1 SISTEMI - MODELLI - AUTOMI

Negli argomenti precedentemente trattati, più volte si è ricorsi ad utilizzare la parola *"sistema"*. Si vuole ora dare una definizione più puntuale di *Sistema* che ne esprima appieno il concetto.

Definizione

Si definisce *Sistema* un insieme composto da più elementi che si influenzano reciprocamente, in modo da determinare, nel complesso, un'entità avente un determinato funzionamento con particolari caratteristiche.

Nella definizione si puntualizza il fatto dell'esistenza di elementi di natura diversa i quali interagendo tra loro, concorrono a garantire un determinato funzionamento. Si pensi ad una macchina, composta da elementi di varia natura (meccanica, elettrica, chimica, pneumatica, ecc.), i quali, agendo ed influenzandosi reciprocamente, concorrono allo svolgimento del regolare funzionamento per il quale la macchina è stata costruita.

1.1 Approccio allo studio del funzionamento di un sistema

Come si è detto, il *sistema* è una entità costituita da più elementi: A,B,C... e può essere così complesso da risultare molto difficoltoso lo studio nel suo insieme.

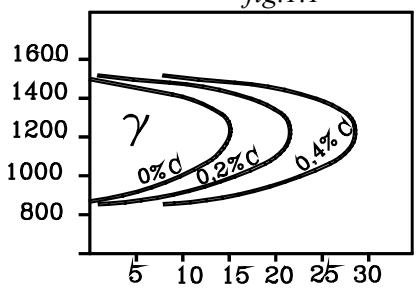
Vi sono due metodi diversi per studiare il sistema: *fisico e sistemico.*

1.1.1 Approccio fisico

Alcune volte il problema può essere semplificato studiando il comportamento reciproco di alcuni elementi fondamentali, lasciando gli altri invariati. Il problema può essere ristretto allo studio dell'influenza reciproca tra due soli elementi . Si introducono poi, ad uno ad uno, gli altri elementi inizialmente lasciati costanti.

Così per esempio si voglia studiare le strutture di un acciaio al cromo (non trattato). Si è in presenza di un sistema chimico a tre elementi: ferro, carbonio, cromo.

fig.1.1



Conviene partire dal diagramma di stato binario "ferro, cromo" (allo stato solido) con 0% di carbonio.

Il cromo abbassa il punto critico A_4 ed innalza A_3, Quindi con una percentuale di circa 15% di carbonio i due punti critici combaciano; in tal modo scompare il campo austenitico e si ha struttura ferritica (cubica a corpo centrato) fino a bassa temperatura. Il comportamento è descritto schematicamente dalla curva parametrata con 0% C.

Studiato il comportamento dei punti critici allo stato solido, tenendo fisso allo 0 % la percentuale di carbonio, si determinano gli stessi punti critici dell'acciaio con un'altra percentuale di carbonio fissa: per esempio 0,2%.

Il carbonio ha, sui punti critici, un'influenza opposta a quella del cromo: abbassa il punto critico A_4 e innalza A_3, per cui, in questo caso, occorre una maggiore % di cromo (circa il 23%) affinché i due punti critici combacino e si abbia la scomparsa dello stato austenitico: curva parametrata con 0,2 % C di fig !.1

Il diagramma di stato si è ottenuto studiando prima le strutture dell'acciaio che si presentano al variare della temperatura, con una percentuale nulla di carbonio; quindi, introducendo man mano una diversa % questo, si sono determinate le strutture che si ottengono al variare della temperatura .

In alcuni sistemi (lineari, vedi oltre) è valida la sovrapposizione degli effetti, per cui si può studiare il comportamento del sistema nella interazione reciproca a due a due degli elementi che lo costituiscono. Così, se gli elementi sono A,B,C, si studia il funzionamento del sistema nella interazione tra: A,B - A,C - B,C.

1.1.2 Approccio sistemico

Nell'approccio sistemico si studia il sistema nel suo insieme e si cerca di spiegare il funzionamento in base ad indagini globali.

Nel caso che il sistema è suddiviso in sottoinsiemi, nello studio globale si cerca l'interconnessione tra di essi.

Nell'approccio sistemico risulta fondamentale individuare le variabili da cui dipende il sistema: quelle che ne descrivono lo stato e quelle che ne determinano il suo funzionamento.

In base alle variabili considerate viene di conseguenza stabilito quale è il sistema che si prende in considerazione. L'introduzione di una nuova variabile amplia il sistema considerato.

Analizziamo in dettaglio le variabili che, in generale, possono interessare lo studio di un sistema.

1.2 Ingressi primari

Sono delle grandezze, funzioni del tempo, introdotte dall'esterno del sistema, che possono considerarsi le cause di un determinato funzionamento del sistema, dipendente da esse.

Gli ingressi primari si possono distinguere in *intenzionali,* che rappresentano segnali utili: di comando o di pilotaggio del sistema e ingressi non intenzionali, detti *di disturbo*, che sono di ostacolo al buon funzionamento del sistema, che occorre, quando possibile, eliminare.

In un istante t la variabile di ingresso si indicherà con *x(t).*

1.2 Uscite

Sono le grandezze, funzioni del tempo, che rilevano gli effetti prodotti dal funzionamento del sistema.

In un istante *t* l'uscita si indicherà con *y(t.*

1.3 Memoria di un sistema - Sistema dinamico - Variabile di stato

La maggior parte dei sistemi reali contengono degli *elementi inerziali*, che ritardano la risposta del sistema ad una richiesta di funzionamento per effetto della variazione di uno o più ingressi.

Da un certo punto di vista, un sistema contenente elementi inerziali *ricorda la condizione (lo stato) in cui esso si trovava, nel momento in cui viene immesso un nuovo ingresso che tende, con il funzionamento, a farlo evolvere verso un nuovo stato.*

L'inerzia, presentandosi come una memoria dello stato del sistema al momento dell'introduzione di una variabile di ingresso, agisce in modo da ritardare il funzionamento e l'evoluzione verso il nuovo stato comandato.

Un problema analogo si è riscontrato e si è affrontato nello studio dei circuiti sequanziali, nei quali vi sono dei componenti che ricordano lo stato assunto negli istanti precedenti a quello di immissione di un comando.

Il circuito sequenziale è un particolare sistema, che ricorda gli stati precedenti e nel quale il passaggio da uno stato all'altro avviene solamente in determinati istanti di tempo.

Qui, quei concetti, già affrontati nello studio del caso particolare dei circuiti sequenziali, si vogliono ampliare in modo da essere validi per un qualsiasi sistema, dove il tempo si intenda che fluisca con continuità e le variabili caratteristiche che descrivono il sistema siano valutabili in ogni istante.

1.3.1 Sistema dinamico

Definizione

Un sistema si definisce *dinamico* quando ha memoria degli stati precedenti. Questo , o per effetto di componenti inerziali o per elementi di memoria appositamente introdotti tra quelli del sistema.

Una conseguenza essenziale che il sistema abbia memoria degli stati precedenti è che: lo stesso valore di una variabile di ingresso, immessa in tempi diversi può determinare effetti diversi.

Non vi è quindi una corrispondenza univoca tra valore della variabile di ingresso e variabile di uscita del sistema, in quanto uno stesso valore della variabile di ingresso può dare uscite diverse a seconda dell'istante in cui viene immesso nel sistema.

1.3.2 Variabili di stato

Da quanto detto occorre introdurre delle variabili che, in ogni istante, descrivano compiutamente la condizione (stato) nel quale si trova il sistema. A tali variabili viene dato il nome di *variabile di stato.*

Una variabile di stato verrà indicata con S(t).

Nei sistemi dinamici, per effetto della memoria degli stati precedenti, quando viene introdotto nell'istante "t" una variazione di una variabile di ingresso, non si ha istantaneamente la variazione della variabile di stato: essa mantiene nell'istante "t" il *valore attuale* S(t) e solamente dopo un certo tempo Δt assumerà il valore dello stato futuro $S(t+\Delta t)$, nel tempo $t+\Delta t$, in conseguenza della variazione all'ingresso.

Per rilevare il funzionamento del sistema, occorre, per prima cosa, determinare la legge che descriva l'evoluzione della variabile di stato tra l'istante "t" e $t+\Delta t$, e poi da un istante iniziale " t_0 " all'istante "t" considerato.

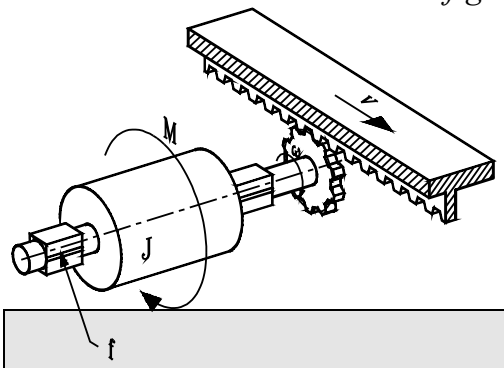
Conosciuta la variabile di stato nell'istante "t" occorre determinare le variabili di uscita del sistema.

Per capire il metodo di studio, ci si riferisce all'esempio di *fig1.2* di sistema meccanico, contenente un elemento inerziale. In questo caso è facilmente determinabile la legge matematica di evoluzione del sistema da un istante "t" ad un istante successivo "$t+\Delta t$" e poi, mediante integrazione, rilevare la legge di evoluzione tra l'istante iniziale "t_0" e l'istante "t".

Più in generale, risulta determinabile l'evoluzione del sistema da un istante "t" ad un istante successivo "$t+\Delta t$", ma non facilmente ottenibile, con integrazione, la legge di evoluzione del sistema tra l'istante iniziale "t_0" e l'istante "t". È proprio su questi casi che occorrerà soffermarsi per indicare metodi di soluzione.

Determiniamo ora la legge di evoluzione del sistema: intendendo con ciò ottenere la relazione che lega la variabile di stato futuro S(t+Δt) nell'istante "t+Δt" a quella attuale S(t) nell'istante "t".

fig.1.2



Il sistema è costituito da un rotore, che, mosso da un momento motore "M(t)", trasmette il moto ad un rocchetto, ingranante con una cremagliera, solidale ad una slitta. Il rotore è supportato da due perni di strisciamento, abbondantemente lubrificati in modo che l'attrito che si manifesta nel contatto sia di tipo viscoso.

Per descrivere il sistema assumiamo come variabile di stato la velocità angolare ω dell'albero motore..

Come variabile di uscita del sistema si assume la velocità "*v*" della slitta.

Sia "f" il coefficiente di attrito viscoso e "J" il momento di inerzia equivalente del sistema " rotore - albero, rocchetto slitta " riportato sull'asse dell'albero.

Il momento M applicato sull'albero provoca nell'incremento di tempo Δt una accelerazione angolare:

$$\varepsilon = \frac{\Delta \omega}{\Delta t} \qquad (1.1)$$

Per l'equilibrio dinamico il momento motore "M" viene equilibrato da un momento dovuto all'attrito viscoso e da un momento dovuto all'inerzia

Attrito viscoso È proporzionale alla velocità relativa delle superfici a contatto strisciante ed ha senso opposto a quello di strisciamento. L'attrito determina un momento che si può esprimere rispetto alla velocità angolare:

$$f \cdot \omega$$

Inerzia Il momento d'inerzia equivalente "J" determina un momento, che risulta proporzionale alla accelerazione angolare ed ha senso opposto a questa.

$$J \cdot \varepsilon$$

Per l'equilibrio si ha:

$$M(t) - J \cdot \varepsilon(t) - f \cdot \omega(t) = 0 \qquad (1.2)$$

Da cui si può ricavare ε

$$\varepsilon = \frac{1}{J} \cdot M(t) + \frac{f}{J} \cdot \omega(t) \qquad (1.3)$$

Dalla espressione (1.3) si può ricavare immediatamente la relazione che lega la variabile di stato futuro $\omega(t + \Delta t)$ nel tempo $(t+\Delta t)$ a quella attuale (ωt) nel tempo *"t"*.

Sostituendo la (1.1) nella (1.3):

$$\frac{\Delta \omega}{\Delta t} = \frac{1}{J} \cdot M(t) + \frac{f}{J} \cdot \omega(t) \qquad \text{dove} \quad \Delta \omega = \omega(t + \Delta t) - \omega(t) \quad \text{quindi}$$

$$\frac{\omega(t + \Delta t) - \omega(t)}{\Delta t} = \frac{1}{J} \cdot M(t) + \frac{f}{J} \cdot \omega(t) \qquad \text{da cui:}$$

$$\boxed{\omega(t + \Delta t) = \omega(t) + \frac{1}{J} \cdot M(t) \cdot \Delta t + \frac{f}{J} \cdot \omega(t) \cdot \Delta t} \qquad (1.4)$$

L'espressione (1.4) formula la legge di evoluzione del sistema dall'istante *"t"* a *"t+Δt"* e determina il passaggio della variabile ω dallo stato attuale a quello futuro.

L'espressione (1.4) quindi è un *modello matematico* capace di descrivere il funzionamento del sistema tra l'istante *"t"* e *"t+Δt"*.

Dal modello si deve poi risalire alla legge che fornisca in ogni istante "*t"* la variabile di stato, come si dice occorre integrare la relazione fondamentale data dal modello.

Nel caso in esame la relazione fondamentale del modello matematico si traduce in una equazione differenziale facilmente integrabile (vedi oltre).

Ponendo che nell'istante iniziale $t=0$ risulti $\omega=0$, si ottiene:

$$\omega = \frac{M}{f}\cdot\left(1 - e^{-\frac{f}{J}\cdot t}\right) \quad (1.5)$$

Analizzando la (1.5) si osserva che:

- Nell'istante iniziale $t=0$ risulta $\omega=0$
- Per $t \to \infty \quad \omega \to \frac{M}{f}$

L'effetto dell'inerzia del sistema è quello di ritardare la velocità di regime $\frac{M}{f}$ imposto dalla variabile di ingresso *"momento M"* equilibrato dal momento dovuto all'attrito.

Nei successivi istanti di tempo la variabile di ingresso ha effetti diversi sulla variabile di stato ω. Questo è dovuto all'effetto memoria degli elementi inerziali

Una volta determinata la variabile di stato in ogni istante, si deve ricavare la variabile di uscita in funziona del tempo.

Nel caso in esame la variabile di uscita è la velocità della slitta, facilmente ricavabile dalla velocità angolare, dato il raggio della circonferenza primitiva del rocchetto.

$$v(t) = \omega(t)\cdot r \quad (1.6)$$

1.4 Modello

Nell'analisi del funzionamento del semplice sistema di *fig.1.2* sono delineati le tre fasi fondamentali per lo studio di un sistema.

- Determinazione della relazione fondamentale che dà l'evoluzione della variabile di stato dall'istante *"t"* a *"t+Δt"*: relazione cioè tra $S(t)$ e $S(t+\Delta t)$
- Integrazione della relazione fondamentale, in modo da determinare la legge che dà la variabile di stato *S* nei successivi istanti di tempo, partendo da un istante iniziale t_0 dove essa ha un determinato valore $S(t_0)$.
- Determinazione della variabile di uscita in funzione di quella di stato.

La prima fase è il cardine su cui poggia tutto il resto. Occorre determinare *la relazione fondamentale* di una certa natura: matematica, fisica, grafica, o di altro tipo, capace di riprodurre l'evolversi del sistema dall'istante *"t"* al successivo *"t+Δt"* , alla quale diamo nome di *modello del sistema.*

Definizione

Si definisce modello di un sistema una rappresentazione di natura: fisica, matematica, grafica o di altro tipo, capace di descrivere ed emulare il funzionamento, in grado cioè di riprodurre

l'evoluzione del sistema tra due istanti successivi *"t" e "t+Δt"*, determinando in questi, rispettivamente, la variabile di stato attuale $S(t)$ e quella futura $S(t + \Delta t)$

Il modello può essere reale, di natura fisica che ricopia il meccanismo, magari in scala ridotta oppure di natura astratta: matematica, grafica, probabilistica.

Occorre porre molta cura e attenzione nella scelta del modello del sistema, perché esso determina la relazione fondamentale dell'evoluzione di esso e riproduce il funzionamento nell'incremento di tempo elementare Δt. Una scelta non idonea del modello, conduce poi a dei risultati erronei, che, nel caso di costruzioni ingegneristiche, possono portare a dei disastri.

Occorre osservare che per comodità, nella trattazione, ci riferiamo ad un solo ingresso, una sola variabile di stato, e una sola uscita, ma, in generale, tali variabili possono essere più di una. Le regole, le osservazioni fatte per una sola variabile si possono estendere alle altre.

Così da ora in poi quando ci si riferisce agli ingressi del sistema, per brevità si indicheranno con la sola lettera *"x"*, così si indicheranno con la sola lettera *"S"* le variabili di stato e con *"y"* quelle di uscita.

1.5 Simulazione

Ottenuto il modello del sistema, e quindi la relazione fondamentale dell'evoluzione di esso tra l'istante *"t"* e *"t+Δt"*, occorre, poi, integrare tale relazione per ottenere la legge che dia il valore della variabile di stato S in ogni istante, a partire da un istante iniziale t_0 ove assume un determinato valore $S(t_0)$.

Dalla variabile di stato S si risale facilmente a determinare la variabile di uscita *"y"*.

Vi sono molti sistemi fisici, economici... nei quali si riesce a determinare la legge fondamentale di evoluzione tra gli istanti *"t"* e *"t+Δt"*, ma questa risulta difficilmente integrabile, e non si riesce a ricavare matematicamente la legge che dia, in ogni istante, il valore delle variabili di stato in funzione di quelle di ingresso e, di conseguenza, il valore delle uscite.

Quando però la legge fondamentale, che dà l'evoluzione del sistema come trasformazione della variabile di stato nel tempo, è discretizzata tra listante t e il successivo $t+\Delta t$,allora è facilmente integrabile con un programma eseguito al calcolatore, il quale prende il nome di *simulazione*.

Definizione

La simulazione è un *programma* che, sfruttando la relazione fondamentale discretizzata espressa dal modello, determina, nei successivi istanti di tempo, l'evolversi del sistema, producendo la corrispondente variabile di stato, nelle diverse condizioni possibili di funzionamento.

La simulazione riveste un ruolo importante nell'analisi dell'evolversi di sistemi complessi e in alcuni casi è il solo mezzo utilizzabile.

Importante è la determinazione del modello il quale deve rispondere fedelmente all'evoluzione discretizzata del sistema reale.

Va poi posto in rilievo che il modello dipende dal sistema considerato e questo dipende a sua volta dalle variabili e dai parametri presi in considerazione. L'esclusione di qualche parametro, non considerato nel sistema preso a campione, può condurre ad un comportamento di questo diverso da quello reale. Si può ottenere così un modello la cui simulazione sortisce un funziona-

mento, in casi limiti, diverso da quello reale. Si formulano così delle previsioni erronee con conseguenze che possono essere anche gravi.

1.6 Processo

La parola *processo* in generale, sta ad indicare il modo di procedere (procedura) per raggiungere un determinato scopo.

Nel caso dello studio dei sistemi si può dire che:

Per processo si intende la procedura corrispondente alla individuazione del modello del sistema e di un metodo di natura matematica, fisica... o di un programma di simulazione, mediante il quale si possono determinare, nell'evoluzione del sistema, i valore delle variabili di uscita in corrispondenza a quelli delle variabili di ingresso.

1.7 Variabili e funzioni caratteristiche di un sistema dinamico

Si è già definito un sistema dinamico come quello che ha memoria degli stati precedenti o per effetto di elementi inerziali che lo compongono o per elementi di memoria appositamente introdotti.

Analizziamo ora più puntualmente tutte le variabili che intervengono nella descrizione del funzionamento del sistema. Si introdurranno poi le funzioni che ne regolano l'evoluzione, determinando, infine, le uscite in corrispondenza degli ingressi.

Le variabili di ingresso uscita e di stato sono state già introdotte precedentemente; vengono qui elencate per completezza.

1.7.1 Insieme dei tempi - Sistemi continui - Sistemi discreti

L'evoluzione del sistema presuppone *"un prima"* e *"un dopo"* e quindi il fluire del tempo in un insieme T.

Con il modello si vuole studiare il passaggio di stato dall'istante $"t"$ al successivo $"t+\Delta t"$.

L'incremento Δt può essere infinitamente piccolo, allora l'insieme T appartiene ai numeri reali e il sistema si dice *continuo*.

Se l'incremento Δt è finito l'insieme dei tempi T può essere espresso da numeri interi, che ne esprimono i successivi istanti. In questo caso il sistema si dice *discreto*. Così nello studio di circuiti pneumatici con finecorsa, interessano gli istanti t_i e t_{i+1} corrispondenti a due fasi successive di un ciclo.

1.7.2 Insieme delle variabili di ingresso

Sono delle grandezze, funzioni del tempo, introdotte dall'esterno del sistema, che possono considerarsi le cause di un determinato funzionamento del sistema, dipendente da esse.

Possono essere *intenzionali* o *di disturbo*.

Le variabili di ingresso vengono indicate con la lettera $"x"$.

1.7.3 Insieme delle funzioni di ingresso

Le variabili di ingresso sono funzioni del tempo. Per studiare il funzionamento del sistema vengono introdotte funzioni di ingresso appartenenti ad un certo insieme ammissibile dal sistema. Per lo studio di molti sistemi vengono introdotte in ingresso delle funzioni tipo, dalla cui risposta si può giudicare il buon funzionamento o meno del meccanismo in analisi.

1.7.4 Insieme delle variabili di stato

Sono le variabili che individuano e definiscono in ogni istante lo stato del sistema.

Le variabili di stato vengono indicate con la lettera *"S"*.

1.7.4 Insieme delle uscite

Le variabili di uscita sono le grandezze, funzioni del tempo, che rilevano gli effetti prodotti dal funzionamento del sistema.

In un istante t l'uscita si indicherà con *y(t)*.

1.7.5 Funzioni caratteristiche per lo studio di un sistema dinamico

Dall'esempio di analisi del funzionamento del semplice sistema dinamico di *fig.1.2* si sono rilevate tre fasi di studio del sistema.

Le prime due fasi di modellazione e di simulazione (o integrazione delle equazioni differenziali) portano alla determinazione delle variabili di stato $S(t)$ nell'istante *"t"*, partendo da un istante iniziale t_0 ove assumono, ciascuna un determinato valore $S(t_0)$.

Conosciute la variabile di stato nell'istante *"t"* si determinano poi le variabili di uscita $y(t)$.

Occorre quindi definire due funzioni caratteristiche: una che determina la variabile di stato $S(t)$ in relazione alle variabili di ingresso e l'altra la variabile di uscita in relazione alla variabile di stato ottenuta.

1.7.5.1 Funzione trasformazione di stato

Come si è detto nel paragrafo riguardante la simulazione, in molti sistemi è possibile determinare la relazione fondamentale della evoluzione dello stato in forma discretizzata *(modello)*, ma non è possibile, con integrazione diretta, determinare in forma matematica compiuta la funzione $S(t)$ nell'istante *"t"* considerato.

Comunque, anche se non esprimibile con formula matematica, attraverso la simulazione si può determinare la variabile di stato $S(t)$ nella sua evoluzione dall'istante iniziale t_0 fino all'istante *"t",* in relazione alle variabili di ingresso $x(t)$.

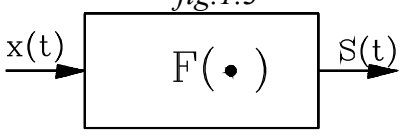
Perciò cui si può introdurre un simbolo che rappresenti, in forma generica, la relazione (di qualsiasi natura) che determina la variabile di stato $S(t)$, nella sua evoluzione dall'istante iniziale t_0 all'istante *"t"* in funzione della variabile di ingresso $x(t)$, applicata dall'istante t_0 all'istante *"t"* . Detta relazione verrà indicata con $F(\bullet)$, dove il simbolo "$\bullet$" sta ad indicare la sua indeterminazione espressiva. Alla relazione $F(\bullet)$ si da nome di *trasformazione di stato*.

Definizione

> Si definisce funzione di *trasformazione di stato* la relazione di qualsiasi natura capace di ottenere il valore della variabile di stato $S(t)$ nella sua evoluzione dall'istante iniziale t_0 fino all'istante *"t" considerato,* in funzione della variabile di ingresso$x(t)$, applicata dall'istante iniziale t_0 all'istante *"t"*.

La funzione di trasformazione di stato si può indicare con un blocco rappresentativo.

fig.1.3



Si può dare della *trasformazione di stato* una espressione formale nella quale, per la determinazione della variabile di stato $S(t)$ vengono poste in rilievo le variabili che ne regolano l'evoluzione dall'istante iniziale t_0 all'istante *"t"* considerato.

Ci possiamo domandare: da quali parametri dipende la variabile di stato nell'istante t ?
per rispondere a questa domanda, occorre fare alcune considerazioni.

Come si è detto, in un sistema dinamico, per effetto della memoria degli stati precedenti, una stessa variabile di ingresso, immessa in tempi diversi, può sortire differenti effetti.

Da questa osservazione ne viene che, attualmente, la *variabile di stato* $S(t)$ del sistema non dipende soltanto dal valore che la variabile di ingresso $x(t)$ assume in quell'istante. Detta *variabile di stato* $S(t)$ potrà assumere valori diversi a seconda di quelli che la stessa variabile di ingresso ha assunto negli stati precedenti a quello considerato; fino a risalire all'istante iniziale $"t_0"$ in cui la variabile di stato è partita con un certo valore iniziale noto $S(t_0)$.

Si può dire in modo espressivo che la variabile di stato $S(t)$ dipende dalla *storia* della variabile di ingresso. ovvero dai valori assunti dalla variabile di ingresso a partire dall'istante iniziale t_0 fino all'istante considerato $"t"$.

Per esprimere che della variabile di ingresso x si vuole la storia di essa e non solamente il valore nell'istante $"t"$ si impiega il simbolo $x(\bullet)$.

Il simbolo $x(\bullet)$ sta ad indicare la storia della variabile di ingresso x, ovvero l'insieme dei valori che essa ha assunto dall'istante iniziale t_0 fino all'istante considerato $"t"$.

Il simbolo $x(\bullet)$ comparirà come elemento da cui dipende la *trasformazione di stato* $F(\bullet)$

Le precedenti osservazioni conducono ad una logica conseguenza:

La variabile di stato $S(t)$ nell'istante $"t"$ dipende dall'istante iniziale t_0 da cui si inizia a considerare l'evoluzione del sistema e dal valore iniziale $S(t_0)$ che in quell'istante ha assunto la variabile di stato.

Ciò è evidente. Considerando che un sistema dinamico ha memoria degli stati precedenti, esso dipenderà ovviamente dall'istante iniziale da cui si considera l'inizio dell'evoluzione e dal valore iniziale assunto dalla variabile di stato.

L'istante iniziale $"t_0"$ e il valore iniziale $S(t_0)$ della variabile di stato compariranno come elementi da cui dipende la *trasformazione di stato* $F(\bullet)$

Vi è poi un ultima variabile da cui esplicitamente può dipendere la variabile di stato ed entrare come variabile esplicita nella *trasformazione di stato* F : questa variabile è il tempo.

Si osservi che il tempo rientra già implicitamente nella storia della variabile di ingresso $x(\bullet)$. Esprimere che la variabile di stato $S(t)$ e quindi *la trasformazione di stato F* dipenda esplicitamente dal tempo, vuol significare che lo stato del sistema varia nel tempo pur restando le stesse le altre variabili (stessa storia della x, stesso istante iniziale t_0 e valore iniziale $S(t_0)$ della variabile di stato). Ciò può avvenire se i parametri (fisici, chimici...) degli elementi che compongono il sistema risultano variabili nel tempo. Nell'esempio del sistema meccanico di *fig.1.2* si avrebbe la variazione nel tempo del coefficiente di attrito $"f"$ o del momento di inerzia $"J"$.

In definitiva la trasformazione di stato dipende: dal valore iniziale $S(t_0)$ della variabile di stato, dall'istante iniziale t_0 da cui si considera l'inizio dell'evoluzione del sistema, dalla storia $x(\bullet)$ della variabile di ingresso e dal tempo *"t"*. Ciò si rappresenta formalmente con l'espressione:

$$S(t) = F\left(t_0,\ S(t_0),\ x(\bullet),\ t\right) \quad (1.7)$$

Si osservi che la variabile di stato compare nella (1.7) sia al primo che al secondo membro, ma riferita ad istanti diversi: nel secondo membro viene indicato il valore che la variabile di stato assume nell'istante iniziale $S(t_0)$, da cui dipendono tutti gli stati del sistema nei successivi istanti di tempo; nel primo membro, invece, è indicato il valore che la variabile di stato assume nell'istante *"t"* considerato.

Il fatto di trovare la variabile di stato nel primo e al secondo membro di una relazione con significati diversi è già noto al lettore, che l'ha incontrato in occasione nello studio delle memorie, nel 2° volume.

La espressione (1.7) è una estensione di quei concetti di evoluzione nei quali occorre considerare un *prima* e un *dopo*; cosicché il valore che la variabile di stato ha assunto prima $S(t_0)$ influenzerà quello che essa assumerà dopo $S(t)$.

1.7.5.2 Funzione velocità di transizione di stato

L'effetto della memoria nel sistema dinamico fa sì che nei successivi istanti di tempo l'evoluzione non avviene sempre alla stessa maniera: la variabile di stato da un istante *"t"* al successivo *"t+Δt"* non subisce lo stesso incremento.

Ciò vuol dire che la velocità $S'(t)$ con cui varia la variabile di stato *(la sua derivata)* dipende dal valore che essa ha attualmente nell'istante considerato.

Inoltre, essendo il calcolo della velocità (la derivata di S) ristretta ad un tempo infinitesimo *"dt"*, il suo valore dipende da quello $x(t)$ che la variabile di ingresso ha nell'istante *"t"* . Infatti la storia dell'ingresso *x*, ristretta nel tempo infinitesimo *"dt"* coincide con il valore assunto nell'istante *"t"*.

Nel caso che i parametri caratteristici degli elementi che compongono il sistema varino nel tempo , allora la velocità di transizione o trasformazione di stato $S'(t)$ dipende esplicitamente dal tempo *"t"*.

la relazione che lega la velocità di trasformazione a quella di ingresso si può rappresentare formalmente con l'espressione:

$$S'(t) = V(S(t), x(t), t) \quad (1.8)$$

Nella quale la variabile di ingresso $x(t)$ è quella che si ha nell'istante considerato *"t"*

Nella espressione è posto in rilievo che la velocità di trasformazione dipende dal valore che le variabili di stato e di ingresso hanno nell'istante nel quale si vuole rilevare detta velocità.

1.7.5.3 Funzione di trasformazione d'uscita

L'ultima fase dello studio di un sistema riguarda la determinazione delle variabili di uscita $y(t)$ che ne rilevano il funzionamento.

Così ne caso semplice di sistema meccanico di *fig.1.2* il meccanismo ha come obiettivo il moto della slitta; la variabile assunta come uscita del sistema è la velocità della slitta.

Le variabili di uscita dipendono ovviamente dai valori che le variabili di stato assumono nell'istante *"t",* in quanto queste sono proprio quelle che definiscono in quell'istante il sistema e determinano univocamente le uscite.

Ora le variabili di stato, come si è rilevato, dipendono dalla storia degli ingressi $x(\bullet)$ e quindi questi, implicitamente influenzano le uscite.

In generale, può avvenire: non solo che le uscite dipendano implicitamente dagli ingressi attraverso le variabili di stato, ma che dette variabili di ingresso compaiano anche come variabili esplicite della funzione della trasformazione d'uscita.

Inoltre, nel caso che i parametri caratteristici degli elementi che compongono il sistema variano nel tempo, allora la funzione della trasformazione d'uscita di pende esplicitamente dal tempo *"t"*.

In conclusione le variabili di uscita, in generale, dipendono: dai valori delle variabili di stato nell'istante *"t"* considerato, dai valori delle variabili di ingresso nell'istante *"t"* considerato e in modo esplicito dal tempo *"t"*.

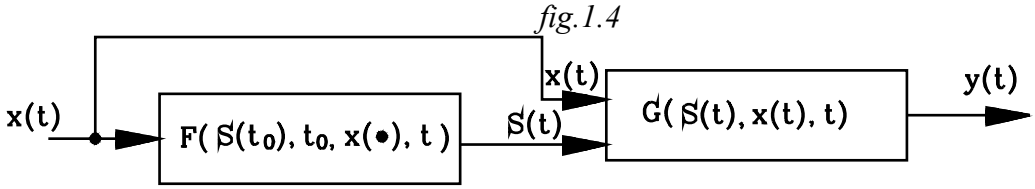
La funzione di trasformazione d'uscita si può rappresentare formalmente con l'espressione:

$$y(t) = G(S(t), x(t), t) \quad (1.9)$$

Osservando l'espressione (1.9) si può rimarcare che le variabili di uscita di pendono dallo stato attuale $S(t)$; inoltre può dipendere esplicitamente: dalla variabile di ingresso nell'istante *"t"* e dal tempo.

In un sistema a blocchi si può rappresentare il processo che dalle variabili di ingresso $x(t)$ conduce a quelle di uscita.

fig.1.4



In figura, la variabile $x(t)$ viene rappresentata all'ingresso della funzione di trasformazione di stato $F(t_0, S(t_0), x(\bullet), t)$ dalla quale si ha in uscita la variabile di stato $S(t)$ nell'istante *"t"*. Nell'ingresso della funzione di trasformazione d'uscita $G(S(t), x(t), t)$ vengono introdotte la variabile di stato $S(t)$ e quella di ingresso $x(t)$.

1.8 Particolari sistemi dinamici

Il processo di determinazione delle uscite rispetto agli ingressi rappresentato nello schema a blocchi di *fig.1.4,* si riferisce ad un sistema dinamico nella più ampia generalità. Si possono presentare dei sistemi dinamici nei quali la funzione di trasformazione di stato o quella di d'uscita risulta modificata rispetto alla espressione generale, per una o più particolarità delle variabili da cui dipende il sistema.

1.8.1 Sistemi puramente dinamici

Un sistema si dice puramente dinamico quando la trasformazione d'uscita non dipende in modo esplicito dalla variabile di ingresso.

In questo caso la variabile di uscita dipende solamente da quella di stato, la quale a sua volta, dipende dalla storia della variabile di ingresso.

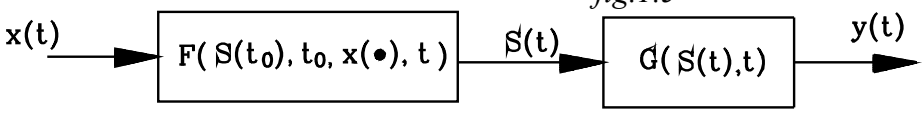
Quindi la variabile di ingresso influenza l'uscita attraverso la variabile di stato, ma non direttamente. In molti casi, quando ciò avviene, risulta conveniente far coincidere la *variabile di uscita con quella di stato*.

Ciò si è applicato nello studio delle memorie con precedenza alla cancellazione o alla scrittura (Volume 2°).

Nella espressione della funzione di trasformazione della variabile d'uscita non compare esplicitamente la variabile di ingresso.

$$y(t) = G(S(t),t) \quad (1.10)$$

fig.1.5



Il processo di determinazione delle variabili di uscita rispetto a quelle di ingresso, per un sistema puramente dinamico è rappresentato in *fig.1.5*. In questa all'ingresso della funzione di trasformazione d'uscita si ha solamente la variabile di stato $S(t)$.

1.8.2 Sistemi stazionari

Nelle funzioni di trasformazione di stato o d'uscita di un sistema dinamico compare esplicitamente la variabile tempo, quando uno più parametri caratteristici degli elementi che compongono il sistema variano nel tempo.

È molto frequente invece che il sistema in studio, nel tempo di funzionamento considerato, sia costituito da elementi i cui parametri caratteristici rimangono costanti.

Così il sistema meccanico di *fig.1.2* è composto da elementi (rotore cuscinetti ecc.) i cui parametri (coefficiente d'attrito, momento d'inerzia) nel breve tempo di funzionamento non variano.

Definizione

Un sistema dinamico si dice stazionario quando il tempo *"t"* non compare come variabile esplicita nelle funzioni di trasformazione di stato e di trasformazione d'uscita.

Nei sistemi stazionari i parametri caratteristici degli elementi che lo costituiscono sono costanti nel tempo..

Per quanto detto, nei sistemi dinamici stazionari non compare nelle funzioni di trasformazione di stato e d'uscita.

Dette funzioni formalmente hanno le seguenti espressioni:

$$\begin{cases} S(t) = F\left(t_0,\ S\left(t_0\right),\ x(\bullet)\right) \\ y(t) = G(S(t),\ x(t)) \end{cases} \quad (1.11)$$

1.8.3 Sistemi puramente dinamici e stazionari

In questo caso valgono entrambi le proprietà dei punti "1.8.1 -1.8.2"

Essendo il sistema stazionario nelle funzioni di trasformazione di stato e d'uscita non compare il tempo come variabile esplicita; essendo anche puramente dinamico, la funzione di trasformazione d'uscita non dipende esplicitamente dalla variabile di ingresso.

Formalmente le funzioni di trasformazione di stato e d'uscita hanno, rispettivamente le seguenti espressioni:

$$\begin{cases} S(t) = F(t_0, S(t_0), x(\bullet)) \\ y(t) = G(S(t)) \end{cases} \quad (1.12)$$

Come si nota, l'espressione della trasformazione d'uscita dipende solamente dalla variabile di stato, tanto che questa, in molti casi, si può far coincidere con la variabile d'uscita.

Per esempio il sistema meccanico di *fig.1.2* si presenta come un sistema puramente dinamico stazionario.

Infatti risulta stazionario, in quanto il coefficiente d'attrito *"f"* e il momento d'inerzia *"J"* si possono sicuramente considerare costanti nel breve tempo nel quale si osserva il funzionamento del sistema; inoltre esso risulta puramente dinamico, in quanto la variabile di uscita *"v"* dipende solamente da quella di stato ω: $v = \omega \cdot r$.

1.9 Sistemi algebrici

Vi sono dei sistemi nei quali non esistono o possono essere trascurati gli elementi inerziali: in questo caso il sistema non ha memoria degli stati precedenti.

Un sistema si dice di *tipo algebrico* quando non ha memoria degli stati precedenti. In questi sistemi non vi sono elementi inerziali tra quelli che lo compongono.

In un sistema algebrico si può anche definire una funzione di stato ma questa non dipende più da uno stato precedente, essa risulta funzione della variabile di ingresso nell'istante *"t"* , e dal tempo, se i parametri caratteristici degli elementi che lo compongono non sono costanti.

Le funzioni di trasformazione di stato assume così la seguente espressione:

$$S(t) = F(x(t), t) \quad (1.13)$$

Si osservi che la trasformazione di stato dipende esplicitamente dalla variabile di ingresso $x(t)$ nell'istante *"t"* e non dalla sua storia a partire da un istante iniziale t_0.

La trasformazione d'uscita dipende poi dalla variabile di stato nell'istante *"t"* e può dipendere dalla variabile di ingresso e dal tempo.

$$y(t) = G(S(t), x(t), t) \quad (1.14)$$

Si può formalmente sostituire la (1.13) nella (1.4) ottenendo

$$y = G(F(x(t), t), x(t), t) \quad (1.15)$$

Quindi in ultima analisi la funzione di trasformazione di uscita può essere espressa esplicitamente rispetto alla variabile di ingresso nell'istante *"t"* e al tempo.

$$y = G(x(t), t) \quad (1.16)$$

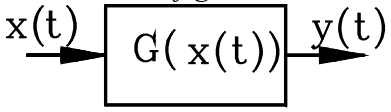
Se poi il sistema è stazionario, al funzione di trasformazione d'uscita dipende solo dalla variabile di ingresso.

$$y = G(x(t)) \quad (1.17)$$

Si ha così, tra la variabile di ingresso e quella d'uscita, un *legame* diretto al quale si dà il nome di *funzione di trasferimento o funzione di trasferta, oppure funzione di trasmissione.*

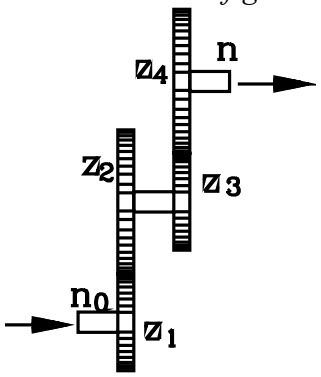
Nello schema a blocchi si pone all'ingresso la variabile $x(t)$ e in uscita $y(t)$

fig.1.6



Così, per esempio, trascurando l'inerzia delle ruote dentate nella trasmissione di *fig.1.7,* il numero di giri di uscita *n* dalla trasmissione è esprimibile direttamente rispetto al numero di giri *"n_0"* posto all'ingresso.

fig.1.7



Dal rapporto di trasmissione τ si ha :

$$\tau = \frac{n}{n_0} = \frac{z_1 \cdot z_3}{z_2 \cdot z_4} \quad \text{da cui} \quad n = \frac{z_1 \cdot z_3}{z_2 \cdot z_4} \cdot n_0$$

1.10 Sistemi lineari

Vi sono dei fenomeni nei quali l'effetto è proporzionale alla causa che l'ha determinato. La legge che esprime la dipendenza tra causa ed effetto è rappresentabile con un tratto di retta: per questo prende il nome di *legge lineare* o *legge di linearità.*

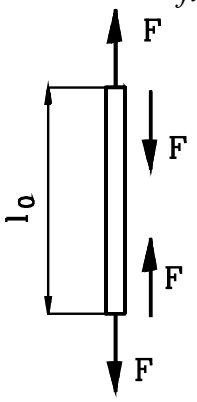
Un sistema si dice lineare quando risponde alla legge di linearità, secondo la quale la variabile di uscita risulta proporzionale a quella d'ingresso.

Così un sistema, costituente una struttura metallica, sollecitata entro un certo limite, risponde ad un comportamento lineare, rispettando la legge di Hooke:

Le sollecitazioni unitarie σ sono proporzionali alle deformazioni unitarie ε . Il loro rapporto è una costante *E,* denominata modulo di elasticità longitudinale.

$$\frac{\sigma}{\varepsilon} = E$$

fig.1.8



Si consideri così un tirante di lunghezza *lo* e sezione S_0 sollecitato a trazione da forze *F* dirette verso l'esterno.

Imprimendo una deformazione di allungamento Δl, il tirante reagisce con una tensione interna che determinerà una reazione pari e contraria alla forza *F* che si è applicata dall'esterno.

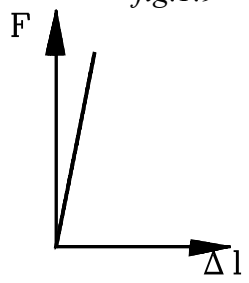
Per la legge di Hooke risulterà:

$\sigma = \varepsilon \cdot E$ dove: $\sigma = \frac{F}{S_0}$ e $\varepsilon = \frac{\Delta l}{l_0}$ nella quale Δl è l'allungamento subito dal tirante. Sostituendo si ha:

$\frac{F}{S_0} = E \cdot \frac{\Delta l}{l_0}$ da cui $F = \frac{E \cdot S_0}{l_0} \cdot \Delta l$ ponendo $\frac{E \cdot S_0}{l_0} = k$ si ottiene:

$$F = k \cdot \Delta l \quad (1.18)$$

fig.1.9



La forza di reazione F risulta proporzionale all'allungamento Δl . Il grafico della relazione (1.18) che lega F a Δl è una retta passante per l'origine.

La legge di proporzionalità è valida fino ad una certa sollecitazione: come noto fino al "*carico di scostamento dalla proporzionalità*".

Principio di sovrapposizione degli effetti

La linearità comporta una proprietà fondamentale che va sotto il nome di principio di sovrapposizione degli effetti, che si può così formulare:

In un sistema (o fenomeno) vale il principio di sovrapposizione degli effetti quando alla somma della cause corrisponde la somma degli effetti.

Le cause di un determinato funzionamento del sistema corrispondono alle variabili di ingresso; gli effetti sono le variabili di uscita.

Allora, se $x_1(t)$ è un ingresso che determina l'uscita $y_1(t)$ e $x_2(t)$ è un altro ingresso che determina l'uscita $y_2(t)$, per il principio di sovrapposizione degli effetti, ne deriva che all'ingresso $"x_1(t) + x_2(t)"$ vi corrisponde l'uscita $"y_1(t) + y_2(t)"$.

Si consideri così di nuovo il tirante di *fig.1.9*, sollecitato entro il limite ove si può ritenere valida la legge di Hooke.

Si imprima una deformazione Δl_1, lo sforzo sarà

$$F_1 = k \cdot \Delta l_1 \quad (1.19)$$

Imprimendo una seconda deformazione Δl_2 lo sforzo sarà:

$$F_2 = k \cdot \Delta l_2 \quad (1.20)$$

Se ora si ora si imprime una deformazione Δl somma delle prime due:

$$\Delta l = \Delta l_1 + \Delta l_2 \quad (1.21)$$ si otterrà lo sforzo:

$$F = k \cdot \Delta l \quad (1.22)$$

Sostituendo nella (1.22) la (1.21) si ottiene:

$F = k \cdot (\Delta l_1 + \Delta l_2)$ $\quad F = k \cdot \Delta l_1 + k \cdot \Delta l_2$ ma per le (1.19) e (1.20) si ottiene:

$$F = F_1 + F_2 \quad (1.23)$$

Le relazioni che si sono ricavate nell'esempio sono di tipo algebrico. potrebbe quindi sembrare che la sovrapposizione degli effetti sia valida solamente per i sistemi algebrici. Ciò è del tutto limitativo.

Più in generale per giudicare se un sistema è lineare o meno si può analizzare se per esso è valido il principio di sovrapposizione degli effetti.

Un sistema nel quale è valida la sovrapposizione degli effetti è lineare.

Si può quindi dire, più in generale, che un sistema è lineare, quando l'evoluzione che esso ha nella applicazione di più ingressi, risulta uguale alla somma delle evoluzioni prodotte da ciascuno di essi.

1.11 Altri elementi di classificazione dei sistemi

I tipi di sistemi precedentemente studiati possono appartenere a delle classi aventi particolari proprietà.

1.11.1 Sistemi deterministici

Un sistema si definisce deterministico quando è possibile stabilire con certezza il suo evolversi verso lo stato futuro, note le condizioni iniziali, nell'istante iniziale, il tempo trascorso a partire da quello e le variabili introdotte nell'ingresso.

In questi sistemi si possono univocamente determinare le funzioni di trasformazione di stato e d'uscita.

Il sistema meccanico di *fig.1.2* è deterministico, così lo sono tutti i sistemi combinatori e sequenziali che si sono studiati nella pneumatica.

1.11.2 Sistemi stocastici

Sono dei sistemi nei quali l'evoluzione è casuale e determinato dalla legge della probabilità.

1.11.3 Sistemi a stati infiniti

In questi sistemi le variabili di stato possono assumere infiniti valori entro il campo di esistenza. La suddivisione del tempo e di osservazione del funzionamento si può spingere all'infinitesimo. La variazione tra uno stato e l'altro avviene con continuità, potendo così assumere, la variabile di stato, tutti gli stati intermedi.

I segnali analogici provengono da sistemi a stati infiniti. Così il sistema meccanico di *fig.1.2* è un sistema a stati infiniti.

1.11.4 Sistemi a stati finiti

Un sistema si dice a stati finiti quando la variabile di stato può assumere solamente un numero finito di valori.

Un circuito a relè o pneumatico a valvole direzionali con finecorsa ecc. sono sistemi a stati finiti. Infatti un relè può assumere solamente due stati, così pure una valvola pneumatica direzionale può assumere due o tre posizioni.

1.12SISTEMI A STATI FINITI - AUTOMI

Nel precedente paragrafo si è data la definizione di sistemi, a stati finiti.

Di particolare interesse sono i *sistemi dinamici a stati finiti,* nei quali tutte le variabili: gli ingressi, quelle di stato, le uscite, possono assumere solamente un numero finito di valori.

Si consideri una particolarità in più di tali sistemi: i successivi valori avvengono solamente in determinati istanti di tempo, ottenendo un sistema discreto. In questo caso il sistema si definisce *un automa.*

Affinché un sistema possa definirsi *automa* occorre che:

- Sia un sistema dinamico (o algebrico).
- Sia a stati finiti.
- I successivi valori degli ingressi, e di conseguenza quelli degli stati e della uscite, vengano assunti solamente in *determinati istanti di tempo*.
- Si abbia così una discretizzazione nel tempo dei valori assunti dalle variabili: queste potranno assumere solamente specifici valori nei determinati istanti di tempo, ma non i valori intermedi.

Definizione di automa

Si definisce *automa* un sistema dinamico (o algebrico) a stati finiti, nel quale il passaggio da uno stato all'altro avviene solamente in determinati istanti di tempo, ottenendo una discretizzazione degli stati e del sistema.

Qui si fa l'ipotesi che i parametri degli elementi che compongono l'automa siano costanti nel tempo; si considerano quindi solamente *sistemi stazionari*.

Nello studio degli automi conviene esprimere le variabili in particolari forme che meglio si adattino a descrivere l'evolversi del sistema.

Analizziamo di nuovo gli insiemi delle variabili che interessano un sistema, adattandoli, nella forma, ad essere utilizzati per un *automa.*

1.12.1 Insieme dei tempi

È un insieme discreto. Si possono associare i successivi istanti di tempo, nei quali avviene il passaggio di stato, all'insieme dei numeri interi con segno, in modo da poter distinguere un istante dal successivo o dal precedente. Così:

$$\ldots t_{-1}, t_0, t_1, t_2, \ldots . t_{i-1}, t_i, t_{i+1} \ldots$$

possono rappresentare i successivi istanti di tempo, nei quali avviene il passaggio da uno stato all'altro. Così t_{i-1} precede t_i e questo è seguito da t_{i+1}

1.12.2 Insieme degli ingressi

Nell'automa le variabili di ingresso forniranno i valori nei determinati istanti di tempo; per cui costituiscono un insieme discreto, del tipo:

$$x(t_{-1}), x(t_0), x(t_1), \ldots . x(t_{i-1}), x(t_i), x(t_{i+1}), \ldots$$

Essi appartengono all'insieme delle possibili funzioni di ingresso.

1.12.3 Insieme delle uscite

Come per gli ingressi l'automa fornirà i valori delle uscite nei determinati istanti di tempo; per cui le uscite costituiscono un insieme discreto, del tipo:

$$y(t_{-1}), y(t_0), y(t_1), \ldots . y(t_{i-1}), y(t_i), y(t_{i+1}), \ldots$$

1.12.4 Insieme degli stati finiti

È un insieme finito: può assumere solamente un numero finito di valori.

Nell'automa, in ogni determinato istante, si ha una *variabile di stato* che lo descrive e ne definisce *lo stato*.

Questo dipende:

- dalla variabile di ingresso e dalla sua storia.
- dall'istante iniziale t_0 da cui si inizia ad analizzare l'evoluzione del sistema.
- dal valore che ha la variabile di stato nell'istante iniziale.
- dall'istante in cui si sta considerando l'evoluzione dell'automa.

Occorre puntualizzare, che nella funzione di trasformazione di stato non compare il tempo come variabile esplicita, avendo preso in considerazione solamente sistemi *stazionari.*

Nella descrizione dell'evoluzione del sistema, non ha importanza quale sia l'istante iniziale, assunto come riferimento per lo studio dell'evoluzione del sistema fino all'istante considerato.

Come istante iniziale se ne può assumere uno qualsiasi tra quelli determinati: basta che, in corrispondenza ad esso, si associ la relativa variabile di stato che lo definisce in quell'istante.

Da ciò ne viene che come istante iniziale, si può assumere quello precedente all'istante nel quale si vuole determinare lo stato del sistema e la relativa uscita in corrispondenza della immissione della variabile di ingresso.

Indichiamo così con t_i l'istante iniziale in cui si determina il passaggio dallo *stato* $S(t_i)$ a quello successivo $S(t_{i+1})$ nell'istante t_{i+1}, ove si vuole considerare l'evoluzione del sistema.

Lo stato iniziale $S(t_i)$, che si ha nell'istante iniziale t_i in cui inizia il passaggio di stato, lo chiameremo *stato attuale.*

Lo stato $S(t_{i+1})$ verso il quale evolve il sistema nell'istante successivot_{1+1} lo chiameremo *stato futuro*.

1.12.5 Funzione di stato futuro

Con queste precisazioni, ne viene che la storia delle variabili di ingresso si restringono al valore che esse hanno nell'istante t_i. Inoltre, la *variabile di stato*, nell'istante iniziale, è *l'attuale* $S(t_i)$, mentre la variabile di stato che si vuole determinare è quella dello *stato futuro* $S(t_{i+1})$.

La funzione di trasformazione di stato, tenendo conto di aver considerato sistemi stazionari, si presenta nella forma:

$$S(t_{i+1}) = F(S(t_i), x(t_i)) \quad (1.24)$$

Per brevità di rappresentazione l'istante t_i si indicherà con il simbolo *"i"* (sottintendendo che si riferisce al tempo t_i); così *"*t_{i+1}*"* si indicherà con *"i+1"*.

La funzione di trasformazione di stato viene rappresentata nella forma:

$$S(i+1) = F(S(i), x(i)) \quad (1.25)$$

La funzione di trasformazione di stato (1.25) esprime il passaggio dell'automa dallo stato attuale nell'istante *"i"* a quello futuro *"i+1"*. Ad essa si dà nome di *funzione dello stato futuro*

Si definisce *funzione dello stato futuro* di un automa la funzione di trasformazione di stato, che determina la *variabile di stato futuro* del sistema $S(i+1)$ nell'istante futuro *"i+1"*, in relazione al valore che la variabile di stato ha nell'istante attuale *"i"* e al valore che la variabile di ingresso ha nello stesso istante.

Si osservi che i termini nella funzione di stato futuro (1.25) hanno il seguente significato.

$S(i+1)$ Variabile di stato futuro. Rappresenta la variabile di stato dopo la transizione: quella che si ha nell'istante futuro *"i+1"*, successivo a quello attuale *"i"*

$S(i)$ Variabile di stato attuale. È la variabile di stato che si ha nell'istante attuale *"i"* prima della transizione.

$x(i)$ Variabile di ingresso nell'istante attuale *"i"*. Essendo questo anche l'istante iniziale, la $x(i)$ definisce anche la storia dell'ingresso.

1.12.6 Funzione di trasformazione d'uscita

È evidente che la discretizzazione del tempo, degli ingressi e degli stati, comporta la conseguente discretizzazione delle uscite.

Le uscite $y(i)$ sono legate alle variabili di stato attuale e agli ingressi secondo l'espressione formale:

$$y(i) = G(S(i), x(i)) \quad (1.26)$$

Essendo l'automa considerato come un il sistema stazionario, il tempo nella espressione (1.26) non compare come variabile esplicita.

Si noti che l'insieme dei tempi è espresso dai termini della successione *...i-1, i, i+1, i+2, ...* che rappresentano i successivi istanti di tempo nei quali si ha una transizione: il passaggio da uno stato all'altro.

La cadenza del tempo può avvenire in modo sincrono, scandita da un orologio (clock) oppure no.

Si possono così considerare due tipi di automi: *Sincroni* ed *asincroni.*

1.13 Automi sincroni

Negli automi sincroni le transizioni di stato avvengono in corrispondenza di istanti di tempo scanditi da un segnale di sincronizzazione.

Lo stato futuro $S(i+1)$ nell'istante *"i+1"* sarà ovviamente la conseguenza del valore della variabile di stato e di ingresso considerati nell'istante precedente *"i"*. Dove gli istanti *"i"* ed *"i+1"* sono scanditi da un orologio.

Saranno ovviamente valide le due espressioni formali:

$$S(i+1) = F(S(i), x(i)) \quad (1.25)$$
$$y(i) = G(S(i), x(i)) \quad (1.26)$$

1.13 Automi asincroni

In un automa asincrono non vi è un segnale di clock che scandisce il tempo di transizione.

Nell'automa asincrono si ha un passaggio di stato per effetto della variazione o di una variabile di ingresso primario o, anche, di una variabile di stato del sistema.

È da rimarcare che si può avere una transizione anche con la variabile di ingresso primaria che rimane costante mentre varia quella di stato. Ciò si è riscontrato nell'esempio del circuito sequenziale del volume 2° *(punto 6.1.2)* con funzione di trasmissione:

$$y = x \cdot \bar{y}$$

nel quale, rimanendo lo start x attivato, si ha nei successivi istanti di tempo una variazione dello stato e, risultando sempre $S(i+1) \neq S(i)$, si innesca un ciclo continuo di passaggi di stato, senza raggiungere mai uno stato stabile fino a che non si disattiva lo *Start*.

Stato stabile

In un determinato istante *"i"* uno stato è stabile quando, restando costanti gli ingressi primari $x(i)$, lo stato futuro $S(i+1)$ risulta uguale a quello attuale:

$$S(i+1) = S(i) \quad \text{con } x(i+1) = x(i) \quad (1.27)$$

Stato instabile

In un determinato istante *"i"* uno stato è instabile quando, restando costanti gli ingressi primari $x(i)$, lo stato futuro $S(i+1)$ risulta diverso da quello attuale:

$$S(i+1) \neq S(i) \quad \text{con } x(i+1) = x(i) \quad (1.27)$$

In conclusione:

In una evoluzione di un automa da un passaggio di stato all'altro con ingresso costante, si raggiunge uno stato stabile, quando il valore della variabile di stato futura $S(i+1)$ risulta uguale a quello della variabile di stato attuale $S(i)$.

$$S(i+1) = S(i) \quad \text{con} \quad x(i+1) = x(i) \qquad (1.27)$$

Da quanto detto il passaggio da uno stato all'altro $S(i) \rightarrow S(i+1)$ avviene o comandato da un clock che scandisce gli istanti *"...i, i+1..."* , oppure, con un ritardo Δt. (dovuto agli elementi inerziali), per effetto della variazione di una variabile di stato o dell'ingresso primario.

In quest'ultimo caso Δt rappresenta l'incremento di tempo che si ha tra l'istante attuale *"i"* e l'istante futuro *"i+1"*.

Si ha formalmente:

$$S(i+1) = F(S(i), x(i)) \qquad (1.25)$$

La variabile di stato compare nei due membri con significato e valore diverso: nel II membro è indicato il valore che esso ha attualmente, nell'istante attuale *"i";* nel I membro è indicato il valore che assumerà, nell'istante futuro *"i+1"*, ritardato di Δt rispetto ad *"i"*.

Il valore $S(i+1)$ assunto dalla variabile di stato futuro avviene con un ritardo di tempo Δt rispetto al valore assunto dalla variabile di stato attuale $S(i)$. Dove Δt è la differenza tra i due istanti *"i+1"* ed *"i"*.

$$S(i) \xrightarrow{\text{ritardo } \Delta t} S(i+1)$$

Nei riguardi della determinazione del valore delle variabili di stato futuro, le variabili di stato attuale si presentano come degli ingressi.

Le variabili di stato attuali $S(i)$ si denominano *"ingressi secondari"* per distinguerli da quelli $x(i)$, provenienti dall'esterno dell'automa, ai quali si è associato l'aggettivo di *primari.*

1.15TIPI DI AUTOMI

l'automa finora descritto è un sistema dinamico a stati finiti discreto, caratterizzato dalle due funzioni:

Funzione dello stato futuro $S(i+1) = F(S(i), x(i))$ (1.25)

Trasformazione d'uscita $y(i) = G(S(i), x(i))$ (1.26)

In generale la funzione di *trasformazione d'uscita* dipende oltre che dalla variabile di stato attuale $S(i)$ nell'istante *"i",* anche dall'ingresso primario $x(i)$. Questo tipo più generale di automa va sotto il nome di *Automa di Mealy.*

1.15.1 Automa di Mealy

Definizione

Si definisce *Automa di Mealy* un sistema dinamico a stati finiti, nel quale il passaggio di stato avviene in determinati istanti di tempo, con conseguente discretizzazione del sistema, e nel

quale la trasformazione d'uscita dipende oltre che dalla variabile di stato attuale $S(i)$, anche *esplicitamente* dalla variabile di ingresso primaria $x(i)$, nell'istante attuale *"i"*.

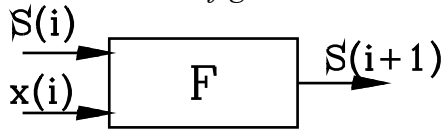
Più brevemente un *Automa di Mealy* è un automa, con sistema dinamico, nel quale la funzione di trasformazione d'uscita dipende esplicitamente dalla variabile di ingresso $x(i)$ primaria.

Si può effettuare una rappresentazione dell'automa di Mealy.

La funzione di stato futuro $S(i+1) = F(S(i), x(i))$, per brevità indicata con *F*, ha come ingressi: lo stato attuale $S(i)$ e la variabile di ingresso primaria $x(i)$.

Attraverso la funzione *F*, introdotte la variabile di ingresso primario $x(i)$ e quella di stato attuale $S(i)$, si ottiene la variabile di stato futuro $S(i+1)$.

fig.1.10

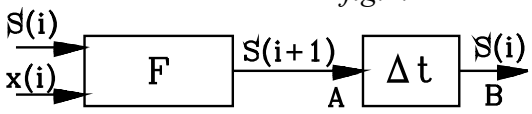


Nella rappresentazione a blocchi, si pone entro il blocco l'indicazione della funzione di stato futuro *F*, all'ingresso dello stesso la variabile di stato attuale $S(i)$ e la variabile di ingresso primaria $x(i)$. All'uscita del blocco, indicante la funzione di stato futuro *F,* viene indicata la variabile di stato futuro$S(i+1)$, prodotta da *F*.

Occorre osservare che la variabile di stato futura $S(i+1)$ si ottiene con un ritardo Δt rispetto a quella attuale $S(i)$: $S(i+1)$ rappresenta *il dopo*, mentre $S(i)$ il prima.

Rappresentata così nella *fig.1.11* la variabile di stato futuro $S(i+1)$, occorre andare a ritroso nel tempo del *ritardo* Δt per poter rappresentare la variabile di stato $S(i)$.

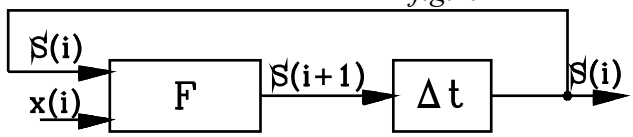
fig.1.11



Così, nel sistema a blocchi di *fig 1.11*, dopo $S(i+1)$, si pone il blocco dicante il ritardo Δt.

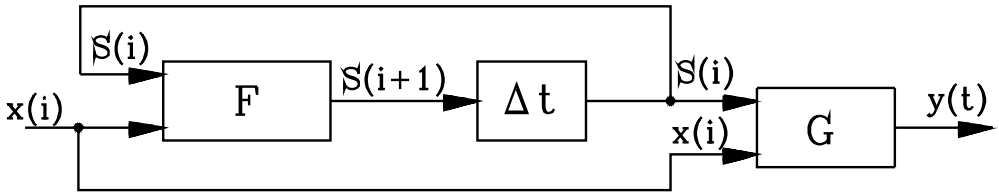
Andando dal punto *A* al punto *B* si torna indietro del tempo Δt. Il punto *A* rappresenta *il dopo*: l'istante futuro *"i+1"* dove è rappresentata la variabile di stato futuro $S(i+1)$; mentre il punto *B* rappresenta *il prima*: istante attuale *"i"* ove viene rappresentata la variabile di stato attuale $S(i)$.

fig.1.12



La variabile di stato attuale $S(i)$ che si ottiene nel punto *B* va poi collegata all'ingresso della funzione di stato futuro *F*, costituendo la variabile di ingresso secondaria *fig.1.12*.

Nell'automa di tipo Mealy la trasformazione d'uscita è funzione della variabile di stato attuale $S(i)$, ed anche *esplicitamente* della variabile di ingresso primaria $x(i)$, nell'istante attuale *"i"*. Per cui all'ingresso del blocco rappresentante la funzione di trasformazione d'uscita *G* pervengono sia la variabile di stato attuale $S(i)$ che la variabile di ingresso primaria $x(i)$. Si ottiene la rappresentazione in schema a blocchi dell'automa di Mealy di *fig.1.13*. *fig.1.13*



Consideriamo in modo descrittivo il possibile funzionamento dell'automa di Mealy rappresentato dallo schema a blocchi di *fig.1.13*.

È evidente che se nell'istante *"i"* si introduce un ingresso diverso da quello che si aveva nell'istante precedente *"i-1"*, attraverso la funzione *F,* si avrà il passaggio di stato da $S(i)$ ad $S(i+1)$:

$$S(i+1) = F(S(i), x(i)) \quad \text{con} \quad S(i+1) \neq S(i)$$

e in uscita si avrà: $$y(i) = G(S(i), x(i))$$

Ora può avvenire che nell'istante successivo *"i+1"* vari ancora l'ingresso: $x(i+1) \neq x(i)$. Allora si avrà comunque una nuova uscita, anche se rimane costante la variabile di stato.

Lo stato futuro all'istante *i+1* sarà:

$$S(i+2) = F(S(i+1), x(i+1))$$

L'uscita nell'istante *i+1* sarà:

$$y(i+1) = G(S(i+1), x(i+1)) \quad \text{con} \quad x(i+1) \neq x(i)$$

Questa uscita $y(i+1)$ è diversa da $y(i)$ anche se la variabile di stato è rimasta costante $S(i+1) = S(i)$. Infatti la trasformazione d'uscita, nell'automa di Mealy, dipende esplicitamente dalla variabile di ingresso primario $x(i+1)$, che si è supposta diversa da $x(i)$.

Si può dare il caso che nell'istante successivo *"i+1"* all'istante *"i"* resti costante la variabile di ingresso primaria: $x(i+1) = x(i)$, ma vari la variabile di stato: $S(i+1) \neq S(i)$. Per effetto di questa variazione si ha una nuova uscita nell'istante *i+1* , anche se rimane costante la variabile primaria di ingresso.

Lo stato futuro all'istante *i+1* risulterà:

$$S(i+2) = F(S(i+1), x(i+1))$$

L'uscita nell'istante *i+1* sarà:

$$y(i+1) = G(S(i+1), x(i+1)) \quad \text{con } S(i+1) \neq S(i)$$

Questa uscita $y(i+1)$ è diversa da $y(i)$ anche se la variabile di ingresso è rimasta costante, in quanto è variata la variabiledi stato $S(i+1) \neq S(i)$.

Può infine accadere che dall'istante *"i"* all'istante *"i+1"* rimangano costanti sia la variabile di stato, che quella di ingresso primaria. In tal caso si raggiunge uno stato stabile con il quale negli istanti successivi la variabile di stato rimane costante.

Infatti, si supponga che risulti:

$$x(i+1) = x(i) \quad \text{e} \quad S(i+1) = S(i)$$

Si avrà tra gli istanti $i \rightarrow (i+1)$:

il passaggio di stato: $S(i+1) = F(S(i), x(i))$

e l'uscita $y(i) = G(S(i), x(i))$ (1.26)

Tra gli istanti $(i+1) \rightarrow (i+2)$ si avrà:

il passaggio di stato: $S(i+2) = F(S(i+1), x(i+1))$

e l'uscita $y(i+1) = G(S(i+1), x(i+1))$ (1.27)

Avendo supposto che risulti: $x(i+1) = x(i)$ e $S(i+1) = S(i)$ osservano le (1:26) e (1.27) risulterà che le due uscite nei due istanti successivi *"i"* ed *"i+1"* sono uguali

$$y(i+1) = y(i)$$

Nell'automa di Mealy , dall'istante *"i"* all'istante *"i+1"* si ha una nuova uscita sia nel caso che vari solamente la variabile di stato, sia nel caso che vari solamente l'ingresso.

Per riconoscere l'automa di Mealy si ponga l'attenzione sul fatto che da un istante all'altro la variazione della variabile di ingresso primaria $x(i)$ può determinare comunque direttamente la variazione dell'uscita, anche se la variabile di stato rimane costante.

Nell'automa di Mealy la variabile di ingresso primaria influenza direttamente la variabile di uscita.

1.15.2 Automa di Moor

Come si è posto in rilievo , nell'automa di Mealy, la variazione della variabile di ingresso primaria $x(i)$ può determinare comunque una nuova uscita, anche se rimane costante la variabile di stato. Può essere preso in esame un automa dove ciò non avviene: la variabile primaria di ingresso non influenza direttamente in modo esplicito la variabile di uscita.

Si consideri un automa nel quale la funzione di trasformazione d'uscita dipenda solamente dalla variabile di stato e non esplicitamente da quella primaria di ingresso. A questo automa si da nome di *Automa di Moor.*

La funzione di trasformazione d'uscita può essere così rappresentata dalla espressione:

$$y = G(S(i)) \quad (1.28)$$

Se la funzione di trasformazione d'uscita non dipende esplicitamente dalla variabile di ingresso vuol dire che si tratta di un *sistema puramente dinamico*

Per la funzione di stato futuro resta l'espressione:

$$S(i+1) = F(S(i), x(i)) \quad (1.25)$$

Nell'automa di Moor si ha una nuova uscita solamente se varia la variabile di stato.

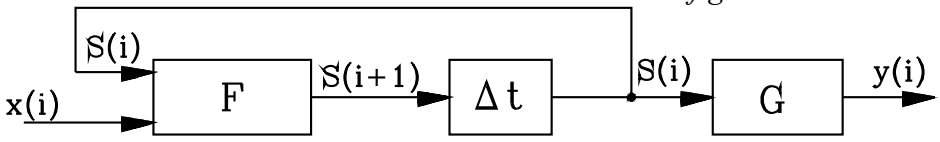
La variabile d'uscita da un istante all'altro rimane costante se risulta invariata quella di stato.

Definizione

Si definisce *Automa di Moor* un sistema puramente dinamico astati finiti, nel quale il passaggio di stato avviene in determinati istanti di tempo, con discretizzazione del sistema, e nel quale la trasformazione d'uscita dipende solamente dalle variabili di stato e non esplicitamente da quelle di ingresso primarie..

Lo schema a blocchi dell'automa di Moor si può ricavare da quello di Mealy , togliendo la dipendenza della funzione di trasformazione d'uscita dalle variabili di ingresso primarie $x(i)$.

fig.1.14



Funzione dello stato futuro $S(i+1) = F(S(i), x(i))$
trasformazione d'uscita $y = G(S(i))$

1.15.3 Elemento di ritardo

Un particolare automa di Moor è rappresentato dall'elemento di ritardo, nel quale un segnale d'ingresso viene trasferito in uscita con un certo ritardo, indicato con Δ. Ciò si ottiene con un automa di Moor nel quale la funzione di stato futuro e di trasformazione d'uscita sono unitarie: $F=1$. La relazione che lega l'ingresso alla variabile di stato futuro si esprime nella forma:

$$S(i+1) = 1 \cdot x(i)$$

e quella d'uscita:

$$y(i) = S(i)$$

Con il significato che la variabile di stato futuro assume il valore dell'ingresso nell'istante *"i+1"* con ritardo Δt pari alla differenza tra i due istanti *"i+1"* ed *"i"*; mentre la variabile di uscita $y(i)$ ha il valore della variabile di stato $S(i)$ assunto nell'istante *"i"*.

Tutto ciò corrisponde a dire che l'uscita assumerà il valore dell'ingresso con un certo ritardo Δ
Infatti:

Dall'istante *"i"* ad *"i+1"* $S(i+1) = x(i)$ (a)
$y(i) = S(i)$ (b)

Dall'istante *"i"* ad *"i+1"* $S(i+2) = x(i+1)$ (c)
$y(i+1) = S(i+1)$ (d)

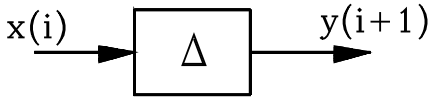
Comparando la (a) con la (d) si ottiene l'uguaglianza:

$$y(i+1) = x(i) \quad (1.29)$$

Nella quale è espresso che l'uscita nell'istante *"i+1"* è uguale alla variabile di ingresso immessa nell'istante precedente *"i"*.

Lo schema a blocchi è rappresentato in *fig.1.15*

fig.1.15



1.15.3 Automa puramente combinatorio

Come per i sistemi in generale, anche per gli automi si può considerare il caso della mancanza di memoria degli stati precedenti.

Si considera così un automa come sistema puramente algebrico e stazionario.

In tal caso non ha interesse la funziona di stato futuro e si può determinare una dipendenza diretta tra gli ingressi e le uscite.

Un automa di questo tipo si definisce *Automa puramente combinatorio*.

Definizione

Si definisce *automa puramente combinatorio* un sistema a stati finiti puramente algebrico (che non ha memoria degli stati precedenti), nel quale i passaggi di stato avvengono solamente in determinati istanti di tempo.

Nell'automa puramente combinatorio, mancando gli elementi di memoria, la risposta in uscita ad un segnale di ingresso è istantanea.

Anche se non è necessario, si può sempre , formalmente, definire una variabile di stato, la quale assume un valore nello stesso istante nel quale viene immesso il segnale di ingresso.

Tale funzione di stato non dipende dagli stati precedenti ma solamente dalla variabile di ingresso nell'istante *"i"*. Per cui la relazione tra la variabile di stato e quella di ingresso si può scrivere nella forma:

$$S(i) = F(x(i)) \quad (1.30)$$

Questa è una espressione puramente algebrica; anzi essendo, il sistema a stati finiti e discreto risulta di tipo combinatorio.

La trasformazione d'uscita è sempre di tipo algebrico (nella espressione non compaiono due stati riferiti a due istanti diversi).

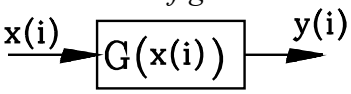
$$y(i) = G(S(i)) \quad (1.31)$$

Considerando la (1.30) si può scrivere:

$y(i) = G(F(x(i)))$ e quindi:

$$y(i) = G(x(i)) \quad (1.32)$$

fig.1.16



Lo schema a blocchi rappresenta il legame diretto tra la variabile di ingresso e quella di uscita.

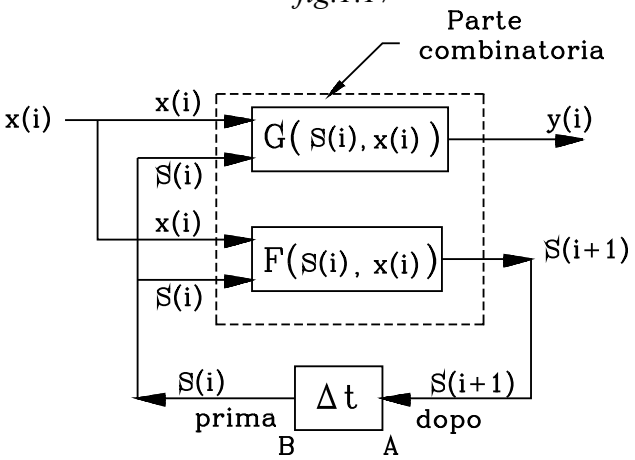
In un automa puramente combinatorio la funzione di trasformazione d'uscita è espressa da una relazione combinatoria che lega direttamente la variabile d'uscita a quella di ingresso.

1.15.4 Schematizzazione dell'automa di Mealy mediante la combinazione di un automa combinatorio e un ritardo unitario

Con l'introduzione dei due automi, ritardo unitario e automa combinatorio, l'automa di Mealy si può schematizzare come una combinazione di essi.

Si disegni lo schema a blocchi dell'automa di Mealy, ponendo i singoli blocchi uno sopra l'altro come in *fig.1.17*.

fig.1.17



Nei blocchi *G* e *F* entrano sia la variabile di ingresso primaria che quella secondaria, costituita dalla variabile di stato nell'istante attuale *"i"*.

Dal blocco *G* esce la variabile di uscita attuale $y(i)$, mentre la *F* predispone lo stato futuro verso cui deve evolvere il sistema.

Gli elementi inerziali determinano un ritardo del comando, per cui , quando la *F* sta predisponendo lo stato futuro, in ingresso il sistema si trova ancora nello stato dell'istante precedente.

Lo slittamento nel tempo dell'evolversi del sistema dallo stato attuale a quello futuro è rappresentato dal ritardo unitario Δt, che rappresenta la memoria del sistema. Gli altri due blocchi racchiusi nell'unico tratteggiato costituiscono una parte puramente combinatoria.

Così il generico automa di Mealy si può pensare coma la combinazione di un automa combinatorio e un ritardo unitario come rappresentato in *fig.1.18.*

Per il ritardo Δt , in A il blocco combinatorio sta predisponendo lo stato futuro $S(i+1)$ rispondendo alla variabile primaria di ingresso $x(i)$, mentre il sistema si trova ancora allo stato precedente $S(i)$.

Come si è detto nella rappresentazione delle funzioni, per semplicità, più variabili dello stesso tipo sono state indicate con un unico simbolo. Così quando si richiamavano le variabili di ingresso queste sono state indicate con l'unico simbolo $x(i)$.

fig.1.18

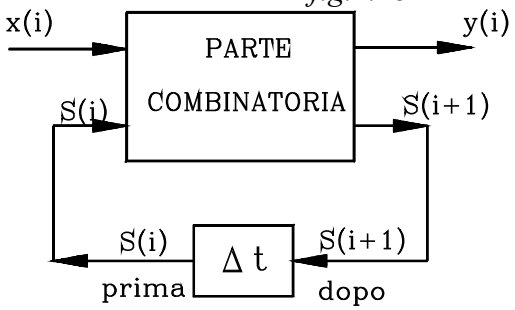
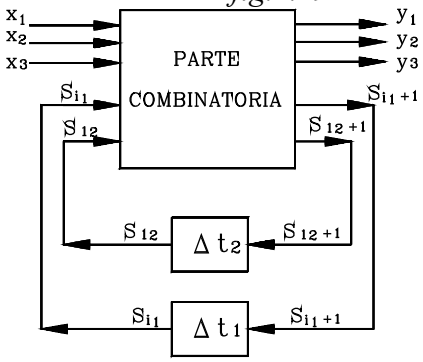


fig.1.19



Sia nelle espressioni delle funzioni che negli schemi a blocchi le diverse variabili si possono indicare come rappresentato in *fig.1.19*. Dove x_1, x_2, x_3,... rappresentano le diverse variabili di ingresso primarie, S_{i1}, S_{i2}, S_{i3}... rappresentano le variabili di stato attuali e così via...

1.16 STUDIO DELL'AUTOMA DI MEALY

Per spiegare il metodo di studio di un automa e in particolare l'automa di Mealy, conviene riferirsi ad un esempio pratico semplificato nelle condizioni al contorno.

Problema

Una navetta (o un carrello) contenente particolari attrezzature deve potersi spostare in 3 postazioni diverse di lavoro, indicate rispettivamente con 1, 2, 3.

In ciascuna postazione vi è un pulsante di chiamata e uno di permanenza in stazione.

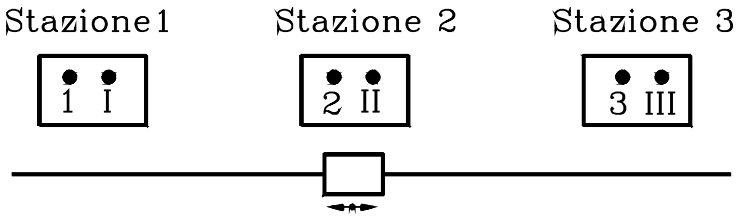
I pulsanti di chiamata sono contrassegnati dallo stesso numero indicante la postazione (1,2,3), mentre quelli di permanenza in postazione (stazione) vengono contrassegnati con un numero romano, corrispondente a quello di stazione (I, II, III).

Quando la navetta è in una postazione ed è pigiato il corrispondente pulsante di permanenza in stazione (con aggancio permanente), la navetta rimane in quella stazione anche se vengono pigiati i pulsanti di chiamata in altre stazioni.

La navetta può portarsi da una postazione all'altra quando in questa viene pigiato il pulsante di chiamata e non è pigiato il pulsante di permanenza nella postazione ove la navetta sta stazionando. Solamente in questo caso la navetta può portarsi da una stazione all'altra.

Durante lo spostamento della navetta da una postazione all'altra nessun pulsante può essere attivato.

fig.1.20



Il problema è stato espresso nei termini generali. In sintesi. Per chiamare il carrello in una postazione occorre che, quando si preme il pulsante di chiamata, la navetta non stia stazionando in una postazione con il relativo pulsante di permanenza agganciato (vuol dire che il carrello momentaneamente è utilizzato in quella postazione).

Una volta che il carrello ha raggiunto la postazione in cui è stato chiamato, per farlo rimanere in stazione basta agganciare il relativo pulsante di permanenza.

Dovendo introdurre un metodo di analisi e soluzione del problema, perché questo non diventi troppo gravoso per la trattazione, e per questo perda di chiarezza, conviene fare delle semplificazioni, tralasciando dei casi possibili di azionamento dei pulsanti.

Si tralascia il caso che preveda l'azionamento simultaneo di due o tre pulsanti di chiamata nelle stazioni.

Ciò non toglie alcuna validità alla spiegazione del metodo di studio: ne abbrevia lo svolgimento.

È evidente che capito il metodo di soluzione di un problema semplice , lo si può, poi, applicare alla soluzione di più complessi, scomponibili in più problemi semplici.

Descritto il sistema nella sue linee generali , occorre definire l'insieme delle variabili.

1.16.1 Insieme dei tempi

Il passaggio da uno stato all'altro avviene solamente in determinati istanti di tempo: quando risultano attivati i pulsanti di chiamata. I tempi non sono sincronizzati da un clock.

1.16.2 Variabili di ingresso primarie

Si assumono come variabili di ingresso l'attivazione dei pulsanti di chiamata e di permanenza in stazione.

Come variabile di ingresso si considera anche i caso di pulsante non premuto, contrassegnato con 0

Le variabili associate ai pulsanti di chiamata sono indicate con lo stesso numero che contrassegna la postazione con l'aggiunta dello "0" corrispondente al pulsante non premuto: 0, 1, 2, 3

Le variabili associate ai pulsanti di permanenza in stazione sono contrassegnati da un numero romano corrispondente a quello arabo che contrassegna la postazione, più lo "0" corrispondente al pulsante non premuto: 0, I, II, III

1.16.3 Insieme delle funzioni di ingresso

Le possibili funzioni delle variabili ingresso che si prendono in considerazione sono la combinazioni possibili tra i pulsanti di chiamata e quelli di permanenza in stazione.

Non si considerano (non fanno parte dell'insieme) le combinazioni di pulsanti dello stesso tipo come: (1,2) - (1,3) - (2,3) - (1,2,3); (I, II) - (I, III) - (II, III) - (I,II,III).

Le variabili che si pongono quindi come ingressi del sistema sono date dalle combinazioni tra i pulsanti di chiamata premuti (0,1,2,3) e quelli di permanenza in stazione (0, I, II, III)
Premuti pulsante chiamata 0 con quelli di permanenza (0,0) - (0,I) - (0,II) - (0,III)
Premuti pulsante chiamata 1 con quelli di permanenza (1;0) - (1,I) - (1,II) - (1,III)

Premuti pulsante chiamata 2 con quelli di permanenza (2;0) - (2,I) - (2,II) - (2,III)
Premuti pulsante chiamata 3 con quelli di permanenza (3;0) - (3,I) - (3,II) - (3,III)

1.16.4 Variabili di stato

Come variabili di stato occorre assumere le più significative. Nel caso in esame è evidente che lo stato che interessa attualmente è la postazione occupata dal carrello e lo stato futuro la postazione che occuperà nell'istante successivo.

Come variabili di stato si assumono le postazioni occupate dal carrello e indicate con i numeri 1, 2, 3.

1.16.5 Variabili d'uscita

L'uscita di un sistema è data dall'azione, segnale , ecc. provocato dalla variazione di una variabile di ingresso: primaria $x(i)$ o di stato $S(i)$ (variabile secondaria di ingresso).

Nel caso in esame, nel passaggio di stato si ha un moto del carrello da una postazione all'altra.

Come variabile di uscita si assume il numero di postazioni di cui si è spostato il carrello per effetto del pulsante di chiamata attivato.

Preso come riferimento la postazione n° 1, si considera positivo lo spostamento del carrello quando avviene nel senso che va da una postazione con numero inferiore ad una con numero superiore e negativo in senso inverso.

Così:

- Se il carrello si sposta dalla postazione 1 alla 3 → l'uscita è +2
- Se il carrello si sposta dalla postazione 2 alla 1 → l'uscita è –1

1.16.5 Condizioni al contorno

Per definire un sistema occorre imporre particolari condizioni al contorno, che regolano alcune relazioni tra gli ingressi e l'effetto sul sistema.

Abilitazione dei pulsanti

Durante lo spostamento del carrello tutti i pulsanti sono disabilitati.

Effetto preponderante tra i pulsanti di chiamata e quelli di permanenza in stazione

Occorre prevedere un effetto preponderante tra i due pulsanti di chiamata e di permanenza in stazione quando sono pigiati contemporaneamente.

Per il coretto funzionamento del sistema occorre che:

1. Il pulsante di chiamata abbia effetto preponderante rispetto a quello di permanenza in stazione, quando la navetta staziona in una postazione diversa da quella ove si trova il pulsante di chiamata e di permanenza in stazione.

   Così, se la navetta si trova nella stazione 2 e sono pigiati contemporaneamente il pulsante di chiamata 1 e di permanenza in stazione III (1, III), ha preponderanza il pulsante di chiamata 1. Infatti il pulsante III di permanenza in stazione non si aggancia non essendo la navetta nella stazione 3 (ricordiamo che il pulsante di permanenza in stazione si aggancia solamente quando la navetta è in quella stazione ove è posto il pulsante)

2 Il pulsante di permanenza in stazione è preponderante su quello di chiamata quando la navetta si trova nella stazione ove è installato detto pulsante di permanenza. Questo si aggancia e non permette al carrello di spostarsi in altre postazioni, fino a che non viene disattivato.

Così se la navetta è nella postazione 2 e sono pigiati i pulsanti (*3,II*), essa rimane nella postazione 2, in quanto il pulsante di permanenza in stazione II risulta attivato (essendo la navetta nella stazione ove è installato detto pulsante) e impedisce lo spostamento in altre stazioni.

In questo caso, nel quale il carrello si trova nella postazione 2 ove è installato il pulsante II di permanenza in stazione, questo prevale su quello di chiamata 3 nella postazione 3.

1.16.6 Tipo di sistema

Il sistema è ora compiutamente definito nelle variabili, nelle condizioni al contorno e possiamo analizzarne le caratteristiche.

- *Il sistema è un automa*
  Infatti è un sistema dinamico a stati finiti nel quale il cambiamento di stato avviene solamente in determinati istanti.
  È dinamico in quanto ricorda gli stati precedenti e, azionando un pulsante (ingresso) si ha uno stato futuro diverso a seconda dello stato attuale in cui si trova.

  È a stati finiti, in quanto questi possono essere solamente quelli contrassegnati con: 0, 1, 2, 3
  Le transazioni di stato avvengono solamente in determinati istanti.
- *Il sistema è un automa asincrono*
  Infatti non vi è un segnale di clock, che dà i tempi nei quali si ha una transizione di stato. Questa avviene quando viene pigiato un pulsante di chiamata in un'altra postazione e non è pigiato quello di permanenza nella postazione ove si trova attualmente il carrello.
- *Il sistema è un automa di Mealy*
  Ciò è evidente in quanto la combinazione dei, pulsanti di chiamata e permanenza in stazione pigiati influiscono direttamente sull'uscita, data dallo spostamento del carrello.

Si può ora definire la funzione di stato futuro e quella di trasformazione d'uscita.

1.16.7 Tabella della funzione dello stato futuro

La funzione di stato futuro definisce la legge d'evoluzione dell'automa dallo stato attuale a quello futuro. Viene espressa da una tabella, contenente: le variabili di ingresso , gli stati attuali, gli stati futuri.

Nella legge di evoluzione occorre definire come l'introduzione delle variabili d'ingresso x_i (combinazione dei pulsanti pigiati) e secondarie S_i conducono allo stato futuro S_{i+1}.

Si possono dare le seguenti ulteriori condizioni:

1. Raggiunto uno stato, questo non varia se rimangono costanti le variabili primarie di ingresso
2. Da un passaggio da uno stato all'altro si raggiunge sempre uno stato stabile, nel quale, restando costanti le variabili primarie di ingresso x_i, la variabile di stato futuro coincide con quello attuale:

$$\begin{cases} per\ x_i = Costante \\ S_{i+1} = S_i \end{cases} \qquad (1.33)$$

L'imposizione (1.) semplifica la tabella della funzione di stati futuri, in quanto come variabili di ingresso significativi si possono porre solamente le variabili primarie, tralasciando le secondarie , in quanto queste rimangono invariate alla costanza di quelle.

Tenendo conto della preponderanza imposta sui pulsanti di chiamata e di permanenza in stazione, definita nel punto 1.16.5, si può stendere la tabella della funzione di stato futuro.

Stazioni Stati iniziali	VARIABILI PRIMARIE DI INGRESSO Combinazione dei pulsanti pigiati di chiamata e di permanenza in stazione															
	0 0	0 I	0 II	0 III	1 0	1 I	1 II	1 III	2 0	2 I	2 II	2 III	3 0	3 I	3 II	3 III
1	1	1	1	1	1	1	1	1	2	1	2	2	3	1	3	3
2	2	2	2	2	1	1	2	1	2	2	2	2	3	3	2	3
3	3	3	3	3	1	1	1	3	2	2	2	3	3	3	3	3
	STATO FUTURO															

La tabella degli stati futuri è costituita da *3* righe e *16* colonne.

- Le *16* colonne si riferiscono agli ingressi primari, combinazioni dei pulsanti premuti di chiamata e di permanenza in stazione, riportate alla sommità delle colonne. È indicato con 0 il pulsante non premuto.
  Per esempio l'indicazione *"1 II"* sta a significare che sono premuti contemporaneamente il pulsante di chiamata *"1"* e il *"II"* di permanenza nella stazione *2*.
- Le tre righe si riferiscono agli stati iniziali nei quali si può trovare la navetta quando viene introdotta la variabile di ingresso. Gli stati iniziali sono segnati all'estremità sinistra delle righe.
- In ogni casella di incrocio tra la colonna indicante la variabile primaria introdotta (pulsanti premuti) e lo stato iniziale, si pone lo stato futuro verso il quale evolve l'automa.

Per comprendere la tabella, analizziamo gli stati futuri segnati nelle caselle della prima riga, in funzione dello stato iniziale, corrispondente al carrello nella posizione 1 (prima riga) e delle diverse combinazione dei pulsanti contemporaneamente premuti (indicati, ciascuno, alla sommità delle colonne).

Le caselle vengono qui designate, indicando prima la postazione attuale (stato iniziale) e poi la combinazione dei pulsanti premuti.

Casella *1 - 0 , 0*

Non è premuto alcun pulsante. la navetta rimane nella postazione *1: stato futuro 1*

Casella *1 - 0 , I*

La navetta è nella postazione 1. Premuto il solo pulsante "I" di permanenza in stazione, questo si attiva essendo la navetta proprio nella postazione ove esso è installato. La navetta rimane nella stazione 1*: stato futuro 1*

Casella *1 - 0 , II*

La navetta è nella postazione *1*. Premuto il solo pulsante *"II"* di permanenza nella stazione *2*, questo non si attiva essendo la navetta nella postazione diversa da quella ove è installato detto pulsante. La navetta rimane nella stazione 1*: stato futuro 1*

Casella *1 - 0 , III*

La navetta è nella postazione 1. Premuto il solo pulsante *"III"* di permanenza nella stazione 3, questo non si attiva essendo la navetta nella postazione diversa da quella ove è installato detto pulsante. La navetta rimane nella stazione 1*: stato futuro 1*

Caselle *1 -1 , 0 ; 1 -1 , I ; 1 -1 , II ; 1 -1 , III*

La navetta è nella postazione 1. Premuti i tasti indicati, la navetta rimane nella posizione *1*

Infatti nella combinazione *"1-0"* la navetta è chiamata nella postazione 1; nella combinazione *"1, I"* vi è la chiamata e la permanenza nella stazione *1*.

Nelle combinazioni *"1, II"* e *"1 , III"* il pulsante di chiamata 1 è preponderante su quello di permanenza in postazione diversa da quella ove si trova attualmente la navetta.

La navetta in tutti e quattro i casi rimane nella stazione 1*: stato futuro 1*

Caselle 1 -2 , 0 ; 1 -2 , II ; 1 -2 , III

La navetta è nella postazione 1. Premuti i tasti indicati, la navetta si porta nella posizione 2.

Infatti nel caso "2 , 0" è premuto solamente il pulsante di chiamata nella postazione 2.

Nelle combinazioni "2 , II" e "2 , III" risulta sempre preponderante il pulsante di chiamata nella postazione 2 rispetto agli altri pulsanti di permanenza in postazione diversa da quella ove attualmente si trova la navetta.

La navetta in tutti e tre i casi si porta nella stazione *2: stato futuro 2*

Casella *1 - 2 , I*

La navetta è nella postazione *1*. Se è premuto il tasto *"I"* di permanenza in stazione *1*, esso è attivato e prevale sul tasto di chiamata nella postazione 2. La navetta rimane nella stazione 1*: stato futuro 1*

Caselle *1 -3 , 0 ; 1 -3 , II ; 1 -3 , III*

La navetta è nella postazione 1. Premuti i tasti indicati, la navetta si porta nella stazione *3*, risultando preponderante il pulsante di chiamata *3* rispetto agli altri di permanenza in postazione diverse da quella ove attualmente si trova la navetta.

Casella 1 - 3 , I

La navetta è nella postazione *1*. Se è premuto il tasto *"I"* di permanenza in stazione 1, esso è attivato e prevale sul tasto di chiamata nella postazione *3*. La navetta rimane nella stazione 1*: stato futuro 1*.

Alla stessa maniera si può ragionare per gli stati futuri indicati sulle caselle delle altre due righe.

1.16.8 Grafo della funzione stato futuro

Una rappresentazione grafica efficace della funzione di stato futuro è data dal *grafo*, già adoperato nel vol. 2°, in occasione della descrizione del funzionamento delle memorie con preferenza alla cancellazione o alla scrittura.

Ricordiamo gli elementi essenziali per la rappresentazione:

- I valori assunti dalla variabile di stato vengono scritte entro cerchi, detti *nodi* del grafo.

- L'evoluzione del sistema da uno stato all'altro, per effetto della variazione di una variabile primaria di ingresso, viene rappresentata da un arco orientato, con freccia rivolta dallo stato di partenza a quello di arrivo.

- Sugli archi che congiungono i nodi vengono scritti gli ingressi che hanno determinato il passaggio dallo stato attuale del nodo di partenza a quello di arrivo, rappresentante lo stato futuro.
- Più ingressi, che determinano lo stesso effetto sul passaggio da uno stato all'altro, vengono scritti sullo stesso arco che li rappresenta.

- Se un ingresso mantiene inalterato lo stato, questo evento viene rappresentato da un arco, che parte dal corrispondente nodo e si chiude su di esso. Tale arco viene detto *autoanello*.

Si rappresenta ora il grafo.

- Le tre variabili di stato "1, 2 ,3", corrispondenti alle stazioni ove si può trovare la navetta, vengono scritte entro i cerchi costituenti nodi del grafo.
- Si disegnano gli archi indicanti i passaggi di stato:

$$1 \Leftrightarrow 2 \qquad 1 \Leftrightarrow 3 \qquad 2 \Leftrightarrow 3$$

sugli archi si scrivono gli ingressi che determinano il passaggio di stato da un nodo all'altro nel senso della freccia.

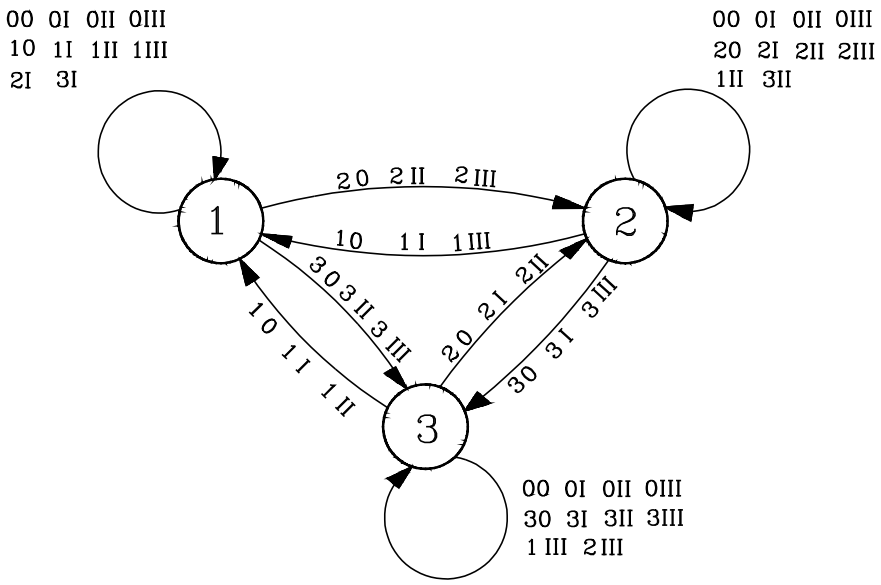
- Si disegnano gli autoanelli indicanti gli ingressi che non cambiano lo stato dell'automa.

Dall'analisi del grafo si possono effettuare le seguenti osservazioni:

1. Si ha il passaggio da uno stato iniziale ad uno futuro diverso, quando viene premuto il pulsante di chiamata, corrispondente al secondo stato e non è attivato quello di permanenza al primo stato ove risiede il carrello.

   Così si ha il passaggio dallo stato *1* al *2* nelle combinazioni *"2, 0 - 2, II - 2, III"* nelle quali è premuto il pulsante di chiamata *"2"* nella postazione *2*, e: o non è premuto alcun pulsante di permanenza in stazione *(0)*, oppure è premuto, oltre a quello di chiamata, un pulsante (*II* o *III*) di permanenza in una stazione diversa da quella ove si trova il carrello (stazione *1*)

fig. 1.21



2. Si ha un autoanello quando:
   a) Si preme solamente un pulsante di permanenza in stazione
      Infatti, o questo appartiene alla stazione ove si trova il carrello e quindi , essendo attivato, impone la permanenza nella postazione; oppure, appartenendo ad altre stazioni diverse da quella occupata dal carrello, non si attiva e non dà alcun effetto. Lo stato iniziale coincide con quello futuro.
   b) Quando viene premuto il pulsante di chiamata nella postazione ove si trova il carrello (corrispondente al nodo considerato) in combinazione o no con gli altri pulsanti di permanenza in stazione. Ad esempio sul nodo 1 si ha autoanello nelle combinazioni *"1, 0 - 1, I - 1, II - 1, III"*. Infatti *"1, 0"* impone la chiamata nella postazione 1 ove già si trova il carrello; la combinazione *"1, I"* impone la chiamata e la permanenza nella stazione 1; nelle combinazioni *"1, II" e "1, III"* è preponderante il pulsante di chiamata nella postazione *1* rispetto a quelli di permanenza in stazioni diverse da quella ove si trova il carrello.
   c) Il caso, più significativo è quello nel quale è premuto il pulsante di permanenza nella staziona ove si trova il carrello, che impone la permanenza in postazione, e viene premuto un altro pulsante di chiamata in una stazione diversa da quella di stazionamento del carrello. In questo caso è preponderante il pulsante di permanenza su quello di chiamata. Così nel nodo 1 (carrello attualmente in postazione 1) determinano autoanello le combinazioni dei pulsanti *"2, I"* e *"3, I"*, in quanto risulta attivato il pulsante di permanenza in stazione 1, che non permette lo spostamento del carrello in postazioni diverse da quella occupata.

1.16.9 Trasformazione d'uscita

Nell'automa di Mealy la trasformazione d'uscita dipende sia dallo stato attuale che dalle variabili di ingresso primarie.

Nel caso in analisi è chiaro che lo spostamento del carrello dipende direttamente dalla combinazione dei pulsanti premuti, costituenti le variabili primarie d'ingresso, e quindi il sistema è, come si è detto, un automa di Mealy.

La trasformazione d'uscita è definita da una tabella nella quale, in funzione dello stato attuale e della combinazione dei pulsanti premuti (variabili primarie d'ingresso) viene scritto lo spostamento di postazione del carrello ottenuto.

fig.1.22

Si dà un orientamento agli spostamenti del carrello, considerandoli positivi quando avvengono da una postazione contrassegnata da un numero più basso a quella con numero più alto.

Così se la navetta si sposta dalla postazione *1* alla *3* l'uscita è *"+2"*. Se , invece, lo spostamento avviene dalla postazione *2* alla *1*, l'uscita è *"-1"*.

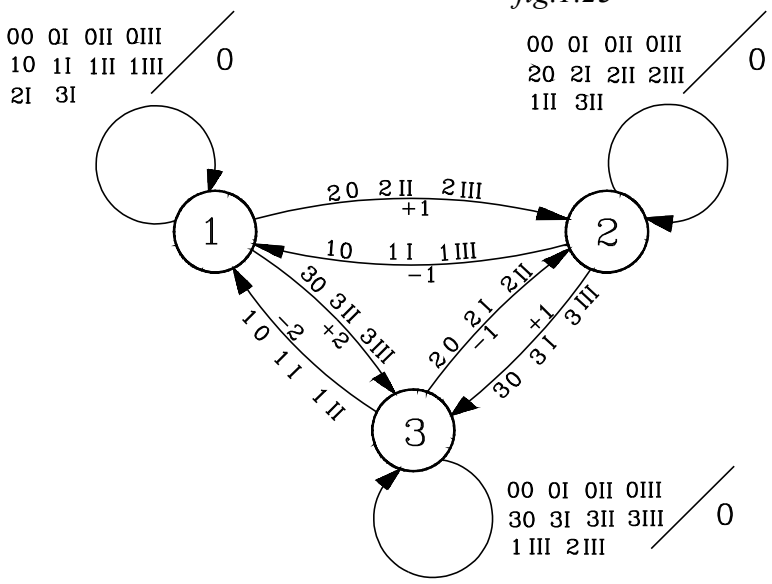
Quando la combinazione dei pulsanti premuti non provoca spostamento, questo si indica con *"0"*.

Analizzando la tabella degli stati futuri si ricava facilmente quella di trasformazione d'uscita.

Stazioni Stati iniziali	VARIABILI PRIMARIE DI INGRESSO Combinazione dei pulsanti pigiati di chiamata e di permanenza in stazione															
	0 0	0 I	0 II	0 III	1 0	1 I	1 II	1 III	2 0	2 I	2 II	2 III	3 0	3 I	3 II	3 III
1	0	0	0	0	0	0	0	0	+1	0	+1	+1	+2	0'	+2	+2
2	0	0	0	0	-1	-1	0	-1	0	0	0	0	+1	+1	0	+1
3	0	0	0	0	-2	-2	-2	0	-1	-1	-1	0	0	0	0	0
	USCITE															

Nel grafo si pongono sugli archi, indicanti il passaggio di stato, accanto alla combinazione dei pulsanti premuti costituenti le variabili d'ingresso, i valori delle uscite.

fig.1.23



1.1-7.1 Determinazione dello schema funzionale di un automa di Mealy

Per spiegare il metodo di determinazione dello schema funzionale ricavato dalle funzioni di stato futuro e di trasformazione d'uscita, conviene riferirsi ad un automa più semplice del precedente.

Tema

Degli oggetti sono spinti su una piattaforma dalla quale vengono prelevati per essere impaccati a 3 a 3.

Un sensore di prossimità rileva ogni oggetto che viene introdotto nella piattaforma, emettendo un segnale impulsivo.

Ogni volta che viene spinto il terzo oggetto nella piattaforma ed elaborato il terzo impulso dal trasduttore deve essere emesso un segnale luminoso.

Dopo il terzo oggetto deve iniziare di nuovo il conteggio da 1 a 3

Ripercorriamo brevemente il procedimento effettuato per lo studio del precedente automa.

Ingresso

Il segnale di ingresso è dato dall'impulso elaborato dal sensore e trasduttore di prossimità P_r. La variabile di ingresso può assumere due soli valori 0,1

Uscita

Il segnale di uscita è dato dall'accensione della lampada. L'uscita può assumere solamente due valori 0,1

Variabili di stato

La variabile di stato si fa coincidere con il numero che contrassegna l'oggetto che viene spinto nel porta oggetti e che fa parte del gruppo dei tre da impaccare.

La variabile di stato può assumere solamente i tre valori : "1, 2, 3".

Si tratta di un sistema dinamico a stati finiti, nel quale la transizione di stato avviene in determinati istanti di tempo (quando si introduce un oggetto nella piattaforma)

Da quanto esposto non si può associare ad ogni stato la rispettiva uscita. Infatti allo stato 3 possono corrispondere le due diverse uscite "0, 1" a seconda il valore della variabile di ingresso (0, 1).

L'uscita dipende in modo esplicito dalla variabile d'ingresso.

In conclusione, il sistema è un automa di Mealy

Tabella della funzione di stato futuro

Si procede come per l'automa precedente. In corrispondenza di uno stato attuale e di un ingresso si pone il corrispondente stato futuro.

Gli stati attuali si scrivono all'inizio delle righe; la variabili di ingresso sulla sommità delle colonne.

Nella casella di incrocio tra stato attuale e variabile di ingresso (valore 0,1 del sensore) si pone lo stato futuro.

STATO INIZIALE	VARIABILE DI INGRESSO		
	0	1	
1	1	2	STATO FUTURO
2	2	3	
3	3	1	

Così per esempio, se lo stato iniziale è 1 (vi è un oggetto nella piattaforma) e la variabile di ingresso è 1 (si introduce un altro oggetto), lo stato futuro di viene 2.

Tabella delle trasformazione d'uscita

La funzione di trasformazione d'uscita è riportata nelle seguente tabella. Si ha lo stato logico *1* solamente quando lo stato iniziale è *2* e la variabile di ingresso è *1.* Infatti, solamente quando viene spinto il 3° oggetto nella piattaforma viene emesso il segnale luminoso.

STATO INI-ZIALE	VARIABILE DI INGRESSO		
	0	1	
1	0	0	USCITE
2	0	0	
3	0	1	

Le variabili di stato costituiscono i segnali di memorie del sistema. Per poter risolvere il problema in logica binaria, occorre che tutte le variabili in gioco possano essere codificate in valori binari.

Le variabili di ingresso e di uscita sono già di per se stesse delle grandezze binarie "0, 1". Occorre codificare i tre stati "1, 2, 3" in combinazioni di valori binari.

Nei due volumi precedenti sono stati introdotte due tipi di memorie, rispettivamente: con precedenza alla cancellazione e precedenza alla scrittura. Queste, a seconda delle due variabili di ingresso, offrono due uscite opposte, associabili ai valori binari "0, 1".

Per codificare le tre variabili di stato in valori binari si fa corrispondere a ciascuna di esse una combinazione delle uscite binarie di un gruppo di *"n"* memorie.

Occorre così scegliere *"n"* memorie in modo che il numero totale delle uscite binarie da esse offerte siano superiori o uguali alla 3 variabili di stato.

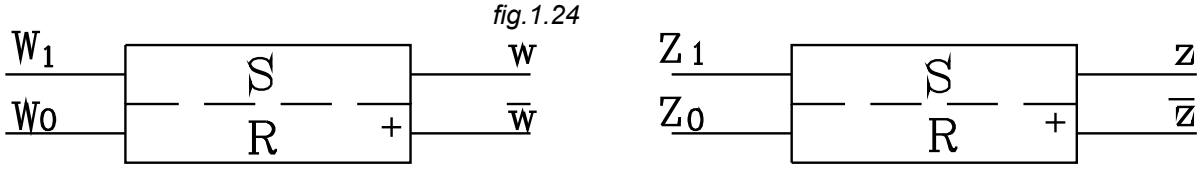
Ricordiamo che *"n"* memorie binarie rendono disponibili 2^n uscite. Per cui occorre un numero *"n"* di memorie tale che risulti:

$$2^n \geq 3$$

Da cui risulta che, per la codifica, necessitano *n=2* memorie con $2^2 = 4$ uscite delle quali, una non viene utilizzata.

Scegliamo come memoria quella con precedenza alla cancellazione, indicate rispettivamente con "W" e "Z"

fig.1.24



Associamo le tre variabili di stato alla seguenti combinazioni binarie delle uscite delle due memorie:

$$\begin{aligned} &Stato\ 1 \quad \rightarrow \quad \overline{w}\,\overline{z} \quad 0,0 \\ &Stato\ 2 \quad \rightarrow \quad \overline{w}\,z \quad 0,1 \\ &Stato\ 3 \quad \rightarrow \quad w\overline{z} \quad 1,0 \end{aligned} \qquad (1.34)$$

Si possono ora stendere le tabelle della funzione di stato futuro e di trasformazione d'uscita sostituendo alla indicazione degli stati la codifica binaria (1.34).

Riuniamo per concisione le due tabelle in un'unica, ponendo, accanto al codice binario dello stato futuro, l'uscita; determinate, queste, dalle variabili di ingresso e di stato nell'istante attuale *"i"*.

STATO INIZIALE	Stato iniziale in codice binario	VARIABILE DI INGRESSO		
		0	1	
1	0 0	0 0/ 0	0 1/ 0	STATO FUTURO/ USCITE in codice binario
2	0 1	0 1/ 0	1 0/ 0	
3	1 0	1 0/ 0	0 0/ 1	

Così considerando la 3° riga, si ha:

- Stato iniziale 3 : codice binario 1 0
- Con ingresso 0 lo stato rimane invariato: rimane allo stato 3, che in codice binario è 1 0. L'uscita è 0
- Con lo stato iniziale 3 e ingresso 1 (terzo oggetto posto sul portaoggetti) la variabile di stato torna nello stato iniziale *1*, che in codice binario è "0 0" e si ha il segnale luminoso in uscita. L'uscita è 1

Da quest'ultima tabella si possono ricavare due tabelle della verità.

Una tabella è di comando dell'uscita, che dà i valori dell'uscita *L* in funzione degli stati iniziali e degli ingressi.

Un'altra tabella è di comando di eccitazione delle memorie che determina i segnali di eccitazione e diseccitazione delle memorie di codifica degli stati, in funzione delle variabili di stato iniziali e degli ingressi.

Tabella di comando delle memorie

Nella tabella vengono riportati:

- Nella prima colonna gli stati iniziali nell'istante attuale *"i".*
- Nella seconda e terza colonna i valori delle memorie *W, Z* corrispondenti allo stato iniziale (questi coincidono con la codifica binaria dello stato).
- Nella 4° colonna viene riportato il valore della variabile di ingresso P_r, data dal segnale 0,1 del sensore di prossimità.
- Nella 5° colonna è riportato lo stato futuro raggiunto nell'istante *"i+1"* per effetto di quello iniziale e della variabile di ingresso immessa.
- Nella 6° e 7° colonna sono, riportati i valori delle memorie nell'istante futuro *"i+1"*, corrispondenti alla codifica dello stato futuro.
- Nella 8° e 9° colonna vengono riportati i segnali di settaggio e resettaggio delle memorie *W, Z* necessari per provocare l'evoluzione dallo stato iniziale a quello futuro.

Per determinare i segnali di comando delle memorie occorre analizzare i valori che esse hanno nello nell'istante iniziale *"i"* (stato iniziale) e quello che assumeranno nell'istante futuro *"i+1"* (stato futuro).

Dal confronto tra i valori delle memorie nello stato iniziale e quelli che assumeranno nello stato futuro si desumono i segnali, di settaggio e resettaggio da impartire alle memorie.

Così, per esempio, consideriamo la 2° riga della tabella. Lo stato iniziale *"1"* (prima colonna) è contraddistinto dai valori "0, 0" delle memorie: w=0 z=0 (2° e 3° colonna).

Stato iniziale delle memorie $w = 0 \quad z = 0$ (1.35)

Quando il sensore rileva un oggetto: $P_r = 1$ (4° colonna), allora l'automa subisce una transizione dallo stato 1 alla stato 2. Questo è riportato nella 5° colonna.

Lo stato futuro 2 raggiunto si è codificato con i valori binari "0, 1" . Questi sono i valori che dovranno rispettivamente assumere le memorie *W, Z.*

Stato finale delle memorie $w = 0 \quad z = 1$ (1.35)

Dal confronto degli stati iniziali (1.34) e finali (1.35) delle memorie si desume che:

- Lo stato della memoria *W* deve rimanere invariata: *w=0.*
  Ciò si può ottenere non attivando il settaggio della memoria, che rimane disattivata essendo tale nello stato iniziale: $S_w = 0$. Il segnale di reset R_w, affinché la memoria rimanga disattivata può indifferentemente assumere i valori 0 , 1 (se non si attiva R_w la memoria resta nello stato iniziale *w=0*, se si attiva R_w, a maggior ragione risulta *w=0*)
  Per esprimere l'indifferenza dei valori "0,1" che può assumere la variabile R_w si pone una "x" come contrassegno di detto valore.
- Lo stato della memoria *Z* deve subire una transizione dallo stato 0 allo stato 1, per cui occorre che si invii un segnale di settaggio $S_z = 1$, mentre deve essere inattivo il segnale di Reset $R_z = 0$, essendo la memoria con precedenza alla cancellazione.

In conclusione i segnali di comando delle memorie sono:

$$S_w = 0 \;/\; R_w = x \qquad S_z = 1 \;/\; R_z = 0$$

Alla stessa maniera occorre ragionare per le altre righe della tabella.

Stato iniziale automa	Stato iniziale delle memorie		I ingresso	Stato futuro automa	Stato futuro delle memorie		Segnali di Set e Reset delle memorie	
	w_i	z_i	P_r		w_{i+1}	z_{i+1}	S_w / R_w	S_z / R_z
1	0	0	0	0	0	0	0 / X	0 / X
	0	0	1	2	0	1	0 / X	1 / 0
2	0	1	0	2	0	1	0 / X	X / 0
	0	1	1	3	1	0	1 / 0	X / 1
3	1	0	0	3	1	0	X / 0	0 / X
	1	0	1	1	0	0	X / 1	0 / X
X	1	1	0	x	X	X	X/X	X/X
	1	1	1	x	X	X	X/X	X/X

Nella stesura della tabella occorre tenere sempre presente che la memoria è con disattivazione prevalente, e quindi, attivando contemporaneamente i segnali di set S e reset R, risulta preponderante quest'ultimo, per cui la memoria ha come uscita futura "0".

Così, considerando la 4° riga della tabella, la memoria *W* , dovendo passare dallo stato "0" allo stato "1", deve settarsi; mentre la memoria *Z* deve resettarsi, dovendo effettuare la transizione dallo stato iniziale 1 a quello finale "0".

Per settare la memoria *W* occorre che sia attivato il segnale di settaggio: $S_w = 1$ e inattivo quello di resettaggio: $R_w = 0$ (se quest'ultimo fosse contemporaneamente attivato resetterebbe la memoria).

Dovendo resettare la memoria *Z* occorre attivare il segnale di resettaggio: $R_z = 1$, mentre quello di settaggio può essere, indifferentemente, attivo o inattivo ($S_z = 1$ oppure $S_z = 0$). Questo perché la memoria è con prevalenza alla disattivazione e quindi, anche se è attivo il segnale di Set quello di Reset porta l'uscita a "0".

Dalla tabella della verità di comando delle memorie si possono ricavare i comandi di settaggio e resettaggio.

Come al solito la funzione binaria di comando riporta al primo membro lo stato futuro nell'istante *"i+1"* della variabile binaria che si vuole determinare , mentre al secondo membro viene riportata la combinazione dei valori logici degli ingressi e delle variabili di stato nell'istante attuale *"i".*

Segnale di settaggio della memoria W

Si ha una transizione della memoria *W* dallo stato "0" allo stato "1", con le combinazioni dei valori logici, dell'ingresso e degli stati iniziali delle memorie, riportate nella 4° riga, ove risulta $w_i = 0$ e $w_{i+1} = 1$.

Il segnale di set coincide con la variabile w_{i+1} che assume il valore "1". Questo avviene quando contemporaneamente risultano: $w_i = 0 \quad ,z = 1 \quad P_r = 1$. Quindi

$$S_w = w_{i+1} = \overline{w}_i \cdot z_i \cdot Pr$$

Segnale di resettaggio della memoria W

Ragionando alla stessa maniera del punto precedente, si conclude che il segnale di resettaggio della memoria *W*, che avviene quando questa ha la transizione dallo stato attuale "1" a quello futuro "0", si ottiene (6° riga) quando contemporaneamente risultano: $w_i = 1 \quad ,z = 0 \quad P_r = 1$.

Quindi:

$$R_w = \overline{w}_{i+1} = w_i \cdot \overline{z}_i \cdot P_r$$

Segnale di settaggio della memoria Z

La memoria *Z* effettua una transazione dallo stato logico "0" nell'istante *"i"* allo stato logico "1" nell'istante successivo *"i+1"* (riga 2°), quando contemporaneamente risultano: $w_i = 0 \quad ,z = 0 \quad P_r = 1$

Quindi:

$$S_z = z_{i+1} = \overline{w}_i \cdot \overline{z}_i \cdot Pr$$

.*Segnale di resettaggio della memoria Z*

La memoria *Z* effettua una transazione dallo stato logico "1" nell'istante *"i"* allo stato logico "0" nell'istante successivo *"i+1"* (riga 4°), quando contemporaneamente risultano: $w_i = 0 \quad ,z = 1 \quad P_r = 1$

$$S_z = \overline{z}_{i+1} = \overline{w}_i \cdot z_i \cdot Pr$$

Omettendo, nelle equazioni logiche binarie, le indicazioni dell'istante attuale e futuro, sottintendendo che al primo si riferiscono le variabili scritte al secondo membro e al secondo quelle scritte al primo, le espressioni di comando si possono scrivere nella seguente forma:

$$S_w = \overline{w} \cdot z \cdot Pr \quad (1.36)$$
$$R_w = w \cdot \overline{z} \cdot P_r \quad (1.37)$$
$$S_z = \overline{w} \cdot \overline{z} \cdot Pr \quad (1.38)$$
$$R_z = \overline{w} \cdot z \cdot Pr \quad (1.39)$$

Tabella della verità di comando dell'uscita

Nella stesura della tabella si riportano, in corrispondenza degli stati iniziali dell'automa, scritte all'inizio delle righe, rispettivamente:

- Gli stati iniziali delle memorie, scritte sulla sommità della *2°* e*3°* colonna, corrispondenti alla codificazione binaria dello stato iniziale dell'automa: Stato *1* $\rightarrow$ *0,0* ; Stato *2* $\rightarrow$ *0,1* ; Stato *3* $\rightarrow$ *1,0*
- Il valore binario della variabile di ingresso, corrispondente all'attivazione del sensore, riportato alla sommità della 4° colonna
- Il valore dell'uscita, corrispondente alla emissione del segnale luminoso

Stato iniziale automa	Stato iniziale delle memorie		I ingresso	USCITA Segnale luminoso
	w_i	z_i	P_r	L
1	0	0	0	0
	0	0	1	0
2	0	1	0	0
	0	1	1	0
3	1	0	0	0
	1	0	1	1
X	1	1	0	x
	1	1	1	x

Dall'analisi della tabella si rileva che il segnale di uscita è uguale a "1" quando le memorie e il segnale di ingresso hanno i seguenti valori logici: $w = 1 \quad z = 0 \quad Pr = 1$.

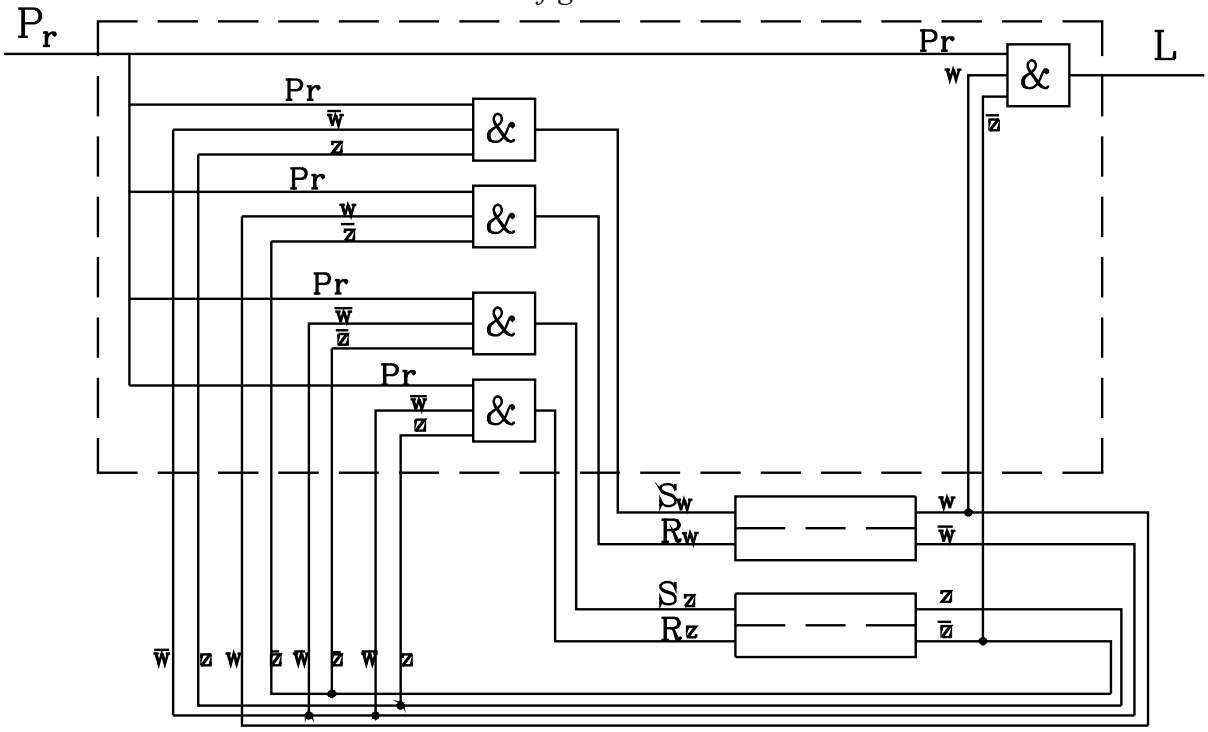
Il segnale di comando dell'uscita risulta quindi:

$$y = L = w \cdot \bar{z} \cdot Pr \quad (1.40)$$

Dalle equazioni logiche (1.36) , (1.37) , (1.38) , (1.39) , (1.40) , si ricava facilmente lo schema logico riportato in *fig.1.25*.

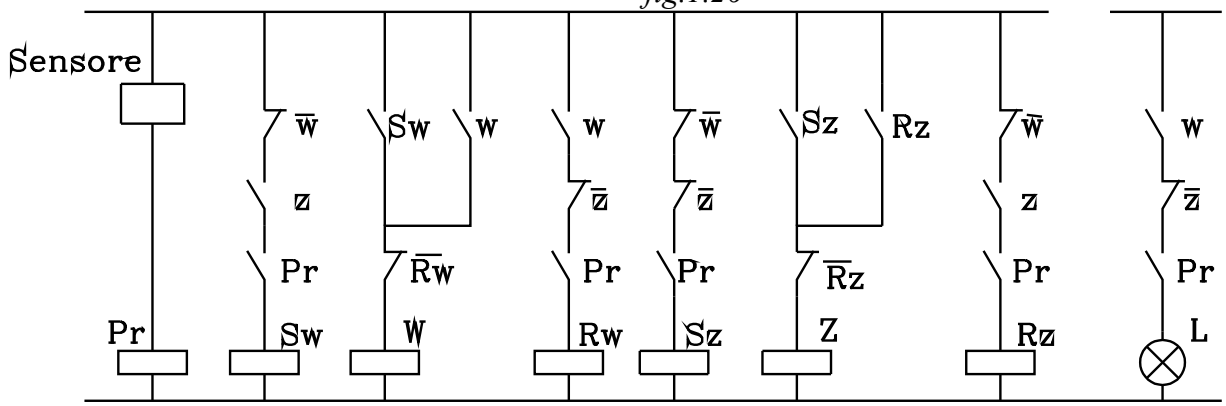
Lo schema è stato architettato in modo da rendere evidente la parte strettamente combinatoria (entro il riquadro tratteggiato)del sistema dalle memorie che ne delineano la dinamicità.

fig.1.25



Dalle equazioni logiche binarie si, ottiene lo schema funzionale:

fig.1.26



1.18Esempio di un automa di Moor

Come semplice esempio di automa di Moor si può considerare la memoria con precedenza alla cancellazione introdotta nel 2° volume .

Il problema pratico di riferimento può formularsi nella seguente maniera:

Si deve progettare un sistema elettropneumatico nel quale pigiando un pulsante P_1 si abbia l'alimentazione dell'impianto, che rimane attivato anche se si disattiva P_1. Premendo un pulsante P_2 di stop si teglie l'alimentazione, qualunque sia lo stato di P_1.

Si ripercorre brevemente il procedimento adottato per lo studio dei precedenti automi.

1.18.1 Variabili di ingresso primarie

Sono date dai pulsanti P_1 di start e P_2 di stop, che possono assumere i due stati finiti *0,1*.

1.18.2 Insieme delle variabili

Fanno parte dell'insieme delle variabili di ingresso tutte le combinazioni degli stati dei pulsanti di start P_1 e di stop P_2.

P_1, P_2: *0,0 - 0,1 - 1,0 - 1,1*

1.18.3 Variabile di uscita y

Come variabile di uscita si assume il segnale di alimentazione dell'impianto

1.18.4 Variabile di stato S

La variabile di stato *S* si può far coincidere con quella di uscita *y*.

1.18.5 Condizioni al contorno

Viene imposta la preponderanza dell'azione del pulsante di stop P_2 su quella dello start P_1. Si ha quindi che:

- L'impianto si attiva se viene pigiato il pulsante di start P_1 e non viene pigiato quello di stop P_2 .
- L'impianto si disattiva se viene pigiato il pulsante P_2 qualunque sia la condizione di P_1.
- L'impianto si disattiva se vengono pigiati contemporaneamente i due pulsante P_1, P_2.

1.18.6 Tipo di sistema

Il sistema è ora compiutamente definito nelle variabili, nelle condizioni al contorno e possiamo analizzarne le caratteristiche.

- *Il sistema è un automa*
  Infatti è un sistema dinamico a stati finiti nel quale il cambiamento di stato avviene solamente in determinati istanti.
  È dinamico in quanto ricorda gli stati precedenti e, azionando un pulsante (ingresso) si ha uno stato futuro diverso a seconda dello stato attuale in cui si trova.
  È a stati finiti, in quanto lo stato del sistema coincidente con l'uscita può assumere solamente gli stati *0,1*.
- *Il sistema è un automa asincrono*
  Infatti non vi è un segnale di clock, che dà i tempi nei quali si ha una transazione di stato.
- *Il sistema è un automa di Moor*
  Infatti la variabile di stato si è potuta far coincidere con quella di uscita, e questa quindi dipende solamente da quella e non esplicitamente dalla variabile di ingresso.

1.18.6 Funzione dello stato futuro

Avendo fatto coincidere la variabile di stato con quella di uscita, la funzione di stato futuro determina univocamente anche quella di trasmissione dell'uscita.

La funzione di stato futuro è definita come al solito da una tabella, nella quale vengono indicati gli stati futuri $S_{i+1} = y_{i+1}$, corrispondenti ai valori dell'uscita nell'istante *i+1* (y_{i+1}), in funzione delle combinazioni dei pulsanti P_1, P_2 premuti e dello stato $S_i = y_i$ del sistema coincidente con l'uscita nell'istante attuale *"i"*.

Riportiamo gli stati iniziali $S_i = y_i$ all'inizio delle righe della tabella, e le variabili di ingresso, combinazione dei pulsanti P_1, P_2 premuti, alla sommità delle colonne.

In ogni, casella di incrocio tra la riga corrispondente al valore dell'uscita nell'istante attuale *"i"* e della colonna, corrispondente alla combinazione dei pulsanti premuti, si riporta sia il valore dello stato futuro del sistema S_{i+1} nell'istante *"i+1"*, sia quello dell'uscita nello stesso istante y_{i+i} .

I due valori S_{i+1} e dell'uscita y_{i+1} coincidono.

Stato iniziale del sistema nell'istante *i*, coincidente con l'uscita	Ingressi P_1, P_2				
$S_i = y_i$	*0 0*	*0 1*	*1 0*	*1 1*	
0	*0 / 0*	*0 /0*	*1 /1*	*0 /0*	Stato futuro S_{i+1} coincidente con l'uscita y_{i+1}
1	*1 /1*	*0 /0*	*1 /1*	*0 /0*	

l'interpretazione della tabella è ormai evidente. Così , per esempio, considerando la seconda riga, riferita allo stato iniziale *1* dell'uscita, in corrispondenza della combinazione *0 1* dei pulsanti premuti *(start $P_1 = 0$ e stop $P_2 = 1$)* si ha come stato futuro $S_{i+1} = y_{i+1} = 0$.

Infatti premendo lo stop si ha la disattivazione dell'impianto e nella casella si è segnato come stato futuro $S_{i+1} / y_{i+1} = 0 / 0$.

Sulla stessa riga (stato logico iniziale *1)*, in corrispondenza della colonna indicante i due pulsanti premuti *(1 1)* si riportano i valori della variabile di stato futuro e dell'uscita uguali a *"0" (0 / 0)*.

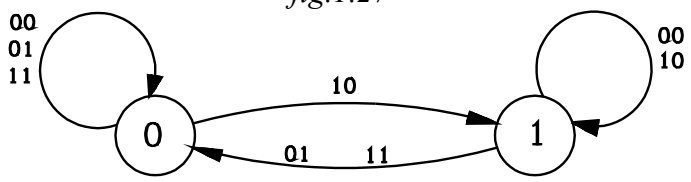
Infatti per la preponderanza dello stop P_2 sull'azione dello start P_1, si ha la disattivazione dell'impianto.

1.18.6 Grafo

Nei nodi del grafo si riportano gli stati possibili del sistema coincidenti con quelli delle uscite.

Nel caso in esame si hanno solamente due stati: *0 , 1.*

fig.1.27



In ogni ramo orientato, che indica l'evoluzione da uno stato ad un altro, si riporta la combinazione dei pulsanti premuti che la determinano.

Dalla lettura della tabella della funzione dello stato futuro si ottiene il grafo di figura *fig.1.27*

1.18.6 Schema logico

Dalla tabella della funzione dello stato futuro o dal grafo si ricava facilmente la funzione logica che regola il sistema.

Occorre considerare tutte le combinazioni tra lo stato iniziale dell'uscita y_i e gli stati dei pulsanti premuti, che determinano lo stato futuro con valore logico *1*.

Nel grafo si considerano i rami che conducono al nodo *1* e si considera la combinazione tra lo stato del nodo da cui parte il ramo e i pulsanti premuti scritti su di esso.

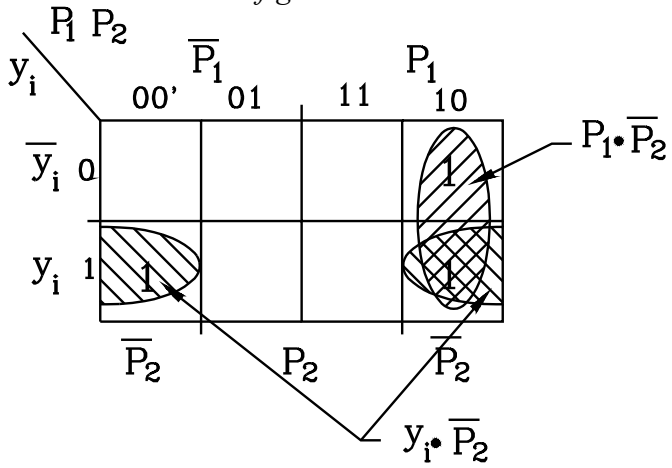
Sul nodo *1* sono indirizzati due rami:

- Un ramo parte dal nodo *0* e raggiunge il nodo 1. L'evoluzione rappresentata dal ramo è provocata dalla combinazione *1 0* dello stato dei pulsanti P_1, P_2.
  Quindi la combinazione tra lo stato iniziale dell'uscita *(0)* e gli stati dei pulsanti *(1, 0)* che fornisce lo stato futuro $y_{i+1} = 1$ è :
  $$\overline{y} \cdot P_1 \cdot \overline{P_2}$$
- L'altro ramo, costituente un autoanello, parte dal nodo *1* e torna su di esso. Le combinazioni degli stati dei pulsanti P_1, P_2 che lo determinano sono *0 0 - 1 0*.
  Quindi le combinazioni tra lo stato iniziale dell'uscita *(1)* e gli stati dei pulsanti *(0, 0 - 1,0)* che forniscono lo stato futuro $y_{i+1} = 1$ sono :
  $$y_i \cdot \overline{P_1} \cdot \overline{P_2} \quad ; \quad y_i \cdot P_1 \cdot \overline{P_2}$$

Ciascuna delle combinazioni considerate può rendere lo stato logico dell'uscita futura uguale ad 1; quindi l'operazione *OR* tra esse determina la funzione di stato futuro:

$$y_{i+1} = \overline{y} \cdot P_1 \cdot \overline{P_2} + y_i \cdot \overline{P_1} \cdot \overline{P_2} + y_i \cdot P_1 \cdot \overline{P_2} \quad (1.18.6.1)$$

fig.1.28



Nella *fig1.28* la funzione è rappresentata sulla mappa di Karnaugh.
La funzione può essere minimizzata, presentando due coppie di caselle adiacenti. Si ottiene:

$$y_{i+1} = P_1 \cdot \overline{P_2} + y_i \cdot \overline{P_2} \quad (1.18.6.2)$$

Ponendo in evidenza $\overline{P_2}$ si ottiene la funzione di stato futuro coincidente con la trasformazione di uscita:

$$y_{i+1} = (P_1 + y_i) \cdot \overline{P_2} \quad (1.18.6.3)$$

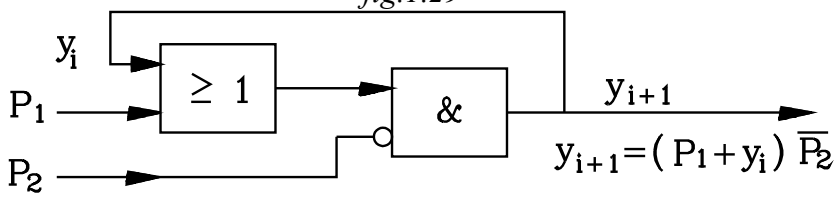
Per brevità la funzione si può scrivere nella forma:

$$y = (P_1 + y) \cdot \overline{P_2}$$

Sottintendendo che la y al secondo membro rappresenta lo stato dell'uscita nell'istante *"i"* attuale, mentre quella al primo indica lo stato futuro dell'uscita nell'istante *"i+1"*.

Dalla funzione dello stato futuro si può ricavare lo schema logico di *fig.1.29*

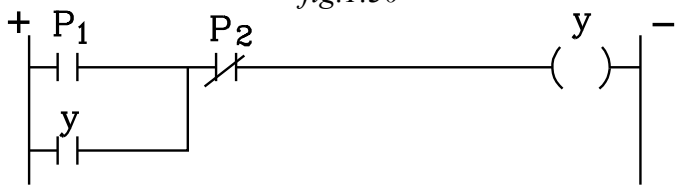
fig.1.29



Questo a seconda della tecnologia usata si può tradurre in un circuito pneumatico elettropneumatico ecc.

Dallo schema logico di *fig.1.29* si ricava il noto schema a contatti di *fig.1.30*

fig.1.30



2 CONTROLLORI A LOGICA PROGRAMMABILE - PLC

2.1 Generalità

I problemi di comando realizzati con tecnologie, elettroidrauliche, elettropneumatiche, possono, essere risolti utilizzando sistemi a logica *cablata o a logica programmata*.

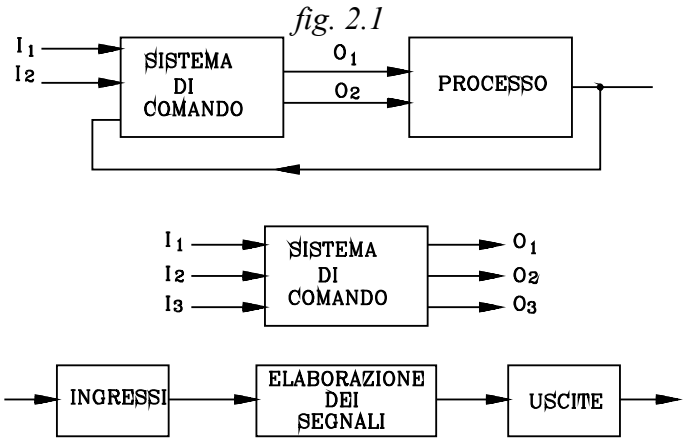
Con la logica cablata il problema viene risolto realizzando fisicamente il circuito con componenti di tipo pneumatico, oleodinamico ecc.

In tal modo il programma eseguito dal circuito è fisso. Dovendo effettuare una qualsiasi variazione del programma, occorre fisicamente variare il circuito *(cambiare collegamenti o componenti)*

Un sistema di comando è composto da:

- Un insieme di ingressi ai quali pervengono segnali, o imposti *(ad esempio START)* o provenienti da informazioni sullo stato del sistema o di processo *(finecorsa, sensori, parametri fisici che debbono subire un'elaborazione)*
- Un circuito cablato o un processore capaci di elaborare e trasformare le informazioni ricevute dagli ingressi secondo un programma prestabilito, fornendo in uscita i segnali voluti.
- Un insieme di uscite alle quali pervengono segnali elaborati dal programma e che vanno a comandare il sistema o il processo.

fig. 2.1

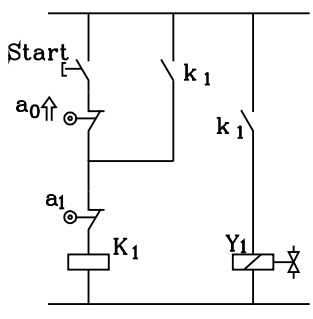


Nella logica cablata il sistema di comando (il blocco tra gli ingressi *I* e le uscite *O*) è costituito dai circuiti fisici: pneumatici ,oleodinamici, elettronici...

Nella logica programmata tale blocco è costituito da un processore, un computer dedicato alla soluzione di problemi di comando.

L'elaborazione dei segnali di ingresso e la trasformazione di essi in segnali di uscita avviene, in questo caso ,via software, attraverso un programma digitato su tastiera e memorizzato sotto forma di istruzioni nella memoria interna del *PLC*.

Con il PLC non occorre effettuare fisicamente il circuito che realizza il programma. Si debbono solamente collegare i sensori i finecorsa, ecc. agli ingressi e le uscite agli attuatori: elettrovalvole motori ...



Così si debba effettuare il semplice ciclo:

$$A^+ \; A^-$$

fig.2.2

Sia utilizzata una valvola di potenza monostabile.

Le equazioni logiche di comando sono:

$A^+ = a_0 \cdot Start$

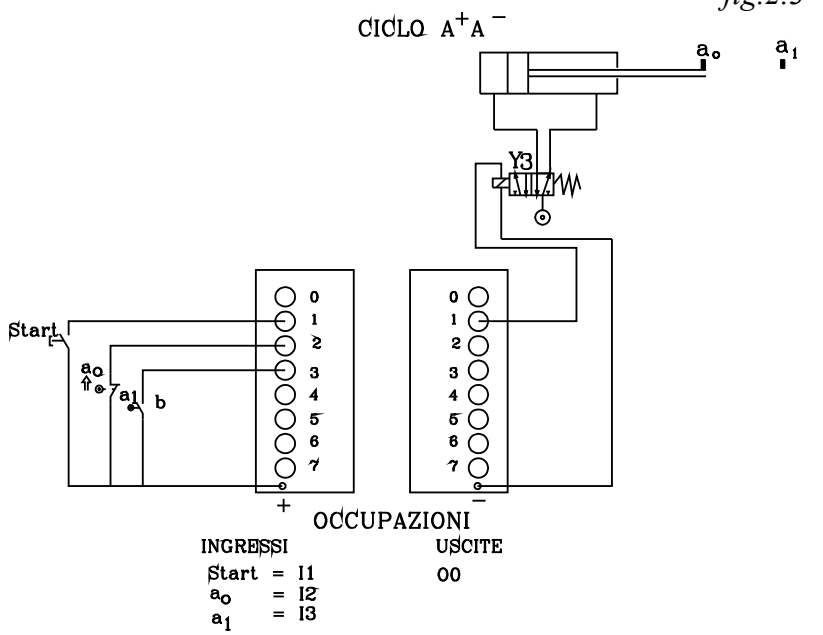
$A^- = a_1$

Adoperando una valvola di potenza monostabile il segnale di comando risulta:

$k_1 = \left(a_0 \cdot Start + k_1 \right) \cdot \overline{a_1}$

$Y_1 = k_1$

fig.2.3



Nella *fig.2.3* è schematizzato un modulo di ingresso ed uno di uscita del *PLC*, costituiti da *8* ingressi e *8* uscite.

Come si può notare basta effettuare i seguenti collegamenti:

Lo START ======> ingresso I1

Fine corsa a_0 ======> ingresso I2

Finecorsa a_1 ======> ingresso I3

Elettrovalvola Y1 ======> uscita O1

Il circuito, che realizza la sequenza, viene impostato in un programma, digitato su tastiera, e memorizzato nella memoria del *PLC*.

2.1.1 Modalità di esecuzione del programma

Nella logica cablata la lettura del circuito è immediata; le funzioni impostate nel circuito sono lette contemporaneamente

La velocità di risposta di un circuito elettrico sarà quella occorrente alla corrente per attraversare fisicamente le varie linee.

Si dice che la lettura in un circuito cablato è in parallelo.

Nella logica programmata, le operazioni da compiere vengono scomposte in una serie sequenziale di istruzioni semplici, che debbono essere eseguite una di seguito all'altra.

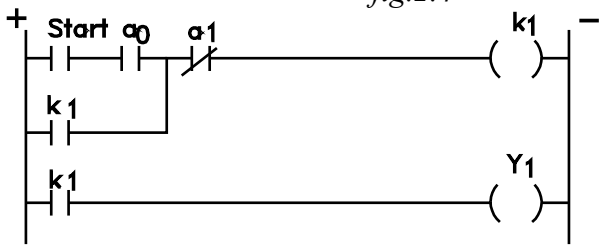
Così, esaminiamo il circuito già considerato.

Nel PLC, lo schema a contatti si svolge in una sequenza di linee orizzontali, poste tra due verticali, delle quali: quella di sinistra è considerata a potenziale positivo "+" e quella a destra, negativo "-".

I contatti sono quelli adoperati nello studio delle funzioni binarie.

Lo schema del circuito di *fig.2.34* si presenta nella forma:

fig.2.4



Le istruzioni vengono lette in modo sequenziale una alla volta e sono del tipo *(vedi oltre: linguaggio Ladder)*:

SE (START=1 E a_0=1 OPPURE K1=1) E NOT a_1

ALLORA ATTIVA K1
ecc.

-Il programma viene elaborato in sequenza, una istruzione alla volta a partire dal primo indirizzo di memoria *(indirizzo 0)*

-Il programma viene letto in modo ciclico, dalla prima istruzione all'ultima, per poi iniziare di nuovo dalla prima.

Il tempo di elaborazione dei segnali nella logica programmata è più lungo di quello occorrente per la logica cablata.

Occorre osservare che i cicli di lettura e di elaborazione nel PLC sono rapidi; per cui i tempi di risposta risultano in genere più piccoli di quelli di operatività degli attuatori.

In pratica, quindi, la risposta nella logica cablata appare come se fosse in parallelo.

2.1.2 Scelta del sistema

La scelta del tipo di sistema da adottare dipende da vari fattori.

- Tipo di segnale.
- Numero di segnali da trattare.
- Flessibilità.
- Ingombro.
- Economicità.
- Addestramento personale.

2.1.2.1 Tipo di segnale

Con la logica cablata si possono trattare direttamente segnali *Analogici*

Per adottare la logica programmata il segnale deve essere *digitale*.

Un computer può trattare solamente segnali digitali.

Per poter trattare un segnale analogico con un computer, occorre prima trasformarlo da Analogico in Digitale *(A/D*). Elaborato, secondo programma, il segnale di ingresso, dal computer uscirà un segnale che è in forma digitale. Questo, se occorre, dovrà, poi, essere trasformato in Analogico *(D/A)* per comandare il processo che lo richiede in detta forma.

2.1.2.2 N° di segnali

La logica cablata è adatta a trattare un limitato numero di segnali.

Un'alta quantità di essi comporterebbe l'impiego di un gran numero di equazioni logiche e di circuiti che le realizzino. In tal caso, risulta conveniente scegliere la logica programmata.

Si pensi soltanto, che al posto di montare numerosi componenti fisici e collegamenti, occorrenti per la realizzazione del circuito in logica cablata, scegliendo quella programmata, viene solamente digitato su tastiera un lungo programma.

2.1.2.3 Flessibilità

La logica cablata presenta una flessibilità scarsa o nulla.

Se necessita effettuare, con gli stessi attuatori, un cambiamento di programma, occorre cambiare il circuito ed effettuare i nuovi collegamenti.

Nella logica programmata, per cambiare il programma, basta effettuare le variazioni digitandole su tastiera (viene cambiato il programma via software e non via hardware)

2.1.2.4 Ingombro

La logica cablata è adatta a realizzare piccoli circuiti con poco ingombro e che non debbono subire variazioni di programma.

Quando viene richiesto di limitare l'ingombro, che si avrebbe con la complessità del circuito cablato, allora risulta conveniente la logica programmata.

2.1.2.5 Economicità

Riguardo alla economicità della scelta , da quanto si è detto, questa dipende da vari fattori.

La logica cablata comporta la realizzazione di un circuito fisso che ha poca flessibilità.

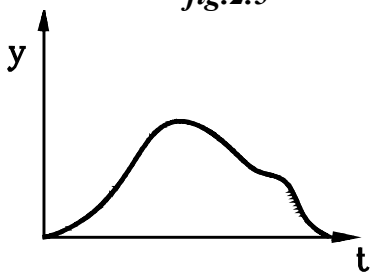
È evidente, che se occorre effettuare un circuito, dedicato ad una o più funzioni, che non debbono variare per tutta la vita del circuito stesso, la logica da scegliere sarà quella cablata.

La logica programmata risulta vantaggiosa: quando si vuole flessibilità - il circuito risulta complesso - vi sono molti segnali ed equazioni logiche da trattare.

2.1.3 Tipi di segnali

2.1.3.1 Segnale Analogico

fig.2.5



È un segnale che varia con continuità nel tempo e può assumere tutti i valori nel campo di variazione.

Si tratta di una funzione continua del tempo o scomponibile in tratti di funzioni continue.

I segnali analogici sono adatti alla regolazione di processi.

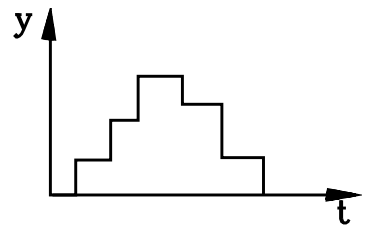
Ad esempio, la temperatura di un termometro a dilatazione varia con continuità, nella scala di misurazione: dal valore minimo al fondo scala.

Così sono Analogici :

- L'indicazione della pressione di un manometro meccanico
- L'indicazione della corrente di un amperometro a bobina mobile
- ecc.

2.1.3.2 Segnali Digitali

fig2.6



Un segnale digitale è discontinuo nel tempo. Esso può assumere, nel campo di variazione , un numero discreto di valori, tutti multipli di una unità fondamentale.

Il segnale digitale varia così a gradini da un livello zero fino, ad un livello max.

I valori compresi tra un livello e il successivo vengono riferiti allo stesso livello.

Il n° di livelli (valori), disponibili per rappresentare il segnale digitale, è fisso e dipende dal tipo di sistema.

Nel PLC, per rappresentare un segnale, vengono usualmente adoperati *8 bit*, corrispondente alla parola di *1 byte*.

Così, in un controllore, un segnale potrà essere espresso, al massimo, con 2^8=256 valori diversi nel campo della sua variazione nel tempo *(dal livello 0 a 255)*.

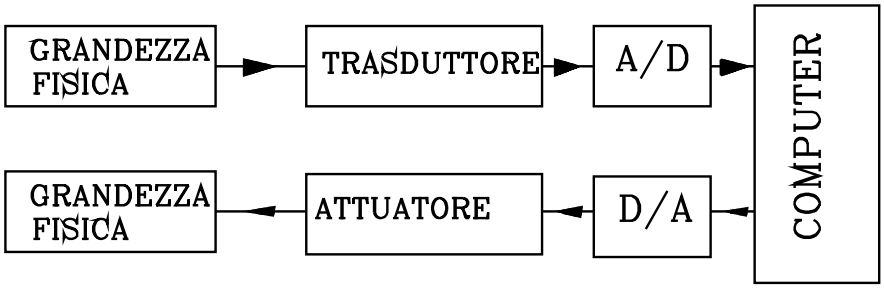
2.1.3.3 Conversione del segnale da Analogico a Digitale - A/D

La comunicazione di un calcolatore con il mondo esterno avviene con dispositivi di interfaccia seriali o paralleli. In entrambi i casi vengono acquisiti o prodotti segnali elettrici digitali.

Un segnale di ingresso *"I"* o di uscita *"O"* di un calcolatore deve essere sempre un valore, multiplo di una unità fondamentale q, tramutato in un insieme di bit detto "parola".

Se occorre manipolare una grandezza fisica, generalmente di natura analogica, bisogna prima, se non lo è, tramutarla in grandezza elettrica , quindi trasformarla da *Analogica in* Digitale *" A/D "*. La grandezza, manipolata dal calcolatore, dovrà, poi, essere trasformata di nuovo da Digitale in Analogica *"D/A"*.

fig.2.7

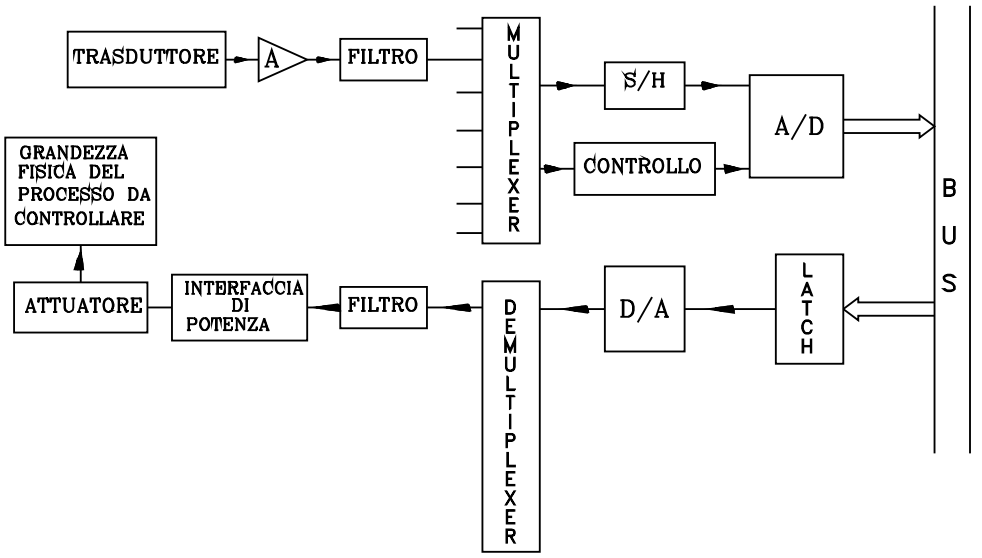


Così (vedi *fig.2.7*) una grandezza fisica (temperatura, forza, posizione...) viene trasformata da un trasduttore in grandezza elettrica e poi tramutata da Analogica in Digitale " *D/A* ". Introdotto nel computer il segnale digitale, questo viene elaborato, mediante programma, in un corrispondente segnale digitale in uscita.
Il segnale di uscita verrà poi trasformato da Digitale in Analogico " *D/A* " e inviato all'attuatore.
Questo effettuerà la funzione inversa del trasduttore: trasformerà la grandezza elettrica analogica in una grandezza fisica (rotazione, spostamento, pressione, forza ecc.)

Prima di analizzare più in dettaglio la trasformazione *A/D*, occupiamoci del problema riguardante la immissione di più segnali analogici nell'ingresso del calcolatore.

Riferiamoci allo schema a blocchi della *fig.2.8* seguente. In questa sono rappresentate più in dettaglio le operazioni alle quali vengono sottoposti i segnali, per controllare o regolare una grandezza fisica, attraverso la manipolazione in un computer.

fig.2.8



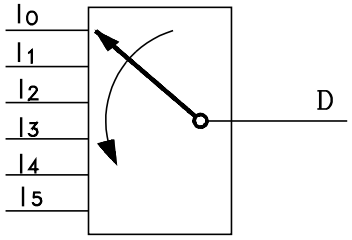
Anche qui, in linea generale, la grandezza fisica, trasformata in elettrica da un trasduttore, poi in digitale dal blocco *"A/D"*, viene inviata al *BUS* di un computer , e da questo elaborata, secondo programma. Dopo l'elaborazione è disponibile sul *BUS* il segnale di uscita. Il quale, trasformato in Analogico va a comandare l'attuatore, il quale determina il controllo o la regolazione del processo.

Nel sistema a blocchi, che viene assunto come riferimento, sono indicate altre operazioni che si vogliono analizzare.

2.2 Multiplexer

Nelle schede di acquisizione dati analogici vi possono essere più ingressi da dover leggere, che corrispondono a più segnali analogici, proporzionali alle grandezze fisiche da elaborare o funzioni di esse.

fig.2.9



Vi sono più ingressi analogici da elaborare e dovere quindi convertire in digitali, ma, nell'impianto, vi è un solo convertitore *A/D,* dato l'elevato costo di questo.

Per risolvere il problema, occorre che gli ingressi vengano letti uno alla volta, e il segnale analogico corrispondente venga inviato al convertitore *A/D*.

Occorre un selettore elettronico, detto MULTIPLEXER , che ponga alternativamente gli ingressi, uno alla volta, in comunicazione con il convertitore A/D.

l *multiplexer* è costituito da:

- N ingressi *"I"* corrispondenti ai segnali analogici da leggere.
- *n* linee di controllo (*o indirizzi*) *"S*", aventi lo scopo di portare, uno alla volta, i segnali di ingresso in uscita.
- una sola uscita *"D"*.

Ciascuna delle *n* linee di controllo può assumere i due stati *0-1*.

Ciascuna delle combinazioni dei due valori *0-1*, che si possono effettuare con le *n* linee di controllo, viene posta in *AND* con un particolare ingresso I_i.

Le combinazioni degli stati delle linee di controllo variano e si ripetono ciclicamente. Al verificarsi di *una combinazione,* viene portata nell'unica uscita *D* l'ingresso che è in *AND* con quella.

Il n° di combinazioni possibili degli stati 0,1 delle n linee di controllo sono: 2^n

Il n° di ingressi che possono essere selezionati con le n linee di controllo sono:

$$N=2^n$$

Esempio

Per una migliore comprensione ci si riferisca ad un semplice esempio:

Vi siano 2 linee di controllo indicate con S_0 S_1

Il numero di ingressi, che possono essere selezionati con le due linee di controllo *(indirizzi)* S_0 S_1, sono:

$N = 2^n$ per n=2 risulta $N = 2^2 = 4$

Effettuiamo la tabella della combinazioni possibili con i due indirizzi e, ad ogni combinazione, associamo un ingresso

INDIRIZZI		INGRESSI
S_0	S_1	I_i
0	0	I_0
0	1	I_1
1	1	I_2
1	0	I_3

Le combinazioni possibili, poste in AND con l'ingresso associato, vanno poi inviate all'unico OR di uscita

Gli ingressi I_0, I_1, I_2, I_3 sono selezionati uno alla volta, e portati in uscita del multiplexer, quando si verificano le rispettive combinazioni degli stati degli indirizzi in *AND* con essi, e riportate sulle corrispondenti righe della tabella.

Così se ad esempio si verifica la combinazione $s_0 = 0$ $s_1 = 0$ nell'OR di uscita vi sarà solamente l'ingresso I_0, che in AND con la combinazione $\overline{s_0} \cdot \overline{s_1}$ è allo stato logico 1:

$$\overline{0} \cdot \overline{0} \cdot I_0 = I_0$$

Precisamente:

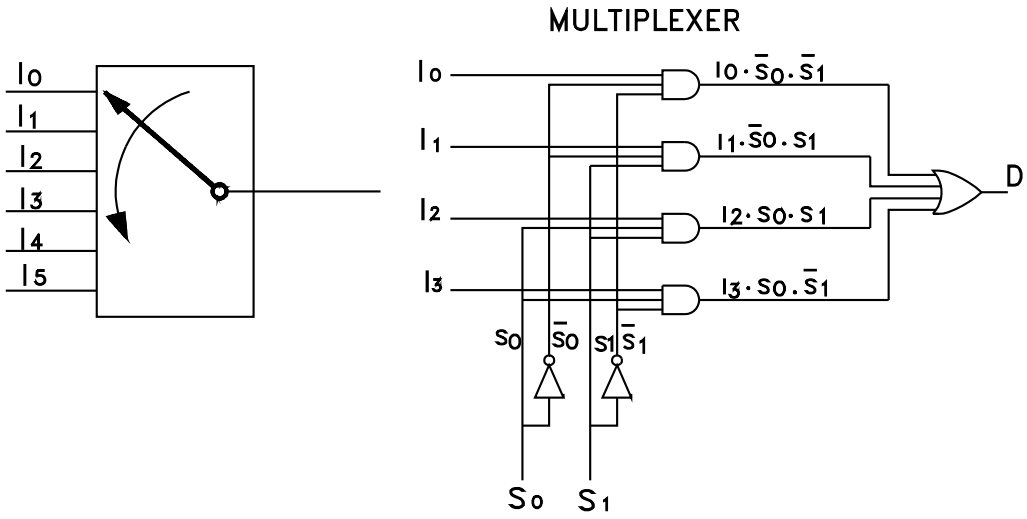
Ingresso I_0 $\overline{S_0}$, $\overline{S_1}$ *in AND con* I_0: $D = \overline{S_0} \cdot \overline{S_1} \cdot I_0$

Ingresso I_1 $\overline{S_0}$, S_1 *in AND con* I_1: $D = \overline{S_0} \cdot S_1 \cdot I_1$

Ingresso I_2 S_0 , S_1 *in AND con* I_2: $D = S_0 \cdot S_1 \cdot I_2$

Ingresso I_3 S_0 , $\overline{S_1}$ *in AND con* I_3: $D = S_0 \cdot \overline{S_1} \cdot I_3$

fig.2.10



Gli ingressi in *AND* con gli indirizzi, vanno inviati nell'unica uscita **D** la cui equazione logica risulta:

$$D = \overline{S}_0 \cdot \overline{S}_1 \cdot I_0 + \overline{S}_0 \cdot S_1 \cdot I_1 + S_0 \cdot S_1 \cdot I_2 + S_0 \cdot \overline{S}_1 \cdot I_3$$

2.3 Trasformazione A/D

L'acquisizione di grandezze analogiche e la loro trasformazione in digitali debbono essere viste come un problema di misurazione di grandezze variabili nel tempo.

Occorre tener conto di tre elementi:

- Processo di quantizzazione
- Durata della misura
- Frequenza di campionamento

2.3.1 Quantizzazione

La misura di una grandezza analogica, per la sua trasformazione in digitale, è legata al processo di quantizzazione.

Il segnale analogico, proveniente dai trasduttori, deve essere *quantizzato* per la conversione *A/D*.

Occorre, in un determinato tempo, ottenere un numero, "*quanto*", proporzionale o funzione della grandezza analogica da misurare, che risulti multiplo di una unità fondamentale *(bit)*.

L'operazione di quantizzazione si ripete in tempi successivi

I quanti ottenuti debbono essere trasformati, mediante una codifica binaria, in una parola *"insieme di bit"*

L'uscita del convertitore *A/D* è costituito da *n* linee, indicate con *0,1..n-1*, ciascuna delle quali può assumere gli stati logici *0,1*.

La combinazione degli stati logici delle linee di uscita dell'*A/D* determina la parola di codifica del quanto

L'ampiezza del quanto tra due valori consecutivi fornisce la sensibilità dell'apparato di misura: fornisce, cioè, la precisione della misura.

Il processo di quantizzazione, ovviamente, precede l'operazione di codifica binaria del quanto nella corrispondente parola, data dalla combinazione degli stati logici *0,1* delle uscite di *A/D*

Il valore del segnale analogico da misurare, del sistema considerato, può arrivare fino ad un valore massimo di fondo scala *VFSR (Voltag Full Scale Range)* , che rappresenta il valore di riferimento del sistema.

Detto segnale di riferimento "VFSR" viene suddiviso in un numero di quanti, corrispondente al n° di combinazioni, che si possono effettuare con le "n" linee digitali di uscita.

Con n uscite si possono effettuare 2^n combinazioni binarie 0,1

Si indichi con V_m il valore misurato, in un certo istante, nella discretizzazione del segnale analogico . Questo con una codifica a n linee viene espresso dalla somma di n valori associati agli *n* bit, indicati, rispettivamente con: $B_0, B_1, B_2, ..., B_{n-1}$ ciascuno dei quali può assumere i due valori 0,1.

Il bit B_0 è al livello più basso e B_{n-1} a quello più alto.

Una codifica usuale è quella nella quale i bit della parola vengono pesati, per la determinazione del valore discreto del segnale analogico , secondo le potenze crescenti della base 2, partendo dal bit al livello più basso B_0 a quello più alto B_{n-1} .

In questo caso il segnale digitale si presenta nella forma:

$$V_m = k \cdot \left(B_0 \cdot 2^0 + B_1 \cdot 2^1 + B_2 \cdot 2^2 + ...+ B_{n-1} \cdot 2^{n-1} \right)$$

Dove *k* è una costante di proporzionalità.

Si moltiplichi e si divida per 2^n si ottiene:

$$V_m = k \cdot \frac{2^n}{2^n} \cdot \left(B_0 \cdot 2^0 + B_1 \cdot 2^1 + B_2 \cdot 2^2 + ...+ B_{n-1} \cdot 2^{n-1} \right)$$

$$V_m = k \cdot 2^n \left(\frac{B_0}{2^n} + \frac{B_1}{2^{n-1}} + \frac{B_2}{2^{n-2}} + ...+ \frac{B_{n-1}}{2} \right)$$

I numeratori $B_0, B_1, B_2, ..., B_{n-1}$ rappresentano le *n* linee di uscita che possono assumere i due valori *0,1* secondo un codice *(per esempio quello binario)*. A seconda della combinazione dei valori, si ottengono i livelli nei quali viene suddiviso il segnale analogico.

Il prodotto $k \cdot 2^n$ rappresenta il fondo scala.

$$k \cdot 2^n = VFSR$$

per cui il valore associato al segnale digitale è composto dai valori:

$$V_m = VFSR \cdot \left(\frac{B_0}{2^n} + \frac{B_1}{2^{n-1}} + \frac{B_2}{2^{n-2}} + ...+ \frac{B_{n-1}}{2} \right)$$

Come valore da misurare si considera anche il livello 0, che si ottiene con tutte le uscite allo stato logico 0 $(B_0=0, B_1=0,...,B_{n-1}=0)$.

Il livello significativo più basso misurabile, diverso da 0, che rappresenta il quanto di riferimento unitario *q*, si ottiene quando solamente l'uscita B_0 è allo stato logico *1*, mentre tute le altre sono allo stato logico *0*.

In tal caso, per $B_0=1$ e gli altri bit allo stato 0, risulta:

$$q = VFSR \cdot \frac{1}{2^n}$$

Il valore massimo misurabile del segnale analogico si ottiene quando tutte le linee sono allo stato logico 1 *($B_0=1$, $B_1=1$,...,$B_{n-1}=1$)*. Sostituendo e ordinando secondo le potenze crescenti di 2, si ottiene:

$$V_{m\,max} = VFSR \cdot \left(\frac{1}{2} + \frac{1}{2^2} + \frac{1}{2^3} + \ldots + \frac{1}{2^n} \right)$$

L'espressione entro parentesi:

$$\left(\frac{1}{2} + \frac{1}{2^2} + \frac{1}{2^3} + \ldots + \frac{1}{2^n} \right) = S$$

è una serie geometrica di n termini di ragione $r = \frac{1}{2}$

Si ricorda che la somma *S* di n termini di una progressione geometrica, di ragione *r*, è data dalla espressione:

$$S = \frac{r^n - 1}{r - 1} \cdot a_1 \qquad \text{oppure} \qquad S = \frac{1 - r^n}{1 - r} \cdot a_1$$

Dove *r* è la ragione e a_1 è il primo termine della progressione.

Nel caso in esame risulta: il primo termine $a_1 = \frac{1}{2}$ e la ragione $r = \frac{1}{2}$. Sostituendo si ottiene:

$$S = \left(\frac{1}{2} + \frac{1}{2^2} + \frac{1}{2^3} + \ldots + \frac{1}{2^n} \right) = \frac{1 - \left(\frac{1}{2} \right)^n}{1 - \frac{1}{2}} \cdot \frac{1}{2} = \frac{1 - \frac{1}{2^n}}{\frac{1}{2}} \cdot \frac{1}{2} = 1 - \frac{1}{2^n}$$

$$S = \frac{2^n - 1}{2^n}$$

Per cui sostituendo, si ottiene:

$$V_{m\,max} = VFSR \cdot \frac{2^n - 1}{2^n}$$

Si constata che, in questo tipo di quantizzazione, il livello massimo raggiungibile dalla grandezza analogica misurabile, risulta inferiore al fondo scala di riferimento *VFSR*. L'avvicinamento a questo del valore massimo è tanto più prossimo quanto maggiore è il numero dei bit della parola *(e quindi del n° delle linee di uscita)*

Supponiamo che il segnale analogico, immesso nel sistema, venga trasformato in digitale con valori discreti e il valore di fondo scala sia VFSR=10 V

Per semplicità di trattazione, si voglia quantizzare con soli 3 bit detto segnale (3 uscite).

Il numero di combinazioni con 3 bit danno un n° di parole

$$2^3 = 8 \text{ parole}$$

Il valore del quanto fondamentale risulta, in questo caso:

$$q = \frac{VFSR}{2^n} \quad \text{essendo} \quad VFSR = 10V \quad \text{risulta} \quad q = \frac{10}{2^3}$$

$$q = \frac{10}{7} = 1,25\ V$$

Il minimo valore del segnale analogico misurabile risulta pari al quanto fondamentale q=1.25 V. Tutti gli altri valori quantizzati (livelli) della grandezza analogica si ottengono dalla espressione:

$$V_d = 10 \cdot \left(\frac{B_0}{2^3} + \frac{B_1}{2^2} + \frac{B_2}{2} \right)$$

dando a B_0, B_1, B_2 i valori *0,1*, secondo il codice binario.

Si ottengono 8 livelli compreso lo zero. Ad ogni livello si associa una parola

Nel codice binario puro la quantizzazione dà in uscita le seguenti parole:

tab.2.3.1

Livello	Ingresso	parola			Uscite		
		B_2	B_1	B_0	O_2	O_1	O_0
0,00	I_0	0	0	0	0	0	0
1,25	I_1	0	0	1	0	0	1
2,50	I_2	0	1	0	0	1	0
3,75	I_3	0	1	1	0	1	1
5,00	I_4	1	0	0	1	0	0
6,25	I_5	1	0	1	1	0	1
7,50	I_6	1	1	0	1	1	0
8,75	I_7	1	1	1	1	1	1

I valori riportati nella colonna dei livelli, si ottengono ponendo al posto di B_0, B_1, B_2 della espressione del segnale V_d i valori della corrispondente parola.

Così si ha:

$parola = 0\ 0\ 0 \qquad V_d = 10 \cdot (0 + 0 + 0) = 0$

$parola = 0\ 0\ 1 \qquad V_d = 10 \cdot \left(\frac{1}{2^3} + \frac{0}{2^2} + \frac{0}{2} \right) = \frac{10}{8} = 1,25$

$parola = 0\ 1\ 0 \qquad V_d = 10 \cdot \left(\frac{0}{2^3} + \frac{1}{2^2} + \frac{0}{2} \right) = \frac{10}{4} = 2,5$

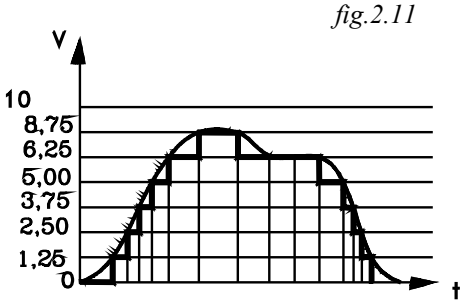
$parola = 0\ 1\ 1 \qquad V_d = 10 \cdot \left(\frac{1}{2^3} + \frac{1}{2^2} + \frac{0}{2} \right) = 3,75$

parola = 1 0 0 $V_d = 10 \cdot \left(\frac{0}{2^3} + \frac{0}{2^2} + \frac{1}{2} \right) = \frac{10}{2} = 5$

....

parola = 1 1 1 $V_{d\,max} = 10 \cdot \left(\frac{1}{2^3} + \frac{1}{2^2} + \frac{1}{2} \right) = 8{,}75$

fig.2.11



l segnale analogico, nel semplice esempio in esame, viene suddiviso in $2^3 - 1 = 7$ quanti *(8 compreso lo zero)*. Il fondo scala è $VFSR = 10\ V$. Il livello massimo è $V_{d\,max} = 8{,}75V$

I valori vengono trasmessi in uscita a gradini: quelli compresi tra un livello e il successivo vengono riferiti al primo

Si noti che l'espressione del valore misurato V_m nella disrcetizzazione del segnale analogico può essere formulata rispetto al quanto q. Infatti si effettui lo stesso denominatore entro la parentesi e si ottiene:

$$V_m = \frac{VFSR}{2^n} \cdot \left(B_0 \cdot 2^0 + B_1 \cdot 2^1 + B_2 \cdot 2^2 + \ldots + B_{n-1} \cdot 2^{n-1} \right)$$

$$V_m = q \cdot \left(B_0 \cdot 2^0 + B_1 \cdot 2^1 + B_2 \cdot 2^2 + \ldots + B_{n-1} \cdot 2^{n-1} \right)$$

2.3.2 Circuito logico di codifica

Riferiamoci all'esempio precedente, nel quale la quantizzazione è effettuata con *n=3 bit*, potendo ottenere, con la codifica in binario, $2^3=8$ parole diverse.

Ad ogni livello viene associata una parola; ciascuna corrisponde ad una combinazione degli stati *0,1* che possono assumere le *3* linee di uscita.

Si considerano i livelli come ingressi. Così:

- Al livello 0, si fa corrispondere l'ingresso I_0. Ad esso viene associata la parola 000, fornita dalle uscite: $O_0=0\ O_1=0\ O_2=0$
- Al livello del primo quanto *1,25* si fa corrispondere l'ingresso I_1. Ad esso viene associata la parola *001*, fornita dalle uscite: $O_0=1\ O_1=0\ O_2=0$
- Al livello del secondo quanto *2,50* si fa corrispondere l'ingresso I_2. Ad esso viene associata la parola 010, fornita dalle uscite: $O_0=0\ O_1=1\ O_2=0$
- E così via...

Dalla tabella della verità si nota che l'uscita O_0 deve essere attivata *(è allo stato logico 1)* per fornire le parole associate: all'ingresso I_1, Oppure all'ingresso I_3, Oppure all'ingresso I_3, Oppure all'ingresso I_5, Oppure all'ingresso I_7. Si ha:

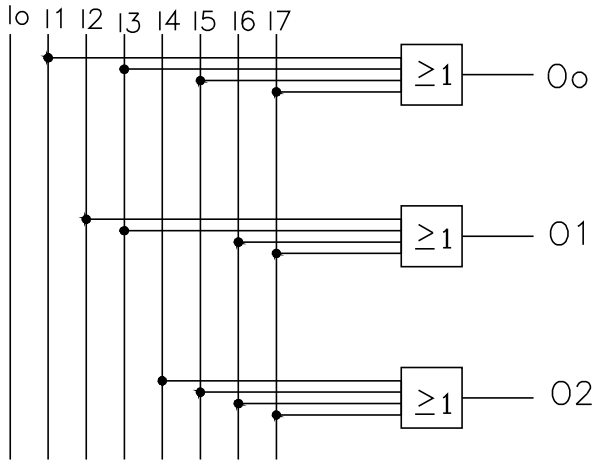
$$O_0 = I_1 + I_3 + I_5 + I_7$$

Alla stessa maniera si ha:

$$O_1 = I_2 + I_3 + I_6 + I_7$$
$$O_2 = I_4 + I_5 + I_6 + I_7$$

Dalle equazioni logiche delle linee di uscita si ricava il circuito logico.

fig.1.12



Così per esempio quando il valore analogico è *3,8* è attivato l'ingresso I_3 *(3,75)* e la parola di uscita è *0 1 1 (O_0 =1 O_1=1 O_2=0).*

2.3.3 Sensibilità o Range Dinamico

Il valore del quanto:

$$q = \frac{VFSR}{2^n}$$

rappresenta la minima variazione apprezzabile dal sistema di misura.

Tutta la scala di misura, dal minimo valore al massimo, è suddivisa in un numero n di livelli *(compreso lo 0)*. Ad ogni livello corrisponde una parola digitale *(in un certo codice).*

Il valore della grandezza analogica viene confrontata con la discretizzazione effettuata.

Qualunque tensione di ingresso compresa tra un livello ed il successivo è convertita nella stessa parola digitale corrispondente al primo.

Il valore del quanto determina la sensibilità della apparecchiatura.

Chiamiamo *Range Dinamico* il rapporto espresso in *decibel dB*:

$$Rang\ \ dinamico = 20 \cdot log_{10}\left(\frac{Tensione\ \ fondo\ \ scala}{quanto}\right) = 20 \cdot log_{10}\left(\frac{VFSR}{q}\right)$$

sostituendo a q la sua espressione si ha:

$$Rang\ \ dinamico = 20 \cdot log_{10}\left(\frac{VFSR}{\frac{VFSR}{2^n}}\right) = 20 \cdot log_{10}\left(2^n\right)$$

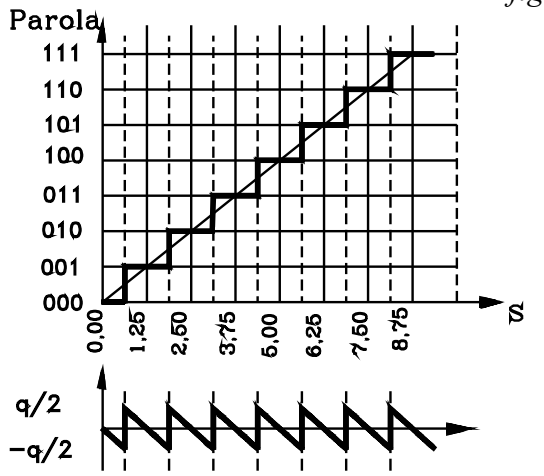
$$Rang\ \ dinamico = 20n \cdot \log_{10} 2$$

Il Range dinamico risulta proporzionale al n° di bit che costituisce la parola e quindi al numero delle linee di uscita

2.3.4 Rumore di quantizzazione

Nella quantizzazione del segnale analogico i valori compresi tra due livelli vengono codificati con la stessa parola: ciò costituisce un errore detto *rumore di quantizzazione.*

fig.1.13



Il segnale *S* può considerarsi formato dal segnale analogico più l'errore di quantizzazione *(vedi fig.2.13)*

L'errore varia tra *-q/2÷+q/2*

È evidente che all'aumentare di *n* diminuisce *q* e quindi diminuisce l'errore di quantizzazione.

2.3.5 Precisione

La precisione è data dal rapporto % tra l'errore e il valore della misura.

$$e\% = \frac{Errore}{Misura} \cdot 100$$

L'errore della più piccola quantità apprezzabile è *q/2* sostituendo si ha:

$$e\% = \frac{\frac{q}{2}}{Misura} = \frac{1}{2 \cdot Misura} \cdot q \qquad \text{ma } q = \frac{VFSR}{2^n} \text{ sostituendo si ha:}$$

$$e\% = \frac{1}{2 \cdot Misura} \cdot \frac{VFSR}{2^n} \qquad e\% = \frac{VFSR}{2^{n+1} \cdot Misura}$$

Se il segnale di ingresso ha un valore max troppo piccolo rispetto al fondo scala si ha un errore più elevato, in quanto il segnale viene poco campionato

Per diminuire l'errore occorre o aumentare la risoluzione (e quindi n), oppure amplificare il segnale prima della conversione A/D

2.3.6 Durata della misura

2.3.6.1 Tempo di conversione:

È il tempo che intercorre tra l'istante di inizio conversione *A/D* e l'istante nel quale si presenta in uscita la *parola* (1÷200 μs)

2.3.6.2 Tempo di apertura

È il tempo impiegato per il campionamento, durante il quale viene misurato il segnale analogico, portato in uscita e codificato in una parola

È evidente che durante il tempo di apertura il segnale analogico continua a modificarsi.

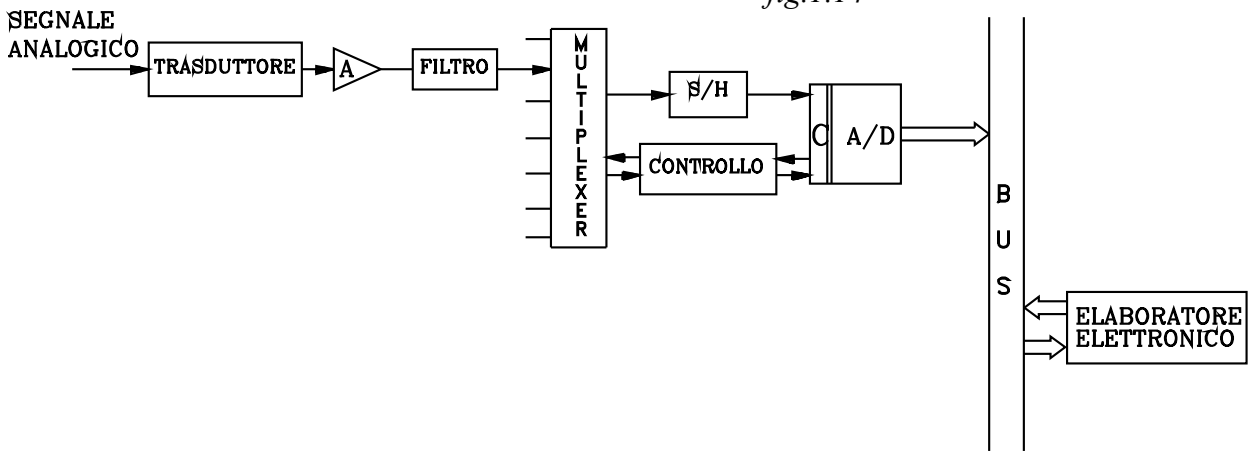
Affinché non vi siano incertezze nella misura occorre che durante il tempo di apertura il segnale analogico non vari più di $\pm q/2$

Se ciò avviene allora il numero binario risulta corretto, altrimenti vi è incertezza.

Diminuendo il tempo di apertura deve corrispondentemente diminuire il tempo di conversione, però questo non deve essere tale da far variare il segnale analogico più di $\pm q/2$ nel tempo di apertura.

Riassumiamo la trasformazione A/D riferendoci allo schema di fig.

fig.1.14



Il segnale analogico, proveniente da un processo, generalmente di natura fisica diversa dall'elettrica, viene trasformato in questa attraverso un trasduttore .
Il segnale amplificato *A*, e opportunamente filtrato , viene portato al Multiplexer .

Il *Multiplexer* ha la funzione di un commutatore, nel quale convergono più segnali analogici in ingresso, che porta in uscita, secondo l'ordine stabilito nel programma, quel segnale che per un dato tempo deve essere convogliato verso l'elaboratore.

I canali di ingresso vengono interrogati sequenzialmente, ma un determinato canale di ingresso verrà abilitato a convogliare in uscita il segnale verso l'elaboratore quando questi lo richiede.

Il canale rimane abilitato per il tempo necessario per la lettura e la sua conversione *A/D*. Quando il dato richiesto è pronto viene portato in uscita e disabilitato il canale.

Il computer elaborerà poi il dato digitale lo confronterà con quello programmato di riferimento, se il valore letto è diverso, l'elaboratore emetterà un segnale in uscita di correzione.

Nella trasformazione del segnale da *A/D* occorre che esso non vari troppo bruscamente durante il tempo necessario per la conversione, altrimenti si ha un errore di lettura.

Se la frequenza del segnale da convertire è elevata (variazioni brusche) occorre diminuire il tempo di apertura (aumentare la frequenza di campionamento); ma contemporaneamente occorre diminuire anche il tempo di conversione (aumentare la frequenza di conversione). In tal caso occorre utilizzare un convertitore con frequenza di conversione elevata e quindi costosa.

Per poter impiegare un convertitore con frequenza di conversione meno elevata *(tempo di conversione più grande)*, e quindi meno costoso, occorre impiegare un particolare circuito detto Samlpe & Hold "*S/H*".

Con li circuito *S/H* il segnale analogico viene campionato in tempi molto piccoli *(segnale di apertura di elevata frequenza)* e il valore campionato si mantiene stabile per tutto il tempo di conversione nella corrispondente parola emessa nell'uscita dell'*A/D.*

Tra il multiplexer e il convertitore *A/D* vi è una linea di controllo per il comando e l'interrogazione di inizio e fine conversione.

Quando l'elaboratore, secondo programma, è pronto ad accettare un determinato ingresso viene emesso un bit che lo seleziona e dà inizio alla conversione.
Viene controllato il termine di fine conversione: alla conferma di *fine,* la parola viene letta sulle linee di uscita dell'*A/D.*

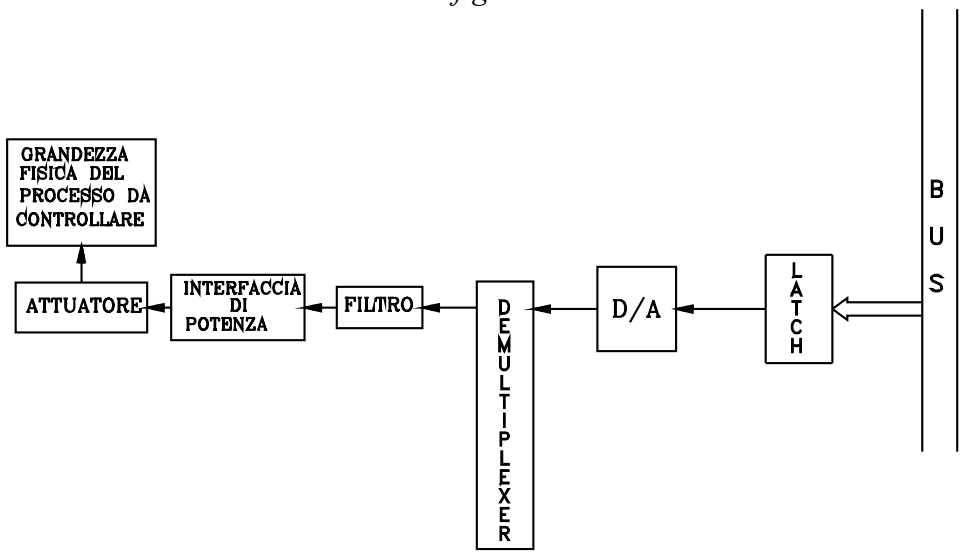
2.4 Conversione del segnale da Digitale in Analogico "D/A"

La conversione D/A è l'operazione inversa dell'*A/D*
Il microprocessore, secondo programma, emette, nel bus dei dati, la parola digitale che va all'ingresso del convertitore.

Il bus dati si interfaccia con il convertitore *A/D* attraverso un *latch*, di memorizzazione temporanea, che mantiene stabile la parola digitale fino a che non avviene una successiva variazione nel bus.

Nella figura viene riportato lo schema a blocchi, già precedentemente rappresentata, riguardante la trasformazione del segnale digitale emesso dall'elaboratore, in un comando di un attuatore che lo trasforma in una grandezza fisica *(temperatura , spostamento, velocità...).*

fig.2.15



La parola che viene immessa nel convertitore *D/A*, corrisponde ai diversi stati 0,1 dei fili del bus. Questi comandano l'apertura o la chiusura di interruttori elettronici che collegano

diversamente una rete di resistenze, ottenendo all'uscita una tensione diversa a seconda della parola

La struttura è costituita da una rete di resistenze di precisione, che vengono inserite da una serie di switch in serie, tanti quanti sono i bit della parola.

La parola di ingresso apre o chiude gli switch e varia la tensione in uscita.

Così, per esempio, con un convertitore a 3 bit il più piccolo valore di tensione risulta:

$$q = \frac{VFSR}{2^n} = \frac{1}{2^n} \cdot VFSR$$

Se VFRS=10 V $q=1,25$

Indichiamo con $B_0\ B_1\ B_2$ le linee del bus le cui combinazioni del loro stato logico 0,1 danno le $2^3=8$ parole, ad ognuna delle quali viene associato un livello di tensione multiplo del quanto di riferimento e costituente un'uscita Qi.

La tabella della verità si ottiene dalla *tab.2.3.1* scambiando gli ingressi con le uscite. Gli ingressi sono costituiti dalle linee $B_0\ B_1\ B_2$ ... che con la loro combinazione degli stati logici *0,1* determinano la parola corrispondente al segnale digitale; l'uscita è il valore che si ottiene dalla rete di resistenze, coincidente con quello dato dalla espressione:

$$V_m = VFSR \cdot \left(\frac{B_0}{2^n} + \frac{B_1}{2^{n-1}} + \frac{B_2}{2^{n-2}} + \ldots + \frac{B_{n-1}}{2^2} \right)$$

In pratica, nel convertitore *D/A*, nei successivi istanti di tempo, in ingresso entrano le parole, corrispondenti alle combinazioni degli stati logici delle linee $B_0\ B_1\ B_2$...ed escono, associati ad essi, valori di tensione V_m.

Ingressi			Parola	livello	Uscita
B_2	B_1	B_0			
0	0	0	000	0,00	Q_0
0	0	1	001	1,25	Q_1
0	1	0	010	2,50	Q_2
0	1	1	011	3,75	Q_3
1	0	0	100	5,00	Q_4
1	0	1	101	6,25	Q_5
1	1	0	110	7,50	Q_6
1	1	1	111	8,75	Q_7

Con un bus a *8* linee di dati si hanno $2^8=256$ combinazioni possibili. Considerando come primo livello lo *0*, corrispondente a tutte le linee nello stato logico *0*, rimangono da combinare *256-1=255* stati possibili.

Se la tensione di fondo scala è *10 V* il valore minimo corrispondente alla variazione di un bit risulta:

$$q = \frac{10}{256} = 0{,}0390625$$

Così se il segnale digitale da convertire è *00010010*, con $I_4=1$ $I_1=1$ e tutti gli altri canali nello stato *0*, la parola da convertire avrà peso:

$$2^4 + 2^1 = 16 + 2 = 18$$

La tensione che si avrà in uscita sarà:

$$V_m = 18 \cdot q \qquad V_m = 18 \cdot 0{,}0390625 = 0{,}703125V$$

2.4.1 Circuito decodificatore

Il circuito decodificatore deve effettuare l'operazione inversa della codifica.

Il codificatore trasforma un livello (valore multiplo del quanto q) in una parola, come combinazione degli stati *0,1* sulle n linee del bus di uscita.

Il decodificatore , al contrario, tramuta una parola, costituita dalla combinazione degli stati *0,1* delle n linee del bus di ingresso in un corrispondente livello *(valore)*.

Combinando gli stati 0,1 delle n linee del bus di ingresso si possono ottenere fino a $N=2^n$ valori in uscita.

Riferiamoci all'esempio fatto di codifica con soli *3 bit.*

Il fondo scala è stato diviso in $2^3=8$ *livelli*, compreso lo zero e ad ogni livello è stato associata una parola, effettuata con i *3 bit*, corrispondenti alla combinazione degli stati *0,1* delle 3 linee del bus di uscita.

Nel decodificatore occorre effettuare l'operazione inversa.

- In ingresso del decodificatore vi sarà un bus di *3 linee*, costituenti i *3 bi*t, i cui stati 0,1 si possono combinare in $2^3=8$ modi diversi.
- Ogni combinazione a *3 bit* degli stati *0,1* delle *3* linee di ingresso costituisce una parola di ingresso.
- Ad ogni parola viene associato un valore, che rappresenta un livello di quantizzazione da zero al valore massimo
- Gli 8 livelli, associati alle 8 parole, ripristinano in uscita tutti i valori quantizzati da zero al valore massimo.

Dalla tabella della verità della pag. precedente si ha:

Uscita *Q0* Si ottiene dalla combinazione dei *3 bi*t *000* degli stati delle tre linee del bus di ingresso *B0=0 B1=0 B2=0*. Q0 corrisponde a $V_m=0{,}00\ V$

Uscita *Q1* Si ottiene dalla combinazione 001 degli stati delle 3 linee del bus di ingresso. Q_1 corrisponde al valore V_m=1,25 V

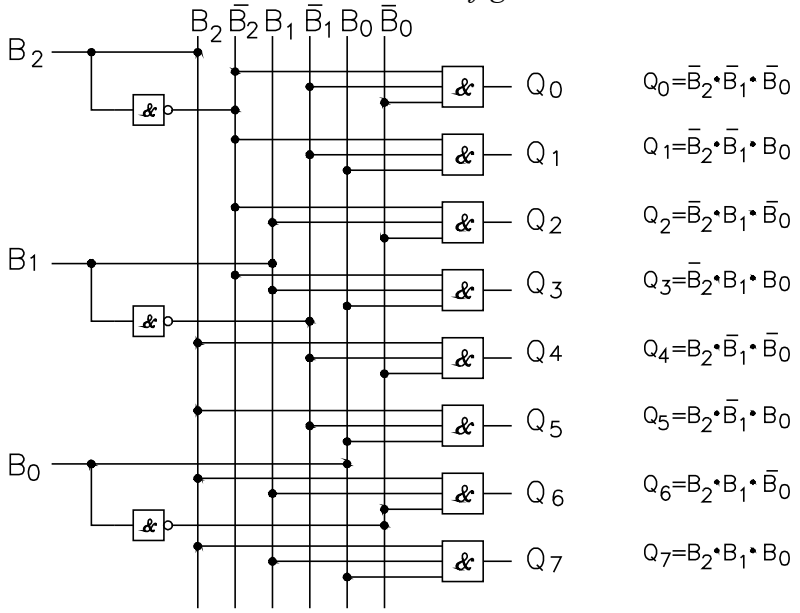
Così per le altre uscite...

Si ha:

$$Q_0 = \overline{B_2} \cdot \overline{B_1} \cdot \overline{B_0}$$
$$Q_1 = \overline{B_2} \cdot \overline{B_1} \cdot B_0$$
$$Q_2 = \overline{B_2} \cdot B_1 \cdot \overline{B_0}$$
$$Q_3 = \overline{B_2} \cdot B_1 \cdot B_0$$
$$Q_4 = B_2 \cdot \overline{B_1} \cdot \overline{B_0}$$
$$Q_5 = B_2 \cdot \overline{B_1} \cdot B_0$$
$$Q_6 = B_2 \cdot B_1 \cdot \overline{B_0}$$
$$Q_7 = B_2 \cdot B_1 \cdot B_0$$

Dalle equazioni si ottiene il circuito logico di figura *fig.2.16*:

fig2.16



I valori *Q0,Q1,Q2,Q3,Q4,Q5,Q6,Q7* vengono inviati uno alla volta, a seconda della parola, all'unica uscita, ripristinando il segnale analogico. Questo risulta segmentato del quanto q.

La scheda di uscita di una apparecchiatura presenta un solo convertitore *D\A*, ma, a volte, servono più uscite, in ognuna delle quali si deve inviare un differente segnale analogico per portarlo in un attuatore diverso.

Occorre in tal caso un commutatore, detto Demultiplexer, avente la funzione inversa del Multiplexer.

2.5 Demultiplexer

Il Demultiplexer è un commutatore nel quale converge, secondo un determinato programma, di volta in volta, un segnale analogico diverso, e questo viene smistato in una uscita diversa secondo un ciclo prefissato.

Il demultiplexer è costituito da un solo ingresso, che indichiamo con *Q* e più uscite che indichiamo con *Ui.*

Le uscite *Ui* vengono abilitate da *n* linee di controllo.

Le n linee di controllo possono assumere i due valori *0,1.*

Con la combinazione dei due stati *0,1* delle n linee di controllo si possono abilitare:

$$N=2^n$$

linee di uscita dal Demultiplexer.

Per semplicità di trattazione consideriamo come esempio un Demultiplexer con due linee di controllo *S0 S1*

Le combinazioni dei due stati delle due linee di controllo vengono poste in *AND* con l'unico ingresso *Q* per abilitare, di volta in volta, una determinata uscita.

Una possibile tabella della verità è la seguente:

Stati delle linee di controllo		Combinazioni dell'ingresso *Q* con gli stati delle linee di controllo	Uscite U_i
S_1	S_0	$Q \cdot S_1 \cdot S_0$	
0	0	$Q \cdot \overline{S_1} \cdot \overline{S_0}$	U_0
0	1	$Q \cdot \overline{S_1} \cdot S_0$	U_1
1	0	$Q \cdot S_1 \cdot \overline{S_0}$	U_2
1	1	$Q \cdot S_1 \cdot S_0$	U_3

Le equazioni logiche delle uscite sono

$U_0 = Q \cdot \overline{S_1} \cdot \overline{S_0}$
$U_1 = Q \cdot \overline{S_1} \cdot S_1$
$U_2 = Q \cdot S_1 \cdot \overline{S_0}$
$U_3 = Q \cdot S_1 \cdot S_0$

Da queste si ricava il circuito logico di *fig.2.17*

fig.2.17

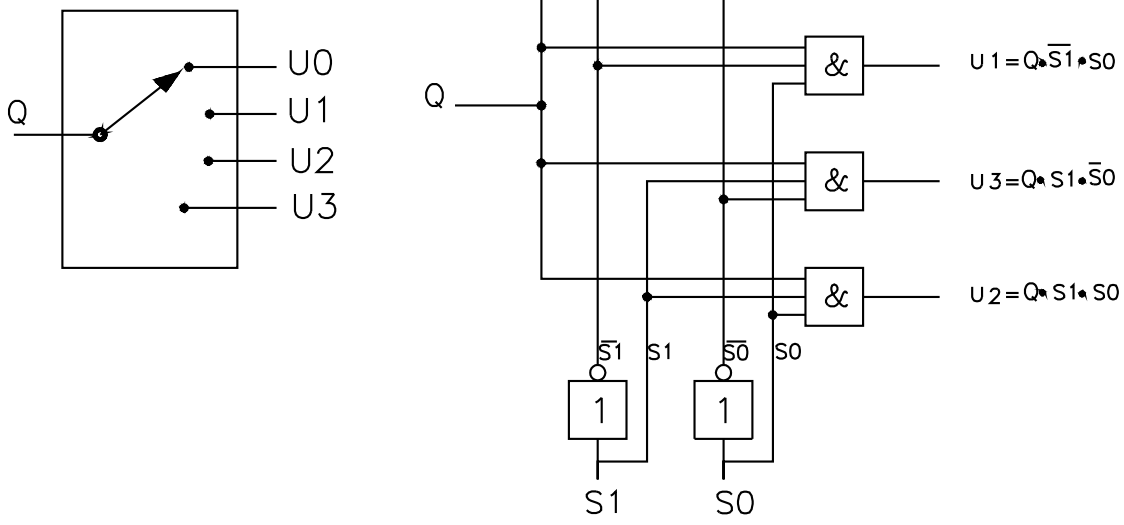
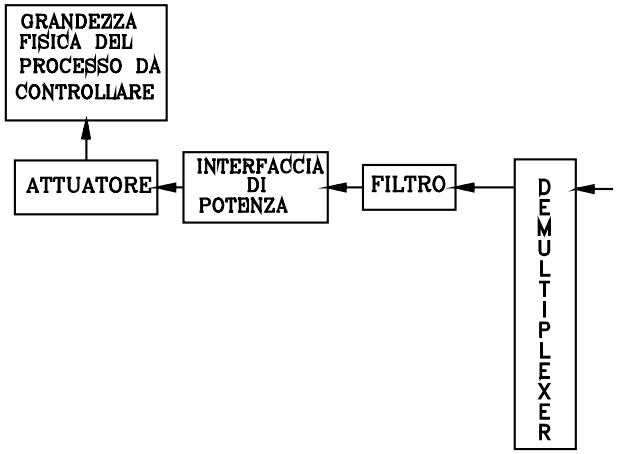


fig.2.18



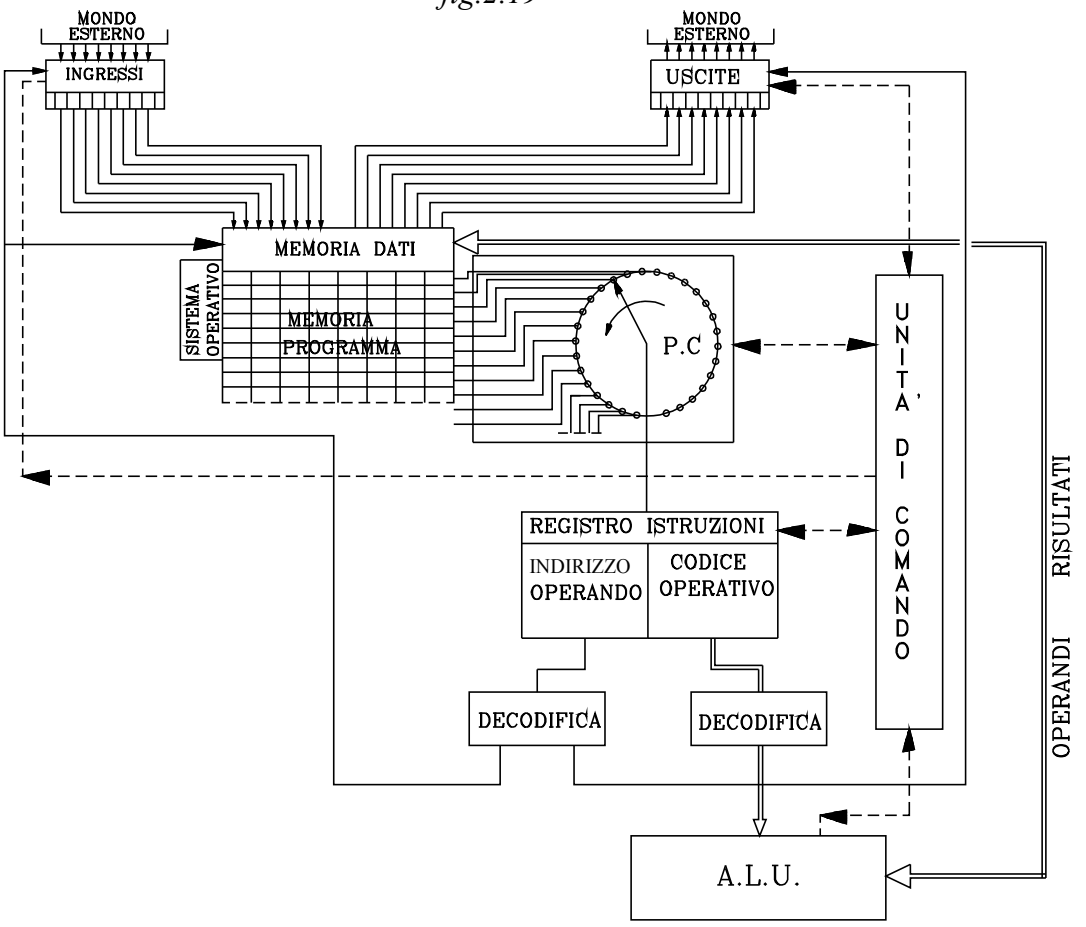
Dal demultiplexer i singoli segnali di uscita vanno, poi, indirizzati ai rispettivi attuatori di comando degli apparati, dopo aver subito un opportuno filtraggio ed essere stati amplificati.

2.6 ARCHITETTURA DEL PLC

Nello schema a blocchi di figura viene rappresentata la struttura dell'unità centrale di un PLC nelle sue parti essenziali.

L'architettura ricalca quella già analizzata nel primo volume, nelle note di spiegazione di un calcolatore.

fig.2.19



Si dà qui una breve descrizione del funzionamento dei blocchi indicati nello schema e del loro ruolo nella elaborazione e trasmissione dei segnali. La spiegazione più particolareggiata di essi verrà ripresa in seguito.

2.6.1 Moduli di ingresso e di uscita

Il PLC si collega con il mondo esterno attraverso i moduli di ingresso *"Input I"* e di uscita *"Output O"*

Nei moduli di ingresso "I" vengono introdotti i segnali provenienti dal processo controllato: inviati cioè dai sensori, finecorsa ecc. che indicano lo stato attuale del processo, in base al quale verranno elaborati i segnali di uscita, che determineranno l'evoluzione verso lo stato futuro del sistema.

Dai moduli di uscita "O" escono i segnali elaborati dal PLC, secondo un programma, che vengono inviati agli attuatori per modificare lo stato del processo e farlo evolvere verso quello desiderato.

2.6.2 Dati -Operandi - Operazioni logiche

I segnali di ingresso rappresentano alcuni degli operandi, dei *dati* cioè sui quali verranno effettuate delle operazioni logiche, secondo il programma caricato nella memoria del PLC.

Ricordiamo che:

1. All'ingresso del *PLC* arrivano i segnali provenienti dal processo (Mondo esterno) e indicanti il suo stato.

2. Sui segnali di ingresso vengono effettuate le operazioni logiche contenute nel programma memorizzato nel *PLC*.
   Per cui i segnali di ingresso rappresentano dei dati *"Operandi"* sui quali vengono eseguite le operazioni logiche *AND, OR, NOT...*

3. A seconda del risultato delle operazioni logiche eseguite sui dati, si ottiene un specifico segnale di uscita. Questo è un dato che attiva in un particolare modo (parola) l'uscita del PLC.

4. Il segnale di uscita dal PLC viene inviato all'attuatore, per modificare lo stato del processo e farlo evolvere verso quello voluto.

5. Occorre puntualizzare che i *dati di uscita* possono essere interrogati e si possono effettuare su di essi operazioni logiche che contribuiscono a determinare il valore delle uscite stesse.

2.6.3 Memorie

I dati di ingresso e di uscita vengono memorizzati in una memoria di tipo RAM. Qui stazionano, e possono, successivamente, essere interrogati, prelevati per effettuare su di essi le operazioni logiche contenute nel programma memorizzato nel PLC.

Nel PLC sono previste, a seconda dei casi, diversi tipi di memoria, che verranno descritte in seguito.

Riguardo all'utilizzo della memoria, questa si può suddividere in due sezioni principali:

Memoria di Sistema In questa sezione della memoria viene caricato il Sistema Operativo del PLC. È l'interprete dei comandi che si incarica di tradurre le

istruzioni in linguaggio macchina. Non è accessibile all'utente, viene caricata all'avvio del PLC

Memoria applicativa Questa è accessibile all'utente e contiene i dati e il programma

Memoria Dati In questa memoria vengono scritti i dati corrispondenti agli Ingressi , Uscite ai Temporizzatori e ai Flag

Memoria Programma In questa parte della memoria viene contenuto il programma, consistente nelle operazioni logiche da effettuare sui dati per ottenere i segnali di uscita.

Le operazioni sui dati costituiscono le istruzioni, che si susseguono sequenzialmente una dietro l'altra.

2.6.4 Program Counter

Le istruzioni da eseguire sono contenute nelle celle della memoria di programma e da questa debbono essere prelevate una alla volta , sequenzialmente, da un apparato detto *Program Counter* "P.C." *(Contatore del Programma) fig.2.19.*

Ciascuna istruzione costituisce una parola, contenuta in una cella della *memoria di programma,* contraddistinta da un indirizzo.

Gli indirizzi, e quindi le celle di memoria, sono ordinate secondo i numeri crescenti naturali.

Il Program Counter legge sequenzialmente le celle della memoria di programma, partendo dall'indirizzo di ordine più basso (00000000) fino all'ultimo di ordine massimo, secondo una cadenza data da un orologio interno *clock.* Letta l'ultima cella con indirizzo più alto il Program Counter torna a leggere la prima di livello più basso.

Si hanno così cicli successivi di lettura delle istruzioni del programma.

L'istruzione letta dal Program Counter va memorizzata temporaneamente nel *Registro Istruzioni*

2.6.4.1 Registro Istruzioni - Istruzioni - A.L.U

Una istruzione, come noto (1° Volume), è costituita dal *Codice Operativo* e dagli *Operandi*.

Il *Codice operativo* costituisce l'operazione logica che deve essere eseguita sugli *Operandi* e questi sono i *dati*, forniti dagli *Ingressi*, *Uscite*, *Flag, Temporizzatori, Contatori*.

L'istruzione, nelle sue due parti, viene registrata, temporaneamente nel *Registro delle Istruzioni*, fino ché non viene eseguita e non serve più per il proseguo del programma.

L'operazione logica, contenuta nella istruzione, deve essere eseguita nell'A.L.U. *(Unità Aritmetica Logica)*. Per ottenere ciò occorre prelevare i dati *(gli Operandi: Ingressi Uscite, Flag...)*, dalla memoria dati e inviarli nell'A.L.U, dove perviene, decodificato, il codice operativo della istruzione, contenente l'operazione logica da eseguire su quei dati.

Per rintracciare i dati viene associato ad ognuno di essi un indirizzo decodificato dalla istruzione che perviene nel Registro delle Istruzioni.

Eseguite nell'A.L.U le operazioni logiche contenute nell'istruzione, si ottiene il risultato, costituente il dato di uscita, che va nella memoria dati. Da questa può essere prelevato e inviato

in uscita verso gli attuatori, oppure può essere interrogato per costituire un operando di una istruzione di programma.

La gestione di tutte le operazioni viene effettuata dalla unità detta di *Comando o Controllo* che emette dei segnali su un bus detto *bus di controllo (indicato con linea tratteggiata)*

I dati vengono trasmessi sul *bus dati (doppia linea)* e gli indirizzi su *bus indirizzi (linea continua)*

2.6.5 MODULI DI INGRESSO

Il segnale di ingresso può essere di tipo analogico, digitale, ON/OFF. Occorre in ogni caso tradurlo nel sistema binario nel quale può assumere solamente due valori 0,1 corrispondenti a due livelli di tensione o corrente: livello basso - alto.

Le schede di ingresso rappresentano l'interfaccia tra la logica interna e i segnali esterni. Quindi debbono operare un adattamento dei livelli e delle caratteristiche dei segnali esterni alla logica interna della microelettronica, operante tra *0÷5* Volt in continua.

I segnali che provengono dall'esterno operano, invece con livelli di tensione più alti: *24, 48*, *110, 220* Volt

Le schede di ingresso presentano un n° di ingressi multipli di due : *8, 16, 32*.

Il *PLC* può avere più moduli di ingresso contrassegnati da un simbolo e un numero. Il principale è il modulo *0*.

Così i moduli di ingresso in alcuni programmi di PLC sono contrassegnati dalla lettera *I (Input)*; in altri, di lingua tedesca dalla lettera *E*. Il numero indica quale modulo di ingresso si sta adoperando.

Così l'ingresso I_2 indica il modulo di ingresso *2*.

Ogni modulo, a sua volta, è composto da un numero di ingressi multipli della potenza di due: usuale 8 ingressi. Detti ingressi sono contrassegnati da numeri progressivi partendo dallo 0. Il modulo a 8 ingressi ha questi contrassegnati dai numeri : *0,1,2,3,4,5,6,7.*

Per contraddistinguere lo specifico ingresso che si sta utilizzando di un modulo di ingresso occorre aggiungere alla precedente indicazione del modulo un secondo numero.

Così, ad esempio, per indicare l'ingresso 4 del modulo 2 si scrive:

I2.4

6.6.6.1 Optoisolatori

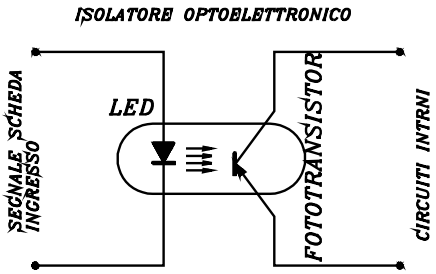
Gli ingressi debbono essere isolati dalla logica interna al *PLC*, in quanto i livelli energetici tra gli elementi esterni e quelli interni al controllo, sono diversi.

Vi deve essere una separazione dei potenziali. Eventuali sbalzi e sovraccarichi non debbono danneggiare i circuiti integrati interni al *PLC*.

Non vi deve essere una continuità elettrica tra ingresso e gli apparati interni del PLC e tra questi e l'uscita.

fig.2.20

La tecnica maggiormente utilizzata per separare gli ingressi dai circuiti interni è quella dei sistemi optoelettronici



Ciascun segnale di ingresso viene inviato ad un *LED* (Light Emitting Diod). Questo è un diodo che, alimentato, emette luce.

Il fascio luminoso, emesso dal *LED*, va ad investire un *FOTOTRANSISTOR* portandolo in conduzione.

Il fascio di luce rappresenta il segnale di gate che porta il transistor dallo stato di interdizione a quello di conduzione.

Si ottiene così una netta separazione elettrica tra la scheda di ingresso e i circuiti interni.

L'isolamento serve anche per evitare false informazioni dovute a disturbi indotte dal mondo esterno.

Per evitare disturbi, si tiene conto della natura impulsiva del disturbo stesso. Si ritiene che, normalmente, il disturbo è di brevissima durata; mentre il segnale, anche se impulsivo, ha una durata maggiore. Si introducono così degli elementi ritardatori: si ritiene che un segnale sia valido se si mantiene per un tempo non inferiore ad un limite prefissato; se è inferiore si ritiene dovuto ad un disturbo e, quindi, non viene considerato.

2.6.6 COMPOSIZIONE DI UNA ISTRUZIONE

Come si è detto, nel PLC vengono elaborati, secondo un programma, i segnali acquisiti in ingresso indicanti lo stato attuale del processo, e si ottengono in uscita i segnali inviati agli attuatori che lo fanno evolvere verso lo stato voluto.

L'elaborazione dei segnali avviene secondo le *istruzioni* contenute nel programma, che vengono attuate sequenzialmente, una alla volta, rispettando l'ordine degli *indirizzi* associati alle istruzioni stesse: partendo dall'indirizzo di ordine inferiore 000000 fino a quello di ordine massimo.

Una istruzione è composta da un insieme di bit 0,1 costituente una parola, memorizzata in una cella della *memoria di programma*.

Ad ogni cella di memoria e, quindi, ad ogni parola è associato un indirizzo.

La parola costituente l'istruzione è composta, come si è detto da due parti fondamentali

Codice Operativo È la parte dell'istruzione che contiene l'operazione logica *AND, OR NOT.*. che deve essere operata sugli operandi: ciò che deve eseguire la *CPU*.

Operando È la parte dell'istruzione che contiene l'informazione sul dato sul quale deve essere effettuata l'operazione indicata nel Codice Operativo: *Ingresso, Uscita, Flag...*

L'operando a sua volta è contraddistinto da due elementi fondamentali:

Tipo Contraddistingue il tipo di operando.

Esso indica se si tratta:

Di un ingresso: Simbolo *I*
Di una uscita Simbolo *O*
Ecc.

Elemento Vi sono più elementi dello stesso tipo. Così gli ingressi sono composti da più moduli e ciascuno di essi è costituito da un certo numero di elementi.

L'elemento specifico può essere indicato da due numeri separati da un punto, che rappresentano, rispettivamente. il modulo e l'ingresso utilizzato. Esempio

I2.4

$$Istruzione \begin{cases} Codice\ Operativo \\ Operando \begin{cases} Tipo \\ Elemento \end{cases} \end{cases}$$

Si consideri così l'istruzione:

AND I2.4

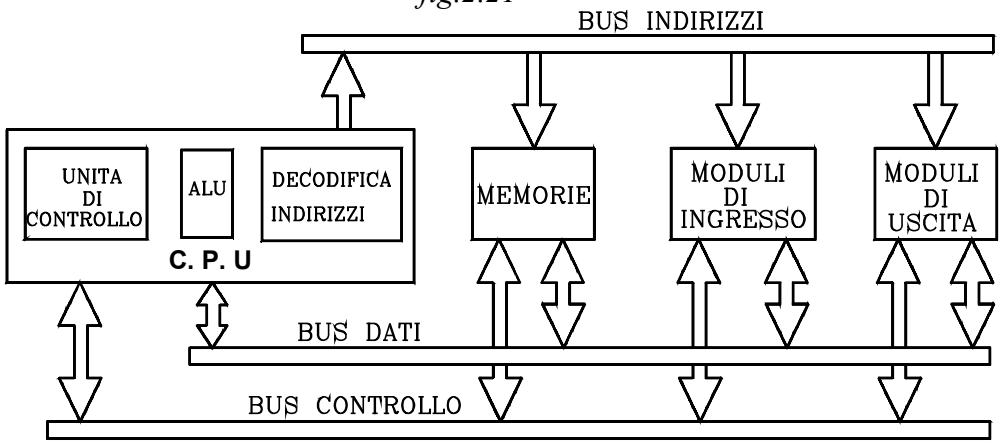
Essa è composta dalle seguenti parti

$$Istruzione\ \ AND\ \ I2.4 \begin{cases} Codice\ \ Operativo: & Operazione\ \ AND \\ Operando \begin{cases} Tipo: & Ingresso\ \ I \\ Elemento & Modulo\ 2\ -\ Ingresso\ 4 \end{cases} \end{cases}$$

2.6.7 BUS

Come già è stato spiegato nel 1° Volume i segnali si trasmettono nelle varie parti di un computer su dei canali fisici detti *BUS*.

fig.2.21



Per la gestione del sistema e trasmettere le istruzioni occorrono tre tipi di segnali che si trasmettono su *BUS* diversi.

Bus Indirizzi È unidirezionale : dal *CPU* alle altre unità. É costituita da *8* o *16* fili; può in questo caso trasportare in parallelo *16* bit, uno per filo.

Bus Dati È bidirezionale; in esso transitano dati e istruzioni nei due sensi tra *CPU* e le altre unità. Il numero di bit che in esso possono transitare definiscono la grandezza della parola del sistema.

Bus Controllo È bidirezionale; gestisce il controllo tra le varie parti del sistema. Ha la capacità di 1 bit.

2.6.8 MEMORIE DI PROGRAMMA

Come si è già accennato nella descrizione dello schema a blocchi di un *PLC*, la memoria totale si può suddividere in una Sezione, contenente il Sistema Operativo, non accessibile all'utente, e in un'altra, indirizzata alle applicazioni da effettuare con il *PLC*.

A seconda del tipo di lavoro che dovrà assolvere il *PLC* vengono scelti e impiegati differenti tipi di memoria.

Come noto le memorie si possono dividere in

Memorie volatili Sono memorie nelle quali le informazioni rimangono memorizzate fino a che sono alimentate: richiedono l'applicazione di una potenza elettrica per conservare la memorizzazione. Tolta questa perdono il loro contenuto

Memorie non volatili Sono memorie nelle quali le informazioni rimangono memorizzate anche quando viene tolta l'alimentazione.

2.6.8.1 Memoria RAM

La RAM *(Ramdom Acces. Memory)* è una memoria a semiconduttore volatile, sulla quale si possono scrivere le informazioni e leggere quelle in essa contenute.

È usualmente usata nei PLC, per contenere, non solamente i dati intermedi, ma anche il programma.

Per impedire che, tolta l'alimentazione, la memoria RAM perda il suo contenuto, questa viene posta in tampone con una batteria che continua ad alimentarla quando viene spento l'impianto.

La RAM per potersi mantenere attiva consuma poca energia, e risulta sufficiente quella fornita dalla batteria per mantenersi in attivazione, durante i periodi di disattivazione dell'impianto. Occorre ben inteso che questi non siano troppo lunghi, in modo da permettere che la batteria si ricarichi, durante il periodo di lavoro.

Per una memorizzazione stabile del programma si può affiancare alla *RAM* una memoria di massa ove memorizzarlo. Il programma poi dalla memoria di massa verrà caricato nella RAM per poter essere eseguito.

Una dissipazione di energia minima a riposo presentano le memorie RAM costruite con famiglia logica *CMOS (Complementary MOS)* , realizzata con coppie di *MOSFET* complementari.

A riposo i due *MOSFET* risultano interdetti e quindi, teoricamente non assorbono potenza . In realtà esiste una piccola corrente di fuga che determina una piccola dissipazione di energia che dovrà essere fornita dalla batteria in tampone.

Occorre ricordare che la memoria *RAM* ha un *tempo di accesso* uguale per qualsiasi locazione, indipendentemente dalla posizione fisica: qualunque sia l'indirizzo della parola da scrivere o leggere. Per questo si dicono ad accesso casuale

L'utilizzo della memoria *RAM* in tampone con una batteria (al litio) come memoria di programma rappresenta il caso più adoperato nei PLC, che sono destinati a funzionare con

programmi diversi o che debbono subire successive modifiche nel corso del funzionamento dell'impianto da essi comandato

2.6.8.2 Memoria ROM

La memoria *ROM (Read Only Memory)* sono di sola lettura: si può leggere il contenuto ma non modificarlo. Sono anche esse a semiconduttore ad accesso casuale come la *RAM*.

Vengono programmate nel momento della costruzione del chip, e le informazioni inserite rimangono permanentemente memorizzate. Sono quindi memorie non volatili.

La memoria ROM è utilizzata come memoria di programma solamente nel caso che debba servire per un sistema con programma fisso, nel quale non sono previste modifiche in tutta la vita dell'impianto e inoltre sono previsti molti *PLC* dello stesso tipo.

In tal modo il costo viene ridotto.

2.6.8.3 Memoria PROM

È, come la ROM, una memoria di sola lettura. Essa però non viene programmata al momento della costruzione del chip ma successivamente dall'utente.

La *PROM*, una volta programmata, non può essere più modificata. È adatta a contenere un programma fisso, non modificabile nel tempo e che soddisfi le esigenze dell'utente.

La programmazione viene effettuata con i *PROM PROGRAMMER* che si interfacciano ai *PLC*.

Vi sono delle memorie non volatili sulle quali si può scrivere un programma e successivamente modificare, in alcune l'intero contenuto, in altre anche una sola parte di esso. Sono quindi soprattutto memorie di lettura ma anche di scrittura. Questa intesa come modifica di un programma precedentemente scritto e memorizzato.

2.6.8.4 Memoria EPROM

La memoria *EPROM (Eraseble Programmeble Read Only Memory)* è principalmente di lettura. Il suo contenuto però può essere cancellato completamente e riscritto in un nuovo programma.

La nuova programmazione può avvenire solamente dopo aver cancellato completamente la memoria.

La cancellazione avviene per esposizione degli elementi a semiconduttore del chip ai raggi ultravioletti. Questi, in una esposizione alla sorgente, possono irradiarsi nell'interno del chip attraverso una apposita finestra trasparente, praticata su di esso.

In una memoria effettuata con transistori di tipo *MOSFET* i raggi ultravioletti forniscono energia sufficiente per muovere gli elettroni intrappolati nei Gate nella operazione di memorizzazione .

Per la cancellazione occorre estrarre il chip dalla scheda, togliere il ricoprimento di materiale fotoisolante, posto sulla finestra trasparente, ed esporre questa alla sorgente di raggi ultravioletti per un tempo di circa 15 minuti.

Occorre poi ricoprire di nuovo la finestra trasparente con il materiale fotoisolante e montare di nuovo il chip nella scheda.

La programmazione di una memoria *EPROM*, dopo la cancellazione avviene con una apposita apparecchiatura detta *EPROM - PROGRAMMER* che può interfacciarsi con il *PLC*.

Da quanto esposto una nuova programmazione di una memoria *EPROM* risulta poco agevole, non ha la versatilità presentata dalla memoria *RAM*. Rispetto a questa ha il vantaggio di essere non volatile

La memoria *EPROM* è adatta per contenere programmi che si mantengono inalterati in una grande - media serie di prodotti nei quali non occorrono modifiche.

2.6.8.5 Memoria EEPROM

La memoria *EEPROM (Electrically Eraseble Programmable Read Only Memory)* è non volatile, principalmente di lettura; il suo contenuto può essere cancellato e riprogrammato elettricamente tutto o in parte.

La cancellazione e riprogrammazione avviene per via elettrica, agendo direttamente sul chip senza smontarlo dalla scheda e dal circuito in cui è inserito. Occorre evidentemente un circuito di supporto per collegarsi con il chip sul quale si agisce con segnali a 24 Volt, ottenendo la cancellazione e la riprogrammazione

Le memorie *EEPROM* sono adatte a contenere programmi utente.

2.6.8.6 Memoria EAROM

La memoria *EAROM (Eletrically Altereble Read Only Memory)* è una memoria non volatile principalmente di lettura; il suo contenuto può essere, come in una *RAM*, modificato tutto o in parte.

La modifica del programma avviene per via elettrica e, come per la *EEPROM*, non occorre smontare il chip dalla scheda: la modifica avviene, agendo direttamente su di esso, per via elettrica.

2.6.9 MEMORIE DI LAVORO

Come si è già accennato nella descrizione dell'architettura del *PLC*, durante l'elaborazione di un programma, i dati, i risultati intermedi, debbono essere memorizzati temporaneamente su una parte della memoria, per poter essere poi estratti o manipolati. Questa è la *memoria di lavoro* della quale, una parte non accessibile all'utente, è destinata a contenere dati forniti dal sistema operativo, un'altra parte accessibile, contiene i dati degli stati di ingresso, di uscita, i contatori, i temporizzatori , i flag, i registri.

Diamo una breve descrizione delle parti speciali quali i registri, flag. ecc.

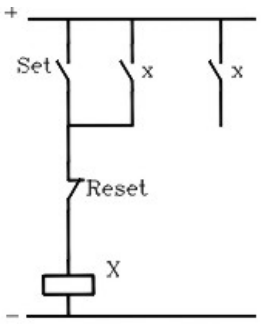
2.6.9.1 Registri

Nella elaborazione di un programma debbono essere estratte delle informazioni parziali, oppure si ottengono dei risultati intermedi che debbono essere memorizzati temporaneamente su parti di memoria che vanno sotto il nome di *registri*.

In questi le informazioni memorizzate sono composte da più bit.

2.6.9.2 Flag

fig.2.22



Per comprendere il concetto di *flag*, conviene riferirsi ad esempi di circuiti a contatti. In questi per risolvere alcuni problemi occorre utilizzare delle bobine ausiliarie.

Così, se in un comando necessita di una memoria *x*, questa si ottiene con un circuito che eccita una bobina ausiliaria *X*, il cui contatto (NA o NC) fornisce il *segnale di memoria*.

La bobina ausiliaria *X* si presenta come un *risultato intermedio (viene posta nella zona d'azione)* che può essere interrogato, attraverso i suoi contatti, e fornire segnali in altre parti del circuito di comando.

Nel *PLC* il circuito elettrico di comando è sostituito dal programma, e le bobine ausiliarie sono sostituite da elementi di memoria detti *flag*.

Questi si presentano come dei risultati intermedi che possono essere interrogati, nel proseguo del programma, il numero di volte che necessita.

Nel linguaggio a contatti (Ladder , vedi oltre), un *flag* va posto nella zona d'azione come se fosse una bobina ausiliaria, e i suoi contatti vanno designati con lo stesso simbolo del flag. In effetti quando il programma legge il contatto fittizio del *flag* viene interrogato lo stato logico del corrispondente elemento di memoria.

I *flag* costituiscono tanti elementi di memoria organizzati in una matrice di *r* righe e *c* colonne; e possono essere scritti o letti mediante la scrittura o il richiamo dell'indirizzo costituito, rispettivamente, dalla riga e colonna di appartenenza del bit costituente il *flag*.

Così supposto che *F* sia il simbolo assoluto dei *flag*, la scrittura:

F5.3

rappresenta il flag che è sulla *5* riga e *3* colonna

Lo stato del flag (la cella di memoria) come una bobina di memoria può essere posta allo stato logico *1* o *0*; si può quindi settare o resettare. Come per i contatti di una bobina si può interrogare lo stato logico della cella di memoria del flag in forma vera o negata.

2.6.9.3 Temporizzatori

Nei programmi di automazione risulta necessario effettuare delle temporizzazioni di alcuni segnali, i quali, a seconda dei casi, debbono subire un ritardo dell'attivazione o della disattivazione dall'istante del comando di settaggio o resettaggio; oppure debbono durare per un certo tempo prefissato.

La temporizzazione può essere attuata all'esterno del *PLC* per via hardware o all'interno per vie software.

La temporizzazione per via software è di tipo digitale, vi è quindi un tempo minimo di impostazione, corrispondente al quanto *"q"* unitario.

La impostazione di una temporizzazione corrisponde ad un multiplo del quanto unitario.

Per la temporizzazione via software vi è un particolare programmatore. Per iniziare la temporizzazione occorre un segnale di avvio, il quale, con la sua variazione di livello (fronte di

salita o di discesa), determina il settaggio del temporizzatore che si pone dallo stato logico *0* allo stato 1. Alla fine del periodo di temporizzazione esso torna allo stato logico *0*.

Lo stato logico 1 del temporizzatore viene comunicato all'Unità di Controllo che può così verificare se: non è iniziato, è in atto o è terminato il periodo di temporizzazione.

Il temporizzatore può essere interrogato in qualsiasi punto del programma come se fosse una uscita. Esso, in un linguaggio con schema a contatti, viene posto nella zona di azione e, la sua interrogazione, viene effettuata con dei contatti contraddistinti con la stessa sua etichetta, (vedi linguaggio Ladder).

In conclusione, per programmare una temporizzazione occorre:

1. Impostare il periodo di temporizzazione (multiplo del quanto unitario).
2. Immettere la condizione di settaggio come variazione di un segnale. Questa pone il temporizzatore allo stato logico *1*.
3. Terminato il periodo di temporizzazione impostato, il temporizzatore si porta allo stato logico *0*.

Un *PLC* può essere dotato di un certo numero di temporizzatori che possono funzionare contemporaneamente e che rappresentano una sua caratterizzazione.

Un altro elemento che caratterizza il tipo di *PLC* è il minimo tempo impostabile (quanto q digitale), che può essere, per esempio *0.01* s oppure *0.02* s.

Un altro elemento essenziale è la portata massima. Questa può essere dell'ordine della decina di minuti. Per ottenere temporizzazioni superiori si possono adoperare più temporizzatori che vengono interrogati uno di seguito all'altro.

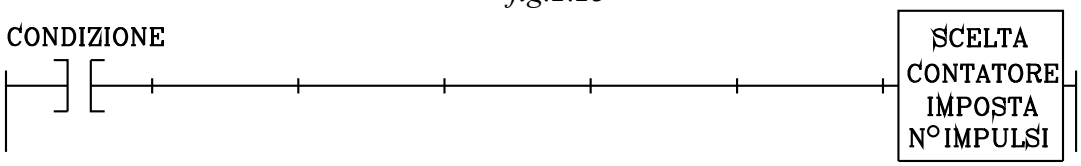
2.6.9.4 Contatori

Nei PLC si possono effettuare per via software operazioni di conteggio.

Il conteggio viene impostato in un particolare blocco nella zona di azione. In esso viene scelto il tipo di contatore e fissato il numero di impulsi da contare. Il valore impostato va memorizzato in un opportuno registro, in modo che esso possa essere confrontato con il valore del conteggio attuale.

Considerando un programma a contatti, l'impostazione viene attivata da una condizione posta nella zona di test di una linea di corrente, nella cui zona d'azione viene posto il blocco contatore.

fig.2.23

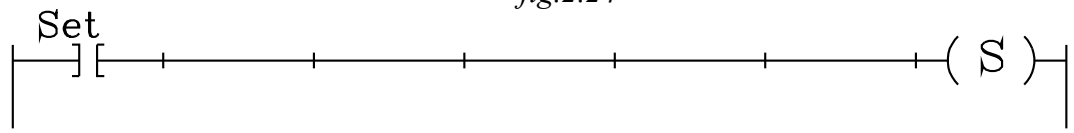


Nel blocco viene scelto il tipo di contatore e impostato il valore del conteggio desiderato.

Il contatore può essere indicato con una etichetta simbolica alla quale occorre, poi, associare l'indicazione dell'operando assoluto, riconosciuto dal programma.

Il contatore una volta impostato deve essere, poi, attivato da un segnale di Set. Questo può essere coincidente con la condizione alla quale si è fatta corrispondere l'impostazione oppure è un altro segnale successivo.

fig.2.24



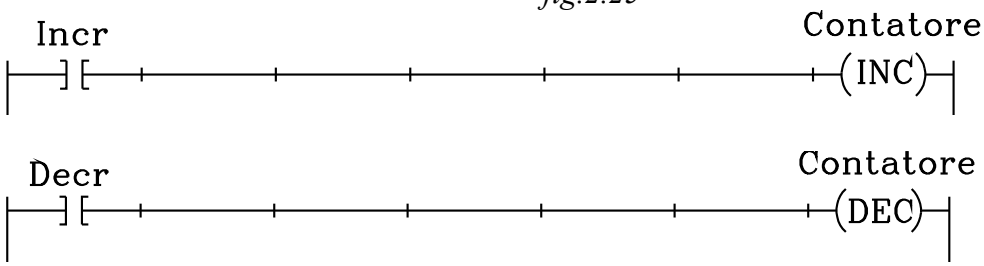
Settato il contatore, questo si pone in un registro allo stato logico 1 e può essere interrogato nel proseguo del programma. In quello a contatti come un contatto NA o NC posto nella zona di test.

Se l'interrogazione del contatore, nella zona di test, viene utilizzato come un contatto NA, esso corrisponde ad un ingresso che si porta allo stato logico 1 quando è attivato il contatore. Se, invece, è utilizzato come contatto un NC, questo si pone allo stato logico 1 quando è disattivato il contatore.

Il conteggio può essere effettuato *in su* "con incremento" o *in giù* "con decremento".

Nella zona di test vi deve essere un segnale che determina l'incremento o il decremento del conteggio, a seconda della indicazione riferita al contatore posto nella zona d'azione.

fig.2.25



2.6.10 ALU

È l'unità nella quale vengono eseguite operazioni aritmetiche e logiche.

Essa è costituita da una rete combinatoria, realizzata da circuiti elettronici elementari, collegati in modo da effettuare le porte logiche fondamentali AND OR NOT

Tali circuiti vengono abilitati dal codice operativo della istruzione decodificata, che seleziona il tipo di operazione da effettuare sugli operandi

La rete che costituisce l'ALU non contiene memorie, perciò, occorre munirla di memorie di servizio, utilizzate per memorizzare temporaneamente i dati che vi sono agli ingressi e alle uscite delle porte logiche.

Le memorie di servizio che contengono i dati di ingresso o di uscita dalle porte logiche vengono denominati *Registri* o *Accumulatori*.

Occorrono anche altre memorie di servizio che contengono particolari condizioni. Queste memorie vengono denominate *Flag di controllo*.

Come si è più volte detto il *CPU* di un *PLC* funziona in logica sequenziale e ciclica: viene eseguita una istruzione alla volta, letta dal Program Counter nella memoria di programma dalla cella di indirizzo più basso *00000000* al più alto e inviata *nell'ALU*.

Occorre che alla fine della scansione, raggiunta la cella di memoria di indirizzo più alto, vi sia una istruzione di fine programma, in modo che l'ALU faccia ripartire il *Program Counter* a leggere la prima cella di memoria di indirizzo *00000000* e ottenere così un funzionamento ciclico.

2.6.11 ELABORAZIONE DI UNA ISTRUZIONE

Come già si è detto, le istruzioni, poste nella memoria di programma, vengono lette una alla volta dal Program Counter, con una sequenza ciclica secondo l'ordine degli indirizzi, partendo dall'ordine più basso a quello più alto. Raggiunto questo, il Contatore di Programma torna a leggere la cella di memoria di ordine più basso, e così via.

Si hanno così successivi cicli di lettura che si svolgono secondo una cadenza data da impulsi emessi da generatori interni.

Il contenuto delle celle della memoria di programma, lette sequenzialmente dal Program Counter, viene trasferito nel Registro Istruzioni.

Le istruzioni vengono poi decodificate estraendo così la parte contenente gli indirizzi degli operandi sui quali si debbono effettuare le operazioni logiche nell'ALU e il codice operativo contenente l'operazione da effettuare in quello.

Per chiarezza seguiamo i passi che si succedono per eseguire una istruzione.

L'istruzione consista nell'effettuare l'operazione di AND tra due ingressi e trasferire il risultato in una uscita.

Gli ingressi siano:

I1.2 È l'ingresso *2* del modulo *1*.

I1.4 È l'ingresso 4 del modulo *1*.

L'uscita sia

O1.3 È l'uscita *3* del modulo *1*.

L'operazione da eseguire sia:

I1.2 AND I1.4= O1.3

L'istruzione corrisponde ad effettuare l'operazione logica AND tra lo stato logico dell'ingresso *I1.2* e *I1.4* e trasferire lo stato logico risultante nell'uscita *O1.3*.

L'operazione viene eseguita svolgendosi nei seguenti passi successivi. *Riferirsi alla fig.2.19*.

- Il Program Counter legge la cella che contiene l'istruzione *I1.2* che rappresenta l'operando riferito all'ingresso 2 del modulo 1.
- L'istruzione viene scritta nel Registro Istruzioni, dove viene temporaneamente memorizzato e suddivisa: nell'indirizzo ove reperire il dato e nell'operazione da svolgere
- Viene decodificato l'indirizzo dell'istruzione, che corrisponde all'ingresso *I1.2*. L'informazione transita nel bus indirizzi
- Viene decodificato il codice operativo dell'istruzione, che corrisponde a trasferire lo stato logico dell'ingresso, individuato dall'indirizzo, nell'accumulatore *dell'ALU*.
  Così, il dato, corrispondente allo stato logico dell'ingresso *I1.2*, si trasferisce, attraverso il bus dati, dalla *memoria dati all'accumulatore dell'ALU*, dove viene temporaneamente memorizzato.

- Il Program Counter si è intanto incrementato di una unità e va a leggere l'istruzione successiva.

- L'istruzione è l'operazione *AND* che viene trasferita nel Registro Istruzioni e, decodificata, attraverso il bus dati va a selezionare e abilitare una porta logica *AND* presente *nell'ALU*.
  Ad un ingresso della porta *AND* è associato lo stato logico dell'ingresso *I1.2*, precedentemente memorizzato nell'accumulatore.

- Il Program Counter si è intanto incrementato di una unità e va a leggere l'istruzione successiva.

- L'istruzione è di lettura dello stato logico dell'ingresso *I1.4*. L'istruzione viene letta dal P.C. e trasferita nel *R.I.*, ove viene decodificato l'indirizzo dell'ingresso *I1.4* e il Codice Operativo. Questo trasferisce attraverso il bus dati lo stato logico di *I1.4* dalla memoria dati all'accumulatore, che lo assegna al secondo ingresso della porta logica *AND* precedentemente abilitata.

- A questo punto viene effettuata l'operazione *AND* e all'uscita della porta logica è disponibile lo stato logico risultante

- Il Program Counter si è intanto incrementato di una unità e va a leggere l'istruzione successiva.

- L'ultima istruzione *"=O1.3"* è di trasferimento dello stato logico in uscita dalla porta AND all'uscita 3 del modulo 1.
- L'istruzione viene letta dal *P.C.* e trasferita nel *R. I.*
- Viene decodificato l'indirizzo dell'istruzione, che corrisponde all'uscita *O1.3*. Questa viene individuata attraverso il bus indirizzi
- Viene decodificato il codice operativo dell'istruzione, che corrisponde a trasferire lo stato logico dell'uscita della porta *AND nell'ALU*, attraverso il bus dati, nella memoria dati e da questa nell'uscita *O1.3* .

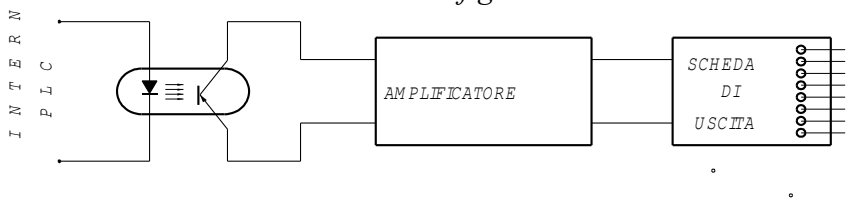
2.6.12 MODULI DI USCITA

I segnali di ingresso vengono elaborati nel *CPU* del *PLC* secondo il programma immesso nella memoria di programma che viene svolto ciclicamente, un'istruzione alla volta.

Si ottengono, come risultato, i segnali prelevabili nelle schede di uscita

Queste rappresentano le interfacce tra i segnali elaborati dal *CPU* del *PLC* e gli attuatori esterni, che fanno evolvere il processo controllato verso lo stato futuro voluto

fig.2.26



Occorre anche in questo caso isolare elettricamente la parte interna al PLC, operante tra 0÷5 Volt e la parte esterna, operante a più alta tensione (*24, 48, 110, 220* Volt).

Per ottenere detto isolamento vengono utilizzati, usualmente gli optoisolatori

I segnali di uscita dall'unità centrale sono di bassa potenza; essi, per poter agire sugli attuatori, debbono essere amplificati.

Dopo l'optoisolatori, il segnale viene inviato in un amplificatore di potenza.

Importante è la massima corrente che può essere assorbita dagli attuatori. Le schede di uscita, in generale possono fornire fino ad un massimo di 2 A; per correnti superiori, nel caso di segnali *ON/OFF*, il segnale di uscita viene inviato ad una bobina la cui eccitazione chiude un contato che alimenta l'attuatore

Come per gli ingressi le uscite possono essere costituite da più moduli ed ogni modulo contiene un numero di uscite multiplo di 2. Usuali sono 8 , 16 uscite.

Una uscita è identificata da un simbolo rappresentante il tipo di operando (uscita): ad esempio *O*, e da una parte indicante l'elemento utilizzato, composto dal modulo e numero di uscita

Esempio:

O2.3

indica l'uscita *(O)* numero 3 del modulo 2.

2.6.13 LINGUAGGI DI PROGRAMMAZIONE

Come si è già detto nel primo volume, le istruzioni di programma possono essere scritte direttamente il linguaggio macchina.

Il CPU non è altro che un circuito combinatorio di integrazione su larga scala (*LSI*), programmabile solamente con logica combinatoria, nella quale le istruzioni sono delle parole costituite da una sequenza dei caratteri 0,1 presi dall'alfabeto binario {*0,1*}.

Come si è detto, scrivere un programma in questa forma, pur se è possibile, risulta difficoltoso in quanto richiede una approfondita conoscenza del sistema.

Occorre un linguaggio più facilmente comprensibile dall'operatore del PLC, che usualmente non è un esperto di informatica, ma più dei processi produttivi.

I linguaggi di programmazione possono essere:

- Linguaggi orientati alla macchina, detti a basso livello: *Linguaggi Assembly*
- Linguaggi orientati alla soluzione del problema, detti ad alto livello

Nel seguente schema vengono indicati i linguaggi più usualmente utilizzati nel *PLC*

- *Linguaggi*
  - *basso livello: Assembly*
  - *alto livello*
    - *Lista istruzioni*
    - *Ladder*
    - *Funzionale*
    - *Informatico*

2.6.13.1 Linguaggi Assembly

Nel linguaggio macchina occorre adoperare parole composte dai due caratteri 0,1 dell'alfabeto binario {*0,1*}. Ciò comporta l'impiego di molto tempo, ed è richiesta una approfondita conoscenza del sistema macchina.

Si sono messi a punto dei linguaggi che sostituiscono con parole simboliche quelle del linguaggio macchina composte con i due *0,1* caratteri dell'alfabeto binario.

È evidente che anche con questa sostituzione le singole istruzioni sono particolarmente legate al tipo di macchina che si sta programmando. Si sono soltanto sostituite con parole simboliche delle lunghe sequenze di *0,1* che portavano facilmente ad errori e ad un allungamento del tempo di programmazione.

Il linguaggio Assembly non è direttamente eseguibile dalla macchina, in quanto questa non riconosce le parole simboliche che lo costituiscono.

Occorre tradurre le parole simboliche dell'Assembly in linguaggio macchina: in sequenze *0,1* le uniche che essa può riconoscere

Il linguaggio Assembly è strettamente legato al tipo di macchina e richiede una specifica abilità e conoscenza.

2.6.13.2 Linguaggi ad alto livello

Sono linguaggi che possono essere utilizzati in differenti CPU di PLC. L'operatore nella programmazione è rivolto alla soluzione del problema e non deve pensare alla particolare costituzione della macchina.

Scritto il programma nel linguaggio ad alto livello, si ottiene il *Programma Sorgente*.

Questo non è comprensibile dalla macchina; occorre che venga *tradotto* o *compilato* in linguaggio macchina ottenendo *il file in Codice Oggetto*. Questo è composto da istruzioni con parole in binario, comprensibili dal CPU del PLC e, quindi, eseguibile.

Per ottenere il programma in linguaggio macchina, come si è già detto nel 1° volume, vi sono due modi di traduzione: attraverso il *compilatore* o il *traduttore* .

Con il *compilatore* viene tradotto tutto il programma, digitato nel linguaggio ad alto livello, in un altro programma in linguaggio macchina, ottenendo così il **f***ile in Codice Oggetto*, che può essere interamente eseguito dalla macchina. Il paragone: è come se un intero libro in lingua inglese venga tradotto in italiano: un Italiano potrà leggere interamente il secondo libro tradotto.

Con il *Traduttore* invece, il programma, scritto in linguaggio ad alto livello, viene direttamente posto in esecuzione, inviandolo ad un traduttore, il quale , istruzione per istruzione, lo traduce in linguaggio macchina eseguibile dal CPU del PLC. Il paragone: è come se un Italiano, dovendo parlare con un Cinese di cui non conosce la lingua, per farsi capire, impiegasse un Interprete. Questo tradurrà, ogni parola che viene detta al momento, dall'italiano in cinese.

Analizziamo ora brevemente le caratteristiche dei diversi linguaggi ad alto livello.

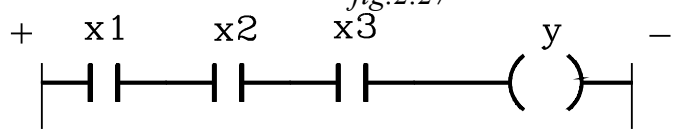
2.6.13.2 Lista di istruzioni

Come si è posto in rilievo nel 1° volume una funzione logica binaria può essere rappresentata da un circuito a contatti e viceversa.

Così la soluzione di un problema di automazione risolto con un circuito elettrico a contatti può tradursi in funzioni binarie, contenente relazioni booleane e viceversa.

Così, per esempio, porre più contatti in serie corrisponde ad effettuare una operazione AND tra le variabili rappresentate dai contatti.

fig.2.27



$y = x_1 \cdot x_2 \cdot x_3$

L'espressione $y = x_1 \cdot x_2 \cdot x_3$ si può tradurre nella istruzione:

$$x_1 \cdot x_2 \cdot x_3 \xrightarrow{istruzione} x_1 \ AND \ x_2 \ AND \ x_3$$

Risulta evidente che il programma viene ad essere costituito da una lista di istruzioni, che si susseguono in ordine una dietro l'altra, con la possibilità di poter effettuare anche dei salti condizionati o no.

Conviene effettuare il diagramma di flusso prima di stendere la lista di istruzioni.

I programmi in lista di istruzione dei PLC, a seconda della casa costruttrice, contengono, oltre alle funzioni logiche principali *AND OR NOT*, un *set di istruzioni* con parole riservate adibite a particolari funzioni.

Così per il settaggio di una uscita in alcuni PLC è riservata la parola *SET* seguita dalla indicazione della particolare uscita:

SET O2.1

In altri PLC al settaggio di una uscita è riservata la parola *OUT*...

Il programma in lista di istruzioni viene memorizzato in un file detto "**f***ile sorgente*". Questo non è direttamente compreso ed eseguibile dal CPU del PLC, esso viene trasformato nel *"file in codice oggetto"* composto di istruzioni formate di parole in binario che la macchina può comprendere ed eseguire.

La stesura del programma concettualmente è la stessa per tutti i tipi di PLC, diverge ovviamente nel *set di parole riservate*. Per cui, per lo studio operativo di un linguaggio in lista di istruzioni, conviene riferirsi ad un determinato *PLC*.

2.6.13.3 Linguaggio ladder

È il linguaggio che si riferisce ad un programma che si risolve con schema a contatti, con lo stesso criterio utilizzato nella logica cablata.

La dizione Ladder proviene dal fatto che il circuito si svolge secondo linee orizzontali come i pioli di una scala: ladder tradotto in lingua inglese.

Si debba così effettuare il semplice ciclo $A^+ \ A^-$ con un cilindro a doppio effetto, impiegando una valvola monostabile.

Si indichi con Start il contatto di avvio del ciclo e, con a_0 , a_1 rispettivamente il finecorsa che controlla il rientro dello stelo e quello della fuoriuscita

Le equazioni logiche dei due stati sono

$$\begin{array}{ll} A^+ = a_0 \cdot Start & \begin{cases} Segnale \ di \ Set & A^+ = a_0 \cdot Start \\ Segnale \ di \ reset & A^- = a_1 \end{cases} \\ B^+ = a_1 & \end{array}$$

Adottando la stessa logica impiegata nell'elettropneumatica, essendo la valvola utilizzata monostabile, si impiega un circuito di memoria, per ottenere l'autoritenuta una volta che si toglie il segnale di Set.

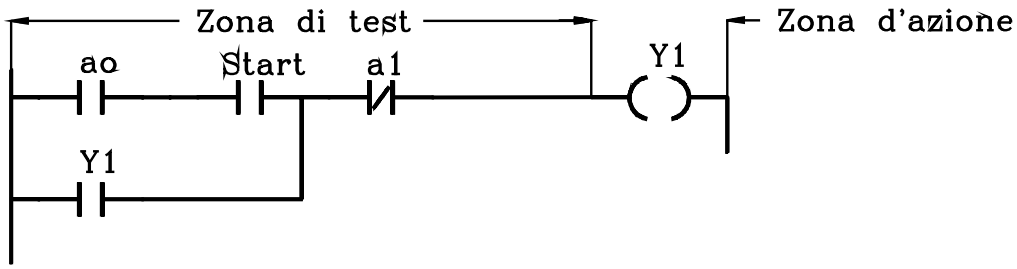
Nel PLC si possono interrogare anche le uscite, quindi non occorre impiegare una bobina ausiliaria, che costituirebbe un flag.

Se con Y1 si indica la bobina dell'elettrovalvola monostabile, l'equazione di comando di essa si può scrivere:

$$Y1 = (ao \cdot Start + Y1) \cdot \overline{a1}$$

Da cui si ottiene il circuito a contatti:

fig. 2.28



In alcuni *PLC* si può caricare un programma grafico con il quale si può digitare direttamente sul video il circuito a contatti. Questi vengono immessi o da tastiera pigiando opportuni tasti, oppure vengono selezionati da menu, ai quali si accede mediante tastiera o con il mouse.

Nel caso che il programma del *PLC* è grafico, esso si risolve digitandolo sullo schermo il circuito a contatti, ottenuto con gli stessi metodi utilizzati nell'elettropneumatica: metodo delle *mappe di Karnaugh*, delle *memorie in cascata*, dei *circuiti sequenziali*, più altri che sfruttano la versatilità del *PLC* in dotazione.

Osservando il circuito a contatti assunto come esempio, viene posto in evidenza che la linea di corrente si può suddividere in due parti:

Zona di test È la zona contenente i contatti. Essa determina il segnale di uscita quando risulta 1 la combinazione degli stati logici dei contatti rappresentati.

Zona d'azione È la zona nella quale si ha il risultato della combinazione degli stati logici dei contatti rappresentati nella *zona di test*. Questa zona rappresenta un risultato, e quindi una uscita esterna collegata agli attuatori; oppure una uscita interna, che attiva un flag o imposta e setta un temporizzatore...

Come si è detto il circuito a contatti può tradursi in una funzione logica binaria, e questa può essere espressa con una lista di istruzioni, costituente il *file sorgente*.

Nel programma didattico della *FESTO* per *FPC 400* il software permette di programmare digitando da tastiera i contatti che compaiono sul video su linee logiche orizzontali. Il file ha l'estensione *".KOP"*; viene tradotto nel *file sorgente* in lista di istruzione in codice ASCII con estensione "*.AWL"*.

Il *file sorgente* può essere digitato con un qualsiasi programma di scrittura in caratteri ASCII. Occorre memorizzarlo con l'estensione "*.AWL*"

Si noti che, in effetti, la *zona di test* del circuito posto in esempio rappresenta un condizionale, nel senso che: solamente se si verificano le condizioni logiche dei contatti, la cui combinazione rappresentata dal circuito dà come risultato lo stato logico 1, allora viene attivata l'uscita: Y1 risulta allo stato logico 1.

Si ha così una istruzione del tipo *IF... THEN... ELSE* che tradotto si può scrivere:

```
SE              ( A0=1  E  START=1   OPPURE    Y1=1 ) E A1=0
ALLORA          ATTIVA Y1
ALTRIMENTI      DISATTIVA Y1
```

Nel programma della *FESTO* l'istruzione di attivazione è data dalla parola riservata SET, la disattivazione dalla parola *RESET*.

Per esprimere che un elemento è allo stato logico 1 si sottintende l'uguaglianza ad 1. Così l'espressione *A0=1* è indicata solamente dal simbolo A0.

Lo stato logico *0* viene espresso dalla negazione dello stato logico 1. Così l'espressione A1=0 viene data dall'istruzione *NOT A1*.

Il programma sorgente in forma compatta si può così scrivere:

```
IF ( A0  AND START  OR  Y1 )  AND  NOT A1
THEN        SET Y1
PSE
ELSE        RESET Y1
PSE
END
```

La parola riservata *PSE* va posta prima e dopo l'ultima istruzione, che precede *END*. Essa predispone il *CPU* a porsi in fine programma e, dopo END, ad iniziare di nuovo a far leggere al *P.C* la memoria di programma, a cominciare dalla cella di indirizzo più basso *0000000*.

In effetti come si è detto il Program Counter legge una istruzione alla volta.
Il programma sorgente ha l'estensione AWL ed è scritta in caratteri ASCII. Leggendolo con *TYPE* il file *AA.AW*L si ottiene:

```
PROGRAM B:AA
"
"
"$ Percorso corr.N.  1
"
IF      (     A0
    AND       START
    OR        Y1
        )
    AND   NOT   A1
THEN  SET       Y1
    PSE
ELSE  RESET     Y1
    PSE
"
"
"$
END
```

File con estensione ".DEC"

Nel file sorgente, per gli operandi (ingressi ,uscite, flag...), vengono usati dei simboli che più facilmente sono riconoscibili dall'utente, ma che non corrispondono a quelli accettati dal programma (operandi assoluti). Perciò occorre associare ad ogni *operando simbolico* quello *assoluto* riconosciuto dal programma (vedi oltre)

Così, il finecorsa A1 è un ingresso. Al simbolo A1 si può associare un operando assoluto, ad esempio *I2*...

In un file con estensione "*.DEC*" si associa ciascun operando simbolico ad uno assoluto.

Alla fine occorre tramutare il *file sorgente* nel *file codice oggetto*, contenente il linguaggio macchina riconoscibile dal *PLC.*

2.6.13.4 Linguaggio funzionale

Come si è più volte detto, una funzione logica si può tradurre in schema a contatti e viceversa.

Una funzione logica può essere rappresentata con schemi funzionali. Ne viene che un qualsiasi problema di comando risolvibile con funzioni logiche può essere risolto: scrivendo queste direttamente in lista di istruzioni, o con schema a contatti, oppure con schemi funzionali.

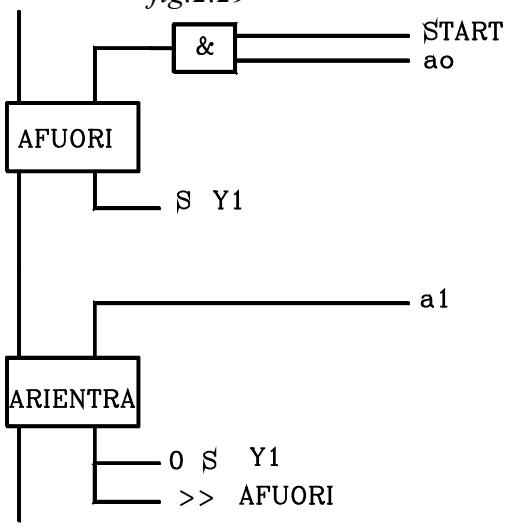
Sono stati messi a punto dei pacchetti software nei quali le istruzioni vengono immesse portando su video o da tastiera o con il mouse i blocchi logici costituenti i successivi passi del programma.

Così, per esempio, consideriamo ancora il semplice ciclo A^+ A^- con l'impiego di una elettrovalvola monostabile *Y1*.

Si indichino ancora con:

START Pulsante di start
a_0 Finecorsa in posizione di stelo rientrato
a_1 Finecorsa in posizione di stelo fuoriuscito
Y1 Bobina dell'elettrovalvola monostabile

fig.2.29



Le uscite possono essere:

- Settate permanentemente (con memorizzazione): Indicazione *S*
- Settate non permanentemente (senza memorizzazione): Indicazione *NS*

- Resettate 0 davanti a *S* o *NS*

Con le possibilità di settaggio permanente in un passo e resettaggio nel passo successivo il diagramma funzionale si presenta nella forma riportata in figura (vedi oltre)

Il file dello schema funzionale ha l'estensione "*.FUP*". Questo viene poi tradotto nel seguente *file sorgente* con estensione "*.AWL*"

```
PROGRAM  B:AA
"
"
"$ Step No.  0
"
STEP  AFUORI
IF              START

    AND         A0
THEN  SET         Y1
"
"
"$ Step No.  1
"
STEP  ARIENTRA
IF              A1
THEN  RESET       Y1
    JUMP TO     AFUORI
ELSE  SET         Y1
    JUMP TO     ARIENTRA
"
PSE
"
"$
END
```

2.6.13.5 Linguaggio informatico

Negli ultimi anni la tendenza delle case costruttrici è di porre in commercio pacchetti software che possono essere impiegati nei *PC* e girare con il sistema operativo *MS - DOS*.

Il programma viene digitato sulla tastiera del computer e visualizzato sul video ad esso collegato. Il programma viene poi tradotto in linguaggio macchine ed inviato al *PLC*.

Da questa tendenza alcune case costruttrici hanno messo a punto dei programmi per *PLC* con linguaggi che più usualmente vengono adoperati per programmare su un Personal Computer, quale il linguaggio Basic (vedi oltre).

In questo caso il programma si identifica con la stesura del *flow-char*t.

Ricordiamo che un circuito a contatti con la zona di Test e quella di Azione, corrisponde ad un condizionale, che si traduce nel flow-chart in un blocco di diramazione a più ingressi. Da questa semplice constatazione risulta evidente la possibilità di poter tradurre uno schema a contatti in un *flow-chart*.

Effettuato il *flow-chart* ogni blocco viene tradotto in corrispondenti istruzioni.

Alla fine il programma scritto viene tradotto, istruzione per istruzione in linguaggio macchina da un Interprete, programma memorizzato in una memoria *EPROM*.

Clic per continuare

Clic qui per la pagina iniziale