

# SFC

## Sequential Function Chart

### **Il Sequential Function Chart (SFC) è:**

- un formalismo grafico per la descrizione del ciclo operativo di macchine automatiche.
- definito dalla Norma IEC 61131-3 tra gli elementi comuni, ma trova diretta implementazione come linguaggio di programmazione in molti tools di sviluppo.
- facilita la scomposizione gerarchica del funzionamento della macchina e la strutturazione del programma in sottoparti più semplici.
- evidenzia il comportamento sequenziale della macchina o di sue componenti.

### **Il Sequential Function Chart (SFC) nasce per:**

- Necessità di un linguaggio formale. Il progetto di una macchina passa attraverso una serie di passaggi (dalla analisi della commessa del cliente, la traduzione in specifiche tecniche, e la realizzazione tecnica) in cui persone con diversi profili professionali lavorano sul progetto comune. Tali persone necessitano di un linguaggio formale comune.
- Necessità di un approccio top-down. Il funzionamento della macchina viene scomposto in passi fondamentali analizzati separatamente, riducendo così la complessità di ogni singolo elemento che compone, in questo caso, la logica del sistema di controllo.
- Necessità di semplificare la descrizione del funzionamento. Descrivendo il comportamento ingresso–uscita della macchina, occorre fornire una configurazione di controllo per una qualunque combinazione degli stimoli (ingressi acquisiti dai sensori). Una descrizione ingresso-stato-uscita porta ad una descrizione molto più semplice.

### **Storia dell' SFC**

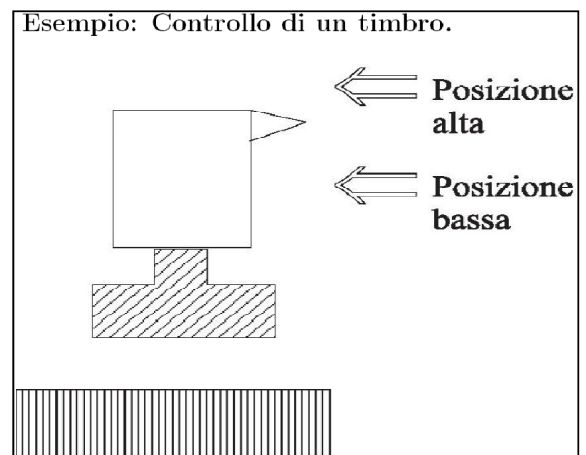
- Le reti di Petri vengono inventate nel 1962 da Carl Adam Petri durante la sua tesi di dottorato. Esse definiscono un formalismo utilizzato in ambito accademico per l'analisi di sistemi che evolvono in sequenze di stati successivi (Sistemi Dinamici a Eventi Discreti, DEDS).
- Nel 1988, la IEC pubblica lo standard IEC 848 che definisce la sintassi del linguaggio Grafcet (<http://www.lurpa.ens-cachan.fr/grafcet.html>), derivato dalle Reti di Petri.
- Successivamente il Grafcet viene portato all'interno dei linguaggi della norma IEC 61131-3 con qualche estensione per l'integrazione con gli altri linguaggi della norma, dando origine al Sequential Function Chart

## Elementi fondamentali dell' SFC

- Stati logici di funzionamento (Steps).
- Transizioni tra gli stati logici, condizionati dal valore di segnali sensoriali (Transitions).

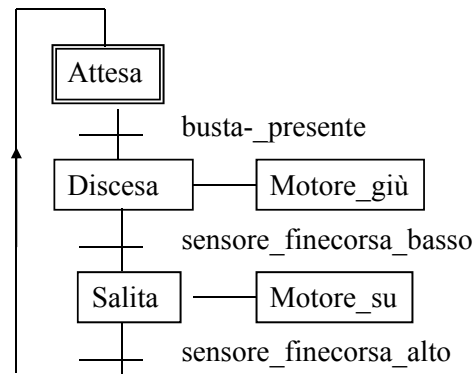
### Descrizione del funzionamento

La timbratrice deve ripetitivamente salire e scendere per effettuare timbri su buste che vengono poste sul supporto inferiore da un'altra macchina.



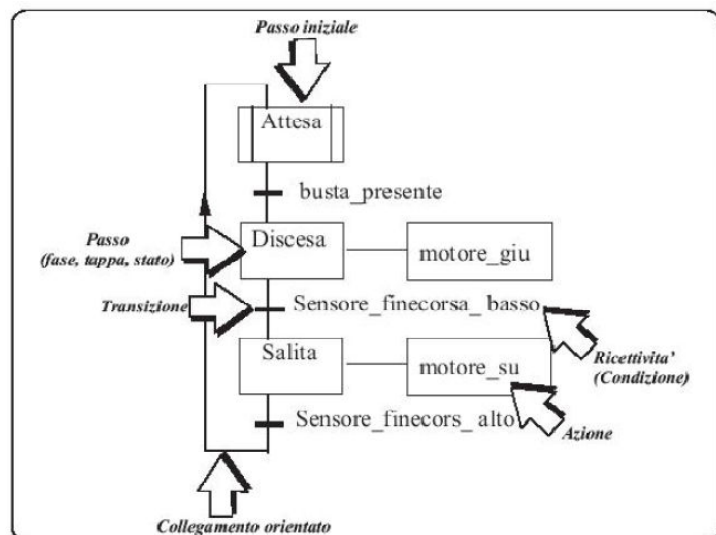
- 1) Quando il sistema di controllo della timbratrice riceve il segnale di “busta presente”, avvia il motore in direzione di discesa fino a che il sensore di presenza inferiore segnala che il pistone di timbratura è arrivato al finecorsa.
- 2) A questo punto il motore deve invertire la sua corsa e far risalire il pistone fino a che non viene attivato il sensore di fine corsa superiore.
- 3) A questo punto la macchina timbratrice segnala che la busta può essere rimossa dalla sede di lavorazione.

## SFC del controllo



## Concetti base dell' SFC

- **Il Passo** (o fase o tappa o stato)  
→ Ad ogni passo viene associata un'azione
- **La transizione**  
→ Ad ogni transizione viene associata una ricettività (o condizione)
- **Il collegamento orientato**



## I Passi (Steps)

- Un passo rappresenta una situazione in cui il comportamento del controllore (rispetto agli ingressi e uscite) segue le regole definite dalle azioni associate allo stato.
- Il passo può essere attivo o disattivo.
- Lo stato di attività di un passo può essere rappresentato dal possesso di un “token”, rappresentabile graficamente con un punto pieno.
- Ad ogni istante il sistema di controllo è descrivibile attraverso:
  - Gli stati attivi.
  - L’insieme delle variabili interne e di uscita.
- In ogni schema deve essere presente uno ed uno solo stato iniziale.

### Standard IEC 61131-3: sintassi del passo (step)

<pre>         +-----+    ***    +-----+                   </pre>	Step - Graphical form. “***” = step name	
<pre>         +=====+     ***                 +=====+                   </pre>	Initial step - Graphical form. “***” = Name of initial step	
***.X	Step flag - “***” = Step name; ***.X = Boolean 1 when *** is active, Boolean 0 otherwise	
***.T	Step elapsed time - “***” = Step name; ***.T = A variable of type TIME	
<pre> STEP *** : (* Step body *) END_STEP           </pre>	Step - Textual form without directed links “***” = Step name	posso definire anche testualmente il comportamento di un passo dell’SFC. Questo è dovuto alla notevole “plasticità” della norma
<pre> INITIAL_STEP *** : (* Step body *) END_STEP           </pre>	Initial step - Textual form without directed links “***” = Name of initial step	

**N.B.**

Il nome del passo **\*\*\*.X** e il temporizzatore implicito associato al passo **\*\*\*.T** sono variabili che possono essere utilizzate nel programma ma non possono venire modificate esplicitamente tramite una istruzione.

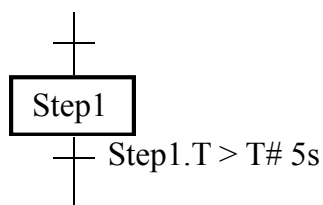
In altre parole, le istruzioni:

S4.X := 1 ; (\* ERROR \*)

S4.T := t#100ms ; (\* ERROR \*)

debbono essere segnalate come errore.

Es.



Quando il programma arriva nello stato Step1, vi permane per 5s prima di passare allo stato successivo. Quindi la variabile Step1.X varrà 1 per 5 secondi, poi tornerà a 0

## Le transizioni

- Una transizione rappresenta la condizione secondo la quale il controllo passa da un passo (passi) predecessore ad un passo (passi) successore secondo la topologia stabilita dalla rete di connessioni.
- La transizione è graficamente rappresentata da una linea orizzontale che taglia il collegamento verticale.
- Ogni transizione deve avere associata una condizione, espressione dalla cui valutazione deve risultare un valore booleano.
- Una condizione sempre vera deve essere identificata dal simbolo “1” oppure dalla parola chiave “TRUE”.

### Standard IEC 61131-3: sintassi transizioni

1	<pre>         +-----+  STEP7  +-----+               + %IX2.4 &amp; %IX2.3         +-----+  STEP8  +-----+         </pre>	<p>Predecessor step</p> <p>Transition      condition using ST language</p> <p>Successor step</p>
2	<pre>               +-----+        STEP7        +-----+   %IX2.4  %IX2.3    +---  -----  -----+                                    +-----+        STEP8        +-----+         </pre>	<p>Predecessor step</p> <p>Transition      condition using LD language</p> <p>Successor step</p>

Definisco una  
transizione SFC  
mediante il  
ladder

3	<pre>               +-----+        STEP7        +-----+               &gt;TRANX&gt;-----+               +-----+        STEP8        +-----+         </pre>	<p>Use of connector: Predecessor step</p> <p>Transition connector</p> <p>Successor step</p>
3a	<pre>                 %IX2.4 %IX2.3       +---  -----  -----&gt;TRANX&gt;         </pre>	<p>Transition condition: Using LD language</p>
4	<pre> STEP STEP7: END_STEP TRANSITION FROM STEP7 TO STEP8     := %IX2.4 &amp; %IX2.3 ; END_TRANSITION STEP STEP8: END_STEP </pre>	<p>Textual equivalent of feature 1 using ST language</p>

5	<pre>               +-----+        STEP7        +-----+               + TRAN78               +-----+        STEP8        +-----+         </pre>	<p>Use of transition name:</p> <p>Predecessor step</p> <p>Transition name</p> <p>Successor step</p>
5a	<pre> TRANSITION TRAN78:   %IX2.4 %IX2.3 TRAN78   +---  -----  ----- ( )----+   END_TRANSITION </pre>	<p>Transition condition using LD language</p>
5b	<pre> TRANSITION TRAN78     := %IX2.4 &amp; %IX2.3 ; END_TRANSITION </pre>	<p>Transition condition using ST language</p>



## **Le azioni**

- L'azione descrive il comportamento del controllore in corrispondenza della attivazione di un passo.
- Ad ogni passo sono associate zero o più azioni.
- Un passo di attesa (wait) ha zero azioni associate.

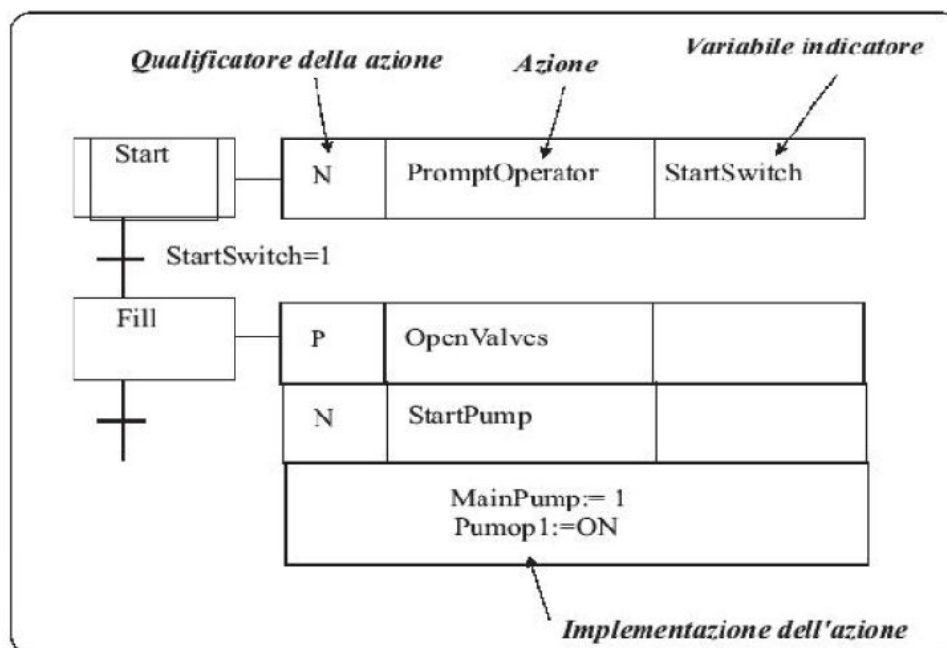
## **Action blocks**

- Un action block è un elemento grafico che definisce le proprietà dell'azione associata al passo.
- Un action block è definito tramite:
  - Il qualificatore dell'azione. (quale tipo di azione?)
  - L'identificativo dell'azione. (come si chiama?)
  - Una variabile indicatore booleana. (Che può essere utilizzata per segnalare lo stato di esecuzione.)

### **Un' azione è definita mediante un blocco grafico in cui sono specificati i seguenti tre elementi:**

- Il qualificatore dell'azione, che definisce il tipo dell'azione da intraprendere (se impulsiva, continua, temporizzata, etc.).
- L'identificativo dell'azione stessa. L'implementazione effettiva dell' azione può avvenire mediante diversi meccanismi che verranno discussi più avanti. In generale essa sarà implementata tramite un sottoprogramma di una qualche complessità, per cui l'effettiva implementazione dell'azione è rimandata ad un altro punto del programma. Il collegamento tra collocazione dell'azione e sua implementazione è reso possibile attraverso l'identificativo.  
In casi particolarmente semplici, l'azione può venir descritta direttamente in corrispondenza dell'identificativo nell'action block, o, se l'azione corrisponde semplicemente a modificare una variabile booleana, l'identificativo e l'implementazione dell'azione coincidono.
- La variabile indicatore, è una variabile booleana il cui stato indica il completamento dell'azione. La variabile indicatore è modificata dal codice che implementa l'azione. Nel caso in cui l'implementazione dell'azione si riduca ad una variabile booleana, la variabile indicatore viene chiaramente omessa, in quanto coincide con l'identificativo.

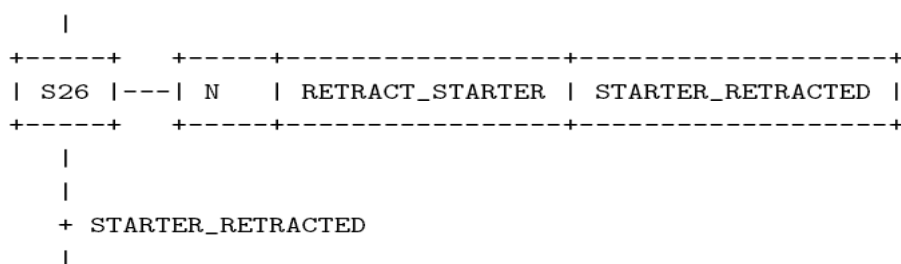
## Un esempio di Action Block



Ing. Elena Mainardi

SFC

## Un esempio dell'uso della variabile indicatore



La variabile indicatore, è una variabile booleana il cui stato indica il completamento dell'azione. La variabile indicatore è modificata dal codice che implementa l'azione.

Ovvero: starter\_retracted sarà una variabile booleana che verrà messa a valore logico vero da qualche istruzione (testuale o grafica) del codice che implementa l'azione retract\_starter. Quando questo succederà, l'SFC evolverà dallo stato S26 al successivo. Es.

```

STEP S26:
  cont:=cont+1;
  somma:=A+B+C;
  starter_retracted:=true;
END_STEP

```

Ing. Elena Mainardi

SFC

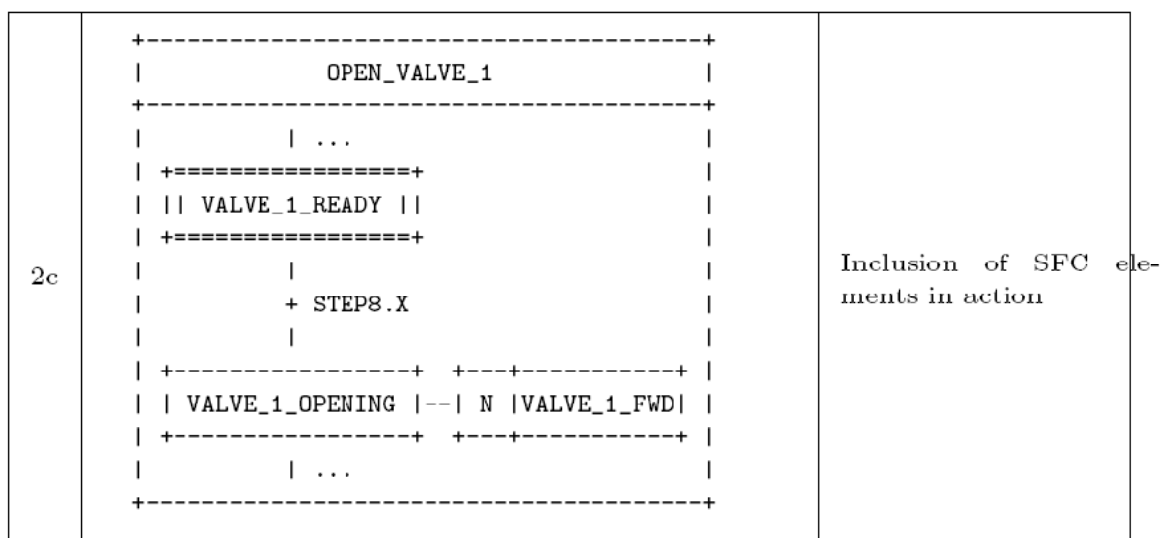
## Implementazione delle azioni

Un' azione può essere implementata tramite:

- Il set (o reset) di una variabile booleana.
- Una serie di istruzioni in Instruction List (IL).
- Una serie di istruzioni in Structured Text (ST).
- Una serie di “rung” in Ladder Diagram (LD).
- Una rete di blocchi funzionali (FBD).
- Un Sequential Function Chart (SFC).

### Standard IEC 61131-3: dichiarazione delle azioni

1	Any Boolean variable declared in a VAR or VAR_OUTPUT block, or their graphical equivalents, can be an action.	
2a	<pre> +-----+                 ACTION_4                 +-----+          %IX1  %MX3  S8.X  %QX17             +---+ -----   -----   -----( )----+  +-----+                             +----+ EN ENO                                       C--   LT   ------(S)----+            D--  +-----+               +-----+ </pre>	Graphical declaration in LD language
2b	<pre> ACTION ACTION_4:   %QX17 := %IX1 &amp; %MX3 &amp; S8.X ;   FF28(S1 := (C&lt;D));   %MX10 := FF28.Q; END_ACTION </pre>	Textual declaration in ST language

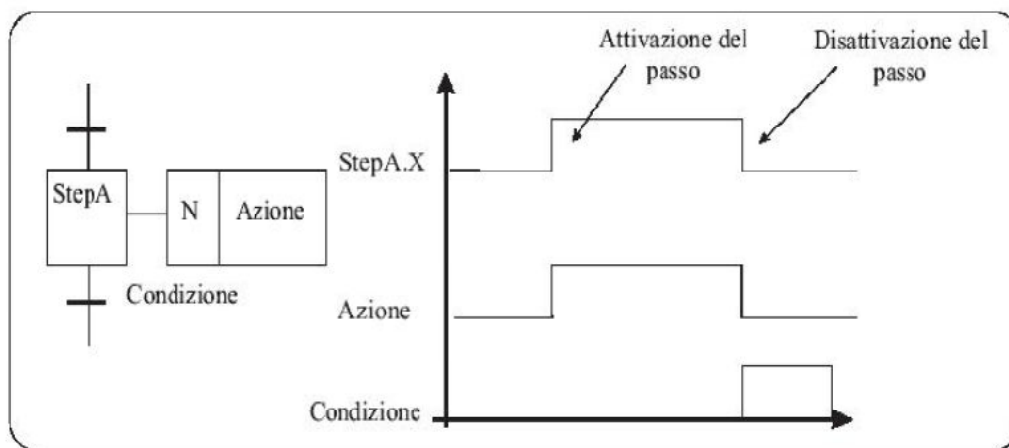


### Qualificatori delle azioni

Qualificatore	Significato	Descrizione
nessuno		Di default ha lo stesso significato di N
N	Non-stored	L'azione termina quando il passo si disattiva
R	Reset	Termina un'azione attivata con i qualificatori S, SD, SL o DS
S	Set (stored)	L'azione continua anche quando il passo si disattiva, e termina solo quando viene resettata
L	Time Limited	L'azione comincia quando il passo diventa attivo e continua finchè il passo si disattiva oppure trascorre un certo intervallo di tempo

Qualificatore	Significato	Descrizione
D	Time delay	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	Pulse	L'azione comincia quando il passo diventa attivo o disattivo e viene eseguita una sola volta
SD	Stored and Time Delayed	L'azione comincia dopo un ritardo anche se il passo diventa inattivo e continua finchè non è resettata
DS	Delayed and Stored	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 viene resettata
SL	Stored and Time Limited	L'azione comincia quando il passo diventa attivo e continua finchè non viene resettata o non trascorre un certo intervallo di tempo

## Azioni continue



Nel caso in cui la condizione di transizione sia considerata erroneamente indipendente dallo svolgimento dell'azione associata al passo attivo l'azione stessa potrà essere interrotta prima della sua conclusione, in modo potenzialmente errato.

### Esempio:

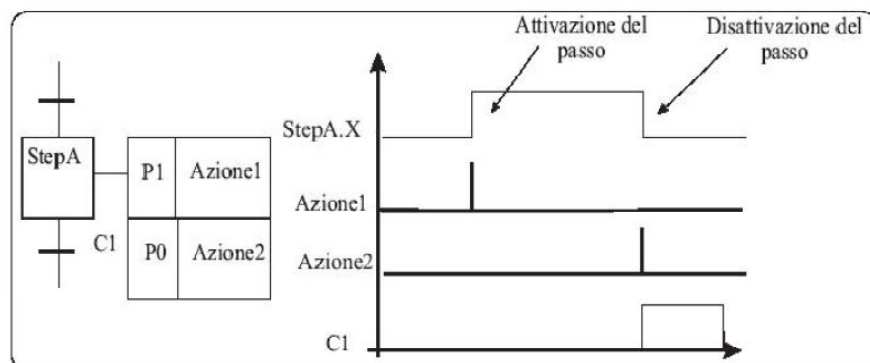
Il movimento di una timbratrice è controllato da un motore elettrico. Nelle pause di lavorazione il timbro viene inchiostrato, e quindi, quando il sensore di presenza della busta si attiva, la timbratrice viene fatta scendere sulla busta.

Quando la busta arriva alla macchina si ha la transizione tra il passo "Inchiostraggio" ed il passo "Discesa timbro", e quindi l'azione di inchiostraggio termina anche se questa non era perfettamente completata, e quindi la qualità della lavorazione potrebbe non essere ottimale.

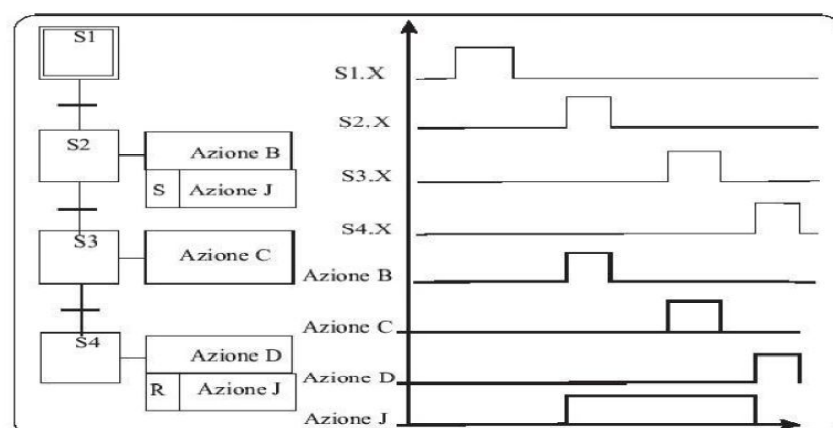
Il motivo alla base del potenziale malfunzionamento consiste nella mancata sincronizzazione tra un evento esterno (l'arrivo della busta) ed un evento interno (il termine dell'azione di inchiostraggio).

Per eliminare questo inconveniente sarà sufficiente introdurre una condizione di termine della fase di inchiostraggio.

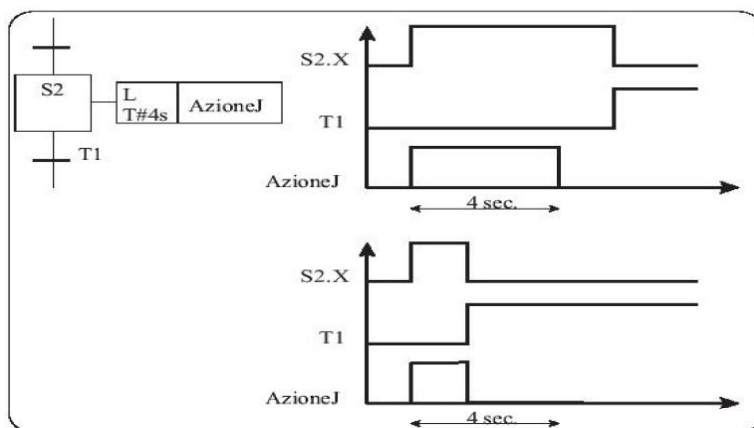
### Azioni impulsive



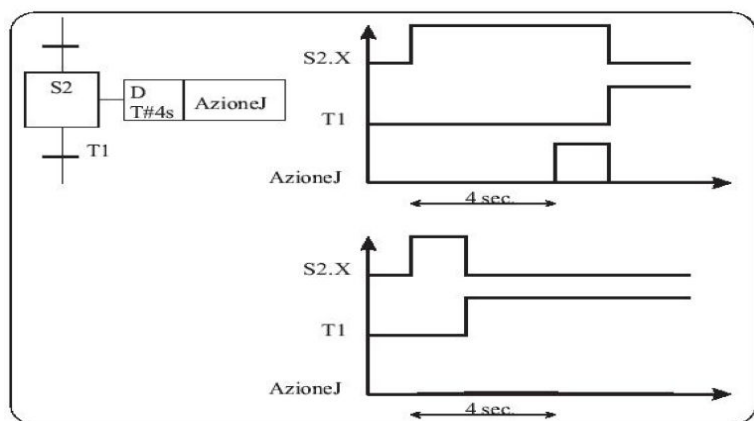
### Azioni prolungate (Set – Reset)



### Azioni “time limited”



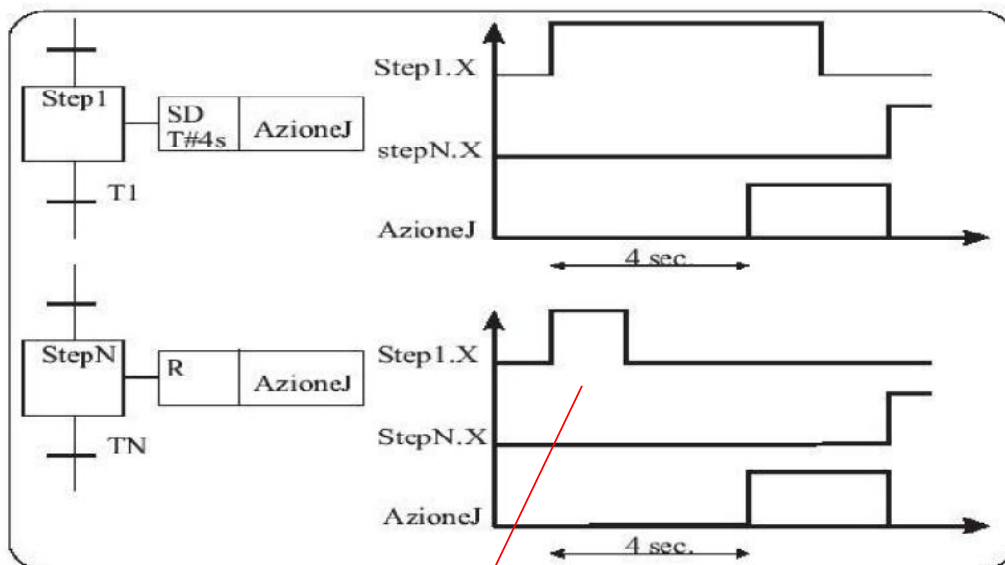
### Azioni “time delayed”



Ing. Elena Mainardi

SFC

### Azioni “stored and time delayed” (o delayed and set)

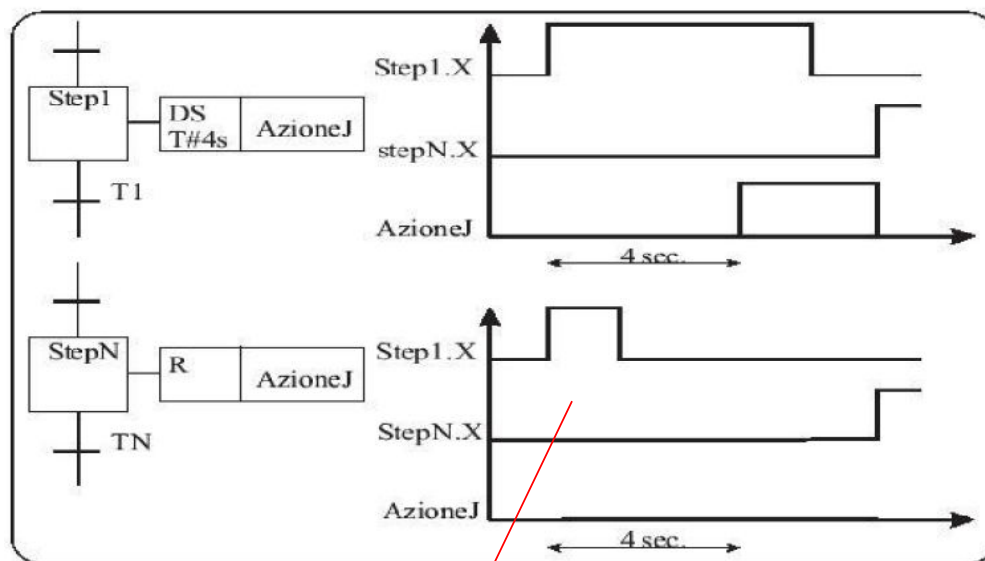


Se anche lo step Step1 dura poco, l'azione parte comunque

Ing. Elena Mainardi

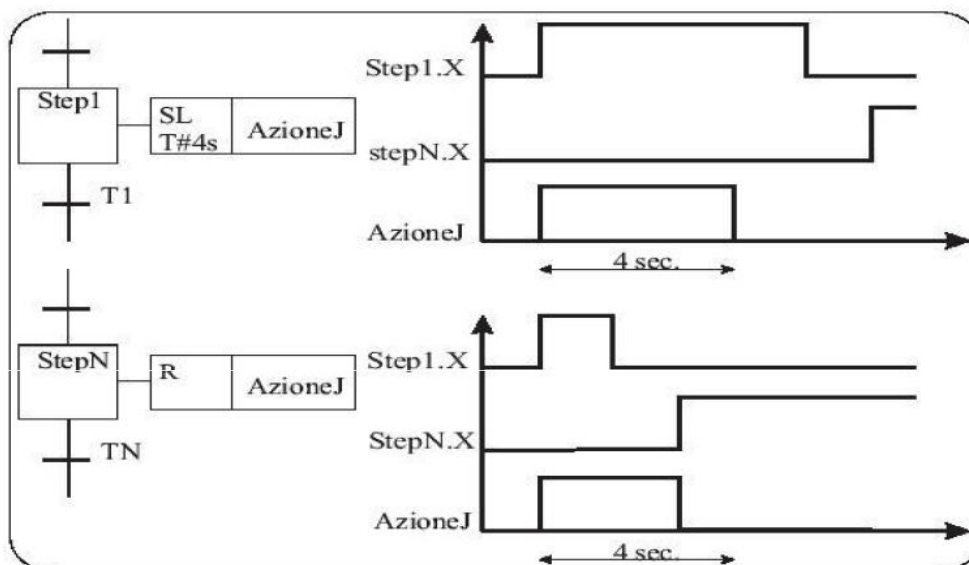
SFC

## Azioni “time delayed and stored”



Se Step1 ha una durata inferiore al tempo settato, l'azione non parte

## Azioni “stored and time limited”



Ovvero ha una durata coincidente al minimo tra :

- il tempo settato
- la distanza temporale che intercorre tra il passo con associata l'azione SL e il passo con associato il reset della stessa azione



## Esecuzione di un SFC

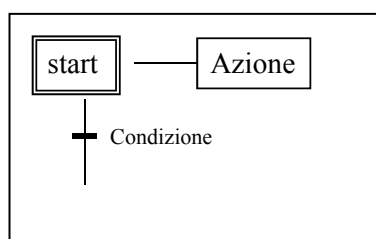
- Schemi SFC possono essere usati per descrivere:
  - Il comportamento di POU (Programs e Function Blocks)
  - Il comportamento delle azioni associate ai passi in SFC
- L'esecuzione di un SFC all'interno di una POU è legata all'esecuzione della TASK che controlla l'esecuzione della POU stessa

L'SFC può essere utilizzato per descrivere il comportamento di una POU. Ogni POU è associato al tempo di esecuzione ad una TASK.

La TASK è responsabile della riesecuzione della POU, e quindi dello schema SFC che la POU contiene.

In base a questa proprietà si osserva che ogni azione attiva all'interno di un SFC viene eseguita con la stessa frequenza di attivazione della TASK.

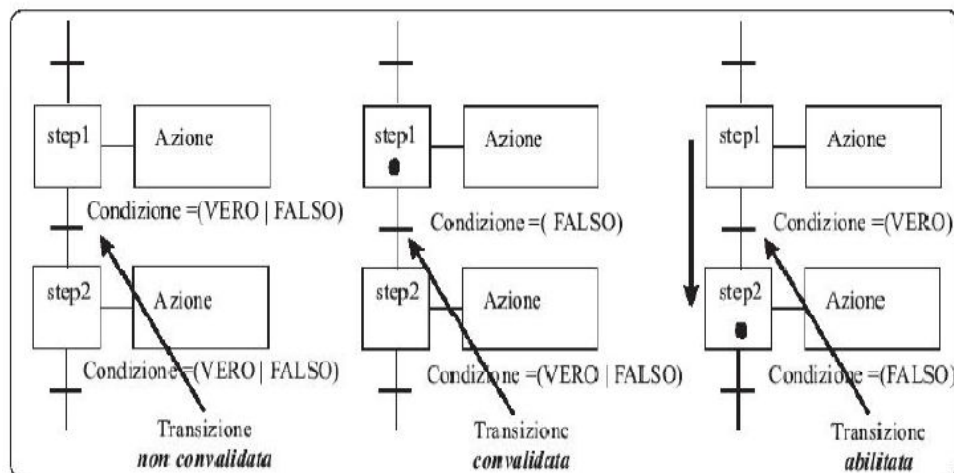
## Regole di evoluzione di uno schema SFC: 1-Inizializzazione



- All'avviamento, in ogni sequenza SFC vi è un passo attivo senza alcuna condizione preliminare: passo iniziale.
- Le azioni associate al passo iniziale vengono immediatamente eseguite.
- Deve esserci sempre uno ed uno solo passo iniziale per ogni sequenza, ma può essere che un programma SFC si componga di più sequenze, nel qual caso ci saranno ovviamente più passi iniziali, uno per ogni sequenza

## Regole di evoluzione: 2-Transizioni

1. Una transizione si dice convalidata (enabled) quando il passo (i passi) precedente è attivo.
2. Una transizione si dice abilitata (cleared) quando essa è convalidata e la condizione associata è vera.
3. Quando una transizione è abilitata, allora avviene l'evoluzione dello stato di attività dei passi interessati.



Ing. Elena Mainardi

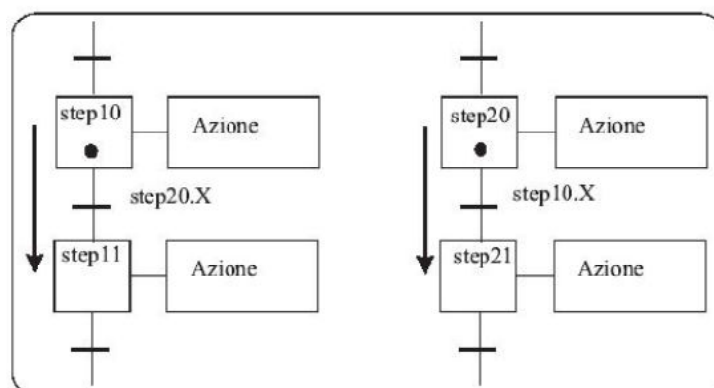
SFC

## Regole di evoluzione: 3- Conseguenza della transizione

L'abilitazione di una transizione provoca:

- L'attivazione del passo (dei passi) immediatamente successivo.
- La disattivazione del passo (dei passi) precedente.

## Regole di evoluzione: 4-Simultaneità



Più transizioni simultanee superabili saranno superate simultaneamente

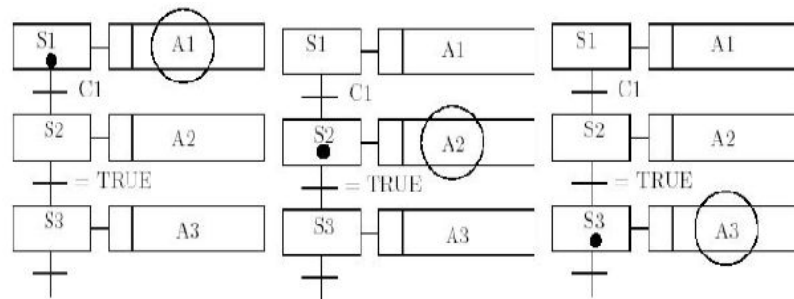
Ing. Elena Mainardi

SFC

## Regole di evoluzione: 5-propagazione dell'effetto della transizione

- Il test delle condizioni associate alla transizione seguente ad un passo attivo non è considerato prima che tutti gli effetti dell'attivazione del passo non si siano propagati attraverso la POU in cui agisce l' SFC.
- Esempio: L'azione associata ad un passo attivo avente una transizione a valle con condizione associata sempre vera, viene eseguita sempre almeno una volta.

Es



Ing. Elena Mainardi

SFC

## Regole di evoluzione: 6-Conseguenze dell'attivazione

- L'attivazione di un passo provoca l'esecuzione di tutte le azioni ad esso associate.
- La disattivazione di un passo provoca l'interruzione di tutte le azioni ad esso associate, a meno che non si tratti di azioni tipo S, SL, SD.....

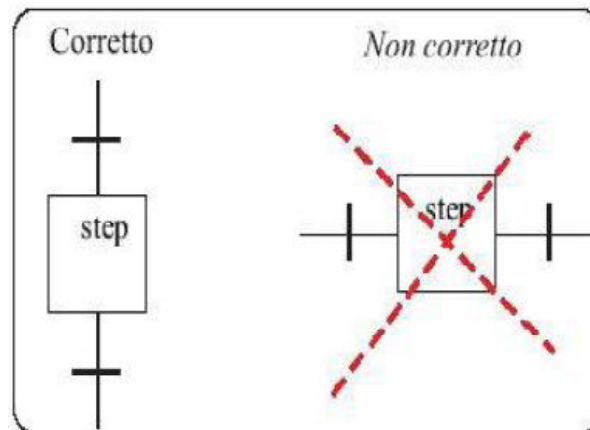
Ing. Elena Mainardi

SFC

## Sintassi dei Passi e delle Transizioni

- Due passi debbono essere separati da una transizione.
- Due transizioni debbono essere separati da un passo.

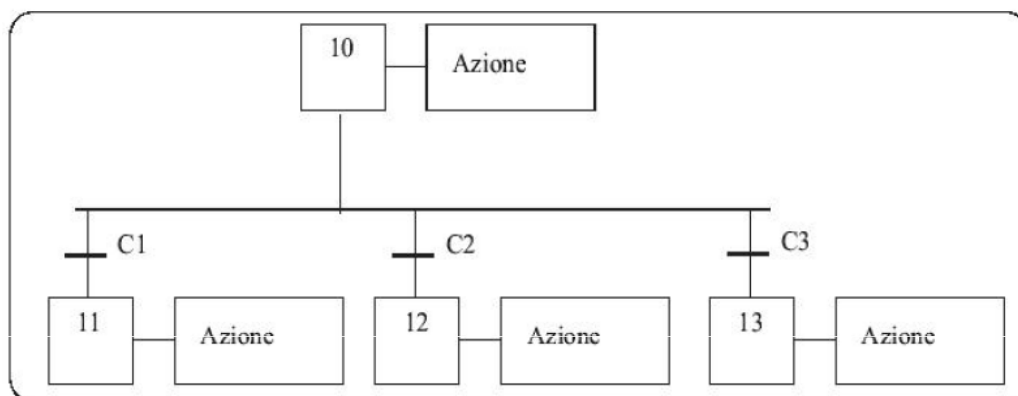
I collegamenti arrivano ai passi o se ne dipartono in posizione verticale.



Ing. Elena Mainardi

SFC

## Divergenza opzionale



La divergenza opzionale esprime un criterio di scelta.

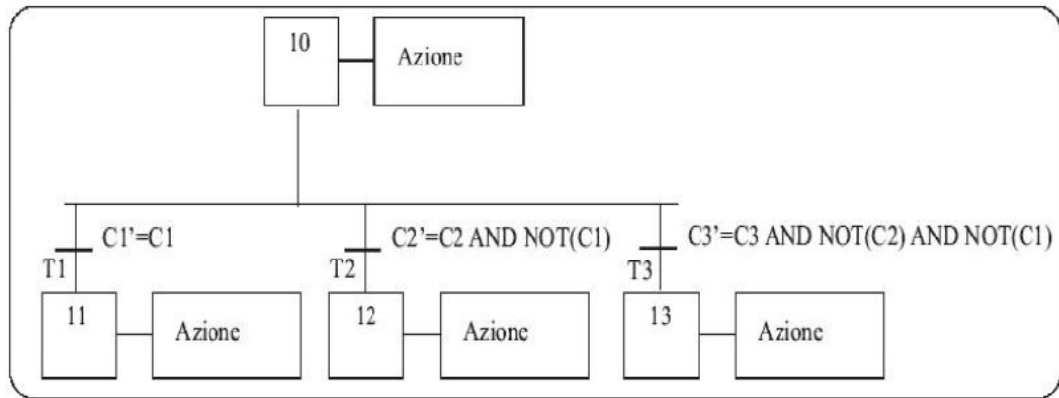
Riferendoci alla figura, in base al verificarsi di una delle condizioni  $\{C1, C2, C3\}$  sarà attivato uno dei passi  $\{X11, X12, X13\}$ .

Tuttavia, se le condizioni non sono implicitamente mutuamente esclusive (es. possono essere vere  $C1$  e  $C2$  nello stesso momento), è possibile che vi sia una attivazione contemporanea (involontaria) dei passi ed una conseguente attivazione (errata) di percorsi simultanei.

Ing. Elena Mainardi

SFC

## Divergenza non ambigua



In questo caso l'SFC originario, in cui erano presenti le condizioni  $C1, C2, C3$  è stato modificato in modo che le condizioni siano riscritte come:

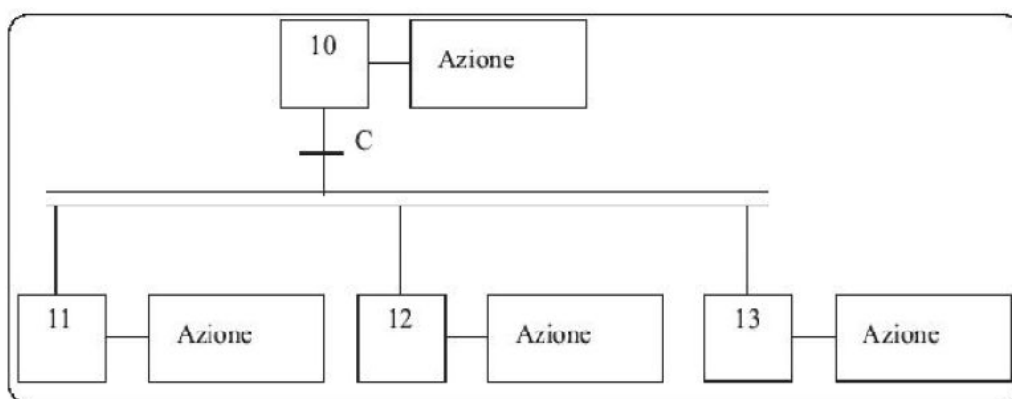
$C1' = C1$

$C2' = C2 \text{ AND NOT}(C1)$

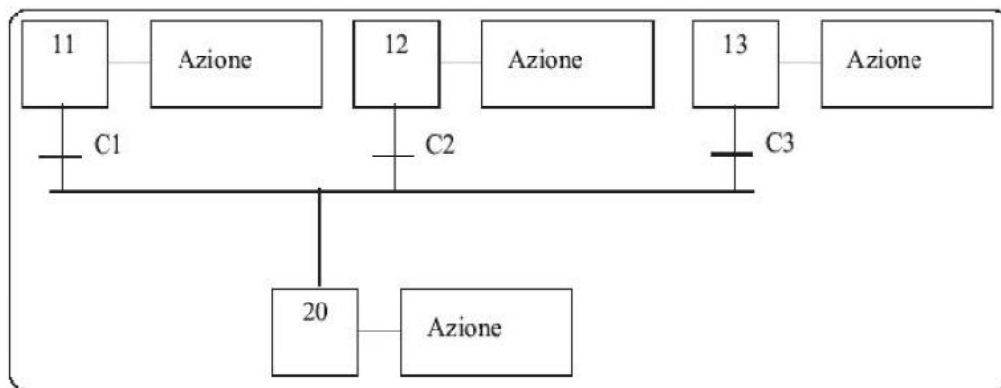
$C3' = C3 \text{ AND NOT}(C2) \text{ AND NOT}(C1)$

In questo modo si rendono mutuamente esclusive le tre transizioni, introducendo implicitamente una priorità: la transizione T1 verrà comunque eseguita una volta che C1 sia vera, inibendo le altre due transizioni, T2 viene eseguita se C2 è vera e C1 falsa, mentre T3 sarà eseguita solo se C3 è vera e le altre false.

## Divergenza simultanea (Parallelismo)



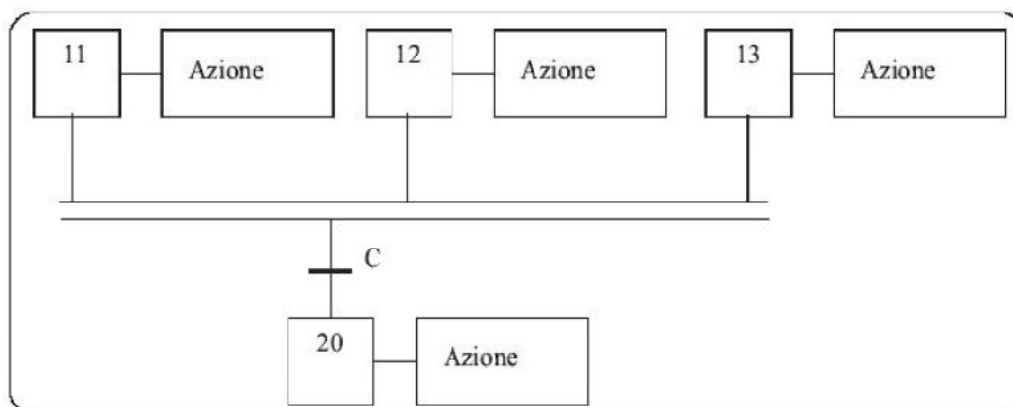
## Convergenza opzionale



Siccome nella divergenza opzionale solo un percorso viene attivato, le transizioni (e quindi le condizioni) debbono essere specificate per tutte le sequenze alternative.

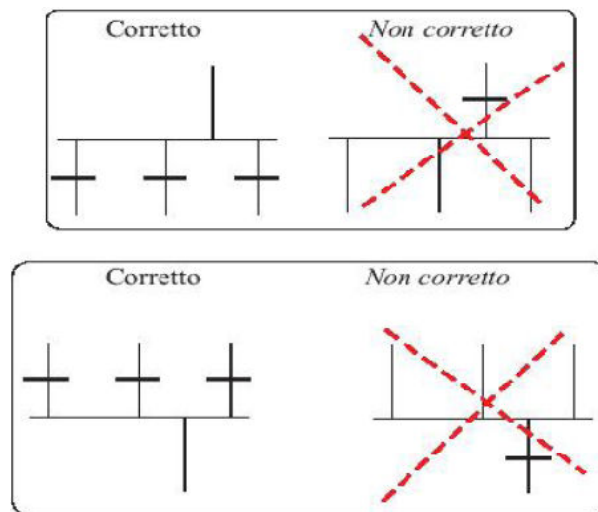
Una divergenza opzionale si può richiudere con una convergenza opzionale o con i rami divergenti che approdano a stati differenti e non si richiudono

## Convergenza simultanea (Sincronizzazione)



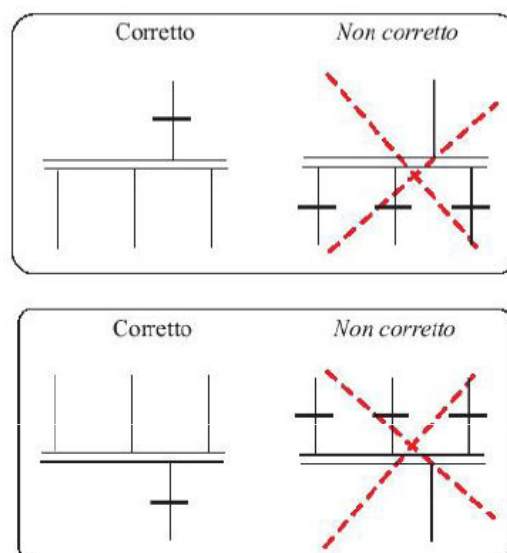
N.B.

Se ho una divergenza simultanea, questa verrà **OBBLIGATORIAMENTE** richiusa da una convergenza simultanea



### Collegamenti multipli - I

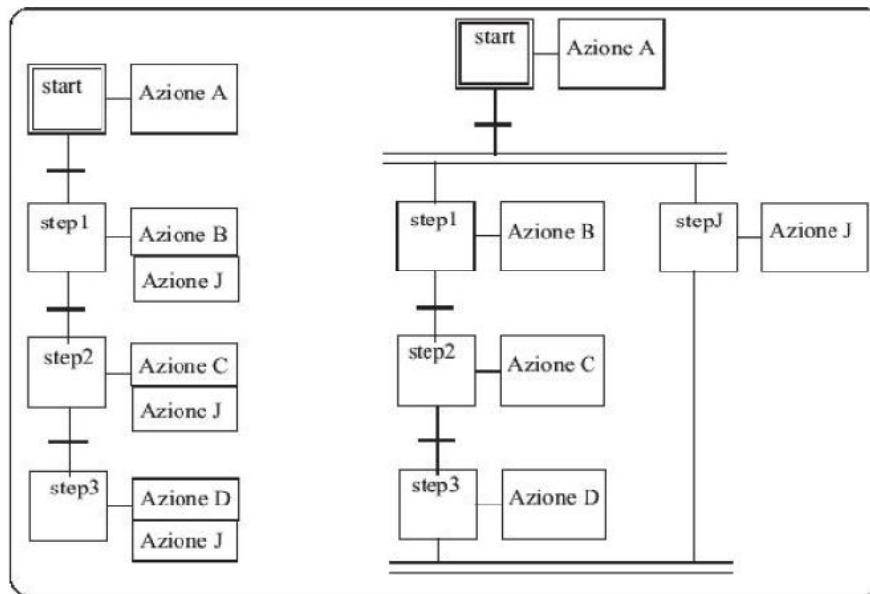
- Le divergenze opzionali non possono mai essere precedute da una sola transizione.
- Le convergenze opzionali non possono mai essere seguite da una sola transizione.



### Collegamenti multipli – II

- Le divergenze simultanee debbono essere precedute da una sola transizione.
- Le convergenze simultanee debbono essere precedute da una sola transizione.

## Definizione alternativa delle azioni prolungate

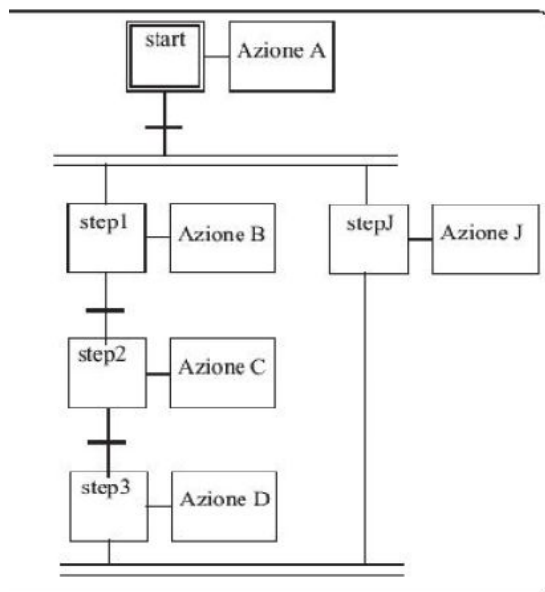


Una frequente difficoltà incontrata quando si traccia lo schema SFC concerne la rappresentazione di Azioni il cui effetto deve perdurare durante un certo numero di passi consecutivi. In tal caso è possibile raccomandare due tipi di esecuzione:

- Effetti prolungati di azioni continue non memorizzate, mediante (i) la ripetizione dell'Azione in tutti i Passi interessati, oppure (ii) utilizzando la struttura delle sequenze simultanee.
- Effetti prolungati da Azioni memorizzate. Le Azioni sono indicate nei Passi in cui hanno inizio (dove cioè se ne effettua l'attivazione allo stato logico "vero"), o dove si esauriscono (disattivazione allo stato logico "falso").



## Sincronizzazione



La sequenza di destra (step j) è obbligata ad attendere che la sequenza di sinistra (step1 + step2 + step3) sia giunta in step3 prima di poter proseguire.

## Diagnosi degli errori

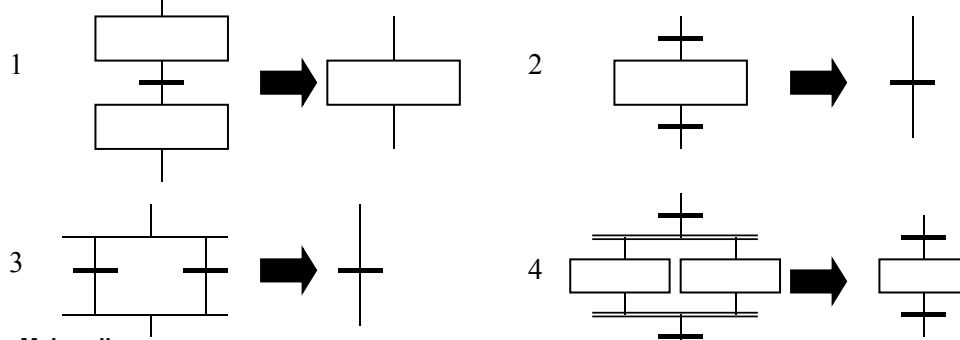
Esiste un metodo puramente grafico, che prescinde dal nome dei passi e delle transizioni, dalle azioni ad esse associate e via dicendo, per vedere se uno schema SFC segue tutte le regole della sintassi SFC.

Se, a partire dal nostro schema, disinteressandoci di cosa esso effettivamente faccia e considerando solo graficamente gli step e le transizioni:

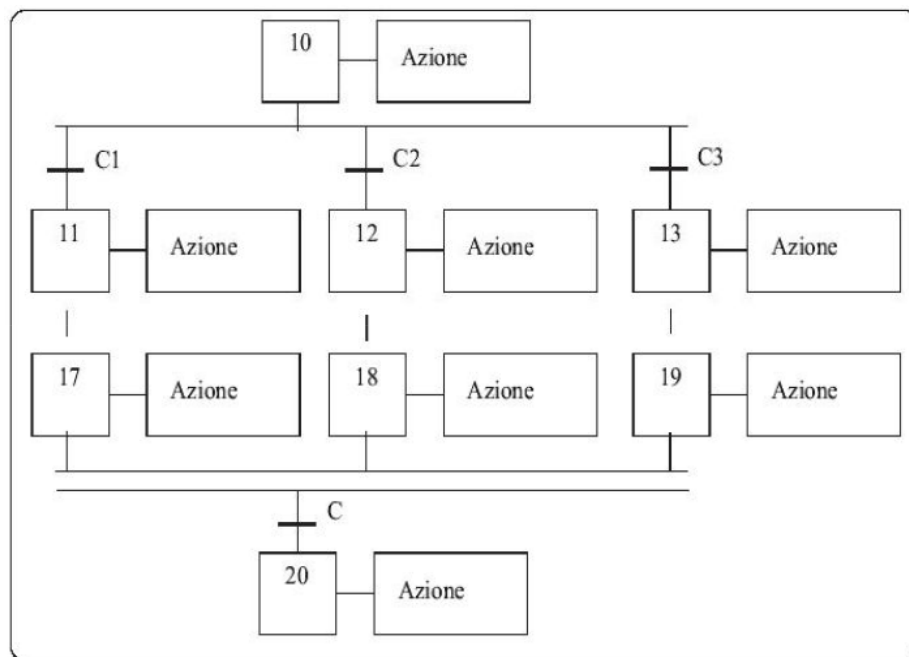
- 1 Ad ogni sequenza passo-transizione-passo sostituisco un passo
- 2 Ad ogni sequenza transizione-passo-transizione sostituisco una transizione
- 3 Sostituisco ogni divergenza opzionale con una singola transizione
- 4 Sostituisco ogni divergenza simultanea con un passo

e alla fine tutto lo schema si riduce ad un solo step, oppure ad uno step con una transizione, allora esso è formalmente corretto.

Ovviamente la correttezza formale non garantisce la correttezza logica.



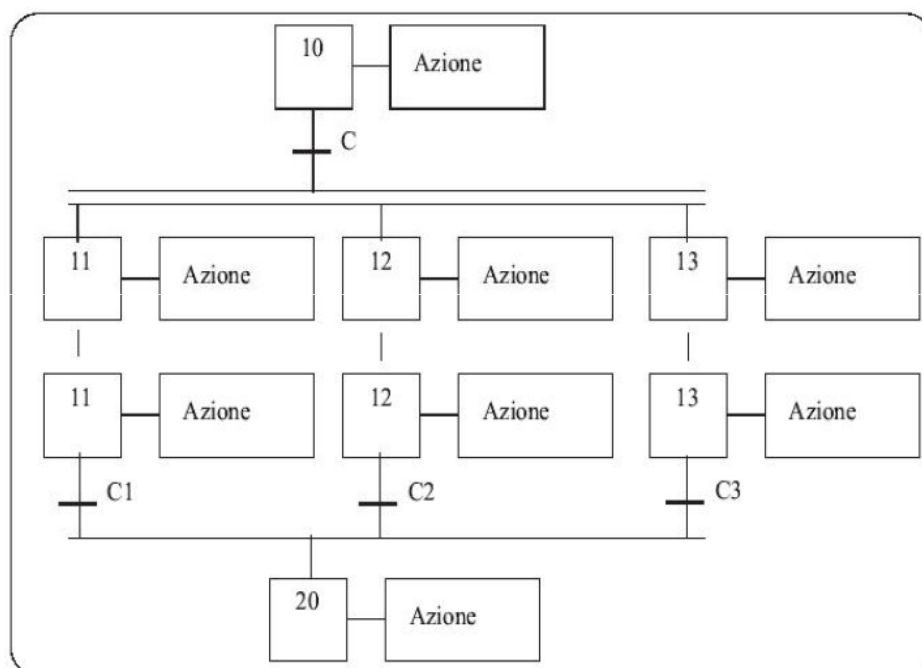
## Possibili errori - I



Ing. Elena Mainardi

SFC

## Possibili errori - II



Ing. Elena Mainardi

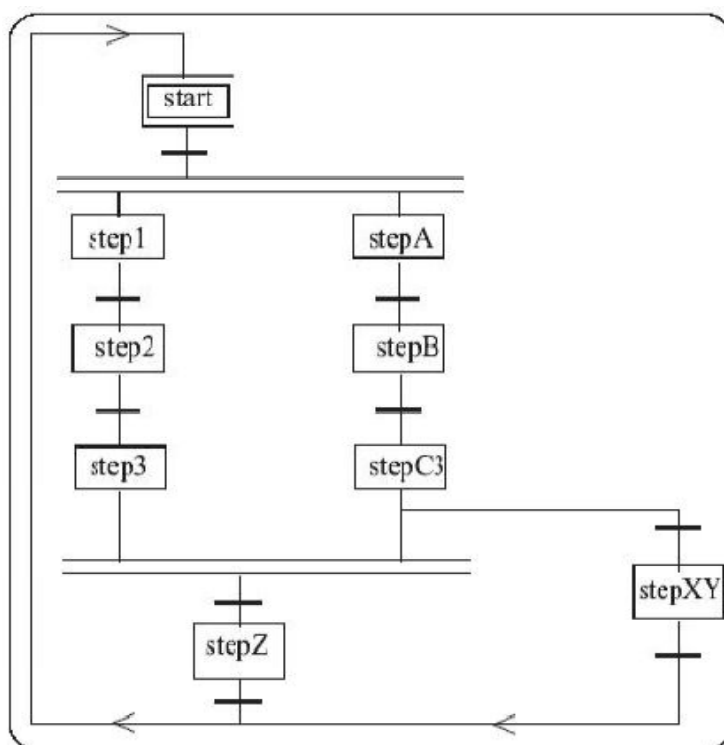
SFC

## Possibili errori - III

Se applicassi il metodo grafico di riduzione, non mi risulterebbe solo uno step, perché il passo stepXY è messo in posizione sbagliata. Non si può scavalcare una convergenza simultanea!!!

### Nota Bene:

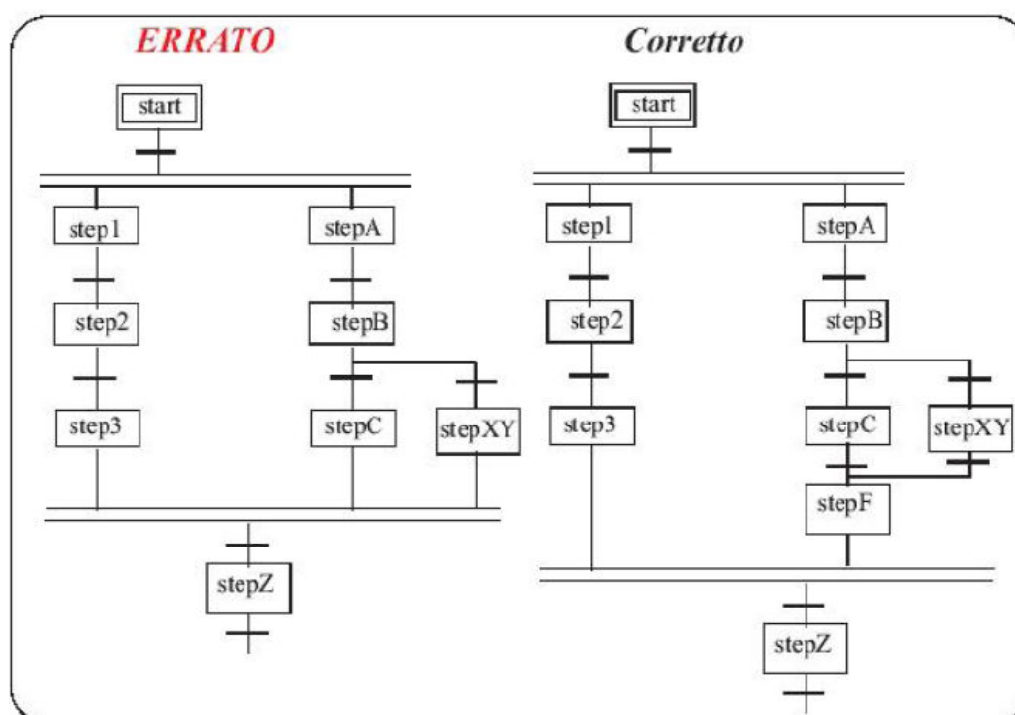
Alcuni ambienti di sviluppo consentono di implementare formalmente un codice del genere, ma comunque il programmatore deve evitarlo, in quanto si generano errori e comportamenti imprevedibili



Ing. Elena Mainardi

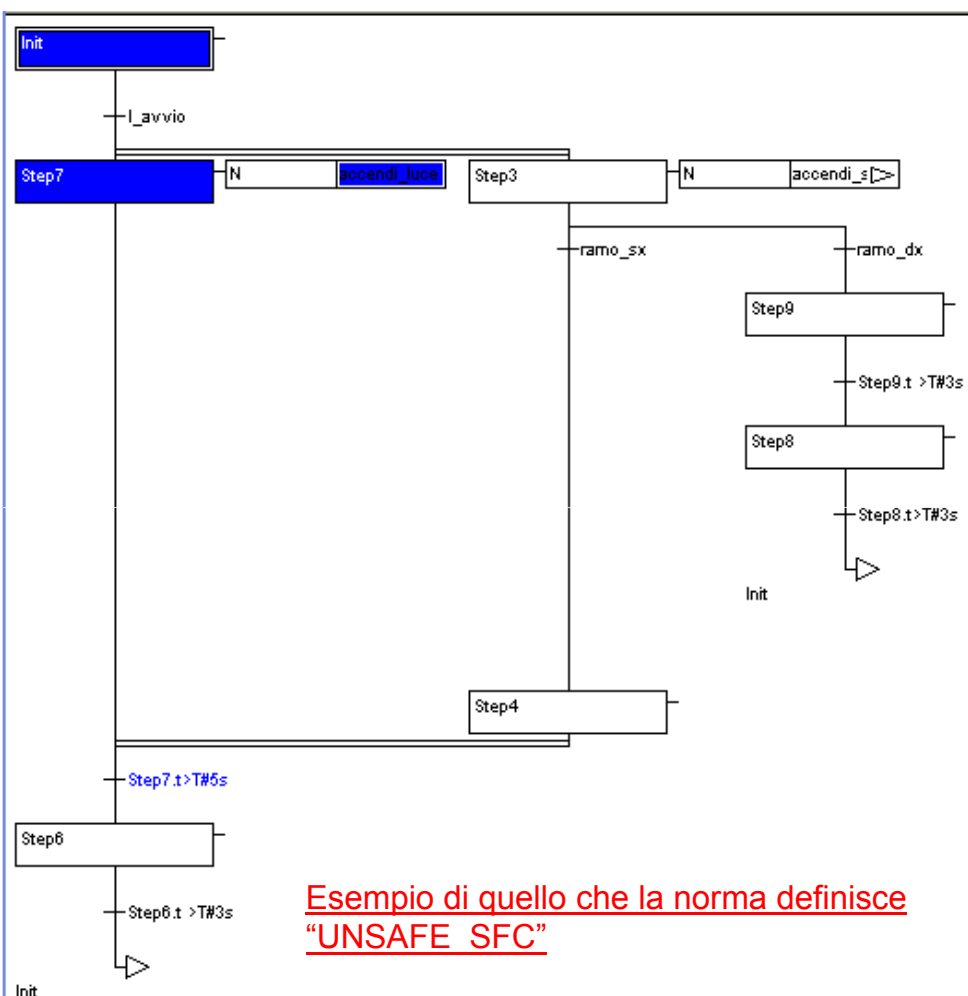
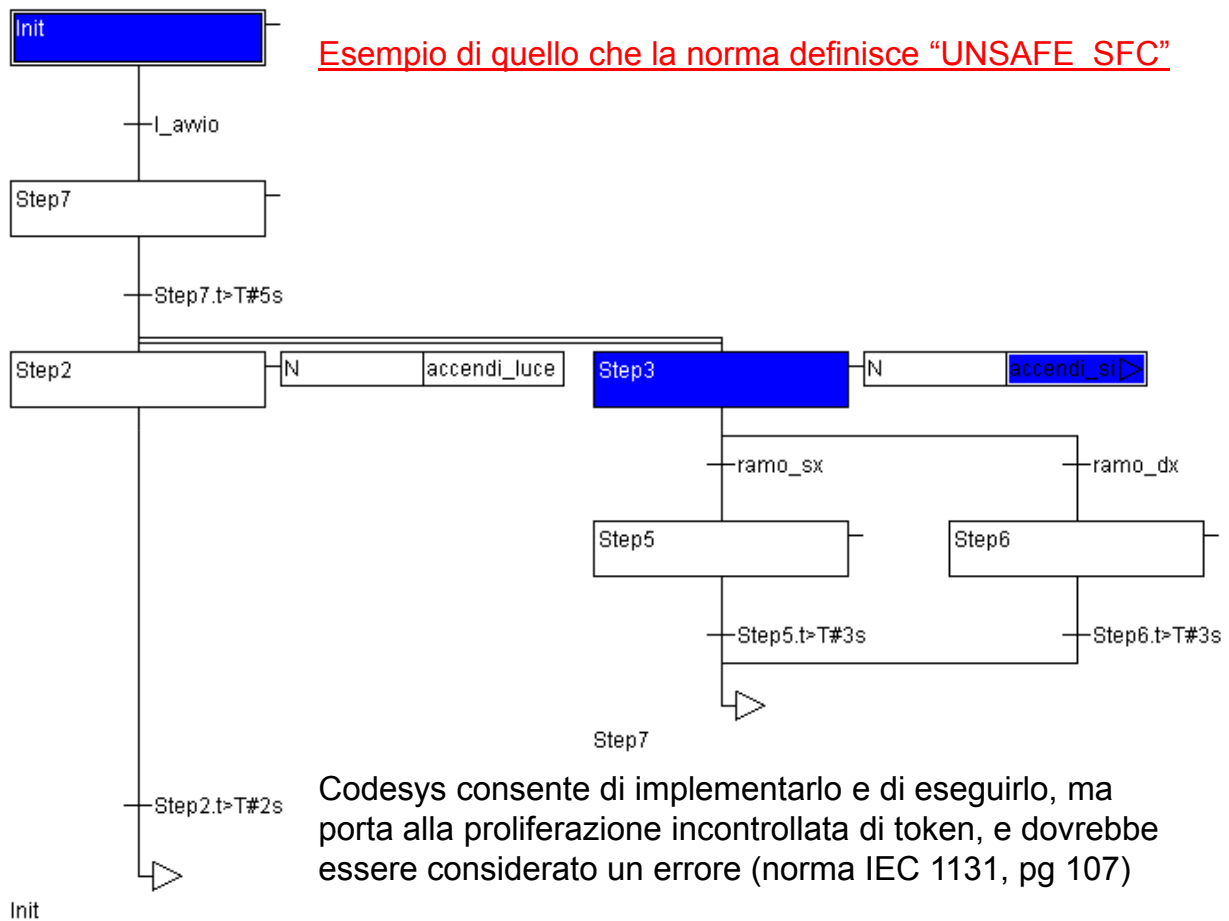
SFC

## Possibili errori - IV



Ing. Elena Mainardi

SFC



Codesys consente di implementarlo e di eseguirlo, ma mentre lo eseguo, se è vera ramo\_dx arrivo ad essere contemporaneamente nello stato INIT e nello stato Step7 senza che la divergenza simultanea si sia richiusa: l'esecuzione è in contemporanea anche nello stato Step7, perché la divergenza attendeva che il ramo di destra arrivasse in Step4 per richiudersi, cosa che non può avvenire se ramo\_dx è vera → questo porta ad un comportamento EERATO!!!

## Ricapitolando:

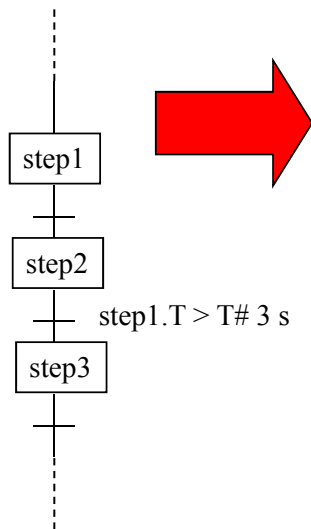
- La norma stabilisce alcune regole formali che non possono essere violate, in quanto ogni ambiente di sviluppo le segnala come errori e non consente di implementarle (ad esempio è impossibile mettere due stati conseguenti senza una transizione in mezzo, l'ambiente di sviluppo non lo consente fisicamente)
- La norma stabilisce altre regole che non dovrebbero essere infrante in quanto porterebbero a comportamenti imprevedibili, e sono dunque considerate una fonte di ERRORE. UN programmatore può violarle, nel senso che l'ambiente di sviluppo consente di farlo e non segnala errori, però ciò causa un andamento pericoloso del codice creato

## Esempio analogo nel linguaggio C

```
For (i=0; i<10; i++)  
{  
    i = i-1;  
}
```

Il compilatore non lo segnala come errore, però non è affatto corretto modificare il contatore di un ciclo for all'interno del ciclo stesso

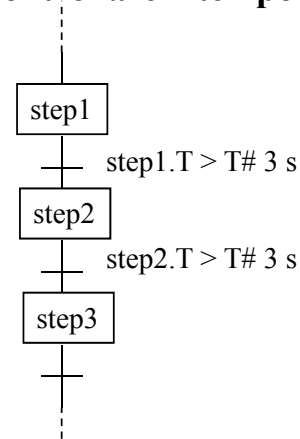
## Altri possibili errori



sbagliata

**NO!!!!**

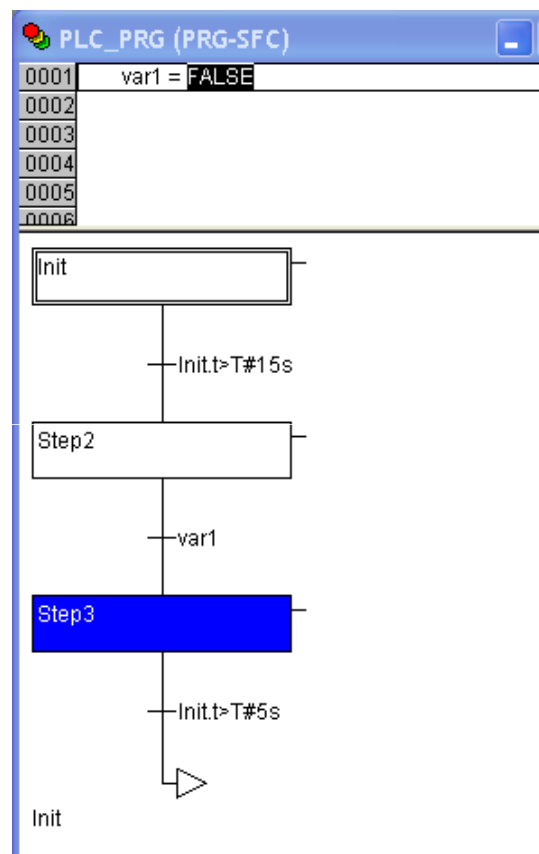
**Il timer di un passo si può usare SOLO nella transizione presente subito dopo quel passo, oppure per controllare il tempo di un'altra sequenza**

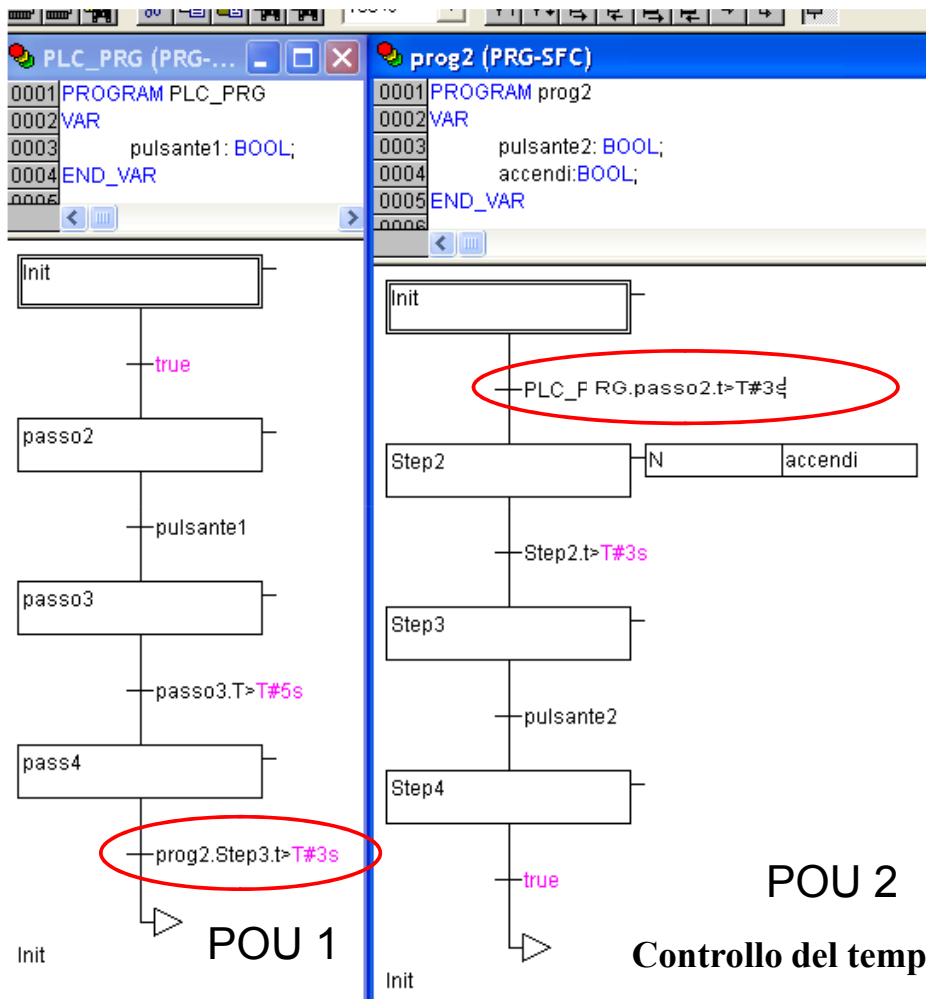


corretta

## Altri possibili errori

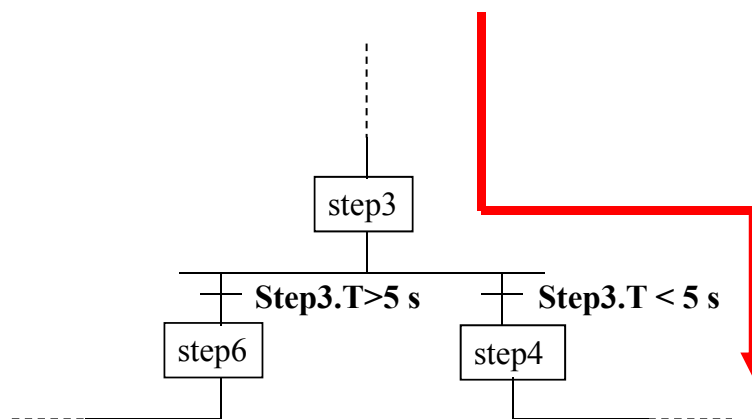
Codesys consente di implementarlo e di eseguirlo, ma se arrivo in Step3 vi permango all'infinito → ERRORE!!!





62

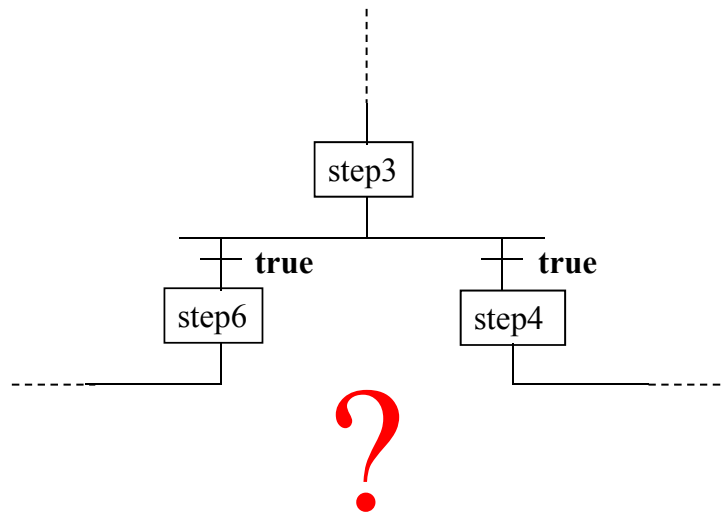
## Altri possibili errori



**NO!!!!**

**Vado sempre giù per il ramo destro**

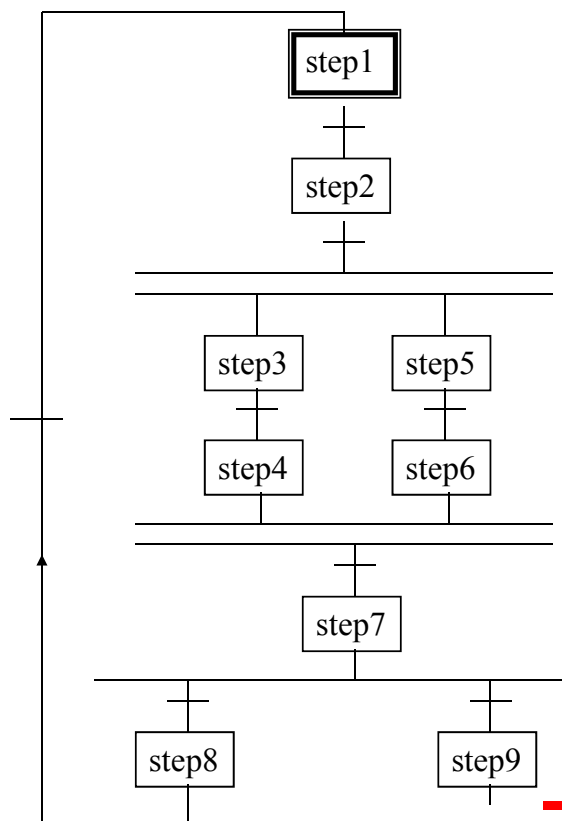
## Altri possibili errori



**NO!!!!**

**Ha un comportamento imprevedibile**

## Altri possibili errori



**NO!!!**

Non si possono lasciare  
dei rami a “penzoloni”  
nel vuoto!!

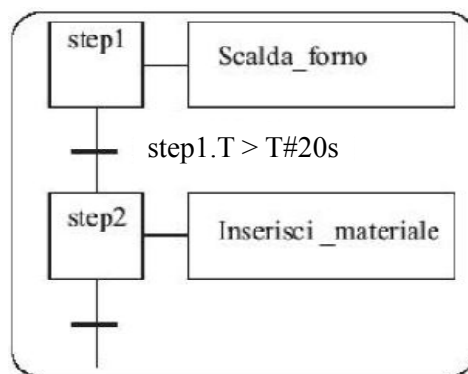
Ci si deve sempre  
ricongiungere a qualcosa



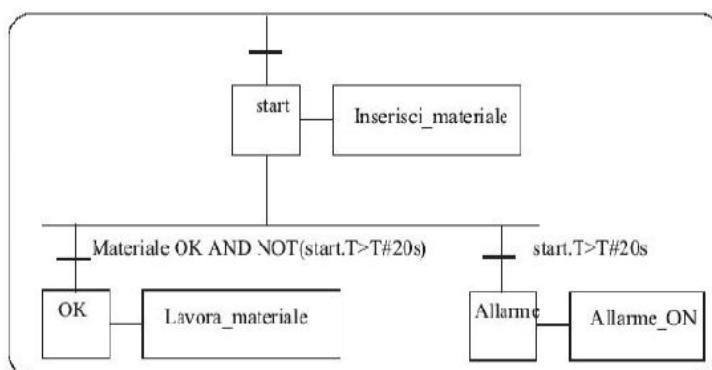
## Variabili temporali.

Ad ogni passo è implicitamente associato un temporizzatore:

- $\text{nomepasso.T}$



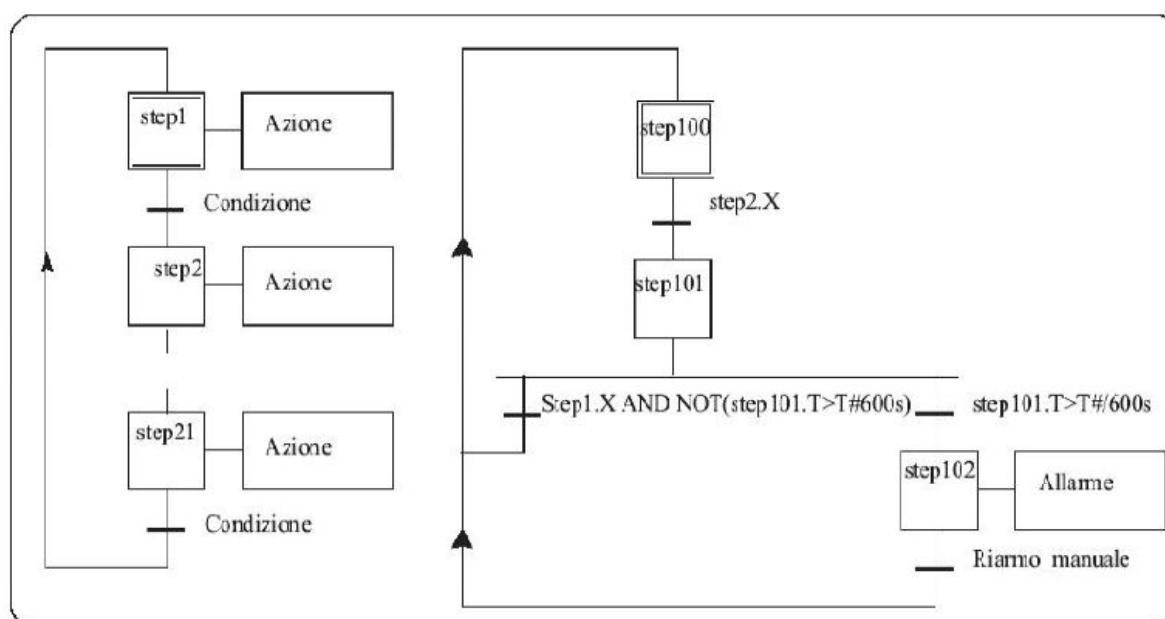
## Watchdog timer software



Ing. Elena Mainardi

SFC

## Watchdog timer di una sequenza.



Ing. Elena Mainardi

SFC

## Watchdog timer

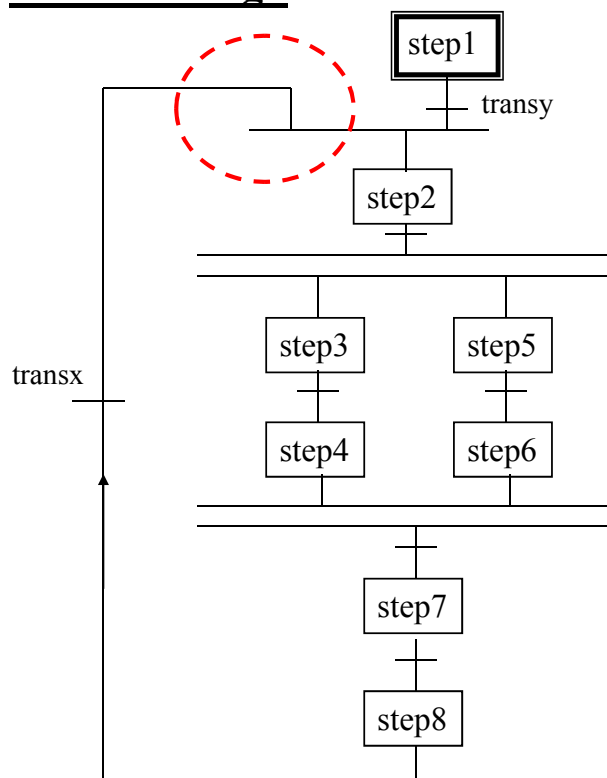
Il watchdog è un dispositivo che può essere utilizzato per segnalare anomalie rispetto alla sequenza programmata utilizzando informazioni temporali.

I due esempi sono relativi alle situazioni:

1. Allarme dovuto al mancato funzionamento di un particolare componente.
2. Allarme generale dovuto al mancato funzionamento di un qualche componente che produce il fallimento globale della sequenza.

Occorre notare che la tipologia di watchdog di cui sopra differisce da quella di un PLC, infatti nel PLC il watchdog serve a segnalare il mancato rispetto del tempo di ciclo del programma in esecuzione, mentre qui si è solo implementata una funzione stile watchdog software

## Alcuni dettagli



La parte evidenziata è solo un modo grafico per rendere più leggibile l'SFC.

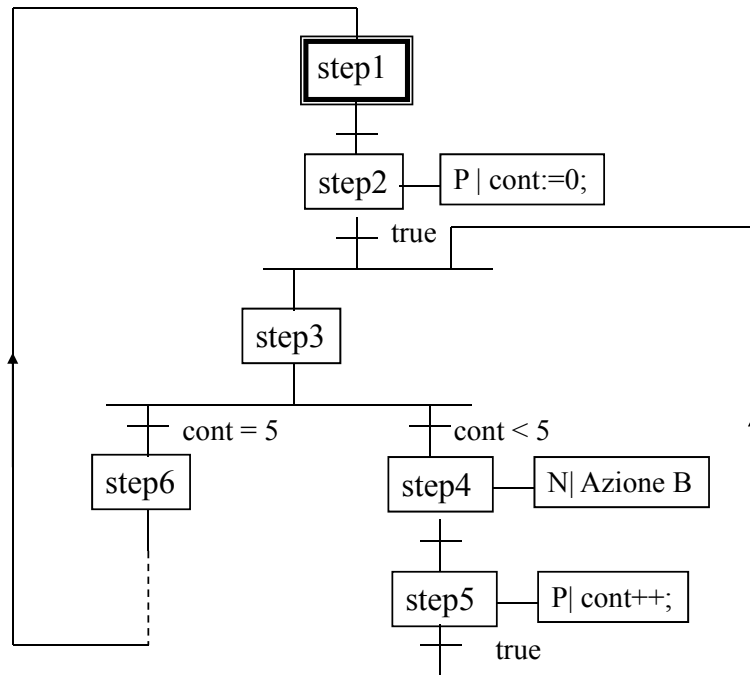
Non è una transizione!!!

Semplicemente:

- 1) quando il programma si trova nello stato step1 e si attiva la transizione transy, il programma va nello stato step2
- 2) quando il programma si trova nello stato step8, se si attiva la transizione transx il programma torna allo stato step2 e riparte da lì.

## Alcuni dettagli

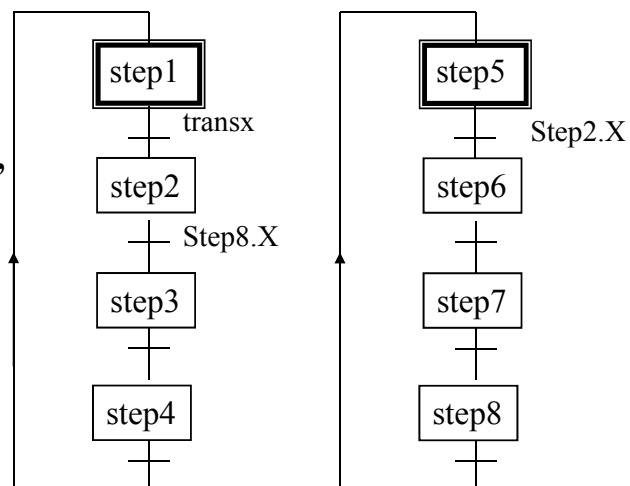
### Come fare un contatore



L'azione B viene eseguita 5 volte.  
Bisogna ricordarsi di inizializzare il contatore!!!!

## Alcuni dettagli

### Gestire più sequenze in parallelo, in modo "incrociato"



Quando il programma viene avviato, se la transizione transx è vera si va nello stato step2. Una volta giunti qui, la variabile step2.x è vera, quindi si avvia anche l'SFC di destra. Durante la sua evoluzione, prima o poi raggiungerà lo stato step8, quindi la variabile step8.x diventerà vera. A questo punto si attiverà quindi la transizione che, nell'SFC di sinistra, farà passare dallo stato step2 allo stato step3.

**Come si scrive un programma in SFC? → con appositi editor grafici**

