

## Grafcet

### Il controllo di Sistemi ad Eventi Discreti

Si definiscono con il termine di *Sistemi ad Eventi Discreti* quei sistemi in cui la evoluzione dello stato dei componenti avviene ad eventi piuttosto che in maniera continua. Anche se il considerare l'evoluzione di un sistema ad eventi discreti è solo una *approssimazione della realtà*, tale semplificazione risulta spesso efficace ed è tale da snellire notevolmente un problema di automazione.

Nella maggior parte dei casi, infatti, l'oggetto dello studio è un sistema molto complesso ed eterogeneo di macchine per cui, piuttosto che dover affrontare il problema di controllo della singola macchina, si richiede la realizzazione di un *sistema di automazione e di sincronizzazione di più macchine*.

Lo stesso controllo delle singole componenti del processo viene progettato per astrazioni successive, sintetizzando preventivamente il controllo delle macchine per poter soddisfare dati requisiti di continuità, di rapidità e di qualità di automazione, in modo tale da poter approssimare il comportamento della singola macchina ad una semplice successione di stati ben definiti.

I metodi di sintesi del controllo dei processi sono soprattutto derivati dal tipo di "macchine" che vengono utilizzate oggi per il controllo di processo, i PLC (Controllori Logici Programmabili). Questi dispositivi discendono dalle reti a relais molto utilizzate fino agli anni '70 in gran parte dei processi industriali automatizzati e ne ereditano in gran parte la modalità di funzionamento. Il funzionamento di un PLC può essere così sintetizzato:

- A. Lettura istantanea dello stato del processo controllato tramite dispositivi di misura specializzati;
- B. Elaborazione, in funzione di tutte le letture dello stato in ingresso, delle azioni da compiere per modificare lo stato del processo nel modo desiderato;
- C. Attuazione delle uscite tramite attuatori comandabili direttamente;
- D. Ripeti da (A);

Uno dei maggiori problemi nell'utilizzo dei PLC, almeno all'origine del loro utilizzo, è stato la mancanza di metodi efficaci per la progettazione del controllo, nonostante la grande capacità di elaborazione da loro offerta. All'origine del gap tra metodi e tecnologia, un fenomeno che tuttora affligge gran parte delle macchine moderne, è il rapido accrescere delle capacità di elaborazione dei PLC rispetto a ciò che le reti a relais - progettate con metodi statici ed inadeguati per sistemi dinamici come i PLC - erano in grado di offrire. La mancanza di metodi, insieme alla differenziazione di prodotti offerta dalle case costruttrici di PLC (fino alla comparsa di nuovi standard, la maggior parte dei PLC era assolutamente incompatibile da modello a modello), ha reso molto critica la manutenzione dei processi di automazione, per cui la necessità di nuovi metodi è diventata oltremodo pressante.

Nel solco di questa problematica, nel 1975 in Francia si cominciò a sviluppare il **GRAF CET** (**Gr**aphe de **Co**ordination **E**tapes **T**ransitions), un rappresentazione del processo di automazione mediante un diagramma funzionale standardizzato, capace di rappresentare processi sequenziali (processi il cui svolgimento avviene a passi e la transizione da un passo al successivo è subordinato a certe condizioni).

### Il metodo basato sul Grafcet

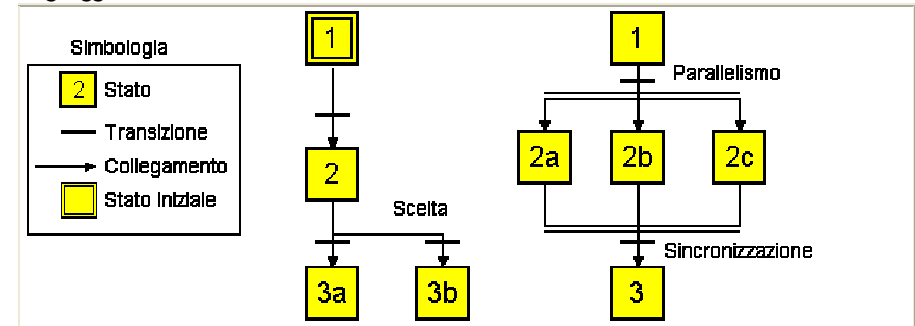
Fino alla definizione del Grafcet nel 1982, esistevano diversi metodi che potevano essere applicati alla risoluzione di problemi logici dinamici, la maggior parte dei quali diventano gradualmente più inadeguati al crescere della complessità del problema da risolvere.

Il Grafcet, diagramma funzionale standardizzato dall'UTE (Union Technique de l'Electricité) che fa uso del concetto di *Stato*, *Transizione* e di *Collegamento Orientato*, consente di strutturare un problema secondo livelli di astrazione successiva, offrendo inoltre la possibilità di traduzione diretta della sua struttura in uno dei linguaggi a basso livello normalmente utilizzati nella programmazione dei PLC (ad esempio il linguaggio grafico LADDER o il linguaggio assembler dei PLC).

Nel corso degli anni '80 il Grafcet fu ulteriormente sviluppato e il successo fu tale che nel 1987 fu assunto come standard internazionale dall'IEC (Comitato Elettrotecnico Internazionale). Infine la norma IEC 1131 - 3 del 1993 (recepita in Italia nel 1996), cercando di mettere ordine nei vari linguaggi di programmazione dei PLC, ha incluso anche il Grafcet tra i linguaggi di programmazione con la nuova denominazione di **Sequential Functional Chart**

### Sequential Functional Chart

Il Grafcet è assimilabile ad un *diagramma degli stati* o ad un *diagramma di flusso*, ma ne differisce per il fatto che mentre in questi ultimi ogni stato è individuato da una ben precisa configurazione (invariante) delle uscite, nel Grafcet uno Stato è individuato da un insieme di *Azioni*, mentre una transizione tra due stati è condizionata ad un evento che determina la fine di un'azione e l'inizio della successiva. Al contrario di molti diagrammi di flusso, inoltre, più stati possono essere contemporaneamente attivi e la possibilità di coordinazione di questi è una delle più importanti proprietà di questo linguaggio.

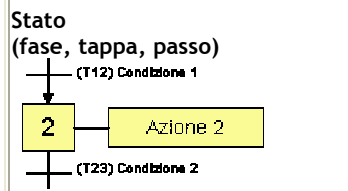
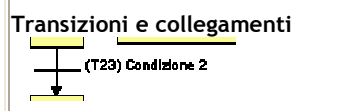


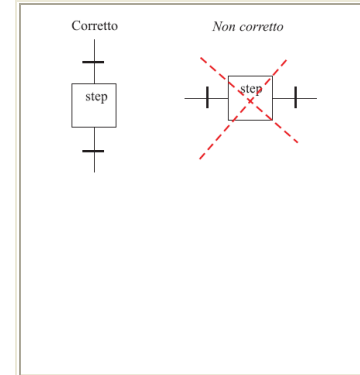
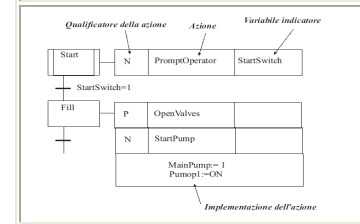
La formalizzazione nata con questo linguaggio permette di definire degli **stati iniziali** (stati che sono attivati all'attivazione del programma creato con il Grafcet), un collegamento di *Scelta alternativa* nell'attivazione di due o più stati, l'attivazione contemporanea di più stati attraverso un collegamento di *Parallelismo*, la convergenza sincronizzata di più stati attraverso un collegamento di *Sincronizzazione*. Per maggiori dettagli, è opportuno fare riferimento al documento che ne definisce in modo completo la formalizzazione.

## La formalizzazione del Grafcet

IL GRAFCET è un formalismo standardizzato nato per la descrizione e la progettazione del ciclo operativo di macchine ed impianti. Il formalismo è stato standardizzato dal Comitato Elettrotecnico Internazionale (IEC) come **Standard IEC-1131-3** con il nome di **Sequential Functional Chart (SFC)**. Come struttura, esso è assimilabile ad un diagramma degli stati e risulta indipendente dalla tecnologia utilizzata per l'implementazione. Un progetto in Grafcet può essere infatti tradotto in una serie di linguaggi diversi, posto che esistano dei modi, per questi linguaggi, per rendere possibile l'interfacciamento con il processo. L'uso di un formalismo astratto semplifica la rappresentazione stimolando la scomposizione in sotto-problemi. La scomposizione è opportuna poiché, in realtà, ad ogni situazione operativa solo un sotto-insieme delle informazioni provenienti dal processo è indispensabile per il controllo.

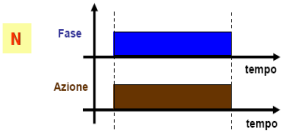
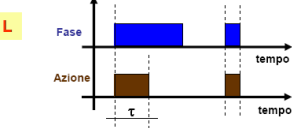
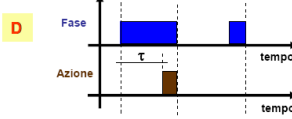
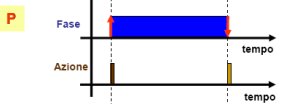
### Concetti di base sul Grafcet

<p><b>Stato</b> (fase, tappa, passo)</p> 	<p>uno stato è una condizione operativa della macchina alla quale è associato un ben preciso algoritmo di controllo (Azioni), diverso da quelli associati agli altri stati. L'attivazione di un particolare evento forza la transizione dallo stato attuale ad/ai successivi. In generale, durante il tempo di permanenza in uno stato le uscite del sistema di controllo possono variare in risposta alle variazioni degli ingressi o allo scorrere del tempo.</p> <p>La suddivisione in stati è possibile quando l'evoluzione temporale del funzionamento di un impianto complesso è descrivibile mediante una successione temporale di situazioni operative più semplici, nelle quali solo un sottoinsieme degli ingressi e delle uscite è attivo.</p> <p>Ad ogni stato vanno associate le AZIONI da intraprendere quando si è in quello stato. Un esempio di azione può essere un algoritmo di controllo, attivo solo quando il sistema si trova in quello stato. Ogni stato è individuato da un numero che lo identifica in modo univoco rispetto agli altri stati.</p>
<p><b>Transizioni e collegamenti</b></p> 	<p>Sono la possibilità di evoluzione da uno stato ad un altro. Non tutti gli stati ammettono tra loro una transizione. Ad ogni transizione è associata una condizione che deve essere verificata affinché la transizione sia attiva e sia quindi possibile l'evoluzione degli i stati che collega. Tra uno stato e il successivo in ordine di evoluzione il collegamento è rappresentato tramite un arco orientato. L'orientamento naturale prefissato è quello dall'alto verso il basso. Nel caso che il collegamento effettivo sia dal basso verso l'alto è obbligatorio orientare l'arco tramite una freccia.</p> <p>Ad ogni transizione va associata una sola CONDIZIONE (che però può essere la composizione logica di più eventi) che ne determina l'attivazione (passaggio ad</p>

	<p>un nuovo stato). Due transizioni successive non separate da uno stato sono proibite. Ogni transizione è individuata da un simbolo del tipo "Txy" (oppure "Tx-y") dove x e y sono lo stato di provenienza e di arrivo.</p> <p>Come regola di stesura dello schema, è importante sottolineare che uno schema SFC si sviluppa sempre in senso verticale, in cui l'evoluzione dei passi attivi avviene dall'alto verso il basso. I collegamenti arrivano ai passi o se ne dipartono in posizione verticale</p>
	<p>Il <b>qualificatore dell'azione</b>, che definisce il tipo della azione tra intraprendere (se impulsiva, continua, temporizzata, etc.). L'<b>identificativo</b> dell'azione stessa. La <b>variabile indicatore</b>, è una variabile booleana il cui stato indica il completamento dell'azione.</p>

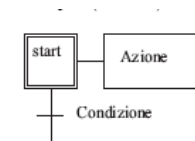
## I qualificatori

Qualificatore	Significato	Descrizione
nessuno		Di default, ha lo stesso significato di 'N'.
N	<i>Non-stored</i>	L'azione termina quando il passo diventa disattivo.
R	<i>Reset</i>	Termina un'azione attivata con i qualificatori S, SD, SL o DS.
S	<i>Set (stored)</i>	L'azione continua anche quando il passo diventa inattivo, e termina quando l'azione viene resettata.
L	<i>Time Limited</i>	L'azione comincia quando il passo diventa attivo; e continua finchè il passo diventa inattivo oppure trascorre un certo intervallo di tempo.
D	<i>Time Delay</i>	Un timer viene settato quando il passo diventa attivo; se il passo è ancora attivo dopo l'azzeramento del timer, l'azione comincia e termina quando il passo si disattiva.
P	<i>Pulse</i>	L'azione comincia quando il passo diventa attivo/disattivo e viene eseguita una sola volta.
SD	<i>Stored and time Delayed</i>	L'azione comincia dopo un ritardo anche se il passo diventa inattivo e continua finchè non è resettata.
DS	<i>Delayed and Stored</i>	Un timer viene settato quando il passo diventa attivo; se il passo è ancora attivo dopo l'azzeramento del timer, l'azione comincia e continua finchè non è resettata.
SL	<i>Stored and time Limited</i>	L'azione comincia quando il passo diventa attivo e continua finchè non viene resettata o non trascorre un certo intervallo di tempo.

	<p><b>L'azione di controllo (Non stored)</b></p> <ul style="list-style-type: none"> <li>• comincia nello stesso istante in cui la fase si attiva</li> <li>• termina nello stesso istante in cui la fase diviene inattiva.</li> </ul>
	<p><b>L'azione di controllo (time Limited)</b></p> <ul style="list-style-type: none"> <li>• comincia nello stesso istante in cui la fase si attiva</li> <li>• termina quando in cui la fase diviene inattiva</li> <li>• oppure quando è trascorso un tempo <math>\tau</math></li> </ul>
	<p><b>L'azione di controllo (Pulse)</b></p> <ul style="list-style-type: none"> <li>• comincia nello stesso istante in cui la fase si attiva o disattiva (secondo le opzioni)</li> </ul>
	<p><b>L'azione di controllo (Pulse)</b></p> <ul style="list-style-type: none"> <li>• comincia nello stesso istante in cui la fase si attiva o disattiva (secondo le opzioni)</li> </ul>

## L'evoluzione di un diagramma Grafcet

**Inizializzazione:** Occorre definire gli stati attivi all'inizio del funzionamento stati iniziali. Gli stati iniziali possono essere più di uno e si indicano con due quadretti uno dentro l'altro. Gli stati iniziali possono anche non essere i primi stati di uno schema.



## TRANSIZIONE

La **condizione logica** è il risultato di combinazioni logiche di più variabili:

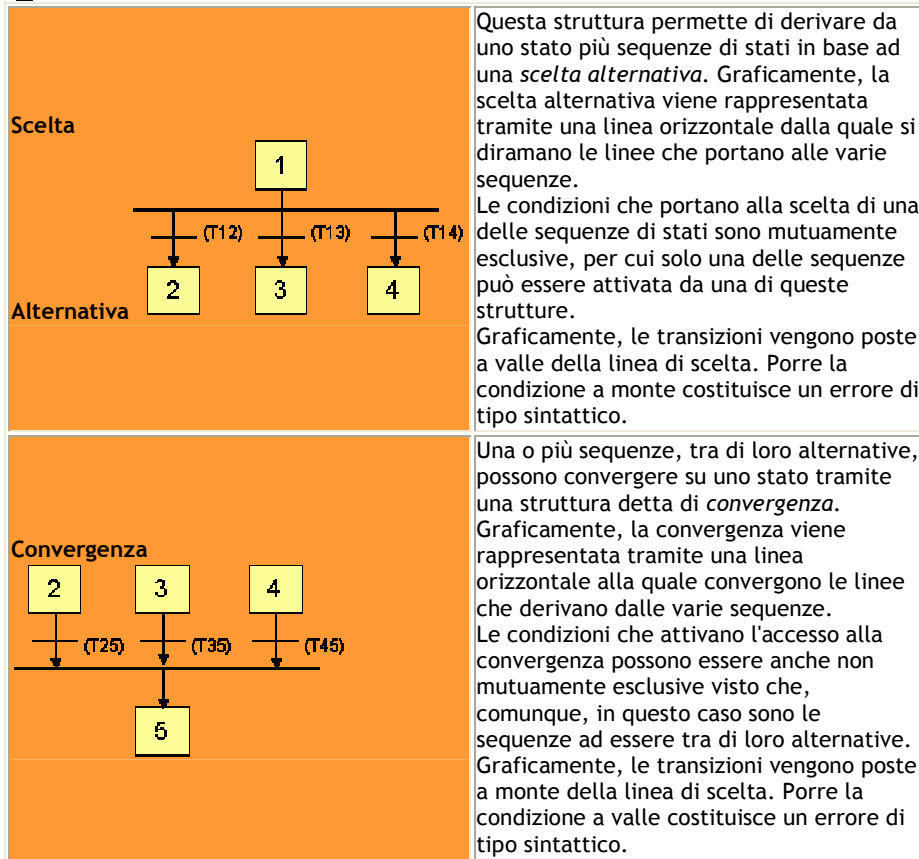
- **variabili di ingresso:** possono essere booleane (PresenzaPezzo, ...) o reali (pressione, temperatura, ...). Le variabili booleane possono essere a livello (alto/basso) oppure a fronte (salita/discesa).
- **variabili temporali:** si tratta di temporizzatori (timer) associati alle fasi. La loro uscita vale inizialmente 0 e rimane tale anche quando si attiva la fase cui sono associati. L'uscita commuta a 1 quando è trascorso un intervallo  $\tau$ .
- **variabili di stato:** variabili booleane X associate alle fasi. Esse indicano lo stato logico dalla fase.
- **variabili 'condivise':** variabili interne manipolate mediante le azioni e utilizzate per condizionare le transizioni.

Lo scatto di una transizione provoca:

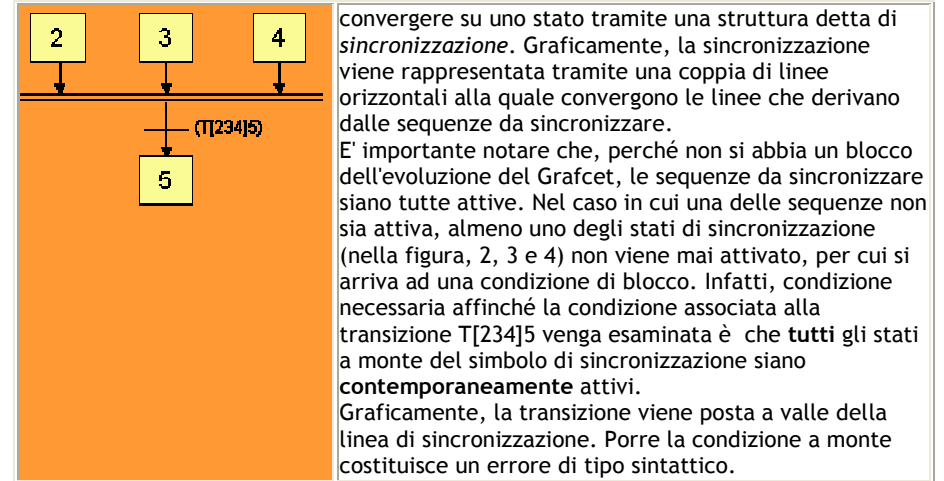
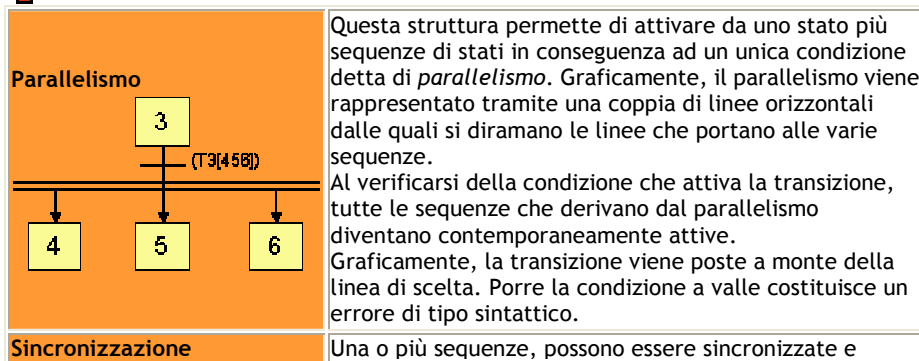
- la disattivazione della fase (o fasi) a monte
- l'attivazione della fase (o fasi) immediatamente a valle.

## Le strutture di collegamento

### ■ Scelta alternativa e Convergenza



### ■ Parallelismo e Sincronizzazione



Perché un grafcet sia sintatticamente corretto, è bene rispettare anche alcune regole:

- Le sequenze attivate da un *parallelismo*, prima o poi, devono sempre convergere in una *sincronizzazione*;
- Le sequenze che discendono da una *scelta alternativa* devono sempre convergere in una *convergenza*;

## Il Grafcet per il progetto di un Processo Automatizzato

L'idea di utilizzare il Grafcet è nata proprio per colmare il gap tra potenzialità del linguaggio grafico e la complessità del controllo da attuare sul processo automatizzato. Anche il Grafcet, come le reti di Petri, permettono una formalizzazione matematica successiva al progetto grafico del controllo. Al contrario delle reti di Petri, però, il fine della formalizzazione non è l'analisi del processo automatizzato ma la stesura del codice che realizza l'automazione desiderata. Per comprendere la procedura di conversione di un diagramma Grafcet in codice è necessario introdurre alcuni elementi fondamentali che la caratterizzano.

Dato un diagramma Grafcet che sintetizza l'automazione di un impianto, il procedimento di conversione di un diagramma in codice consiste, innanzitutto nel definire una serie di variabili booleane associate al processo:

- $S_i \in B^n$ : Vettore degli Stati: l'elemento i-esimo indica, se pari ad 1, che il relativo stato è attivo;
- $T_j \in B^m$ : Vettore Transizioni: l'elemento j-esimo indica, se pari ad 1, che le condizioni

che rendono attiva la relativa transizione sono soddisfatte;  
 A seconda del linguaggio utilizzato per la traduzione, occorrerà effettuare un diverso tipo di dichiarazione, come mostra la tabella che segue:

Tipo di linguaggio	Dichiarazione delle variabili di stato e di transizione
PLC: LADDER (simbolico)	Occorrerà dichiarare una serie di variabili corrispondenti ai due tipi di variabile. Con questo tipo di linguaggio è possibile riservare una serie di simboli del tipo "S1", "S2" ... "Sn" e "T1", "T2", ..., "Tm" come variabili interne, ovvero variabili di memoria. Normalmente, esiste una zona dell'ambiente di programmazione LADDER dove effettuare questa dichiarazione.
PLC: LADDER (non simbolico)	In questo tipo di linguaggio è necessario allocare una serie di celle di memoria. La convenzione che può essere scelta è normalmente casuale. Per il simulatore LADDER presente nelle pagine di questo sito, ad esempio, potrà essere scelta la corrispondenza variabili booleane - stati del tipo: <ul style="list-style-type: none"> <li>■ S1 = MB0.1, S2 = MB0.2, ... S7 = MB0.7, S8 = MB1.0, etc.</li> <li>■ T1 = MB4.1, T2 = MB4.2, ... T7 = MB4.7, T8 = MB5.0, etc.</li> </ul> E' importante che i byte scelti per l'allocazione delle variabili di stato siano distanti da quelli scelti per le variabili di transizione, in modo da evitare confusioni o sovrapposizioni nel momento in cui sia necessario definire nuove variabili di transizione o di stato. Può essere utile, per migliorare la leggibilità della traduzione, nominare gli stati allo stesso modo in cui è possibile nominare le variabili. In questo caso, gli stati saranno indicati come: S0.1=MB0.1, ..., S1.3=MB1.3, utilizzando la stessa convenzione per le transizioni: T0.1=MB10.1, T4.1=MB14.1.
PC: VISUAL BASIC	Per i linguaggi strutturati, la definizione di variabili di stato si presenta molto semplice. La classica definizione di un array di booleani risolve il problema: <ul style="list-style-type: none"> <li>■ Dim S(1 To 12) As Boolean</li> <li>■ Dim T(1 To 15) As Boolean</li> </ul> In questo caso è stato utilizzato N=12 e M=15. Si ricorda che, nel caso si voglia creare un programma in VB non contenuto in un unico Form, sarà necessario creare un modulo (.BAS) nel quale inserire le precedenti definizioni (come Public invece che come Dim).
PC: linguaggio C	Come per il VB, ma con la difficoltà aggiuntiva di avere array in base 0 invece che 1, e con il tipo di dato booleano non nativo (un booleano, al minimo, consiste in un unsigned character). La definizione più ovvia è la seguente (usando il Visual C): <ul style="list-style-type: none"> <li>■ bool S[12];</li> <li>■ bool T[15];</li> </ul>

:

- 1) **Inizializzazione:** Inizializzazione degli elementi del vettore S<sub>i</sub> degli stati a zero per tutti gli stati tranne quelli considerati come stati iniziali;
- 2) **Verifica Transizioni:** Si testano tutte le condizioni relative alle transizioni T<sub>j</sub> che discendono da stati attivi. Si impone l'elemento del vettore delle transizioni pari ad 1 se le condizioni che attivano la relativa transizione sono verificate;

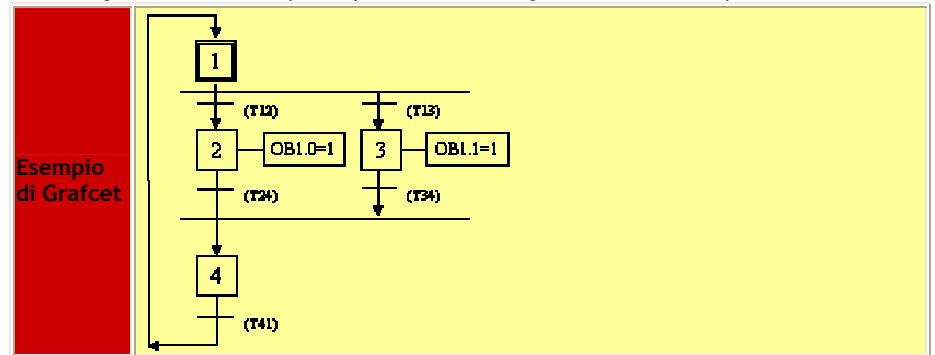
- 3) **Esecuzione:** Tutte le azioni collegate a stati attivi vengono eseguite o vengono lasciate attive (in funzione del tipo di azione);
- 4) **Cambio di Stato:** Si disattiva ogni stato S<sub>i</sub> che è seguito da una transizione T<sub>j</sub> attiva e si attiva ogni stato S<sub>i</sub> che è preceduta da una transizione T<sub>j</sub> attiva;

Delle 4 funzioni solo la prima viene eseguita all'attivazione del processo e dopo ogni reset dell'elaboratore che esegue il controllo. Le altre 3 vengono eseguite in successione ad ogni ciclo di esecuzione dell'elaboratore.

Esempio di Traduzione di un Grafcet

Se è vero che una figura vale più di mille parole, sarà vero anche che, eseguendo direttamente un esempio, sarà possibile comprenderlo con maggiore semplicità la modalità di traduzione di un grafcet.

Dato un grafcet molto semplice, per ora non collegato ad alcun esempio reale:

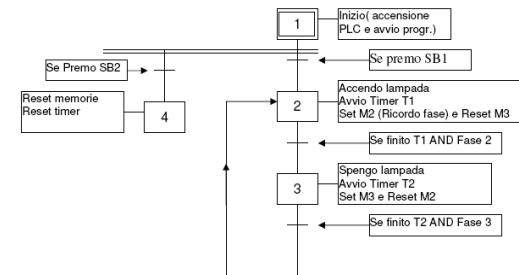


Nel grafcet sono presenti 4 stati e 5 transizioni. In due dei quattro stati sono presenti delle azioni che terminano non appena lo stato non è più attivo. Gli stati collegati ad azioni (2 e 3) sono attivati tramite una *scelta alternativa* e tramite una *convergenza* convergono nello stato 4. Lo stato 1 è lo stato iniziale.

ESEMPLI

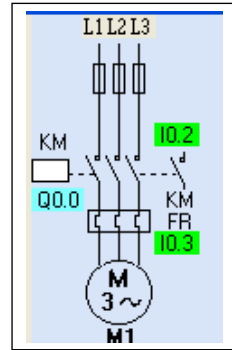
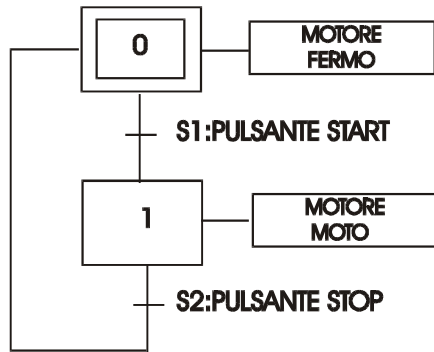
**Esempio 1°: uso di SFC su un problema di automazione**

Una lampada HL1 deve accendersi in modo intermittente dopo la pressione di un pulsante e si arresta dopo la pressione di un altro pulsante. L'intermittenza è di 2 s



### ESEMPIO 2: controllo marcia/arresto di un motore

Occorre gestire l'avviamento e l'arresto di un motore elettrico con due pulsanti (uno per l'avviamento, l'altro per l'arresto).



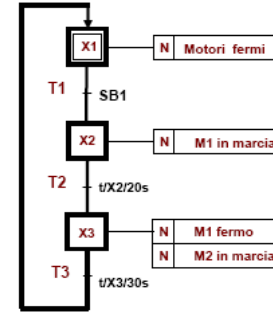
### ESEMPIO 3: controllo motori temporizzato

**Descrizione:** occorre avviare due motori elettrici M1 e M2 secondo la seguente sequenza:

Alla attivazione del pulsante SB1 si avvia M1.

Dopo 20 s si arresta M1 e si avvia M2.

Dopo ulteriori 30 s si arresta anche M2.



All'avviamento il sistema è nella fase X1, cui è associata l'azione 'Motori fermi'.

La transizione T1 è abilitata. All'attivazione del pulsante SB1 avviene lo scatto di T1, che

- disattiva la fase X1
- attiva la fase X2, e pone in marcia il motore M1.

L'attivazione della fase X2

- abilita la transizione T2
- fa partire un timer che dopo 20 s fa scattare la transizione T2.

Lo scatto di T2 provoca

- la disattivazione della fase X2
- l'attivazione della fase X3: si arresta M1 e si pone in marcia M2.

L'attivazione della fase X3

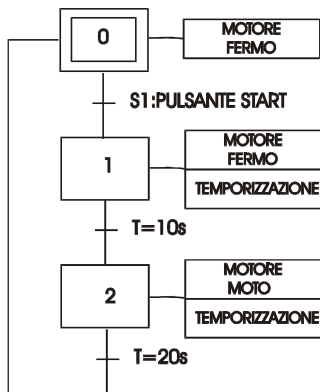
- abilita la transizione T3
- fa partire un timer che dopo 30 s fa scattare T3.

Lo scatto di T3

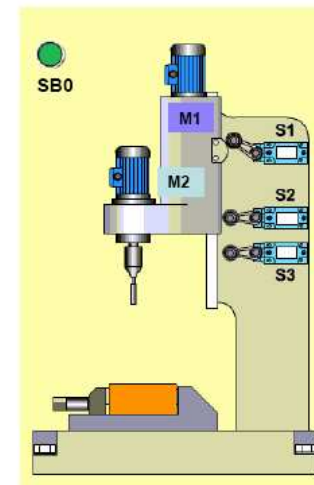
- disattiva X3
- attiva X1: si arrestano entrambi i motori.

### Esempio 2 BIS°: uso di SFC su un problema di automazione

Un motore si deve avviare dopo 10 s che si è premuto il pulsante di start e deve funzionare per altri 20 s e poi fermarsi in modo automatico



### APPLICAZIONE: CONTROLLO TRAPANO AUTOMATICO



#### DESCRIZIONE DELL'AUTOMATISMO

Premendo il pulsante SB0 si alimenta il motore M1.

Il carrello porta utensile scende alla velocità v1.

L'attivazione del finecorsa S2 modifica la velocità di discesa del carrello v2 < v1 e attiva il motore M2.

L'attivazione del finecorsa S3 fa risalire alla velocità v1 il carrello.

Durante la risalita si attiva il finecorsa S2 e si arresta il motore M2.

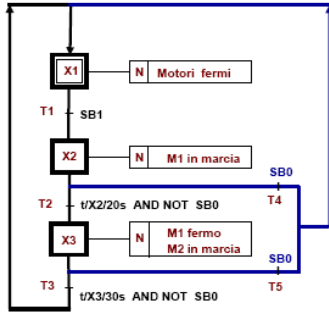
L'attivazione del finecorsa S1 arresta il motore M1.

Progettare il controllo.



**ESEMPIO 5: controllo motori temporizzato con pulsante di arresto**

**Descrizione:** si tratta del controllo relativo all'ESEMPIO 3 in cui è stato aggiunto un pulsante di arresto **SB0**. In qualsiasi momento all'attivazione del pulsante **SB0** i motori devono arrestarsi.



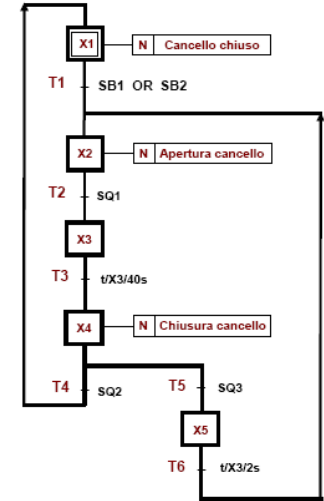
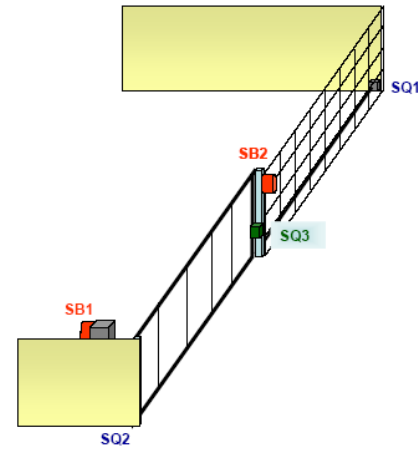
L'attivazione della fase X2 abilita le transizioni T2 e T4.

L'attivazione del pulsante SB0 fa scattare solo la transizione T4 e il processo torna allo stato iniziale X1.

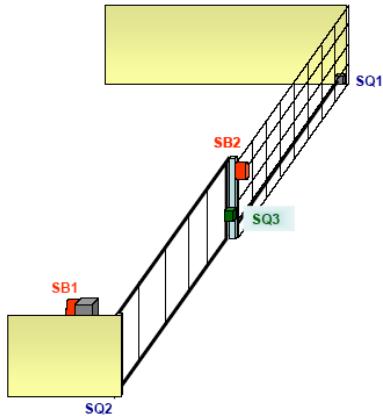
In conclusione si ritorna allo stato X1 in seguito allo scatto di una delle transizioni T3, T4, T5.

**NB:** non è possibile che tra queste transizioni possa presentarsi uno scatto simultaneo.

**ESEMPIO 6: controllo cancello con azionamento motore elettrico**



**ESEMPIO 6: controllo cancello con azionamento motore elettrico**



**Descrizione:**

Due pulsanti **SB1** e **SB2** comandano l'apertura del cancello.

Completata la fase di apertura, con l'attivazione del finecorsa **SQ1**, il cancello resta fermo per 30 s e, successivamente, comincia la fase di chiusura.

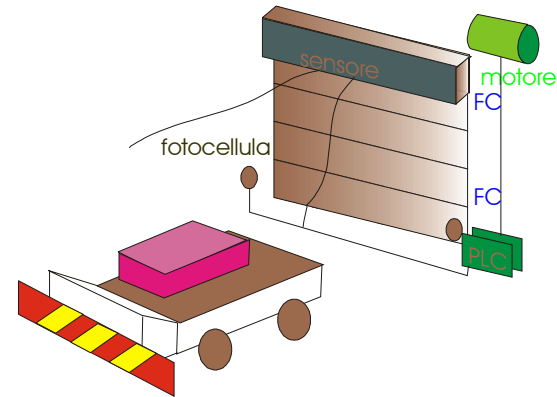
Durante la fase di chiusura la fotocellula **SQ3** controlla che nessun oggetto o persona entri nel raggio d'azione del cancello.

Una attivazione della fotocellula ferma la chiusura e, dopo una pausa di 2 s, ripete l'apertura.

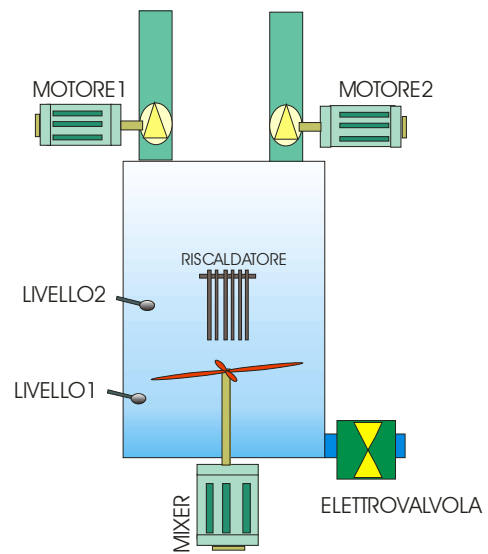
Il completamento della chiusura attiva il finecorsa **SQ2**.

**Apri saracinesca**

Si vuole realizzare un sistema controllato da PLC per l'automatica apertura di una saracinesca quando un veicolo giuge in prossimità della saracinesca e si richiude automaticamente al passaggio segnalato dalla fotocellula BL.



Sono presenti due finecorsa **SQ1** e **SQ2** per segnalare l'apertura della saracinesca e la chiusura



### MIXER INDUSTRIALE

1. PREMENDO UN PULSANTE S1 SI ATTIVA LA POMPA M1
2. RAGGIUNTO IL LIVELLO 1 SI SPEGNE IL MOTORE 1 E SI ACCENDE POMPA M2
3. raggiunto l2 si spegne M2
4. SI ACCENDE UN RISCALDATORE SI SPEGNE DOPO 20 s
5. E CONTEMPORANEAMENTE INTERVIENE IL MOTORE DEL MIXER SI SPEGNE DOPO 30 s IL MIXER
6. AL TERMINE DELLE FASI 3 E 4 SI APRE L'ELETTRVALVOLA.