UNIVERSITA'
DEGLI STUDI
DI BERGAMO

AUTOMAZIONE INDUSTRIALE

Prof. Andrea Cataldo

Anno Accademico 2013/2014

INDICE:

RIFERIMENTI TESTUALI:

- ✓ “PLC e automazione industriale” - P. Chiacchio; Mc Graw Hill
- ✓ “Fondamenti e controlli automatici” – P. Bolzern, R. Scattolini, N. Schiavoni; Mc Graw Hill
- ✓ "Microchip, PIC16F87X Data Sheet, 28/40-Pin 8-Bit CMOS FLASH Microcontrollers"
- ✓ Dispense di “Automazione industriale” (2008/09) - Luigi Piroddi

Introduzione

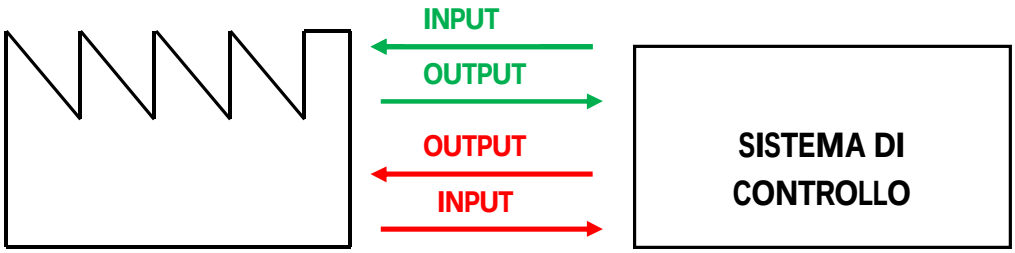
Prima di entrare nel vivo degli argomenti trattati durante il corso è utile fare una piccola introduzione che chiarifichi i concetti base che verranno affrontati.

Automazione: insieme di tecnologie e metodi che gestiscono macchine e processi fisici tramite l'utilizzo di dispositivi, in particolare per mezzo di Sistemi di Controllo. L'automazione, riduce in genere l'intervento umano nelle operazioni più standard e ripetitive ma soprattutto è impiegata nelle situazioni in cui sia richiesta un alto grado di prestazioni e sicurezza. Negli impianti produttivi in genere comporta la riduzione dei tempi e costi di lavorazione, a fronte anche di un aumento della qualità.

Cos'è e com'è fatto un Sistema di Controllo?

Sistema di Controllo automatico: ha l'obiettivo di modificare il comportamento del sistema da controllare. Acquisisce dal sistema informazioni o variabili in ingresso, dette INPUT, e dopo averle opportunamente elaborate, impartisce al sistema sotto controllo delle azioni tramite segnali di uscita chiamate OUTPUT. Il controllo del sistema è affidato ad uno specifico dispositivo, il Controllore, progettato e configurato specificatamente sulla base dei requisiti di funzionamento del sistema da controllare e sulla base del modello fisico-matematico del sistema in esame e sotto controllo.

Per meglio capire i concetti summenzionati, possiamo semplicemente far riferimento ad un impianto o dispositivo da controllare, interfacciato al relativo sistema di controllo.
Dal **punto di vista dell'impianto**, il flusso delle informazioni scambiate tra processo fisico e sistema di controllo è rappresentato dalle frecce di colore verde, in cui gli Input per il processo da controllare sono le azioni che il controllore gli impartisce mentre gli Output rappresentano le variabili di processo che esso rende disponibile al sistema di controllo.
Viceversa, se il punto di vista viene ribaltato, vi saranno degli Input che rappresentano le informazioni sulla stato del sistema da controllare (necessarie al controllore per capire in quale stato il sistema sotto controllo si trova) ed Output che rappresentano le azioni di controllo fornite al processo (frecce rosse).



Normalmente il punto di vista utilizzato è quello del Sistema di Controllo (frecce rosse), quindi per INPUT intenderemo le informazioni acquisite dal processo da controllare (variabili di processo) e che permettono di conoscere lo stato dell'impianto. Gli Output sono invece le azioni che il Sistema di Controllo fornisce all'impianto per portare il sistema controllato nello stato di funzionamento desiderato.

I dispositivi tecnologici che consentono di acquisire gli INPUT sono i sensori o trasduttori e trasformano una grandezza fisica misurata in un segnale interpretabile dal controllore.

I dispositivi che permettono di attuare l'azione di controllo in azioni fisiche sono gli attuatori.

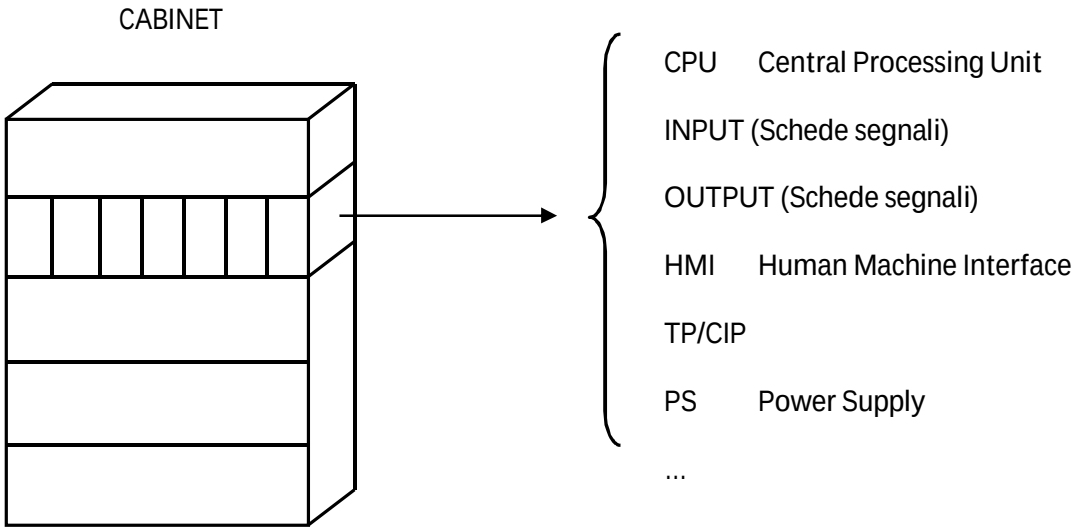
Riferendosi al contesto di questo corso, il controllo si struttura in tre filoni:

- ✓ LOGICO (implementato da PLC - Programmable Logic Controller): utilizzato per sistemi ad eventi discreti ossia laddove sono in gioco informazioni binarie/booleane (0, 1);
- ✓ MODULANTE (implementato da DCS - Distributed Control System): utilizzato per processi in cui le variabili in gioco sono continue nel tempo (ad esempio velocità v, pressione p, portata q, temperatura t, ecc…);
- ✓ MICROCONTROLLORE: (implementato da Micro-controllori) adatti soprattutto per applicazioni poco costose e non critiche dal punto di vista della sicurezza del sistema sotto controllo.

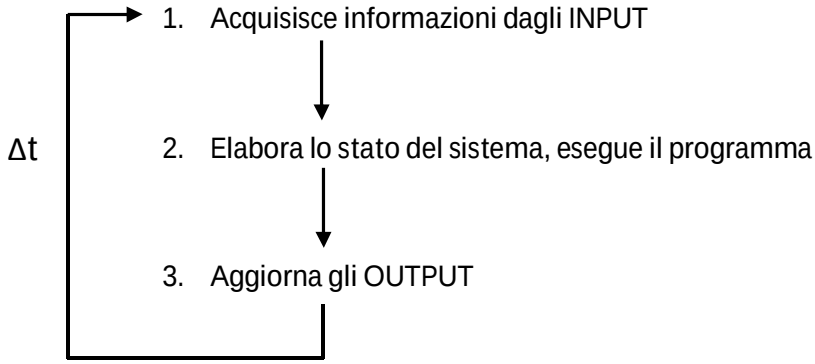
Controllo Logico - PLC

Il PLC (Programmable Logic Controller) o Controllore Logico Programmabile ha in genere una struttura modulare.

Si può pensare al PLC come ad un "armadio" detto CABINET, formato da ripiani (RACK) a loro volta divisi da separatori (SLOT). In ogni ripiano sono quindi contenuti più moduli.



Il PLC, per poter controllare un processo fisico ad eventi discreti, compiete tre precise operazioni:



1. Innanzitutto vengono acquisiti gli INPUT dal sistema sotto controllo al fine di conoscere lo stato del sistema, campionando i segnali forniti dai sensori dislocati sul processo fisico;
2. Successivamente viene elaborato il programma inserito nella CPU del PLC, in modo da elaborare, secondo il programma sw appunto, le informazioni appena acquisite e determinare quindi le azioni di controllo opportune;
3. Le variabili determinate sulla base della fase precedente, vengono mandate in uscita al sistema sotto controllo attraverso le schede fisiche di Output.

Tali operazioni sono eseguite in modo **ciclico** e **deterministico**.

Ciclico: una volta arrivati al termine dell'ultima operazione (3), si riprende nuovamente con il passo 1 poi con il 2 e così via; ovvero la sequenza 1-2-3 viene ripetuta ciclicamente.

Deterministico: il tempo che trascorre tra un ciclo e il successivo è ben determinato, fissato a priori in base alla scelta del PLC. Tale tempo ciclo, Δt, di fatto rappresenta il lasso di tempo che intercorre tra un'acquisizione degli Input e la successiva.

La scelta del Δt è fondamentale per il buon controllo del sistema fisico. Si consideri a titolo di esempio un carrello che viaggia alla velocità "V" e che deve fermarsi quando viene fornito dal sensore di posizione il segnale opportuno. Nel caso peggiore, nel momento in cui il carrello passa davanti al sensore di posizione, il PLC si potrebbe trovare ad eseguire la fase (2) di elaborazione del programma che sta processando le vecchie informazioni di Input, quindi non sta elaborando il fatto di essere giunto in posizione di arresto. Per cui le azioni di controllo impartite al sistema fisico (3) non consentono di arrestare il carrello. Vengono acquisiti i nuovi Input (1) e solo ora il PLC rileva che il segnale di posizione è attivo. Nella fase (2) viene elaborato il programma con le informazioni Input aggiornate. Nella fase (3) vengono aggiornati gli Output e quindi viene fermato il carrello. Complessivamente ci sono volute 5 operazioni da parte del PLC. Se quindi il PLC impiega Δt [s] per compiere le fasi (1) - (2) - (3), significa che impiegherà circa 5/3 Δt nel caso peggiore per fermare il carrello. In questo tempo il carrello avrà percorso uno spazio, oltre la posizione di arresto, pari a V* Δt [m]. Se per esempio il carrello viaggia alla velocità di 10 [m/s] ed il tempo ciclo Δt del PLC è di 10 [ms], il carrello percorre oltre la posizione di arresto uno spazio di 0,1 [m]. Si intuisce che per ridurre tale spazio, e ciò dipende dai requisiti del sistema da controllare, è necessario scegliere un PLC con tempo ciclo Δt più piccolo.

Il PLC, essendo un dispositivo elettronico Programmabile, necessita di linguaggi di programmazione specifici.

Lo standard internazionale IEC 61131-3 definisce cinque linguaggi di programmazione standard per il PLC. Questi sono:

1. LD (Ladder diagram)
2. SFC (Sequential function chart)
3. FBD (Function block diagram)

} LINGUAGGI GRAFICI

4. IL (Instruction list)
5. ST (Structured text)

} LINGUAGGI TESTUALI

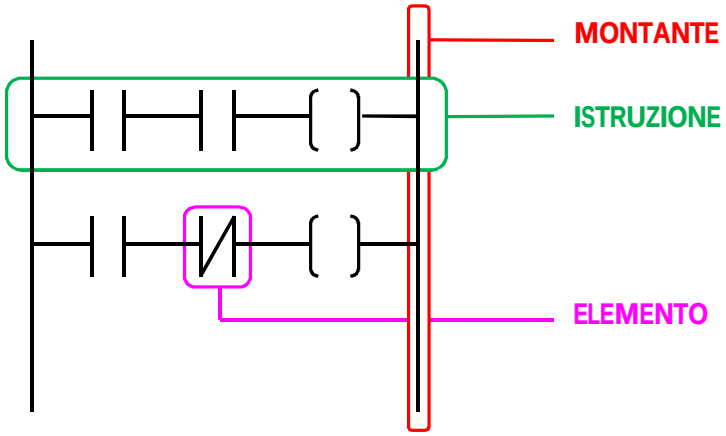
I primi tre linguaggi (SFC, LD, FBD) sono grafici, cioè che fanno uso di simboli, i restanti due (IL e ST) sono linguaggi testuali, cioè fanno uso di parole.

In questo corso vengono analizzati due di questi linguaggi: il Ladder Diagram (più vecchio ma anche più diffuso) e l'SFC.

Ladder Diagram

Il linguaggio LD (Ladder Diagram) è un Linguaggio a Contatti detto anche Diagramma a Scala o a Pioli e presenta delle regole di sintassi (scrittura) ben precise:

- ✓ Il programma va scritto all'interno di due linee verticali, detti montanti;
- ✓ Tra i due montanti vengono scritte le istruzioni;
- ✓ Ogni "Piolo" che unisce i due montanti costituisce un'istruzione (rung);
- ✓ Un'istruzione è formata da più elementi grafici;



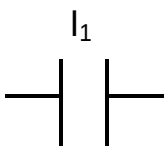
Il PLC, ogni volta che elabora il programma, esegue tutte le istruzioni nel programma, partendo dalla prima in alto all'ultima in basso; ciascuna istruzione è interpretata leggendola da sinistra verso destra.

Di seguito si analizzano nello specifico gli elementi grafici di cui si compone il linguaggio LD standard, come si comportano e quando è opportuno il loro utilizzo per la programmazione di un PLC.

Elementi grafici del LADDER:

Ad ogni elemento va sempre associata una variabile booleana (di INPUT o di OUTPUT, la prima sarà qui indicata con I_n, la seconda con O_n).

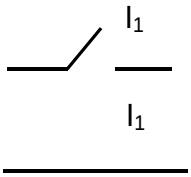
- ✓ **Contatto normalmente aperto**



I_1 è una variabile di INPUT booleana/binaria.

Assume valore [0,1]

Per comprenderne il significato si consideri un'analogia con il contatto elettrico:

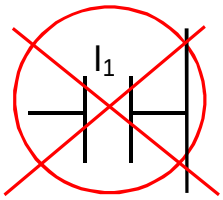


Se I_1=0 allora **non c'è** continuità attraverso il contatto.

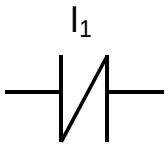
Se I_1=1 allora **c'è** continuità attraverso il contatto.

Il contatto non può mai essere direttamente collegato al montante di destra.

Il contatto può non essere direttamente collegato al montante sinistro.



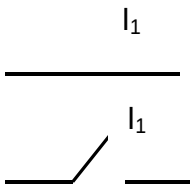
- ✓ **Contatto normalmente chiuso**



I_1 è una variabile di INPUT booleana/binaria.

Assume valore [0,1].

Per comprenderne il significato si consideri un'analogia con il contatto elettrico:

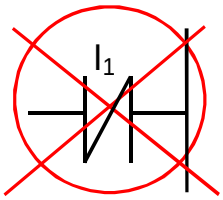


Se I_1=0 allora **c'è** continuità attraverso il contatto.

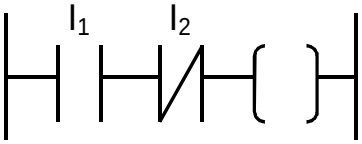
Se I_1=1 allora **non c'è** continuità attraverso il contatto.

Il contatto non può mai essere direttamente collegato al montante di destra.

Il contatto può non essere direttamente collegato al montante sinistro.



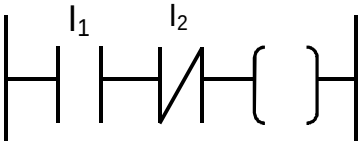
Esempio:



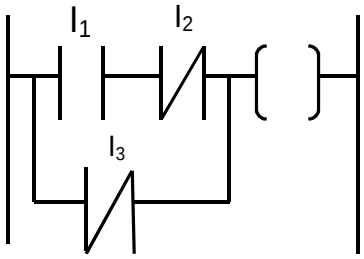
I contatti sono associati alla stessa variabile I_1. Non ci sarà mai continuità attraverso i contatti perché quando uno è aperto l'altro è chiuso.

$$I_1 \cdot \bar{I}_1 = 0$$

Due contatti in serie equivalgono all'espressione/funzione logica AND o (·).



I contatti sono associati a due variabili distinte I_1 e I_2.
Ci sarà continuità attraverso i contatti se I_1=1 e I_2=0.



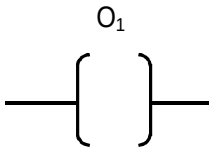
Nel ramo in alto ci sarà continuità attraverso i contatti se I_1=1 e I_2=0. Percorrendo il ramo in parallelo formato dal contatto I_3 , potrebbe invece esserci continuità nel caso in cui I_3= 0.

$(I_1 \bar{I}_2) + \bar{I}_3$

Due contatti in parallelo equivalgono all'espressione/funzione logica OR o (+).

Attenzione: il PLC interpreta sempre tutta l'istruzione fino in fondo, anche se ad un certo punto dell'istruzione non c'è più continuità. Ciò accade in quanto dovrà essere definito il valore della variabile di Output, come si vedrà più avanti.

✓ **Bobina**



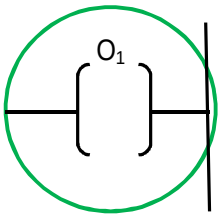
O_1 è una variabile di OUTPUT booleana/binaria.

Assume valore [0,1].

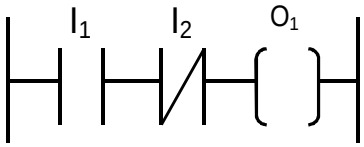
Se **c'è** continuità a sinistra della bobina allora O_1=1.

Se **non c'è** continuità a sinistra della bobina allora O_1=0.

La bobina deve sempre essere direttamente collegata al montante di destra.

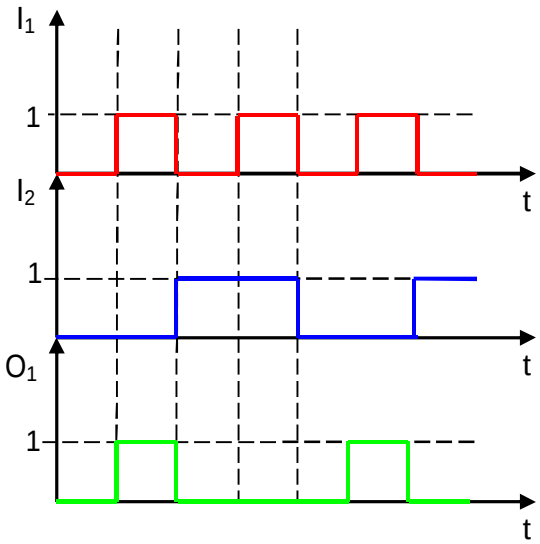


Esempio:

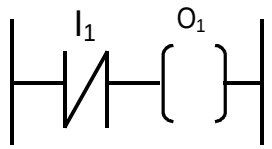


$I_1 \cdot \bar{I}_2 = O_1$

AND		
I_1	$\bar{I}_2$	O_1
0	0	0
0	1	0
1	0	1
1	1	0



La variabile O_1 ha valore 1 solo quando I_1=1 e I_2=0.

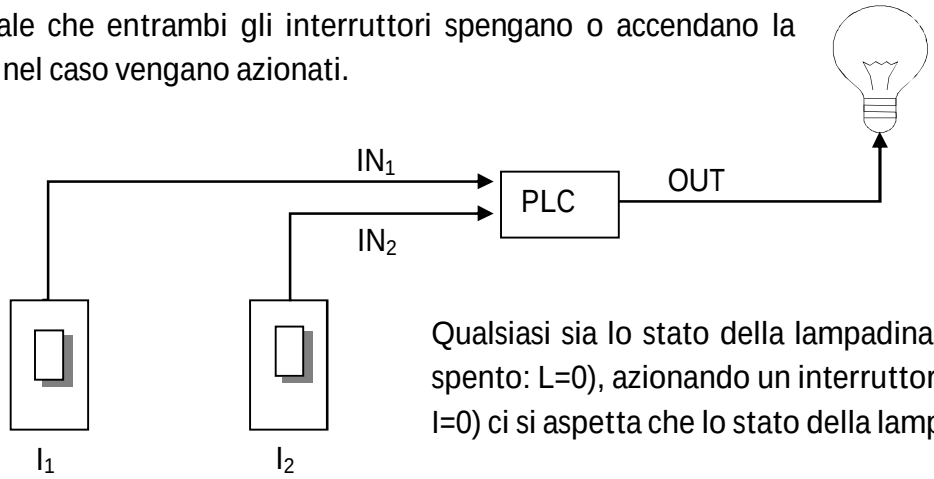


$\bar{I}_1 = O_1$

NOT	
I_1	O_1
0	1
1	0

○ **Esempio: Lampadina**

Si consideri il caso di una lampadina collegata a due interruttori I_1 e I_2. Si vuole gestire il funzionamento di tale lampadina per mezzo di un PLC, in modo tale che entrambi gli interruttori spengano o accendano la lampadina nel caso vengano azionati.



Qualsiasi sia lo stato della lampadina (acceso: L=1, spento: L=0), azionando un interruttore (on: I=1, off: I=0) ci si aspetta che lo stato della lampadina cambi.

Inizialmente entrambi gli interruttori sono nello stato di "off": I1=I2=0, di conseguenza anche la lampadina è spenta.

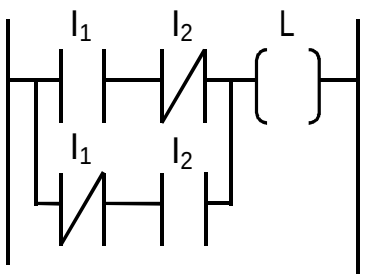
Azionando il primo interruttore, ad esempio, otterremo I1=1 e I2=0: la lampadina è accesa. Per ottenere un valore alto (1) in uscita, in presenza di I1=1 e I2=0 è necessario mettere in serie due contatti, il primo normalmente aperto e il secondo normalmente chiuso.

Analogo ragionamento vale nel caso I1=0 e I2=1.

Inoltre possiamo dedurre che le due serie di contatti dovranno essere in parallelo tra loro, per contemplare entrambi i casi.

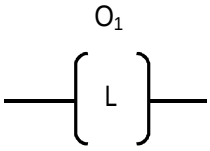
Infine notiamo che nel caso in cui i due interruttori si trovino nello stesso stato, on od off che sia, non vi è continuità, dunque la lampadina sarà spenta.

Graficamente:



Questa configurazione risulta essere l'operazione logica XOR.

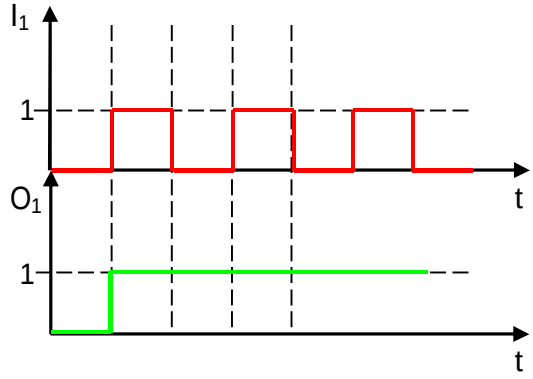
- ✓ **Bobina Latch**



O_1 è una variabile di OUTPUT booleana/binaria.

Assume valore [0,1].

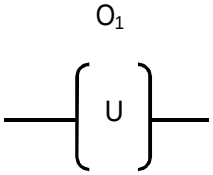
Se c'è continuità a sinistra della bobina allora O_1=1. Se viene a mancare la continuità a sinistra della bobina O_1=1 ossia O_1 mantiene il valore logico Alto.



Es. Contatto I1 pilota la bobina Latch O1

Quando viene a mancare la continuità a sinistra della bobina latch, il valore della variabile O_1 rimane alto (1).

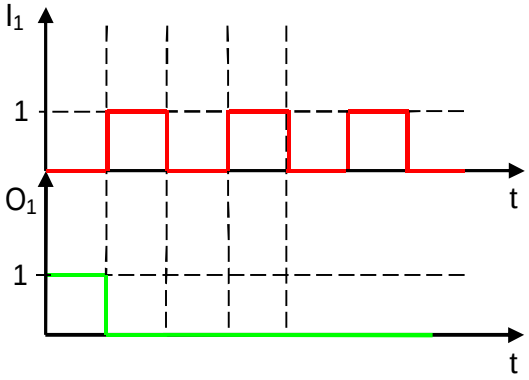
- ✓ **Bobina Unlatch**



O_1 è una variabile di OUTPUT booleana/binaria.

Assume valore [0,1].

Se c'è continuità a sinistra della bobina allora O_1=0. Se viene a mancare la continuità a sinistra della bobina allora il valore di O_1 continua ad essere pari a 0.



Es. Contatto I1 pilota la bobina Unlatch O1

Quando viene a mancare la continuità a sinistra della bobina (I_1= 0) il valore della variabile O_1 rimane pari a 0.

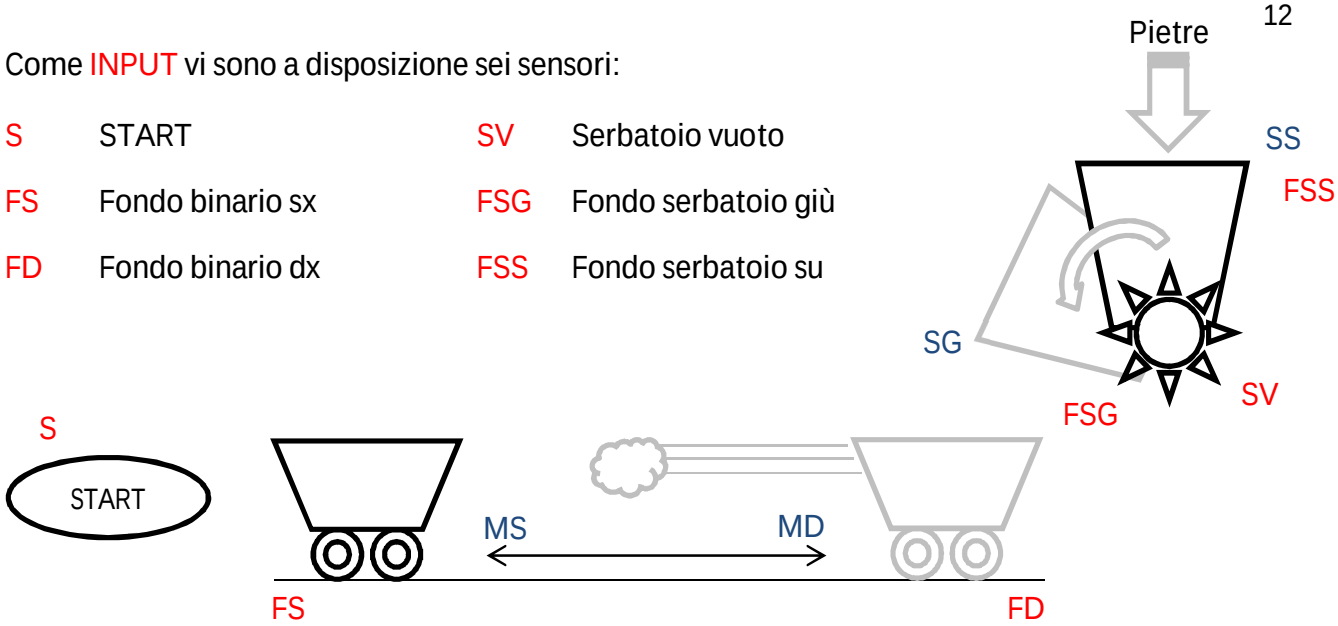
- ○ **Esempio: Carrello e Serbatoio**

Si consideri un sistema di trasporto di pietre per mezzo di un carrello.

L'operatore determina l'inizio del ciclo tramite un apposito pulsante START. Il carrello percorre per intero il binario da sinistra a destra e si arresta in attesa di essere caricato. Le pietre, dopo essersi accumulate in un serbatoio, vengono meccanicamente trasferite nel carrello, il quale deve automaticamente muoversi lungo il binario da destra a sinistra.

Come INPUT vi sono a disposizione sei sensori:

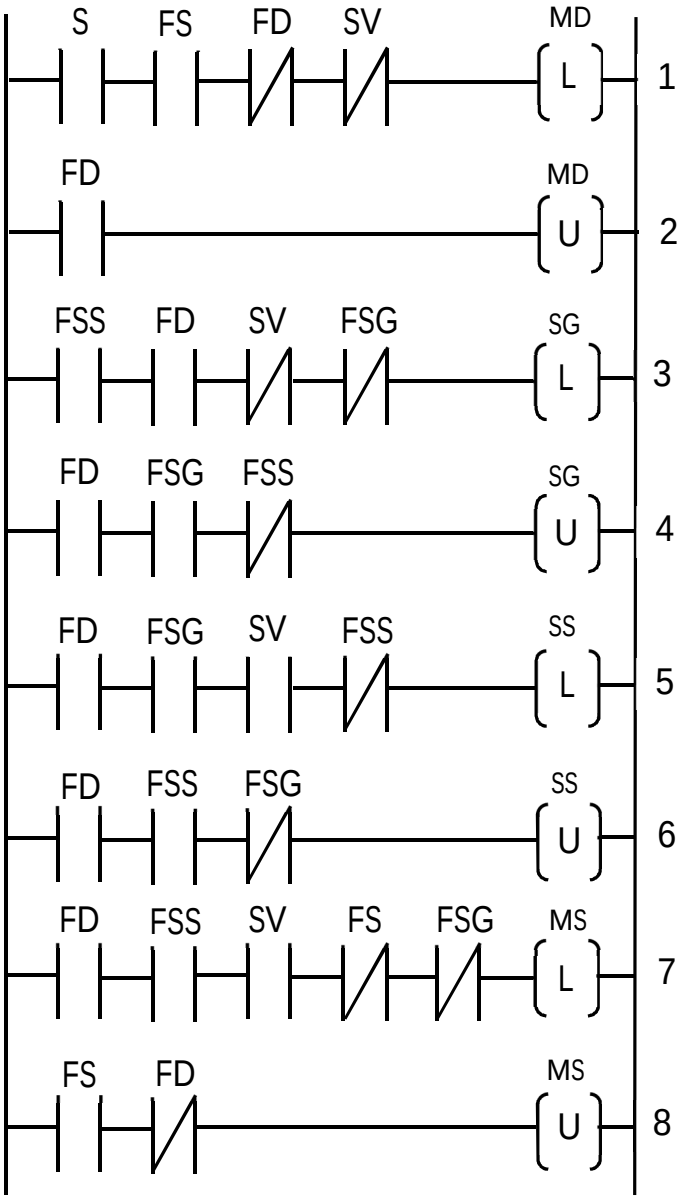
S	START	SV	Serbatoio vuoto
FS	Fondo binario sx	FSG	Fondo serbatoio giù
FD	Fondo binario dx	FSS	Fondo serbatoio su



Come OUTPUT troviamo:

MD	Motore carrello verso dx	SG	Motore serbatoio in giù
MS	Motore carrello verso sx	SS	Motore serbatoio in su

Ladder:



Per azionare il motore verso dx, l'operatore aziona il pulsante START (S=1). FS e FD sono i sensori che garantiscono che il carrello si trovi a sx del binario. SV ci assicura che il serbatoio non sia vuoto. Utilizzo una bobina ritenuta.

Il carrello si muove verso dx, fino a fine corsa dx, quando FD=1. Il motore verso dx si spegne tramite una bobina reset.

Se il carrello è a dx, il serbatoio è alzato, e quindi non abbassato, e non vuoto, si abbassa il serbatoio.

Ci si assicura che il carrello sia a dx; quando il serbatoio si abbassa fino a fine corsa (quindi non è alzato), si spenge il motore che abbassa il serbatoio.

Per rialzare il serbatoio bisogna aspettare che sia vuoto, inoltre ci si assicura che il carrello sia a destra e che il serbatoio sia ancora abbassato (e non alzato).

Il motore "serbatoio in su" si spegne quando il serbatoio è alzato (e non abbassato); il carrello è a dx.

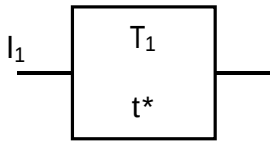
Si accende il motore che del carrello verso sx solo quando, essendo il carrello a dx (e non a sx), il serbatoio è vuoto e alzato (e non abbassato).

Il motore verso sx si spegne una volta che il carrello arriva al fondo sinistro (e non destro).

Osservazioni:

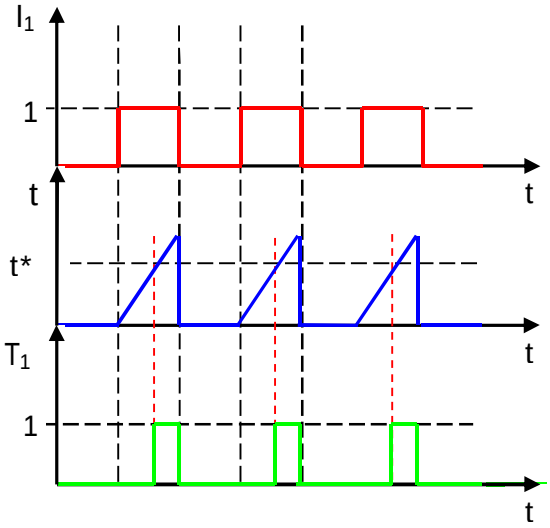
- Nell'istruzione 7 si suppone che il serbatoio rimanga vuoto per un certo tempo.
- Si ricorda che tutte le variabili utilizzate sono booleane: esse ci indicano SE il carrello è a sx/dx, non QUANTO è vicino al fondo sx/dx.
- La ridondanza (x es. specificare sia FS=1, sia FD=0) rafforza la sicurezza, ad esempio in caso di rottura di qualche sensore. Nel precedente esempio, per comprendere meglio, le prime istruzioni sono più "semplici" mentre le ultime presentano numerose ridondanze.

✓ Temporizzatore semplice



Se a sinistra del temporizzatore c'è continuità, esso inizia a contare il tempo t.

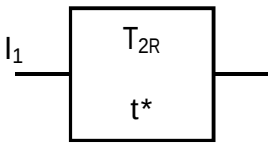
Se t >= t* allora T_1 =1, con T1 variabile booleana associata.



La variabile t cresce nel tempo solo se I_1 = 1. Dopo che t supera una certa soglia (t*), T_1 assume il valore 1 finchè I_1 non torna a 0, resettando in tal caso sia t che T1.

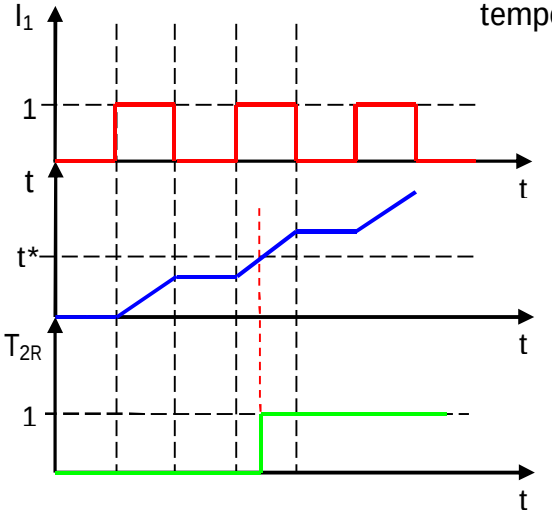
Il Temporizzatore deve sempre essere direttamente collegato al montante di destra.

✓ Temporizzatore a ritenuta

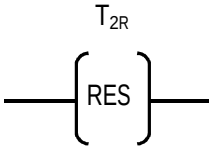


Alla prima continuità a sinistra del temporizzatore, inizia a contare il tempo t.

Se t >= t* allora T_{2R} =1, ma a differenza del temporizzatore semplice, viene conservato il valore del conteggio di t anche se la variabile in ingresso I_1 fa cadere la continuità alla sinistra del temporizzatore.

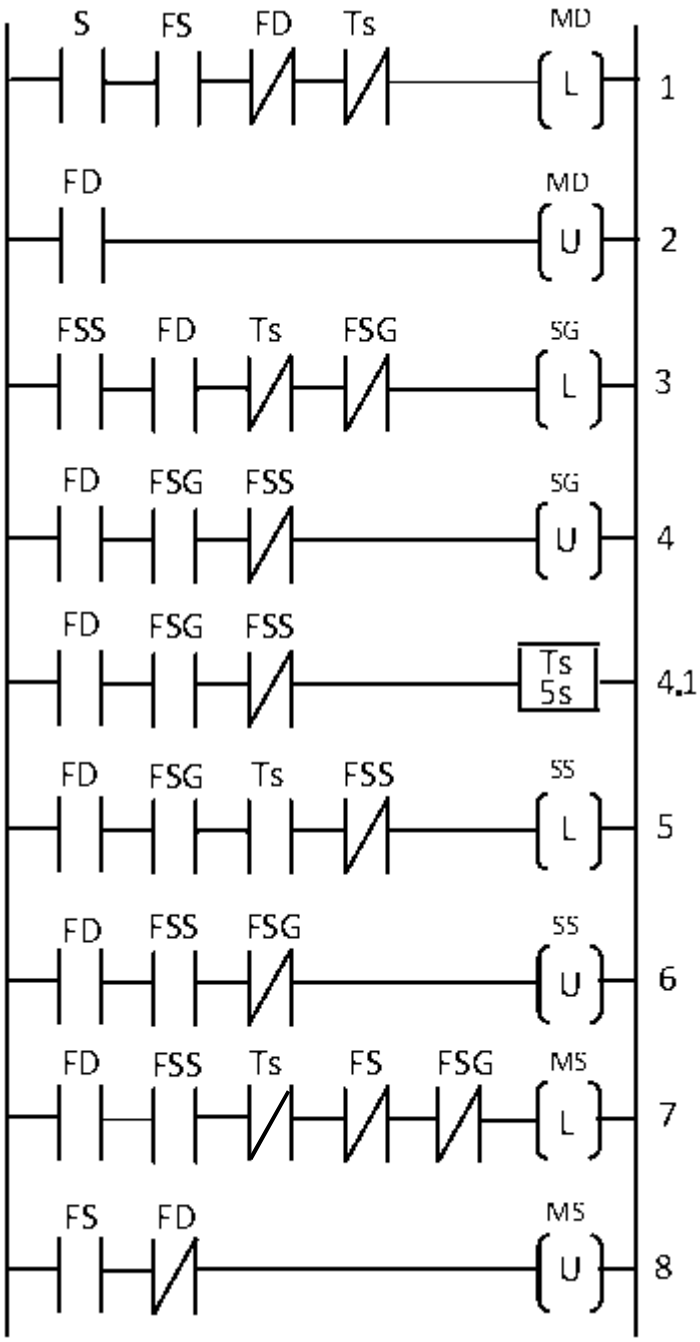


Il Temporizzatore a ritenuta per essere azzerato ha bisogno di un'operazione di RESET che azzera il conteggio t e la relativa variabile booleana associata T_{2R}



Si consideri nuovamente l'esempio del carrello: si ipotizza di eliminare in sensore di "serbatoio vuoto". Si suppone che il serbatoio impieghi 5 secondi, al massimo e nel peggiore dei casi, a svuotarsi completamente.

Per fare ciò si aggiunge un temporizzatore semplice innestando un ramo dopo l'istruzione 4:



In questo caso la variabile SV può semplicemente venire sostituita da Ts, proprio perché Ts=1 solo quando si suppone che il serbatoio sia vuoto.

Una volta che il serbatoio è completamente abbassato, si lasciano passare 5 secondi prima di alzarlo nuovamente.

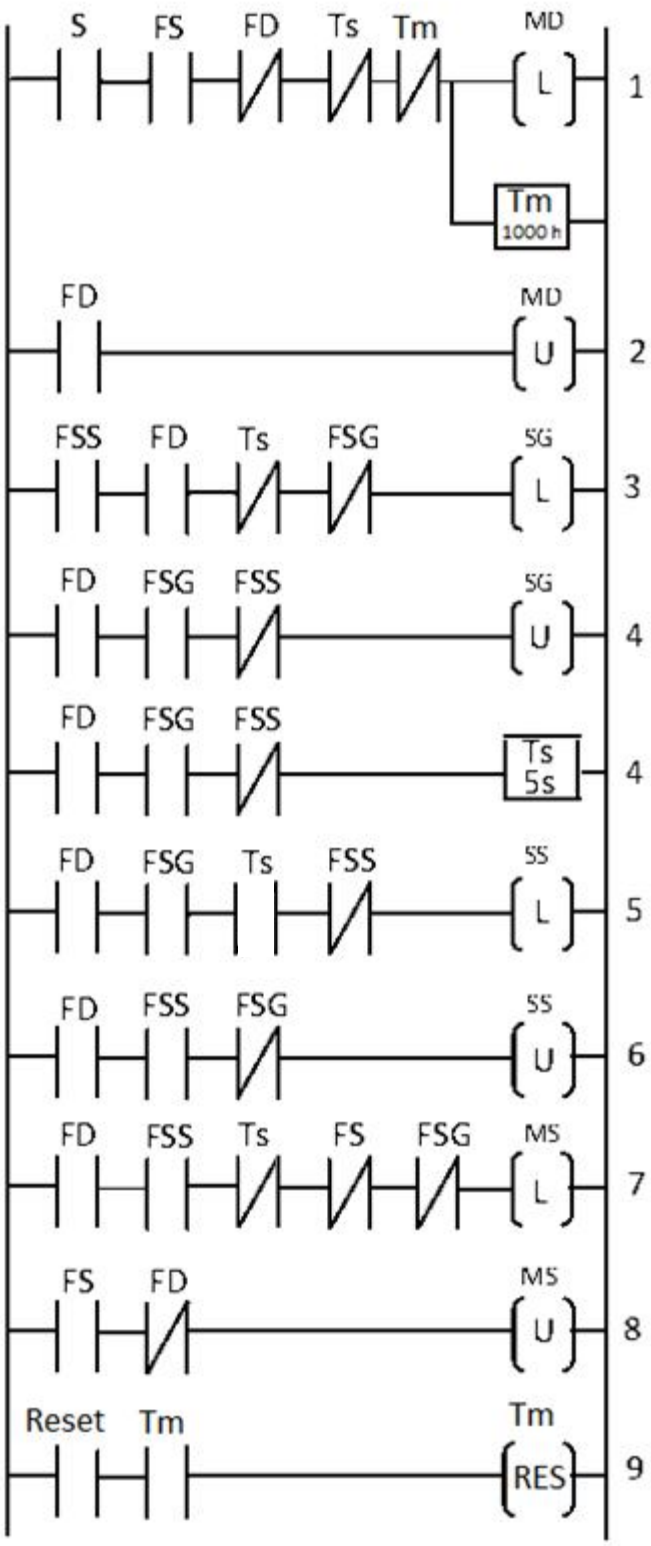
Ts=1 solo dopo 5 secondi in cui a sinistra ha una continuità

Ts=0 poiché viene a mancare la continuità prima del temporizzatore.

Nell'istruzione 7 Ts deve essere associato a un contatto normalmente chiuso, a differenza dell'esempio precedente.

È fondamentale capire che il PLC esegue tutto il codice come se fosse un blocco unico, quindi non possiamo pensare che durante la stessa esecuzione Ts passi dal valore logico di 1 della riga 5 al valore logico 0 richiesto alla riga sette. Durante l'esecuzione del codice in cui si ha Ts=1, tutte le righe di codice che richiedono il valore opposto saranno interpretate ed eseguite ma non produrranno output. Al ciclo di esecuzione successivo il PLC noterà che alla riga cinque non vale più FSG=1, con il conseguente azzeramento di Ts che consentirà il latch su MS.

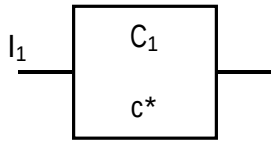
Valutiamo ora il caso di voler imporre un limite di ore prima di uno stop per manutenzione. Per fare ciò è necessario utilizzare un temporizzatore a ritenuta Tm che faccio partire non appena il carrello entra in movimento.



Come si può vedere dal ramo 1 si ha la partenza del temporizzazione contemporanea all'inizio del movimento. Aggiungiamo un vincolo di sicurezza su Tm negato per garantire che la macchina non possa partire una volta scattato il temporizzatore.

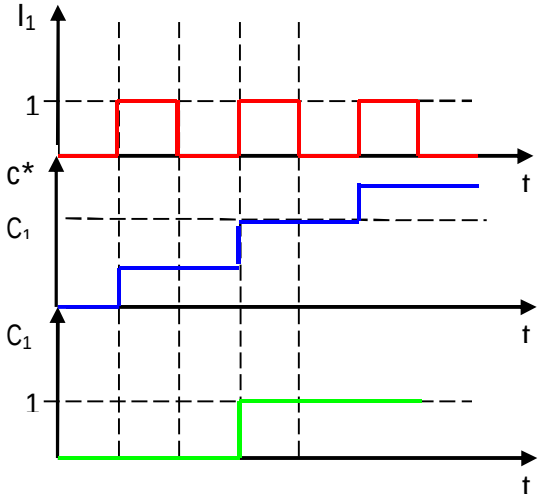
L'ultima istruzione è fondamentale ed è l'operazione di reset del temporizzatore a ritenuta. Il contatto Tm impedisce all'operatore di resettare la variabile Tm prima che il temporizzatore scatti.

✓ Contatore

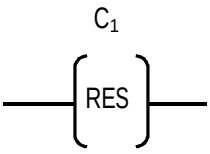


Conta quante volte, tramite la variabile in ingresso I_1 c, si presenta la continuità, cioè quando I1 passa da 0 a 1.

Se c >= c* allora C_1 assume il valore 1.

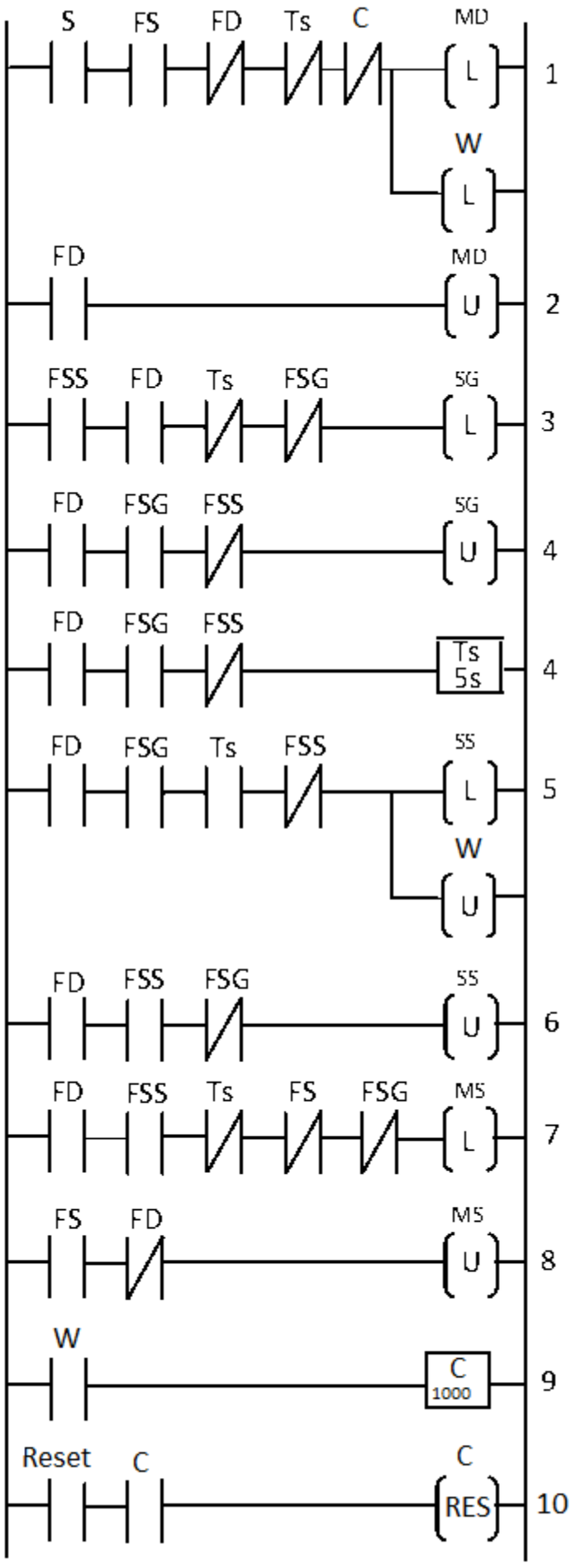


Il contatore per essere azzerato ha bisogno di un'operazione di RESET che azzera il conteggio c e la variabile booleana C_1.



Il Contatore deve sempre essere direttamente collegato al montante di destra.

Rivediamo ora come sarebbe possibile interpretare l'esempio della manutenzione periodica basando il vincolo non sulle ore di lavoro ma bensì sul numero di utilizzi.
Per avere una maggiore sicurezza uso una variabile interna W che acquista valore 1 all'accensione.
L'uso di una variabile interna come filtro impedisce che il conteggio aumenti se l'addetto preme più volte il pulsante di start.



La variabile locale W scatta all'accensione del motore di destra e rimane al valore logico 1 fino all'unlatch.

L'unlatch nel programma potrebbe trovarsi in ogni punto ad eccezione dell'ultimo ramo, perché in questo caso entrerebbe in conflitto con la prima istruzione. Pertanto svolgiamo l'operazione nel ramo 5, dove non crea alcun problema.

Il contatore viene azionato sul fronte di salita di W. Data la non sequenzialità del ladder, questo ramo potrebbe trovarsi in qualsiasi altro punto del programma.

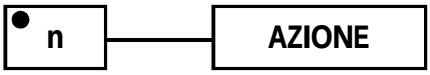
Sequential Function Chart

L'SFC è un altro dei linguaggi standard precedentemente elencati per la programmazione del PLC, più recente e di più semplice scrittura e comprensione.

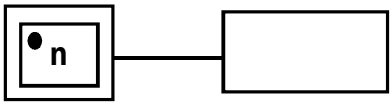
La principale differenza con il linguaggio Ladder sta nell'interpretazione del programma scritto. Mentre nel Ladder Diagram viene scandita l'intera sequenza di istruzioni ogni volta che il PLC deve elaborare il programma stesso, nel SFC tutto quello che succede a monte di una fase attiva ed a valle della transizione abilitata non incide sull'evoluzione del sistema automatizzato. Ciò facilita la comprensione del programma e la relativa progettazione da parte del controllista.
Come per il Ladder si analizziamo gli elementi costitutivi di questo linguaggio.

Elementi grafici del SFC:

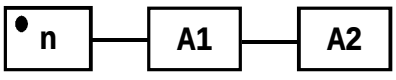
1. Fase o passo



E' indicata con un rettangolo e un numero, non necessariamente, progressivo (n).
Ad ogni fase possono essere associate una o più azioni.
Se la fase è attiva, viene contrassegnata da un pallino, viceversa non lo è.
La fase attiva scatena l'azione associata.
Durante l'esecuzione più fasi possono essere contemporaneamente attive.



La fase iniziale, quella da cui parte l'esecuzione del programma, viene contraddistinta da un doppio rettangolo. Ad essa possono essere associate azioni.



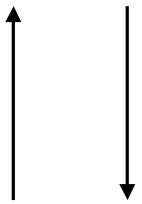
Se ad una fase sono associate più azioni, esse vengono indicate in sequenza ma la relativa esecuzione è contemporanea.

2. Transizione

Consente al programma di evolvere cioè di passare da una o più fasi a monte ad una o più fasi a valle.

Alla transizione T_n è associata una condizione che permette alla transizione, se verificata, di essere superata.

3. Arco orientato

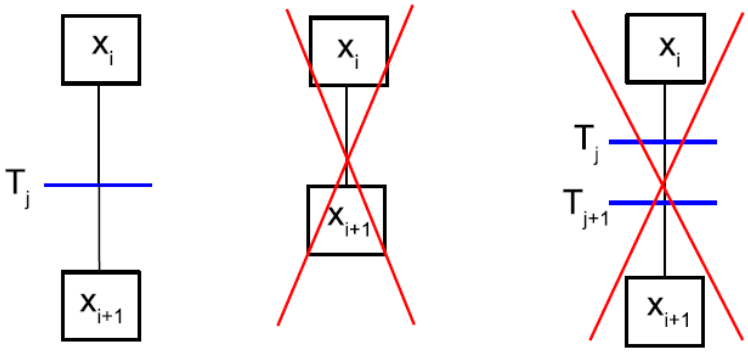


E' il segmento orientato da una freccia che indica il verso di evoluzione del programma.

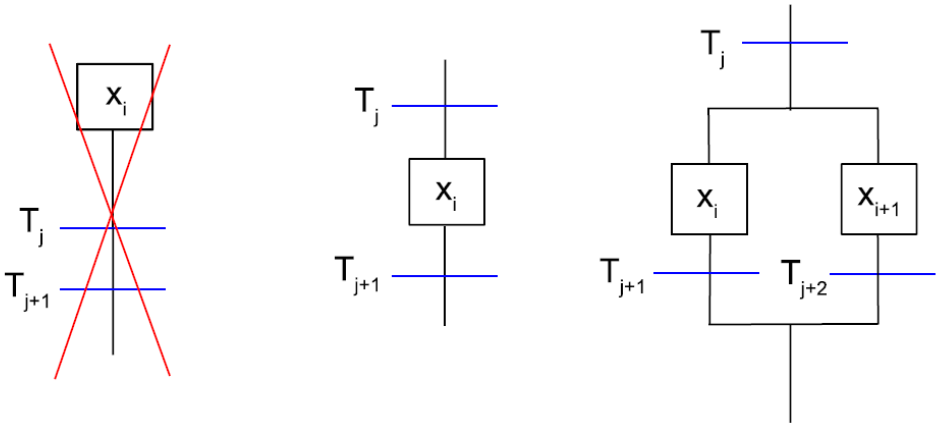
La freccia è necessaria se l'arco va dal basso verso l'alto, mentre si può tralasciare nel caso opposto.

Regole di costruzione:

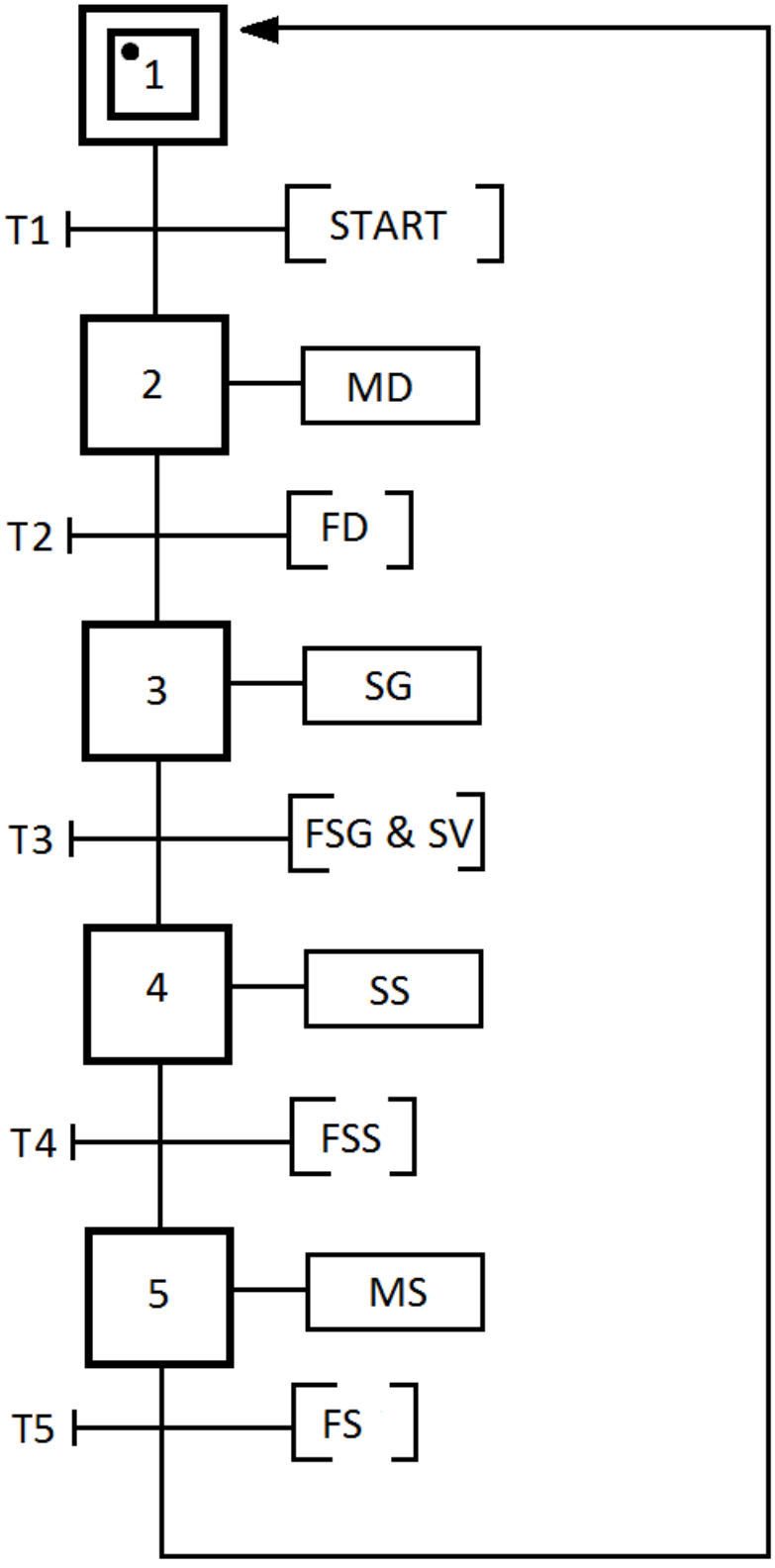
1. Tra due Fasi esiste sempre una e una sola Transizione.



2. Tra due Transizioni esiste almeno una Fase.



Rivediamo ora in logica SFC come apparirebbe l'esempio del carrello



Si nota subito come questo linguaggio grafico risulti molto più intuitivo e comprensibile del Ladder.

L'SFC infatti viene usato per poter programmare più facilmente in Ladder.

Dopo una prima fase di start, in sequenza, avvengono le azioni sopra riportate (viene azionato il motore destro, il serbatoio si abbassa, ecc.) in corrispondenza di certe condizioni (start, serbatoio vuoto, ecc.).

Alla transizione 3 è associata una condizione che include due variabili. Equivalente sarebbe stato il caso di due transizioni, entrambe associate ad una condizione con una sola variabile, intervallate da una fase senza un'azione associata.

Regole di evoluzione:

Una transizione è superabile quando:

1. Tutte le fasi a monte sono attive (in tal caso si dice che la Transizione è abilitata).

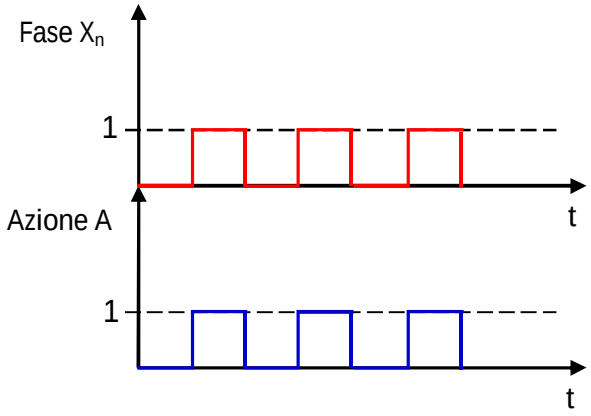
2. La condizione associata alla transizione è vera.

Se le condizioni 1 e 2 sono entrambe vere, allora la transizione viene istantaneamente superata. A questo punto si disattivano tutte le fasi a monte e si attivano tutte quelle a valle.

NB: le regole sopracitate sono semantiche, non sintattiche!

Tipi di Azioni:

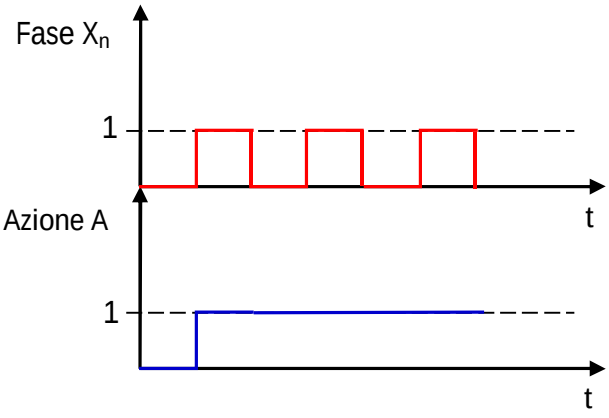
1. Azione continua



Allorquando la fase è attiva (valore del marker pari a 1) l'azione è esercitata; quando la fase è disattiva, l'azione non viene più eseguita.

Quindi l'azione viene eseguita per tutta la durata di tempo in cui la fase associata è attiva.

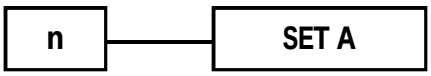
2. Azione di set



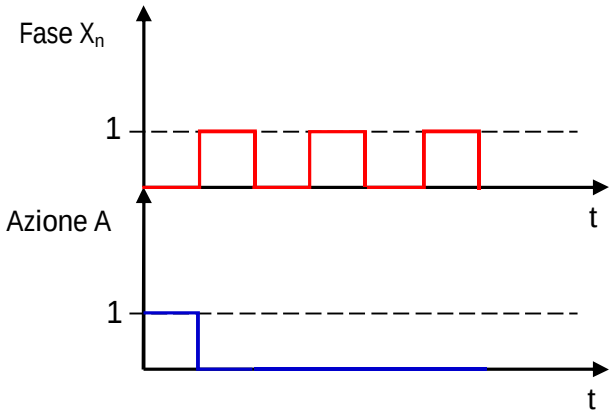
Il comportamento è simile alla bobina latch per il linguaggio Ladder.

Quando la fase si attiva (valore marker pari a 1) l'azione viene esercitata e rimane tale per tutto il resto del tempo.

Si indica con:

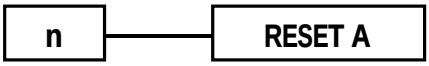


3. Azione di reset

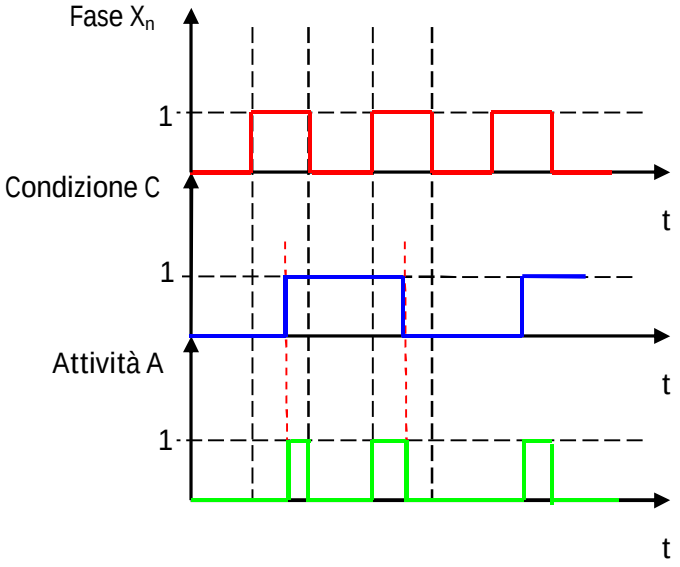


Il comportamento è simile alla bobina unlatch per il Ladder e opposto dell'azione di set.
Quando la fase Xn si attiva (valore marker pari a 1) l'azione cessa di essere eseguita e rimane tale per tutto il resto del tempo.

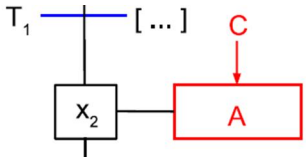
Si indica con:



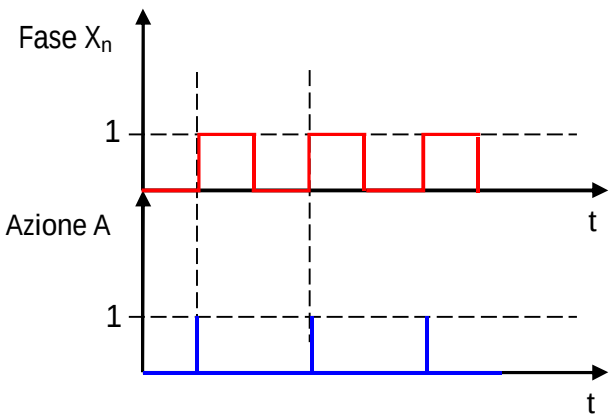
4. Azione condizionata



L'azione viene eseguita solo se la fase è attiva e se contemporaneamente è verificata la condizione esplicitata.
Se la fase è disattiva, l'azione sicuramente non viene eseguita; se la fase è attiva ma la condizione non è vera (valore logico della condizione pari a 0) l'azione ancora una volta non viene eseguita.



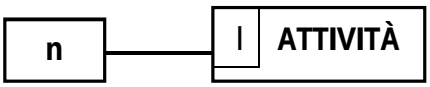
5. Azione impulsiva



Quando la fase diventa attiva (passaggio tra il fronte basso e il fronte alto del marker associato), l'azione viene eseguita per la durata di un impulso, cioè solo nell'istante in cui si attiva la fase stessa.

Questo tipo di comando è utile per incrementare un conteggio (come se fosse un contatore di eventi).

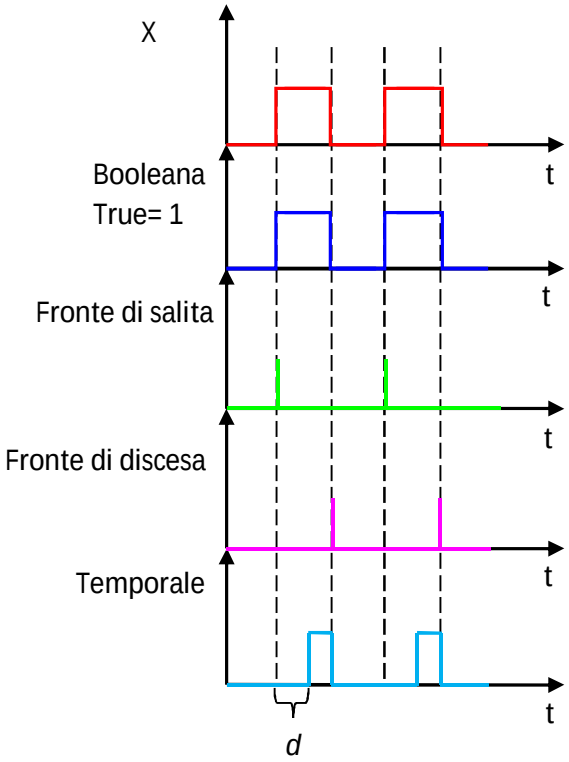
Si indica con:



Tipi di Variabili:

I tipi di variabili utilizzate nel linguaggio SFC ed associate alle condizioni delle transizioni sono:

1. Booleana o binaria: [0,1]
2. A fronte di salita o di discesa: [↑x,↓x]
3. Temporali: [t|X_n|*d*]



Quando la variabile booleana è vera o pari a 1 significa che la condizione associata alla transizione è vera.

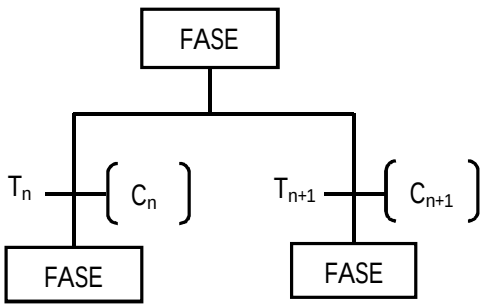
La variabile a fronte di salita è vera solo per la durata di un impulso in corrispondenza del fronte di salita della variabile booleana associata X.

La variabile a fronte di discesa è vera solo per la durata di un impulso in corrispondenza del fronte di discesa della variabile booleana associata X.

La variabile temporale si può assimilare alla variabile booleana associata ad un temporizzatore semplice nel Ladder. Essa inizia a contare dopo che la variabile booleana ad essa associata X è diventata pari a 1. Trascorso il tempo *d*, la variabile temporale diventa vera.
Quando la fase si disattiva la variabile temporale si resetta. Tali variabili vengono in genere utilizzate per per fissare la durata di un'azione (temporizzare la fase). Attenzione: non ha senso temporizzare un'azione di Set, né un'azione Impulsiva.

Costrutti in SFC:

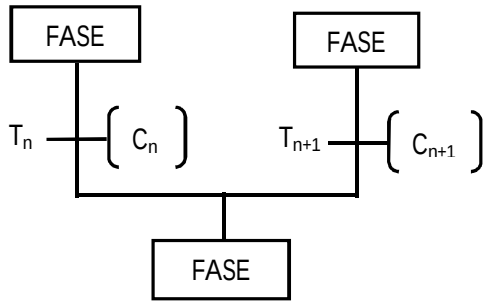
1. Scelta



T_n: transizione n
C_n: condizione n

Le condizioni devono essere mutuo-esclusive (o C_n o C_{n+1}), non entrambe contemporaneamente. In tal modo viene percorso solo uno dei due rami.

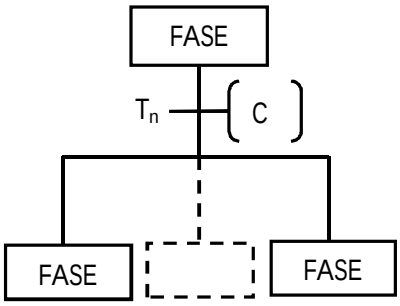
2. Convergenza



T_n: transizione n
C_n: condizione n

Ogni volta che viene effettuata una scelta, questa deve essere chiusa con una convergenza. I due rami si riuniscono in un'unica sequenza.

3. Parallelismo

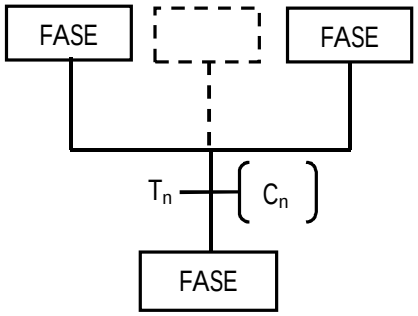


Costrutto duale alla scelta.

A valle di una transizione si attivano due o più fasi contemporaneamente poiché sottostanno ad un'unica condizione C_n.

Tale costrutto viene in genere utilizzato per svolgere rami del programma SFC contemporaneamente.

4. Sincronizzazione

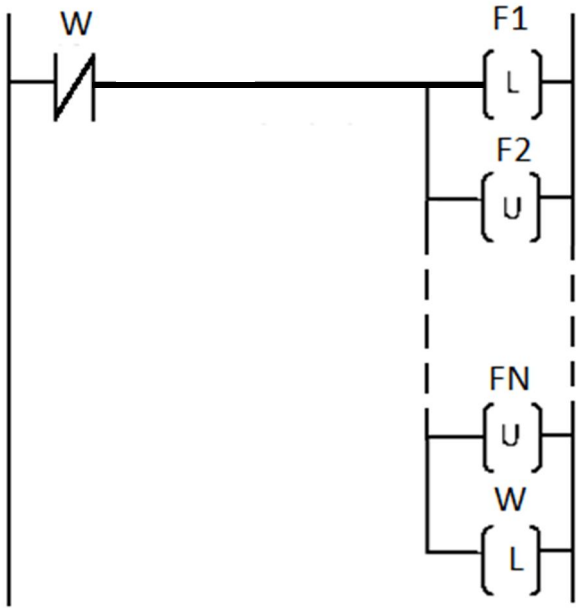


Un costrutto di parallelismo va chiuso con la sincronizzazione ovvero la chiusura di più rami in parallelo in un'unica sequenza. Si chiama sincronismo in quanto tutte le fasi a monte della transizione di chiusura devono essere attive prima di proseguire nella sequenza; ciò significa sincronizzare l'esecuzione dei rami in parallelo del programma.

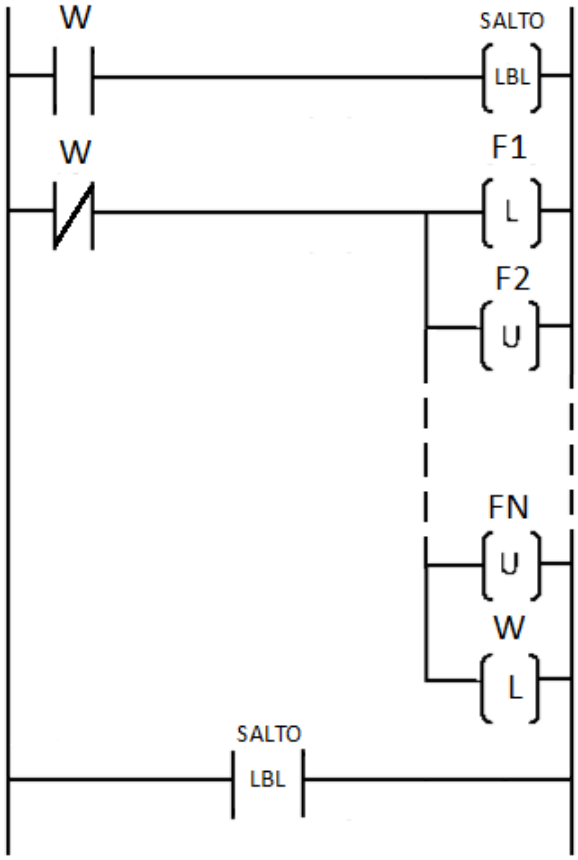
Conversione SFC - LADDER

È possibile convertire l'SFC in Ladder, rendendo più semplice la programmazione? Si, tenendo a mente alcune differenze dovute alla diversa natura dei due linguaggi. La conversione SFC-LADDER si compone di quattro fasi:

1. **Inizializzazione delle variabili:** è necessario settare le variabili affinché al primo ciclo di esecuzione risulti attiva solo la variabile corrispondente alla prima fase dell'SFC. In questo modo si rispetta la sequenzialità che è caratteristica dell'SFC

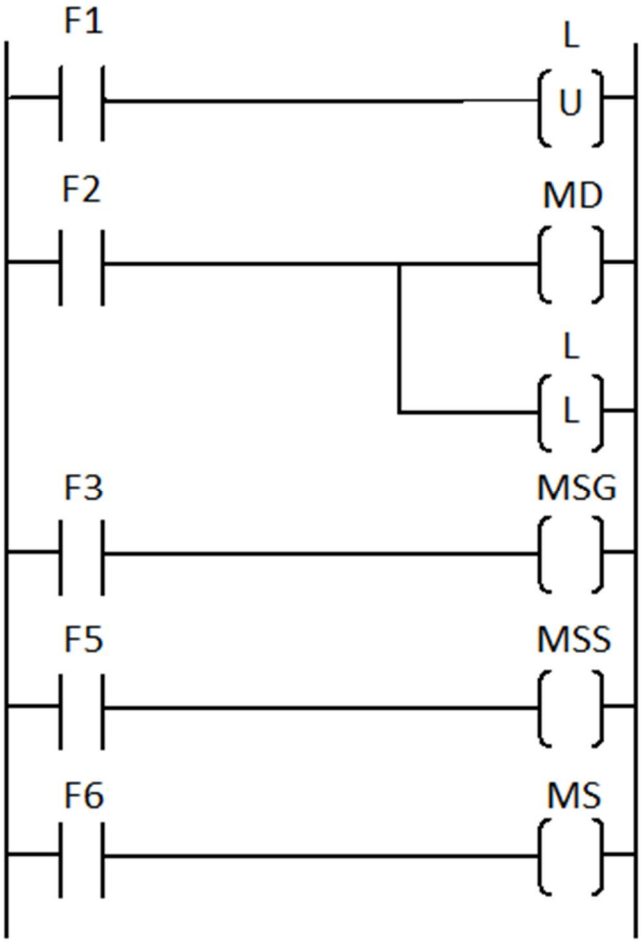


Nell'inizializzazione è fondamentale garantire che tutte le fasi da F2 a FN siano inattive per coerenza con le regole di evoluzione dell'SFC. Il Latch su W garantisce che la fase sia eseguita solo la prima volta che viene utilizzato il macchinario. Per render più prestante il PLC è possibile utilizzare un'istruzione di salto che setti analogamente le variabili ma poi non venga più letta dal PLC.



Inizializzazione con salto.

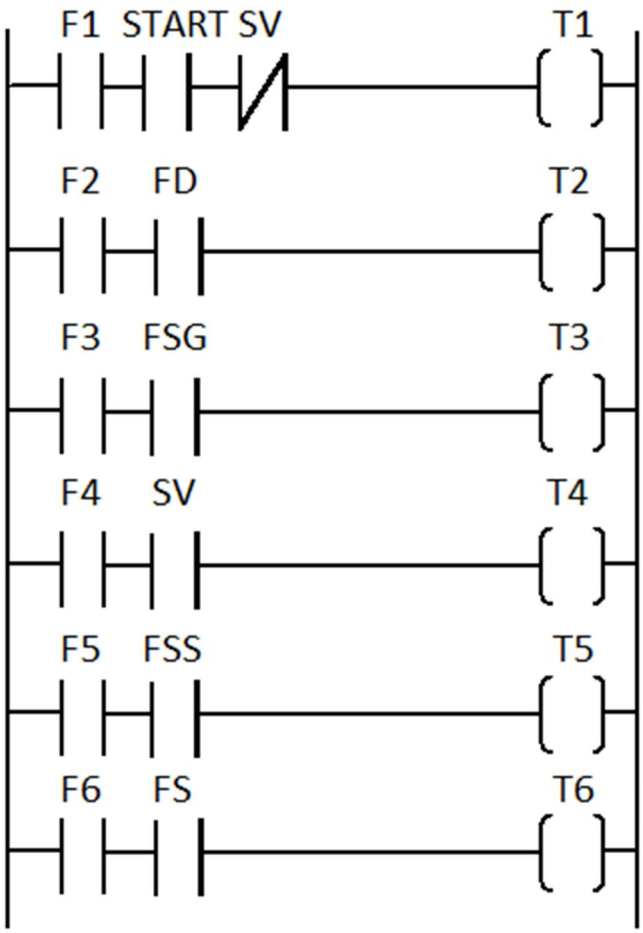
2. **Esecuzione delle azioni:** dall'SFC si abbina a ogni fase l'azione corrispondente come illustrato nell'esempio:



Ad ogni fase si associa l'azione che le è connessa nell'SFC. Le variabili F sono le variabili interne che regolano l'avanzamento e fanno sì che il ladder si comporti in modo sequenziale, in quanto sarà attiva non più di una fase alla volta (a meno di parallelismi). Su L si esegue un latch perché è necessario

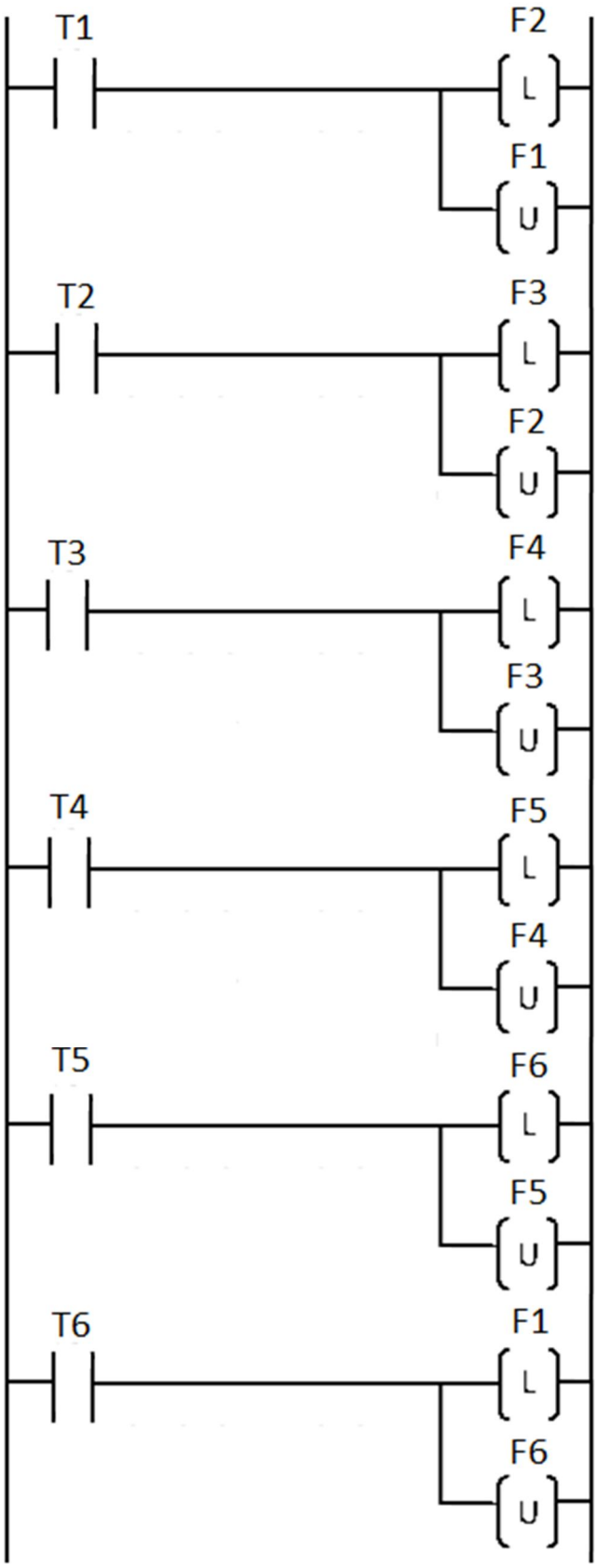
Le azioni condizionate sono da inserire come azioni normali, dove in serie alla fase va posto il contatto normalmente aperto (o chiuso) della condizione.

3. **Superamento delle transizioni:** per consentire il passaggio da una fase all'altra è necessario riportare in logica Ladder il superamento delle transizioni, che nell'esempio sono indicate come variabili interne di nome T seguite dal numero associato.



Le condizioni della transizione sono poste in serie alla fase a garantire che non possa scattare per errore una transizione per cui la corrispettiva fase non sia attiva.

4. **Scatto delle fasi:** l'ultimo aspetto della traduzione consta nel consentire il passaggio tra una fase e l'altra: in questo modo attraverso il latch della fase si è in grado di garantire la sequenzialità dell'SFC anche al Ladder:



Per ottenere lo scatto tra due fasi è necessario che la condizione tra di esse sia superata. Il latch della fase garantisce la possibilità di proseguire l'elaborazione e l'unlatch della precedente garantisce la sequenzialità.

Il latch di F1 durante la transizione T6 svolge sostanzialmente la stessa funzione della freccia verso l'alto, tipica dell'ultimo passo dell'SFC. Quindi, per garantire la sequenzialità e una corretta interpretazione delle regole di evoluzione, all'unlatch della fase 6 corrisponde il latch di F1.

NOTA BENE: è possibile accorpare le fasi 3 e 4 della traduzione SFC ladder; questa operazione rende il codice più compatto, ma fa sparire le variabili interne associate alle transizioni, con una conseguente perdita di parallelismo tra il ladder stesso e l'SFC. Inoltre, mantenere il sistema modulare consente una maggiore sicurezza e una migliore lettura del codice in fase di manutenzione o di debugging.

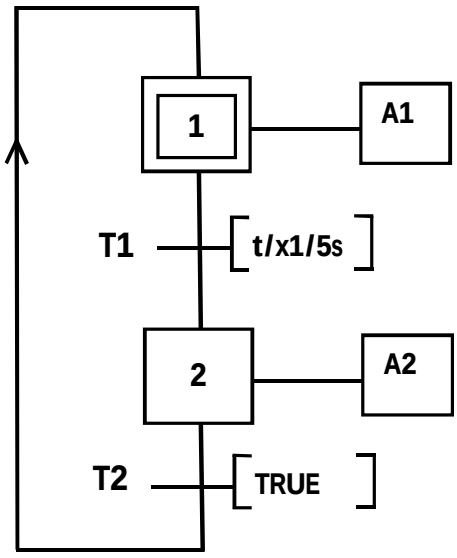
I pezzi più complessi, come i temporizzatori, devono essere inseriti nel blocco delle azioni per quel che riguarda l'aggiornamento del valore della variabile interna, ma, a questa operazione, corrisponde anche un aumento di controllo sulla fase di superamento delle transizioni, che è da realizzare ponendo in serie alla fase corrispondente la variabile controllata dal contatore.

Altri costrutti tipici dell'SFC sono riconoscibili nella traduzione grazie a particolarità nelle fasi di superamento, scatto ed esecuzione delle azioni.

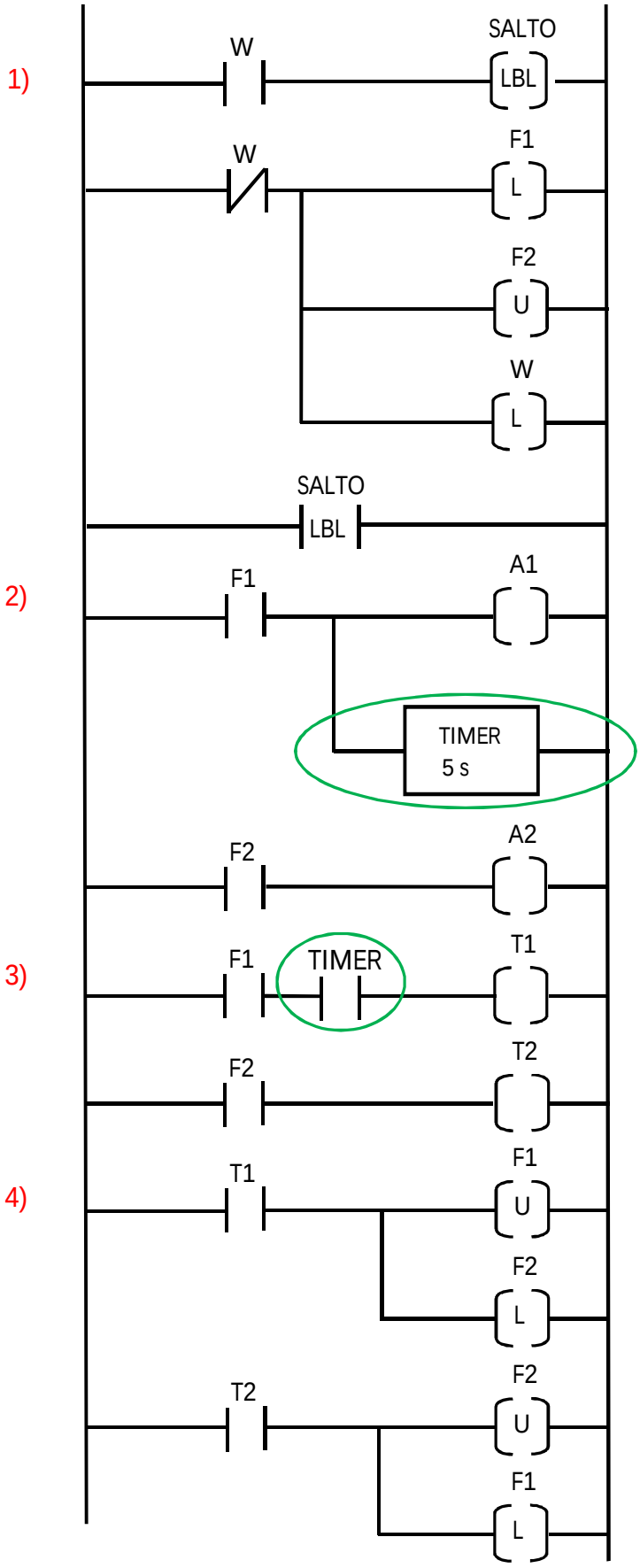
Il costrutto di scelta si nota poiché prima delle bobine dell'uscita sono posti in serie i contatti delle condizioni. Il parallelismo si sviluppa soltanto nel blocco dello scatto delle fasi ed è l'unico caso in cui ad un unlatch è associato più di un latch (ad indicare che il programma proseguirà concorrentemente). Il sincronismo nel punto 4 si manifesta come l'abbinamento di più unlatch con un'unica fase di latch che consente al programma di ritornare ad un unico percorso di esecuzione. Nel superamento delle transizioni, il sincronismo si presenta con una serie di fasi che devono essere contemporaneamente attive per consentire la transizione. La convergenza, invece, si nota solo nel punto 4 perché si hanno due rami che operano latch sulla stessa bobina.

Conversione da SFC a Ladder di una variabile temporale

SFC

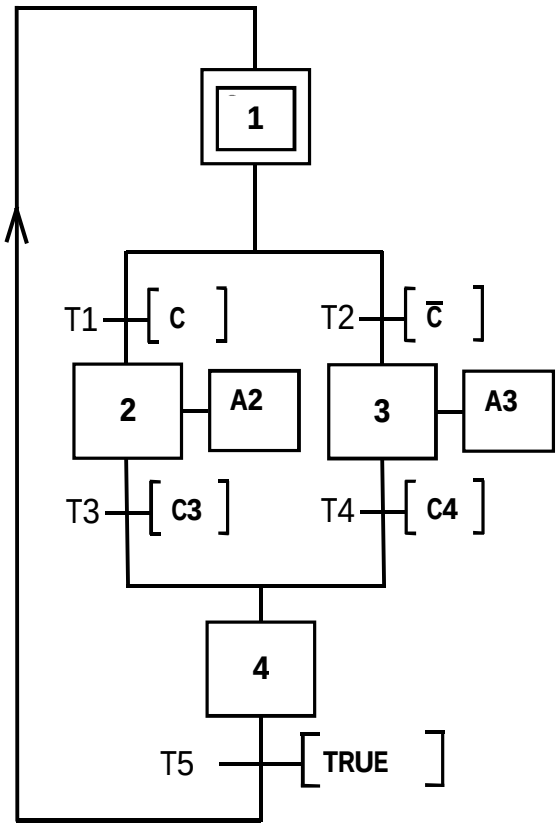


Ladder



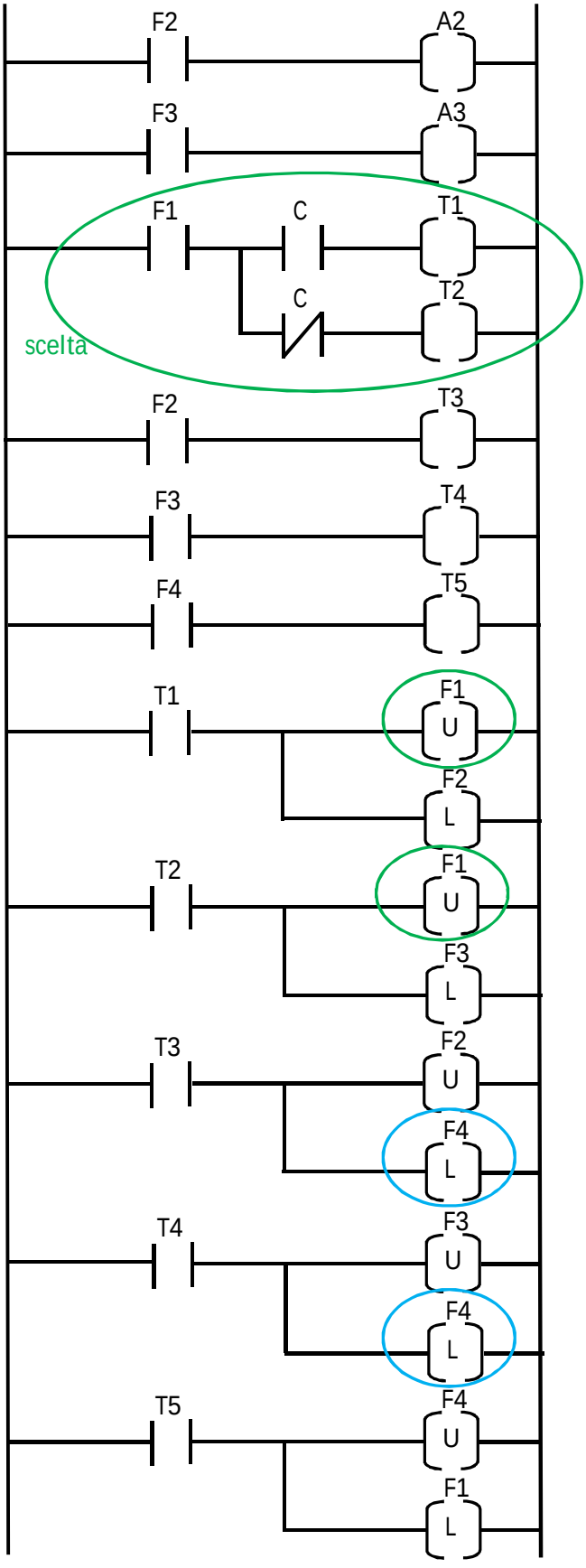
Conversione da SFC a Ladder di una scelta e di una convergenza

SFC



Ladder

N.B. la fase 1) è stata omessa

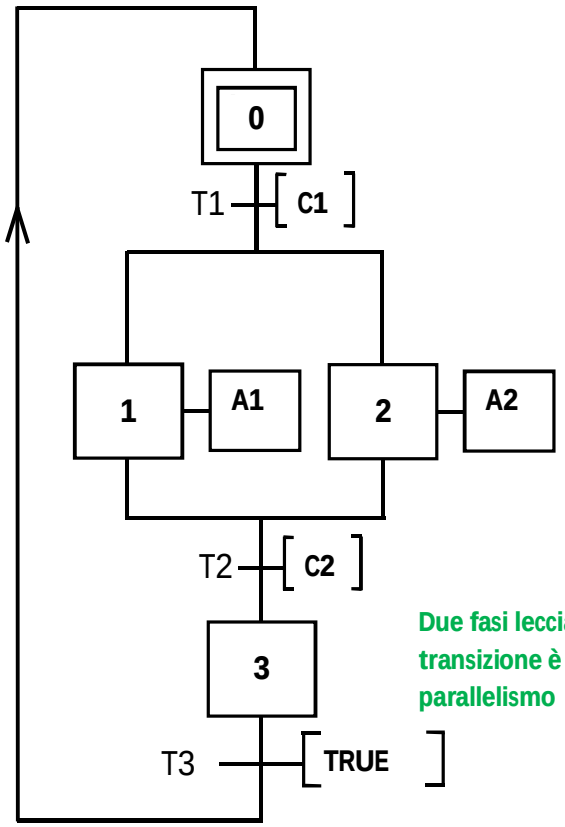


Una fase unlecciata da più transizioni è una scelta

Una fase lecciata da più transizioni è una convergenza

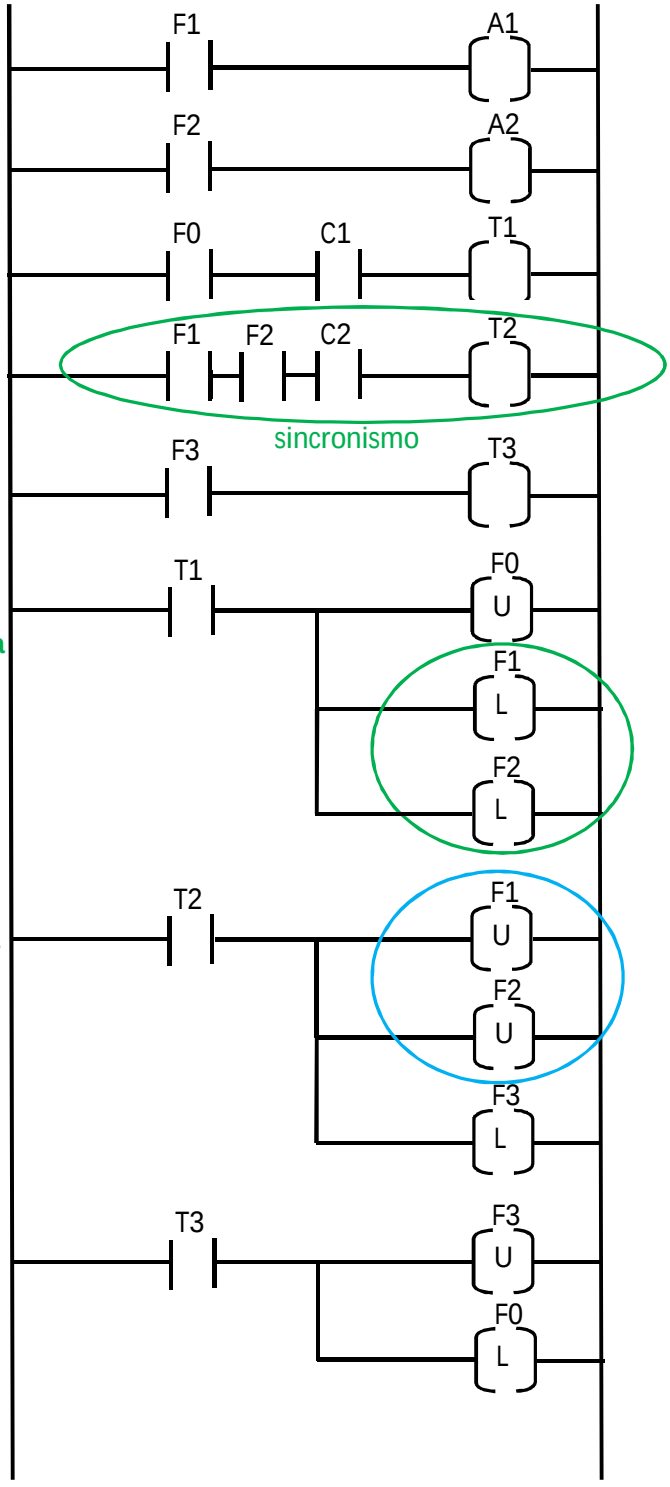
Conversione da SFC a Ladder di un parallelismo e di un sincronismo

SFC



Ladder

N.B. la fase 1) è stata omessa



Due fasi lecciate da una transizione è un parallelismo

Due fasi unlecciate da una transizione è un sincronismo

ISaGRAF

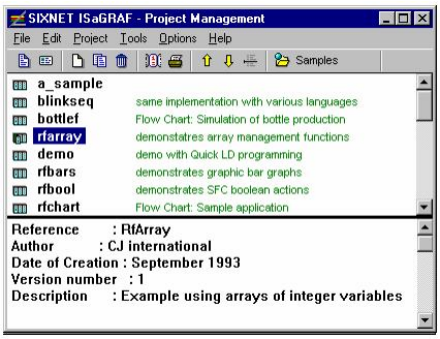
ISaGRAF è un ambiente di sviluppo per sistemi di controllo logico, basato su linguaggi standard di programmazione di PLC.

Un progetto ISaGRAF è una collezione di programmi, sotto-programmi e funzioni che formano un'applicazione di controllo completa (eseguibile su un controllore target). Ogni programma controlla una parte specifica dell'applicazione.

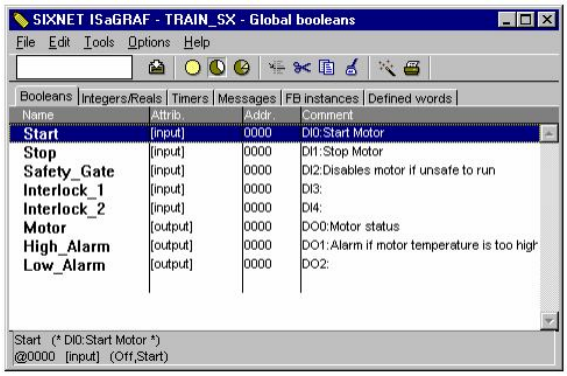
Come utilizzare il programma:

- Aprire il programma tramite il file exe "Progetti" nell'apposita cartella nel menù START
- Creare un nuovo progetto:
    - File»Nuovo:
        - Associare un titolo: ad es. RfArray
        - Associare una descrizione
        - Selezionare in Configurazione di I/O: "sim_all" nel menù a scorrimento

        - La sezione superiore contiene la lista dei progetti, quella inferiore una descrizione del progetto selezionato:

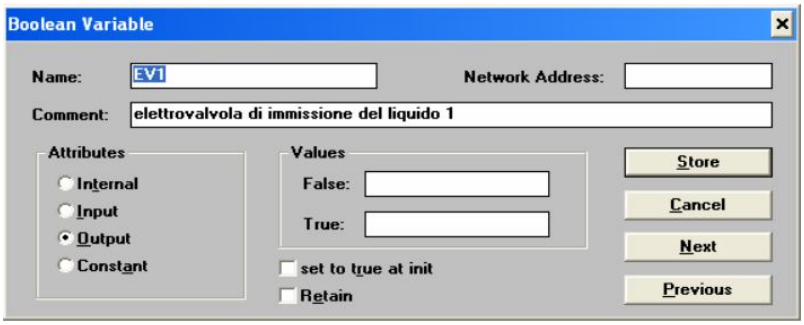


- Aprire il nuovo progetto (doppio click sul nome del progetto)
- Aggiungere le variabili aprendo l'apposita icona "Dizionario". Il dizionario è la collezione di tutte le variabili interne, di ingresso o di uscita usate nei programmi di un progetto. Le variabili di ingresso e uscita sono associate a variabili del controllore o a ingressi e uscite virtuali.



Quindi:

- Cancellare tutte le variabili preesistenti:
  - Selezionare le icone»Taglia, in ogni finestra delle variabili (booleane, interi/reali, ecc.)
- Aggiungere le nuove variabili (utilizzare la finestra “booleani” per le variabili booleane, “interi/reali” per gli interi, ecc):
  - Tasto destro»Nuovo: riempire gli appositi spazi e stabilire la natura delle variabili (ingresso, uscita, interna, ecc)



- Chiudere il Dizionario: rispondere “Si” alla finestra automatica di salvataggio

➢ Associare le variabili al canale di ingresso o di uscita: cliccare sull’icona “Connessione I/O. Tutte le variabili definite nel dizionario come variabili di ingresso e uscita devono poi essere mappate su dei dispositivi fisici oppure su delle “schede virtuali”. È necessario inserire delle schede di ingresso su cui mappare gli ingressi, e delle schede di uscita su cui mappare le uscite:

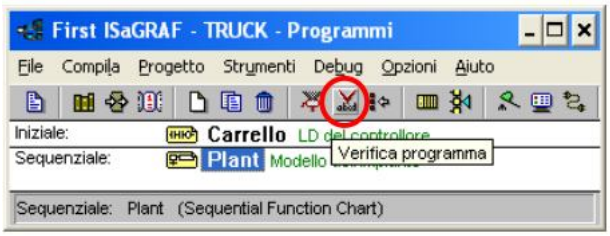
- Nell’apposita sezione XBI8 dovremo associare ogni casella ad una variabile in ingresso:
  - Casella 1»doppio click e associo la variabile con doppio click
  - Casella 2»doppio click ecc..

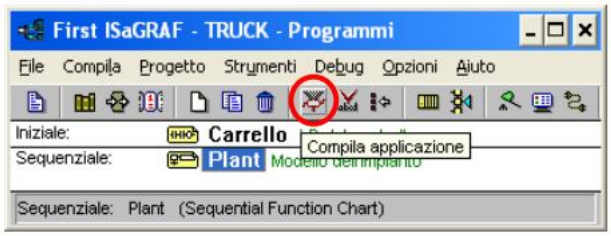
- Nell'apposita sezione XBO8 dovremo associare ogni casella ad una variabile in uscita:
  - Casella 1»doppio click e associo la variabile con doppio click
  - Casella 2»doppio click ecc..
- In questo modo ho configurato la scheda di INPUT&OUTPUT
- NB: le variabili interne non devono essere associate ad alcun canale
- Chiudere la finestra di Connessione I/O: rispondere "Si" alla finestra automatica di salvataggio

➢ Creare l'algoritmo in linguaggio Ladder:

- Sempre nella finestra del nostro progetto aperta:
  - File»Nuovo»Quick LD:Ladder Diagram
- Doppio click: si apre la finestra dell'editor
- È consigliato cliccare sull'icona apposita per far comparire la griglia
- Costruire il proprio algoritmo (è possibile aiutarsi con i tasti F2, F3 ecc.)
- Tramite l'icona "—|?|—" è possibile modificare, selezionandoli prima, contatti e bobine, da normalmente aperti a normalmente chiusi e da bobine semplici a bobine latch (Set=Latch: —(S)—) e unlatch (Reset=Unlatch: —(R)—)
- Per quanto riguarda il contatore: inserire un blocco "CTU" a sinistra della bobina, cliccare vicino a "PV" per inserire una variabile intera (la soglia del contatore), cliccare vicino a "RESET" per inserire la variabile booleana di reset, "CV" indica il conteggio.
  ATTENZIONE: a sinistra del contatore è necessario inserire un blocco Trigger per evitare che il contatore venga incrementato di più unità ad ogni ciclo

➢ Prima di essere simulato, il codice deve essere verificato e compilato:

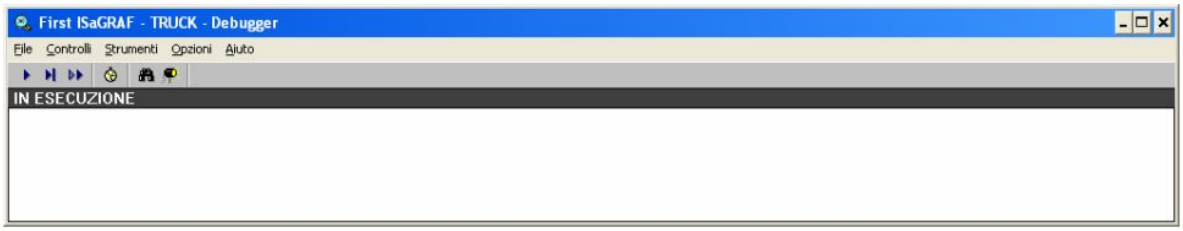




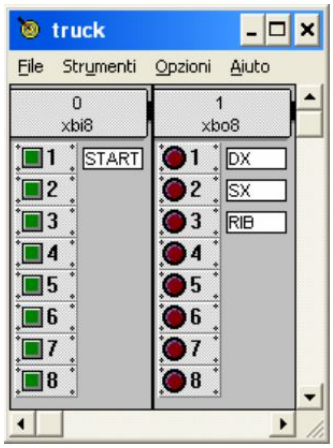
- Una volta compilato il codice è possibile eseguire una simulazione oppure un debug del programma utilizzando il comando Debug (oppure Opzioni»Compila) »Simulazione.
  ATTENZIONE: se il programma non viene salvato e compilato prima di eseguire la simulazione, essa potrebbe risultare scorretta, in quanto l'algoritmo potrebbe non essere stato adeguatamente aggiornato

➢ Simulazione del codice, cliccare sull'apposita icona "Simulazione":

- Si aprirà una finestra "Debugger": finestra principale utilizzata per commutare tra la modalità di esecuzione in tempo reale e quella passo passo (utile in fase di debug).

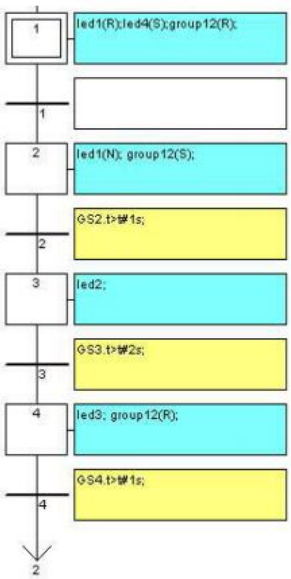


- Si aprirà una finestra "Pannello di Controllo": rappresenta le variabili mappate sulle schede virtuali di ingresso ed uscita e viene utilizzata per controllare lo stato degli ingressi e per visualizzare lo stato delle uscite. I pulsanti verdi rappresentano gli ingressi del PLC (uscite dell'impianto controllato), mentre le spie rosse rappresentano le uscite del PLC (ingressi dell'impianto controllato).



- "Giocare" con i tasti di ingresso del PLC (verdi), simulando il processo fisico che si è descritto con l'algoritmo
- Osservare l'effetto degli ingressi sulle uscite
- Durante la fase di simulazione è possibile aprire le finestre del codice in modo da seguirne l'esecuzione. Nei programmi LD viene evidenziato lo stato delle variabili (rosso = vero, blu = falso), dei contatti e delle bobine (rosso=aperto, blu=chiuso). Nei programmi SFC vengono invece evidenziati i passi attivi.

➢ SFC: Sequential Function Charts: l'utilizzo di questo linguaggio è relativamente intuitivo; il risultato finale della trascrizione dell'algoritmo in SFC sarà simile a:



Dove i riquadri azzurri contengono le azioni, separate l'un l'altra dalla virgola ";", mentre i riquadri gialli contengono le condizioni.
La prima fase, quella iniziale, è indicata con un doppio riquadro.
Per simulare il processo si ricorda di salvare e compilare prima di far partire la simulazione.
Similmente a quanto accade nel Ladder si aprirà una finestra tramite la quale controllare gli ingressi e osservare l'effetto sulle uscite.
Si noti come tale linguaggio è sequenziale e quindi non si cura di nessuna variabile in ingresso, se non di quella (o quelle) presente nella condizione associata alla trasizione corrente.