Sequential Functional Chart (SFC)

Cesare Fantuzzi e Marcello Bonfè

Dipartimento di Ingegneria, Università di Ferrara

Versione 1, 5 Dicembre 2000

Capitolo 1

La descrizione di una sequenza di controllo

In questo capitolo viene introdotto il formalismo **Sequential Functional Chart** (SFC) che consente di descrivere il funzionamento di machine sequenziali. Infatti, molte macchine automatiche presentano un comportamento sequenziale, in cui le operazioni che esegue la macchina possono essere associate a passi operativi alternati a transizioni. Tale formalismo può essere utilizzato per la descrizione e documentazione del software di controllo, oppure utilizzato direttamente come linguaggio di programmazione, se il sistema di sviluppo lo consente.

Lo SFC è un linguaggio grafico la cui comprensione è molto intuitiva, per cui può essere utilizzato efficaciemente anche per la documentazione del software di controllo

1.1 Introduzione

Gli applicativi di uso industriale sono caratterizzati dalla esecuzione ciclica del software di controllo, necessario a causa della natura *real–time* del sistema di controllo per la macchina automatica. L'esecuzione di ciascun ciclo di programma comprende in genere tre fasi:

1. *Acquisizione sensori.*
2. *Elaborazione di un algoritmo di controllo.*
3. *Attuazione dei segnali di controllo.*

Nella maggior parte dei PLC industriali le fasi di acquisizione ed attuazione dei segnali sono eseguiti in modo trasparente dal sistema operativo, mentre il progetto dell'algoritmo di controllo è a carico del programmatore.

Una seconda caratteristica importante del software di controllo è la necessità di considerare stati interni di elaborazione. Infatti una macchina automatica è un sistema con un comportamento spiccatamente sequenziale (dotato di stato interno), e quindi per poterla controllare efficaciemente occorre tenere traccia dei passi di elaborazione eseguiti dalla macchina all'interno del ciclo produttivo. In sostanza:

- I sistemi automatici sono rappresentabili mediante passi sequenziali descrivibili mediante sistemi a stati finiti.
- I sistemi di controllo debbono replicare questa proprietà in modo che lo stato di esecuzione del programma di controllo corrisponda con lo stato di evoluzione della macchina.

Infatti, in generale un software industriale interagisce con un sistema dinamico modellabile mediante una sequenza di stati di lavorazione. Lo stato interno del software deve essere congruente con

lo stato del processo controllato (un classico esempio consiste nell'avviamento a *caldo* e avviamento a *freddo* di una macchina automatica).

▽ *Esempio: Stati di funzionamento CPU S7*

Come esempio significativo, consideriamo il funzionamento della CPU S7 della Siemens [1] in cui vengono descritti gli stati di funzionamento della CPU (cap. 9). La CPU può essere in 4 stati (si veda figura 1.1), STOP, AVVIAMENTO, RUN e ALT. In ciascuno di questi stati il PLC ha un determinato funzionamento e comportamento rispetto ai segnali che riceve dal campo.

Nello stato di funzionamento STOP, la CPU verifica la presenza di tutte le unità configurate o utilizzate tramite lindirizzamento di default e pone la periferia in uno stato di base predefinito. Nello stato di funzionamento STOP, il programma utente non viene elaborato.

Per quanto riguarda lo stato di funzionamento AVVIAMENTO, occorre distinguere tra nuovo avviamento e riavviamento:

- con il nuovo avviamento, il programma viene elaborato nuovamente dall'inizio con una "impostazione di base" dei dati di sistema e delle aree di operandi utente (temporizzatori, contatori e merker non ritentivi vengono resettati).
- con il riavviamento, lelaborazione del programma viene ripresa dal punto in cui era stata interrotta (temporizzatori, contatori e merker non vengono resettati). Il riavviamento è possibile solo nelle CPU S7-400.

Nello stato di funzionamento RUN, la CPU elabora il programma utente, aggiorna gli ingressi e le uscite, elabora interrupt e messaggi di errore. Nello stato di funzionamento ALT, l'elaborazione del programma utente viene fermata ed è possibile effettuare il test del programma utente passo per passo. Lo stato di funzionamento ALT si può raggiungere solo durante il test con il PG (dispositivo di programmazione). In tutti questi stati di funzionamento la CPU è in grado di comunicare tramite linterfaccia MPI.

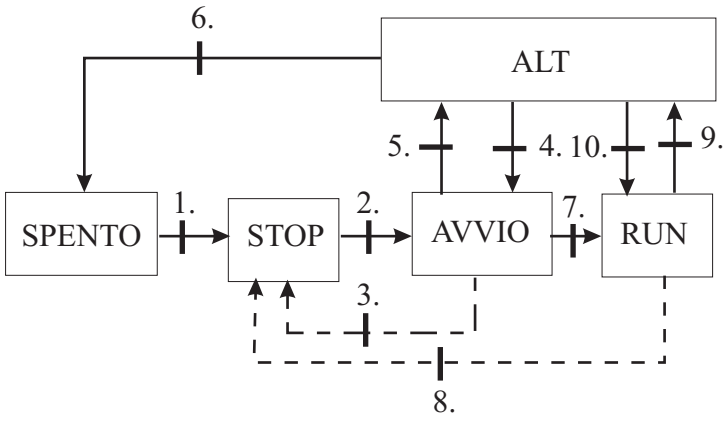


Figura 1.1: Stati di funzionamento e transizioni di stato di funzionamento di un PLC Siemens Step 7 (*Pag. 9.2 Manuale di Programmazione Step 7*)

△

La modellazione mediante sequenze (combinazioni di stati logici di funzionamento e transizioni fra stati successivi) presenta alcuni vantaggi, tra i quali possiamo riassumere:

- In ogni stato logico di funzionamento solo *un sottoinsieme di sensori ed attuatori* sono di interesse, per cui possiamo *semplificare* notevolmente la logica di comando legandola allo stato in elaborazione al momento.

- Risulta agevole eseguire la *verifica del software*, in quanto è sufficiente seguire i passi del programma e parallelamente gli stati di funzionamento della macchina.
- Risulta agevole valutare le possibili eccezioni (*errori ed allarmi*) che possano avvenire nei vari passi operativi della macchina, e quindi è più semplice produrre il codice per opportune contromisure.

La descrizione dei passi operativi e delle transizioni di una macchina automatica possono venire implementate mediante un efficacie formalismo grafico denominato **Sequential Function Chart (SFC)**, definito da una normativa internazionale (norma IEC 1131-3).

Questo formalismo è orientato alla descrizione grafica del funzionamento della macchina automatica, particolarmente adatto alla fase di analisi e documentazione, ma che trova un valido utiizzo anche nella implementazione stessa dell'applicativo. Il formalismo facilita la scomposizione gerarchica *top-down* del funzionamento della macchina, e quindi risulta adatto alla strutturazione e suddivisione dell'applicativo generale per il controllo di macchina.

Volendo riassumere le caratteritiche peculiari del formalismo SFC possiamo affermare che:

- Lo SFC permette la descrizione del funzionamento della macchina attraverso un **linguaggio formale**. Il progetto di un 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 su un progetto comune. Tali persone necessitano di un linguaggio formale comune attraverso cui possono comunicare efficaciemente.
- Lo SFC facilita l'**approccio top-down** al progetto, in cui il funzionamento della macchina viene scomposto in passi fondamentali analizzati in fasi successive, riducendo così la complessità di ogni singolo elemento che compone la logica del sistema di controllo.
- Lo SFC permettere di evidenziare il **Comportamento sequenziale** della macchina. Infatti la descrizione ingresso–uscita della macchina richiede di fornire una configurazione di controllo per una qualunque combinazione degli stimoli (ingressi acquisiti dai sensori). Al contrario, la descrizione ingresso-stato-uscita richiede semplicemente di descrivere una configurazione di controllo semplificata per ciascun passo operativo della macchina, portando ad una descrizione globale molto più semplice.

1.2 Descrizione del formalismo Sequential Functional Chart (SFC)

Il formalismo più rigoroso in senso scientifico per la decrizione di un sistema sequenziale mediante stati di funzionamento e transizioni (cioè del processo di evoluzione tra questi stati) è costituito dalle **Reti di Petri** (Petri Net). Tuttavia questo formalismo, seppure semanticamente corretto e ben strutturato, risulta di difficile applicazione in ambiente industriale.

Un consorzio di industrie ed enti francesi hanno proposto una versione semplificata delle reti di petri chiamata **Grafcet**[1], i cui elementi sintattici sono stati raccolti nel 1988 all'interno dello standard IEC848.

Questo linguaggio viene direttamente utilizzato da alcuni PLC (Telemecanique) per la descrizione di operazioni sequenziali a livello dell'applicativo di controllo.

Nel 1992 il Grafcet viene poi portato all'interno dei linguaggi della norma IEC1131-3 con qualche modifica per una migliore integrazione con gli altri linguaggi definiti della normativa, dando origne allo *Sequential Function Chart*.

Uno schema SFC è descritto mediante un costrutto grafico i cui elementi costitutivi fondamentali sono:

[1] Sito internet di riferimento `http://www.lurpa.ens-cachan.fr/grafcet.html`

- Gli *Stati logici* di funzionamento, che descrivono il comportamento della macchina in corrispondenza di ben determinati stati di lavoro. Le uscite da attuare sull'impianto dipendono non solo dagli ingressi, ma anche dal particolare stato di funzionamento attivo in quell'istante.
- Le *Transizioni* tra questi stati logici, condizionati dal valore che assumono determinati segnali sensoriali.

Per capire meglio tali concetti consideriamo il problema di progettare un algoritmo di controllo per un sistema di timbratura automatico mostrato in figura 1.2, il cui funzionamento può venire descritto come segue:

> La timbratrice deve ripetitivamente salire e scendere per effettuare timbri su buste che vengono poste sul supporto inferiore da un'altra macchina. 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 fine–corsa. 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. A questo punto la macchina timbratrice segnala che la busta può essere rimossa dalla sede di lavorazione.

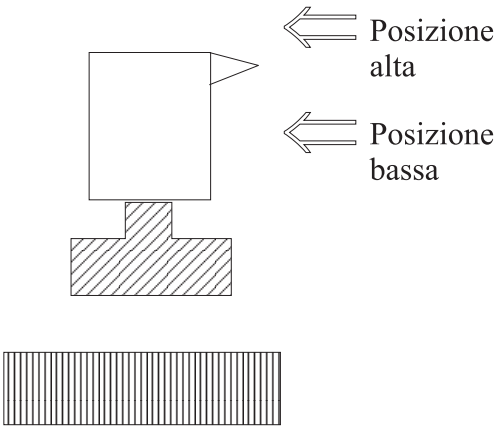


Figura 1.2: Una macchina automatica per la timbratura

Lo schema SFC per il controllo del sistema può essere costruito come rappresentato in figura 1.3.

Il progetto dell'algoritmo di controllo si ottiene isolando le azioni sequenziali che deve compiere la macchna per raggiungere gli obbiettivi produttivi, è cioè (i) *l'attesa di una busta*, (ii) la *discesa del timbro* e la (iii) *risalita del timbro*. A ciascuna azione facciamo corrispondere uno *stato di elaborazione*, in cui vengono valutati solo i segnali di ingresso-uscita necessari alla esecuzione di quella azione. L'evoluzione tra un passo ed il successivo avviene in modo sincrono al termine dell'azione corrispondente al passo. In sostanza, tale evoluzione è legata all'attivazione di un segnale di ingresso, che in genere è il risultato dell'esecuzione dell'azione stessa (nel nostro caso la (i) *presenza della busta*, l'attivazione dei fine corsa (ii) *alto* e (iii) *basso*).

Riassumendo, i componenti basilari di uno schema SFC sono elencati nella seguente tabella (si veda anche la figura 1.4).

Il *passo* (o fase o tappa o stato)	Ad ogni passo viene associato un'*azione*
La *transizione*	Ad ogni transizione viene associata una *ricettività* (o condizione)
Il *collegamento orientato*	Collega due stati consecutivi mediante una transizione.

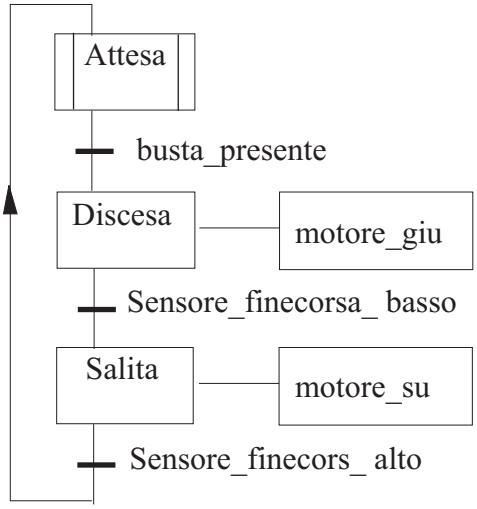


Figura 1.3: Lo schema SFC della macchina automatica per la timbratura

1.3 La sintassi dello Sequential Functional Chart

Vediamo ora di approfondire brevemente quali sono gli aspetti fondamentali della sintassi dello SFC, rimandando al testo della norma per ulteriori approfondimenti [2].

Uno degli elementi fondamentali della sintassi dello SFC è il passo. I Passi (*Steps*) sono caratterizzati da:

- 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**. Nel primo caso le azioni ad esso associate saranno eseguite, nel secondo le azioni saranno disabilitate.
- Ad ogni istante il sistema di controllo è descrivibile attraverso:
  - I **passi** attivi.
  - L'insieme delle variabili **interne** e di **uscita**.

Le transizioni sono il secondo elemento di base della sintassi dello SFC. Le caratteristiche delle transizioni si possono riassumere come segue:

- Una **transizione** rappresenta la condizione secondo la quale il controllo passa da un passo (passi) **predecessori** ad un passo (passi) **successori** secondo la topologia stabilita dalla rete di **connessioni**.
- La **transizione** è graficamente rappresentata da una linea **orrizzontale** che taglia il **collegamento** verticale.
- Ogni transizione deve avere associata un **condizione** dalla cui valutazione deve risultare una singola **espressione booleana**.
- Una condizione sempre vera deve essere identifcata dal simbolo "1" oppure dalla parola chiave "TRUE".

Passi e *Transizioni* debbono sempre alternarsi, in altre parole:

- Due passi consecutivi debbono essere sempre separati da una transizione.

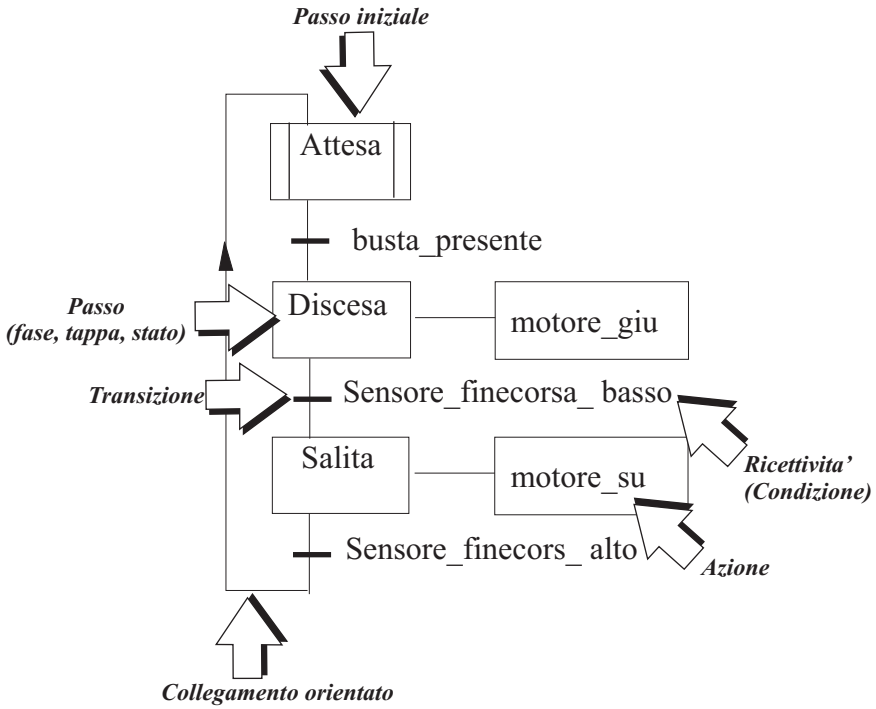


Figura 1.4: Gli elementi di base dello SFC in relazione all'esempio della macchina automatica per la timbratura

- Due transizioni consecutive debbono essere separate da un passo.

Come regola di stesura dello schema, è importante sottolineare che uno schema SFC si sviluppa sempre in senso veritcale, in cui l'evoluzione dei passi attivi avviene dell'alto verso il basso. I collegamenti arrivano ai passi o se ne departono in posizione verticale (fig. 1.5).

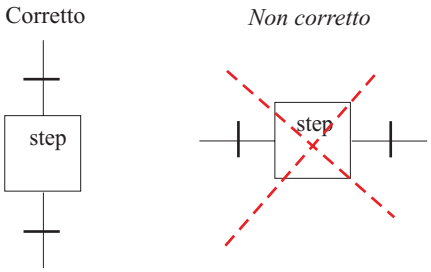


Figura 1.5: I collegamenti sono semre disegnati in senso verticale

L'evoluzione di un automatismo descritto da uno schema SFC può essere visivamente interpretato come una rete di "contenitori" (passi) collegati fra di loro attraverso le transizioni. Lo stato di attivazione di un passo è stabilito dal possesso di un "gettone" (**token**): solo i passi possessori di un gettone sono attivi. Il gettone viene passato da uno stato attivo al successivo in base ai rispettivi collegamenti ed alle transizioni convalidate.

Una **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.

Una azione è definita attraverso tre elementi (Fig. 1.6):

- Il **qualificatore** dell'azione, che definisce il tipo della azione tra intraprendere (se impulsiva, continua, temporizzata, etc.).
- L'**identificativo** dell'azione stessa. L'implementazione effettiva della azione può avvenire mediante diversi meccanismi che verranno discussi oltre. In generale essa sarà implementata tramite un sottoprogramma di una qualche complessità, per cui l'effettiva implementazione della azione è rimandata ad un altro punto del programma. Il collegamento tra collocazione della azione e sua implementazione è reso possibile attraverso un l'identificativo.

  In casi particolarmente semplici, l'azione può venir descritta direttamante in corrispondenza dell'identificativo nell'**action block**, o, se l'azione corrisponde semplicemente a modificare una variabile booleana, l'identificativo e l'implementazione della 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'indentificativo.

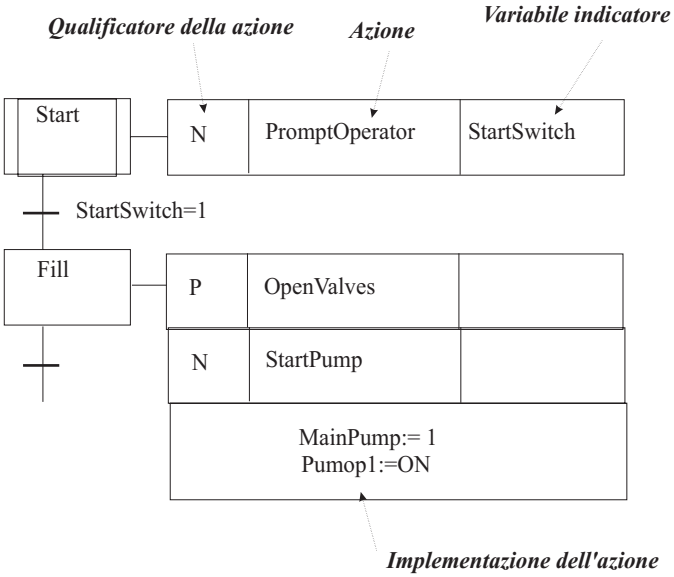


Figura 1.6: La definizione di una azione

Una azione può essere implementata utilizzando i diversi linguaggi definiti dalla norma IEC 61131-3 , in particolare:

- 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** (FB).
- Un **Sequential Function Chart** (SFC).

In termini generali le azioni rimangono attive fintanto che i corrisponednti passi sono attivi. Nei casi specifici in cui occorra una maggiore fessibilità sulla definizione delle azioni vengono utilizzati una serie di qualificatori, definiti nella tabella 1.1.

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

Tabella 1.1: I qualificatori delle azioni

Azioni continue

Le azioni continue permangono attive per tutto il tempo in cui il passo corrispondente è attivo (Fig. 1.7).

Un caso molto comune del progetto del controllo di una automatismo è quello in cui si vogliono sequenzializzare una serie di operazioni. Per garantire che una certa azione verrà completata, sarà sufficiente indicare come condizione della transizione l'evento che corrisponde al termine dell'azione. Si consideri il passo `SpetA` a cui è associata l'azione `Apri la valvola` nell'esempio raffigurato in Fig. 1.8. Come si può notare, la condizione relativa alla transizione, `Sensore di valvola completamente aperta`, dipende dall'effettiva esecuzione dell'azione stessa.

Azioni continue e transizioni

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

▽ *Esempio: Sincronizzazione di una azione*

> 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.

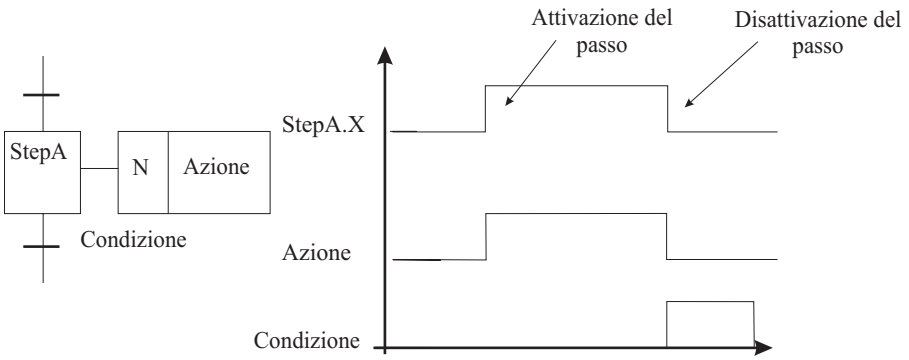


Figura 1.7: Azione continua

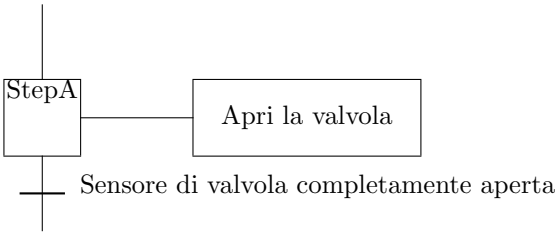


Figura 1.8: Utilizzando questa struttura, lo stato si evolverà solamente al termine dell'esecuzione dell'azione

Una possibile soluzione di controllo è rappresentata dallo schema SFC rappresentato in fig. 1.9

Quando la busta arriva alla macchina si ha la transizione tra il passo 10 ed il passo 11, e quindi l'azione di inchiostraggio termina anche se questa non era perfettamente completata, e quindi la qualità della lavorazione potrebbe non essere ottimale (schema di sinistra).

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 della azione di inchiostraggio). Per eliminare questo inconveniente sarà sufficiente introdurre una condizione di *termine della fase di inchiostraggio* (schema di destra).
△

Azioni impulsive

Le azioni impulsive (Fig. 1.10) sono attive per un solo ciclo di riesecuzione dell'algoritmo di controllo (corrispondente ad un ciclo di PLC), in corrispondenza della attivazione del passo (qualificatore `P1`), oppure alla sua disattivazione (qualificatore `P2`).

Azioni prolungate (Set - Reset)

In base alla regola generale, per specificare una azione perdurante su più passi consecutivi, occorrerebbe ripetere l'azione in corrispondena di tutti i passi specificati. In modo più sintetico è però possibile, in questi casi, utilizzare il qualificatore *set* azione, che rende permanente l'attività della azione fino alla esecuzione di un *reset* azione (Fig. 1.11).

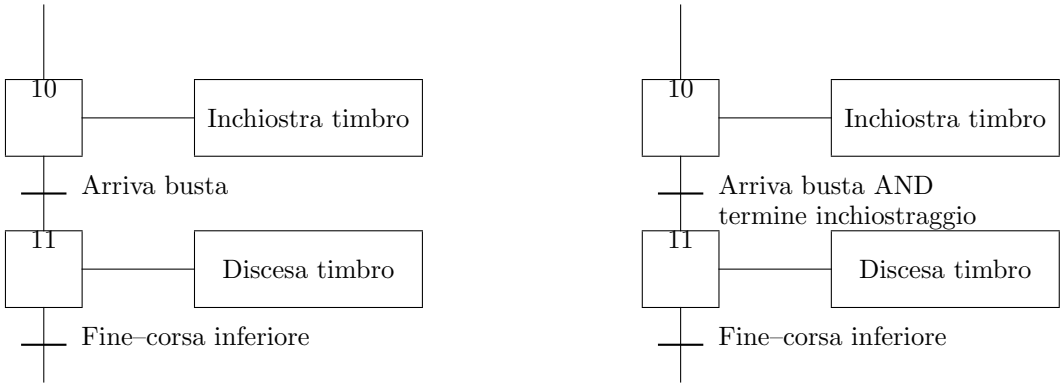


Figura 1.9: Sincronizzazione di una azione

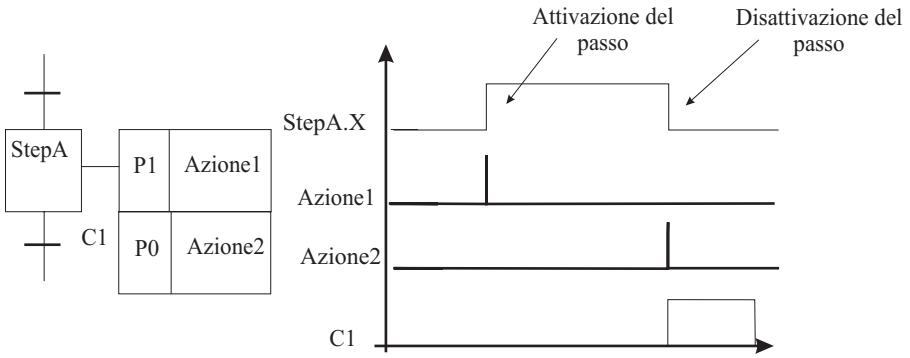


Figura 1.10: Azione impulsiva

Azione "time limited"

L'azione "time limited" permette di specificare la durata della azione tramite una variabile temporale. Se però il periodo di attivazione del passo termina prima del termine specificato dalla variabile temporale, l'aziona cessa ugualmente di essere valida anche se il periodo di tempo impostato non è ancora scaduto (si veda la Fig. 1.12).

Azione "time delayed"

L'azione "time delayed" consente di definire l'attivazione di una azione con un ritardo fissato a partire dalla attivazione del passo corrispondente. Se il passo viene disattivto prima che il ritardo sia terminato, l'azione non viene eseguita (Fig. 1.13).

Azione "stored and time delayed"

L'azione "stored and time delayed" comincia dopo un ritardo impostato tramite una variabile temporale, anche nel caso in cui il passo diventi inattivo, e continua finchè non è terminata da un comando di reset (si veda 1.14).

Azione "time delayed and stored"

L'azione "time delayed and Stored" viene eseguita una volta che un periodo impostato su di un temporizzatore scade. L'azione continua fino a che non viene terminata da una azione di reset. Nel caso però il passo venga disattivato *prima* che il periodo temporizzato termini, l'azione non viene più eseguita (si veda la figura 1.15).

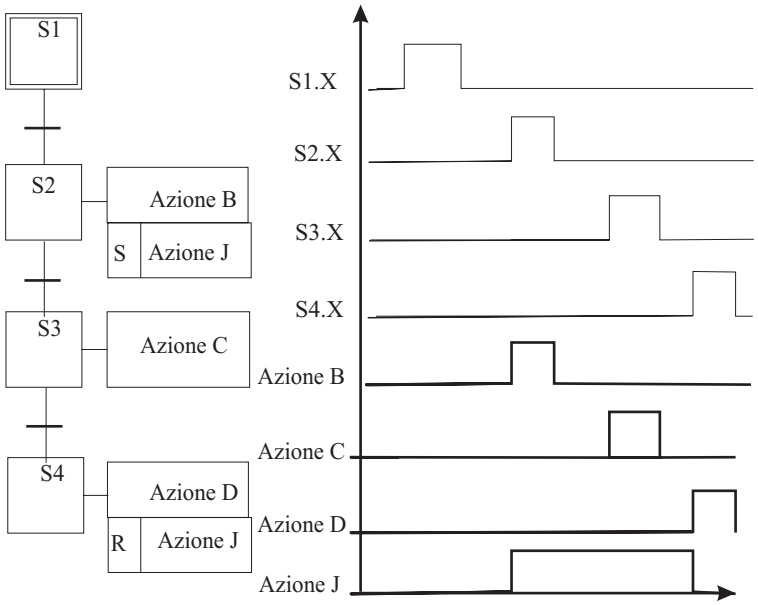


Figura 1.11: Azione prolungata

Azione "stored and time limited"

L'azione "stored and time limited" comincia quando il passo diventa attivo e continua finchè non viene terminata da una azione di reset o non trascorre un certo intervallo di tempo.

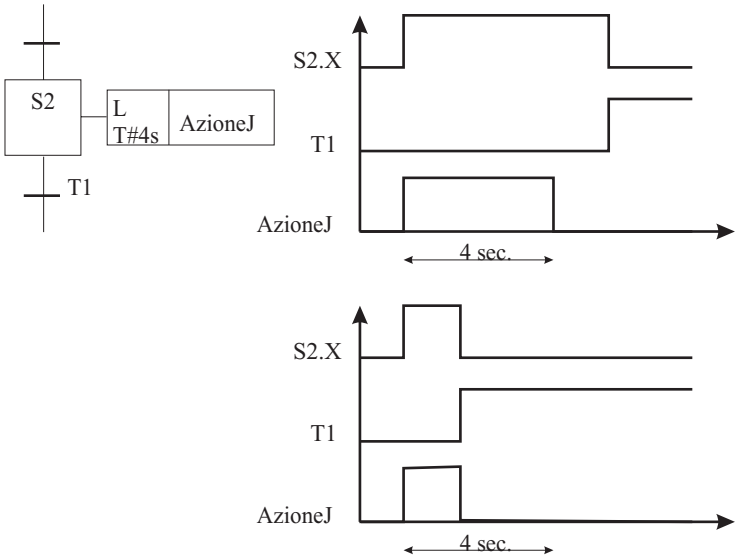


Figura 1.12: Azione “time limited”

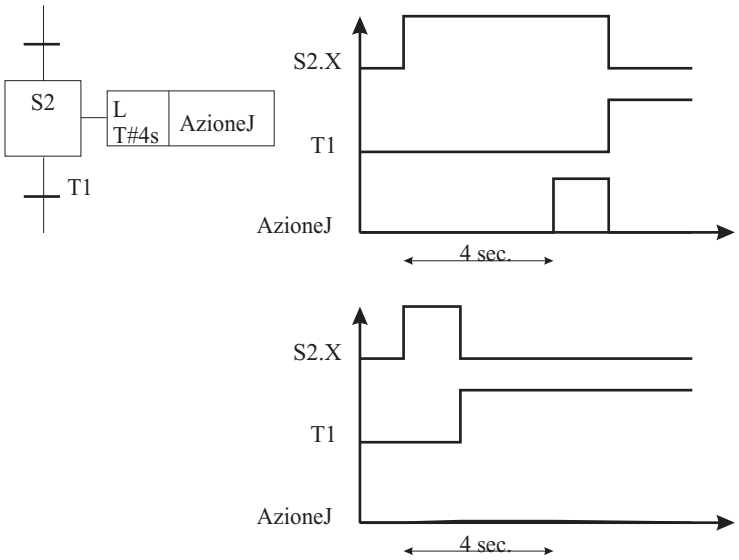


Figura 1.13: Azione “time delayed”

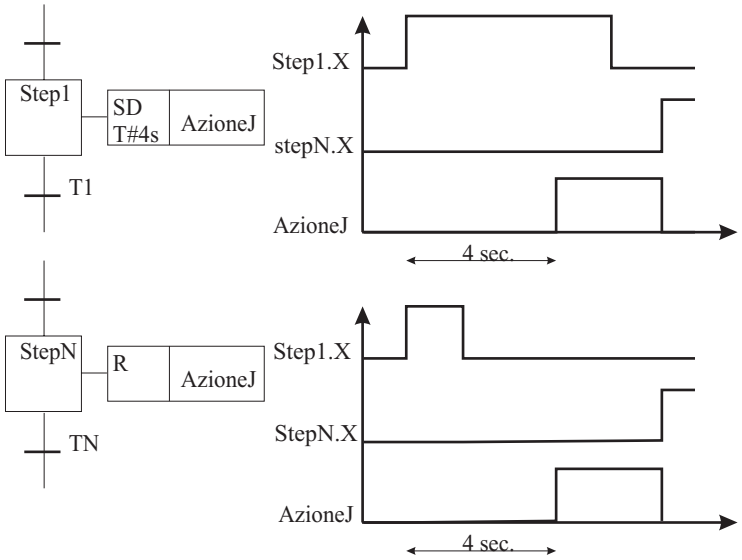


Figura 1.14: Azione “stored and time delayed”

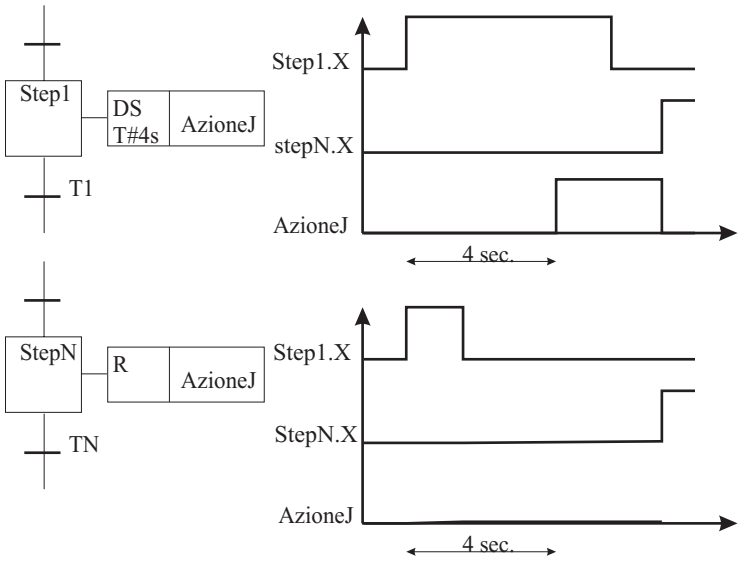


Figura 1.15: Azione “time delayed and stored”

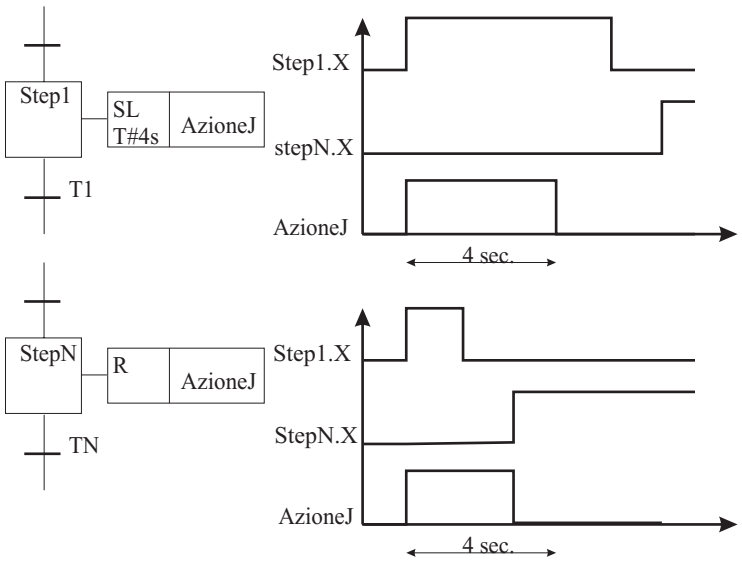


Figura 1.16: Azione “stored and time limited”

1.4 Regole di evoluzione dello SFC

1. **Inizializzazione**

   All'avviamento alcuni passi sono attivi senza alcuna condizione preliminare (*Passi iniziali*). In ogni schema SFC vi deve esserci sempre (almeno) un passo iniziale.

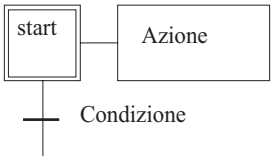


Figura 1.17: Passo iniziale di uno SFC

2. **Transizioni**

   Una transizione è *convalidata* quando il passo (i passi) precedente è attivo. Quando la transizione è *convalidata*, e contemporaneamente la condizioni associata è *verificata*, allora la transizione è *abilitata* e quindi avviene l'*evoluzione* verso il passo successivo.

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.

4. **Simultaneità**

   Nel caso vi siano transizioni simultaneamente abilitate, esse dovranno essere superate (idealmente) allo stesso tempo.

   Per meglio comprendere l'importanza di questa regola, si faccia riferimento alla Fig. 1.18. In questo esempio vi sono due passi attivi (`step10` e `step20`) aventi in cascata due transizioni abilitate. Se, ad esempio, la regola non fosse valida ed avvenisse prima la transizione relativa alla condizione `step20.X`, il passo `step10` non sarebbe più attivo e quindi la transizione in cascata al passo `step20` ne risulterebbe disabilitata, in quando la condizione non sarebbe più verificata. In questo modo si avrebbe una sorta di "corsa critica" tra le due transizioni, che ovviamente, si tradurrebbe in una potenziale errore logico.

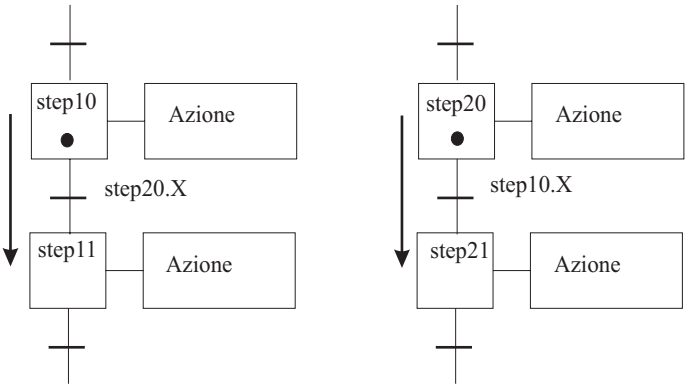


Figura 1.18: Transizioni simultanee.

5. **priorità dell'attivazione**

   Se uno stesso passo deve essere attivato e disattivato simultaneamente, allora esso rimane attivato.

   Un esempio di questo caso è rappresentato in Fig. 1.19, in cui due passi successivi sono attivi. Durante l'esecuzione della transizione, il passo `11` dovrebbe essere contemporaneamente attivato e disattivato. In base alla regola esso rimane (correttamente) attivato.

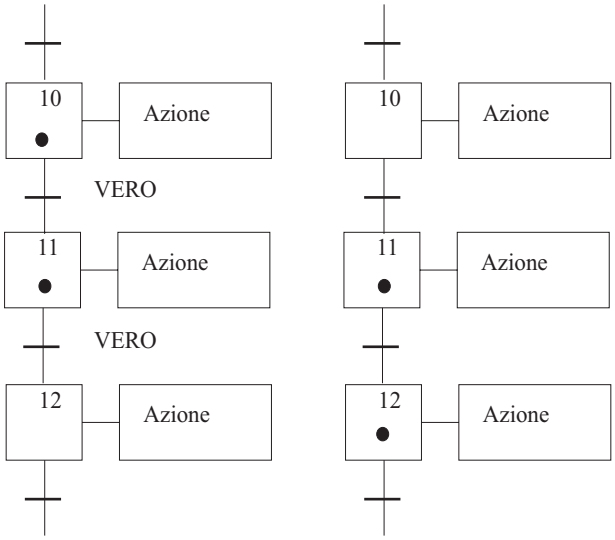


Figura 1.19: Attivazione e disattivazione simultanea.

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.

1.5 Sintassi dei collegamenti

I collegamenti definiscono la topologia dello schema SFC, definendo come evolve lo stato di attivazione dei singoli passi, e quindi, in definitiva, fissando la logica di controllo.

Le principali sequenze logiche presenti in una macchina automatica sono l'esecuzione condizionata di determinate sequenze di programma e l'evoluzione in parallelo di alcune azioni di controllo. Il formalismo SFC consente di definire in modo rigoroso la topologia corrispondente a questi tipi di funzionamento.

Divergenza opzionale

La divergenza opzionale esprime un criterio di scelta. Riferendoci alla figura 1.20, 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. finercorsa destro-sinistro), è possibile che vi sia una attivazione contemporanea (involontaria) dei passi ed una conseguente attivazione (errata) di percorsi simultanei.

Per evitare questo inconveniente, lo SFC originario può essere modificato come mostrato in figura 1.21 in modo che le condizioni sono riscritte come:

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

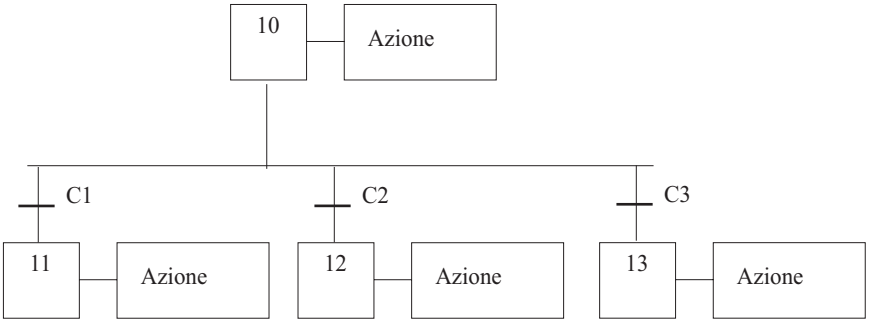


Figura 1.20: Divergenza opzionale

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.

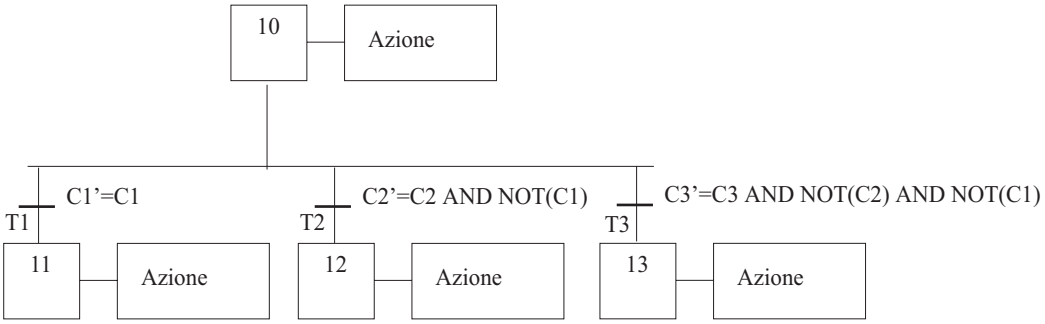


Figura 1.21: Divergenza opzionale con scelte mutuamente esclusive

Divergenza simultanea (Parallelismo)

La divergenza simultanea consente di attivare più sequenze di controllo che evolvono in parallelo. In questo caso è sufficiente specificare una sola transizione (ed una sola condizione), superata la quale, le sequenze evolveranno liberamente in parallelo (fig 1.22).

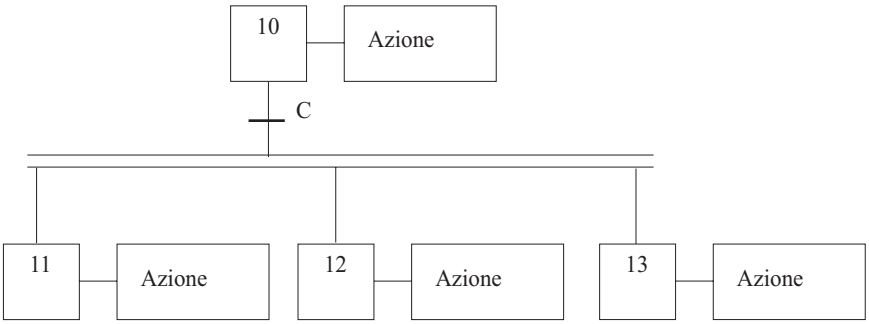


Figura 1.22: Divergenza simultanea

Convergenza opzionale

Il costrutto della *convergenza opzionale* serve a concludere delle sequenze eseguite in modo mutuamente esclusivo. Questa struttura sintattica è chiaramente il complemento della *divergenza opzionale*. Un esempio di divergenza opzionale è mostrata in figura 1.23.

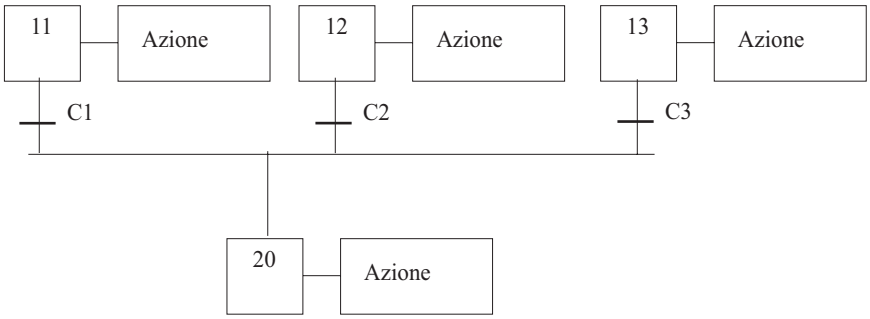


Figura 1.23: Convergenza opzionale

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

Convergenza simultanea (Sincronizzazione)

Il costrutto sintattico della *convergenza simultanea* viene utilizzato per concludere l'esecuzione di sequenze eseguite in parallelo, e quindi risulta immediato che tale costrutto è il complementare della *divergenza simultanea*. Un esempio di convergenza simultanea è mostrata in fig. 1.24.

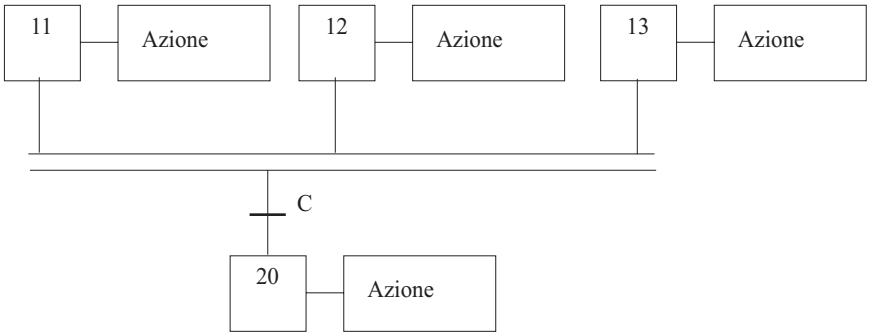


Figura 1.24: Convergenza simultanea

Errori da evitare

Le fig. 1.25 e 1.26 mostrano possibili errori nell'uso delle sequenze simultanee. Nel primo caso una divergenza opzionale viene terminata da una convergenza simultanea. Chiaramente questo costrutto conduce ad una situazione di blocco, in quanto solamente una delle sequenze originate dalla divergenza opzionale sarà attiva, e quindi solo un passo a monte della convergenza simultanea sarà, ad un certo istante, attivo, con la conseguente impossibilità alla abilitazione della transizione corrispondente alla convergenza multipla.

Il secondo caso presenta un errore più subdolo, in cui la divergenza simultanea attiva più sequenze simultaneamente. Al termine di queste sequenze troviamo una convergenza opzionale, in cui ci si aspetta una sola sequenza a monte attiva. Quindi ad ogni "arrivo" del token (passo attivo) da ciascuna sequenza, si attiverà il passo successivo alla convergenza opzionale. Questo produrra una

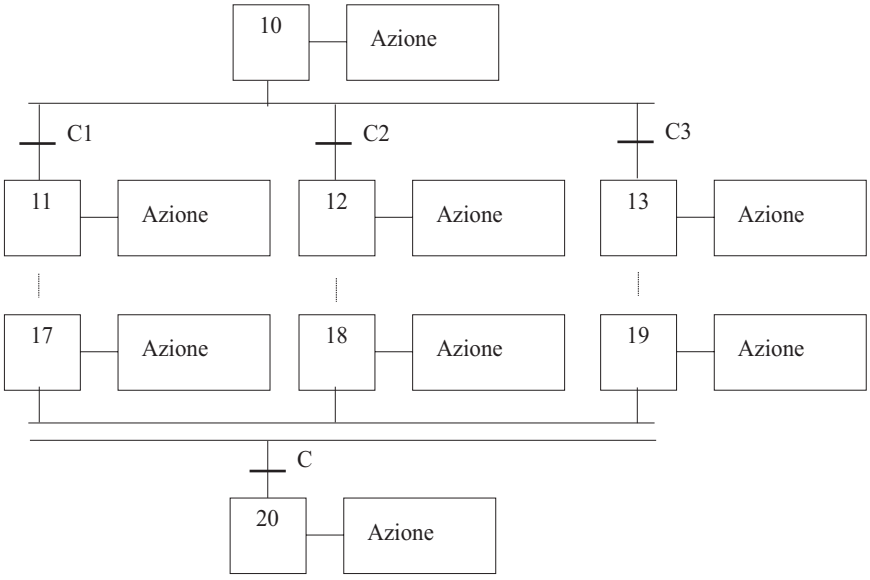


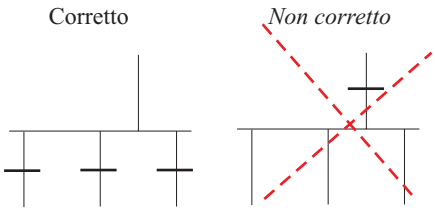
Figura 1.25: Un errore: una divergenza opzionale "chiusa" da una convergenza simultanea

attivazione multipla ed incontrollata dei passi a valle della convergenza opzionale, con potenziali effetti distruttivi sul sistema controllato.

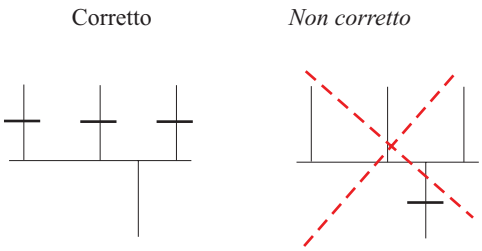
Collegamenti multipli

Passiamo ad enunciare alcune regole relative alla corretta stesura dei collegamenti multipli.

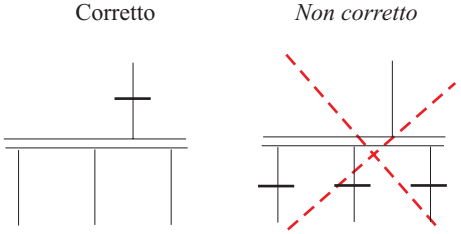
- Le **divergenze opzionali** non possono mai essere precedute da una sola transizione.



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



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



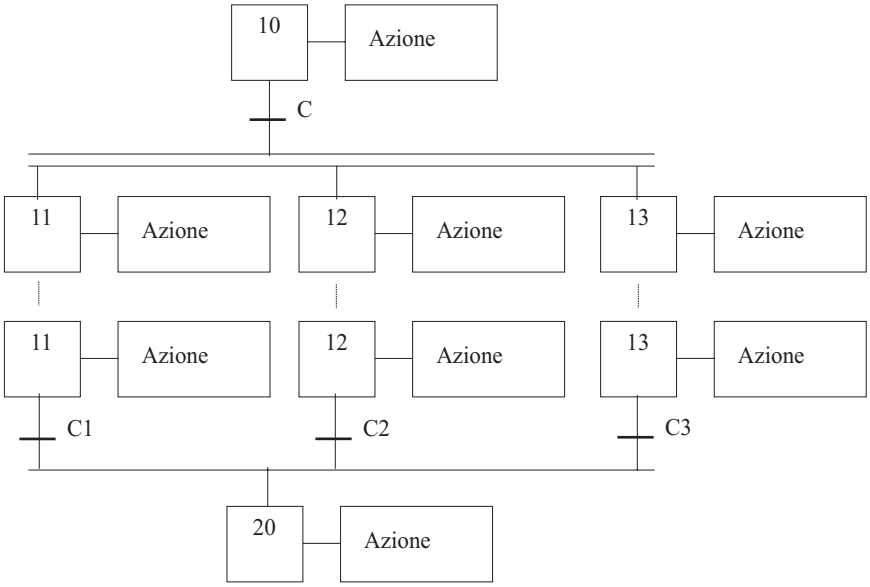
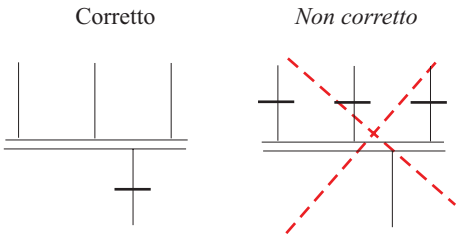


Figura 1.26: Un errore: una divergenza simultanea "chiusa" da una convergenza opzionale

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



Definzione alternativa delle azioni prolungate

Una difficoltà frequente 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 (si veda fig 1.27):

- *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 di 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").

Mutua esclusione

Una *condizione comune* può essere condivisa da diverse sequenze utilizzatrici in modo esclusivo. Questo costrutto serve ad implementare la protezione di *sezioni critiche* di un programma (tipicamente risorse condivise che debbono essere occupate in modo esclusivo) attraverso il meccanismo del *semaforo.*

Con riferimento alla fig. 1.28, la struttura si basa sull'attivazione del passo $X10$ che valida tre transizioni. In base alla attivazione dei passi di ingresso ($X11, X21$ e $X31$) ed alla convalida delle tre condizioni mutuamente esclusive, solo una delle sequenze $S1$, $S2$ o $S3$ verrà eseguita.

Occorre notare, inoltre, che la struttura prevede un meccanismo di gestione delle priorità fisso ($S1$ ha maggiore priorità, mentre $S3$ ha minore priorità).

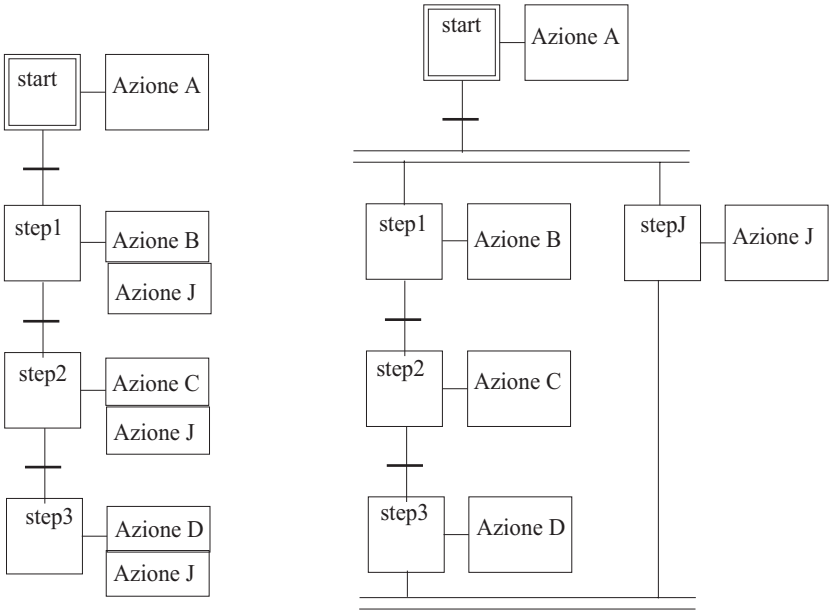


Figura 1.27: Implementazione di azioni continue

Per il corretto funzionamento della struttura semaforica, occorre che il passo "condiviso" $X10$ sia compreso nell'insieme dei passi iniziali.

Sincronizzazione

La sincronizzazione tra due sequenze può essere ottenuta in modo molto semplice utilizzando un passo di *sincronizzazione*. Con riferimento alla figura 1.29, la sequenza di destra è obbligata ad attendere che la sequenza di sinistra abbia raggiunto il passo $X12$ (a cui corrisponde il passo di sincronizzazione $X10$) prima di poter proseguire.

Errori relativi alla topologia dello schema SFC

In base a quanto detto nelle precedenti sezioni, occorre fare molta attenzione relativamente alla stesura *topologica* degli schemi SFC, vale a dire del disegno dei collegamenti tra i vari passi dello schema.

In figura 1.30 è mostrato un esempio di schema SFC errato. Infatti il collegamento che esce dalle sequenza simultanee a valle del passo `stepC3` può produrre una sequenza che non viene "chiusa" dalla convergenza simultanea, portando quindi ad una potenziale attivazione spuria del passo `stepZ`.

La figura 1.31 mostra un esempio errato (a sinistra) e come sia possibile correggere il problema (a destra). Lo schema di sinistra risulta errato in quanto nella sequenza simultanea di sinistra, vi è una divergenza opzionale che non è "chiusa". Questo porta al blocco dell'evoluzione della macchina in quanto un solo passo di una delle due sequenze opzionali sarà attivo e quindi la convergenza simultanea non potrà mai essere abilitata.

Diagnosi degli errori

La diagnosi degli errori in uno schema SFC può essere ricondotto alla analisi di uno schema ridotto mediante l'utilizzo reiterato di semplici regole di riduzione dello schema. Tali regole si possono riassumere come segue:

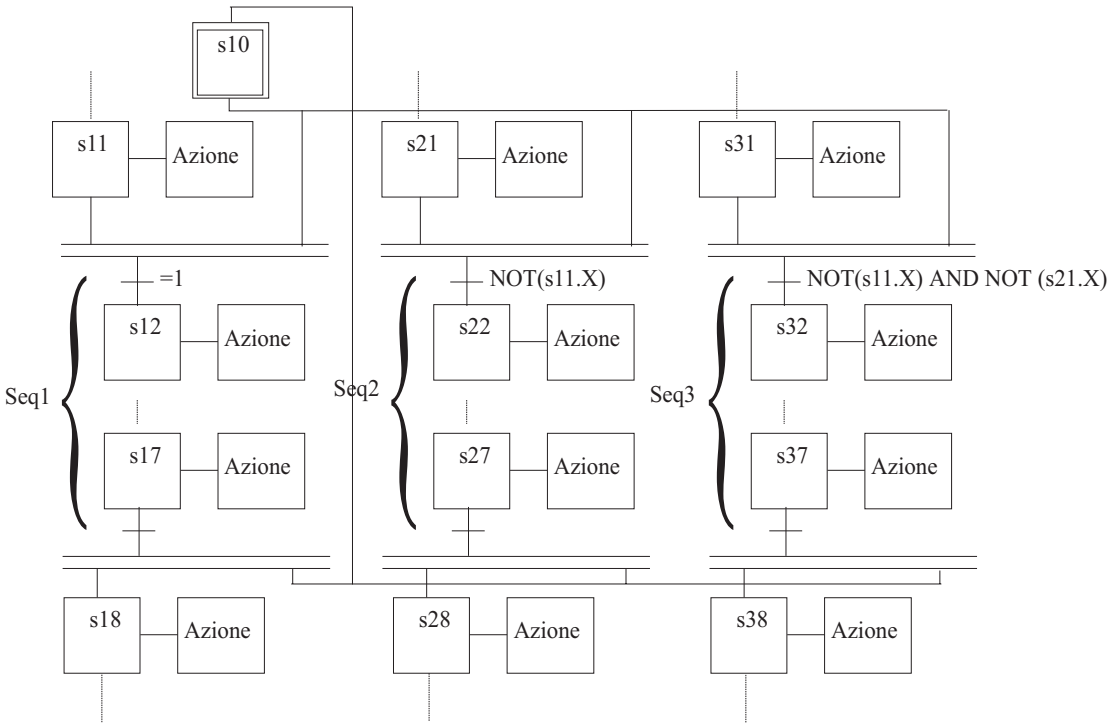
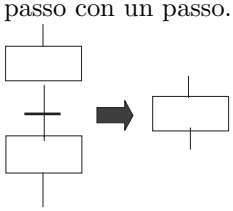
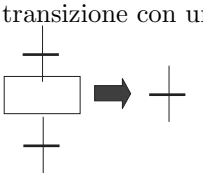


Figura 1.28: Implementazione di sequenze da eseguire in mutua esclusione

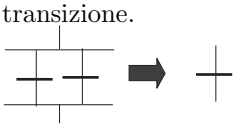
Sostituire una sequenza passo-transizione-passo con un passo.



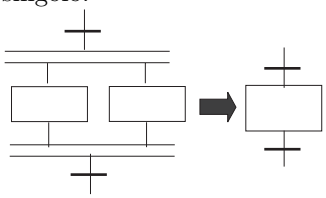
Sostituire una sequenza transizione-passo-transizione con una transizione.



Sostituire divergenze esclusive con una singola transizione.



Sostituire divergenze simultanee in un passo singolo.



Il risultato di tali semplificazioni deve essere **un singolo passo**.

Progetto top-down

Il progetto di un sistema complesso come il controllore di una macchina automatica deve essere affrontato per livelli gerarchici. In generale sarà presente un livello gerarchico "alto", in cui viene specificato solamente la sequenza logica delle operazioni eseguite dalla macchina ("cosa deve fare l'automatismo"), ed un livello logico "basso", che specifica in maggiore ddettaglio le operazioni effettivamente svolte dalla macchina per ottenere il risultato della produzione ("come deve fare l'automatismo"). Chiaramente questo schema concettuale può venire espanso su diversi livelli fino ad un punto di dettaglio desiderato.

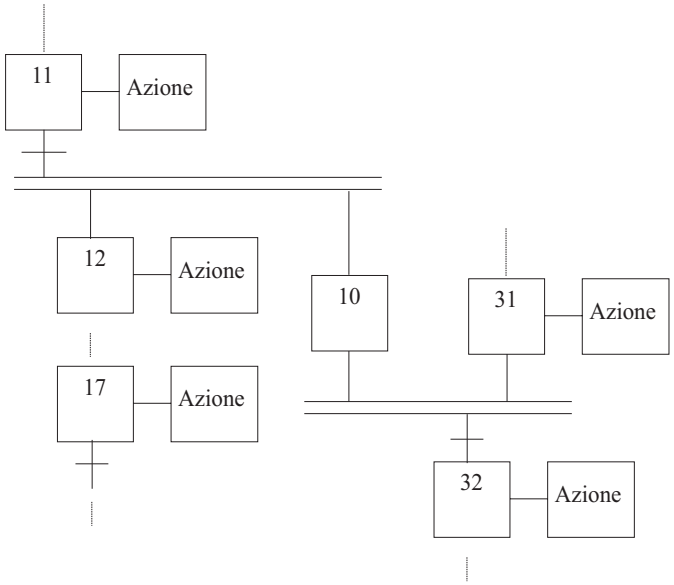


Figura 1.29: Implementazione di una sincronizzazione tra due sequenze

Questo modo di scomporre il problema di controllo consente una rapida "autodocumentazione" dell'applicativo il livello gerarchico più elevato descrive succintamente il funzionamento della macchina, in modo che il lettore possa capirne il funzionamento senza perdersi nei dettagli costruttivi, trattati ad un livello più basso.

Questo modo di procedere è ben rappresentato nello schematismo SFC. In particolare, come rappresentato in figura 1.32, una azione associata ad un passo di uno schema SFC può essere implementata attraverso uno schema SFC "figlio" di grado gerarchico inferiore.

Variabili temporali

Ad ogni passo di uno schema SFC è implicitamente associato un temporizzatore. Il riferimento al temporizzatore associato al passo `X` si ottiene attraverso il costrutto sintattico `X.T`. In figura 1.33 è mostrata una possibile applicazione di una variabile temporale.

Un classico esempio di utilizzo di una variabile temporizzata è quello in relazione alla diagnosi di malfunzionamenti che provocano il blocco della normale esecuzione del programma. In pratica, ogni volta che si suppone la possibilità di un qualche tipo di blocco dell'automatismo in un qualche stato di funzionamento, è possibile utilizzare una sequenza di *watchdog* utilizzando una variabile temporale.

Facendo riferimento alla figura 1.34, si suppone che l'azione corrispondente allo stato `start` sia suscettibile di un blocco imprevisto, per cui è possibile che il segnale di ingresso `Materiale_OK` non venga mai attivato a causa di questo blocco. In tal caso è possibile inserire una divergenza opzionale che evolverà verso una condizione di allarme in caso di blocco. Occorre rimarcare che l'allarme dovrà portare contestiualmente alla identificazione del guasto oltre che al suo segnalamento. Se il sistema, oltre a fornire l'informaizone di allarme, provvede a segnalare anche lo *stato* in cui è avvenuto, risulterà molto agevole l'identificazione del guasto ed il recupero della situazione anomala.

In figura 1.35 è mostrato un esempio di come sia possibile prevedere la segnalazione di un guasto che avvenga in un qualunque punto di una sequenza predeterminata. In sostanza, il sistema segnala se vi sono sono delle condizioni di blocco nella sequenza se la sequenza stessa non vine terminata entro un determinato tempo.

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,

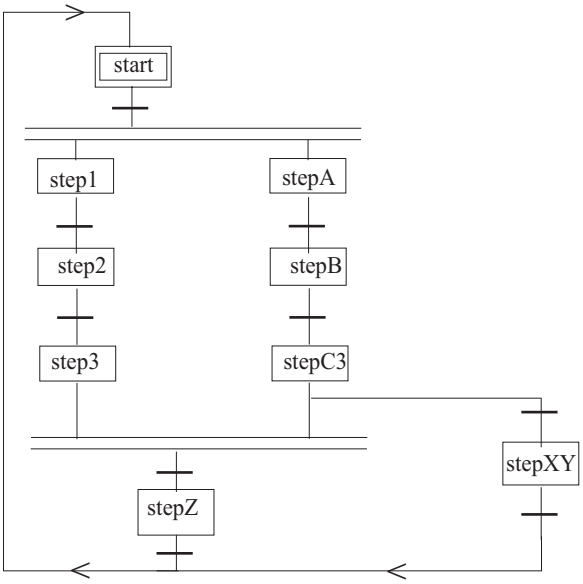


Figura 1.30: Schema SFC potenzialmente errato

mentre negli schemi proposti, il *watchdog* segnala il mancato rispetto dei tempi di esecuzione di una determinata azione svolta dalla macchina.

1.5.1 Esempio riassuntivo: Isola di foratura

Vediamo ora un esempio riassuntivo relativamente ai concetti su cui si basa il progetto di un automatismo metiante il formalismo Sequential Funcional Chart.

In relazione alla figura 1.36, consideriamo una macchina automatica predisposta alla foratura di pezzo costituita da tre isole di lavorazione: nella prima il pezzo viene caricato, nella seconda il pezzo viene forato e la terza si occupa di controllare il foro e scaricare il pezzo.

Un motore elettrico (attuatore) aziona la rotazione del piatto dell'isola di lavorazione in modo da trasferire i pezzi da una fase di lavorazione alla successiva. Il controllo della foratura è eseguito da un tastatore che scende nel foro precedentemente eseguito. Se il foro non è stato eseguito correttamente il tastatore non riesce a scendere completamente e quindi l'automatismo segnala un allarme. Il pezzo a questo punto verrà espulso manualmente e il ciclo potrà iniziare nuovamente.

Lo schema SFC dell'automatismo è strutturato su due livelli gerarchici. Il primo livello mostrato in figura 1.37, descrive le operazioni che debbono essere fatte dalla macchina.

Il livello gerarchico "basso", in cui viene descritti i dettagli implementativi relativi a come vengono effettivamente implementate le operazioni che deve eseguire la macchina automatica, è dettagliato nella figura 1.38, per quanto riguarda la parte relativa all'inserimento del pezzo nella macchina, in figura 1.39, in riguardo alla parte di foratura del pezzo, e in figura 1.40 per quanto riguarda la verifica ed espulsione del pezzo.

La struttura del programma è pensata modulare ed orientata agli oggetti. Le operazioni di **inserimento**, **foratura** ed **espulsione** sono da considerarsi funzioni eseguite su determinati oggetti fisici.

Questi dispositivi fisici venogno trattati come oggetti, quindi le funzioni di controllo debbono venire strutturate come metodi per la gestione di tali oggetti.

Per garantire le proprietà di indipendenza di un oggetto dall'applicativo che ne fa uso, occorre progettare con cura le interfaccie tra i metodi e l'applicativo cliente, avendo come vincolo l'utilizzo della sintassi dello SFC. Questo richiede di progettare con cura la sincronizzazione tra il metodo (la funzione servitore) ed il cliente.

Una possibile soluzione consiste nell'imporre che la funzione metodo venga richiamata **per tutto**

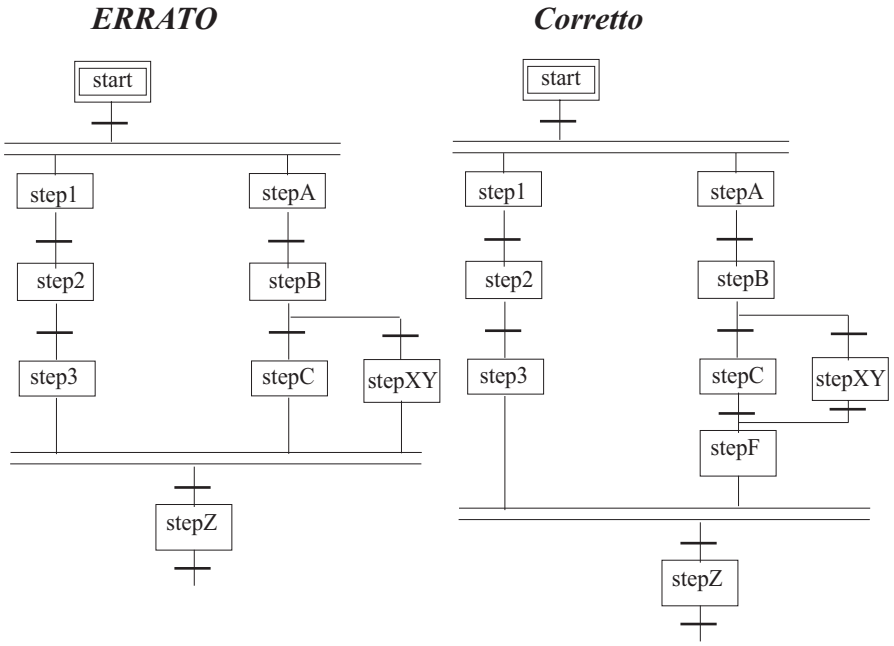


Figura 1.31: Schema SFC potenzialmente errato e sua versione corretta

il periodo di attivazione del passo del cliente attraverso una azione continua. Questo però richiede una accurata sincronizzazione tra cliente e servitore per evitare che il programma servitore venga immediatamante reinizializzato al termine del suo ciclo a causa del persistere della attivazione del passo cliente.

Per evitare questo problema, occorre che il cliente possa evolvere immediatamente quando il servitore segnala il termine del ciclo, in modo da disattivare il passo di richiesta di servizio.

In alternativa si può pensare di differenziare la fase di inzializzazione del servitore dedicandogli un passo a parte.

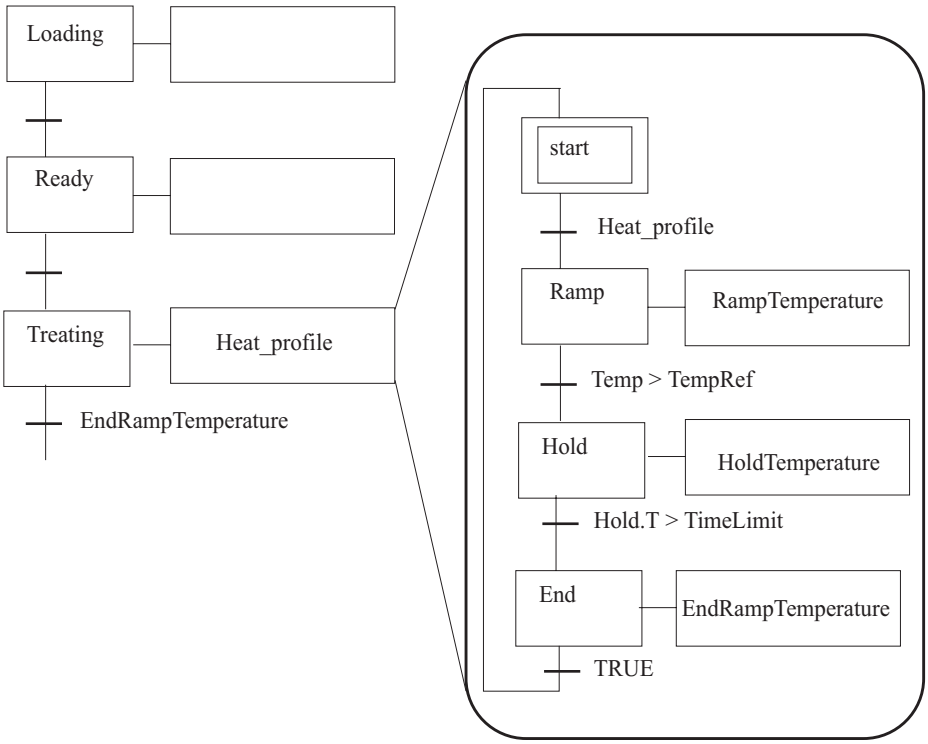


Figura 1.32: Decomposizione gerarchica "top–down" di uno schema SFC

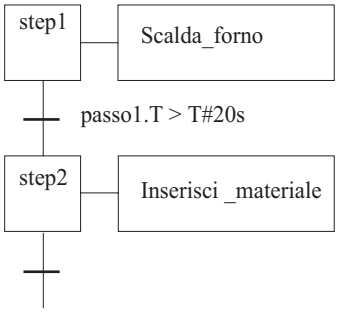


Figura 1.33: Esempio di utilizzo di una variabile temporale

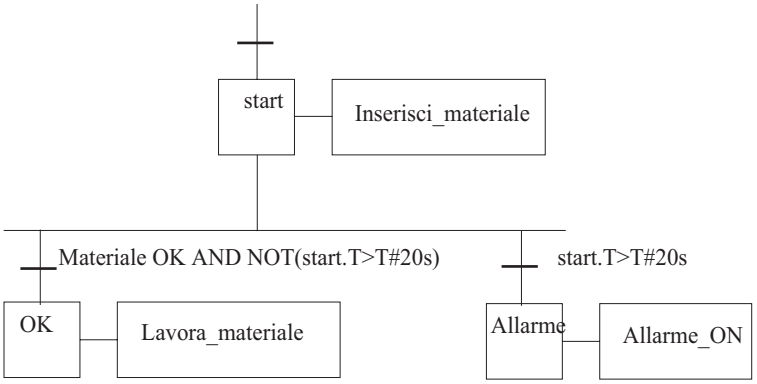


Figura 1.34: Esempio di utilizzo di una variabile temporale per la realizzazione di un *watchdog timer*

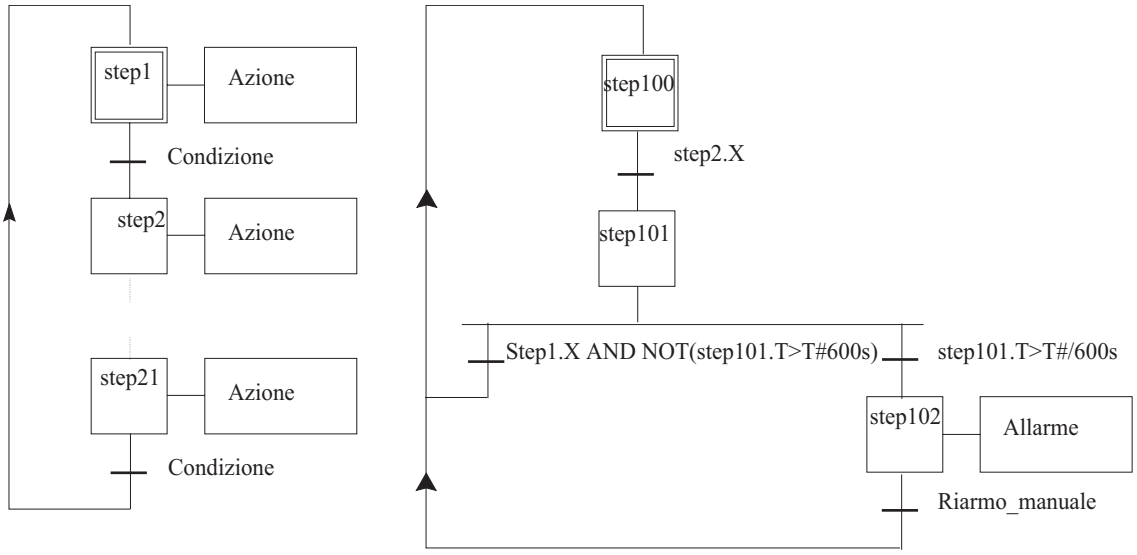


Figura 1.35: Esempio di utilizzo di una variabile temporale per la realizzazione di un *watchdog timer* relativo ad una seuenza operativa

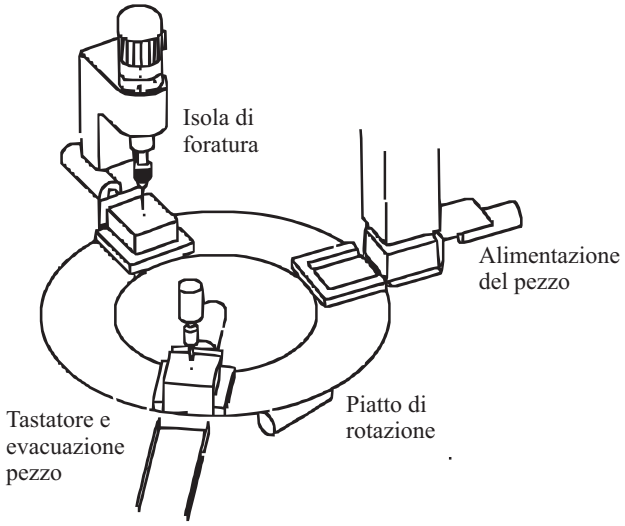


Figura 1.36: Isola di foratura

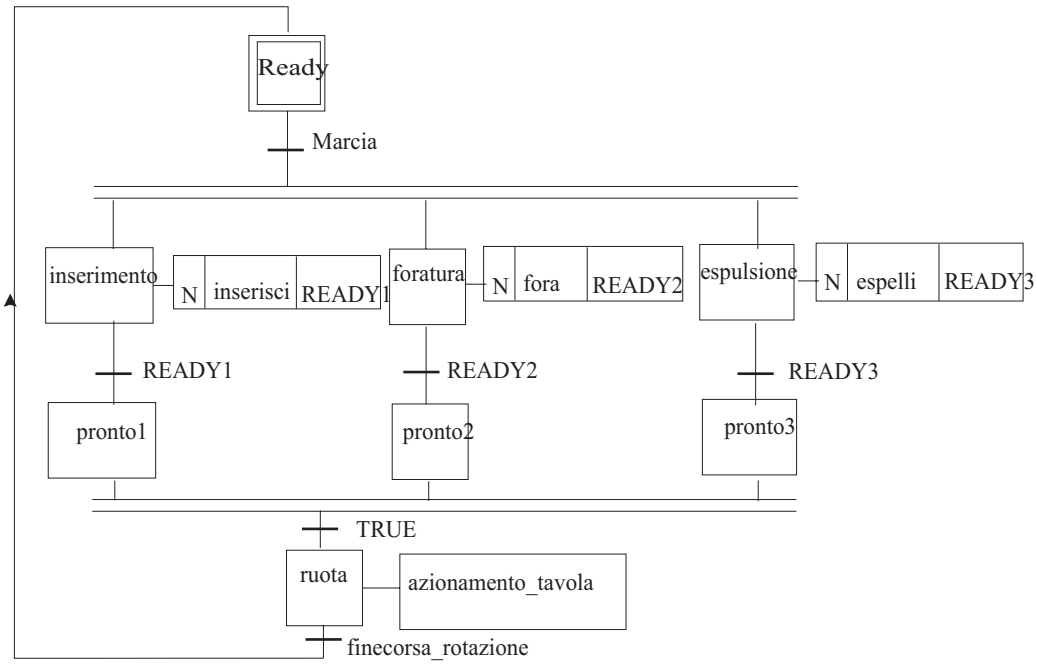


Figura 1.37: Livello gerarchico “alto” dello SFC

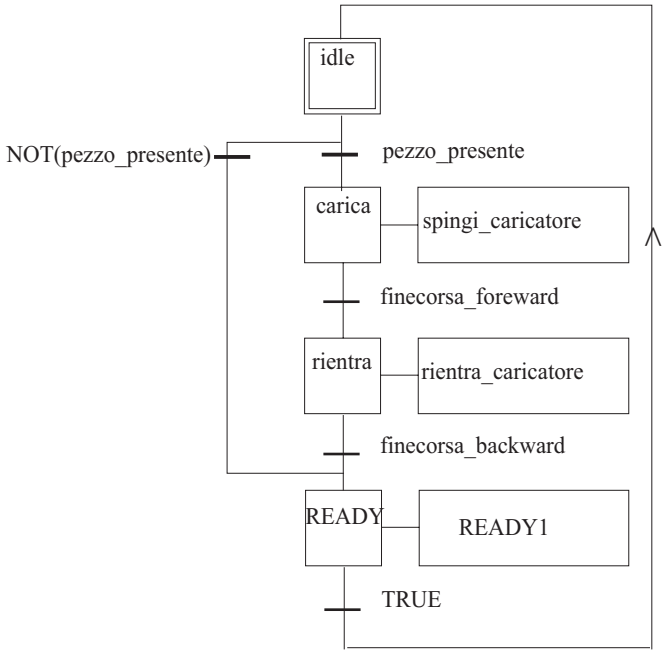


Figura 1.38: SFC relativo all’inserimento del pezzo nella macchina

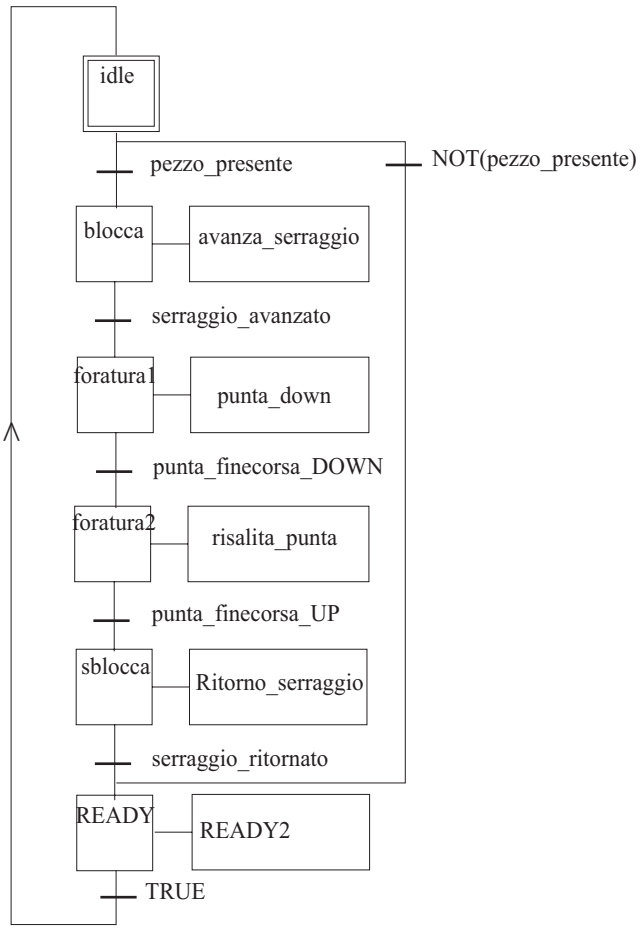


Figura 1.39: SFC della parte di foratura del pezzo

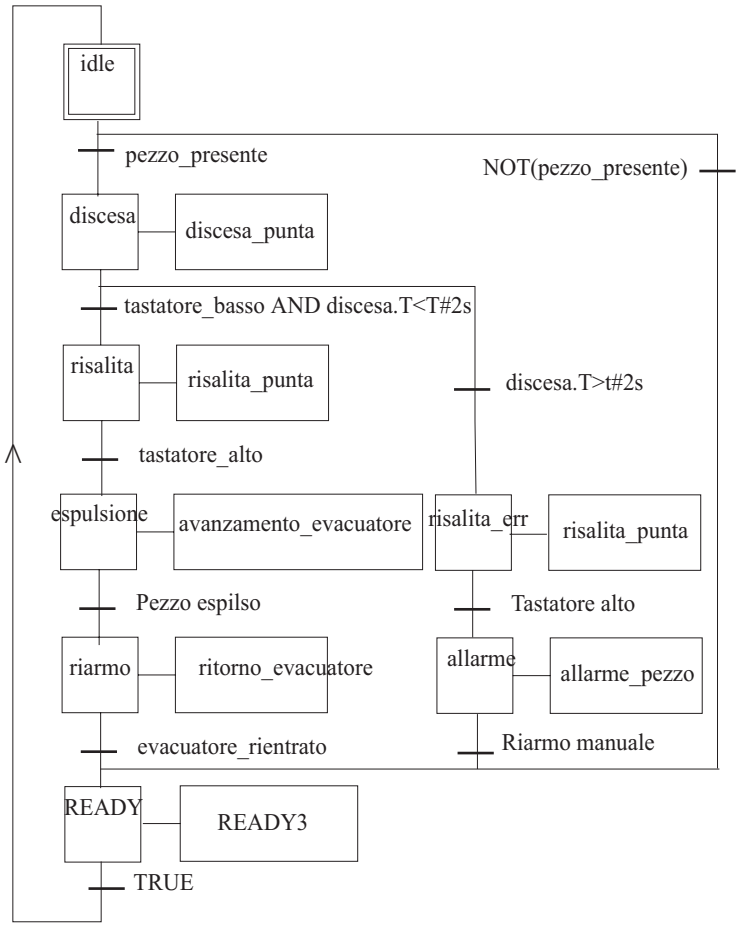


Figura 1.40: SFC relativo alla verifica ed espulsione del pezzo

Bibliografia

[1] “Software di sistema per S7-300/400, Sviluppo di programmi, Manuale di Programmazione”, Manuale Siemens, Cod. C79000-G7072-C506-01.

[2] IEC 61131-3 (1993-03) “Programmable controllers - Part 3: Programming languages”, International Electrotechnical Commission (1993).

Appendice

Appendice A

Le tabelle riassuntive della sintassi dello SFC secondo la norma IEC1131-3

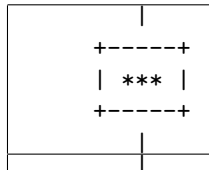
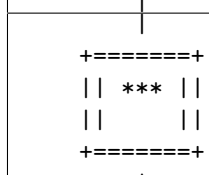
	Step - Graphical form "***" = step name
	Initial step - Graphical form. "***" = Name of initial step
***.X	Step flag - "***" = Step name **.X = Boolean 1 when *** is active, Boolean 0 otherwise
***.T	Step elapsed time - "***" = Step name **.T = A variable of type TIME

Tabella A.1: La sintassi dei passi secondo la norma IEC1131-3

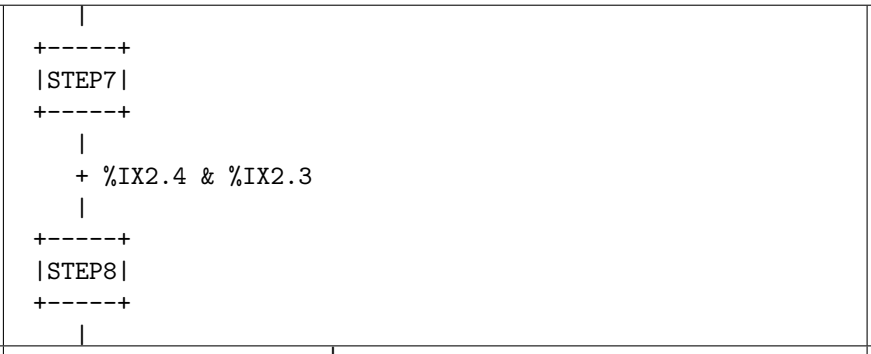
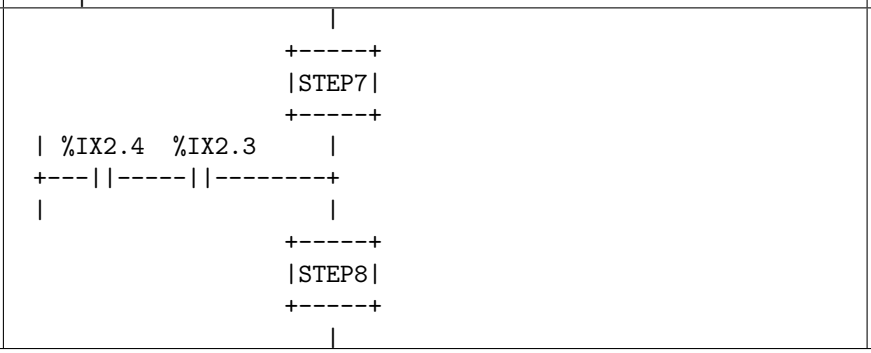
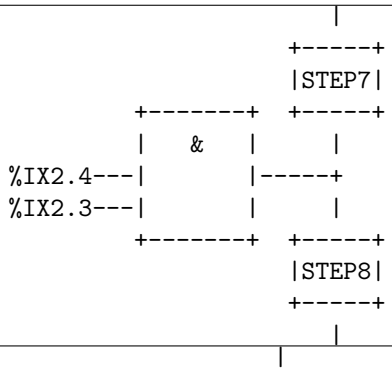
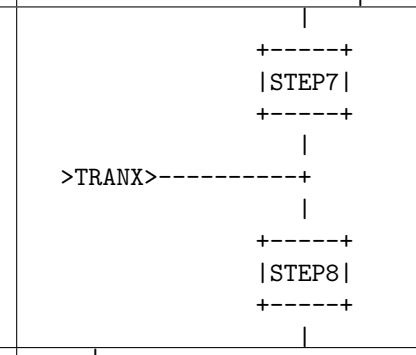
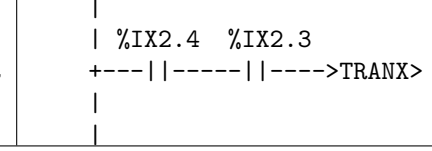
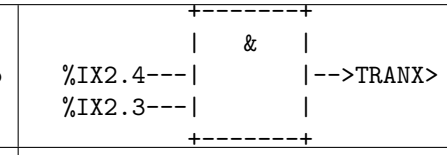
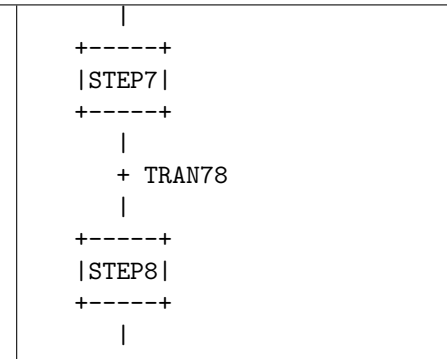
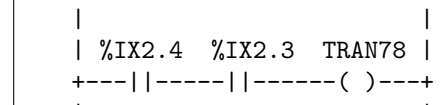
1		Predecessor step Transition condition using ST language Successor step
2		Predecessor step Transition condition using LD language Successor step

Tabella A.2: La sintassi delle transizioni secondo la norma IEC1131-3

3		Predecessor step Transition condition using FBD language Successor step
4		Use of connector: Predecessor step Transition connector Successor step
4a		Transition condition: Using LD language

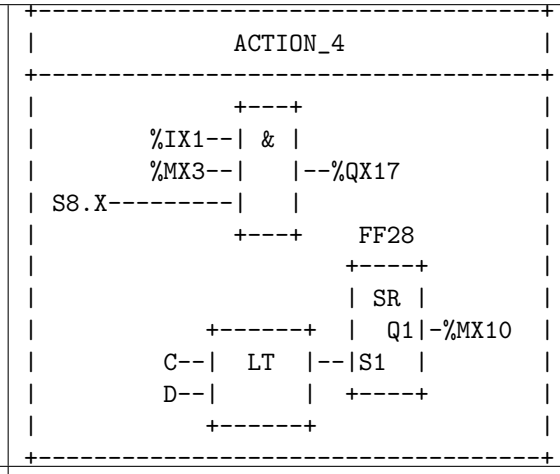
4b		Transition condition: Using FBD language
5	STEP STEP7: END_STEP TRANSITION FROM STEP7 TO STEP8 := %IX2.4 & %IX2.3 ; END_TRANSITION STEP STEP8: END_STEP	Textual equivalent of feature 1 using ST language
6	STEP STEP7: END_STEP TRANSITION FROM STEP7 TO STEP 8: LD %IX2.4 AND %IX2.3 END_TRANSITION STEP STEP8: END_STEP	Textual equivalent of feature 1 using IL language

7		Use of transition name: Predecessor step Transition name Successor step
7a	TRANSITION TRAN78:  END_TRANSITION	Transition condition using LD language

7b	TRANSITION TRAN78: +-------+ | & | %IX2.4---| |--TRAN78 %IX2.3---| | +-------+ END_TRANSITION	Transition condition using FBD language
7c	TRANSITION TRAN78: LD %IX2.4 AND %IX2.3 END_TRANSITION	Transition condition using IL language
7d	TRANSITION TRAN78 := %IX2.4 & %IX2.3 ; END_TRANSITION	Transition condition using ST language

1	Any Boolean variable declared in a VAR or VAR_OUTPUT block, or their graphical equivalents, can be an action.	
2a	+----------+ | ACTION_4 | +----------+ | | %IX1 %MX3 S8.X %QX17 | | | +---||-----||----||-----()---+ | | | | | | | +------+ | | | +----|EN ENO| %MX10 | | | | C--| LT |----------(S)---+ | | | D--| | | | | | +------+ | | +==========+	Graphical declaration in LD language
2b	+==========+ | OPEN_VALVE_1 | +----------+ | | ... | | +==========+ | | || VALVE_1_READY || | | +==========+ | | | | | + STEP8.X | | | | | +----------+ +---+----------+ | | | VALVE_1_OPENING |--| N |VALVE_1_FWD| | | +----------+ +---+----------+ | | | ... | +----------+	Inclusion of SFC elements in action

Tabella A.3: Sintassi delle azioni secondo la norma IEC1131-3

2c		Graphical declaration in LD language

2d	ACTION ACTION_4:
	  %QX17 := %IX1 & %MX3 & S8.X ;
	  FF28(S1 := (C<D));
	  %MX10 := FF28.Q;
	END_ACTION
		Textual declaration in ST language

2e	ACTION ACTION_4: LD S8.X AND %IX1 AND %MX3 ST %QX17 LD C LT D S1 FF28 LD FF28.Q ST %MX10 END_ACTION	Textual declaration in IL language