CAP. 5

5. PROBLEMATICHE DI INTERFACCIAMENTO

5.1 INTRODUZIONE

Il concetto di interfacciamento entra in gioco in tutti i casi in cui debbano interagire tra loro delle entità caratterizzate da una propria funzionalità che può essere attiva (es. CPU) o passiva (es. Memoria).

I **confini** che individuano le entità interagenti possono essere assunti con larga arbitrarietà ed in particolare dipendono dal livello di **astrazione** e **aggregazione** con cui si considera il sistema (livelli di modularità).

Tra le entità interagenti esistono in generale delle **disomogeneità**, il cui superamento costituisce **il problema centrale dell'interfacciamento**.

I problemi di interfacciamento sono ovviamente tanto più articolati quanto più marcate e variegate sono le disomogeneità da superare. Assumiamo quindi risolti da altre discipline e noti i problemi a livello dello stato solido (Fisica), a livello circuitale (Elettronica) e a livello strutturale (Teorie dei circuiti logici) e concentriamo ora la nostra attenzione sull'interazione tra **calcolatori e mondo esterno**.

Nel gergo informatico il calcolatore è considerato il punto di riferimento rispetto alla direzione dei flussi di informazioni; si parla quindi di ingressi ed uscite (I/O = Input / Output) rispetto al calcolatore, salvo che si stia considerando esplicitamente ed isolatamente un altro sottosistema.

Tutto ciò che non è **direttamente** governabile dall'unità centrale del calcolatore tramite i segnali del bus di sistema, viene in genere chiamato **"mondo esterno"**.
Con questa definizione appartengono al mondo esterno anche tutte le unità periferiche e le stesse memorie di massa, indipendentemente dalla loro collocazione fisica eventualmente all'interno dello stesso contenitore del calcolatore. Queste unità sono infatti sede di fenomeni dotati di una propria dinamica indipendente dai ritmi interni della CPU.

Ovviamente è essenziale, per l'utilità di un calcolatore, la possibilità che esso possa ricevere dal mondo esterno informazioni ed emettere verso di esso risultati e comandi in modo corretto e tempestivo.

Tra il calcolatore (elettronico, digitale, programmabile) ed i sistemi del mondo esterno (che possono essere i più disparati) esistono varie disomogeneità, più o meno accentuate a seconda dei casi.

Si hanno in generale disomogeneità:

- **fisiche**
- **formali**
- **temporali**
- **spaziali**
- **attitudinali**

Il problema dell'interfacciamento di un calcolatore con il mondo esterno si risolve affrontando e risolvendo gli aspetti connessi con queste disomogeneità, mediante l'impiego integrato delle tecniche circuitali (HW) e programmative (SW) di volta in volta più opportune, in modo da mantenere in ingresso una corrispondenza tra la rappresentazione interna e i fenomeni esterni e in modo da propagare, in uscita, gli effetti delle elaborazioni interne sui fenomeni esterni.

E' interessante notare che queste disomogeneità vengono in genere superate con catene di **dispositivi interagenti**:

FENOMENO
sensore - trasduttore - mezzo trasmissivo - trasduttore
CALCOLATORE

Queste catene di trasferimenti e trasformazioni di informazioni attraversano diversi strati esterni ed interni al calcolatore (fig. 5.1).

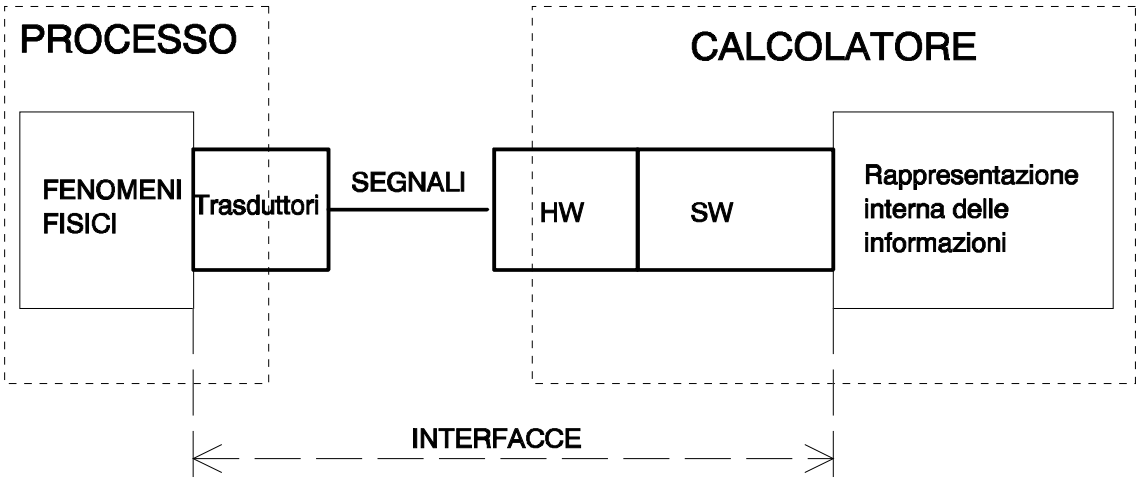

Fig. 5.1 - Elementi dell'interfacciamento

In quest'ottica si hanno numerose e diverse interfacce locali tra i vari elementi della catena che nel suo complesso realizza l'interfaccia globale.
Il progetto di un'interfaccia consiste quindi nell'individuazione di una catena di elementi che complessivamente soddisfano i requisiti globali di trasferimento e trasformazione delle informazioni. I requisiti locali di ogni elemento, e delle sue interfacce con quelli adiacenti, sono ereditati dai requisiti globali con diversi vincoli tecnologici ma anche svariati gradi di libertà di progetto.

Inoltre il modello di comunicazione delle informazioni tra i vari elementi dipende dal livello di astrazione a cui ci si pone, che a sua volta dipende da dove si collocano i problemi dominanti: così un'astrazione Fisica si focalizzerà sulle interazioni tra

fenomeni (**trasformazioni** operate da sensori e trasduttori), un'astrazione Informatica si centra sul problema del significato e della **rappresentazione delle informazioni**, mentre un'astrazione di Telecomunicazione si centra sul problema del loro **trasporto**.

La visione proposta per gli elementi delle catene di comunicazione sarà a "scatole nere" definite cioè dalla loro funzionalità ai "morsetti", riservandoci di adottare di volta in volta il livello di astrazione più adatto ai problemi da evidenziare (scatole cinesi).

Analizziamo ora brevemente i vari tipi di disomogeneità sopra citati.

La **disomogeneità FISICA** sostanzialmente si riferisce al fatto che le grandezze fisiche messe in gioco dai fenomeni del mondo esterno raramente sono grandezze elettriche come quelle in gioco all'interno del calcolatore; non solo, ma anche quando si tratti di grandezze elettriche (in particolare delle tensioni), non sempre esse sono di valore compatibile con quelle dei circuiti di un calcolatore e spesso neppure associate allo stesso riferimento di massa.
Il superamento delle disomogeneità fisiche richiede l'uso di **sensori** e **trasduttori** che, per alcuni tipi di fenomeni e grandezze fisiche, possono essere anche molto complessi, così da costituire a loro volta dei sottosistemi composti di numerosi elementi.

Una **disomogeneità FORMALE** si ha quando, pur avendo a disposizione delle grandezze elettriche, queste si presentano in una forma diversa da quella che dovra` essere poi utilizzata all'interno di un calcolatore. Per esempio casi tipici sono costituiti dalle grandezze analogiche o da codifiche digitali particolari (codici BCD, Gray, ecc.).

Si hanno anche altre disomogeneità formali; per es. un fenomeno esterno può presentarsi sotto forma di impulsi, i quali non corrispondono direttamente a delle informazioni memorizzabili ma per es. dovranno essere contati (mediante totalizzatori) o se ne dovrà rilevare la distanza temporale (misure di velocità).

Rientra in questa categoria di disomogeneità anche quella introdotta dalla "modulazione", che richiede una corrispondente "demodulazione" per estrarre da un segnale le caratteristiche rilevanti agli effetti dell'informazione.

Una **disomogeneità TEMPORALE** deriva dal fatto che i fenomeni del mondo esterno sono asincroni, evolvono cioè con una loro dinamica che non ha nessun legame a priori con i tempi di evoluzione dei programmi all'interno del calcolatore. C'e` quindi un problema di sincronizzazione tra le attività, affinchè le operazioni di acquisizione ed emissione di informazioni avvengano in istanti opportuni rispetto all'evoluzione dei fenomeni esterni.
Si noti che la dinamica dei fenomeni dipende spesso da leggi fisiche su cui non è facile intervenire, ed è quindi in generale il calcolatore a doversi adeguare.

Vengono a questo scopo utilizzate varie tecniche per la sincronizzazione:
- tecniche circuitali,
- i tre meccanismi di base classici:
  - controllo di programma (*polling*)
  - interrupt,
  - DMA (accesso diretto alla memoria)
- meccanismi di sincronizzazione e temporizzazione dei processi di elaborazione interna al calcolatore (trattati nel cap. 10 - Sistemi Operativi).

Il corretto superamento di queste disomogeneità temporali costituisce uno dei problemi centrali dell' **elaborazione in tempo reale**.

La **disomogeneità SPAZIALE** si riferisce alla collocazione del calcolatore e dei dispositivi interagenti con esso, che possono essere posti a distanze anche rilevanti.
Il superamento di questo aspetto viene realizzato mediante opportune tecniche di trasmissione, in cui i vincoli imposti dai problemi di **trasporto** delle informazioni assumono un'influenza rilevante sulle tecniche di progetto da adottare.
Infatti i fenomeni fisici adatti ad un buono sfruttamento dei mezzi trasmissivi portano ad adottare, nella catena di interfacciamento, elementi specificatamente adatti al trasporto (ad es. modulazione), che spesso introducono localmente disomogeneità fisiche, formali e temporali che non sarebbero concettualmente necessarie.

Per quanto riguarda la **disomogeneità ATTITUDINALE** questa si evidenzia nell'interfacciamento tra **elementi attivi**, e tipicamente nell'interfacciamento con l'uomo, che ha caratteristiche di comprensione ed elaborazione delle informazioni molto diverse da quelle del calcolatore.
I problemi di adattamento a queste diversità attitudinali sono risolti di solito con unità periferiche opportune, ("ergonomiche") e con opportuni algoritmi per la gestione di queste unità in modo "*user friendly*" (interattività uomo- macchina).
Disomogeneità attitudinali, sia pure meno marcate, si riscontrano però anche tra diversi dispositivi più o meno "intelligenti", cioè dalla funzionalità più o meno articolata, caratterizzati da diversi modelli di comportamento.
L'adozione di opportuni **protocolli** costituisce il tipico modo di superare queste disomogeneità.

5.2 SEGNALI

Una trattazione formale e completa della teoria dell'informazione e della sua rappresentazione e trasmissione mediante segnali, va oltre gli scopi di questa presentazione, che rimanda quindi i lettori ad altri testi, alcuni dei quali citati in bibliografia. Tuttavia si ritiene utile una serie di richiami dei concetti fondamentali, da un punto di vista informatico ma con intenti interdisciplinari.

I segnali sono **grandezze fisiche utilizzate per rappresentare e trasportare informazioni** che a loro volta possono essere descrittive di fenomeni fisici (ad esempio velocità di rotazione di un motore) o essere di tipo astratto (ad es. caratteri di un testo).

Ogni singolo segnale è prodotto da uno o più **generatori**, in *mutua esclusione*, mentre può venire utilizzato da uno o più **ricevitori** anche *contemporaneamente*.

E' importante notare che il **livello energetico** delle grandezze fisiche che costituiscono i segnali è solo connesso alla loro pratica utilizzabilità fisica in un determinato contesto, cioè per il trasporto e percezione da parte dei riceventi, con **adeguata insensibilità** agli agenti perturbanti (il classico rapporto segnale/rumore), ma che tale livello energetico non ha ulteriore importanza concettuale.

Con i dispositivi elettronici i segnali sono generalmente costituiti da grandezze elettriche, ed in particolare da tensioni o da correnti.

Un crescente interesse presentano anche i segnali luminosi, soprattutto grazie al continuo progresso delle tecnologie di conversione da/verso grandezze elettriche (optoelettronica) e di trasporto (fibre ottiche).

I segnali di tipo **elettronico** rendono economicamente possibili varie elaborazioni a bassi livelli energetici e (quindi) ad elevate velocità e quindi nella maggior parte dei casi i segnali vengono convertiti in questa forma.

Analizziamo ora l'importante ruolo dei segnali di **rappresentare delle informazioni**.
La rappresentazione è basata sulla corrispondenza tra caratteristiche di valore e temporali dei segnali e le informazioni.
Le caratteristiche sono impresse *per costruzione* dal dispositivo generatore del segnale e ad esse devono essere sensibili i dispositivi ricevitori. Queste caratteristiche sono il **valore**, **l'andamento temporale** e gli **eventi** del segnale. L'informazione rappresentata può essere il valore, l'andamento temporale o gli eventi di un fenomeno. E' molto importante notare che la costruzione di un segnale non deve necessariamente trasferire le informazioni sulle caratteristiche omologhe, ma la corrispondenza può essere arbitraria, tecnologie permettendo.
Ad esempio un valore costante può essere rappresentato da un segnale a frequenza costante, un evento può essere rappresentato da un segnale di determinata ampiezza, un andamento esponenziale può essere rappresentato da un segnale che presenta una successione di eventi binari (trasmissione seriale) che codificano il valore istantaneo e la costante di tempo, ecc. Occorre quindi *chiaramente distinguere tra stati, andamenti ed eventi* del **segnale** e del **fenomeno** (cioè dell'informazione) da esso descritto.

La correlazione più spontanea tra informazione e segnale è quella che associa il valore dell'informazione all'ampiezza del segnale, ma sono utilizzate anche altre correlazioni:
- in banda base
  durata (intervallo fra variazioni discordi del segnale)
  periodo (intervallo fra variazioni concordi del segnale)

- con modulazione, cioè trasposizione sui parametri di un andamento sinusoidale

S(t) = A sin (Wt + F)

ampiezza	**A**
frequenza	**W**
fase	**F**

E' importante notare che **tutti i segnali fisici sono continui a tempo continuo** se si adottano i consueti modelli fisici ad energia cinetica / energia potenziale (modelli del secondo ordine) con grandezze finite, mentre **non è detto che siano di tipo continuo le informazioni ad essi associate**.

La **rappresentazione** delle informazioni può infatti essere di tipo **analogico** o di tipo **digitale**, a seconda che l'informazione sia associata biunivocamente al valore del segnale o sia associata al particolare intervallo finito a cui tale valore appartiene (v. nel seguito segnali analogici e digitali).
In altre parole l'attributo "analogico" o "digitale" non si riferisce a caratteristiche intrinseche del segnale ma a **caratteristiche dell'informazione** da esso rappresentata, anche se in genere si parla, per brevità, di segnali analogici e segnali digitali.

NOTA.
Si può avere un segnale digitale in ampiezza ma analogico in durata (oppure in periodo) e che quindi corrisponde ad uno *stato discreto tempo-continuo* (v. cap.2).
Es. controllo di riscaldante a *duty-cycle* variabile. (v. interfacciamento di impulsi - cap.7).

DUTY-CYCLE => percentuale di tempo in cui un segnale digitale binario periodico ha un valore appartenente all'intervallo detto ON (attivo).

Un tipico problema riguardo ai segnali è costituito dagli **scostamenti** rispetto al valore "ideale" a causa di interferenze non volute di vari fenomeni fisici “disturbanti” con quelli utilizzati per la generazione ed il trasporto dei segnali.

Questi scostamenti sono generalmente introdotti da:
- scostamenti in sede di generazione o conversione
- rumore (disturbi) in sede di generazione o conversione
- rumore in sede di trasporto

e tipicamente danno luogo ad **errori** con componenti sistematiche e casuali (v. cap. 2) sulle informazioni rappresentate.

Si tende generalmente a minimizzare l'entità dei disturbi all'origine, ma oltre un certo limite ciò non è (economicamente) possibile.

Alcuni disturbi sono chiaramente **distinguibili** dalle informazioni significative adottando opportuni criteri, generalmente basati sulla distinzione di bande di frequenza o in generale su caratteristiche temporali diverse tra segnale e rumore.
I disturbi, o gli errori ad essi connessi, possono in questi casi essere **eliminati** o **ridotti** con tecniche circuitali (filtri) o algoritmiche (filtraggio numerico o altre elaborazioni). (v. acquisizione di segnali digitali - cap.7 - e analogici - cap.8).

Nel caso di segnali digitali i disturbi che ne deformano il valore senza però portarlo a superare la soglia dell'intervallo significativo, non producono errori nell'informazione rappresentata. Quando invece i disturbi superano la soglia di margine di rumore (v. seguito - segnali digitali) si hanno errori che entro certi limiti possono essere rilevati o anche corretti con l’adozione di particolari codifiche (ad es. bit di parità, codifica Hamming, ecc.).

5.2.1 SEGNALI ANALOGICI.

5.2.1.1 COSA SONO I SEGNALI ANALOGICI

Sono segnali che con la loro ampiezza, o più raramente con altri parametri (ad es. la frequenza), rappresentano direttamente il valore di grandezze fisiche (misure), o comunque di valori numerici (numeri reali), che possono assumere tutti i valori in un certo intervallo finito.

I segnali analogici sono generati da sensori o trasduttori (v. seguito)

5.2.1.2 VALORI TIPICI DEI SEGNALI ANALOGICI

Normalmente i segnali analogici destinati a rappresentare delle misure sono opportunamente condizionati (in genere amplificati) affinchè il loro campo di valori rientri in uno dei seguenti intervalli:

0..10 V
-5..+5 V
0..20 mA
4..20 mA

Questa standardizzazione facilita il reperimento sul mercato e l'uso (compatibilità) di dispositivi con interfacce analogiche come convertitori A/D, registratori, regolatori, attuatori, ecc.

5.2.1.3 CORRISPONDENZA CON LE INFORMAZIONI RAPPRESENTATE

Per un buon progetto delle interfacce è molto importante conoscere in modo accurato la corrispondenza tra valori del segnale e valori rappresentati, e i rispettivi intervalli.
Le corrispondenze tra segnali e valori sono rappresentabili **graficamente** con segmenti o curve su un piano cartesiano, come negli esempi seguenti.
Da un punto di vista **informatico** le corrispondenze vengono descritte con coefficienti di funzioni analitiche oppure con tabelle (array). Queste considerazioni sono riprese nel cap.8.

Nelle figure 5.2 e 5.3 sono rappresentate le corrispondenze tra segnale e valore rappresentato con esempi di segnale lineare e non lineare.

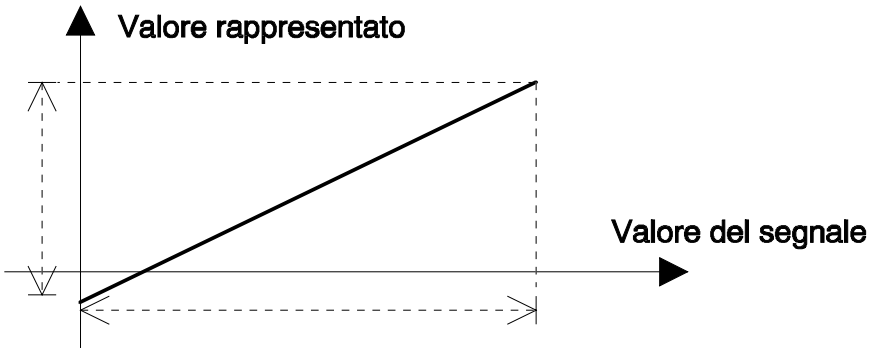

Fig. 5.2 - Segnale lineare.

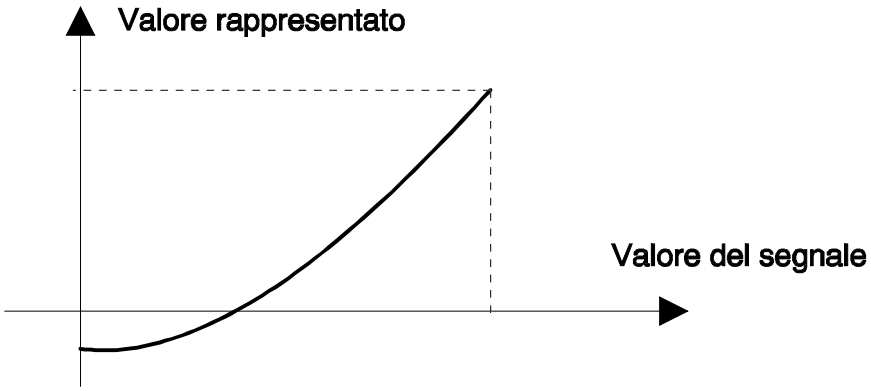

Fig.5.3 - Segnale non lineare

Si usano spesso segnali con "**zero vivo**", per riconoscere il caso di sensore non collegato o non alimentato. Questi segnali presentano un valore diverso da zero anche per la rappresentazione di un valore nullo.

Esempio.

Consideriamo ad esempio una misura di pressione assoluta che, per definizione, non può assumere valori negativi. Con un sensore che generi un segnale nullo per pressione P = 0 sarebbe indistinguibile l'interruzione del collegamento del segnale (situazione anomala e da riparare) dal rilievo di una pressione di 0 bar (situazione potenzialmente normale).

In generale il riconoscimento di anomalie come **interruzione** o **cortocircuito** dei collegamenti sono rilevabili se gli estremi dell'intervallo di valori fisicamente possibili per il segnale sono esterni all'intervallo che rappresenta valori ammissibili dell'informazione rappresentata, come nella fig. 5.4.

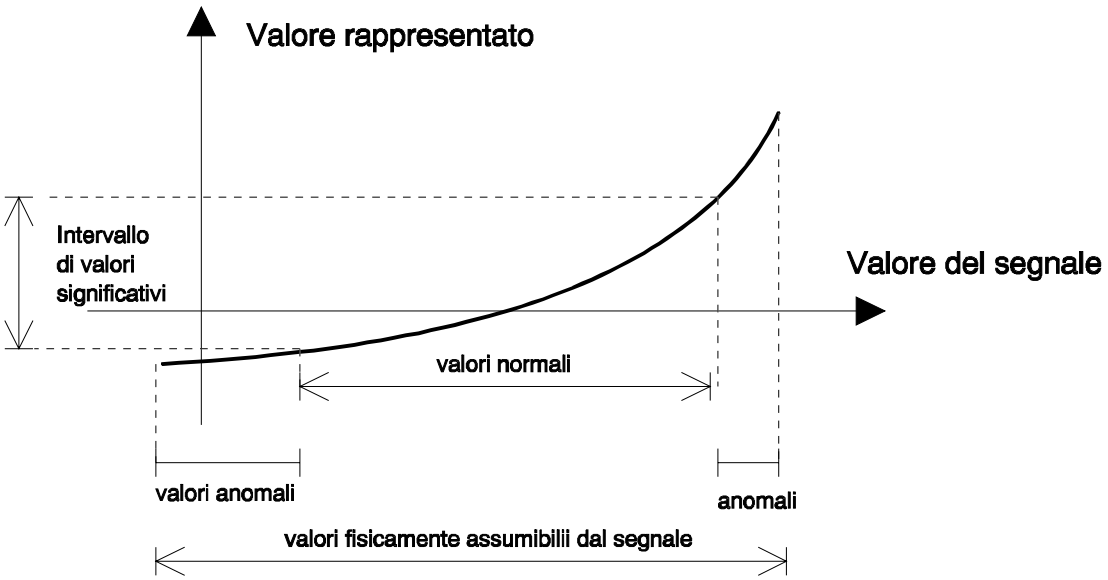

Fig. 5.4 - Caratteristica di un segnale (non lineare) con campo di valori **significativi** interno al campo di valori fisicamente **assumibili**.

5.2.1.4 PREGI DEI SEGNALI ANALOGICI

I segnali analogici presentano vari interessanti aspetti positivi che in molti casi ne garantiscono la sopravvivenza anche in ambienti sempre più invasi dalle tecnologie digitali.

I principali pregi dei segnali analogici sono:

- Esistono vari fenomeni che generano **spontaneamente** segnali analogici. (sensori, trasduttori).

- Esistono vari dispositivi semplici, a funzionalità intrinseca, che sono in grado di **interpretare** e trasformare i segnali analogici (trasduttori, attuatori).

Anche nell'uomo è ben radicato il concetto di "continuo", che spesso è più consono alle attitudini umane.

- Eventuali perturbazioni **deformano** il segnale, ma generalmente **non lo stravolgono** completamente, cioè introducono, nel riconoscimento delle informazioni, errori percentualmente limitati e "proporzionali" all'entità del disturbo.

- E' sufficiente un **unico conduttore** (ed eventuale ritorno) per trasmettere informazioni "ricche" (cioè appartenenti ad un insieme ad **elevata cardinalità**), cioè per rappresentare idealmente gli infiniti valori numerici in un certo intervallo.

5.2.1.5 DIFETTI DEI SEGNALI ANALOGICI

- I dispositivi di elaborazione dei segnali analogici sono relativamente **poco precisi**, poco **stabili** nel tempo e poco **immuni** alle perturbazioni.
Volendo migliorare le loro caratteristiche in tal senso, il loro costo cresce *molto rapidamente*, ed in ogni caso piccoli errori sono *ineliminabili*.

- Le trasformazioni eseguibili direttamente sui segnali analogici con i dispositivi sopra citati sono **poco flessibili** e si riducono sostanzialmente a operazioni di somma algebrica e operazioni di tipo integro/differenziale.

- E' molto difficile realizzare buone **memorie** permanenti di segnali analogici. I circuiti *Sample & Hold* (v. cap.8) sono relativamente costosi e "perdono" gradualmente il valore analogico memorizzato.

- Ogni elaborazione dei segnali analogici produce un **degrado** della precisione, e ciò limita il numero di dispositivi collegabili in cascata e quindi la complessità globale delle elaborazioni effettuabili.

- Nelle applicazioni in cui sono previsti strumenti informatici di elaborazione sono necessarie (costose) tecniche di "**conversione**" (A/D analogico/digitale, e D/A digitale/analogico).

5.2.2 SEGNALI DIGITALI

5.2.2.1 COSA SONO I SEGNALI DIGITALI

Per la rappresentazione di **informazioni digitali** (numero finito di valori) mediante segnali viene creata una corrispondenza convenzionale tra i valori dei segnali (o meglio "fasce" di valori dei segnali) e le informazioni rappresentate, che sono costituite da simboli di un alfabeto finito.

Queste corrispondenze, descritte in forma grafica, assumono tipicamente una forma a scala, in cui l'estensione orizzontale di ogni scalino rappresenta la fascia di valori del segnale associata all'informazione digitale corrispondente all'ordinata dello scalino (fig.5.5).

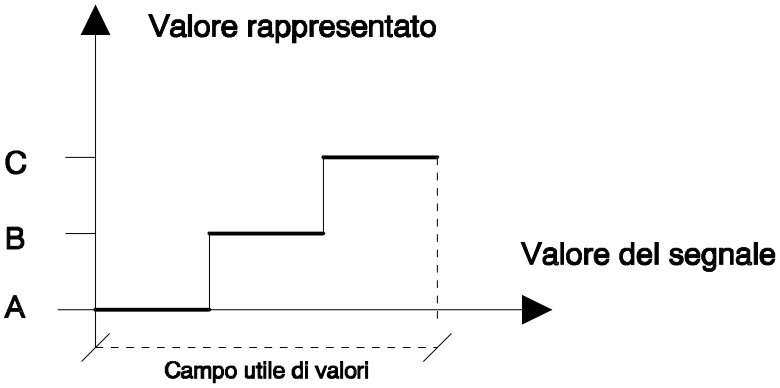

Fig 5.5 - Corrispondenza tra ampiezza e "significato" di un segnale digitale (caso ternario, con valori dell'informazione detti A, B, C)

Poichè per motivi di semplicità costruttiva e di sfruttamento dei segnali si adotta normalmente un numero estremamente limitato di intervalli, cioè un alfabeto a cardinalità (pari a 2 nel caso binario) troppo bassa per molte applicazioni, si assume in generale la seguente impostazione:

CODIFICA => corrispondenza significato <=> simbolo
ogni simbolo è rappresentato da una successione di digit
ogni digit è rappresentato da un segnale "digitale"

Ciò implica che informazioni caratterizzate da una certa ricchezza di valori richiedono vari (talora anche numerosi) segnali (cioè digit) per la loro rappresentazione, come dalla seguente nota relazione:

N = trunc (log base D (S)) + 1

dove

N = n. segnali, cioè n. di digit
D = cardinalità dei digit (n. valori assumibili da un digit)
S = cardinalità dell'alfabeto dei simboli (n. informazioni rappresentabili)

5.2.2.2 SEGNALI DIGITALI BINARI

Sono caratterizzati da due fasce di valori (alto e basso), spesso chiamati H (High) e L (Low) a cui sono convenzionalmente associati i significati:

Logica positiva H = 1 = TRUE = ON
L = 0 = FALSE = OFF

Logica negativa H = 0 = FALSE = OFF
L = 1 = TRUE = ON

NOTA
- H, L fanno riferimento ai valori elettrici.
- 0, 1 sono valori numerici, ma per brevità usati anche in senso traslato.
- TRUE, FALSE sono valori di verità logica.
- ON, OFF sono usati nel gergo dell'automazione per indicare attivazione (ON) o disattivazione (OFF) di dispositivi.

Notiamo che un confine netto tra le fasce di valori del segnale associate a diversi simboli digitali è significativo solo concettualmente. Infatti i dispositivi fisici, per tutta una serie di motivi, presenteranno sempre una, sia pur ristretta, fascia di incertezza di cui occorre tener conto per un'analisi del funzionamento di tali dispositivi.
E' molto importante notare che **non si tratta di un fenomeno di isteresi**. Si ricordi infatti che l'isteresi si riferisce a diversi comportamenti dei dispositivi in corrispondenza a diversi sensi di **variazione** dei segnali.. Naturalmente per alcuni dispositivi, come ad esempio i circuiti *trigger di Schmitt,* si ha un (voluto) fenomeno di isteresi **oltre** ad avere una fascia di indeterminazione.

La rappresentazione della tipica corrispondenza tra valori H e L e valori del segnale binario è rappresentata in fig.5.6. La fascia di valori contrassegnata con il "?" corrisponde all'intervallo di indeterminazione.

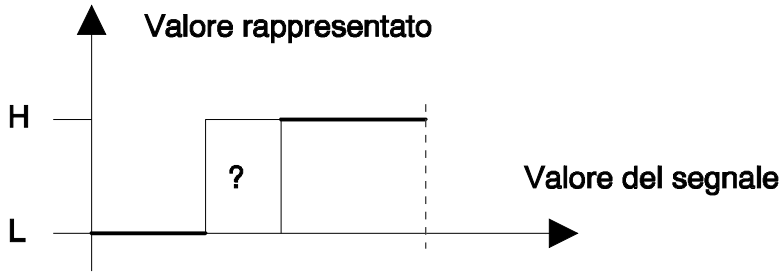

Fig.5.6 - Caratteristica Ampiezza / valore di segnale binario

Nell'intervallo di indeterminazione si accetta che il valore rappresentato non sia precisato a priori.
Si noti bene che non si tratta di un terzo valore del digit rappresentato, ma di un intervallo in cui non è specificato a priori quale dei due valori di informazione verrà di fatto attribuito al segnale dal particolare dispositivo ricevente.

5.2.2.3 VALORI TIPICI DI SEGNALI DIGITALI

Per i segnali digitali si hanno diverse **specifiche** dei valori prodotti in uscita e dei valori accettati in ingresso. In particolare si specificano le fascie di valori **garantiti in uscita** (nelle condizioni di carico previste) e valori **richiesti in ingresso**. Ovviamente la condizione favorevole, e quindi perseguita nei progetti dei circuiti elettronici digitali, è quella in cui le fasce di valori garantiti in uscita sono strettamente contenute nelle fasce dei valori richiesti in ingresso.
Per **margine di rumore** si intende il massimo scostamento, rispetto ai valori di segnale **garantiti in uscita** dal circuito generatore, che rientra ancora nella fascia di **corretta interpretazione del segnale in ingresso** nel circuito ricevitore.
Il margine di rumore è quindi una **proprietà di una coppia ordinata** di circuiti **generatore e ricevitore**. In genere il margine di rumore viene fornito rispetto a

qualunque coppia di circuiti della stessa *famiglia* e quindi omogenei tra loro nelle specifiche rispettivamente sugli ingressi e sulle uscite.

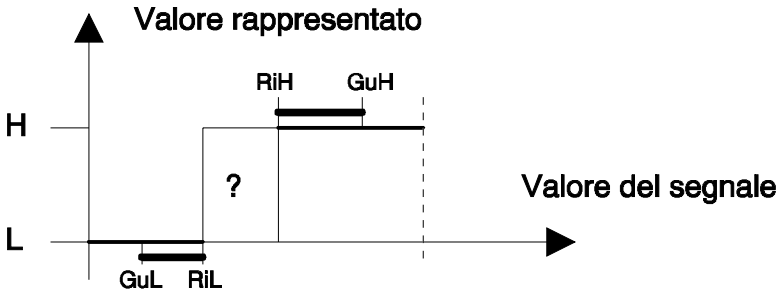

Fig. 5.7 - Margini di rumore per i valori L e H.

GuL = max valore L garantito in uscita	V_{OLmax}
RiL = max valore L richiesto in ingresso	V_{ILmax}
GuH = min valore H garantito in uscita	V_{OLmin}
RiH = min valore H richiesto in ingresso	V_{ILmin}

Quando per i valori in uscita si ha una rilevante differenza tra valore tipico e minimo o massimo garantito (come per es. per la tecnologia TTL), si può parlare di margine di rumore tipico e di margine di rumore garantito.

NOTE
Varie famiglie di circuiti presentano una dissimmetria tra i margini di rumore rispetto al livello alto (H) e al livello basso (L), come pure una dissimmetria nel valore di impedenza d'uscita.
La sensibilità ai disturbi elettromagnetici indotti è crescente con l'impedenza d'uscita del circuito generatore.
La **robustezza** rispetto a fenomeni disturbanti risulta dalla sovrapposizione dei due aspetti appena citati (margine di rumore e impedenza), che possono anche compensarsi se al livello con minore margine di rumore corrisponde un'impedenza d'uscita più bassa che rende tale livello meno perturbabile.
Le caratteristiche appena discusse devono essere tenute presenti nella scelta del significato da attribuire ai due valori (H e L), in modo da adottare il livello più robusto per asserire il valore più critico, scegliendo tra logica positiva o negativa.

CIRCUITI DIGITALI SPECIALI
Alcuni circuiti digitali utilizzano segnali con valori di tensione relativamente ampi per ridurre il peso relativo dei disturbi quando si debbano utilizzare lunghi cablaggi in ambienti industriali o comunque in zone disturbate.
Tensioni on/off
0.V - .12 V
0.V - .24 V
Trasmissioni seriali (EIA RS-232)
- 6 V - + 6 V

CIRCUITI DIGITALI NORMALI

I normali circuiti digitali sono quelli utilizzati per realizzare sottosistemi con connessioni brevi (decine di cm), come i tipici circuiti digitali e i calcolatori.

N.B. Nel seguito tutti i valori, da considerarsi indicativi ed esemplificativi, sono espressi in volt, e si riferiscono a circuiti alimentati con la tensione nominale di 5 V =.
Un'analisi accurata di compatibilità di livelli e margini di rumore richiede la consultazione dei dati tecnici (*data sheet*) forniti dai costruttori dei dispositivi.

TECNOLOGIA TTL

Valori offerti dalle uscite

Valore	min	tip	max
H	2.4	3.4	5
L	0	0.2	0.4

Valori richiesti dagli ingressi

H > 2
L < 0.8

Margine di rumore

tipico	H	3.4 - 2	= 1.4
	L	0.8 - 0.2	= 0.6
garantito	H	2.4 - 2	= 0.4
	L	0.8 - 0.4	= 0.4

TECNOLOGIA CMOS

Valori offerti dalle uscite

H 4.95 5
L 0 0.05

Valori richiesti dagli ingressi

H > 3.5
L < 1.5

Margine di rumore

garantito	H	4.95 - 3.5	= 1.45
	L	1.5 - 0.05	= 1.45

TECNOLOGIA CMOS-HC

Valori offerti dalle uscite

H 4.9 5
L 0 0.1

Valori richiesti dagli ingressi

H > 1.5
L < 0.5

Margine di rumore

garantito	H	4.9 - 1.5	= 1.4
	L	0.5 - 0.1	= 0.4

NOTA

Perchè segnali a due valori invece che a tanti valori ?

- Semplicità circuitale (conduzione/interdizione) anche con una sola tensione di alimentazione.
- Mappaggio immediato dei due valori sulla logica vero/falso.
- La base "e" (e = 2.71828) minimizza il prodotto Ncif*Base.

Se assumiamo come "costo" della rappresentazione di un'informazione il prodotto del numero di cifre (digit) necessarie moltiplicato per la cardinalità delle cifre (che è un indice della loro "complessità"), il costo associato alla rappresentazione del valore numerico K mediante cifre in base x è esprimibile da:

logx (K) * x

dove logx (K) rappresenta il numero di cifre richieste per rappresentare l'informazione.
Per trovare la base x che minimizza tale costo passiamo ai logaritmi naturali (base e):

logx (K) * x = x * ln(K) / ln(x)

la cui derivata è:

ln(K) * (1/ln(x) + x * ((-1/x) / (1/ln(x)) exp 2)

che si annulla per ln(x) = 1 quindi per x = e
Con questa impostazione le basi intere ottimali sono 2 e 3.

5.2.2.4 CODIFICHE BINARIE

I concetti e le tecniche per la rappresentazione binaria delle informazioni sono dati per noti al lettore, e quindi nel seguito si riportano solo sintetici richiami.

L'elemento di informazione binaria è il **bit** (binary digit).

Con N bit si hanno $M = 2^N$ configurazioni diverse (simboli).

Tra M informazioni e le (o un sottoinsieme delle) configurazioni di N bit si può creare una **convenzionale corrispondenza biunivoca**.

L'arbitrarietà della scelta di codifica viene di solito sfruttata per godere di **proprietà utili**, quali:

- - facilità di operazioni aritmetiche tra numeri (codifica pesata e complemento a 2)
- - ordinamento alfabetico tra le lettere (codifica ASCII)
- - ordinamenti naturali delle informazioni
- - raggruppamenti naturali delle informazioni (significato dei singoli bit o di gruppi)
- - facilità di conversione in base 10 (codifica BCD)
- - proprietà di simboli contigui (codifica Gray)
- - riconoscimento e correzione di alcuni errori (codifica Hamming)

La corrispondenza una volta decisa va scrupolosamente rispettata tra generatore del codice (codificatore) e interpretatore del codice (decodificatore) se si vuole ricostruire correttamente le informazioni.

CODIFICHE

- **- ENUMERATIVE**

Alle diverse informazioni sono associate diverse configurazioni di N bit

- **- CARATTERI ASCII**

Si utilizzano 7 bit per rappresentare tutti i caratteri di stampa oltre ad un certo numero di caratteri di controllo

- **- NUMERI NATURALI binario puro** (codifica pesata) in cui il valore V è dato da:

V = Bn*2^n + Bn-1*2^n-1 + ... B1*2^1 + B0*2^0

Questa è la codifica tipica dei tipi `unsigned`, per cui vale l'interessante proprietà:

siano V1, V2 e D valori interi rappresentabili con N bit, cioè minori di 2^N

Es. N=8 bit, V1=250, D=16, V2=10

ADD (X, Y) e SUB (X, Y) siano le istruzioni assembler di somma e sottrazione

se V2 = ADD (V1, D) eventualmente anche con overflow,

allora D = SUB (V2, V1)

Questa proprietà consente di usare un normale contatore (`unsigned`) e calcolare la differenza D tra le letture V2 e V1, indipendentemente dall'*overflow* del contatore.

E' opportuno **ricordare** che i campi di valori rappresentabili sono:

8 bit	0 . . 255	
10 bit	0 . . 1023	1 K
12 bit	0 . . 4095	
16 bit	0 . . 65535	
20 bit	0 . . circa 1 milione	1 M
30 bit	0 . . circa 1 miliardo	1 G
32 bit	0 . . circa 4 miliardi	

- **- NUMERI NATURALI Gray**

Ha la proprietà che due valori contigui si differenziano per il valore di un solo bit

- **- NUMERI NATURALI BCD**

Ogni gruppo di 4 bit rappresenta in binario puro una cifra decimale

- **- NUMERI INTERI complemento a 2** (`integer`)

Questa codifica ha la notevole proprietà di unificare operazioni di somma e di sottrazione nella somma algebrica.

- **- NUMERI** in **virgola fissa**

Costituisce un'estensione alle cifre dopo la virgola della codifica pesata.

Presentano *risoluzione costante in valore assoluto*. - Tipica di conteggi, contabilità, ecc. (v. cap.2)

La risoluzione **assoluta** Ra con *Nf* bit dopo la virgola vale $Ra = 2^{-Nf}$

Si noti che i tipi interi, pur di tener conto esplicitamente di fattori di scala delle unità di misura, possono essere **considerati a virgola fissa**. Ciò consente calcoli anche con valori inferiori all'unità pur usufruendo della maggior velocità delle operazioni tra interi rispetto a quella delle istruzioni tra valori in virgola mobile.

Tutte le codifiche numeriche sopra riportate presentano per tutti i valori rappresentati una **risoluzione costante in valore assoluto**.

- **NUMERI** in **virgola mobile** (`float` o `real`)

Sono separatamente rappresentate la caratteristica e la mantissa.
Presentano *risoluzione costante in <u>valore percentuale</u>* rispetto al valore rappresentato (v. cap.2).
Questa è la rappresentazione tipica di misure e valori ingegneristici.
Risoluzione **relativa** Rr con *Nm* bit dedicati alla mantissa Rr = $\cdot 2^{-Nm}$
che con il tipico Nm = 23 bit vale circa uno su 8 milioni.

STANDARD IEEE di codifica *floating point* su 32 bit
Usato ad es. per il tipo `float` del linguaggio C.
L'attribuzione dei 32 bit è la seguente:

SEEEEEEE EMMMMMMM MMMMMMMM MMMMMMMM

S = segno mantissa (0 = + 1 = -)
E = esponente + 127
M = mantissa

Il valore numerico V vale: $V = (-1)^S \cdot 2^{E-127} \cdot (1.M)$

N.B. 1/10 è *periodico* in base 2 !! Come conseguenza non tutti i numeri frazionari esprimibili esattamente con un numero finito di cifre decimali sono anche esprimibili esattamente con un numero finito di bit.

- **Varianti con ridondanza**

In vari casi ai normali codici si aggiungono bit ridondanti per i seguenti scopi:
-- per rilievo di errori (ad es. bit di parità)
-- per correzione di errori (codici di Hamming)

5.2.2.5 PREGI DEI SEGNALI DIGITALI

I numerosi ed interessanti pregi dei segnali digitali (soprattutto binari) sono alla base della ben nota evoluzione (o rivoluzione) elettronica ed informatica. Citiamo brevemente i principali:

- Possibilità di rappresentare con **esattezza** e senza incertezza simboli di un alfabeto **finito**. L'aumento della precisione (cardinalità dell'alfabeto) ha un costo crescente solo con il logaritmo (numero di bit necessari).
- **Semplicità e robustezza** dei circuiti di generazione e riconoscimento dei segnali.
- Elevata **immunità** ai disturbi (consente la rappresentazione esatta con elevata probabilità).
- Semplicità dei circuiti di **memoria** (bistabili).
- Semplicità dei circuiti per semplici elaborazioni.
- Realizzabilità di elaborazioni complesse con (elevato) numero di circuiti semplici, senza degrado (v. eccezioni) della precisione dei risultati, grazie all'effetto di **rigenerazione** consentito dai margini di rumore.

5.2.2.6 DIFETTI DEI SEGNALI DIGITALI

I difetti dei segnali digitali sono, come è spontaneo pensare, complementari ai pregi dei segnali analogici. Su questi punti vanno quindi effettuate attente analisi quando si debba decidere per l'una o l'altra forma.

- Gli errori possono colpire indifferentemente i bit meno significativi (LSB) o (purtroppo!) i più significativi (MSB)
- Gli errori possono "**stravolgere**" l'informazione (corollario del punto precedente).
- L'interpretazione del valore è poco "immediata" (passa attraverso una codifica).
- Occorrono vari bit (in parallelo o in serie) per rappresentare informazioni "ricche".
- Con bit in parallelo si hanno problemi di commutazioni non contemporanee (corse critiche), da cui l'uso di codice Gray.
- Si ha la necessità di **conversione** (A/D) nei numerosi casi di informazioni che sono intrinsecamente continue, accettando errori di quantizzazione.

5.2.3 CAMPIONAMENTO DI SEGNALI

Il campionamento consiste nell'acquisire il valore di un segnale in istanti temporali a distanza finita, mediante operazioni di INPUT o con trasferimenti in DMA, ed è l'unico modo con cui una macchina con funzionamento **a passi**, come i calcolatori digitali programmabili, può acquisire l'andamento del valore di un segnale nel tempo.
Usualmente il campionamento viene effettuato ad intervalli regolari.
Si effettua così una "discretizzazione" del tempo.

Il campionamento si applica, ovviamente con problematiche diverse, **sia a segnali analogici che a segnali digitali** (stati continui o discreti). Gli aspetti studiati dalla classica teoria del campionamento fanno riferimento ai segnali analogici, ai quali si riferiscono le considerazioni seguenti (riprese nel cap. 8) mentre per i segnali digitali si veda la sezione 5.3.1 e il cap.7.

NOTA. E' importante distinguere **due diversi scopi** per cui si effettua il campionamento di un segnale, che comportano diverse modalità di gestione delle relative informazioni.

> Campionamento per **mantenere aggiornata una rappresentazione** interna del valore di uno stato. Questo scopo si applica sia a segnali **analogici** che a segnali **digitali**.
> Con questo scopo dallo *stato* S(t) di un segnale S variabile nel tempo
> si produce uno *stato immagine* SI(t),
> dove SI(t) ha la proprietà di variare solo in corrispondenza degli istanti di campionamento, più o meno regolari.
>
> Campionamento per **rilevare un andamento temporale**, solitamente di un fenomeno esterno continuo. Questo scopo si applica solo a segnali **analogici**.
> A questa situazione si fa normalmente riferimento quando si tratta la teoria del campionamento, in cui si assume generalmente che il campionamento avvenga con periodo P costante.
> In questi casi da uno *stato* S(t)
> si produce una *sequenza di eventi assoluti* E(t0), E(t0+P),..E(t0+nP) che costituiscono i campioni.

- CONSIDERAZIONI INTUITIVE

Poichè è evidentemente desiderabile che il segnale "campionato" non venga a perdere le informazioni significative contenute nel segnale continuo, è importante chiedersi quale sia la minima frequenza di campionamento senza perdita di informazione.

E' facile immaginare che tale frequenza potrà essere tanto più bassa quanto più lente sono le variazioni nel tempo del segnale.
Quindi la “bontà” del rilievo pone un limite inferiore alla frequenza di campionamento, mentre un limite superiore è imposto dal carico di lavoro sopportabile dal processore.

- TEOREMA DEL CAMPIONAMENTO

Sia:

T = periodo di campionamento

fmax = massima frequenza con ampiezza significativa nello spettro del segnale da campionare.

Affinchè l’andamento del segnale sia ricostruibile sulla base dei valori ottenuti da campionamento, il periodo *T* deve soddisfare la relazione:

$$T < \frac{1}{2 \cdot f_{\max}}$$

- L'ALIASING

E' il fenomeno per cui un campionamento che non rispetti il limite di frequenza del teorema di Shannon provoca la comparsa di variazioni spurie, come una specie di "battimenti". (A causa di questo fenomeno nei film le ruote di veicoli in movimento spesso sembrano ferme o girare a ritroso quando, come spesso avviene, le frequenze di ripresa dei fotogrammi non rispettano il teorema di Shannon rispetto alla velocità di rotazione delle ruote).

- CAMPIONAMENTO PER ELABORAZIONE IN TEMPO REALE

Mentre rispettando il teorema del campionamento si ha concettualmente la possibilità di ricostruire **a posteriori** (*off-line*) l’andamento originale del segnale, per una **fedele** e **tempestiva** (*on-line*) ricostruzione dell’andamento del segnale si adottano normalmente frequenze di campionamento pari a 10 . . 100 volte fmax.
Si ricordi che il campionamento introduce un **effetto di ritardo** pari a T/2.

- NECESSITA' DI FILTRAGGIO

Il filtraggio dei segnali destinati al campionamento è importante per ridurre a valori trascurabili le componenti a frequenze non volute o spurie.

- Filtro passa basso, a livello HW sul segnale, per rispettare il teorema di Shannon ed evitare l'aliasing e per ridurre il contenuto di rumore.

- Filtro sopprimi banda (50 Hz) per ridurre i disturbi indotti dalla rete di distribuzione dell'energia elettrica.

5.3 INFORMAZIONI ASSOCIATE A STATI O AD EVENTI

Distinguiamo fra l'associare delle informazioni a **stati** dei segnali e l'associare informazioni ad **eventi** dei segnali.

Si puo` attribuire una informazione allo **stato** del segnale quando è irrilevante la storia con cui si è pervenuti allo stato, come nel caso tipico di un interruttore che con i suoi due stati può essere usato per comandare una lampadina accesa o spenta, un motore in marcia o fermo, ecc.
Altro esempio tipico è costituito dagli encoder assoluti che in ogni istante forniscono la codifica binaria (generalmente in codice Gray) della loro posizione angolare.

In altri casi l'informazione e` associata ad **eventi**: in questi casi il permanere di uno stato e` considerato irrilevante ma l'informazione che interessa e` costituita dalla transizione da uno stato all'altro (chiusura di un interruttore, pressione di un pulsante, presenza di impulsi).
Gli encoder incrementali sono un esempio di generatori di segnali in cui le informazioni sono associate ad eventi.

In questa sezione vengono analizzati soprattutto i **problemi** da affrontare per un progetto corretto delle interfacce, mentre le possibili soluzioni e le **tecniche** più significative sono solo fugacemente accennate. Le tecniche di interfacciamento di segnali digitali ed analogici verranno presentate più concretamente ed estesamente rispettivamente nei cap.7 e 8.

Gli stati sono situazioni che durano nel tempo, gli eventi sono (considerati) istantanei.
I calcolatori per acquisire informazioni dal mondo esterno procedono per campionamenti successivi (periodiche osservazioni dello stato). Per sua natura quindi il calcolatore **non puo` a livello SW rilevare direttamente gli eventi, ma solo gli stati dei segnali**.

Se l'informazione è associata allo stato del segnale il calcolatore la può acquisire direttamente; se invece è associata agli eventi del segnale, occorrerà adottare delle tecniche particolari.

Sostanzialmente le tecniche per l'acquisizione di eventi sono le seguenti.

- Si attribuisce a dei dispositivi ausiliari (circuiti di memoria come bistabili, contatori, ecc.) la trasformazione di eventi in stati (per essere acquisibili dal calcolatore). Questi dispositivi potranno poi essere **passivi** se destinati ad una lettura da parte del calcolatore a controllo di programma oppure essere resi **attivi** ed in grado di inoltrare richieste di interruzione o di DMA.

- Si desumono gli eventi mediante elaborazioni degli stati acquisiti con campionamento (soluzione software per il rilievo degli eventi).
Si adotta in questi casi un algoritmo opportuno, che dalla successione temporale di stati (campionati) deduce se e quali eventi si sono verificati.

Tra stati ed eventi esiste un legame che assomiglia un po' a derivata/integrale (eventi = variazione degli stati). I dispositivi circuitali di cui si è parlato agiscono in un certo senso da integratori (ricevono degli eventi e assumono stati di volta in volta variabili in conseguenza di questi eventi), viceversa l'algoritmo software che ricava gli eventi dagli stati, agisce come una specie di derivatore (rilevatore di differenze).
Nel progettare un sistema per l'acquisizione (o emissione) di stati ed eventi occorre considerare attentamente alcuni importanti aspetti legati in generale alle specifiche del sistema ed in particolare alle specifiche temporali, come analizzato nei paragrafi seguenti.

5.3.1 ACQUISIZIONE DI INFORMAZIONI ASSOCIATE A STATI

La tecnica tipica per l'acquisizione di informazioni di stato è basata sulla lettura diretta, con istruzioni di INPUT, dei valori binari presentati su una porta di ingresso, eventualmente dopo conversione A/D (analogico / digitale).

Le informazioni di stato vengono infatti acquisite su iniziativa del calcolatore, quando l'esecuzione del programma lo richiede o in istanti di tempo (regolari) dettati da un orologio interno che impone il periodo di campionamento.

Il periodo di campionamento è una scelta di progetto i cui criteri ed importanza dipendono dalle caratteristiche del segnale, ma anche da come sono rappresentate dal segnale stesso le informazioni significative, e vengono brevemente presentati nel seguito, notando che anche in una stessa applicazione sui diversi segnali vengono specificati requisiti diversi.

E' tanto importante quanto ovvio notare che le specifiche temporali portano a problemi di elaborazione in tempo reale e la loro fisica attuabilità dipende dalla velocità (potenza di calcolo) del calcolatore, dal suo carico di lavoro e dal tipo di circuiti di interfaccia di cui è dotato.

Vediamo ora le **tipiche specifiche che si possono imporre in relazione all'acquisizione di stati**.
Si invita il lettore a riflettere sulle parziali sovrapposizioni, ma anche sulle differenze concettuali, delle diverse specifiche.

- RITARDO MASSIMO DI RICONOSCIMENTO DI UNO STATO

Scopo di questa specifica è garantire che il calcolatore reagisca in tempo, con altre acquisizioni o azioni, al verificarsi di uno stato.
E` chiaro che un vincolo sul massimo ritardo con cui il calcolatore deve recepire lo stato di un segnale di ingresso costituisce in realta` un vincolo sul periodo di campionamento (T). **T deve essere minore del massimo ritardo ammesso**.

- DURATA MINIMA DEGLI STATI DA RILEVARE

Scopo di questa specifica è di non perdere stati utili, o di riconoscere impulsi significativi, e riconoscere eventuali stati spurii.
Per essere sicuri che almeno uno dei campioni acquisiti dal calcolatore cada all'interno dell'intervallo di permanenza minima dello stato occorre che **T sia minore di questo tempo minimo**.

Specificare che il tempo di durata di uno stato deve essere superiore ad un certo valore minimo significa assegnare l'etichetta "spurio" ai fenomeni del segnale in esame di durata inferiore a questo tempo.
"Spurio" = non portatore di informazione utile.

- ELIMINAZIONE DI SPURII

Le "false" informazioni associate agli spurii possono essere accettate se non producono effetti indesiderati, ma spesso dalle specifiche dell'applicazione si deduce che tali informazioni **devono** essere esplicitamente eliminate.
Si possono adottare in questi casi provvedimenti di "filtraggio" hardware o software.
Il periodo di campionamento T dovrà in questi casi essere una frazione (es. 1/2, 1/4, ecc.) del tempo minimo specificato come significativo.

- FREQUENZA (MINIMA) DI RILEVAMENTO DI UNO STATO

Questa specifica mira a garantire la riproduzione adeguatamente fedele dell'andamento temporale dello stato.
Con stati analogici (continui e tempo continui) questa specifica può essere dettata dal teorema di Shannon con un opportuno margine (circa 10 . . 100 volte).
Si noti che se chiamiamo *SR* il massimo *slew-rate* (derivata rispetto al tempo) del segnale e *Eam* il massimo errore assoluto accettabile come scostamento in ogni istante tra il valore dell'ultimo campione e il valore corrente del segnale, dovremo scegliere un periodo di campionamento $T < Eam/SR$
La situazione è rappresentata nella fig.5.8.

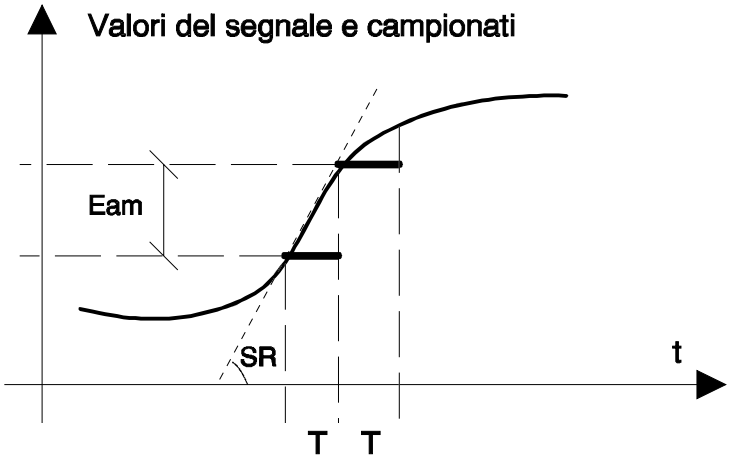

Fig. 5.8 - Relazione tra errore assoluto max Eam, *slew-rate* max SR e periodo di campionamento T.

In generale la frequenza di campionamento di uno stato dipenderà da varie considerazioni, legate ad es. al tipo di algoritmi da eseguire, che possono suggerire frequenze di campionamento maggiori di quelle strettamente necessarie in base alle considerazioni precedenti (ad es. rilievo di spurii, media su N campioni, ecc.).

- PRECISIONE DELLA BASE DEI TEMPI DI CAMPIONAMENTO

Con riferimento a **stati continui e tempo continui** può essere importante una precisione dei tempi assoluti (cioè sul lungo periodo) se si eseguono operazioni di integrazione sui valori di stato come sommatorie dei valori campionati, mentre è importante la stabilità

del clock che assicura una precisione dei tempi relativi (cioè sul breve periodo) per il calcolo di derivate come rapporto incrementale tra campioni successivi.

Si noti comunque che con l'adozione di *time-stamping* (associazione ad ogni valore di campione anche del tempo all'effettivo istante del campionamento) si può ottenere una buona approssimazione dei calcoli di integrali e derivate anche in presenza di campionamenti relativamente irregolari nel tempo.

Per quanto riguarda gli **stati discreti e tempo continui** (ad es. impulsi modulati in durata) la precisione del periodo di campionamento è determinante se si devono effettuare misure di durata degli stati stessi per ricavare l'informazione "analogica" associata a tale durata.

- CORRELAZIONE TRA DIVERSI SEGNALI

Consistenza temporale.
Se lo stato (che si suppone sia variabile nel tempo) da acquisire è descritto da un numero di bit maggiore di quello (generalmente 8) che può essere letto con un'operazione atomica, occorre effettuare una cattura (*latch*) di tutti i bit su appositi registri che poi potranno essere letti anche per parti.
Successione temporale.
In alcuni casi la granularità temporale (e quindi la frequenza di campionamento) richiesta per singoli segnali può essere relativamente grossolana, ma può essere necessario riconoscere precedenze o concomitanze temporali tra diversi segnali. Specifiche stringenti su questi aspetti richiedono in genere di ricorrere a soluzioni circuitali che consentono granularità anche di pochi nanosecondi, cioè 3..4 ordini di grandezza più fini delle granularità tipiche dei meccanismi software.

5.3.2 EMISSIONE DI INFORMAZIONI DI STATO

L'emissione di informazioni di stato, effettuata con istruzioni di OUT su porte di uscita, pone in genere meno problemi rispetto all'acquisizione.
Le specifiche di temporizzazione su queste operazioni sono generalmente le seguenti.

- FREQUENZA DI AGGIORNAMENTO DELLO STATO

Analogamente a quanto visto per gli ingressi, la frequenza di emissione di un valore di stato dipende sostanzialmente dalla granularità temporale necessaria per riprodurre fedelmente l'evoluzione dello stato con la richiesta velocità di variazione.

- PRECISIONE DELLA BASE TEMPI

Si allude qui alla base tempi che comanda l'esecuzione delle operazioni di emissione (output) dei valori di stato.
Anche questo parametro giuoca un ruolo importante nella **fedeltà** di riproduzione del segnale voluto, che viene ovviamente degradata in presenza di *jitter*.

- CORRELAZIONE TRA DIVERSI SEGNALI

In uscita questo aspetto può richiedere particolare attenzione quando l'informazione da emettere è rappresentata con un numero di bit maggiore di quello (spesso 8) gestito da un'operazione elementare di output.
In questi casi sono richieste successive operazioni di scrittura, solo al termine delle quali l'informazione assume un valore significativo.

Se non è accettabile, come spesso avviene, che si abbiano degli intervalli, sia pure di pochi microsecondi, in cui il valore sulle uscite è non significativo, occorre adottare tecniche circuitali che consentano di separare la "predisposizione" su registri temporanei dalla effettiva "emissione" delle informazioni in uscita.

5.3.3 INFORMAZIONI ASSOCIATE AD EVENTI

Come si è precedentemente accennato gli eventi possono essere rilevati **con tecniche algoritmiche** (SW) deducendoli dalle **variazioni degli stati dei segnali** che sono stati acquisiti mediante campionamento.
Quando invece si vogliano "sentire" direttamente gli eventi, la tecnica tipica è costituita dall'uso **dell'interrupt**, mentre semplici eventi, eventualmente sporadici o in rapide successioni, possono essere sentiti ed acquisiti con la tecnica del **DMA**.

Per eventi **complessi**, che si verificano in istanti casuali e comunque **poco frequent***i* (fino a qualche migliaio al secondo) il meccanismo dell'interrupt costituisce spesso la soluzione più efficace.
Quando gli eventi sono **semplici**, cioè non richiedono particolari elaborazioni da eseguirsi immediatamente, e si presentano in **rapide sequenze** ("*burst*") è necessario l'uso del DMA o di altre tecniche HW con memorie incorporate (contatori, code FIFO, ecc.).
Nel caso di eventi **complessi in rapide successioni** (come ad esempio la ricezione seriale in reti di comunicazione ad elevata velocità) si dovrà ricorrere ad un **processore di interfaccia dedicato**.

I problemi che si pongono nell'acquisizione o emissione di informazioni associate agli eventi sono i seguenti:

- NON SI DEVONO PERDERE EVENTI

Questo requisito è importante soprattutto per eventi **incrementali**, ma anche per eventi **assoluti**. (v. cap. 2)
Poiche` gli eventi non sono campionabili direttamente come tali, occorre far si che gli stati, da cui tali eventi siano deducibili, siano acquisiti con un campionamento abbastanza frequente per accorgersi se c'e` stata una commutazione. Due commutazioni in rapida successione rischiano di essere perse (perdita dell'evento).

Esiste quindi una **soluzione algoritmica SW** solo nel caso in cui un evento non viene seguito da un evento che lo annulla se non dopo un certo tempo minimo superiore all'intervallo di campionamento, cioè solo con eventi **sporadici** con **tempo morto** superiore al massimo intervallo di campionamento. Si ricordi che un breve periodo di campionamento può produrre un elevato carico di lavoro sulla CPU.

Un altro modo per risolvere il problema di non perdere eventi è quello di affidare ad un dispositivo HW esterno il compito di catturarli trasformandoli in stati (integratore); si tratta di solito di dispositivi che al verificarsi di un evento commutano lo stato e hanno la caratteristica di mantenere questo stato stabile (non viene annullato da altri eventi).

Un tipico esempio sono i "flip-flop", circuiti che realizzano delle reti sequenziali e come tali presentano all'uscita uno stato che dipende non solo dal valore dell'ingresso in quel

momento, ma anche dalla storia precedente. Se nella storia precedente si e` verificato un certo evento il circuito assume uno stato che ricorda questo evento.
Questa soluzione può essere estesa ad una "coda" (FIFO) di memoria di eventi.
Ovviamente il periodo di lettura dello stato di questa rete sequenziale deve essere minore dell'intervallo dopo cui può verificarsi un evento non più memorizzabile.
In alternativa si può fare in modo che lo stato generi richieste di interrupt.

- NON SI DEVONO DUPLICARE O INVENTARE EVENTI

Anche questo requisito è importante soprattutto per eventi ad informazione **incrementale**.
Questo problema può sorgere proprio a causa delle tecniche utilizzate per evitare la perdita di eventi, mediante la loro memorizzazione in reti sequenziali. Infatti letture successive di uno stato che indica **evento verificato** porterebbero alla falsa rilevazione di diversi eventi se non si procede ad una memorizzazione anche del fatto che l'evento è già stato **acquisito**.

Ciò richiede che ad ogni segnale che costituisce un flusso di eventi sia associato un comportamento rappresentabile mediante un automa con stati appartenenti al seguente insieme:

NV evento non verificatosi
VNR evento verificatosi ma non riconosciuto
VR evento verificatosi e riconosciuto

Si noti che spesso il riconoscimento di ogni evento conclude la storia da memorizzare, cioè lo stato VR può essere considerato equivalente allo stato di riposo NV.
In questo caso è tipico adottare un bistabile con comando di SET attivato dal verificarsi dell'evento e comando di RESET attivato dall'operazione di riconoscimento (lettura).

Le tecniche ora citate hanno in sostanza lo scopo di instaurare un corretto rapporto *produttore-consumatore* tra i fenomeni esterni generatori di eventi e le attività interne al calcolatore che acquisisce tali eventi.

- RICONOSCIMENTO DI SEQUENZE DI EVENTI

In alcuni casi si possono porre problemi di riconoscimento della sequenza temporale con cui si verificano eventi su diversi segnali (flussi). Un tipico esempio è costituito dal riconoscimento del cosiddetto "*primo fuori*", cioè l'evento di **anomalia** in un impianto che si è verificato per primo e che spesso è seguito nel tempo da altre **anomalie indotte**.
Il riconoscimento della **causa**, che ovviamente precede gli **effetti**, richiede di risolvere questo problema per cui sono talora richieste tecniche circuitali (memorie FIFO) se la risoluzione temporale necessaria è molto fine.

5.4 RAPPRESENTAZIONI INTERNE DI STATI E DI EVENTI

Lo scopo principale dell'interfacciamento di un sistema di elaborazione con il mondo esterno è quello di creare una descrizione interna, e quindi elaborabile, dei fenomeni esterni, che ne riproduca le informazioni significative, eventualmente ricostruite a partire da quelle direttamente acquisibili.

Nel seguito si faranno spesso riferimenti ai concetti di stato ed evento, come presentati nel cap. 2.

5.4.1 IMMAGINI DI STATO

Per immagine di stato si intende un valore che rappresenta lo stato corrente di un'entità considerata esterna. Tale valore deve venire aggiornato con opportuna, anche se talora irregolare, cadenza per rispettare specifiche di massimo scostamento ammissibile rispetto al valore istantaneo dello stato rappresentato (v. precedente sezione: Acquisizione di informazioni associate a stati e fig.5.8).

Per le immagini di stato si utilizzano generalmente variabili con classe di allocazione *statica*, cioè persistenti, e spesso globali e condivise, a cui viene assegnato un nuovo valore (sono sovrascritte) ad ogni acquisizione dal mondo esterno.
I **tipi** delle variabili sono generalmente scelti come segue.

`char, unsigned int`	Stati che costituiscono insiemi discreti enumerativi.
`int,long int`	Stati con valori numerici interi
`float,double`	Stati con valori astrazione del continuo

Per queste variabili si instaura il tipico rapporto, detto "**writer-readers**" tra il processo di acquisizione dei segnali dal mondo esterno, che assegna loro il valore (*writer*), ed i processi che utilizzano il valore per le loro elaborazioni (*readers*), cioè un rapporto a priori totalmente asincrono.
Per garantire la consistenza delle informazioni rappresentate occorre rendere **atomiche** le operazioni di sovrascrittura (eventualmente con mutua esclusione tra chi scrive e chi legge).

Può essere utile associare ad ogni immagine di stato un indicatore di "freschezza" (*freshness*), che consente cioè di valutare se lo stato è aggiornato secondo le specifiche.
Tale indicatore può essere:
- valore *boolean* per semplici convalide di freschezza
- *time-stamp* per convalida e calcoli di derivate / integrali di valori continui.

Talvolta occorre mantenere-verificare la "coerenza temporale" tra diverse variabili di stato (importante per stati discreti, desiderabile per stati continui a larga banda, cioè veloci).
Ciò risulta importante quando diverse variabili di stato sono utilizzate per uno stesso calcolo, come ad esempio nella valutazione indiretta di una potenza elettrica istantanea calcolata dai valori istantanei di tensione e corrente: p(t1) = v(t1)*i(t2). Chiaramente la differenza (in parte inevitabile) tra t1 e t2 produrrà errori sul valore di potenza calcolata tanto maggiori quanto maggiore è il rapporto

abs(t2-t1) / Pmin

dove Pmin è il periodo della massima frequenza dello spettro di i(t) o di v(t).

5.4.2 IMMAGINI DI EVENTO

Gli eventi sono in genere più complessi da trattare rispetto agli stati e presentano una casistica più articolata. Ricordiamo che in alcuni casi una successione di eventi di campionamento deve essere utilizzata come descrizione di un andamento temporale di

uno stato, richiedendo in ultima analisi la memorizzazione più o meno permanente di tutti gli eventi. In altri casi ogni singolo evento esaurisce la sua funzione quando è recepito da un processo consumatore.
Per semplicità unificheremo le due situazioni sopra citate, supponendo che si realizzi in ogni caso un processo consumatore di eventi e attribuendo ad esso eventuali ulteriori compiti di memorizzazione di intere successioni di tali eventi.
Con questa ipotesi ogni immagine di evento costituisce un punto di sincronizzazione tra il processo che rileva gli eventi esterni ed il processo che ne utilizza le informazioni associate. Le immagini di evento assumono la forma di messaggi ad un destinatario che generalmente è unico per motivi di semplicità e generalità (è complicato, con tanti destinatari, tener conto di chi ha o non ha ricevuto l'evento).

Un'immagine di evento **assoluto** può essere realizzata con una *variabilie aggregata di tipo record* (o `struct`) con i seguenti campi:

[valore stato precedente .. facoltativo]
valore del nuovo stato
situazione dell'evento (da consumare / consumato)
[time stamping .. facoltativo]

Analogamente, per l'immagine di eventi **incrementali** potremo adottare la struttura:

valore dell'evento
situazione dell'evento (da consumare / consumato)
[time stamping .. facoltativo]

Se gli eventi sono di tipo **incrementale** e non sono "collassabili" (cioè non si possono perdere) occorre instaurare tra il processo che genera la descrizione interna di un evento ed il processo che la utilizza, il tipico rapporto "**producer-consumer**" con una delle tecniche seguenti:
- *sincronizzazione stretta* tra produttore e consumatore (come il classico *rendez-vous* del linguaggio Ada) con cui si evita la perdita di eventi se è verificata la relazione tra gli intervalli di consumo e di produzione:

DT_consumo_max < Dt_eventi_min

- *sincronizzazione lasca*, mediante accodamento delle informazioni associate agli eventi in coda o in buffer circolare, con cui si evita la perdita di eventi se è verificata la relazione:

DT_consumo_med < DT_eventi_med

La situazione di un "flusso di eventi" può essere modellizzata mediante un **automa** a stati finiti (fig.5.9) che gestisce il campo *situazione* della variabile aggregata sopra proposta. Una versione abbastanza completa e generale di tale automa può essere quella presentata nel seguito e i cui **simboli di ingresso** sono:

P produzione di evento corretto
Pee produzione di evento errato
C consumo di evento
Re riconoscimento di errore

ed i cui **stati** sono:

R riposo
Ev evento pendente (ed eventualmente quanti)
Er errore
O overrun (eventi persi)

U underrun (eventi duplicati)

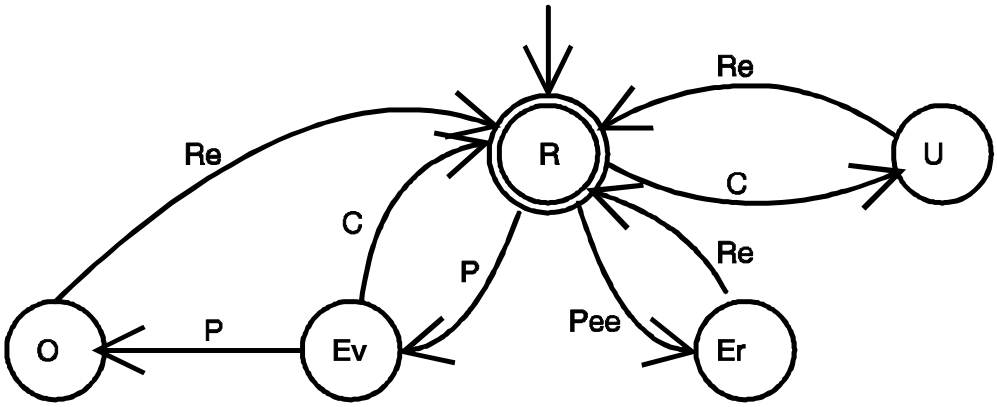

Fig. 5.9 - Modello ad automa di acquisizione di flusso di eventi.

Spesso è sufficiente un automa ridotto ai soli simboli di ingresso P e C e stati riposo (R), evento pendente (Ev) ed eventualmente overrun(O).

Se invece gli eventi sono di tipo **assoluto**, il flusso di eventi dal processo generatore può in alcuni casi essere finalizzato esclusivamente alla ricostruzione di un'immagine dello stato, generalmente corredata di indicatori di aggiornamento e *time-stamp*, ricadendo così nel rapporto *writer-reader(s)* detto sopra per le immagini di stato.

5.5 TEMPORIZZAZIONE E SINCRONIZZAZIONE DI ACQUISIZIONE ED EMISSIONE DI INFORMAZIONI

La funzionalità all'interno del calcolatore è determinata dall'esecuzione delle istruzioni, cosa che richiede un certo tempo in dipendenza dalla rapidità dei circuiti elettronici, dalla frequenza di clock, dal tipo di istruzioni in esecuzione.

In sostanza la velocità di esecuzione del calcolatore dipende da tutta una serie di fenomeni interni che nulla hanno a che vedere con quanto succede nel mondo esterno in cui i fenomeni fisici evolvono con una loro dinamica.

Si tratta cioè di due mondi **asincroni** che necessitano quindi di opportune sincronizzazioni al fine di superare correttamente la disomogeneità temporale di cui si è accennato all'inizio di questo capitolo.
Le **sincronizzazioni** riguardano **eventi** ed **azioni** (solitamente acquisizioni o emissioni di dati) che sarebbero intrinsecamente scorrelati e a cui si vuole **imporre una relazione di causa-effetto** tramite un'opportuna catena di eventi-azioni intermedi. Nei normali calcolatori sono generalmente disponibili come dotazione standard alcuni elementi adatti per queste catene, generalmente noti come meccanismi di interrupt e di DMA.

Si noti che si può distinguere tra due diverse esigenze di sincronizzazione:
- Sincronizzazione **dell'acquisizione di informazioni relative ad un evento esterno** con il verificarsi di **tale evento**. L'evento in questo caso è considerato come indicatore di "valore significativo" da acquisire.

- Sincronizzazione **dell’acquisizione di informazioni di uno stato** esterno con il verificarsi di **eventi temporali**. Si parla in questo caso più specificamente di **temporizzazioni**, come è il caso del campionamento.

In entrambi i casi ci si riconduce in genere agli stessi meccanismi, poichè nelle normali implementazioni gli eventi temporali si manifestano in modo del tutto analogo agli altri eventi esterni, cioè come commutazioni tra stati.

Si noti inoltre che la catena di sincronizzazioni può talora prolungarsi, ma che in genere sono significativi almeno due diversi livelli di sincronizzazione:

- Sincronizzazione di **primo livello** tra **evento** e **trasferimento dei dati associati**, come visto sopra. Per questa operazione la *deadline* (scadenza temporale) è dipendente da quanto tempo i dati generati dall’evento restano disponibili e validi.
- Sincronizzazione di **secondo livello** tra **trasferimento dei dati** e loro **elaborazione**. La *deadline* per questo secondo livello si riferisce alla generazione, mediante opportuna elaborazione, della risposta allo stimolo costituito dall’evento.

Scopo di un'interfaccia è quello di effettuare trasferimenti di informazioni, che per essere significativi devono avvenire con il rispetto di precisi vincoli temporali e con la corretta gestione di rapporti produttore-consumatore tra il sottosistema che genera eventi ed il sottosistema che li deve recepire.

Come è noto le funzioni di interfaccia nei calcolatori sono affidate in parte ad appositi circuiti (HW) ed in parte all’esecuzione di programmi (SW). Una buona suddivisione dei compiti tra HW e SW in base alle loro caratteristiche peculiari e complementari, consente di rispettare le specifiche di progetto e minimizzare i costi complessivi.

5.5.1 PARAMETRI DI VALUTAZIONE DEI MECCANISMI DI I/O

- - TEMPO DI REAZIONE (Latenza)

E' il tempo che intercorre tra l'istante in cui si verifica un evento significativo e l'istante in cui il calcolatore è in grado di acquisire e trattare le informazioni associate.

- - VELOCITA' DI TRASFERIMENTO (*Transfer rate*)

E' la massima frequenza di trasferimento di informazioni e prescinde da latenze e *overhead* iniziali.

- - FLUSSO DI INFORMAZIONI (*Throughput*)

E' la frequenza media (sul lungo periodo) con cui possono essere effettuati trasferimenti e tiene conto delle latenze e *overhead*.

- - EFFICIENZA COMPLESSIVA

Dà un'idea di quanto le operazioni di acquisizione ed emissione di dati degradano le prestazioni relative all'esecuzione dei programmi di elaborazione interna, cioè del carico imposto alla CPU dalle operazioni di sincronizzazione e trasferimento.

- - COMPLESSITA' E COSTO

Si tende ovviamente a minimizzare questi aspetti, nel rispetto delle specifiche del progetto.

5.5.2 TECNICHE DI SINCRONIZZAZIONE DEI TRASFERIMENTI

MECCANISMI	SINCRONIZZAZIONE	TRASFERIMENTO
A controllo di programma	sw	sw
Ad interruzioni (interrupt)	hw	sw
Accesso diretto a memoria (DMA)	hw	hw

Il più semplice meccanismo di sincronizzazione è quello a controllo di programma realizzato con cicli di attesa o comunque con test periodici (detti *polling*), che ottiene sia la sincronizzazione sia il trasferimento dell'informazione tramite istruzioni di programma (tecnica di tipo software).

Il meccanismo ad interrupt effettua a livello circuitale (tecnica hardware) la sincronizzazione tramite componenti che recepiscono l'istante in cui ha senso effettuare il trasferimento che viene però realizzato tramite l'esecuzione di istruzioni di programma (routine di servizio dell'interrupt).

Infine per i meccanismi ad accesso diretto della memoria si utilizzano appositi circuiti che si occupano sia della sincronizzazione con gli eventi esterni che del trasferimento delle informazioni associate.

Varianti del DMA possono essere considerati i cosiddetti *canali* in cui dei processori specializzati sono dedicati ad effettuare i trasferimenti con le dovute sincronizzazioni.

E' immediato notare, nell'elenco di meccanismi sopra citati, una progressiva diminuzione di importanza degli aspetti software ed una tendenza opposta per gli aspetti hardware; il bilanciamento diverso tra le componenti HW e SW può derivare dalle seguenti considerazioni generali:

- Il SW è economico da riprodurre mentre è abbastanza costoso da sviluppare mentre per l'HW è il contrario.

- Il SW è più flessibile e quindi consente maggiori facilità di adattamento a situazioni diverse.

- Il SW esprime azioni eseguite in modo sequenziale e quindi è relativamente lento mentre l'HW può essere molto più veloce (da 100 a 10000 volte circa) e molto più accurato nei tempi.

Esiste quindi un bilancio tra aspetti positivi e negativi e il progettista è chiamato a scegliere per una particolare applicazione quali sono gli aspetti principali di cui tener conto: tipicamente per applicazioni economiche che non hanno grosse esigenze temporali e richiedono una certa flessibilità risulterà preferibile un meccanismo a controllo di programma mentre per prestazioni più elevate si utilizzerà un meccanismo ad interrupt o a DMA.

Analizziamo ora un po' più nei dettagli i vari meccanismi.

5.5.2.1 MECCANISMO A CONTROLLO DI PROGRAMMA

Questo meccanismo è così detto perchè è la successione delle istruzioni del programma che determina i tempi di esecuzione, e quindi deve essere organizzata in modo da garantire la corretta sequenza tra il rilievo di un evento, l'azione di trasferimento delle informazioni (sincronizzazione di primo livello) e l'azione di elaborazione (secondo livello).

Meccanismo.

La sincronizzazione di primo livello è basata sull'acquisizione periodica (*polling*) di informazioni di stato da cui si risale al verificarsi di eventi. L'acquisizione periodica può assumere la forma di un ciclo di attesa che si ripete fino a quando la verifica dà esito positivo se si può dedicare completamente la CPU a questo scopo, altrimenti si adotta la soluzione di eseguire a cadenza temporizzata una sequenza di valutazioni di funzioni booleane degli ingressi che con il loro valore di verità indicano quali eventi sono verificati e richiedono il relativo servizio.

La sincronizzazione è vista come la verifica che sia giunto il momento opportuno per eseguire una certa operazione; tale istante temporale dovrà essere caratterizzato dall'esistenza di qualche fenomeno recepibile: caso tipico è quello in cui si hanno nelle interfacce dei registri di stato i cui bit sono dedicati appunto a queste segnalazioni.

La catena **cause-effetti** che si realizza è a grandi linee la seguente.

HW - L'evento esterno porta appositi circuiti in uno stato.
SW - Il programma esegue istruzioni di lettura dello stato.
SW - Il programma deduce dallo stato il verificarsi dell'evento.
SW - Il programma esegue l'azione (ad es. lettura di un valore).
SW - Il programma esegue le elaborazioni del valore.

Un pregio di questa tecnica è la innegabile semplicità del meccanismo elementare, mentre tra i difetti bisogna far notare i seguenti.

Scarso sfruttamento del tempo macchina.

Deriva dal fatto che si dedica ripetutamente il tempo della CPU a valutare se si sono verificati eventi, che nei casi di eventi sporadici potrebbero verificarsi anche molto saltuariamente.

Possibilità di non rispettare le specifiche temporali e perdere eventi.

Se si riduce il carico di lavoro della CPU rallentando la cadenza di rilievo degli eventi, si introduce una potenziale latenza di rilievo che può essere aggravata se viene rilevato, e quindi servito, un evento meno urgente di altri. In generale il tempo di reazione agli eventi dipende dal periodo di attivazione delle azioni di rilievo degli eventi stessi.

Questo meccanismo rimane utilizzabile anche quando un solo processo SW debba tenere a bada diversi fenomeni concorrenti, ma a costo di una non trascurabile complessità e soprattutto di pesanti rinunce ad una buona struttura del software.

Le applicazioni per cui la tecnica del controllo di programma consente di coniugare semplicità e buone prestazioni, sono tipicamente quelle in cui l'attività del calcolatore può ricondursi ad una ciclica e regolare ripetizione delle azioni di *lettura ingressi* -

elaborazioni - scrittura risultati. Questo approccio è tipico dei PLC (Programmable Logic Controller), ma può essre adottato con successo in numerose applicazioni di acquisizione dati e controllo e viene ulteriormente analizzato nel cap.10.

5.5.2.2 MECCANISMO AD INTERRUPT

Questa tecnica prevede dei dispositivi di interfaccia che comunicano a livello circuitale al processore che si è verificato un evento da acquisire.

Meccanismo.

La comunicazione avviene mediante generazione di una richiesta di interruzione che ha l'effetto di interrompere l'esecuzione del programma che il processore sta svolgendo per attivare l'esecuzione di un **sottoprogramma di servizio** (Interrupt Service Routine - ISR).

All'interno di tale sottoprogramma vi saranno delle istruzioni software, tipicamente trasferimenti di informazioni, che il meccanismo hardware permette di attivare in modo sincronizzato (primo livello) con il verificarsi dell'evento esterno.

Mentre è indispensabile che la routine di risposta all'interrupt salvi e ripristini il contesto di esecuzione in corso, sarà in genere necessario prevedere dei meccanismi mediante i quali comunicare l'esito delle operazioni eseguite (sincronizzazione di secondo livello con i processi), con il **posizionamento di indicatori** (*flag*) o mediante **primitive del sistema operativo** di tipo "*signal*".

Generalmente **si associa un canale di interrupt ad ogni flusso di eventi da "sentire".**

I diversi tipi di processori sono dotati al loro interno, o con circuiti controllori di interrupt esterni, di meccanismi di gestione delle interruzioni con diversi gradi di complessità e diverse caratteristiche che vengono presentate brevemente nel seguito.

Mascherabilità.

In genere i processori consentono di abilitare o disabilitare (mascherare) selettivamente le singole cause di interrupt. Molti processori sono inoltre dotati di un particolare interrupt non mascherabile (NMI = Not Maskable Interrupt) destinato ad eventi per cui è essenziale minimizzare la latenza. L'uso di tale interrupt richiede particolari attenzioni, dato che risulta abilitato fin dalle fasi iniziali (*power-up*) in cui le varie interfacce hanno ricevuto il segnale di *reset* ma non sono ancora completamente inizializzate.

Priorità.

Le priorità hanno lo scopo principale di dirimere i conflitti di richieste di interruzione contemporanee da parte di diversi eventi.

Spesso le priorità dei vari interrupt sono rigidamente prefissate, ma nei processori di controllo è in genere possibile una loro riconfigurazione, sia pure parziale.

Vettorizzazione.

La vettorizzazione è il meccanismo con cui si attiva per ogni interrupt una specifica routine di servizio. La vettorizzazione fissa richiede che le routine di servizio inizino ad indirizzi prefissati, mentre la vettorizzazione programmabile consente di scrivere in opportune celle di memoria (quindi di RAM) gli indirizzi effettivi di inizio delle routine di servizio. In questo secondo caso (tipico ad esempio dei processori Intel 80x86) è possibile e più flessibile la riconfigurazione delle modalità di servizio dei vari interrupt.

Annidamento.

Spesso i circuiti di gestione delle richieste di interruzione sono in grado di mascherare automaticamente le richieste di livello di priorità inferiore o uguale a quello in servizio in un certo momento. Con questi controllori è possibile adottare l'approccio di interruzioni annidate, cioè la possibilità che una richiesta più prioritaria possa interrompere il servizio di una meno prioritaria. Naturalmente con questo approccio le routine di risposta alle interruzioni dovranno essere progettate con maggiore attenzione ai problemi di rientranza, mutua esclusione, ecc.

Cambiamento di contesto.

I processori progettati per applicazioni real-time sono spesso dotati di diversi banchi di registri dedicati a diversi livelli di interrupt, con commutazione automatica al momento dell'interruzione, in modo da ridurre *l'overhead* connesso con il necessario salvataggio del contesto.

La tipica catena **cause-effetti** che si realizza nella gestione di un interrupt è a grandi linee la seguente.

a livello HW

- - **l'evento** pone a TRUE lo stato di un indicatore di richiesta nell'interfaccia
- - un meccanismo di arbitraggio inoltra la richiesta alla CPU
- - la CPU campiona prima di ogni *fetch* lo stato di richiesta
- - la CPU emette un segnale di conferma
- - la CPU disabilita la sensibilità ad ulteriori richieste
- - la CPU riconosce l'eventuale vettore di interrupt
- - la CPU salva (sullo *stack*) il contenuto del Program Counter (e di PSW=flag di stato)
- - la CPU carica nel Program Counter l'indirizzo della routine di servizio (vettore)

a livello SW (routine di servizio)

- - salvataggio dei registri della CPU + contesto
- - [riabilitazione dell'interrupt - se annidati]
- - individuazione della causa dell'interrupt (flusso di eventi)
- - **azione** (gestione dell'evento e trasferimenti delle informazioni associate)
- - rimozione della richiesta
- - impostazione degli indicatori o invocazione della primitiva *signal*
- - [segnalazione di fine servizio - se annidati]

conclusione SW

- - [eventuale cessione del controllo ad uno scheduler]

oppure

- - ripristino dei registri della CPU + contesto
- - riabilitazione dell'interrupt
- - ritorno al programma in esecuzione

gestione di secondo livello SW

- - il processo interessato trova gli indicatori impostati, e da ciò deduce di poter procedere con le elaborazioni del caso

NOTA
Spesso è conveniente organizzare la routine di risposta ad un certo interrupt, come **automa a stati** o comunque dotato di un proprio contesto (variabili statiche che sopravvivono tra un'esecuzione e la successiva).
Si noti che in questo modo la routine di risposta all'interrupt **assume la veste di un vero e proprio processo** che si differenzia da quelli "normali" perchè il meccanismo di attivazione (*dispatching*) è HW, invece che SW come avviene per i processi gestiti dal Sistema Operativo.

Questa soluzione è interessante quando la successione di eventi da rilevare presenta una propria **sintassi** che debba essere verificata rapidamente, cioè all'interno della stessa routine di servizio dell'interrupt, per eventuali reazioni urgenti.

Un esempio tipico di quanto appena detto è costituito da sequenze di caratteri che formano messaggi da ricevere con il meccanismo dell'interrupt, e su cui è opportuno effettuare, già a basso livello, verifiche di sequenza (caratteri di inizio/fine messaggio), corretta parità, calcolo del checksum, ecc..

Un classico (in alcuni sistemi l'unico) utilizzo dell'interrupt è correlato con l'uso di **temporizzatori**: questi vengono in genere utilizzati per attivare, con opportuna periodicità, operazioni come i campionamenti di informazioni di stato, oltre che, naturalmente, per mantenere aggiornato il valore di una variabile “tempo corrente”.

Si noti che l'attivazione periodica di una routine (o meglio di un processo), mediante interrupt generato da timer, costituisce in molti casi un interessante compromesso e connubio tra i vantaggi dell'approccio ad interrupt e quelli del controllo di programma. Si realizza cioè un *polling time-driven* invece che *execution-driven*.

I tempi da considerare con il meccanismo di interrupt sono:
- **latenza** = tempo perchè la richiesta possa essere recepita che dipende dalla durata delle disabilitazioni dell'interrupt dovute a:
  - disabilitazione per protezione (mutua esclusione)
  - disabilitazione per altro servizio di interrupt
- **tempo di attivazione** = tempo necessario per le operazioni preliminari HW e SW (salvare contesto, individuare la causa)
- **tempo di servizio** = tempo per l'esecuzione del SW specifico di servizio
- **latenza di segnalazione** = tempo che trascorre prima che il processo interessato recepisca l'informazione che si è verificato l'interrupt (sincronizzazione di secondo livello).

Il tempo di attivazione di un interrupt è dell'ordine di 1..50 microsecondi; il tempo di esecuzione di una routine di risposta all'interrupt può essere dell'ordine delle centinaia di microsecondi.

E' intuitivo che non ha molto senso, in generale, avere interrupt che hanno una cadenza molto superiore al migliaio di interruzioni al secondo.

Ciò comporta ad esempio che con processori poco potenti la gestione di linee seriali a velocità superiori a 9600 baud (bit/s) di solito non è effettuata ad interrupt (anche se in

alcuni casi si potrebbe arrivare fino a 19200 baud). Se si vogliono ottenere velocità superiori si decide tra la gestione a controllo di programma o il meccanismo del DMA.

Un'analisi accurata dei tempi del meccanismo di interrupt è alla base di decisioni di progetto relative a quali operazioni effettuare nell'ambito della routine di servizio e quali demandare al (o ai) processo che da tale routine può essere attivato. In genere è conveniente effettuare nella routine di risposta solo le operazioni urgenti (trasferimenti delle informazioni, verifiche, accodamenti) per minimizzare i tempi di latenza di altre interruzioni.

Il controllo di programma, pur penalizzato in termini di efficienza, permette tuttavia di raggiungere velocità elevate rispetto ad una gestione ad interrupt che è appesantita dal fatto che bisogna salvare i registri ed eseguire alcune operazioni ausiliarie (come impostare gli indicatori, ecc.).
Per concludere osserviamo che l'interrupt costituisce un meccanismo necessario perchè il sistema (operativo) abbia la sensazione del tempo e possa quindi adottare provvedimenti atti ad assicurare il rispetto dei vincoli di tempo reale.

5.5.2.3 DIRECT MEMORY ACCESS (DMA)

Il metodo ad accesso diretto alla memoria si usa quando i trasferimenti da effettuare avvengono con una cadenza molto elevata, magari anche per tempi brevi ma ad alta frequenza di trasferimento, oppure quando è necessario un brevissimo tempo di ritardo nel trasferimento (nel caso di dati disponibili solo per brevi intervalli di tempo).
Applicazioni tipiche sono le linee seriali ad alta velocità (collegamenti in rete con velocità dell'ordine di 1-10 Mbit/s) e i trasferimenti di informazioni tra memoria operativa e memoria di massa.
Meccanismo.
Il fulcro di tale meccanismo è costituito da un *DMA controller* che ha il compito di sostituirsi alla CPU nella gestione dei trasferimenti di dati assumendo il ruolo di *master* del Bus. Questo *controller* gioca normalmente il ruolo di circuito *slave* (il *master* è la CPU durante l'esecuzione dei programmi) e come tale viene programmato mediante il caricamento nei suoi registri interni di opportuni valori relativi a:

- indirizzo del buffer in memoria
- numero di byte da trasferire
- modalità operative.

La catena **cause-effetti** che si realizza è a grandi linee la seguente.

- Il controllore di DMA recepisce un evento.
- Il controllore di DMA assume il ruolo di master del bus del calcolatore.
- Il controllore di DMA comanda le operazioni di lettura della porta di ingresso e scrittura nell'opportuno indirizzo di memoria (o viceversa).
- Il controllore di DMA incrementa l'indirizzo di memoria e il contatore dei trasferimenti.
- A conteggio raggiunto (Terminal Count) il controllore genera una richiesta di interruzione il cui servizio consiste nel riattivare eventualmente il controllore di DMA per il prossimo trasferimento, e segnalare il completamento di un trasferimento al processo interessato (sincronizzazione di secondo livello).

I tempi in gioco sono molto brevi: dell'ordine di 1 microsecondo per richiesta e conferma ed altrettanto per il trasferimento vero e proprio delle informazioni.
L'efficienza è molto elevata; la CPU non è interessata da tutte queste operazioni ma vede solo una sospensione delle sue attività sul bus per un paio di microsecondi ad ogni trasferimento.

Il costo delle maggiori prestazioni è costituito dal costo relativamente elevato del controllore di DMA e dei circuiti ausiliari necessari, tra cui in genere anche quelli per la gestione di interruzioni necessarie per avvertire il software che il trasferimento è terminato.
Inoltre il DMA una volta programmato presenta una notevole rigidità, nel senso che non consente elaborazioni "on-line" delle informazioni, ma solo semplici trasferimenti a conteggio prefissato.
Infine il DMA può allungare in modo impredicibile i tempi di esecuzione dei processi in corso: talora nei sistemi real-time si evita di usare il DMA anche per evitare questo inconveniente.

Per concludere si riportano in una tabella comparativa le caratteristiche dei meccanismi presentati:

	TEMPO di RISPOSTA	EFFICIENZA	COSTO	COMPLESSITA' hw	COMPLESSITA' sw
contr.prog.	10us	scarsa	basso	bassa	varia
interrupt	50us	discreta	medio	media	media
DMA	2us	ottima	alto	alta	media

5.6 INTERFACCE CIRCUITALI

Si è detto che le interfacce sono costituite dall'insieme di funzionalità HW e SW atte a consentire l'interazione tra elementi, superando le disomogeneità che ne caratterizzano il comportamento.

Spesso è necessario, o conveniente, centrare l'attenzione sugli aspetti circuitali, considerando quelli programmativi come se fossero un'estensione dei primi.
Nel seguito vengono brevemente riprese alcune considerazioni di architettura dei calcolatori, a partire dalle semplici porte di ingresso o uscita e passando poi ad analizzare una tipica interfaccia generalizzata che può essere assunta come modello unificante.

5.6.1 INTERFACCE CIRCUITALI ELEMENTARI

Le interfacce elementari, dette generalmente **porte**, sono costituite da registri o da buffer accessibili dal calcolatore mediante apposite istruzioni di lettura (IN) o scrittura (OUT).

Questi registri sono simili a celle di memoria (e in alcuni calcolatori appartengono allo stesso spazio di indirizzamento, con l'approccio detto "*memory mapped I/O*") ma presentano la caratteristica di essere collegati con elementi del mondo esterno.

Le **porte di ingresso** possono essere realizzate a buffer o a registro.
Per l'acquisizione di informazioni associate a stati, le porte di ingresso sono costituite da "buffer" che, quando sono attivati dalla corrispondente operazione di lettura, immettono sul BUS DATI, in modo trasparente, i valori binari presenti in quel momento ai loro morsetti esterni.

Quando si debbano invece **acquisire informazioni associate ad eventi, occorre utilizzare come porte di ingresso dei registri** (ad esempio di tipo "*latch*") che, come si è detto, siano in grado di memorizzare il valore associato all'evento quando esso accade, mantenendolo "congelato" fino a quando sarà letto dal processore.

Si noti che in questo caso occorrono anche uno o più bit ausiliari che rappresentino la situazione dell'evento (avvenuto, recepito, ecc.) per la gestione del rapporto Produttore (mondo esterno) - Consumatore (calcolatore).

Le **porte di uscita** sono invece viste dal calcolatore come celle di memoria a "sola scrittura" e che rendono disponibile, su opportuni morsetti, il loro contenuto al mondo esterno. **Queste porte devono essere realizzate mediante registri** (*latch*) in cui le informazioni vengono immagazzinate nel breve intervallo corrispondente all'operazione di scrittura e presentate all'esterno con continuità ed immutate fino alla successiva scrittura.

Se si tratta di informazioni di stato questo registro è sufficiente, mentre per informazioni in uscita associate ad eventi occorre corredare il registro di ulteriori indicatori che consentano al calcolatore e ai dispositivi del mondo esterno di riconoscere le varie

situazioni dell'evento, simmetricamente a quanto richiesto in ingresso, per la gestione del rapporto Produttore (calcolatore) - Consumatore (mondo esterno).

5.6.2 INTERFACCIA CIRCUITALE GENERALIZZATA

Da quanto detto risulta che i circuiti di interfaccia potranno in generale comprendere uno o più dei seguenti elementi.

- Buffer o registri per i dati in ingresso, eventualmente organizzati in coda FIFO se è prevista l'acquisizione di eventi (ad es. ricezione di caratteri).
- Registri di dati in uscita.
- Registri di stato i cui bit costituiscono indicatori di situazioni significative, come rilevato evento, overrun, errori vari, ecc.
- Circuiti per la realizzazione HW di funzioni di interfaccia, come conteggi, serializzazione o parallelizzazione di bit, trasferimenti in DMA, ecc.
- Registri di comando in cui i bit, singoli o a gruppi, possono essere scritti in forma codificata per attivare e modificare le funzionalità di interfaccia, modificare gli indicatori di stato, ecc.

Nella fig. 5.10 è rappresentata in forma molto schematica la struttura di un'interfaccia circuitale generale, a cui possono ricondursi molti dispositivi che i costruttori di microprocessori forniscono come elementi di corredo per realizzare la parte di I/O dei calcolatori.

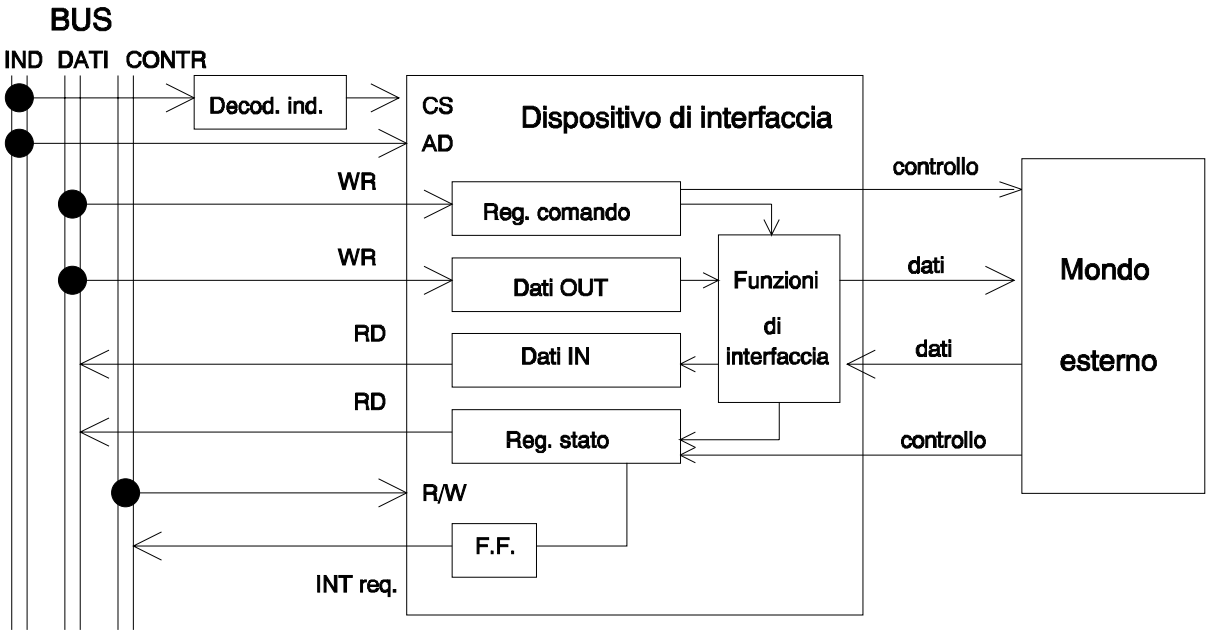

Fig. 5.10 - Schema di interfaccia generalizzata.
CS = *chip select* abilita i trasferimenti sul bus.
AD = *address* seleziona ilregistro interessato dal trasferimento.
R/W = *read / write* determina la direzione del trasferimento.

Notare che in alcune interfacce occorrono diverse operazioni elementari sia di lettura che di scrittura per effettuare correttamente un trasferimento.
Ad esempio un *input* analogico richiede anche comandi in *output*, (come vedremo meglio nel cap. 8) che si possono riassumere come segue.

imposta guadagno
seleziona canale del mux
comando hold
comando start conversione
test fine conversione (eoc)

5.6.3 ESEMPI DI DISPOSITIVI DI INTERFACCIA

Counter / timer

Registri a conteggio, corredati dei circuiti di azzeramento (*clear*), caricamento di valori iniziali (*load*), lettura di un byte alla volta, ecc.

USART

Universal Synchronous Asynchronous Receiver Transmitter. Sono circuiti destinati alla ricezione e trasmissione seriale, corredati delle funzioni di serializzazione e parallelizzazione dei bit (*shift register*), di generazione e riconoscimento dei bit di parità, di riconoscimento di eventuali anomalie, di accodamento dei caratteri in ricezione, dei meccanismi di corretta sincronizzazione con gli eventi "*carattere ricevuto*" e "*carattere trasmesso*", ecc.

PIA

Peripheral Interface Adapter. Termine usato da Motorola per indicare circuiti di gestione di ingresso e uscita di bit, con eventuale riconoscimento di fronti (eventi) di salita o discesa degli ingressi.

5.7 ALTRE LETTURE

Peter D. Lawrence, Konrad Mauch
REAL-TIME MICROCOMPUTER SYSTEM DESIGN: AN INTRODUCTION
McGraw-Hill 1988
Applicativi HW. Medio livello

Cesare Bin
La Trasmissione Dati.
Ed. Buffetti. 1991

R.L. Krutz
Interfacciamento nella Progettazione di Sistemi Digitali.
Ed. Jackson. 1988

L. Mezzalira, N. Scarabottolo
Microcalcolatori: Architettura e Problematiche di Interfacciamento.
Ed. Nuova Italia Scientifica. 1984

G. Jack Lipovsky
OBJECT-ORIENTED INTERFACING TO 16-BIT MICROCONTROLLERS
Prentice Hall 1993
Bibliot. AR-305
Molto interessante
Segnali, architetture, interfacce varie, interrupt, timing.

J.D. Nicoud
Progetto di Interfacce per Microprocessori.
Addison-Wesley Masson 1992
Orientato a HW
Molti esempi interessanti