CAP. 7

7. TECNICHE DI INTERFACCIAMENTO DIGITALE

Data la grande varietà di dispositivi che risultano interessanti da interfacciare con i calcolatori, si è preferito affrontare l'argomento dell'interfacciamento con segnali digitali secondo una classificazione per categorie di segnali, presentando le considerazioni generali del caso e analizzando più in concreto solo alcuni dei casi reali, scelti tra i più significativi.
Si noti poi che tutte le informazioni scambiate tra calcolatore e mondo esterno presentano prima o poi un'interfaccia digitale, e quindi quanto viene presentato in questo capitolo fornisce anche indicazioni utili per gli interfacciamenti con segnali analogici trattati nel capitolo seguente.

Pur trattandosi di segnali digitali non è affatto escluso, anzi è molto frequente, che si debbano superare disomogeneità fisiche e formali, e si debbano quindi prevedere circuiti di condizionamento per esempio dei tipi seguenti.

- Resistenze di *pull-up*.
- Filtri R-C.
- Riduttori di tensione a partitore.
- Resistenze e diodi di *clamp* a protezione da sovratensioni.
- Optoisolatori per isolamento galvanico.

Si invita il lettore a rivedere queste soluzioni circuitali, che sono date per note.

7.1 SEGNALI ON/OFF

Questi segnali rappresentano singoli bit e vengono in genere raggruppati a 8 o 16 per utilizzare completamente le porte di ingresso o di uscita dei calcolatori.
All'interno del calcolatore, a livello SW, vengono eseguite le operazioni di estrazione dei singoli bit acquisiti raggruppati in un byte, o di aggregazione di vari bit per formare un byte da emettere in uscita.

NOTA Nel linguaggio C non esiste il tipo `boolean`, ma si utilizzano i tipi `char` o `int`, con l'associazione convenzionale 0 = FALSE e 1 = TRUE. Nel linguaggio Assembler per il tipo `boolean` è spesso comodo usare un byte e associare 0 = FALSE e 0FFH = TRUE. Il valore esadecimale 0FFH corrisponde al valore binario 1111 1111.

Estrazione del bit in posizione K.

Assegna ad una variabile considerata come `boolean` il valore TRUE o FALSE, a seconda del valore del bit in posizione K della variabile di **immagine** della porta di ingresso in oggetto, mediante l'uso dell'operazione logica AND con una maschera che presenta un solo bit a 1 nella posizione K, e che quindi numericamente vale 2^K.

Esempio in C

```
var_bool = 0;
if (immagine & mask_K)
   var_bool = 1;
```

Variante più compatta:

```
var_bool = (immagine & mask_K) && TRUE;
```

Esempio in Assembler 8086 (versione intuitiva)

```
        MOV   AL, [IMMAGINE]
        AND   AL, MASK_K
        MOV   [VAR_BOOL], AL
        JZ    SKIP
        MOV   [VAR_BOOL], 0FFH
SKIP:
        ..... seguito ......
```

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 ......
```

Inserzione di bit in posizione K

Nella variabile immagine della porta di uscita assegna al bit in posizione K il valore di 0 o 1 a seconda del valore TRUE o FALSE della variabile `boolean` considerata.

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
```

7.1.1 ACQUISIZIONE DI SEGNALI ON/OFF

L'acquisizione di singoli bit, che in gergo non informatico sono spesso detti *segnali on/off*, può essere finalizzata a leggerne lo stato o a rilevarne gli eventi (le commutazioni).

7.1.1.1 Acquisizione di STATO

Per rendere possibile la lettura dello **stato** i segnali sono collegati ai bit di porte di ingresso realizzate mediante **buffer tri-state** che in occasione delle istruzioni di lettura immettono nel bus dati tali valori binari che vengono così trasferiti e caricati nel registro destinazione (in genere l'accumulatore) all'interno della CPU.

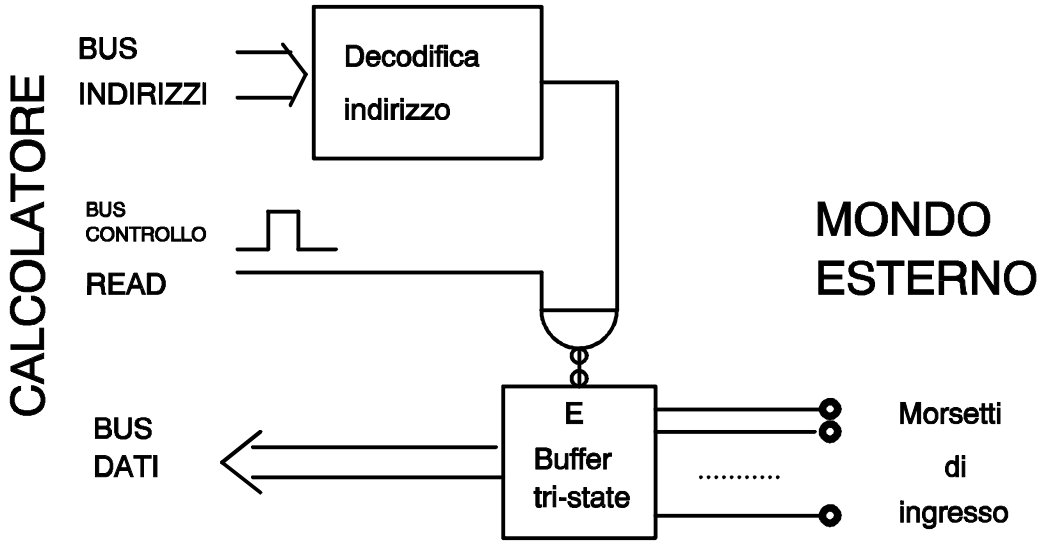


Fig. 7.1 - Schema di principio di porta di ingresso collegata al BUS di un calcolatore.

7.1.1.2 Acquisizione di EVENTI

Per il rilievo degli **eventi** di segnali binari il progettista può scegliere tra due soluzioni:

1) - Rilievo degli eventi a SW. In questo caso la connessione dei segnali viene effettuata sui bit di porte di ingresso, come per il rilievo degli stati, ed il compito di rilevare le commutazioni è affidato al SW che dovrà essere organizzato in modo da effettuare periodiche letture degli stati e confronti di essi con i valori della lettura precedente. Esempi di questa tecnica si trovano nel successivo paragrafo "filtraggio algoritmico".

2) - Rilievo degli eventi ad Interrupt. In questo caso i segnali dovranno essere collegati a particolari bit di ingresso in grado di inoltrare richieste di interrupt alla CPU.
- Se questi morsetti di ingresso sono del tipo **sensibile al fronte**, ad esempio di salita, si potra' effettuare il collegamento diretto se gli eventi significativi sono i fronti di salita, o tramite un invertitore se interessano i fronti di discesa.
- Se il morsetto di interrupt è **sensibile ad un fronte** ma si debbano rilevare **entrambi** i fronti, cioè tutte le commutazioni sia di salita che di discesa, occorre adottare un circuito rilevatore di commutazioni, di cui in fig. 7.2 sono riportate una versione asincrona (che è quella usuale) ed una versione sincrona (se si dispone di un clock a frequenza opportuna (esercizio: quale è il campo di frequenze accettabili per questo clock?).

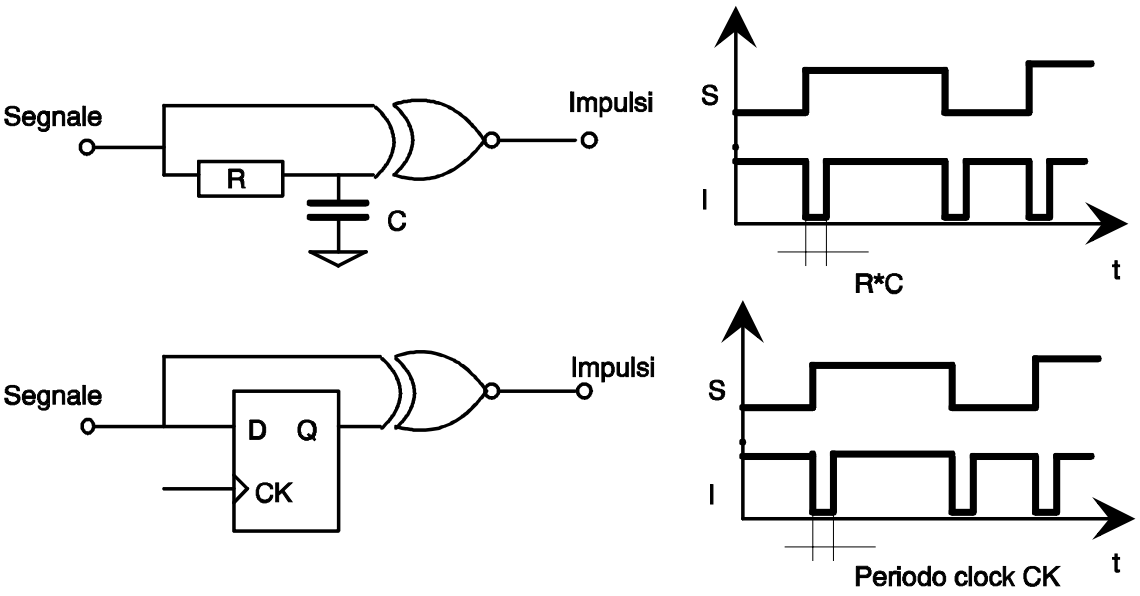


Fig. 7.2 - Circuito asincrono (sopra) e sincrono (sotto) per trasformare ogni commutazione del segnale S in un impulso I. - Si noti il simbolo di XOR.

- Se il morsetto di interrupt è **sensibile al livello**, nell'interfaccia si dovrà prevedere un bistabile S-R. Al morsetto Set si collega il segnale on/off tramite un rilevatore di fronti, mentre al morsetto Reset si collega un segnale di azzeramento prodotto da un'opportuna istruzione di lettura effettuata nell'ambito della risposta all'interrupt (fig. 7.3).

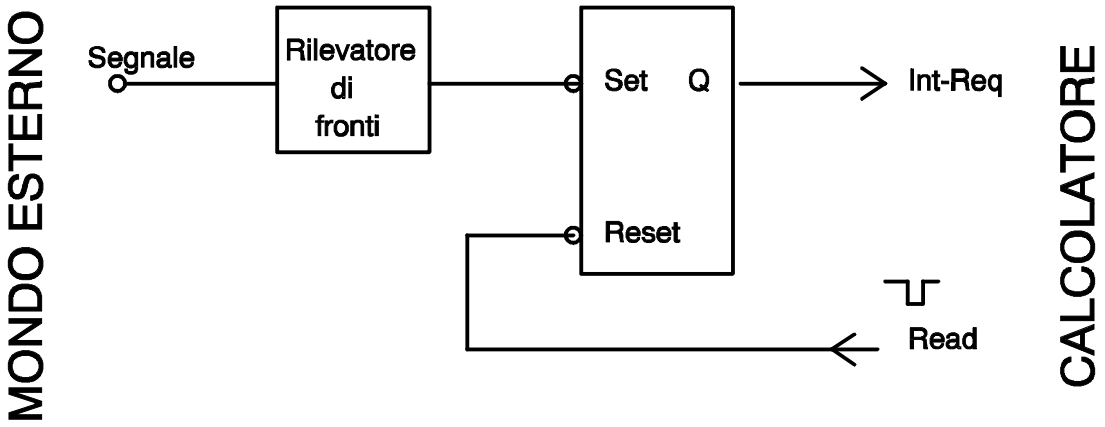


Fig. 7.3 - Circuito di inoltro di richiesta di interrupt per interrupt a livello.

Naturalmente prima del collegamento ai morsetti di circuiti logici si dovranno superare eventuali disomogeneità fisiche tra i livelli dei segnali disponibili nel mondo esterno e quelli richiesti dalla famiglia di circuiti integrati utilizzati per l'interfaccia.

Se si tratta di segnali generati da **circuiti elettronici** i problemi si riducono all'adattamento fisico dei livelli di segnale ed eventuale contestuale separazione galvanica tramite optoisolatori (v. fig. 7.4).

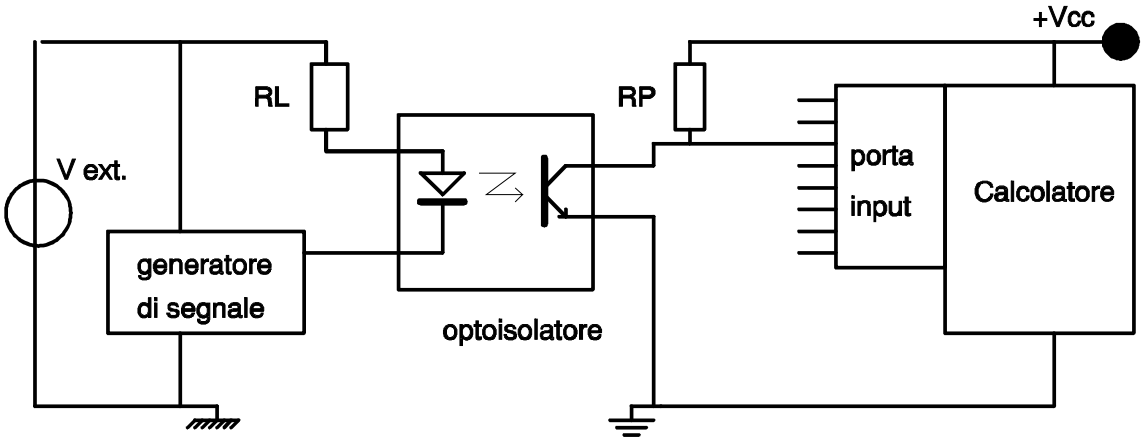


Fig. 7.4 - Schema di principio per collegamento di un bit in ingresso con isolamento galvanico, con l'ipotesi che il dispositivo esterno generi un segnale *attivo basso* e con capacità di *sink* di almeno 5..10 mA..

Nella figura 7.4 **Vext** rappresenta l'alimentatore dei circuiti esterni che generano il segnale. **RL** è una resistenza dimensionata (in base a Vext) per ottenere sul fotodiodo una corrente adatta (5 . . 10 mA). **RP** è una resistenza di *pull-up* di valore abbastanza basso per avere buoni fronti di salita sulla porta di input, ma anche abbastanza elevato per consentire al fototransistore di andare in saturazione. **Vcc** è l'alimentazione del calcolatore e dei relativi circuiti di interfaccia.

Molto spesso i segnali on/off sono prodotti da **contatti meccanici** (pulsanti, interruttori, relè, fine-corsa, ecc) che danno luogo a transitori in chiusura (rimbalzi) ed in apertura (strisciamenti) che possono protrarsi anche per diversi ms e di cui occorre tener conto in base alle specifiche sul significato dell'informazione rappresentata da tali segnali.

Se si acquisiscono informazioni di stato in genere non occorre prendere particolari provvedimenti, mentre è indispensabile un filtraggio **elettrico** o **algoritmico** se le informazioni da acquisire sono gli eventi "commutazione".

7.1.1.3 FILTRAGGIO ELETTRICO

Scopo del filtraggio è l'eliminazione delle commutazioni spurie.
Un tipico circuito di ingresso, con filtraggio elettrico adatto per segnali prodotti da un contatto è riportato in fig. 7.5.
Il trigger di Schmitt è molto opportuno per evitare di presentare in ingresso ad un circuito logico un segnale con fronti di commutazione lenti, come sono quelli causati dalla carica e scarica del condensatore. Infatti un passaggio lento del livello di segnale attraverso la fascia di incertezza induce facilmente un comportamento oscillatorio.

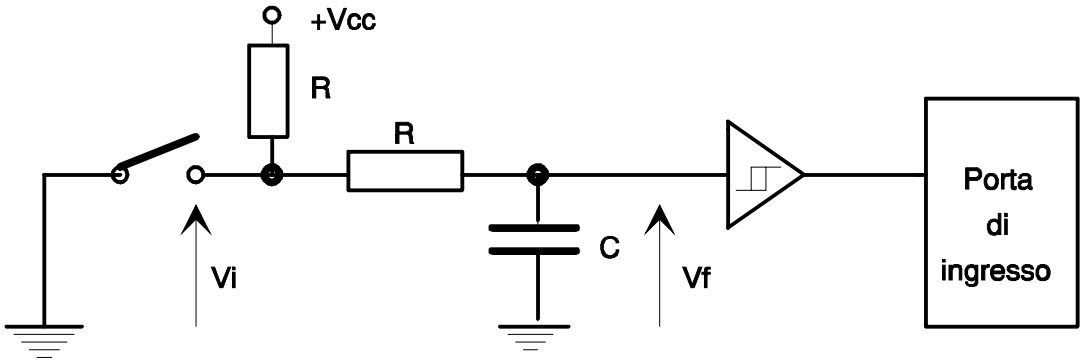


Fig. 7.5 Semplice circuito di filtraggio per contatto in ingresso, con trigger di Scmitt.

Il filtraggio *elettrico* è **indispensabile** se le commutazioni dovranno essere recepite a livello HW, come avviene per il rilievo mediante interrupt oppure per il conteggio diretto mediante un registro contatore.

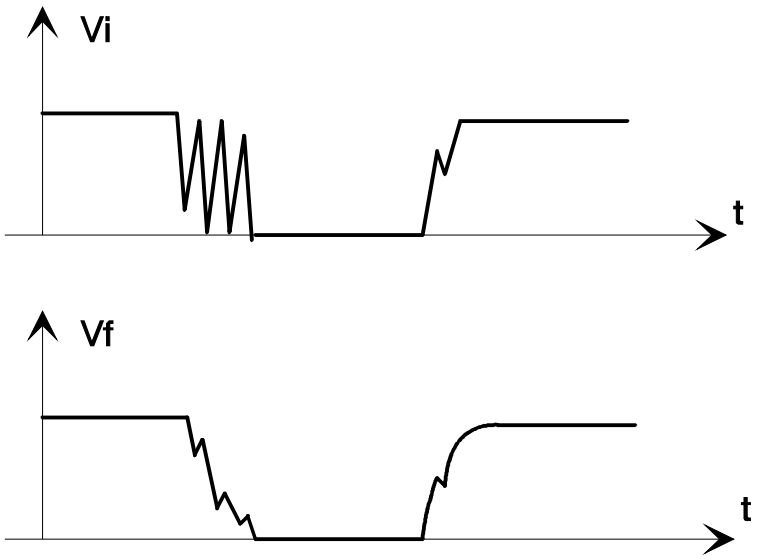


Fig. 7.6 Forme d'onda in ingresso e filtrata.

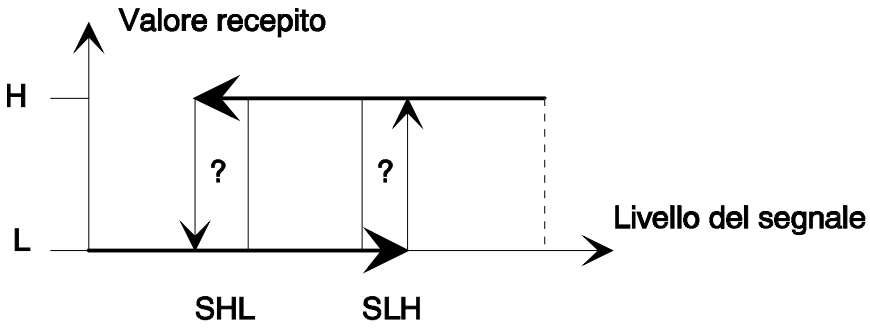
Si noti che i rimbalzi (come del resto gli spurii in generale) sono un esempio di eventi **del segnale** che **non corrispondono** ad eventi **del fenomeno** di interesse, che è ad esempio la pressione del tasto che aziona il contatto.

NOTA SULL'ISTERESI.

L'isteresi in generale designa un comportamento che non dipende solo dal valore di una certa grandezza, ma anche dall'andamento precedente di tale grandezza.

Il comportamento isteretico corrisponde quindi ad una forma di "memoria" di un dispositivo.

Per i circuiti elettronici digitali, come avviene tipicamente nello *Scmitt trigger*, l'isteresi corrisponde ad un comportamento come quello rappresentato in figura, che presenta due diverse soglie di interpretazione del segnale in ingresso come L o H.

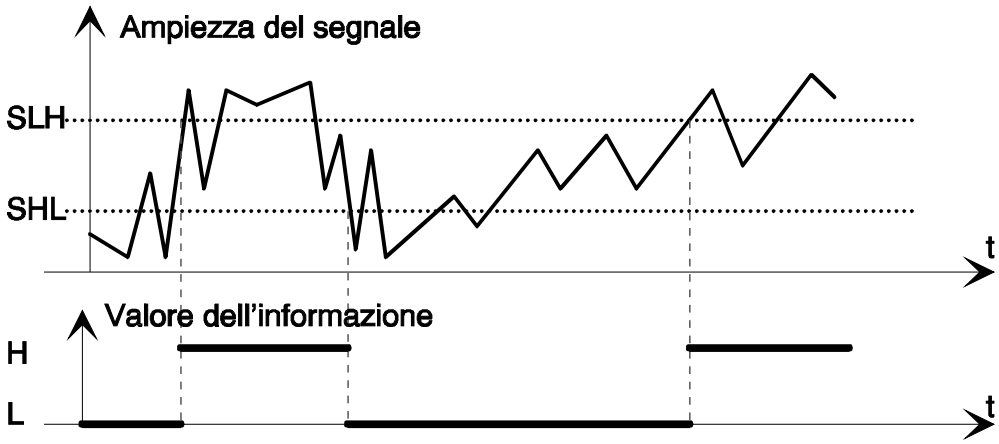


Il comportamento rappresentato in figura è il seguente.

Un segnale di valore inferiore a SHL è recepito come informazione di valore L.

Solo se il livello del segnale supera la soglia SLH viene recepita un'informazione di valore H. A questo punto il segnale deve scendere al di sotto della soglia SHL per essere nuovamente recepito come informazione di valore L.

L'isteresi non va confusa con la fascia di indeterminazione, anzi un circuito con isteresi presenterà due fascie di indeterminazione in corrispondenza delle due soglie di transizione L - H e H - L. Tali fascie vengono generalmente sottintese, anche perchè sono spesso piuttosto strette.



La figura rappresenta un esempio di andamento del segnale in funzione del tempo e la relativa interpretazione come valore L o H.

7.1.1.4 FILTRAGGIO ALGORITMICO

Il filtraggio algoritmico consiste nel ricavare il valore corrente di uno stato interno, che rappresenti l'informazione significativa, a partire dai valori di una successione di campionamenti precedenti dello stato fisico del segnale, acquisiti periodicamente.
Lo scopo e' di **preservare le caratteristiche significative** del segnale in ingresso, eliminandone invece le variazioni che sono considerabili come spurie. Naturalmente il criterio per stabilire quali caratteristiche sono significative dipende dal tipo di informazione associata al segnale. Il tipo di informazione determina un corrispondente tempo morto.

Sono riportati, come esempio di principio, dei sottoprogrammi in Assembler, che per ipotesi dovranno essere chiamati ciclicamente con periodo opportuno.
Si suppone che le variabili (PRECEDENTE e TRANSITORIO) siano inizializzate (per es. a 0) anche se l'algoritmo è robusto ed evolve verso il corretto funzionamento.

A) - RILIEVO DEL PRIMO FRONTE

Rileva la prima variazione del segnale dopo un periodo di riposo, ed accetta subito il nuovo stato come significativo ignorando le variazioni successive durante l'intervallo considerato di transitorio.
Questo algoritmo è significativo con l'ipotesi che non esistano spurii isolati (che questo algoritmo non elimina) ma che eventuali spurii seguano sempre un evento significativo e si esauriscano entro il tempo di transitorio. Queste ipotesi sono ben rispettate da segnali prodotti da contatti meccanici.

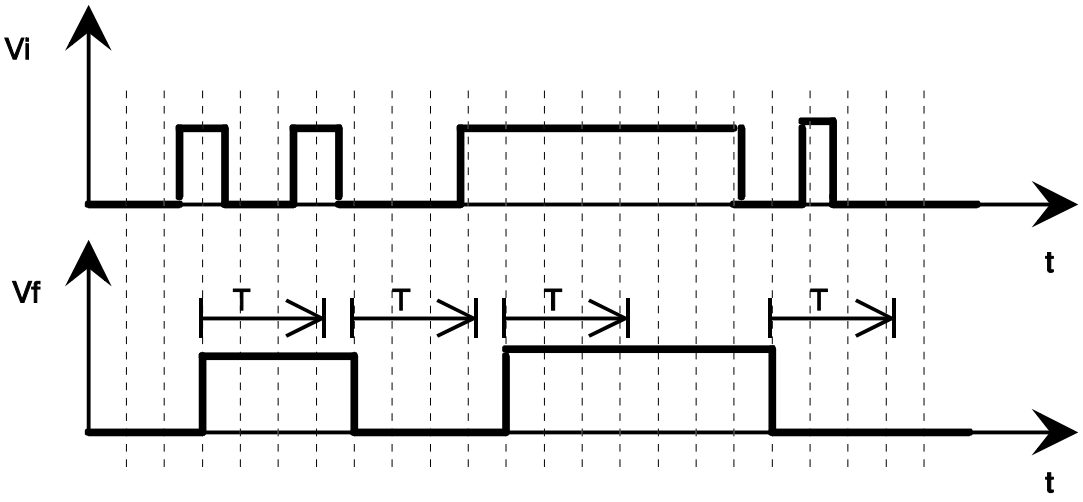


Fig. 7.7 - Valore letto (Vi) e valore filtrato algoritmicamente (Vf).
T è il tempo morto di transitorio e le linee tratteggiate indicano gli istanti di campionamento, cioè di esecuzione della routine di filtraggio riportata nel seguito.

```
; 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
```

B) - RILIEVO A FINE TRANSITORIO

Accetta come significativo un nuovo stato solo dopo che questo permane immutato per il periodo di transitorio. Questo algoritmo produce un effetto simile a quello di un filtraggio elettrico *passa-basso, compreso l'effetto di ritardo.*
Questo algoritmo elimina gli spurii isolati, caratteristici di disturbi indotti da perturbazioni esterne.

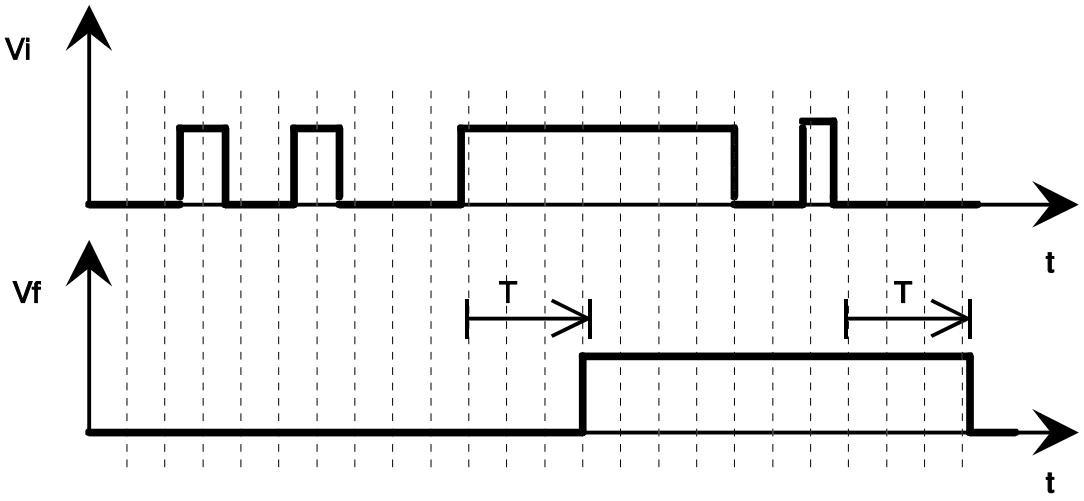


Fig. 7.8 Effetto del secondo tipo di algoritmo.(B)

```
; 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
```

7.1.2 EMISSIONE DI SEGNALI ON/OFF

L'emissione di segnali digitali avviene mediante istruzioni di scrittura su porte d'uscita che devono essere realizzate con **registri** (*latch*) in grado di catturare i bit del BUS dati in corrispondenza dell'operazione di OUT e presentare tali valori permanentemente in uscita.
Per l'emissione di segnali ON/OFF sono importanti i seguenti aspetti:

- All'inizializzazione del sistema, come pure quando vengano rilevate eccezioni non recuperabili, è necessario emettere sulle porte di uscita interessate tutti i bit con i rispettivi valori di "riposo". Il valore di riposo dipende ovviamente dalla realizzazione circuitale dell'interfaccia (invertente o meno) ma anche dai dispositivi collegati. Quando è possibile si assume come valore di riposo il livello 0.
- - Poichè nelle normali porte di uscita non è possibile scrivere singoli bit, è necessario mantenere aggiornato il valore dei vari bit di variabili di "*immagine di uscita*" che vengono utilizzate per la scrittura sulle porte corrispondenti.
  I seguenti esempi sono indicativi della tecnica citata, ed assumono che le sequenze di istruzioni non siano interrompibili.

```
SET_BIT_N:
        MOV     AL,[IMM_P_OUT]      ;legge immagine porta in uscita
        OR      AL,MASK_N           ;mette a 1 il bit voluto
        OUT     PORTA_OUT           ;emette nuovo valore
        MOV     [IMM_P_OUT],AL      ;aggiorna immagine
        RET

RESET_BIT_N:
        MOV     AL,[IMM_P_OUT]
        AND     AL,NOT MASK_N
        OUT     PORTA_OUT
        MOV     [IMM_P_OUT],AL
        RET

COMMUTA_BIT_N:
        MOV     AL,[IMM_P_OUT]
```

```
XOR   AL,MASK_N
OUT   PORTA_OUT
MOV   [IMM_P_OUT],AL
RET
```

7.2 IMPULSI

Per impulsi intendiamo **coppie di commutazioni** di un segnale binario che da uno stato di riposo (L o H) portano allo stato opposto (H o L) e, dopo un "breve" intervallo di tempo, nuovamente allo stato di riposo. Si tratta cioè di **coppie di eventi** di cui spesso uno solo è considerato significativo (quasi sempre il prino, cioè l'inizio dell'impulso) e tra i quali la distanza temporale non assume particolare significato, se non per il fatto che deve essere tale da garantire il corretto funzionamento dei circuiti e/o degli algoritmi di interfaccia che devono rilevare gli impulsi stessi.
Ad un impulso viene quindi generalmente associato *un solo evento*, anche se lo stato finale del segnale (dopo l'impulso) coincide con quello iniziale.
In altri casi invece un aspetto a cui è associata un'informazione significativa degli impulsi è proprio la durata. Si considerano allora significativi **entrambi** gli eventi di inizio e di fine degli impulsi.

7.2.1 ACQUISIZIONE DI IMPULSI

Gli impulsi, come si è detto, rappresentano in genere degli **eventi** rispetto ai quali possono essere significative diverse informazioni che sono ricostruibili mediante elaborazioni che analizziamo brevemente nel seguito.

7.2.1.1 Acquisizione di impulsi per conteggio (counter)

Sono numerose le situazioni applicative in cui gli impulsi devono essere semplicemente contati, a prescindere dalla loro frequenza e durata.

7.2.1.1.1 a) - Impulsi prodotti dal passaggio di oggetti in una catena di produzione.

In questi casi si vuole in genere mantenere aggiornato un contatore totalizzatore rappresentato da una variabile nel programma applicativo. Si noti che sono riconducibili a questo caso anche conteggi di giri di una girante per rilevare la portata di un fluido, ecc.

- I valori dell'informazione utile sono costituiti da numeri naturali (senza segno)

- Occorre specificare il conteggio massimo, da cui dipende il numero di bit necessari per la variabile di conteggio. Per conteggi con valori elevati può essere necessario usare variabili di tipo `unsigned long`.

- In alcuni casi è richiesto che i contatori **non siano volatili** per mantenere memorizzati i conteggi totali nell'arco di lunghi intervalli di tempo. In tali casi si adottano contatori **secondari** in RAM tampone o su memoria di massa. Occorre adottare opportuni algoritmi per l'aggiornamento dei contatori secondari, con eventuali compromessi tra il

tempo macchina richiesto per un aggiornamento molto frequente e il rischio di perdita di conteggi in caso di caduta dell'alimentazione o crash che portano alla perdita del valore dei contatori primari.

- Occorre adottare una tecnica opportuna per le prevedibili operazioni di azzeramento dei contatori che possono essere manuali (su richiesta dell'operatore) o automatiche (a tempo: ad es. ogni inizio turno; o su evento: ad es. ad ogni nuovo lotto di produzione) con eventuale riporto su altri contatori. La sequenza di operazioni *copia-e-azzera* deve costituire un'operazione atomica, cioè non interrompibile, per evitare occasionali perdite di conteggi.

Talvolta si adotta un contatore primario realizzato in HW: ciò è utile per alleggerire il carico di elaborazione SW e diviene necessario se la minima durata degli impulsi o la loro minima distanza temporale risultano inferiori a quelle rilevabili direttamente da SW (alcune decine di microsecondi).
Questo contatore primario HW può essere considerato come:

1) portatore di un'informazione di stato
2) generatore di eventi ad ogni riporto (*overflow*).

Analizziamo brevemente i due casi.

1) - Se si gestisce il contatore primario come supporto di **informazione di stato** si potrà aggiornare la variabile di conteggio software con un algoritmo simile a quello riportato nell'esempio seguente, che va eseguito in istanti a piacere con il solo vincolo che tra un aggiornamento ed il precedente non si siano presentati più di $(2^N - 1)$ impulsi se il contatore primario è di N bit.

ESEMPIO
Sia collegato alla porta PORTA_CONT il contatore da 8 bit.
Il conteggio vada effettuato sulla variabile CONTATORE di tipo unsigned long
La variabile PRECEDENTE sia un byte di immagine dell'ultimo valore letto.
La routine venga chiamata ogni meno di 256 impulsi.

```
AGGIORNA_CONTATORE:
        SUB     AX,AX                   ;azzera AX
        IN      AL,PORTA_CONT
        SUB     AL,[PRECEDENTE]         ;conteggi aggiuntivi
        ADD     [PRECEDENTE],AL         ;aggiorna immagine
        ADD     [CONTATORE],AX          ;aggiorna 16 LSBit contatore
        ADC     [CONTATORE + 2],0       ;eventuale riporto su 16 MSBit
        RET
```

E' molto importante notare che occorrono particolari precauzioni nella lettura di contatori con più bit di quanti possano essere letti con una singola operazione elementare.

La soluzione HW del problema consiste in un registro ausiliario in cui "fotografare" il valore del contatore; tale registro potrà poi essere letto per parti.

La soluzione SW può consistere in una successiva rilettura del valore per verificare che esso non sia cambiato tra la lettura di una sua parte e dell'altra, ad es. con la tecnica seguente in cui il contatore primario di 16 bit è letto dalle porte di indirizzi P_MSBYTE

(byte più significativo) e P_LSBYTE (byte meno significativo), mentre il contatore secondario a 32 bit è costituito dalla variabile CONTATORE.

```
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]         ;conteggi aggiuntivi
        ADD     [PRECEDENTE],AX         ;aggiorna immagine
        ADD     [CONTATORE],AX          ;aggiorna 16 LSBit contatore
        ADC     [CONTATORE + 2],0       ;eventuale riporto su 16 MSBit
```

2) - Se si gestisce il contatore come **generatore di eventi**, lo si utilizza in pratica come un "*prescaler*" o divisore di frequenza, facendo in modo che esso generi una richiesta di interrupt ad ogni riporto. La routine di servizio dell'interrupt provvederà ad aggiornare il contatore secondario, incrementandolo del valore corrispondente al fattore di divisione.
Questo approccio è interessante per ridurre il carico di lavoro SW nel caso di impulsi a frequenze (talora) elevate.
Si noti che con questa tecnica, quando occorre conoscere l'esatto valore di conteggio si deve provvedere ad una lettura diretta del valore raggiunto dal contatore HW (che rappresenta gli “spiccioli”) e a sommare tale valore a quello del contatore secondario.

7.2.1.1.2 b) - Impulsi prodotti dal movimento di una parte di cui si vuole conoscere la posizione

Gli impulsi rappresentano in questo caso spostamenti incrementali e quindi, per trasformarli in posizioni assolute si dovrà gestire correttamente il posizionamento, molto spesso convenzionale, della posizione assunta come origine.
- A seconda della collocazione dell'origine (posizione di zero) si adotteranno le seguenti rappresentazioni:

- -- con origine ad un estremo invalicabile del campo di spostamenti: numeri naturali (senza segno)
- -- con origine in posizione arbitraria: numeri interi (con segno)

- In genere i sensori di spostamento che producono impulsi, cioè eventi incrementali, ad ogni spostamento elementare sono detti "encoder incrementali" ed in realtà generano eventi più complessi che dei semplici impulsi. Ciò risulta necessario nei casi in cui si debbano distinguere gli spostamenti nelle due direzioni.

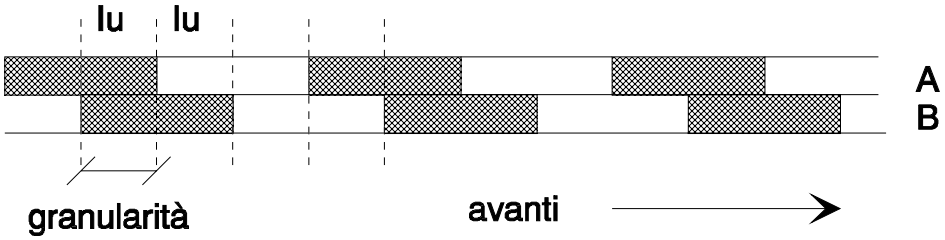


Fig. 7.9 - Aspetto delle tracce di un encoder incrementale. Elementi fotosensibili rilevano le aree opache o trasparenti.

Con questa tecnica si generano due segnali A e B le cui forme d'onda si presentano come in fig.7.10.

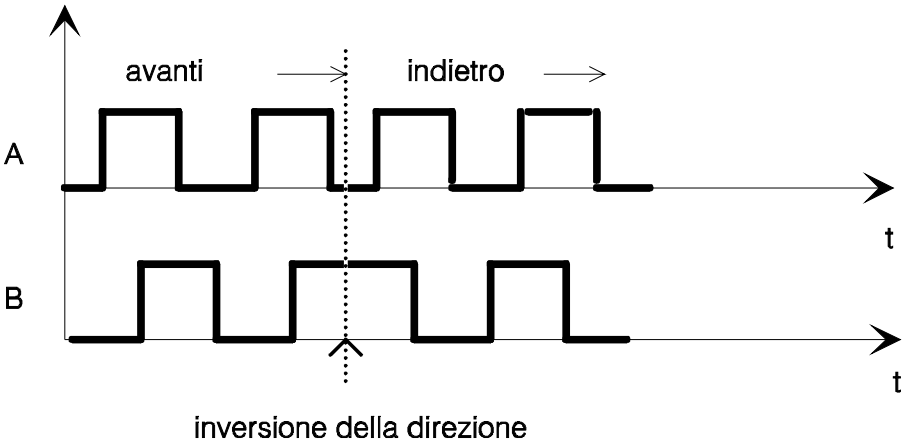


Fig. 7.10 Segnali interallacciati (*overlapping*) di encoder incrementale.

-- Nel caso frequente di encoder rotanti, calettati cioè su un albero di cui si vuole rilevare lo spostamento angolare, viene in genere affiancato un terzo segnale che produce un impulso ad ogni giro, utilizzabile per eventuali risincronizzazioni in caso di perdita o errori di conteggio.
Tale impulso costituisce un evento assoluto rispetto al singolo giro dell'encoder, e un evento relativo rispetto a più giri.

- Il conteggio massimo raggiungibile è in genere noto per traslazioni di parti di macchine che presentano una corsa ben delimitata, mentre dovrà essere una specifica esplicita di progetto per altri casi (es. contachilometri).

- Notevole importanza assume la cosiddetta impostazione dello "zero macchina" che consiste nell'azzeramento del contatore quando la parte di cui si rileva la posizione si trova nell'origine (eventualmente convenzionale come è usuale per le slitte di alcune macchine utensili).

Le operazioni da eseguire per la gestione dei segnali di un encoder incrementale a due segnali (bidirezionale) sono:
- rilievo dell'evento commutazione del segnale A o B
- decisione se si tratta di incremento (avanti) o decremento (indietro) della posizione
- aggiornamento del contatore primario
- eventuale aggiornamento del contatore secondario.

Queste operazioni possono essere realizzate tutte a livello SW oppure, tranne l'aggiornamento del contatore secondario, a livello HW.

ESEMPIO DI GESTIONE **SW** DI ENCODER INCREMENTALE BIDIREZIONALE

Sia il segnale A sul bit B0 e il segnale B sul bit B1 della porta P_ENC
Si consideri positiva le direzione che vede B=1 sui fronti di discesa di A.
Si supponga la routine chiamata abbastanza frequentemente da non perdere più di una commutazione o su A o su B.
Si noti che questa ipotesi non può mai essere garantita se si possono avere inversioni rapide di movimento a cavallo di una commutazione. In particolare ciò si verifica nel caso poco probabile, ma possibile, di arresto in corrispondenza della zona di commutazione di A o di B, in cui l'incertezza del livello del segnale e/o piccole oscillazioni meccaniche possono portare a frequenti commutazioni di segno opposto.
Si verifica facilmente che queste perdite di eventi non comportano errori, trattandosi di eventi che si elidono.

ALGORITMO

- se nulla è cambiato esce
- se commutato B
  - se B = A incrementa
  - altrimenti decrementa
- altrimenti (commutato A)
  - se B = A decrementa
  - altrimenti incrementa

N.B. XOR 1 commuta il bit di posizione 0 (bit meno significativo) !.

```
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
```

ESEMPIO DI GESTIONE **HW** DI ENCODER INCREMENTALE

In fig. 7.11 è riportata la soluzione con una rete sequenziale sincrona al problema del conteggio dei passi di spostamento dai segnali A e B di un encoder incrementale.
Si noti che la frequenza del Clock del circuito in figura dovrà essere maggiore del doppio della frequenza degli eventi, cioè maggiore di 8 volte la frequenza del segnale A (o, che è lo stesso di B). Solo così è garantita l'alternanza di riconoscimenti di commutazioni e di stati di riposo.
I registri a scorrimento, di due bit ciascuno, servono per presentare ai decodificatori i valori dei segnali A e B attuali e quelli in corrispondenza del clock precedente.
In figura non è rappresentato il contatore che deve essere di tipo *up-down*, con due distinti ingressi di conteggio, e dotato di un numero di bit adeguato al campo di conteggi da effettuare. Questo contatore dovrà essere provvisto anche di comando di azzeramento (clear) per effettuare le operazioni di *zero-macchina*.

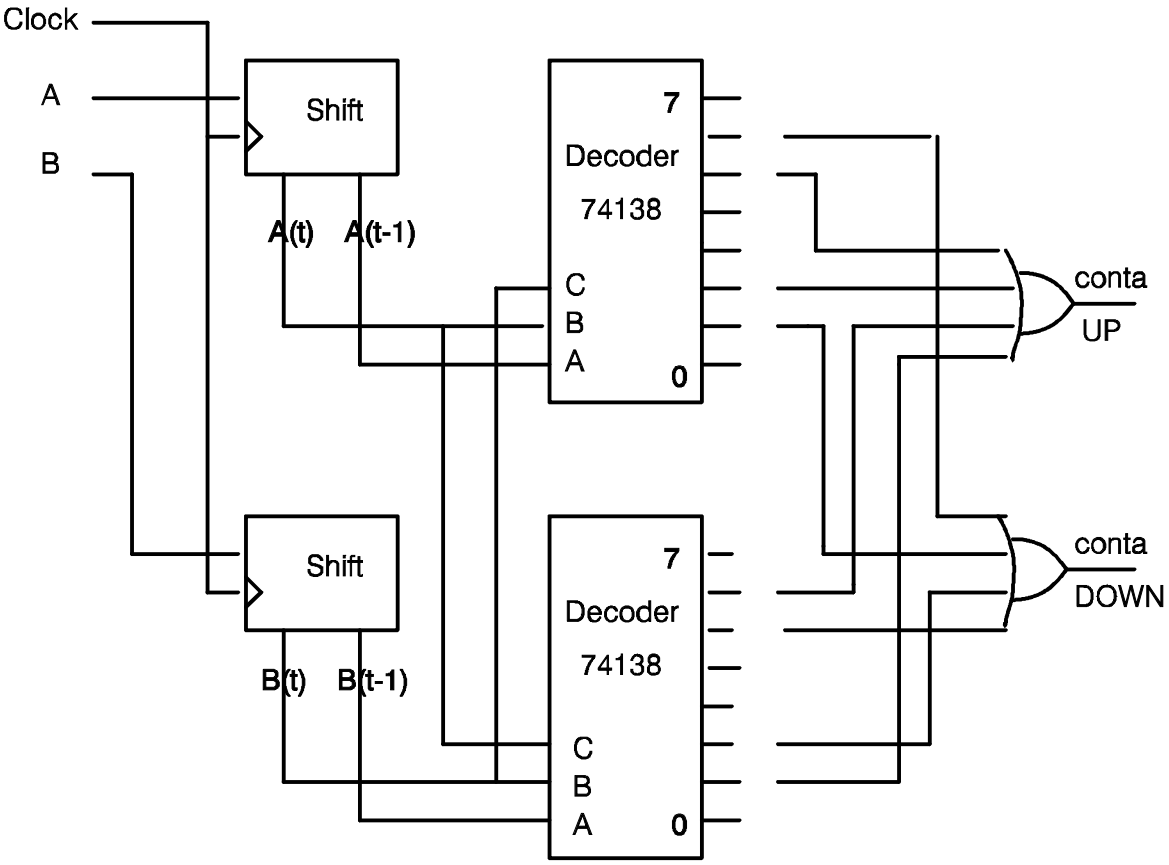


Fig. 7.11 - Esempio di gestione HW di encoder incrementale bidirezionale. Il contatore vero e proprio, di tipo *up-down*, non è rappresentato.

L'errore di quantizzazione che si commette con queste tecniche corrisponde alla granularità della grandezza rilevata (metri lineari o gradi angolari), tra due commutazioni successive dei segnali A o B.

7.2.1.2 Acquisizione di impulsi per temporizzazione (timer)

Questo punto è già stato trattato nel precedente cap.6, dedicato alle temporizzazioni nei calcolatori. Qui ci proponiamo di completare il discorso con l'analisi degli errori che si commettono nel rilievo del un tempo assoluto in cui si verifica un evento, destinato al

time-stamping dell'evento, e nella misura dell'intervallo di tempo tra due eventi, facendo riferimento ad una sezione TIMER come quella presentata nel punto 6.7.3.

a) - Time-stamping di eventi interni

Il valore di time-stamping **Tst** , rispetto al tempo vero **t** in cui si verifica l'evento soddisfa la relazione:

t-TIG+Tel < Tst < t+Tel

con

TIG = granularità temporale del timer

Tel = tempo di esecuzione della lettura del timer,

e quindi l'errore temporale assoluto **Eatst** vale:

Eatst = [-(TIG-Tel) . .+Tel]

N.B. Tra parentesi quadre è riportato l'intervallo di valori [min . . max].

b) - Time-stamping di eventi esterni

Il valore di time-stamping **Tst** , rispetto al tempo vero **t** in cui si verifica l'evento soddisfa la relazione:

t-TIG+Trev+Tel < Tst < t+Tlmax+Trev+Tel

con

TIG = granularità temporale del timer

Trev = tempo di esecuzione riconoscimento dell'evento,

Tel = tempo di esecuzione della lettura del timer

Tlmax = massima latenza della percezione (a polling o interrupt) dell'evento esterno,

e quindi l'errore temporale assoluto **Eatst** vale:

Eatst = [-(TIG-Trev-Tel) . .+(Tlmax+Trev+Tel)]

c) - Misura di intervallo tra due eventi esterni

La misura di un intervallo di tempo è ottenuta per differenza tra i due valori di time-stamping associati ai due eventi, come nel precedente punto b), e quindi, considerando costanti i tempi Trev e Tel, gli estremi dell'errore assoluto **Eam** sono

Eam = [-(TIG+Tlmax) . .+ (TIG+Tlmax)]

Si notino i contributi di granularità e latenza massima. In particolare quest'ultima, per misure con elevati requisiti di precisione, dovrà essere limitata il più possibile con i seguenti accorgimenti:

-- con rilievo degli eventi a *polling* si adotterà un periodo di polling il più breve possibile, compatibilmente con altre considerazioni di efficienza e carico di lavoro della CPU;

-- con rilievo degli eventi ad *interrupt* si adotteranno interrupt ad elevata priorità.

7.2.1.3 ACQUISIZIONE DI IMPULSI PER CALCOLO DI VELOCITA' (FREQUENZA)

In alcuni casi l'informazione che interessa è la frequenza degli impulsi che può, come nel caso degli encoder incrementali, corrispondere ad una velocità (lineare o angolare), a meno di un opportuno fattore di scala.

Si tratta di un problema tipico e che merita una particolare attenzione. Analizziamo quindi le principali scelte di progetto e le corrispondenti prestazioni temporali e di

precisione numerica ottenibili, in modo da essere in grado di valutare i margini di realizzabilità delle specifiche del problema.

DATI DEL PROBLEMA

I **dati del problema** sono tipicamente (o riconducibili a) i seguenti.

- Numero di impulsi/metro (o impulsi/giro) forniti dall'encoder, che è l'inverso della sua granularità *Iu.*

  Questo dato determina il fattore di scala per il calcolo della misura in unità ingegneristiche e gli errori di quantizzazione "spaziale".
- Campo (range) di frequenze *f* oggetto della misura, cioè *fmin* e *fmax,* da non confondere con la frequenza con cui si acquisisce tale misura.
- Durata temporale minima degli impulsi *Dmin,* ricavabile in genere da *fmax*

  Questo dato può condizionare le tecniche di rilievo degli eventi *impulso*.

SPECIFICHE

Le **specifiche** per una misura dinamica di velocità, ricavate in base a varie considerazioni applicative, richiederanno che si ottengano le seguenti prestazioni.

- **Frequenze** di produzione delle misure (assimilabili a frequenze di campionamento) maggiori di una frequenza minima accettabile *fc*.

Come è noto tale frequenza deve essere correlata con le caratteristiche dinamiche dei fenomeni da misurare ed eventualmente controllare.

- **Ritardi di acquisizione**, cioè ritardo tra l'istante in cui una misura è disponibile e l'istante a cui tale misura si riferisce.

Questi ritardi si aggiungono ad eventuali altri ritardi di elaborazione, ad esempio per la chiusura di anelli diregolazione.

- **Errori di misura** (cioè scarti rispetto al valore vero) assoluti o relativi inferiori in modulo ad un certo valore assegnato *Ea* o *Er*.

Gli errori assoluti sono specificati nelle unità di misura della grandezza da misurare, mentre gli errori relativi (generalmente espressi in %) devono essere specificati come rapportati al valore della singola misura (Erm%) o come rapportati al valore di fondo scala (Erf%). Si noti che errori assoluti anche piccoli ma commessi su misure di valore ridotto (ad es. nell'inizio scala) possono corrispondere ad errori percentuali Erm% elevati. Si noti anche che nei problemi ingegneristici si specificano in genere errori relativi, mentre la specifica di errori assoluti è riservata ai casi di grandezze con *origine convenzionale,* come spazio, tempo, ecc.

TECNICHE DI ACQUISIZIONE

Il rilievo di una frequenza può essere effettuato in uno dei seguenti modi tra cui il progettista dovrà scegliere:

1) Conteggio degli impulsi durante un intervallo di tempo prefissato e noto.

2) Rilievo del periodo tra due (o N) impulsi consecutivi e calcolo del suo inverso.

In entrambi i casi si dovranno gestire in modo coordinato due sequenze di eventi: gli eventi costituiti dagli impulsi in esame e gli eventi temporali generati dal clock di rilievo del tempo. Gli errori che si commettono con queste tecniche dipenderanno quindi dalla quantizzazione intrinseca nel conteggio e dagli errori temporali. Come

vedremo nelle analisi seguenti questi errori assumono un peso diverso in base alla tecnica utilizzata.

Alcune **note a proposito di errori**.

- Nelle analisi di tecniche che vengono presentate nel seguito verranno ricavate le espressioni degli errori massimi in valore assoluto, sia per difetto (Einf) che per eccesso (Esup). Non vi è in generale simmetria dei casi peggiori positivi e negativi.
- Gli errori assoluti hanno le dimensioni della grandezza misurata e sono calcolati come
  V*alore Misurato - Valore Vero*
- Gli errori **relativi alle misure** sono adimensionali e sono ottenuti dividendo gli errori assoluti per il valore della misura.
- Non si entra in dettagli sulla distribuzione probabilistica degli errori, notiamo però brevemente che gli errori di **quantizzazione** hanno distribuzione uniforme, mentre gli errori dovuti alle **latenze** possono avere distribuzioni anche molto variabili al variare delle applicazioni e anche dinamicamente nell'ambito di una stessa applicazione, al variare del carico di lavoro della CPU.

Come si è detto, dobbiamo rilevare due tipi di eventi: impulsi esterni ed eventi temporali. E' quindi opportuno un richiamo sulle diverse tecniche adottabili per rilevare eventi.

A) - Software.
Si legge ciclicamente con periodo *Tc* (che costituisce quindi anche la latenza max. di rilievo degli eventi) lo stato del segnale e dal confronto con il valore precedente si deduce la presenza o meno di un fronte di salita o di discesa. Consideriamo trascurabile (rispetto a Tc) il tempo di questo rilievo dell'evento, basato sull'esecuzione di poche istruzioni macchina.
L'errore temporale *Et* che si commette, cioè il ritardo tra l'evento e l'istante in cui esso è rilevato, è compreso tra gli estremi $0 < Et < Tc$.
Tenendo conto anche della granularità temporale *TIG* dell'orologio, gli estremi diventano $-TIG < Et < Tc$.

B) - Interrupt.
Il segnale viene collegato ad un morsetto di richiesta di interrupt, con le tecniche viste precedentemente. Nell'ambito della routine di risposta all'interrupt vengono svolte le azioni dovute, di conteggio o di campionamento.
L'errore temporale in questo caso dipende dalle due componenti:
Tli = latenza della risposta all'interrupt con estremi $0 < Tli < Tlimax$
Te = tempo di esecuzione delle istruzioni nella risposta all'interrupt, che è piuttosto regolare.
L'errore temporale, compreso l'errore di quantizzazione, è quindi
$-TIG+Te < Et < Te+Tlimax$.

C) - Hardware.
I rilievi di eventi sono effettuati da appositi circuiti e gli errori temporali, relativamente costanti, sono di pochi (decine di) nanosecondi ed in molti casi risultano quindi trascurabili rispetto agli errori di quantizzazione, per cui l'errore temporale è
$-TIG < Et < 0$.

In linea di principio si possono adottare tutte le 9 combinazioni possibili delle tre tecniche appena citate, a due a due per gli **impulsi** (I-SW, I-INT, I-HW) e per il **tempo** (T-SW, T-INT, T-HW). Nel seguito vengono presentati gli abbinamenti piu' tipici, lasciando come esercizio al lettore l'analisi delle altre combinazioni.

Innanzittutto un'osservazione: non è possibile, con tecniche digitali SW o HW che siano, il rilievo della frequenza istantanea, ma solo di quella media in un certo intervallo. Naturalmente quanto più breve sarà l'intervallo considerato tanto più si potrà confondere la frequenza media con quella istantanea, attribuendo il valore così ottenuto all'istante finale dell'intervallo di rilievo. Rimane comunque un effetto di "ritardo", nel rilievo della misura, ineliminabile e, per quanto detto tra poco, spesso piuttosto rilevante. In genere il ritardo equivalente è assunto pari a metà del periodo di acquisizione.

Nelle analisi di alcune tecniche implementative che vengono svolte nel seguito, sono riportati i seguenti aspetti significativi per un buon progetto.

VINCOLI

Rappresentano relazioni da rispettare tra il campo di frequenze [fmin . . fmax] misurabili e i parametri di progetto come frequenza di campionamento (*fc*), periodi di ciclo SW (*Tcic*), intervalli di tempo (*DT*) della misura, ecc.

ERRORI

Vari contributi agli errori di misura e ampiezza dell'intervallo di incertezza della misura in funzione della frequenza *f* da misurare.

COMPROMESSI

Sintesi dei vantaggi (pro) e svantaggi (con) che si ottengono incrementando o decrementando i valori dei parametri di progetto su cui è possibile agire.

1) - Conteggio di impulsi in un intervallo prefissato DT.

Con questo approccio si assume generalmente che il periodo Tc di acquisizione delle misure sia Tc = DT. Il vincolo imposto dalla tecnica di misura è comunque Tc ≥ DT.

- **I-SW--T-SW - Conteggio SW degli impulsi e rilievo SW del tempo.**

Gli impulsi sono presentati su un bit di ingresso (I_bit) e un'onda quadra di *periodo* DT su un altro bit di ingresso (T_bit).
Con periodo Tcic viene ciclicamente eseguita una routine schematizzabile come segue:

```
rileva_velocita:
     lettura porta input
     if (fronte salita I_bit)
          contatore++
     if (fronte salita T_bit)
          velocita = Iu * contatore / DT
          contatore = 0
     ritorna
```

VINCOLI

Tcic < Dmin (durata min impulsi) da cui $f \cdot Tcic < \frac{1}{2}$

cioè $fmax < fcic / 2$

fc = 1 / DT (freq. "campionamento")

fmin > 1 / DT (minima frequenza rilevata come ≠ 0

ERRORI

con *np* = numero (reale) di periodi nell'intervallo *DTM*
DTM = intervallo effettivo di conteggio

quantizzazione su contatore	[-1...+1] impulso	$np-1 < n < np+1$
errore temporale su DT	[-Tcic...+Tcic]	$DT-Tcic < DTM < DT+Tcic$

frequenza vera $f = np / DTM$

frequenza misurata $fm = n / DT$

Rispetto alla frequenza vera *f*, le misure *fm* possono variare **nell'intervallo di incertezza**:

$$\frac{f \cdot (DT - Tcic) - 1}{DT} < fm < \frac{f \cdot (DT + Tcic) + 1}{DT}$$

da cui, con semplici calcoli lasciati al lettore si ricavano i valori inferiore e superiore degli **errori assoluti** in Hz:

$$Ea_{\inf} = -\frac{1 + f \cdot Tcic}{DT} \qquad Ea_{\sup} = \frac{1 + f \cdot Tcic}{DT}$$

Gli stessi valori moltiplicati per il coefficiente di scala *Iu* danno gli errori assoluti nelle unità di misura della velocità.

COMPROMESSI

DT - Aumentando DT

pro - si riducono gli errori e si abbassa il limite inferiore di frequenze misurabili con errore relativo accettabile

con - si diminuisce la frequenza di campionamento e si aumenta il ritardo di misura

Tcic - Diminuendo Tcic

pro - si riducono gli errori e aumenta la massima frequenza misurabile

con - aumenta il carico di lavoro della CPU

- **I-HW--T-INT - Conteggio HW degli impulsi e tempo rilevato ad interrupt.**

Si inviano gli impulsi ad un contatore HW di N bit il cui valore sia leggibile tramite porte di ingresso. Gli eventi temporali di inizio e fine intervallo DT generano interruzioni (con latenza massima Tlimax), durante il servizio delle quali si legge il contenuto del contatore e si calcola la nuova velocità.

Routine di risposta all'interrupt dei tempi.

```
salva contesto
temp = leggi_contatore_impulsi
cont = temp - prec        ; conteggio nel periodo
freq = cont / DT          ; calcolo della frequenza
prec = temp
nuovo_valore = TRUE       ; indicatore nuova misura
ripristina contesto
IRET                      ; ritorno da interrupt
```

VINCOLI

$f_{\max} \cdot DT < 2^N$	per non avere overflow del contatore HW
fc = 1 / DT	(freq. "campionamento")
fmin > 1 / DT	(minima frequenza rilevata come $\neq 0$

ERRORI

quantizzazione su contatore	[-1...+1] impulso
errore temporale su DT	[-Tlimax...+Tlimax]

Poichè in questo caso durante le latenze della risposta all'interrupt il contatore HW continua a contare gli impulsi anche dopo lo scadere del tempo DT, gli estremi del valore di frequenza calcolata *fm* sono:

$$\frac{f \cdot (DT - Tli_{\max}) - 1}{DT} < fm < \frac{f \cdot (DT + Tli_{\max}) + 1}{DT}$$

Gli **errori assoluti** della misura calcolata sono quindi dati dalle seguenti espressioni:

$$Ea_{\inf} = -(\frac{1 + f \cdot Tli_{\max}}{DT}) \qquad Ea_{\sup} = \frac{1 + f \cdot Tli_{\max}}{DT}$$

Si noti che per gli **errori relativi alla misura**, ottenuti dividendo per *f* le espressioni dei valori assoluti, il caso peggiore si ha per *f = fmin*.

$$Erm_{\inf} = -(\frac{1}{f_{\min} \cdot DT} + \frac{Tli_{\max}}{DT}) \qquad Erm_{\sup} = \frac{1}{f_{\min} \cdot DT} + \frac{Tli_{\max}}{DT}$$

COMPROMESSI

DT - Aumentando DT

pro - si riducono gli errori e si abbassa il limite inferiore di frequenze misurabili con errore relativo accettabile

con - si diminuisce la frequenza di campionamento e si aumenta il ritardo di misura

Tli - Diminuzione di Tli

pro - si riducono gli errori

con - non ci sono aspetti negativi, se non la maggior difficoltà di progetto e il fatto che la riduzione di latenza ottenuta incrementando la priorità di questo interrupt probabilmente comporta un aumento di latenza per altri interrupt.

- **I-HW--T-HW - Conteggio HW degli impulsi con temporizzazione HW.**

Gli impulsi sono forniti ad un contatore di N bit, dotato di ingressi di *enable* e di *reset*, e leggibile da porte di ingresso. Il periodo DT è generato da un secondo contatore di M bit con clock di periodo Tck = DT/K (K intero), dotato di registro per il valore di fine conteggio, ingressi di *enable* e *reset*. Il circuito si semplifica se si può sceglere $K = 2^P$ con P < M. Questo timer gioca il ruolo "C" presentato nel paragrafo 6.1.3.
Allo scadere del periodo DT si genera un interrupt per consentire al software di acquisire i conteggi per il calcolo della velocità e per attivare l'acquisizione seguente con un comando di *reset* ai contatori. Si noti che i ritardi della risposta all'interrupt non introducono errori, ma riducono solo (un poco) la frequenza fc di campionamento.
In fig. 7.12 è riportato lo schema circuitale di massima di questo approccio, assumendo K=128 (cioè P=7) ed M=8.

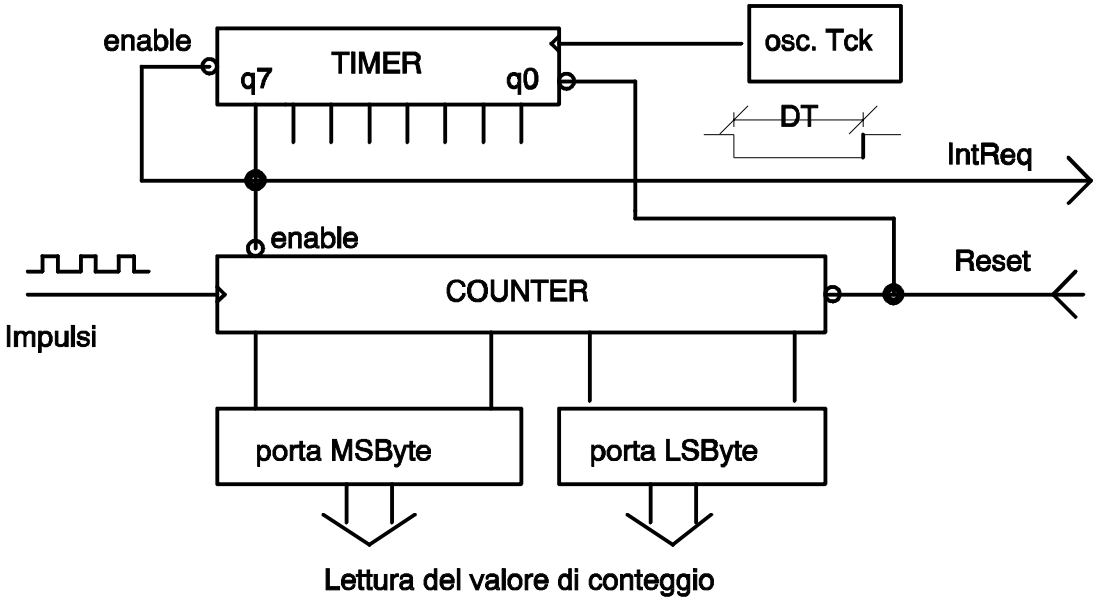


Fig. 7.12 - Circuito di interfaccia per l'approccio I-HW--T-HW alla misura di frequenza di impulsi in ingresso.

VINCOLI

$f_{\max} \cdot DT < 2^N$	per non avere overflow del contatore
$fc = 1/(DT + Tli_{\max})$	frequenza di campionamento

ERRORI

quantizzazione su contatore	[-1...+1] impulso
quantizzazione temporale su DT	[-Tck...+0]

Rispetto alla frequenza vera *f*, le misure *fm* possono variare nell'intervallo:

$$\frac{f \cdot (DT - Tck) - 1}{DT} < fm < \frac{f \cdot DT + 1}{DT}$$

Gli **errori assoluti** della misura calcolata sono dati dalle seguenti espressioni:

$$Ea_{\inf} = -\frac{f \cdot Tck + 1}{DT} \qquad Ea_{\sup} = \frac{1}{DT}$$

e gli **errori relativi alla misura**, per cui il caso peggiore si ha con *f* = *fmin*:

$$Erm_{\inf} = -(\frac{Tck}{DT} + \frac{1}{f_{\min} \cdot DT}) \qquad Erm_{\sup} = \frac{1}{f_{\min} \cdot DT}$$

N.B. Nell'esempio circuitale della figura si ha DT = 128*Tck

COMPROMESSI

DT - Aumentando DT

pro - si riducono gli errori e si abbassa il limite inferiore di frequenze misurabili con errore relativo accettabile

con - si diminuisce la frequenza di campionamento e si aumenta il ritardo di misura

Tck - Diminuendo Tck

pro - a parità di DT si riducono gli errori

con - a parità di DT occorre un timer con più bit

NOTE SUL CONTEGGIO DI IMPULSI DURANTE UN INTERVALLO DT

- La modalità tipica nella realizzazione di questa tecnica consiste nel generare un interrupt allo scadere dell'intervallo di tempo di conteggio e leggere il valore del contatore di impulsi. Un contatore di impulsi HW ed un meccanismo di abilitazione al conteggio parimenti HW riducono a valori trascurabili le incertezze che altrimenti si avrebbero a causa dei ritardi temporali di risposta all'interrupt.

- Per avere una buona risoluzione (piccoli errori di quantizzazione) si deve adottare un intervallo di tempo *DT* molto maggiore del periodo degli impulsi, quindi piuttosto lungo per misure di basse frequenze.

- Gli errori di quantizzazione sono *costanti in valore assoluto* al variare della frequenza, quindi percentualmente molto elevati alle basse frequenze.

L'errore di quantizzazione assoluto rispetto alla frequenza *f* da misurare vale

$$Eq = \frac{1}{DT}$$

dove Eq è l'errore di quantizzazione max. in Hz

DT è la durata dell'intervallo di conteggio

- La situazione di frequenza nulla (o prossima allo 0) non costituisce un aspetto critico dell'algoritmo (a parte gli errori), ma occorre considerare le specifiche di risoluzione sulla frequenza e quindi la scelta del periodo di conteggio *DT* che determina quale frequenza minima è rilevata come diversa da zero ad ogni acquisizione.

2) - Misura dell'intervallo tra due impulsi consecutivi.

Con questo approccio gli eventi esterni e gli eventi temporali si scambiano i ruoli rispetto ai casi precedentemente analizzati. Infatti sono gli eventi esterni a delimitare il periodo durante il quale si effettua il conteggio di impulsi temporali prodotti da un clock di periodo Tck.

- **I-INT--T-HW - Gli impulsi generano interrupt, si legge il tempo contato HW**

Il tempo è rappresentato da un contatore di N bit, che conta in modalità *free-running* un clock di periodo Tck, mentre gli impulsi sono presentati ad un morsetto di richiesta di interrupt, possibilmente del tipo sensibile al fronte (*edge sensitive*). Nella routine di

risposta all'interrupt viene acquisito il tempo ed eseguito il calcolo della velocità, come indicato sinteticamente nel seguito.

Routine di risposta all'interrupt degli impulsi.

```
salva contesto
temp = leggi_contatore_tempo
DT = temp - prec          ; durata del periodo
freq = 1 / DT             ; calcolo della frequenza
prec = temp
nuovo_valore = TRUE       ; indicatore nuova misura
ripristina contesto
IRET                      ; ritorno da interrupt
```

VINCOLI

$Tck \cdot 2^N > 1/f_{\min}$	per evitare il giro completo (overflow) del contatore dei tempi
fc = fmin	frequenza di campionamento minima garantibile
fmax < 1 / Tck	per evitare divisione per 0

ERRORI

quantizzazione dei tempi	[-Tck...+Tck]
temporale sugli impulsi	[-Tlimax...+Tlimax]

Gli estremi per i valori di frequenza *fm* calcolata sono

$$\frac{f}{1+f\cdot(Tck+Tli_{\max})} < fm < \frac{f}{1-f\cdot(Tck+Tli_{\max})}$$

Gli **errori assoluti** che si commettono sono:

$$Ea_{\inf} = -f\cdot\frac{f\cdot(Tck+Tli_{\max})}{1+f\cdot(Tck+Tli_{\max})} \qquad Ea_{\sup} = f\cdot\frac{f\cdot(Tck+Tli_{\max})}{1-f\cdot(Tck+Tli_{\max})}$$

mentre gli **errori relativi**, sono facilmente calcolabili dividendo per *f* gli errori relativi, e presentano il caso peggiore per *f = fmax*.

Si noti che al diminuire del periodo degli impulsi (alle alte frequenze) avvicinandosi a *Tck+Tlimax* l'errore relativo inferiore arriva al 50%, mentre quello superiore tende ad infinito.

COMPROMESSI

Tck - Diminuendo Tck

pro - si riducono gli errori e aumenta la massima frequenza rilevabile con errori accettabili.

con - occorre un contatore del tempo (timer) con più bit

Tli - Diminuzione di Tli

pro - si riducono gli errori e aumenta la massima frequenza rilevabile con errori accettabili.

con - non ci sono aspetti negativi, se non la maggior difficoltà di progetto e il fatto che la riduzione di latenza ottenuta incrementando la priorità di questo interrupt probabilmente comporta un aumento di latenza per altri interrupt.

NOTE SULLA MISURA DELL'INTERVALLO DT TRA DUE IMPULSI

- La modalità tipica per realizzare questa tecnica di misura consiste nel generare un interrupt ad ogni impulso e leggere il conteggio di tempo corrispondente. L'uso di meccanismi HW per attivare e arrestare il "cronometro" evita di introdurre le incertezze del valore di tempo letto dovute al ritardo variabile di risposta all'interrupt.

- La situazione di frequenza (quasi) nulla, cioè di periodo (quasi) infinito costituisce un punto **critico** (per l'algoritmo, non per gli errori di misura) e va gestita con opportuni "time-out", per evitare di rimanere indefinitamente in attesa del prossimo impulso, senza produrre il valore di misura f = 0.

- La cadenza con cui sono effettuabili i rilievi non è costante ma è tanto più rapida quanto più elevata è la frequenza: ciò in alcuni casi può essere vantaggioso, ma complica le operazioni rispetto ai casi di campionamento a cadenza regolare.
Alle basse frequenze la cadenza di acquisizione è lenta.

- La risoluzione relativa delle misure calcolate dipende dalla granularità temporale del "cronometro", ed è comunque massima (cioè più fine) alle frequenze minime mentre si riduce (cioè diventa più grossolana) al crescere della frequenza. Esiste una frequenza massima oltre la quale l'errore di quantizzazione supera la soglia di accettabilità.

- Un compromesso si ottiene misurando l'intervallo di tempo necessario per contare N impulsi, con N dinamicamente variabile in dipendenza della frequenza da misurare.
Si adottano valori di N crescenti al crescere della frequenza "attesa".

In generale le situazioni più critiche si hanno quando si debba coprire un ampio campo di frequenze, con valori che si estendono verso le basse frequenze, e sia richiesto un basso errore di quantizzazione ed una elevata frequenza di "campionamento".

7.2.1.4 Acquisizione di impulsi per sincronizzazione

In alcuni casi gli impulsi costituiscono solo elementi di sincronizzazione, rappresentano cioè gli eventi in corrispondenza dei quali è valida una nuova informazione ad essi associata.
In genere si precisa se è da considerarsi evento significativo il fronte di salita o quello di discesa; tale fronte dovrà attivare la funzione di "*latch*" del registro che costituisce la porta di **ingresso** dati destinato a memorizzare il valore presente in quel momento ai morsetti esterni, e dovrà provocare l'opportuna transizione nei bit della porta di stato, per indicare “evento catturato”.

Gli impulsi di sincronizzazione vengono utilizzati anche quando si devono *emettere* informazioni di tipo "evento".
In questo caso l'impulso fornito da un dispositivo esterno al calcolatore agirà sugli indicatori di stato (*unità periferica pronta a ricevere*), mentre il comando di *latch* del valore sul registro della porta di uscita verrà fornito dalla successiva operazione di scrittura da parte del calcolatore che, in molti casi dovrà essere effettuata entro limiti temporali prefissati. L'impulso in ingresso fungerà cioè da stimolo per la produzione di informazioni in uscita.

7.2.2 EMISSIONE DI IMPULSI

Frequenza, duty-cycle, jitter

In molte applicazioni è necessario generare in uscita degli impulsi. I principali aspetti da analizzare e risolvere in sede di progetto riguardano la **frequenza** (PFM = Pulse Frequency Modulation) e la **durata** (PWM = Pulse Width Modulation) degli impulsi e le relative imprecisioni temporali.

7.2.2.1 Generazione di impulsi PFM

Questo problema si pone quando si debbano fornire segnali a **dispositivi esterni sensibili alla frequenza** degli impulsi.
Un esempio tipico è costituito dal comando di motori "a passo", caratterizzati dal fatto che ad ogni impulso compiono una rotazione corrispondente ad una ben precisa frazione di giro.
Questi motori presentano coppie e potenze limitate e bassissima efficienza energetica, ma sono molto diffusi in macchine in cui il controllo di posizione con tecniche digitali costituisce l'aspetto preponderante, come ad esempio per il movimento del carrello delle stampanti, il movimento radiale della testina magnetica nei "drive" di dischetti (floppy), ma anche in applicazioni di robotica e di macchine utensili.

E' facile notare che il comportamento dinamico del motore con relativa trasmissione e carico non può accettare impulsi di comando forniti in istanti arbitrari, pena la "perdita di passi" che comporta discrepanza tra posizione voluta e posizione reale.

In questo caso i requisiti sono:

- - Durata degli impulsi dettata esclusivamente dalla necessaria circuiteria elettronica di potenza che controlla le correnti in gioco negli avvolgimenti del motore.

- - Frequenza di "start/stop" che rappresenta la massima frequenza fornibile a motore fermo perchè esso inizi a ruotare, o da cui il motore può essere fermato immediatamente, senza perdere passi.
- - Rampa di accelerazione (e decelerazione) per raggiungere velocità superiori a quella di start/stop.
- - L'irregolarità del periodo tra impulsi consecutivi (detta "*jitter*") può causare instabilità con vibrazioni e perdita di passi.

I requisiti temporali e di controllo per la generazione di impulsi PFM sono in genere abbastanza severi da richiedere l'uso di funzionalità HW o di processori dedicati.

7.2.2.2 Generazione di impulsi PWM

In alcuni casi si ha a che fare con **sistemi esterni sensibili al valor medio del segnale** loro fornito.
Un esempio tipico è costituito da alcuni sistemi di controllo di temperatura o in genere di fenomeni a dinamica "lenta".

Consideriamo un segnale costituito da impulsi di ampiezza 'A' con periodo costante 'P' e durata variabile $0 < D(t) < P$.
Il **valor medio** di tale segnale è dato da:

$$Vm(t) = A * (D(t) / P)$$

e se chiamiamo DC(t) (*Duty-cycle*) il rapporto D(t)/P:

$$Vm(t) = A * DC(t)$$

cioè il valor medio del segnale è proporzionale al suo *duty-cycle*.

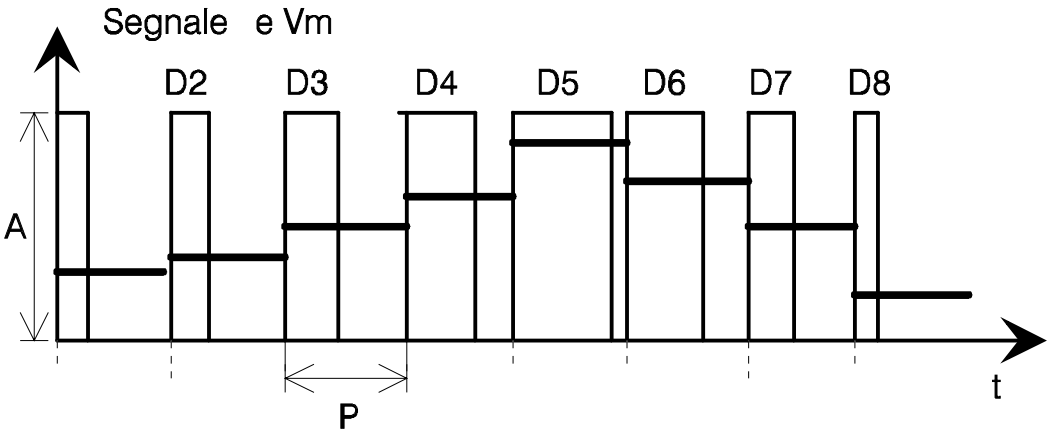


Fig. 7.13 - Esempio di impulsi Di modulati in ampiezza (PWM - *Pulse Width Modulation*) e relativo valor medio (tratto in neretto) in ogni periodo.

L'errore di quantizzazione *relativo* sul valore Vm vale:

$$Eqr = Pck / P$$

dove *Pck* è la granularità temporale con cui siamo in grado di controllare la durata D.

E' interessante osservare che agli effetti di un sistema ad elevata inerzia, cioè con costanti di tempo molto maggiori di P, siamo in grado di produrre un'informazione di tipo analogico pur con la più semplice generazione di un segnale a due soli livelli (ON/OFF), semplicemente controllandone le durate D.

Naturalmente c'è uno scotto da pagare ed è costituito dalla (non grande) difficoltà di controllare il duty-cycle. Se P è abbastanza grande (P > 0.1 secondi) e la precisione richiesta non è eccessiva si può adottare per la generazione di impulsi PWM una gestione completamente SW, mentre per situazioni più difficili occorre affidare alcune funzionalità a circuiti (appositi).

7.2.2.3 Generazione di impulsi per parzializzazione di onde

Un'altra interessante tecnica di controllo del valor medio (o del valore efficace) di un segnale consiste nel "parzializzare" semionde di una tensione sinusoidale mediante diodi controllati (SCR o Triac) "accesi" in istanti di tempo opportunamente ritardati rispetto all'istante di passaggio per lo zero della tensione di alimentazione (fig. 7.14). Questa tecnica, interessante per il controllo di potenze anche elevate, tende a generare disturbi impulsivi che possono richiedere appropriati accorgimenti (*filtri di rete*) a protezione dell'alimentazione dei calcolatori.

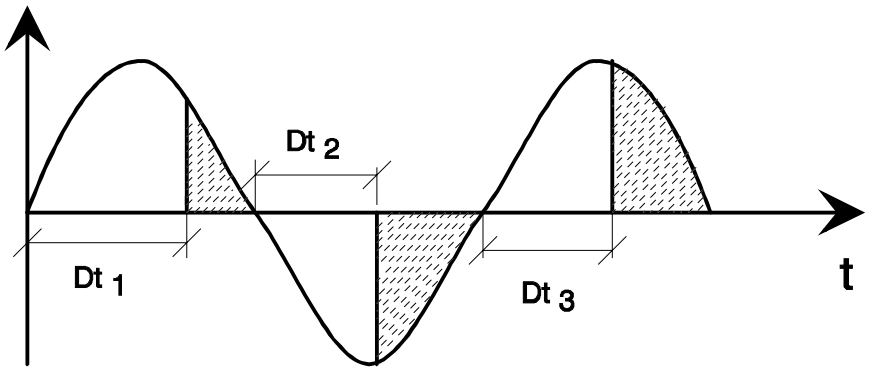


Fig. 7.14 - Parzializzazione di semionde con accensione di un TRIAC dopo tempi Dt(t) dai passaggi per lo zero dell'onda. Le aree tratteggiate corrispondono ai periodi di conduzione di corrente elettrica sul carico.

La tecnica della parzializzazione richiede di recepire gli eventi "passaggio per lo zero" ed emettere *impulsi di comando* dopo un *ritardo* opportunamente calcolato. Occasionali perdite di eventi in questo caso non costituiscono un dramma, ma la risoluzione temporale richiesta (una semionda dura 10 ms) suggerisce di risolvere a livello HW la gestione dei tempi, mentre rimane in genere vantaggiosamente affidata al SW la funzione di calcolo dei ritardi Dt(t) di accensione che, si noti, non presentano una relazione lineare con il valor medio da generare, come avviene per le onde quadre modulate in durata (PWM) viste precedentemente.

7.3 VALORI CODIFICATI

Finora abbiamo considerato informazioni associate a singoli segnali digitali. Come è noto le informazioni che possono assumere 2^N diversi valori possono essere codificate con N (o più) bit a loro volta rappresentati da altrettanti segnali binari.
L'aspetto tipico di questi casi è costituito dal fatto che i segnali non convogliano singolarmente informazioni utili, ma è solo il loro **insieme ordinato** che rappresenta informazioni significative.

Uno degli aspetti critici per la correttezza riguarda quindi la (a rigore irrealizzabile) simultaneità di trasferimento di tutti i bit che rappresentano un'unica informazione.

Mentre si rimanda ad altre letture per l'analisi delle soluzioni tipicamente adottate nell'ambito delle comunicazioni "parallele" e "seriali", che si propongono di trasferire informazioni associate ad eventi (caratteri e bit), nel seguito consideriamo soprattutto i problemi di acquisizione ed emissione di informazioni di stato.

Rimandiamo poi al paragrafo seguente anche la trattazione della gestione delle informazioni di stato analogiche che pure, in una certa "sezione" dell'interfaccia, presentano problemi analoghi a quelli che andiamo considerando.

Nel seguito considereremo "numeriche" le informazioni da trasferire, indipendentemente dal fatto che esse rappresentino effettivamente numeri o generici valori enumerativi.

7.3.1 ACQUISIZIONE DI CODICI NUMERICI

L'acquisizione di valori numerici pone i seguenti problemi.

- Le commutazioni dei vari bit nel passaggio da uno stato all'altro non sono contemporanee e quindi, per sia pur brevi intervalli di tempo, si presentano configurazioni che possono essere anche totalmente non significative. Poco probabili ma possibili operazioni di lettura effettuate durante quegli intervalli possono portare ad "immortalare" quelle configurazioni non significative.

- Le informazioni che richiedono per la loro rappresentazione più bit di quanti possano essere trasferiti simultaneamente (cioè appartenere ad una singola porta elementare) devono essere acquisite in fasi successive, tra l'una e l'altra delle quali possono avvenire commutazioni che portano all'incoerenza dell'informazione complessiva.

7.3.1.1 COMMUTATORI DECADICI

Un tipico dispositivo per acquisizione di codici numerici forniti da un operatore è costituito da uno o più commutatori decadici (o talora esadecimali) detti anche "Contraves". Ogni elemento ("frutto") chiude o apre contatti su 4 linee in modo da presentare la codifica BCD (Binary Coded Decimal) o esadecimale corrispondente alla cifra che l'operatore vede in un'apposita finestrella presente nel dispositivo stesso.

Le informazioni da acquisire sono tipicamente "stazionarie", ma durante le rare commutazioni può verificarsi "di tutto":
- strisciamenti e "rimbalzi" dei contatti
- commutazioni irregolari nel tempo, dei singoli bit
- ogni cifra passa attraverso valori non significativi, prima di raggiungere la posizione definitiva
- le cifre vengono "mosse" con la sequenza casualmente scelta dall'operatore
- le cifre possono essere numerose e richiedere quindi diverse istruzioni di INPUT in sequenza, per la lettura completa del valore numerico.

Si presenta cioè un tipico problema di superamento di disomogeneità attitudinale che, a seconda delle specifiche può essere risolto in uno dei seguenti modi.

- - Si accettano le temporanee incongruenze se gli effetti da ciò derivanti sono altrettanto temporanei (non ci sono corse critiche) ed accettabili.
- - Si adotta (tipicamente a livello SW) una tecnica di "ritardo all'accettazione" che consiste nell'accettare la codifica rilevata solo dopo che sia trascorso un tempo prefissato senza che si siano verificate modifiche nella codifica stessa.
- - Si decide di associare un evento (pulsante) o stato (interruttore) di "convalida", che l'operatore è tenuto ad utilizzare per rispettivamente segnalare quando ha impostato la configurazione voluta o disabilitare la lettura durante i transitori di impostazione.
  Si noti che questo evento o stato possono non essere esclusivamente dedicati alla convalida, ma possono costituire ingressi di comando per altre funzioni (ad es. avviamento di una macchina) alla cui attivazione, per specifica ipotesi, le impostazioni devono essere stabili e significative.

Naturalmente il valore acquisito in codice BCD dovrà essere convertito in forma binaria pura (`integer`) se i calcoli previsti lo richiedono, come è usuale se si programma con linguaggi di alto livello che non prevedono variabili di tipo BCD.

7.3.1.2 ENCODER ASSOLUTI

Un altro caso di stato digitale in ingresso è costituito dall'informazione fornita da particolari sensori di posizione detti "Encoder assoluti". Questi dispositivi sono solitamente di tipo rotante (raramente traslatori lineari) e forniscono su un certo numero di bit la codifica che rappresenta la posizione angolare (rispettivamente lineare) assunta dalla parte mobile.

Il numero di bit (in genere da 8 a 12) determina la risoluzione spaziale ed è limitato da problemi meccanici (e quindi di costo) relativi all'allineamento e alle dimensioni dei fotosensori che generano i vari bit.

Trattandosi di **stati** tipicamente **variabili**, i problemi di coerenza dei vari bit sono risolti alla fonte con l'adozione di **codifiche Gray** (talora Gray-BCD) caratterizzate, come è noto, dalla commutazione di un solo bit nel passaggio da uno stato a quello seguente o precedente.
In questi casi il problema tipico è costituito dalla necessità di trascodifica da codice Gray al codice Binario puro necessario per le elaborazioni interne. Nel seguito sono riportate le versioni in pseudocodice, Assembler e C di algoritmi di conversione Gray-

binario a livello SW, mentre in fig. 7.15 è presentato un circuito (espandibile ad un numero di bit a piacere) per la conversione ad HW.

ESEMPIO DI ALGORITMO DI CONVERSIONE GRAY-BINARIO

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

```
; Versione ASSEMBLER
; Routine che accetta in ingresso la variabile VALORE in codice GRAY su 12 bit
; ed esce con il contenuto convertito in binario puro.
; Riporta il valore convertito anche in AX

GRAY12_BIN:
        MOV   CX,12             ; contatore dei bit
        MOV   AX,800H           ; 1 su bit B11
        SUB   BX,BX             ; azzera BX
CICLO:
        XOR   [VALORE],BX       ; eventualmente commuta
        MOV   BX,[VALORE]
        AND   BX,AX
        SHR   AX,1
        SHR   BX,1
        LOOP  CICLO             ; decrementa CX e ricicla se non 0
        MOV   AX,[VALORE]
        RET
```

```
/* Versione in C */

int Gray12_bin (NumGray);
{
  int commuta,i,mask;
     valore = NumGray;
     i = 12;
     mask = 0x800;
     commuta = 0;
     while (i > 0)
     {
          valore ^= commuta;        // xor bit [i]
          commuta = valore & mask;
          commuta >>= 1;            // shift a destra
          mask >>= 1;
          i--;
     }
     return (valore);
}
```

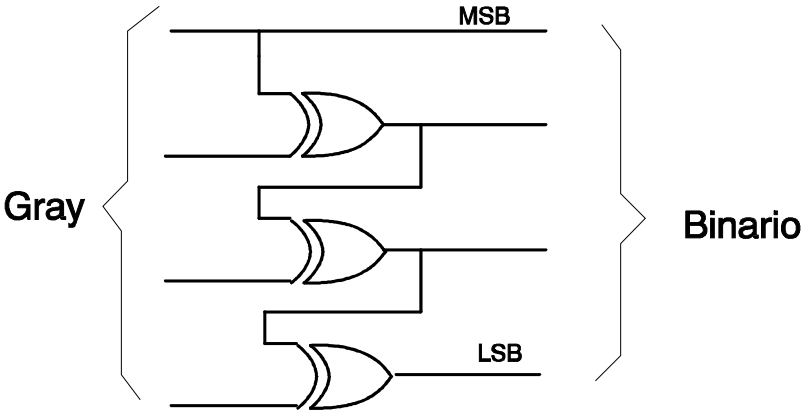


Fig. 7.15 - Conversione Gray-Binario su 4 bit ad hardware.

NOTA
Si noti che questo circuito di fig. 7.15 è riportato come esempio di realizzazione HW di un algoritmo, ma non verrà usato in pratica per la conversione di uno stato variabile in codice Gray in uno a codifica binaria.
Infatti la codifica binaria ottenuta in questo modo non rispetterebbe il requisito di commutazione di singoli bit che è poi il motivo per cui si adotta il codice Gray.
Il circuito può essere invece utile per la conversione di stati di cui sia nota la stazionarietà durante determinati periodi di tempo.

7.3.2 EMISSIONE DI CODICI NUMERICI

A parte eventuali conversioni di codifica, l'emissione di codici numerici può presentare il problema della contemporaneità di emissione di più bit di quanti emessi con una singola istruzione di "OUT".

La soluzione tipica consiste nel dotare l'interfaccia di una doppia serie di registri. Una prima serie, costituita dalle normali porte di uscita, viene scritta con una sequenza di istruzioni di scrittura in ordine arbitrario. Il contenuto di questi registri viene poi trasferito alla seconda serie, che effettivamente emette i segnali sui morsetti verso l'esterno, mediante un esplicito comando di trasferimento.
Il comando di trasferimento può essere generato contestualmente con la scrittura del byte più (o meno) significativo, che quindi dovrà essere l'ultimo ad essere scritto.

7.4 ESERCIZI

-- Discutere i problemi di scelta della frequenza del clock per il circuito rilevatore sincrono di eventi (par. 7.1.1.2).

-- Discutere il concetto di stato o evento spurio nel caso di ingresso ON/OFF da contatto con rimbalzi.

-- Discutere la scelta della frequenza di chiamata di una routine di filtraggio SW rispetto alle caratteristiche del segnale, degli spurii, e ai ritardi accettabili.

-- Che relazione deve valere tra la durata del transitorio di un segnale (ad es. rimbalzi) e il tempo morto del fenomeno che lo genera, per garantire l'eliminazione degli spurii, senza però perdere le informazioni utili?.

-- La routine AGGIORNA_CONTATORE riportata nel par. 7.2.1.1. è corretta anche nel caso che il contatore tra una lettura e la successiva subisca un overflow (passaggio per lo zero)?. Perchè?.

-- Perchè un contatore con più di 8 bit pone problemi per la sua lettura corretta?. Discutere le possibili soluzioni in termini di costo (HW) e tempo macchina (SW).

-- Qual'è la massima frequenza di conteggio per cui è corretta la routine LEGGI_CONTATORE del par. 7.2.1.1.?.

-- Qual'è la minima frequenza di chiamata della routine per acquisizione di impulsi da encoder incrementale?. Perchè si elidono gli eventi in caso di oscillazioni stazionarie?.

-- Con encoder incrementale da 360 impulsi/giro e gestione HW del conteggio, con clock di 100 Khz, qual'è la massima velocità angolare che non produce errori?.

-- Discutere gli errori temporali che si commettono nel rilievo di eventi a SW, interrupt e HW.

-- Perchè la misura del valore di frequenza media in un periodo DT è rilevata con un ritardo pari alla metà di tale periodo?.

-- Per ognuna delle tecniche di misura di frequenza (o velocità) presentate in questo capitolo, si discutano vincoli, errori e compromessi.

-- Si debba rilevare la velocità angolare di un motore a scoppio a 4 tempi e 4 cilindri, a partire dagli impulsi, a livello logico, ottenuti dallo spinterogeno (o, in generale, dal dispositivo che genera gli impulsi per le candele di accensione).
I regimi di rotazione previsti vanno da 600 a 6000 giri/minuto.
La frequenza di campionamento sia di almeno 10 Hz (misure al secondo).
Proporre uno schema di progetto, basato su un tipico microprocessore, e discuterne le prestazioni e gli errori.

-- Discutere il problema della frequenza minima rilevabile con la tecnica della misura del periodo (intervallo tra due impulsi).

-- Discutere la scelta di misurare il periodo sulla base di N impulsi consecutivi. Proporre un algoritmo che scelga dinamicamente il valore di N e discuterne i vantaggi.
Rilevare le analogie di questa tecnica con il caso di autoranging presentato nel cap. 8.

-- Proporre (sensate) tecniche per le seguenti misure di frequenze, discutendone la fattibilità,, la massima frequenza di campionamento fc e gli errori.

- fmin = 10 Hz fmax = 1 Khz
- fmin = 10 Khz fmax = 1 Mhz
- fmin = 1 Hz fmax = 100 Khz

-- Proporre un algoritmo di generazione SW di impulsi per motore a passo. Siano costanti note la frequenza di start/stop (50 Hz) e la massima accelerazione (100 Hz/s), mentre sia passato come parametro variabile il percorso da effettuare espresso in numero di passi. Ci si proponga di minimizzare il tempo di percorrenza.

-- Progettare un sistema di generazione di impulsi PWM in grado di fornire una frequenza di 10 Hz con un errore relativo sul valore medio di un millesimo.

7.5 ALTRE LETTURE

P.D. Lawrence, K Mauch
REAL-TIME MICROCOMPUTER SYSTEM DESIGN: AN INTRODUCTION
McGraw-Hill 1988