ΔT < 2^N (f= pulse frequency); f_P = 1/(ΔT+ T_{Imax}); f_{min} > 1/ΔT

•Errors

Quantization error on pulses identification [-1 … 1] Time error on ΔT estimation [-T_{ck} … 0]

$$f(\Delta T - T_{ck}) - 1/\Delta T < f_m < f\Delta T + 1/\Delta T$$

Considerations on pulse count during time interval

- Typical mode: interrupt for time, pulse count with counter
- for a good resolution $\Delta T >> $ pulse period
- Constant quantization error on frequency measurement ($1/\Delta T$)
- ΔT has to be chosen considering the minimum frequency to be measured and the relative error

TRADEOFF ON ERRORS

- $\Uparrow \Delta T$, $\Downarrow$ errors and measurable frequencies $\Downarrow$, $\Uparrow$ measureent delay
- $\Downarrow T_{ck}$, $\Downarrow$ errors, $\Uparrow$ timer bits
- $\Downarrow T_{latency\ int.}$ $\Downarrow$ errors, $\Uparrow$ process management complexity
- $\Downarrow$ Tc $\Downarrow$ errors and measurable frequencies $\Uparrow$, $\Uparrow$ CPU load

Measure of the interval between two consecutive pulses

- **N bit free-running counter**
- **Time pulses with duration T_{ck}**
- **interrupt to acquire the I and the II pulse**
- **interrupt service routine:**

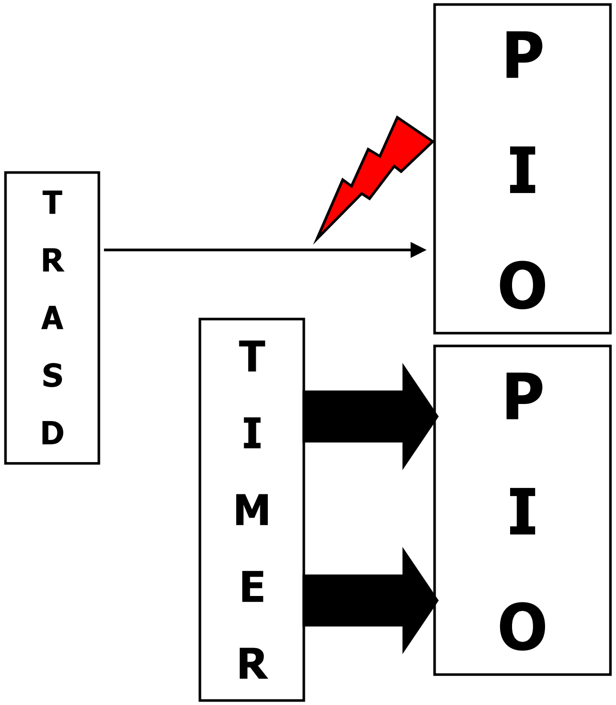
  context_saving

  temp=read counter;

  freq=1/ (temp-prec);

  prec = temp;

  Context_restore



- **Constraints**

$T_{ck}\ 2^N > 1/f_{min}$; f_P = f to measure (minimum f_{min}); $f_{max} < 1/T_{ck}$

- **Errors**

Time quantization error [$-T_{ck}$ … T_{ck}]

Time error on pulses identification [$-T_{Imax}$ … $+T_{Imax}$] T_{Imax}= max. interrupt latency

$$f/1+f(T_{ck}+T_{Imax}) < f_m < f/1-f(T_{ck} + T_{Imax})$$

Considerations on the measurement of the interval between two pulses

- **typical mode: that showed**
- **null frequency measurement is critical**
- **variable acquisition period (f_p) and measurement production**
- **minimum resolution when frequency is high, max. when it is low**
- **measur. interval = N pulses, with N set depending on the chosen resolution**
- **ΔT chosen on the basis of the minimum frequency**

<u>COMPROMESSI SU ERRORI</u>

$\Downarrow T_{ck}$, $\Downarrow$ errors, $\Uparrow f_{max}$ detectable, $\Uparrow$ timer bits

$\Downarrow T_{latency\ int.}$ $\Downarrow$ errors, $\Uparrow f_{max}$ detectable, $\Uparrow$ complexity

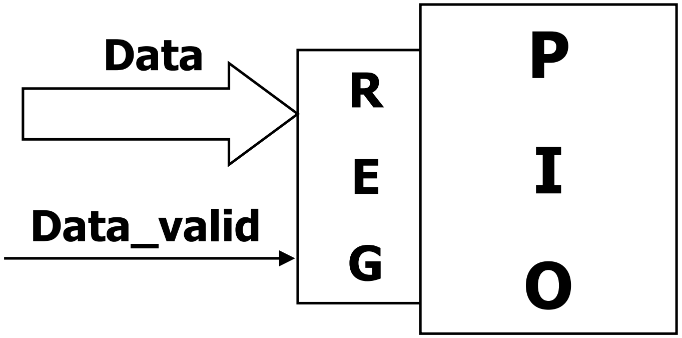
EXAM TEXT (May 2007)

The frequency of a pulse train must be measured in the variable range of [5 KHz … 500 KHz]. The measurement must be provided every ΔT sec, allowing a small tolerance in the periodicity due to the real availability of the CPU:

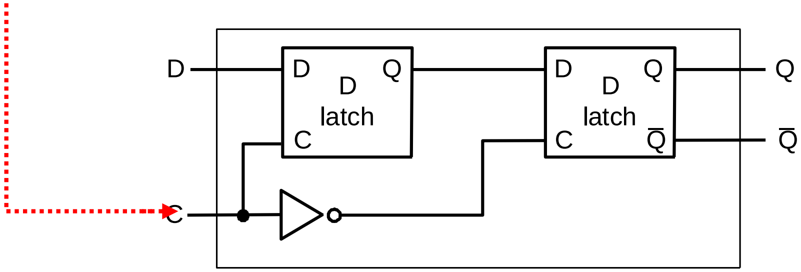
- **Describe how the chosen technique works and choose suitable devices so as the uncertainty range affecting the measure in the worst case is [-0.15% … + 0.1%];**
- **Verify that the constraints required by the chosen technique are satisfied;**
- **Determine the maximum and minimum frequencies really measurable with the imposed constraints;**
- **Calculate the uncertainty range in the frequency measurement if the quantization error on the ΔT interval estimation is supposed to be equal to +/- "quantum".**

Pulse acquisition for synchronising

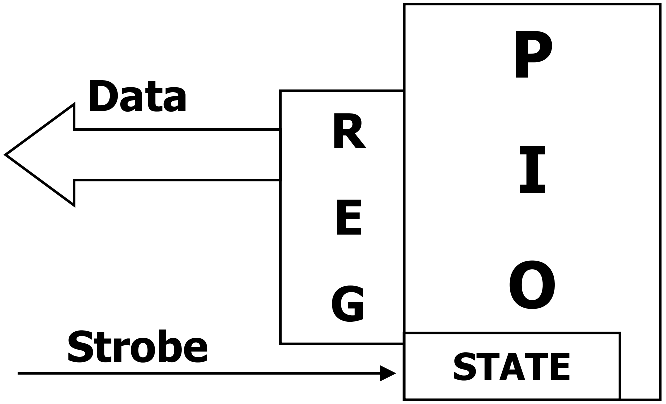
• events corresponding to a valid information



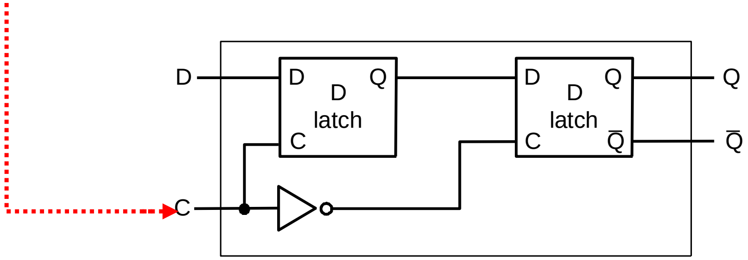
• Data_valid; in A, (PORTA_PIO)



• peripheral ready to receive

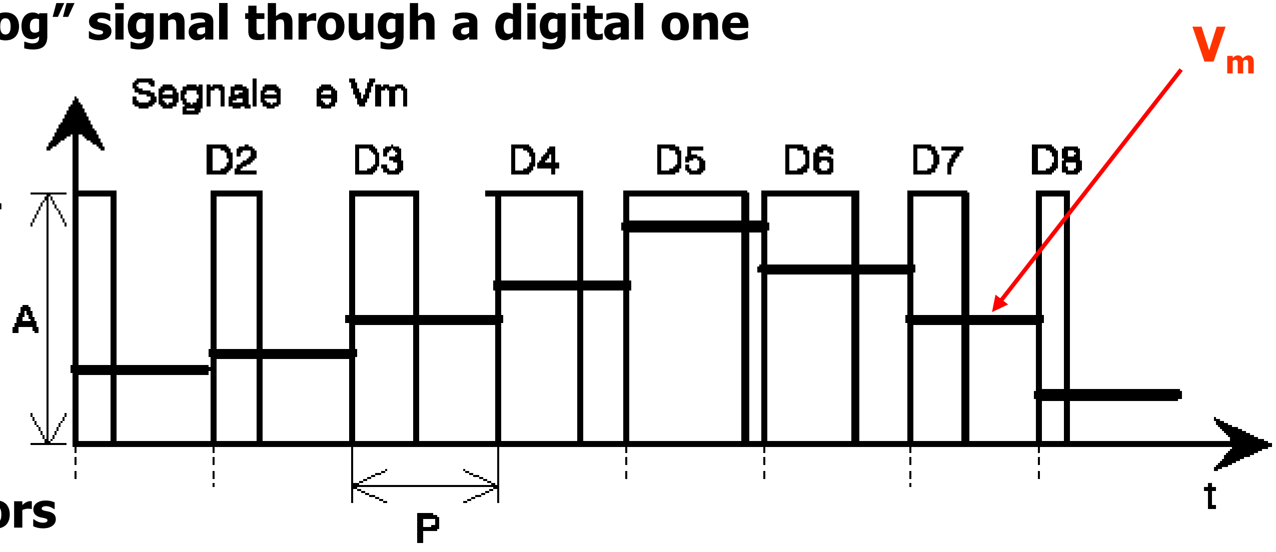


• Strobe from peripheral; out A, (PORTA_PIO)



PWM PULSE EMISSION

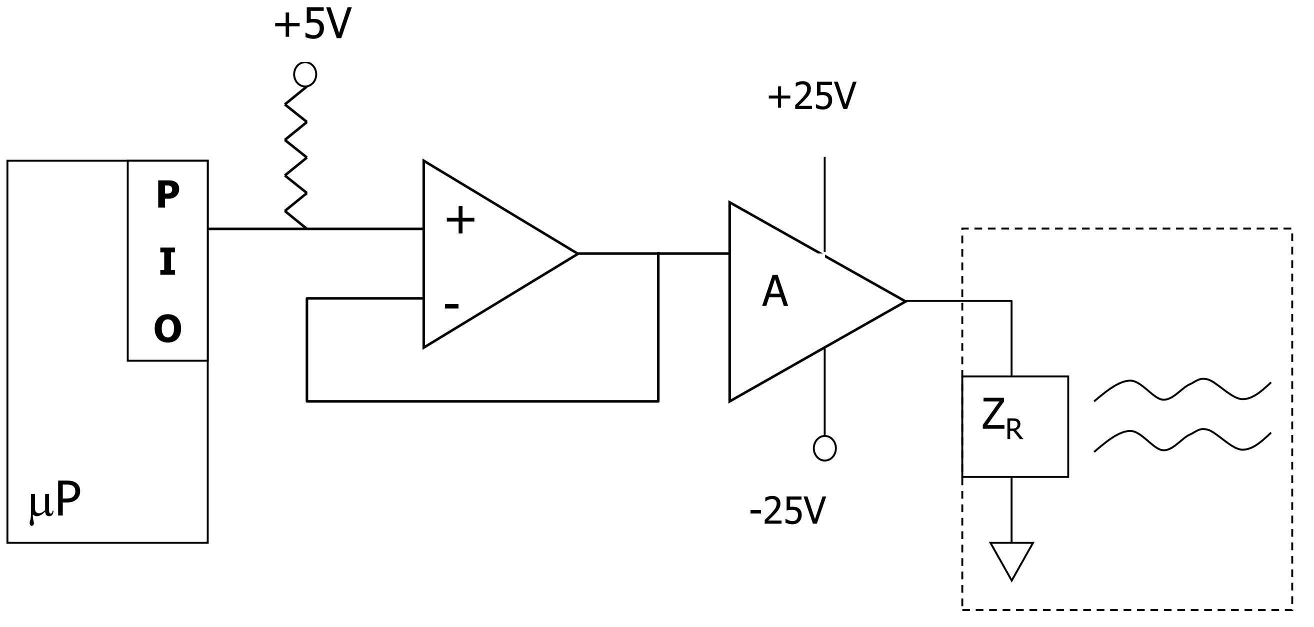
- **Control of processes with a slow behavior that are sensible to the average value of the control signal applied (or of the manipulated variable). For example the environmental temperature**
- **A pulse with A amplitude and D(t) duration where 0<D(t)<P (period)**
- **Average value (V_m)= A*D/P or A*DC(t)** **DC = duty cycle (D/P)**
- **Relative quantization error on V_m is P_{ck} (clock granularity) / P**
- **We achieve an "analog" signal through a digital one**
- **If P is in the order of 100 msec => sw management**



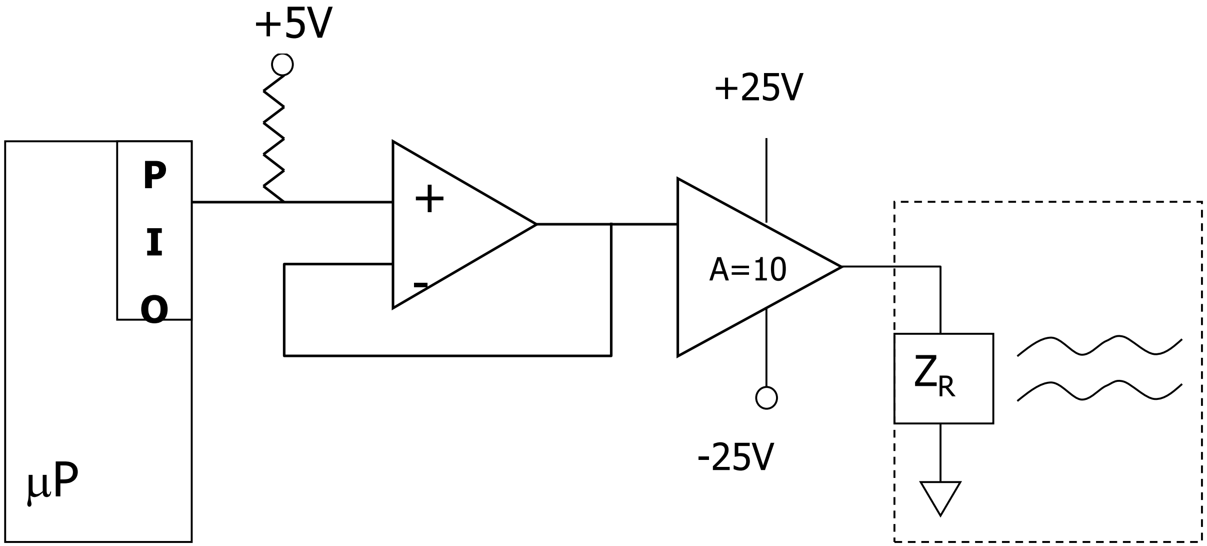
- **Also to drive dc motors**

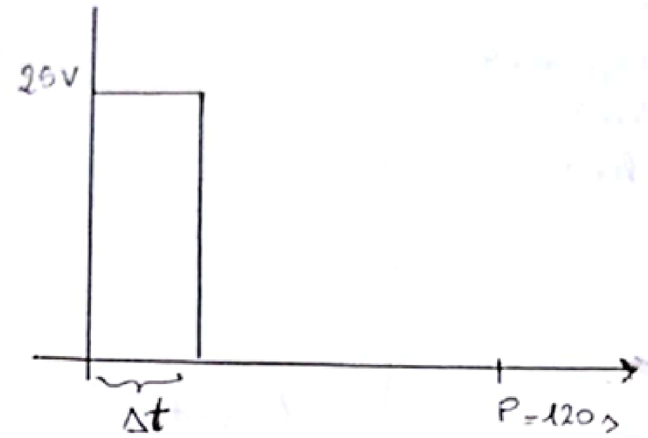
EXAM TEXT (July 2006)

- **Design a system able to heat up an incubator at 20 °C for 2 minutes, using a typical microprocessor equipped with a parallel PIO port connected to a buffer and to a amplification stage with gain A=10**
- **Suppose that the ratio between environmental temperature and the average applied power (thermal efficiency) is 80 °C/W and that the heater can be electrically modeled as a 1 KΩ resistance.**
- **Suppose to heat for 10 minutes at 30°, for 5 minutes at 20°and for 5 minutes at 10°: what is the final temperature reached?**



EXAM TEXT (July 2006)





$$W_{EL} = \frac{V^2}{R} = 0{,}625\ W$$

$$Temp = 0{,}625 \cdot 80 = 50\ °\mathbf{C}$$

$$W_{EL_{20°C}} = \frac{20}{80} = 0.25W$$

$$V_{20°C} = \sqrt{1000 * 0{,}25} = \sqrt{250} = 15{,}8$$

$$\frac{\Delta T}{120} = \frac{15{,}8}{25} \rightarrow \Delta T = 76{,}8s$$

EXAM TEXT (July 2006)

$$\frac{T1 \cdot \Delta T_1 + T2 \cdot \Delta T_2 + T3 \cdot \Delta T_3}{\Delta T_1 + \Delta T_2 + \Delta T_3} = \frac{30 \cdot 10 + 20 \cdot 5 + 10 \cdot 5}{20} = 22{,}5°C$$

$$T_1 = 30 \rightarrow W_{EL1} = \frac{30}{80} \rightarrow V_1 = \sqrt{1000 \cdot \frac{30}{80}} = 19{,}36\,V$$

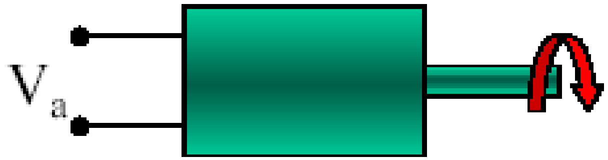
$$T_2 = 20 \rightarrow W_{EL1} = \frac{20}{80} \rightarrow V_2 = 15{,}8\,V$$

$$D_{1,\mathbf{2},\mathbf{3}} = \frac{V_{1,\mathbf{2},\mathbf{3}}}{V_{max}} \Delta t_{1,\mathbf{2},\mathbf{3}}$$

$$T_3 = 10 \rightarrow W_{EL1} = \frac{10}{80} = 0{,}125 \rightarrow V_1 = \sqrt{125} = 11{,}2\,V$$

PWM PULSE EMISSION: DC MOTORS

- Transformation of electric energy in mechanic (rotation vel. and torque)
- If the voltage supply is fixed a dc motor absorbs a current proportional to the exerted torque (thus it depends on the applied load)
- When the motor is in *stale* it absorbs much more current than in the usual case so to avoid damages is important to define two "limit" magnitudes
- **Stalemate current**: max current absorbed by a motor at the nominal voltage, during the stalemate phase (from hundreds of mA to few A).
- **Stalemate torque**: the torque provided when the driveshaft is blocked (at the startup or due to an external load), with a nominal voltage and maximum current (100 Nm)



θ = angular position

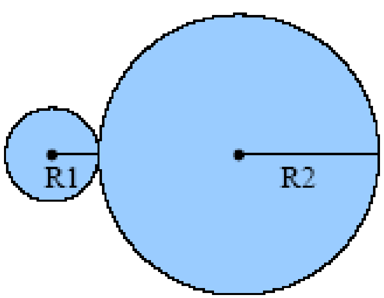
ω = velocity

C_m = torque

PWM PULSE EMISSION: DC MOTORS

- Sometimes the required velocity is less than that the motor can reach
- Suitable gears can be used (adaptors) that diminish ω by a suitable constant factor. The ensemble motor + adaptor can be defined as *motoadaptor*
- The adaptors can be used individually or coupled to regulate the overall angular velocity

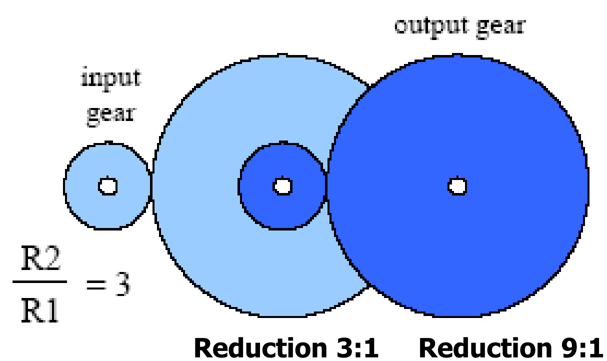
$$\omega_2 R_2 = \omega_1 R_1$$



$$\frac{\vartheta_1}{\vartheta_2} = \frac{R_2}{R_1}$$

$$\frac{\omega_1}{\omega_2} = \frac{R_2}{R_1}$$

Single adaptor



$$\frac{\vartheta_1}{\vartheta_2} = \left(\frac{R_2}{R_1}\right)^2$$

$$\frac{\omega_1}{\omega_2} = \left(\frac{R_2}{R_1}\right)^2$$

Adaptor series

PWM PULSE EMISSION: DC MOTORS

Let’s consider the mechanical power P conservation, $\Gamma \, \omega = \Gamma' \, \omega'$

Thus when the angular velocity drops down due to the effect of an adaptor the exerted torque correspondingly augments

In other words, if $\omega = n\omega'$ then $\Gamma' = n\Gamma$

Finally, since the mechanical behavior of a dc motor can be described by

$$\Gamma = \Gamma_0 + \gamma\omega + J\dot{\omega}$$

Γ_0 effective torque, γ friction and J inertia

$$\Gamma' - n\Gamma_0 = \left(J_R + n^2 J\right)\dot{\omega}' + \left(\gamma_R + n^2\gamma\right)\omega'$$

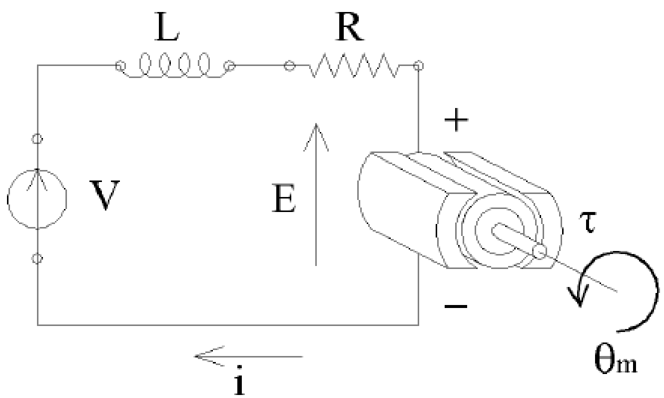
Γ' new torque, γ_R friction and J_R inertia of the adaptor

Significant increase of inertia and friction

Instabilities and hysteretical behavior due to mechanical play

The motoadaptors can be effective where high precision is not required

PWM PULSE EMISSION: DC MOTORS



Model of the motor without load

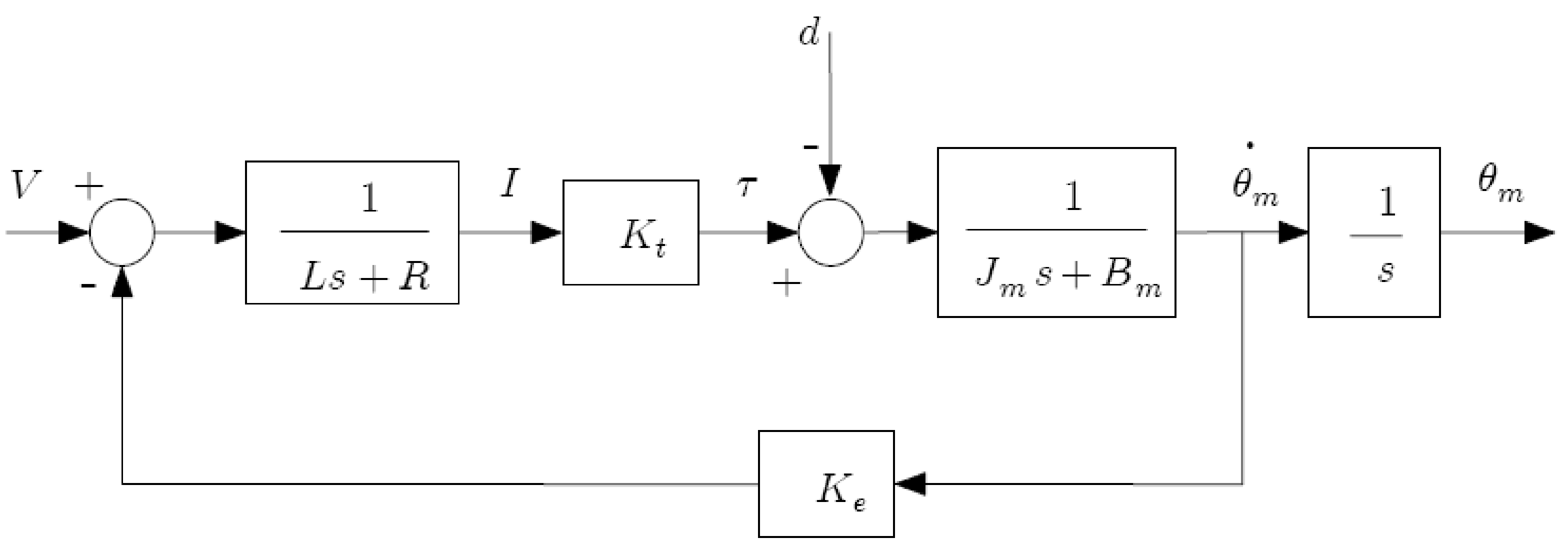
$$E(t) = K_e \cdot \dot{\theta}_m(t) = K_e \cdot \omega_m(t)$$

$$V(t) = E(t) + R \cdot I(t) + L \cdot \frac{dI(t)}{dt}$$

Ke constant for the velocity

R armature resistance

L armature inductance



PWM PULSE EMISSION: DC MOTORS

Model of the mechanical part:

$$\tau = J_m \cdot \ddot{\theta}_m + B_m \cdot \dot{\theta}_m + d + \tau_c$$

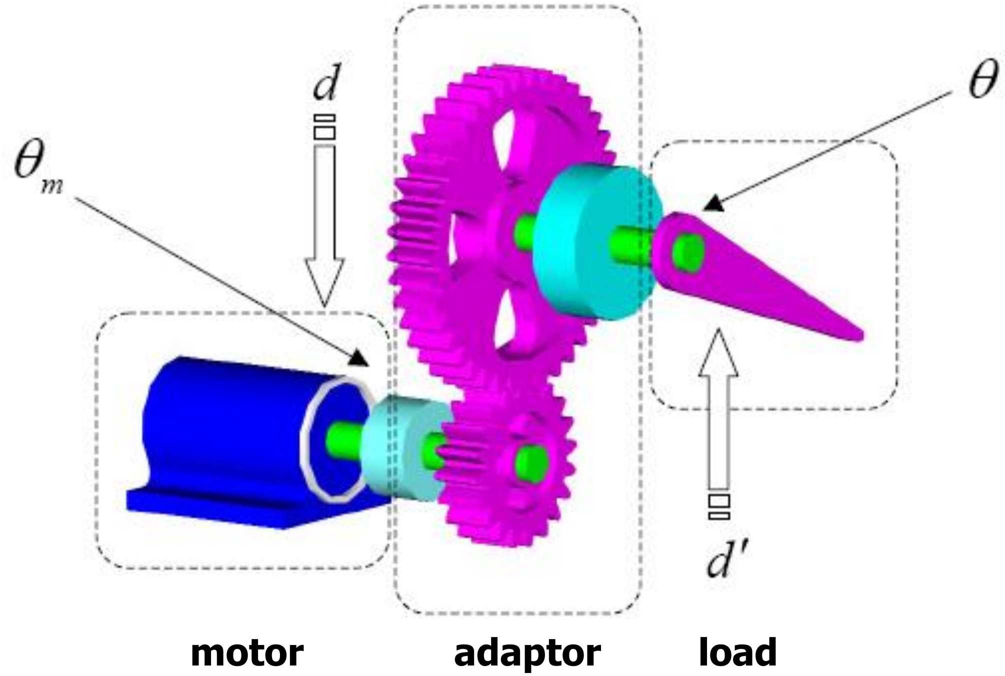
Similar approach to describe the load components:

$$\tau' = J_c \cdot \ddot{\theta} + B_c \cdot \dot{\theta} + d'$$

d = not manipulable inputs, frictions, phenomena without models

τ_c = effective torque applied to load plus adaptor

τ' = load torque

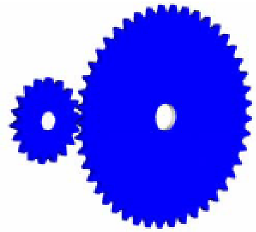


Model of the motor with load and adaptor

PWM PULSE EMISSION: DC MOTORS

Equations of the motor with load and adaptors

θ_m



$$\theta = \frac{1}{n} \cdot \theta_m$$

Reduction ratio $\frac{1}{n}$

$$\theta = \frac{\theta_m}{n} \Rightarrow \dot{\theta} = \frac{\dot{\theta}_m}{n} \Rightarrow \ddot{\theta} = \frac{\ddot{\theta}_m}{n}$$

we impose the conservation of the mechanical power

$$\xi \cdot \tau_c \cdot \dot{\theta}_m = \tau' \cdot \dot{\theta}$$

$$\tau_c = \frac{\tau'}{n \cdot \xi}$$

this brings to

$$\tau_c = J_c \cdot \frac{\ddot{\theta}_m}{n^2 \cdot \xi} + B_c \cdot \frac{\dot{\theta}_m}{n^2 \cdot \xi} + \frac{d'}{n \cdot \xi}$$

The adaptor reduces the effect of disturbs on load

by substituting in the general motor expression …

$$\tau = (J_m + \frac{J_c}{n^2 \cdot \xi}) \cdot \ddot{\theta}_m + (B_m + \frac{B_c}{n^2 \cdot \xi}) \cdot \dot{\theta}_m + \frac{d'}{n \cdot \xi} + d$$

that is

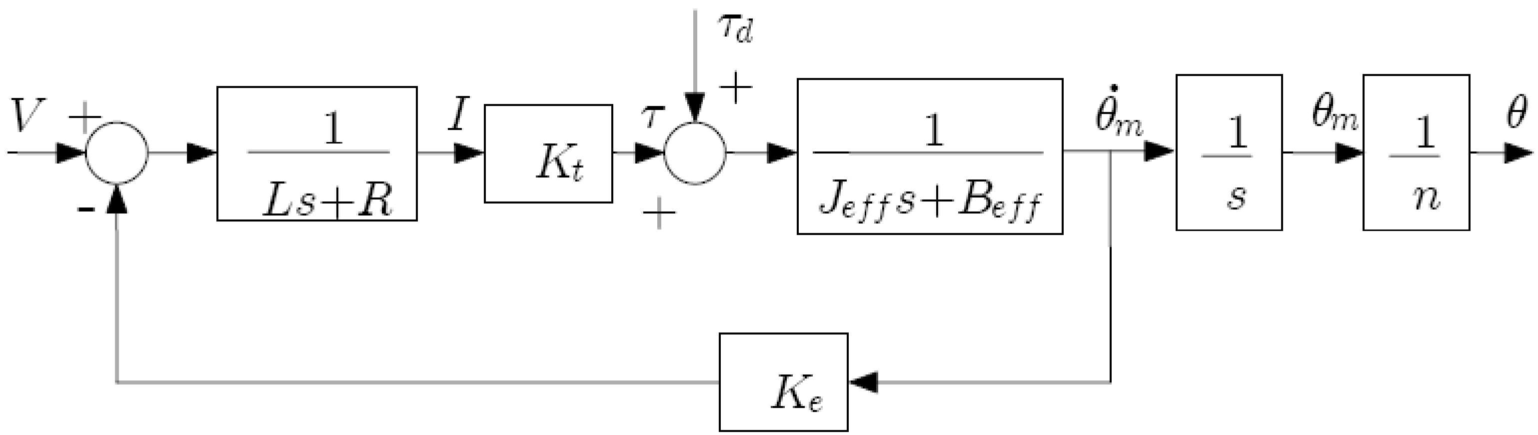
$$\boxed{\tau = J_{eff} \cdot \ddot{\theta}_m + B_{eff} \cdot \dot{\theta}_m - \tau_d}$$

where

$$-\tau_d = \frac{d'}{n \cdot \xi} + d$$

PWM PULSE EMISSION: DC MOTORS

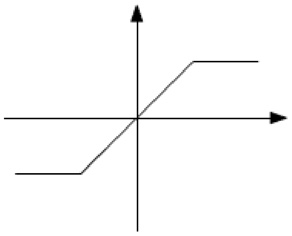
Model of the motor with load and adaptor



PWM PULSE EMISSION: DC MOTORS

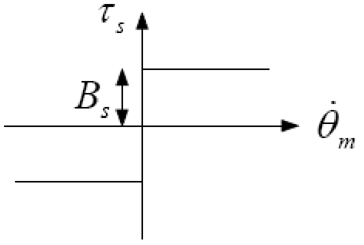
Saturation

(for example due to current limitation)



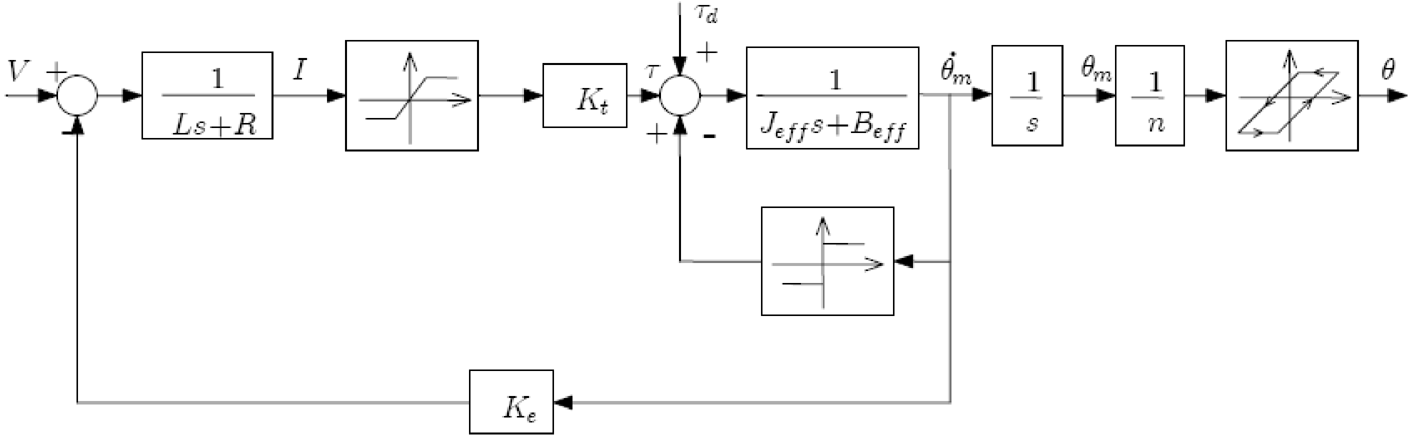
Dry friction

$$\tau_s = B_s sign(\dot{\theta}_m)$$



Possible non linearities

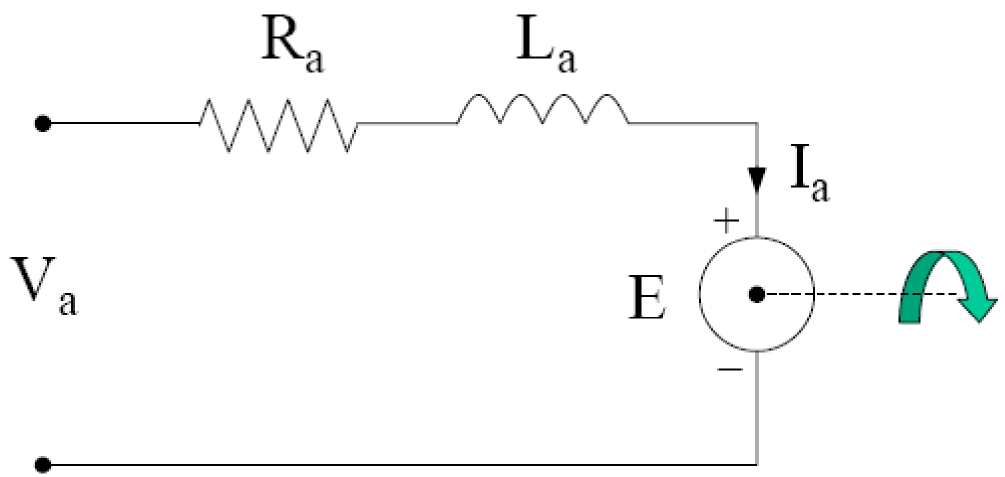
Adaptor hysteresis (due to possible mechanical plays)



PWM PULSE EMISSION: DC MOTORS

DC motor, simplified equivalent electric circuit:

- V_a = supply voltage
- I_a = supply current
- E = back electromotive force
- R_a = armature resistance
- L_a = armature inductance



Model:

- $V_a = E + R_a I_a$ — E back emf
- $E = k_e \omega$ — k_e constant for the velocity
- $P_a = V_a I_a = E I_a + R_a I_a^2$ — absorbed electric power = mechanical power + power lost in the armature

PWM PULSE EMISSION: DC MOTORS

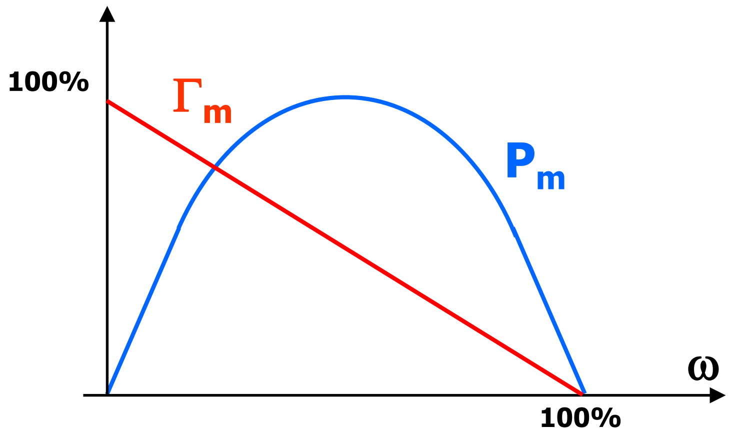
- DC motor, characteristic:

P_m = mechanical power = $E\,I_a = \Gamma_m \omega$

Γ_m = motor torque = $k_t I_a$

- considering that:

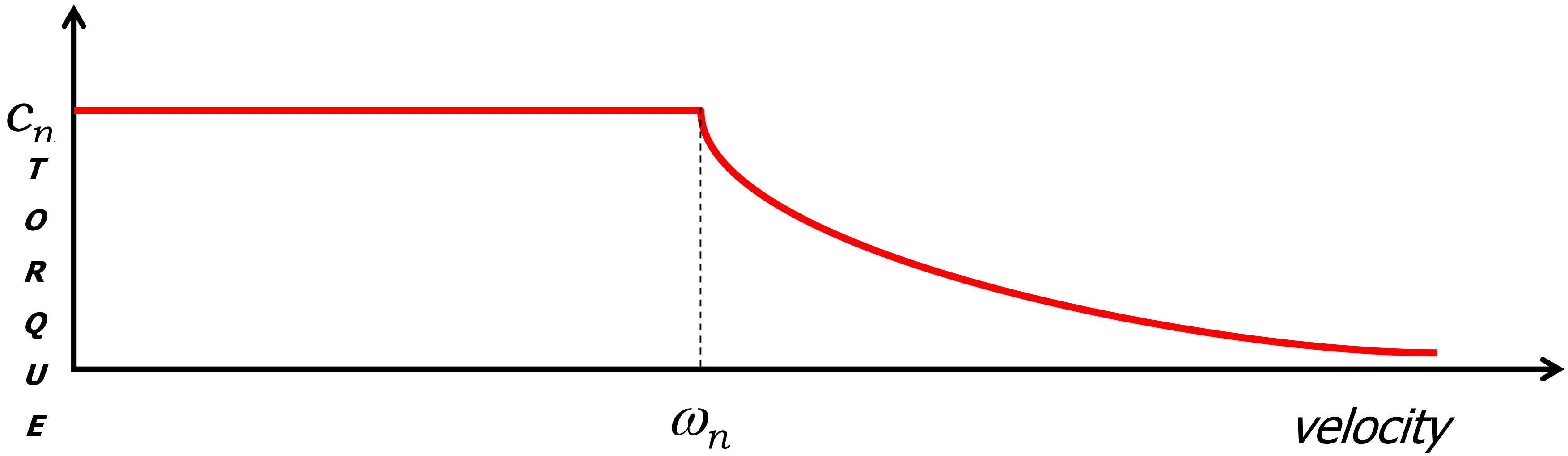
$I_a = (V_a - E)/R_a$ and $E = k_e \omega$ we obtain that $\Gamma_m = k_t(V_a - k_e\omega)/R_a$



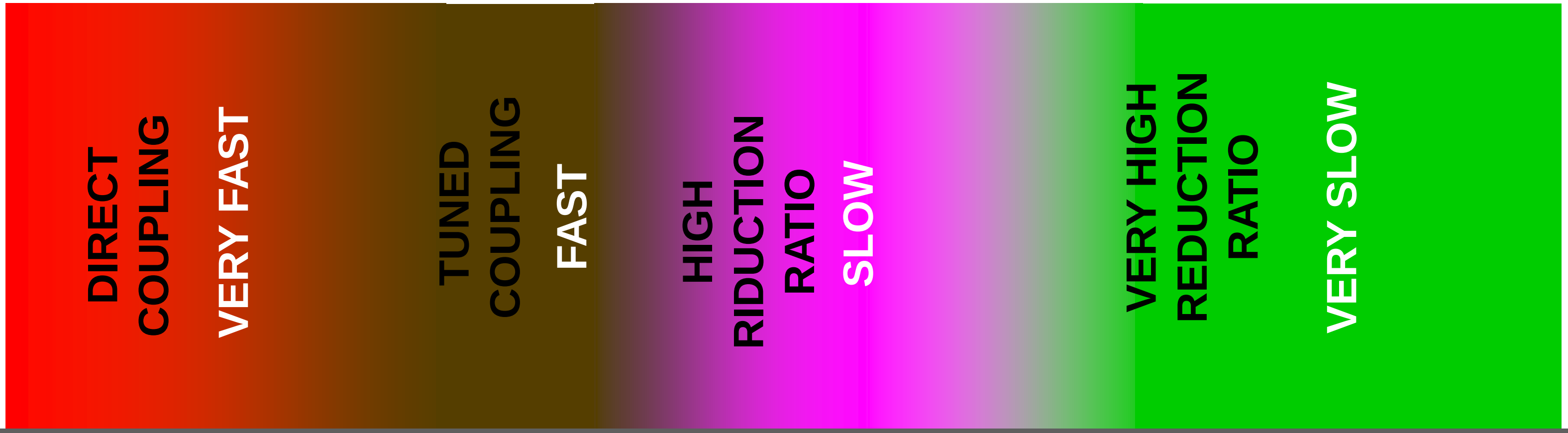
The torque diminishes when the angular velocity increases

At the beginning the mechanical power grows up with the angular velocity then diminishes with a typical "bell" behavior

TORQUE-VELOCITY REAL STATIC CHARACTERISTIC



ADAPTOR MOVEMENT CHARACTERISTICS



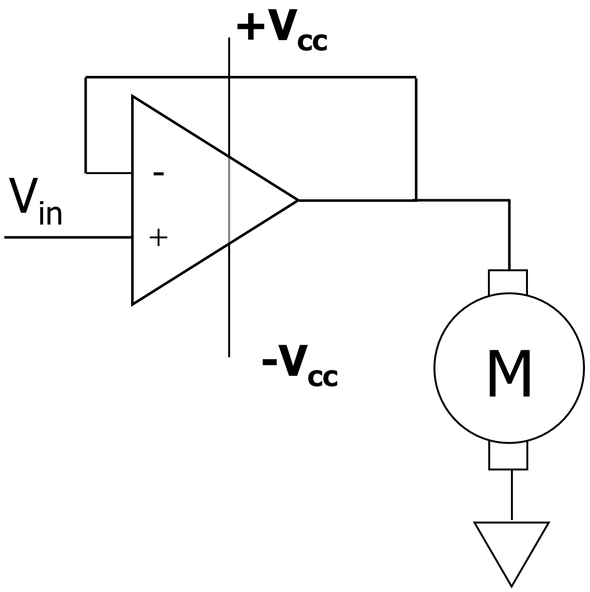
CHOICE OF THE REDUCTION RATIO

PWM PULSE EMISSION: DC MOTORS

•DC motors, power supply:

Motors absorb a lot of current (0.5 : 1) A so requiring specific driving circuits (power drivers)

Two possible kinds of driving: linear and on-off



To manage a bi-directional rotation a bipolar power supply V_{in} +/- is required

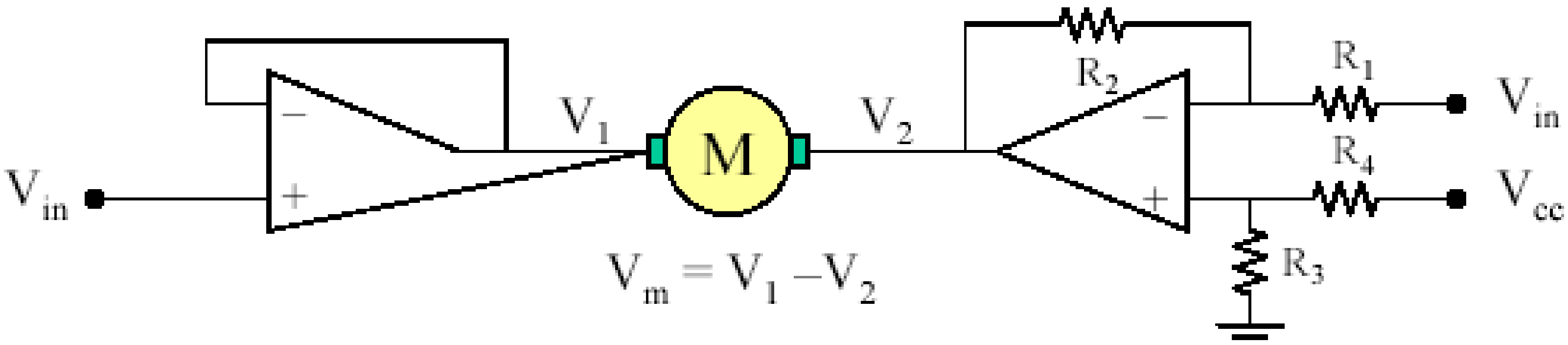
If the resistant torque (the ensemble that opposes to rotation inertia, frictions, disturbs, …) is negligible a good linearity can be obtained between V_{in} e ω

Should not a bipolar supply available, a “bridge” solution can be introduced

PWM PULSE EMISSION: DC MOTORS

- DC motors, linear driving (a bipolar voltage using a unipolar one):

Follower ($V_1 = V_{in}$) Differential ($V_2 = V_{CC} - V_{in}$)



$$V_m = V_1 - V_2$$

if $\frac{R_2}{R_1} = \frac{R_3}{R_4}$ $\quad V_2 = \frac{R_2}{R_1}(V_{cc} - V_{in})$

Thus if $R_1 = R_2 = R_3 = R_4$ we have

$$V_m = V_1 - V_2 = 2V_{in} - V_{cc}$$

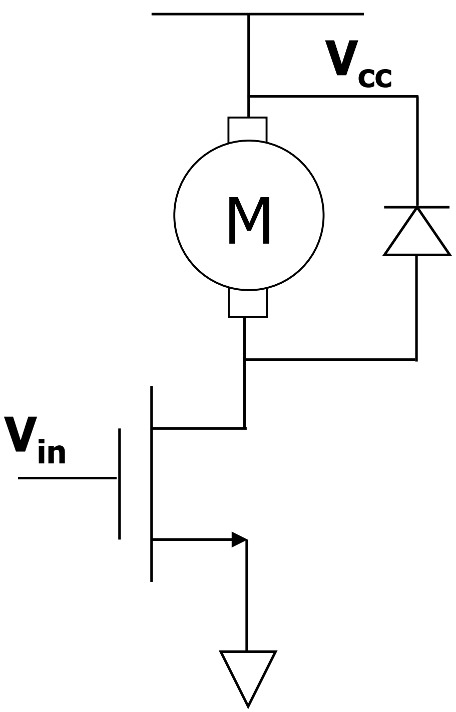
V_{in}	V_1	V_2	V_m
0	0	V_{cc}	$-V_{cc}$
$V_{cc}/2$	$V_{cc}/2$	$V_{cc}/2$	0
V_{cc}	V_{cc}	0	V_{cc}

PWM PULSE EMISSION: DC MOTORS

•DC motor, on off driving:

If the linear driving dissipates too much power, it can be used only in case of motors that absorb few Watts

To reduce the dissipated power MOS switching devices in on (saturation)/off stage



Dissipated power = V_{ce} I

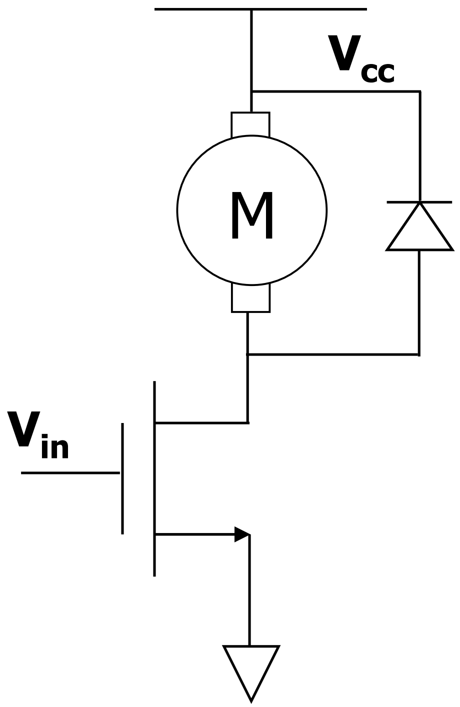
If MOS off, I=0 no dissipated power

If MOS in saturation, $V_{ce} \cong 0$ so dissipated power $\cong 0$

"Outflow" diodes for the current are required to avoid excessive voltages at the MOS drain (although it is fast and able to carry high currents)

Limit: the motor will run at the maximum velocity (only mono-direction) otherwise it will be stopped

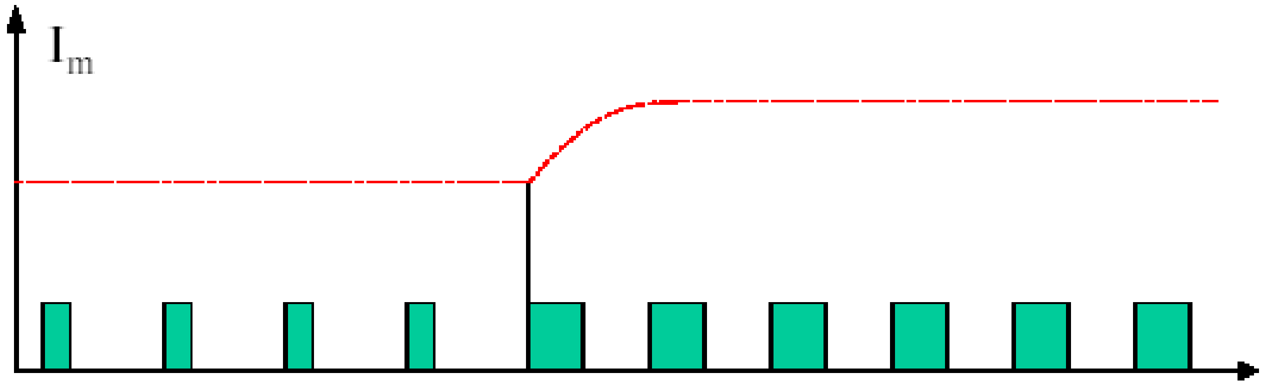
PWM PULSE EMISSION: DC MOTORS



- CC motor, on off driving:

For this reason it is useful to drive the motor through duty cycle variable pulses (PWM) so as to obtain a enough pulse frequency with an average supply voltage proportional to the pulses duty cycle.

However it is still not possible a bi-directional rotation

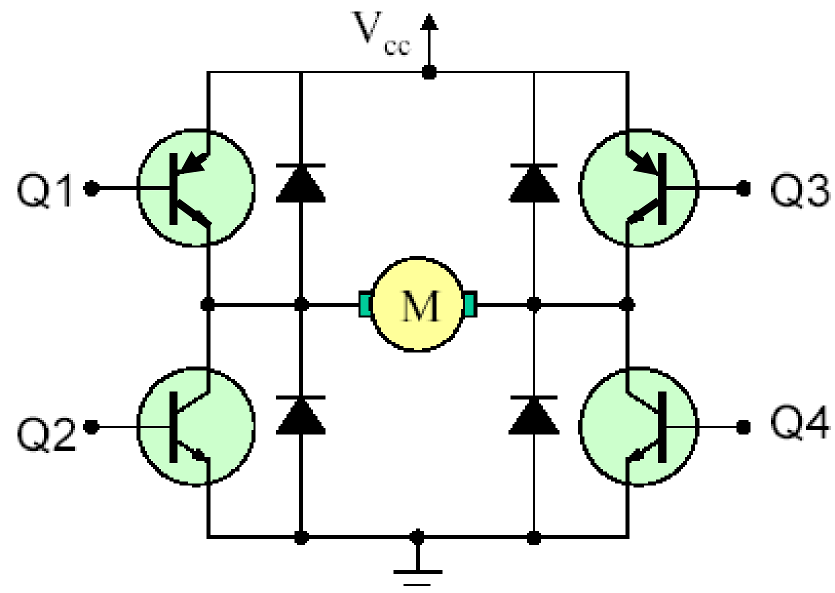


PWM PULSE EMISSION: DC MOTORS

•DC motor, H bridge driving (for small laboratory robots):

the "H-bridge" circuit allows to control the sense of rotation.

Q1 e Q4 on	Clockwise rotation
Q2 e Q3 on	Counterclockwise rot.
Q1 e Q2 on	Q1 e Q2 can fire
Q1 e Q3 on	Braking
All the transistors off	Uncontrolled slowing down



If two transistors on the same side are active (shoot-through) a big current passes between Vcc and ground. However, now the load is no more the motor (who absorbs the most of the current and so the voltage drop) as before thus the bridge could be damaged.

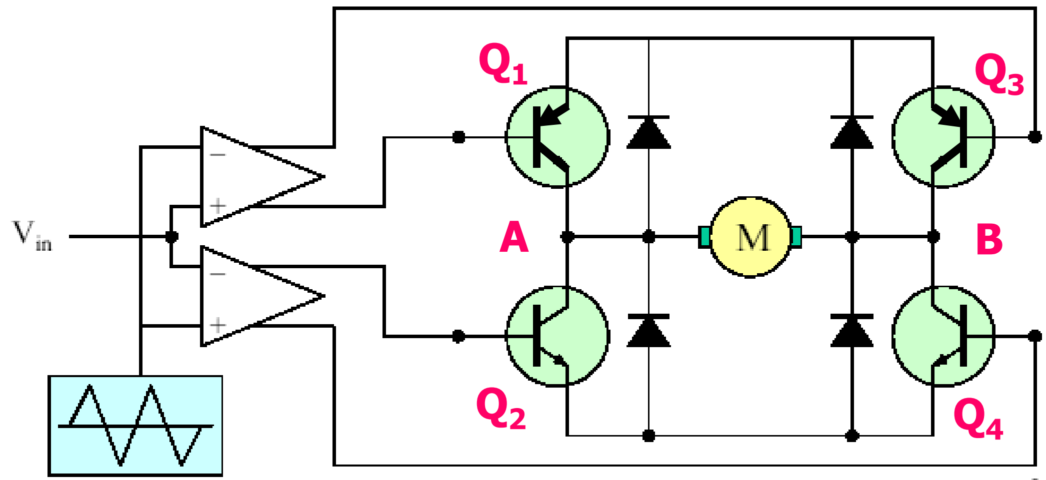
PWM PULSE EMISSION: DC MOTORS

•CC motor, H bridge driving:

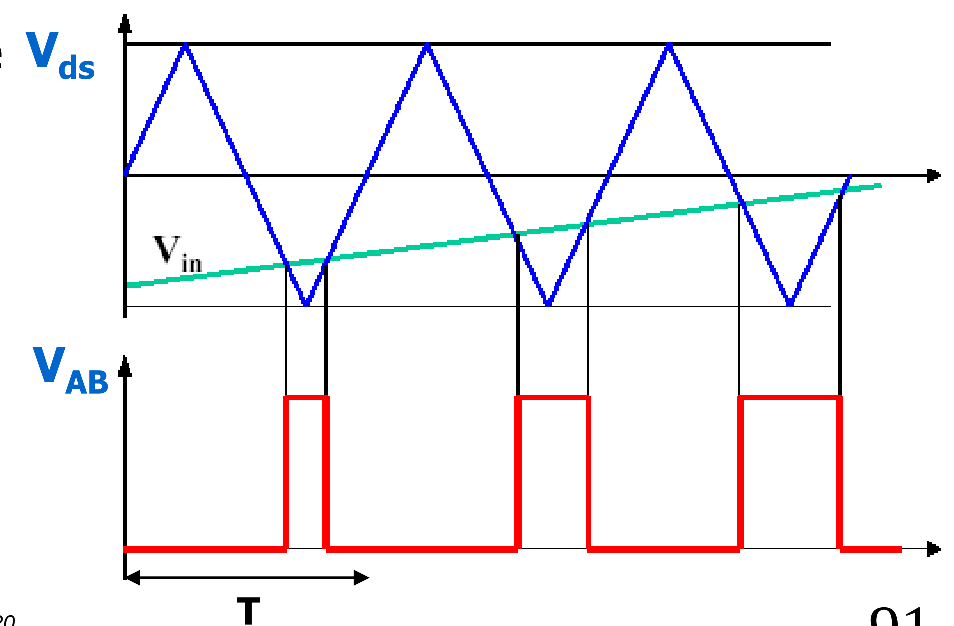
If we do not want to make the microprocessor serving as pulse emitter, a suitable electronic net could be used to activate transistors

$V_{in} > V_{ds}$ Q_3 e Q_2 active

$V_{in} < V_{ds}$ Q_1 e Q_4 active

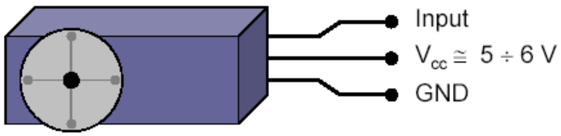


Within a period T of the input signal V_{ds} the motor is commanded to rotate for a certain time in a direction and for the remaining in the opposite direction; for the long-term time the resulting direction will be proportional to the sum of the different rotations in the different intervals: if V_{in} would be horizontal and exactly located in the medium of V_{ds}, the motor would be stopped (**Locked anti-phase PWM)**

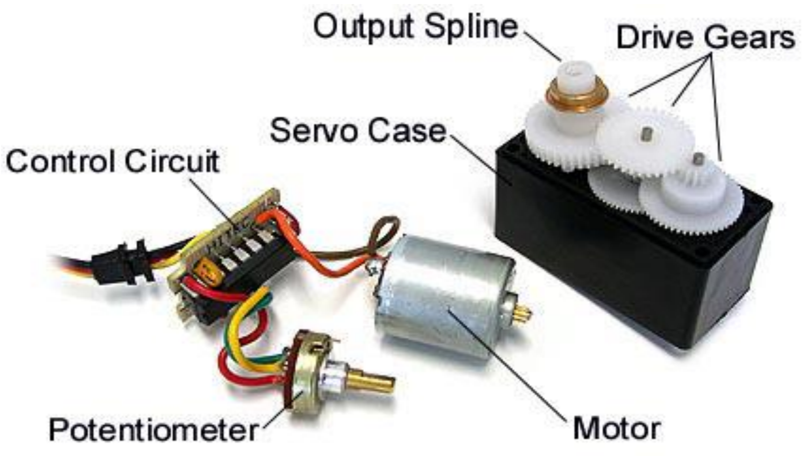


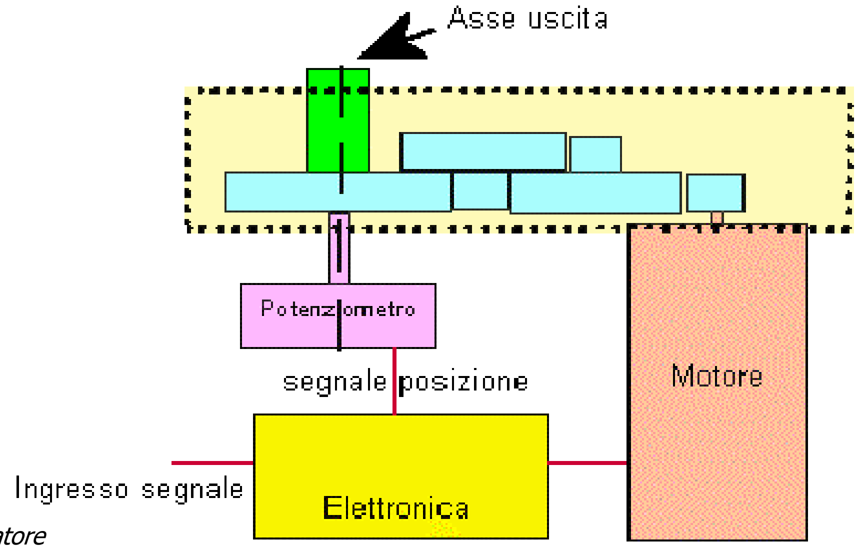
PWM PULSE EMISSION: SERVO MOTORS

They encapsulate dc motor, adaptors, control electronic e position control through a potentiometer for detecting the shaft position and rotation limit switch (to protect the movement area)



Characteristics: torque, rotation velocity, maximum rotation angle. Used in robotics applications





Figures: courtesy from Damino Salvatore

Orders of magnitude:

Rotation angle: ±90°, torque: 3-20 Kg*cm (but even 20 Nm), absorbed current: 1-10 mA on rest and ~1 A full load, rotation velocity: up to 9000 rpm, weight: ~150 gr.

PWM PULSE EMISSION: SERVOMOTORS

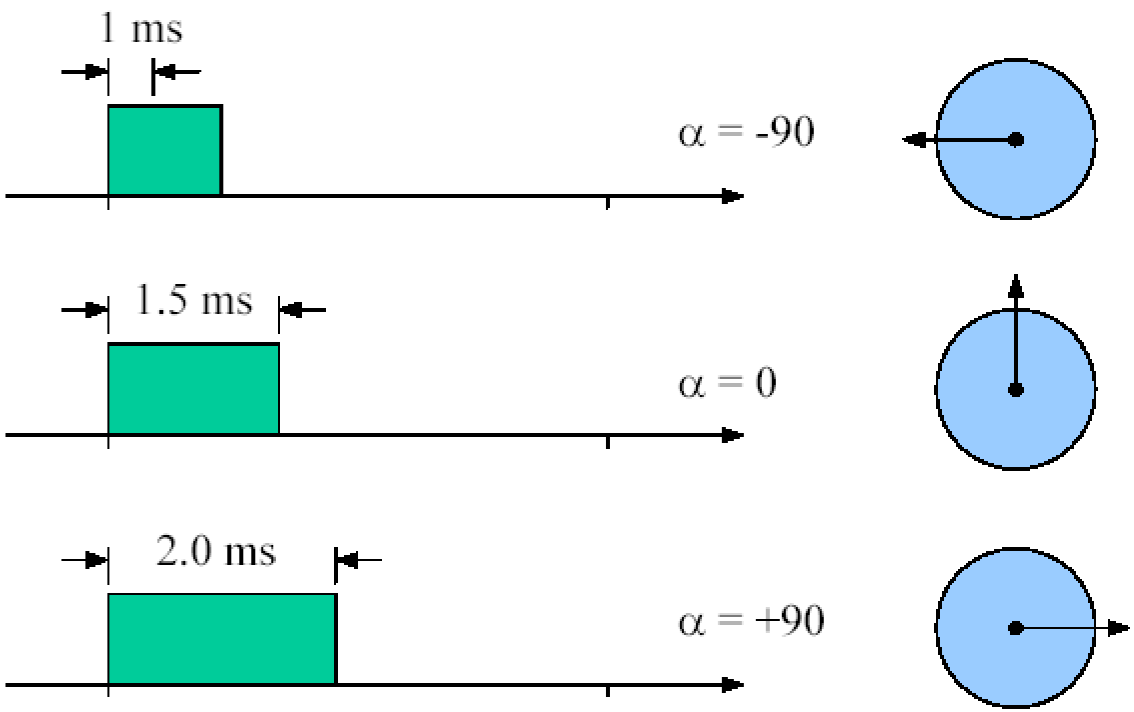
•Servomotors:

PCM pulses (Pulse Code Modulation) are used, and the duty cycle determines the shaft position with respect the previous one (the incremental amount of the rotation): for example, depending on the motor type, a certain pulse duration corresponds to -90° rotation, if the duration is twice the rotation is +90°, if it is half the rotation is 0°. If the servo does not frequently receive pulses it is abandoned without any control.

Usually the repetition period is 20-30 msec, whilst the pulse duration ranges from a 1 msec minimum up to 2 msec max.

The servo is stopped when a "central" pulse duration is provided or due to its natural lost of energy (frictions …)

Open loop position control



PWM PULSE EMISSION: SERVOMOTORS

- Servomotors:

The “response” of the servo depends mainly on the pulse duration more than on the duty cycle since these motors are designed to properly work into a enough wide pulse frequency range within which the behavior is quite similar (not a orthodox PWM).

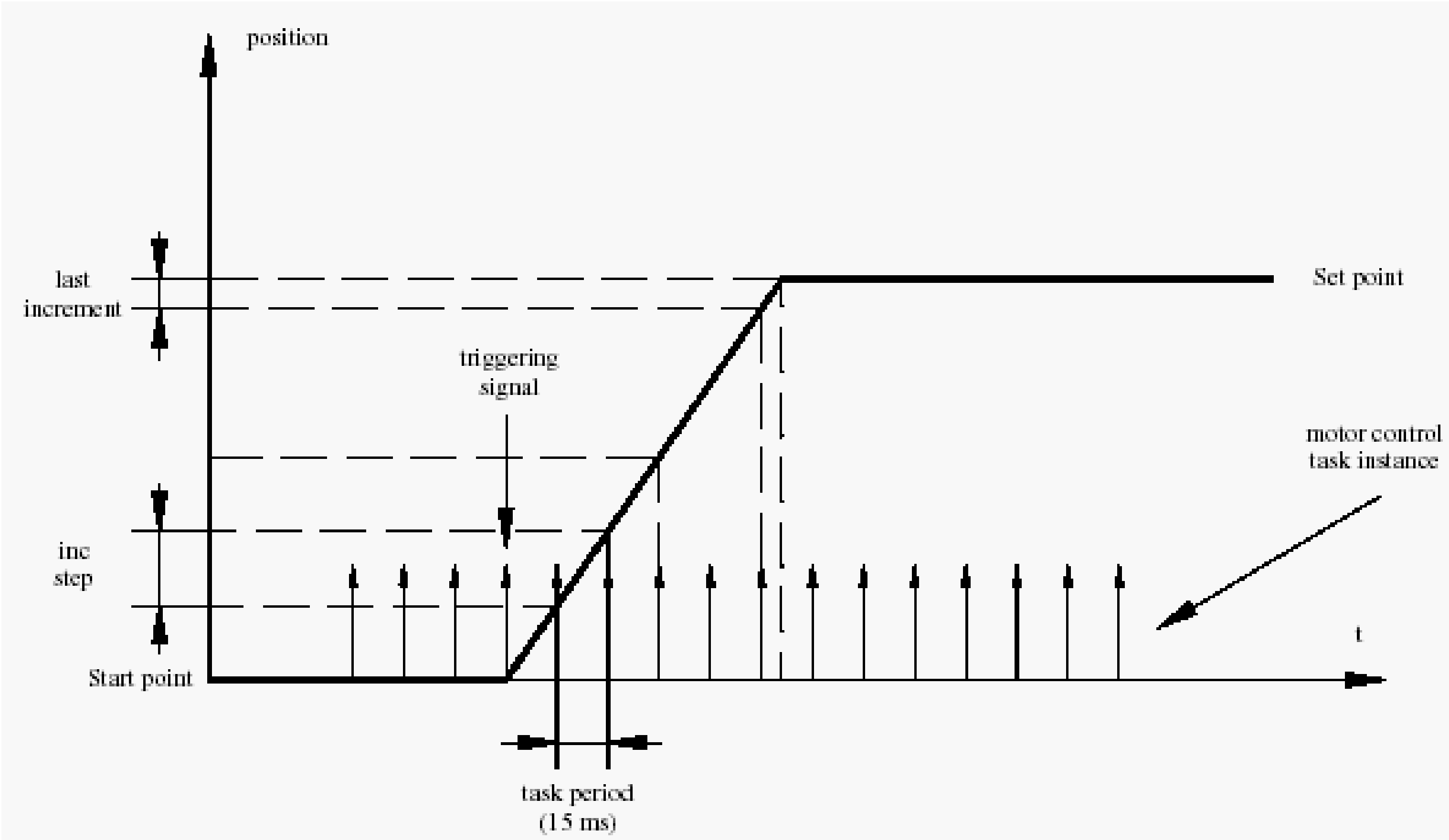
If for example a servo receives pulses with variable period among 0-70 msec, it may happen that with values close to ~10 msec the servo becomes a little unstable (vibrations), while when the period is > 40 msec, the torque diminishes. Thus a period close to 20 msec could be the right choice (moreover is easy ⇒ 50 Hz).

The servo are characterized by a suitable “working curve” to achieve a pre-determined motion dynamic.

The “curve” could consist on variable width steps (larger at the beginning, smaller when close to the target) up to reach the set point. It could be useful to control the movement velocity although this is quite difficult since practically the system moves always at maximum velocity.

How to generate the pulse?: a hybrid solution could envisage a μP triggering a timer so as it exits pulses with a duty cycle set again through the μP. It is not possible to completely devote a μP to pulse emission because the required times (sometimes < msec) ask for high frequency routines, with possible μP overload and inaccuracies.

PWM PULSE EMISSION: DC MOTORS



•Periodic task (15 msec) to set the rotation "step"

•At the beginning the step is wide, when the position is close to the target it diminishes

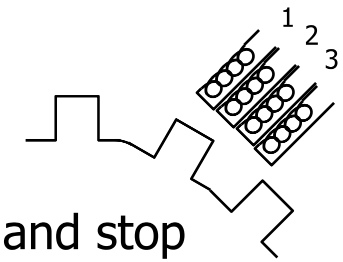
•The position of the rotating shaft can be evaluated time by time

PFM PULSE EMISSION: STEPPER MOTORS

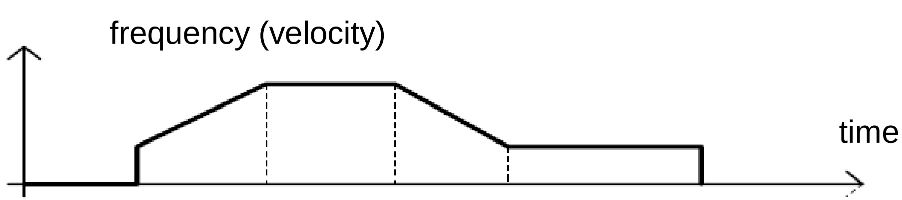
•Step motors: every received pulse correspond to a precise rotation step

•Low torque, good precision, low cost. Applications: computing systems peripherals, robotics, small applied loads

• Specs:

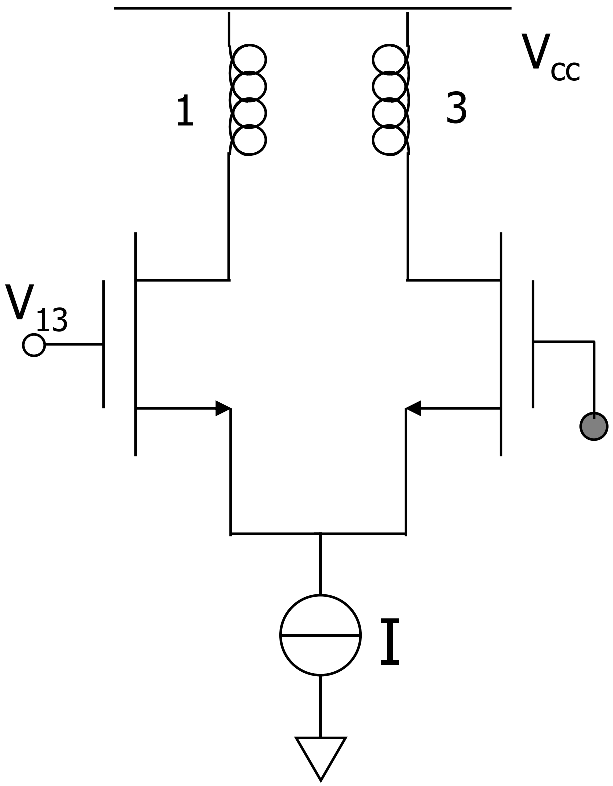


- •Pulse duration enough to drive electronics
- •Start frequency (minimum sequence necessary to start rotation) and stop (minimum frequency below which the motor is stopped)
- •Acceleration and deceleration ramp
- •Possible jitters that imply steps lost or vibrations



- •Dedicated hw or processors (for example PIC: programmable interface controller, Microchip Risc microcontroller, low costs)

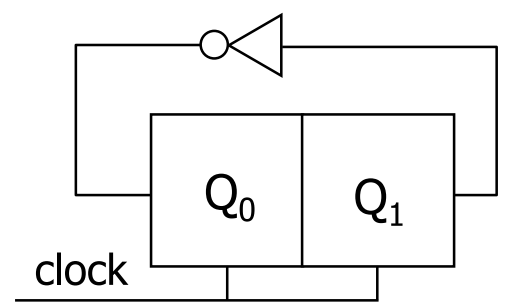
PFM PULSE EMISSION: STEPPER MOTORS



Two-phase sequence,

Most commonly used movement (good torque, not negligible power consumption and heat dissipation)

1	2	3	4	V_{13}	V_{24}
ON	ON	OFF	OFF	+	+
OFF	ON	ON	OFF	-	+
OFF	OFF	ON	ON	-	-
ON	OFF	OFF	ON	+	-
ON	ON	OFF	OFF	+	+



How to provide clock?

PFM PULSE EMISSION: STEPPER MOTORS

1	2	3	4
ON	OFF	OFF	OFF
OFF	ON	OFF	OFF
OFF	OFF	ON	OFF
OFF	OFF	OFF	ON
ON	OFF	OFF	OFF

Wave: less torque with respect two phase (1vs1.4)

Half step: the step number is twice (precision) but the torque is irregular and the power consumption not constant

1	2	3	4
ON	OFF	OFF	ON
ON	OFF	OFF	OFF
ON	ON	OFF	OFF
OFF	ON	OFF	OFF
OFF	ON	ON	OFF
OFF	OFF	ON	OFF

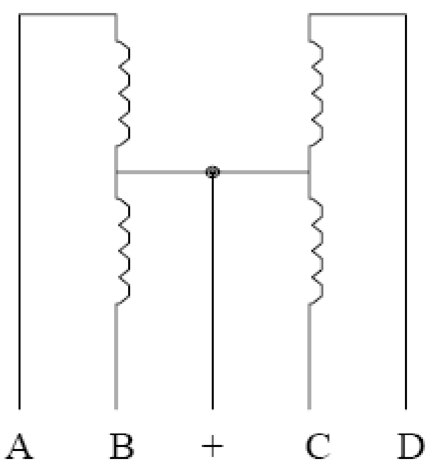


Fig 1. Five-wire stepper motor

matics

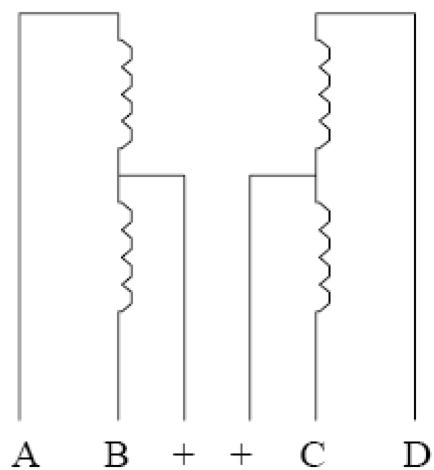


Fig 2. Six-wire stepper motor

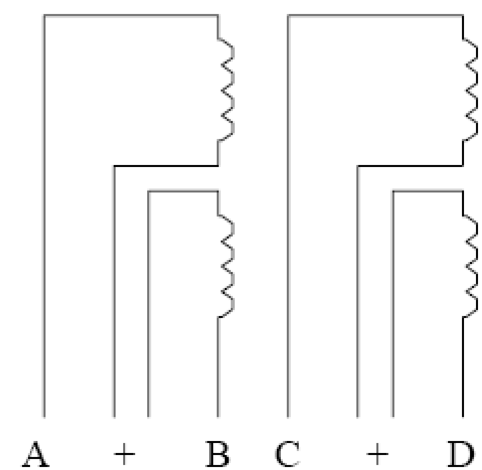


Fig 3. Eight-wire stepper motor

PFM PULSE EMISSION: STEPPER MOTORS

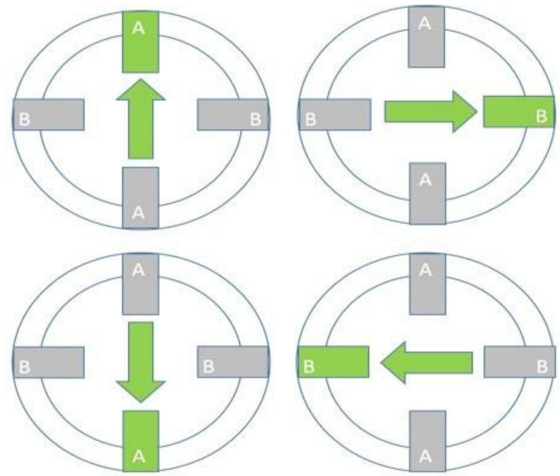


Fig 1 – One phase on – full step

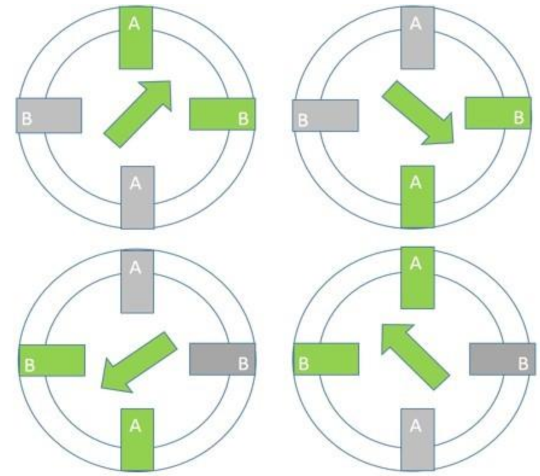


Fig2 – Two phase on – full step

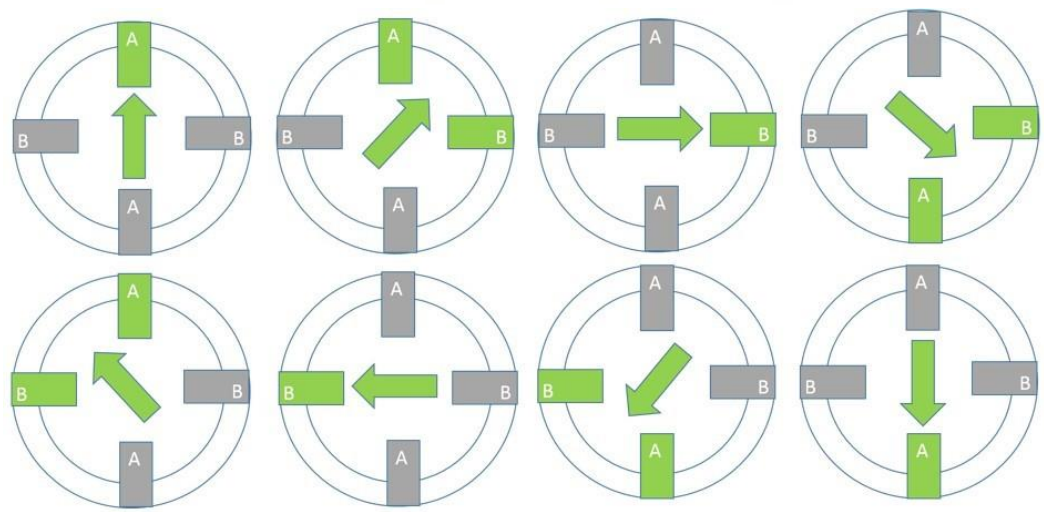
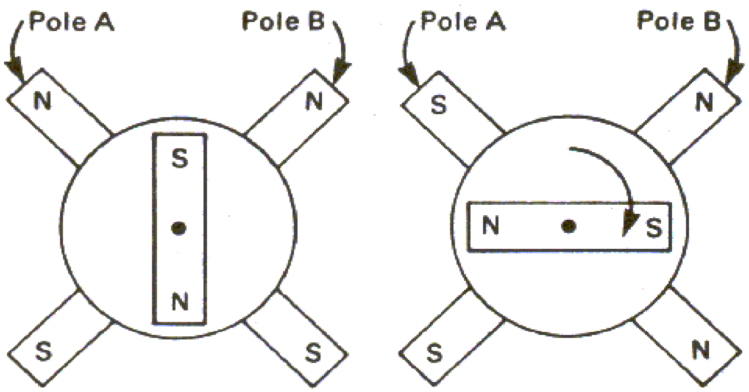
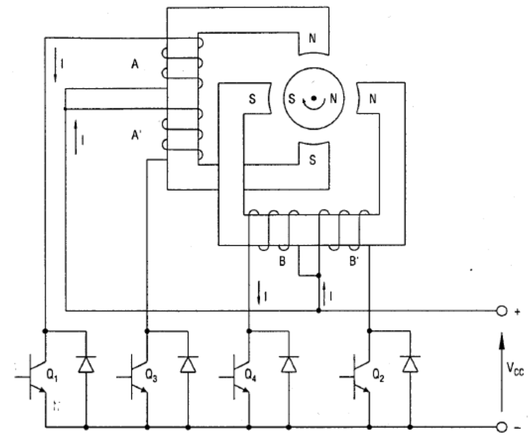
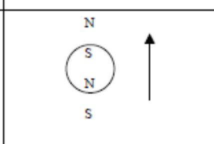
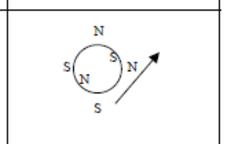
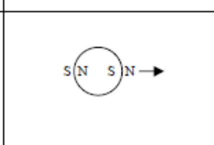
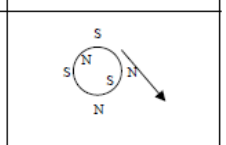
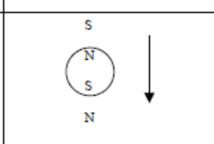
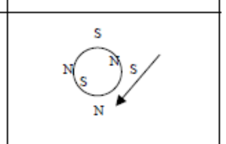
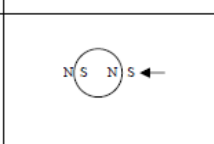
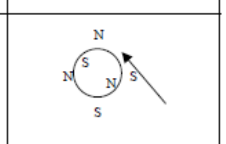


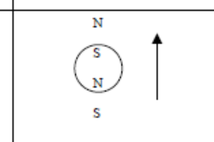
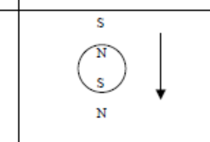
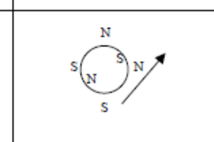
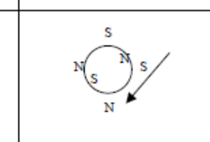
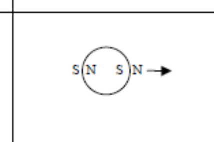
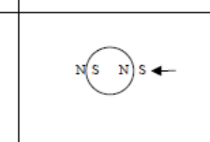
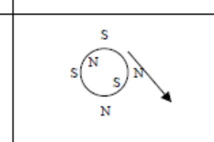
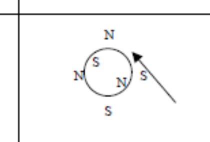
Fig3 - One-two phase on - half step

PFM PULSE EMISSION: STEPPER MOTORS





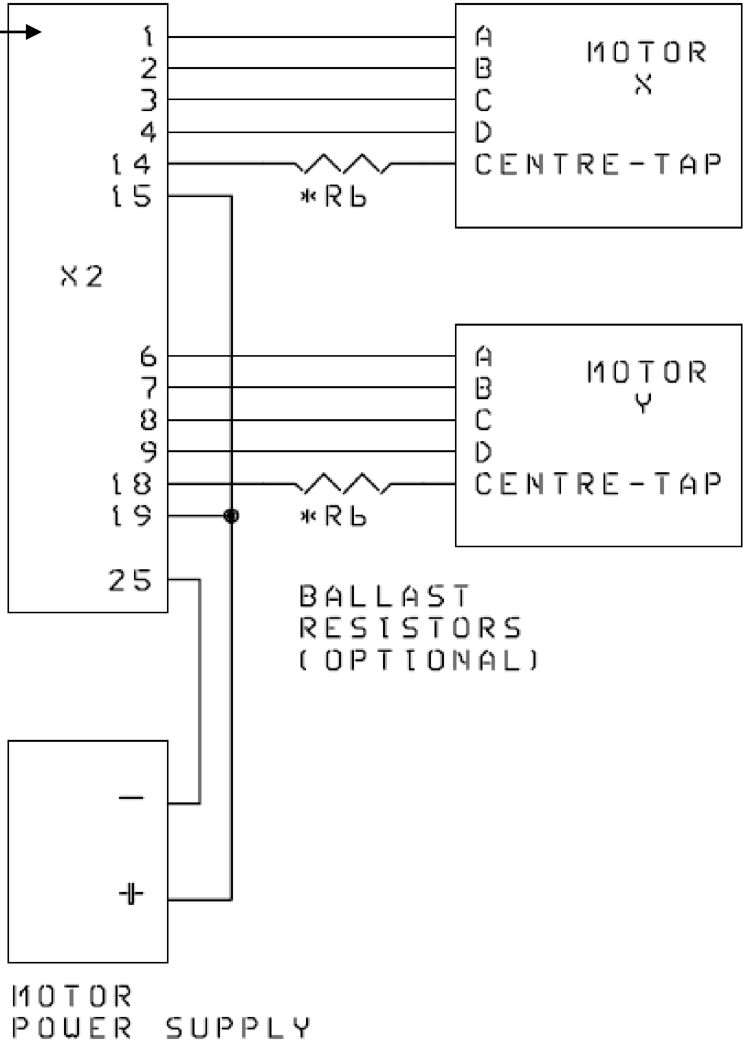
1a. WAVE DRIVE MODE		1b. NORMAL DRIVE MODE	
Transistor Accesi	**Posizione Rotore**	**Transistor Accesi**	**Posizione Rotore**
Q1		Q1 e Q2	
Q2		Q2 e Q3	
Q3		Q3 e Q4	
Q4		Q4 e Q1	

1c. HALF STEP MODE			
Transistor Accesi	**Posizione Rotore**	**Transistor Accesi**	**Posizione Rotore**
Q1		Q3	
Q1 e Q2		Q3 e Q4	
Q2		Q4	
Q2 e Q3		Q4 e Q1	

PFM PULSE EMISSION: STEPPER MOTORS

Interface μP 25 pin

X2 pin	Motor	Coil
1	X	A
2	X	B
3	X	C
4	X	D
14, 15, 16, 17	X	CENTER-TAP
6	Y	A
7	Y	B
8	Y	C
9	Y	D
18, 19, 20, 21	Y	CENTER-TAP
10, 11, 12, 13	INPUTS	
5	NOT CONNECTED	
22, 23, 24, 25	POWER SUPPLY GROUND	



PFM PULSE EMISSION: STEPPER MOTORS

HSTEP motor.....................Set motor to use half-step drive sequence.

2PHASE motor...................Set motor to use two-phase drive sequence. This is the default drive sequence.

WAVE motor......................Set motor to wave drive sequence

RATE motor val1 <val2>...Set the delay rate between steps (in milliseconds). Range from 1 to 50,000.

DIR motor <CW, CCW>....Set the direction of rotation. CW = clockwise, CCW = counter-clockwise. If no direction is given then it is reversed.

STEP motor val1 <val2> ....Step motor by val1 steps. If motor is both then val1 refers to 'X" and val2 refers to 'Y'.

WAIT motor.......................Wait for the previous motor command to finish before executing the next command for that motor.

SPIN motor.........................Continuously step motor. Not affected by 'WAIT'.

STOP <motor> ....Immediately stop the current command for motor. If no motor is specified then stops both. Ignores 'WAIT'.

MANUAL ....Manually step motors. Another menu appears.

LOOPTIL input(n) HIGH,LOW where n=1, 2, 3 or 4.Wait for the specified input to go high or low before continuing.

DELAY milliseconds.........Waits the specified delay time before continuing. Range = 1 to 50,000.

ECHO ON or OFF ....If ON (default) then all commands are echoed to the output.

PRINT message.................to output

VER...................................Print version number

END...................................Wait for all motor commands to finish then quit

QUIT ....Quit program immediately

HELP.................................Display above summary

STEPPER MOTORS: how to power supply

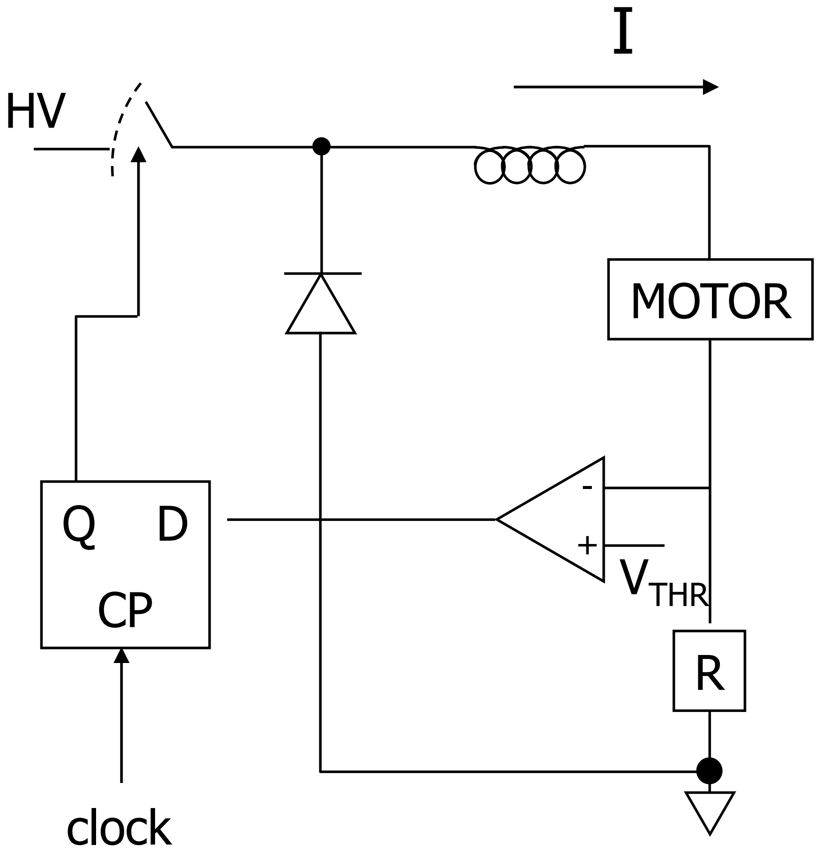
•This actuator responds with a velocity to the control input variable, allowing to determine the new position without any need of a position transducer (i. e. if we fix a 100 pulses/sec velocity, this corresponds to half a round in a second, if the sensibility of the transducer is 1 round after 200 pulses). Sometimes a limit switch is necessary.

•These motors require power supply in the order of few V, with currents in the order of the A ⇒ r in the coils must be relatively low; since the delay is L/r in an inductor, time constants can be high. L moreover can be high if the coils windings are high.

For example if the command voltage is applied to the coil 1, this one due to the initial 'resistance' will inhibit the, at least at the beginning, the current flow. This means that in the MOS drain V_{cc} is nearly present and that the MOS will employ further time to exit from the interdiction towards saturation ⇒ this is acceptable if the working frequency is not so high.

It is better to provide higher voltages (⇒ > resistors, < time constants): however this brings to a strong power dispersion on the MOS (heat). Thus a *PWM* solution is employed with suitable on-off intervals (tunable duty cycle).

STEPPER MOTORS: PWM POWER SUPPLY



The motor receives a constant current, obtained through a HV voltage necessary to provide fast commutations (time constant L/r)

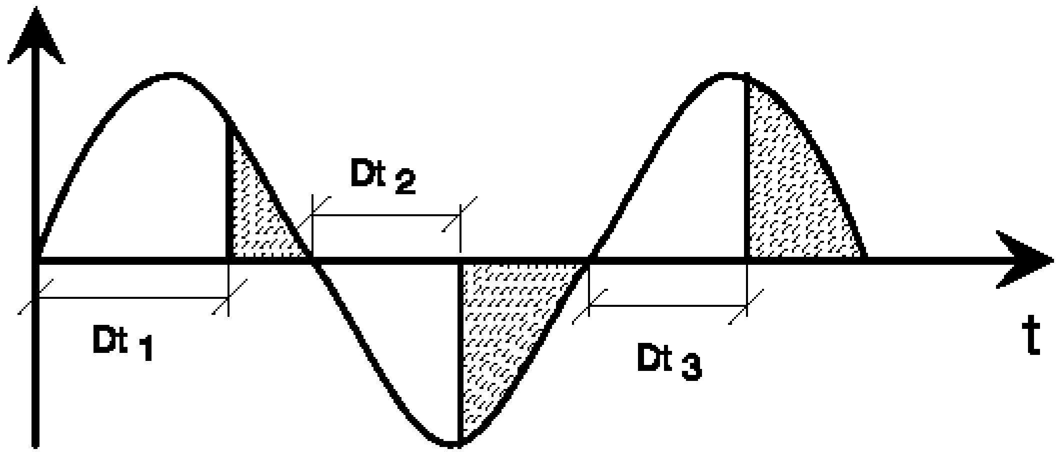
If current is < V_{rif}/R the flip flop close the switch so supplying the motor with a 100 V voltage. Alternatively (open switch), the circuit is grounded through a diode.

The flip flop closes the switch depending on a clock period that allows to regulate the opening and closing intervals (that means the duty cycle).

The inductor serves as "regulation maker" allowing a smooth passage among the levels of the current (high low high low) without shocking the motor.

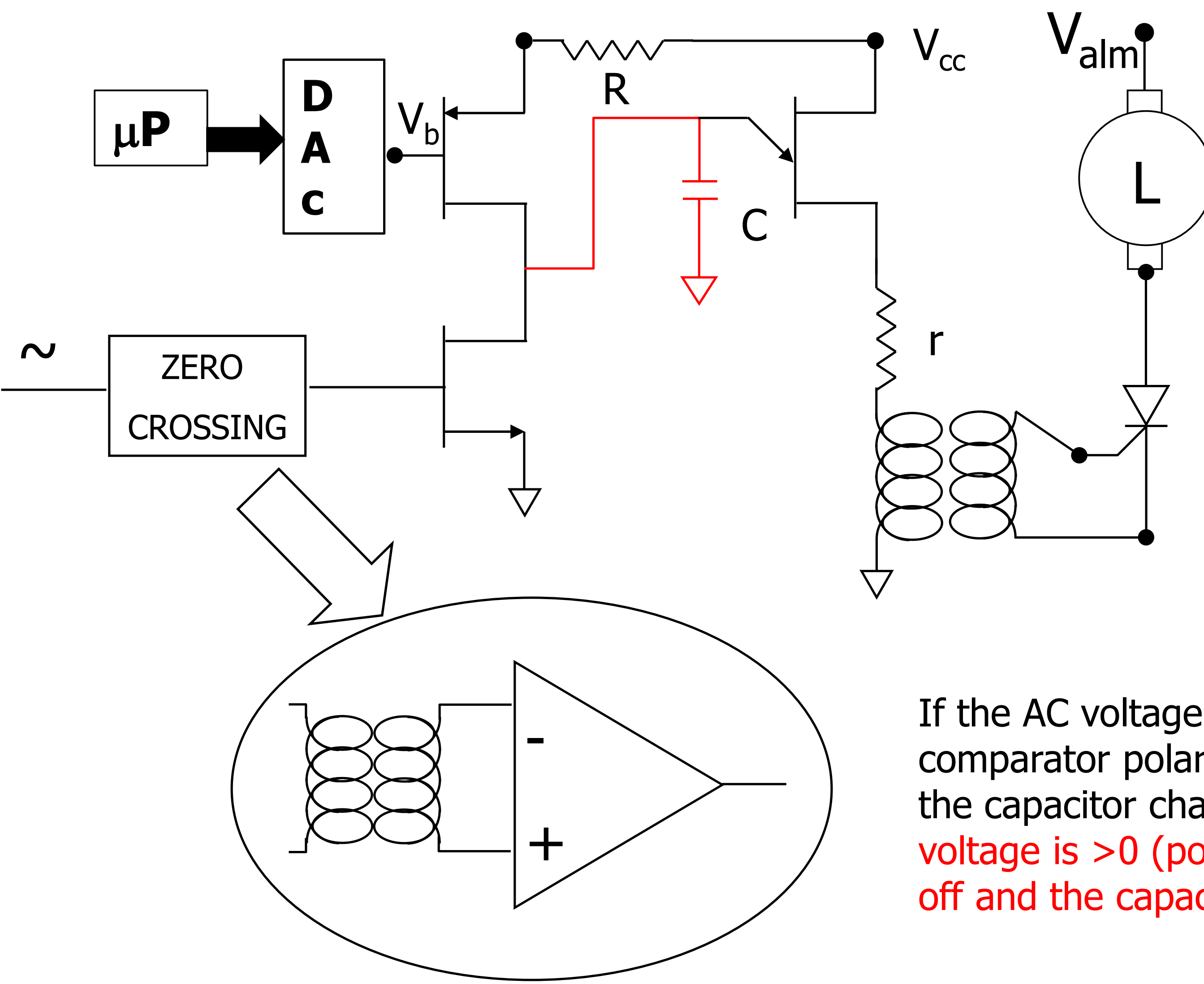
ON OFF control.

PULSE EMISSION: WAVE PARTIALIZATION



- “controlled” AC power supply through SCR/TRIAC
- The wave passage through zero must be revealed then a μP should provide the pulses for SCR/TRIAC activation
- Wave period 20 msec, the passage through zero has to be detected with very high precision ⇒ HW solution
- D(t) can be settled through a sw routine or through a timer with a interrupt to μP
- Non linear relationships between the average power provided and the power up delay

PULSE EMISSION: WAVE PARTIALIZATION

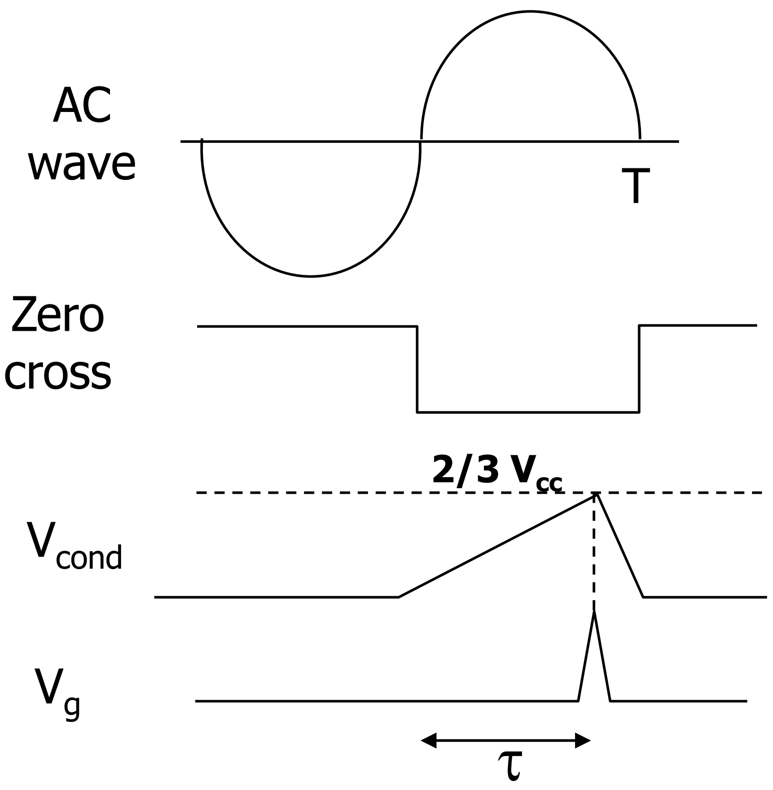


The C capacitor is discharged on the UJT emitter so as to obtained strong and fast pulses able to power up the SCR

When V_{cond} is $2/3V_{cc}$ the UJT transistor for a short time period is active since the capacitor quickly empties

If the AC voltage is <0 (negative wave), the comparator polarises the MOS preventing the capacitor charge, otherwise if the AC voltage is >0 (positive wave), the MOS is cut off and the capacitor is charged.

PULSE EMISSION: WAVE PARTIALIZATION



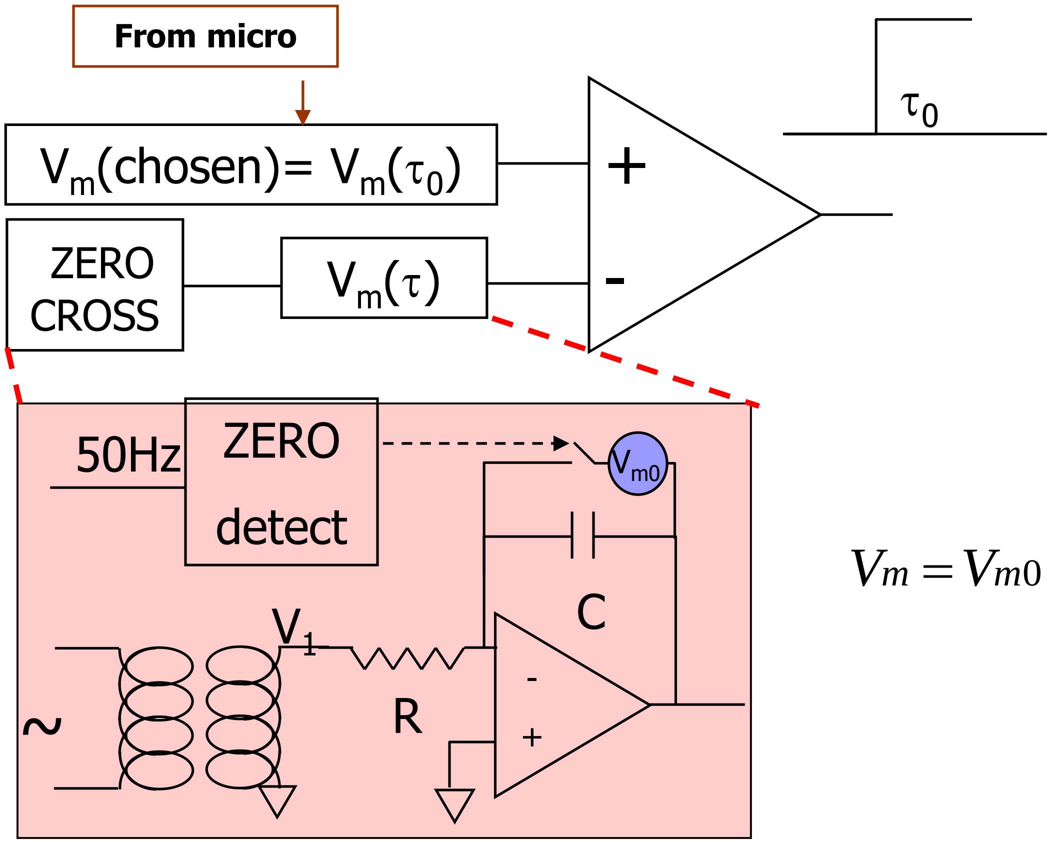
$$T = \frac{\frac{2}{3}Vcc}{Ic} \qquad Ic = \frac{Vcc - Vb - 0.7}{R}$$

$$T = \frac{2\ RC\ Vcc}{3\ (Vcc - Vb - 0.7)}$$

V_b allows to regulate the capacitor charge velocity so to establish when (τ) to power up the SCR (i. e. the provided power).

High non linearity between the delay and the power applied to the load.

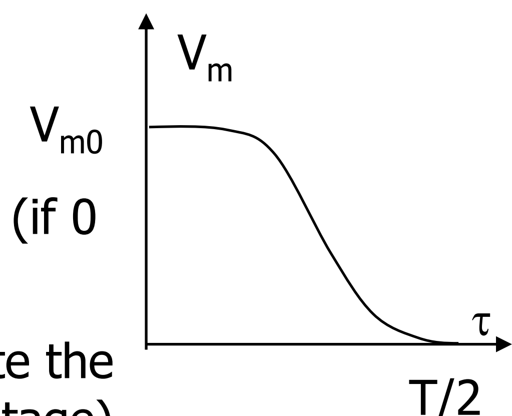
PULSE EMISSION: WAVE PARTIALIZATION



From the μP we set the chosen V_m (related to the desired power), so as when the corresponding τ is reached the comparator exits a pulse to power up the SCR.

How to generate $V_m(\tau)$?

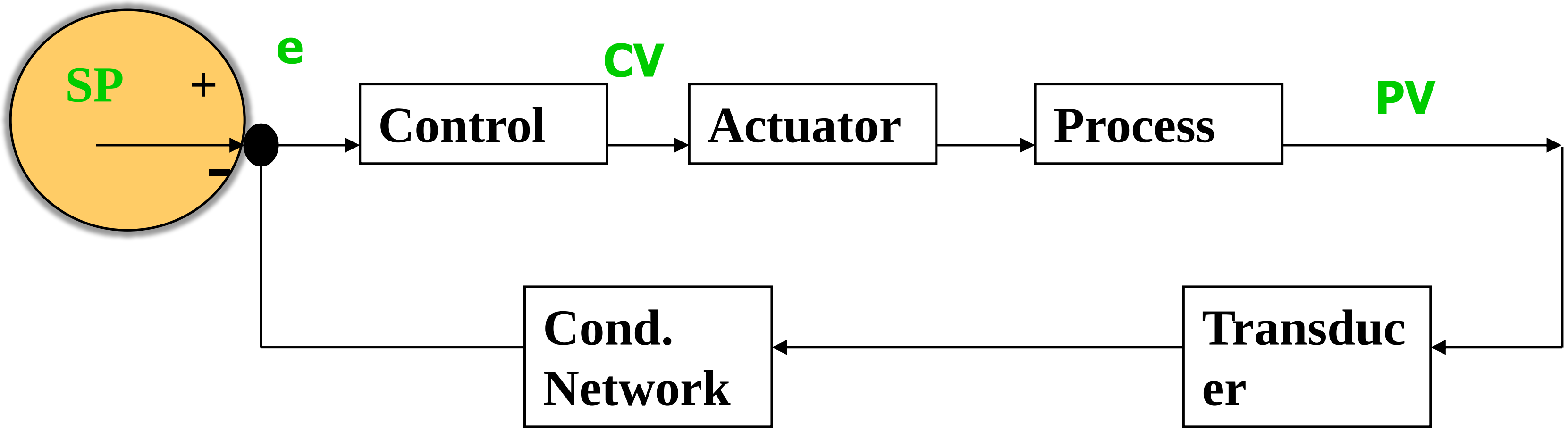
$$V_m = V_{m0} - \int_0^{\tau} \frac{V_1}{RC}\,dt \qquad V_{m0} = \int_0^{T/2} \frac{V_1}{RC}\,dt$$



By setting $V_m(\tau_0)$ from the μP the load voltage can be regulated (if 0 null power if V_{m0} max. power)

As an alternative a suitable lookup table in μP memory can relate the power delay (t) with the corresponding average load power (voltage).

CODED SET POINT VALUES



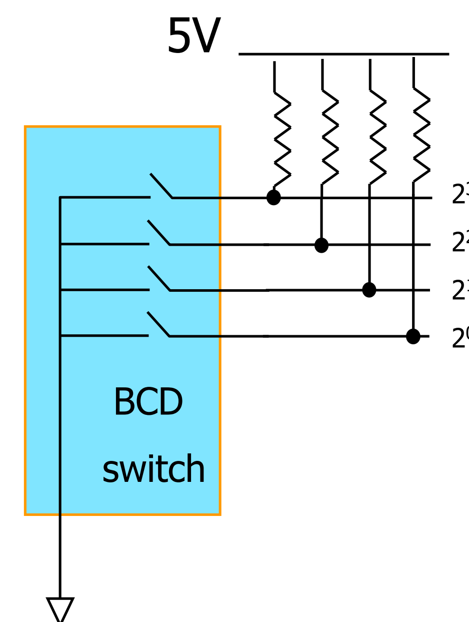
HW CODED (SETUP) VALUES: BCD SWITCHES

Signals alone do not bring significant information: it's their ordered ensemble that represents an information

A typical example is the acquisition of BCD numeric values for the digital set point of a regulation chain = *contraves*

Multiswitch contrave with rotational setup of the digits

Multiswitch contrave with linear (push button) setup of the digits



Equivalent electronic circuit of every terminal element (digit)

HW CODED (SETUP) VALUES: BCD SWITCHES

OEM COURTESY

Possible problems during commutations:

•Brushing, slithering and contacts rebounds

•Irregular commutations due to long usage

•Passage through halfway not significant values

•Casual order through which the operator moves digits

•Several input instructions (if for example more than 4 BCD digits and μP with <= 16 bits)

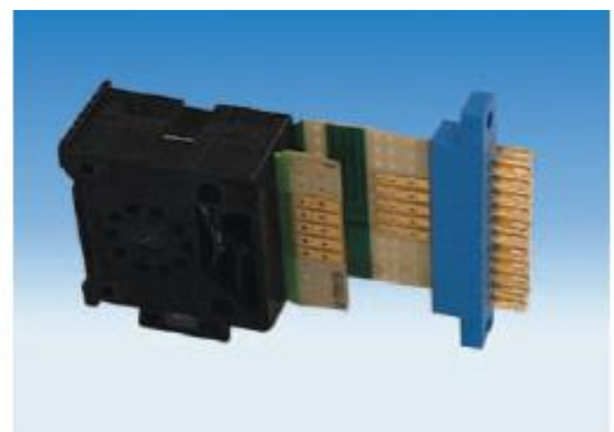
Solutions:

•Ok temporarily "out of range" values

•A suitable "settlement time" (SW) after which values are accepted

•Validation button (HW)

•A sw conversion from the BCD value to the machine binary representation is needed …



CODED VALUES: ABSOLUTE ENCODERS

Gray code is used to minimise the error due to a bad alignment of the optics internal to the encoder.

Pseudocode algorithm

```
commutare := 0
i := N-1
while i >= 0
    BitBin[i] := BitGray[i]
    if commutare = 1
        then BitBin[i] := not BitBin[i]
    commutare := BitBin[i]
    i := i-1
endwhile
```

C version

```
int Gray12_bin (NumGray);
{
  int commuta,i,mask;
      valore = NumGray;
      i = 12;
      mask = 0x800;
      commuta = 0;
      while (i > 0)
      {
            valore ^= commuta;
            commuta = valore & mask;
            commuta >>= 1;
            mask >>= 1;
            i--;
      }
      return (valore);
}
```

CODED VALUES: ABSOLUTE ENCODERS

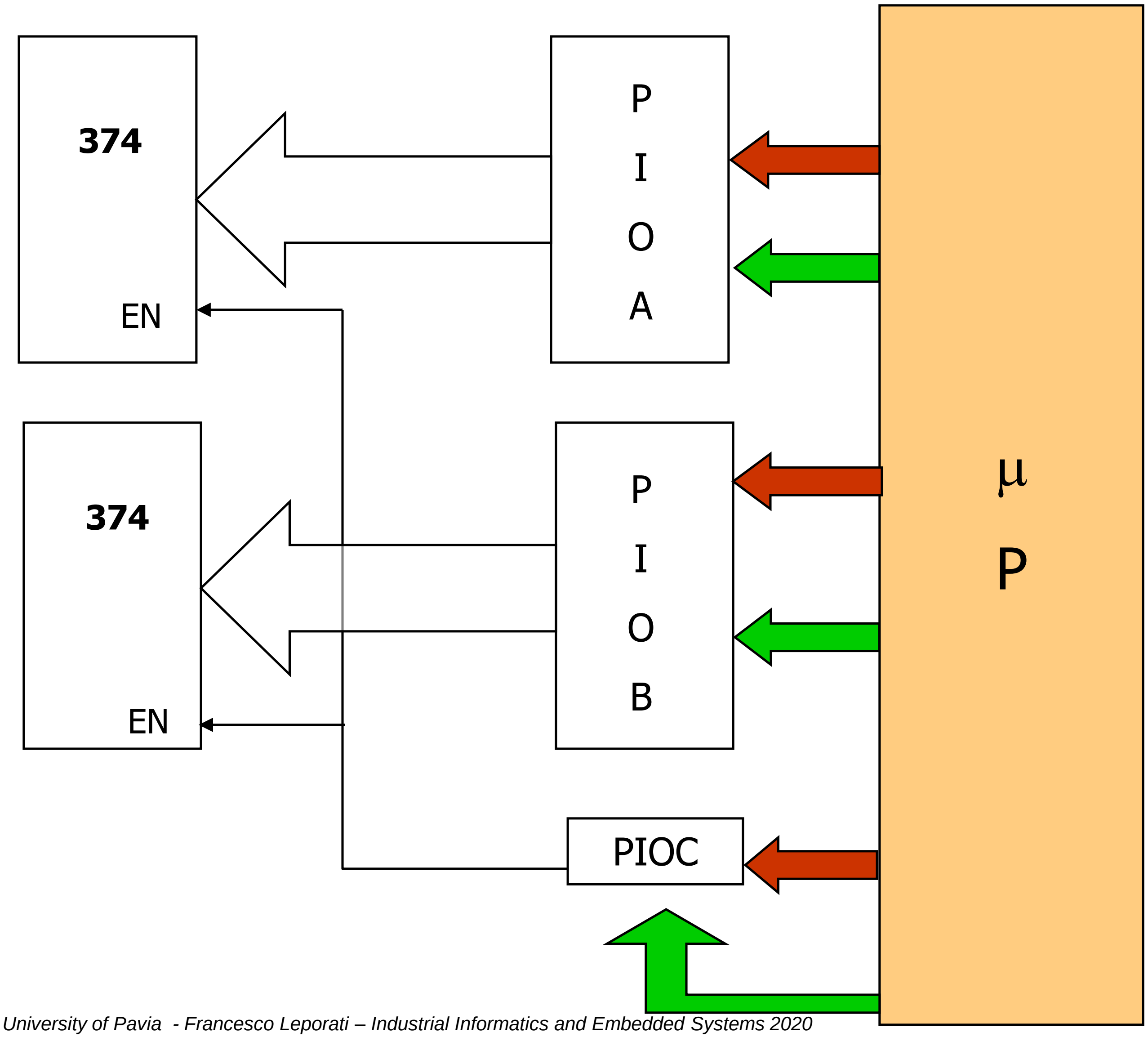
ASSEMBLY VERSION

Routine that takes as input the variable VALORE in Gray code 12 bit and exits with the pure binary converted value in AX (accumulator) register.

```
;
GRAY12_BIN:
    MOV     CX,12
    MOV     AX,800H
    SUB     BX,BX
CICLO:
    XOR     [VALORE],BX
    MOV     BX,[VALORE]
    AND     BX,AX
    SHR     AX,1
    SHR     BX,1
    LOOP    CICLO
    MOV     AX,[VALORE]
    RET
```

Integer	Gray code	Binary code
0	0000	0000
1	0001	0001
2	0011	0010
3	0010	0011
4	0110	0100
5	0111	0101
6	0101	0110
7	0100	0111
8	1100	1000
9	1101	1001
10	1111	1010
11	1110	1011
12	1010	1100
13	1011	1101
14	1001	1110
15	1000	1111

OUTPUT CODED VALUES



We suppose that we have to output more bits than those manageable with a single OUT instruction

OUT (PIOA), reg

OUT (PIOB), reg1

OUT (PIOC), WR_CMD

DATA BUS

ADDRESS BUS