Lezione #10

Sequential Functional Chart

Sommario

- Introduzione e generalità
- Elementi dell'SFC: fasi, transizioni, archi
- Azioni e qualificatori
- Regole di evoluzione
- Vantaggi degli SFC
- Strutture classiche di programmazione
- Macroazioni
- Traduzione in ladder degli SFC

Sequential Functional Chart

- Il **SFC** è uno dei tre linguaggi grafici previsti dallo standard IEC 61131-3

- È utilizzato principalmente per la programmazione **logico/sequenziale**

Sequential Functional Chart

- **Programmare** un sistema di automazione coincide, in buona sostanza, con il **descriverne il comportamento desiderato**

- Tale descrizione va fatta in maniera **formale** e **non ambigua**

→ **SFC** è pensato per assolvere a questo scopo

Sequential Functional Chart

SFC consente di

- **descrivere senza ambiguità** il funzionamento logico/sequenziale desiderato del sistema
- **evidenziare specifiche** che non possono essere chiaramente descritte per via testuale

→ SFC *funzionale*

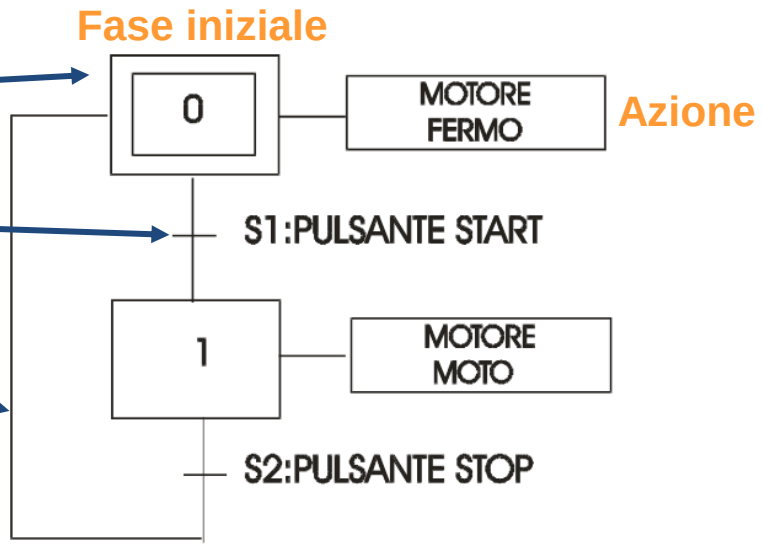
Sequential Functional Chart

- SFC è un efficiente **strumento di progettazione**
- Lo standard ne regolamenta la **sintassi** rendendolo di fatto anche un **linguaggio di programmazione**
- Il comportamento di un SFC dipende dal suo **stato precedente**

→ non posso realizzare funzioni in SFC, ma solo programmi e blocchi funzionali

Sequential Functional Chart

- I principali componenti di un SFC sono
  - Fasi
  - Transizioni
  - Archi orientati



- Un programma può essere composto da **più SFC non connessi tra loro**

Elementi di base
Fasi

- Ogni fase è rappresentata da un rettangolo al cui interno è riportato il **nome** della fase

- A ogni fase sono associate
  - Una **variabile segnalatrice** (o ***marker***) indicata da `<nome_fase>.X`
  - Una **variabile timer** indicata da `<nome_fase>.T`

Elementi di base
Fasi

- Marker di fase
  - 1 se la fase è attiva, 0 altrimenti
  - Viene inizializzato a 0 di default, tranne che per la fase iniziale
  - A volte, una fase attiva è indicata graficamente tramite un pallino
- Timer di fase
  - Inizializzato a T#0s
  - Se la fase è attiva, rappresenta il tempo da cui è attiva; altrimenti, rappresenta la durata dell'ultima attivazione
- Marker e timer di una fase **non possono essere modificate** dall'utente, ma solo utilizzate

Elementi di base
Archi orientati

- Le fasi sono collegate tra loro da **archi orientati**
- Gli archi procedono sempre **dal bordo inferiore** di una fase **a quello superiore** di un'altra
- Sono in genere interrotti da barrette orizzontali rappresentanti le **transizioni**

Elementi di base
Transizioni

- Le **transizioni** indicano le condizioni che devono essere soddisfatte per passare da una fase a un'altra

- La condizione deve assumere un **valore booleano** (VERO o FALSO), ed è spesso indicata in **ladder**

  – Tra due fasi deve esserci sempre una transizione
  – Tra due transizioni deve esserci sempre (almeno) una fase

Elementi di base
Azioni

- A ciascuna fase possono essere associate una o più **azioni**
- Un'azione può consistere in una **variabile booleana** (VERA quando l'azione è eseguita, secondo i qualificatori)
- Un'azione può anche consistere in una **sequenza di istruzioni**, espressa in uno qualsiasi dei linguaggi dello standard (anche SFC stesso → sviluppo **top-down**)

Elementi di base
Azioni

- Le azioni possono avere i seguenti **qualificatori**
  - **N**: continua (normal)
    l'azione è eseguita finché la fase è attiva
  - **L**: limitata (limited)
    l'azione è eseguita per il tempo specificato (o finchè non è disattivata la fase) e poi una volta ancora
  - **D**: ritardata (delayed)
    l'azione viene eseguita dopo un certo tempo
  - **S,R**: set/reset
    l'azione settata viene eseguita finché non viene resettata
  - **P**: impulsiva (pulse)
    l'azione è eseguita solo una volta

Elementi di base
Azioni

- Le azioni possono avere i seguenti **qualificatori**
  - **SD**: (stored/delayed)
    azione set eseguita dopo un certo intervallo di tempo
  - **DS**: (delayed/stored)
    viene eseguita come azione set se lo stato associato resta attivo per un tempo maggiore del ritardo specificato
  - **SL**: (stored/limited)
    azione set che viene terminata dopo un certo tempo oppure con un reset

Elementi di base
Azioni

- A ogni azione è associata anche una variabile booleana implicita `<nome_azione>.Q`
- Tale variabile è vera quando l'azione deve essere eseguita

Regole di evoluzione

- La **condizione** dell'SFC è l'insieme delle sue fasi attive
- Un SFC può cambiare condizione tramite il **superamento delle transizioni**
- Una transizione si dice
  - **abilitata** se tutte le fasi a monte sono attive
  - **superabile** se è abilitata e la condizione associata è vera

Regole di evoluzione

- Se una transizione è superabile, **viene superata**: tutte le fasi a monte sono disattivate e *poi* tutte le fasi a valle attivate

- Se transizioni distinte diventano superabili nello stesso momento, sono **superate contemporaneamente**

Regole di evoluzione

- **Fase instabile**: la transizione a valle è superabile quando la fase viene attivata

- Le azioni associate a quella fase vengono comunque eseguite prima della transizione: una fase **non può avere durata nulla!**

Vantaggi dell'SFC

L'utilizzo degli SFC presenta diversi **vantaggi**

- La programmazione equivale alla **descrizione del comportamento desiderato** del sistema
- Gli SFC possono essere utilizzati per definire le **specifiche funzionali** del sistema
  - Le azioni possono essere descritte in linguaggio naturale
  - È un linguaggio non ambiguo, che consente di individuare specifiche definite in maniera poco chiara

Vantaggi dell'SFC

L'utilizzo degli SFC presenta diversi **vantaggi**

- Consente una progettazione **top-down**
  - Posso spacchettare ogni fase/azioni in più fasi effettive
- È **autodocumentante** e rende il **debugging** particolarmente semplice
  - in genere basta capire perché una certa fase non si è attivata!
- Dà origine a programmi **efficienti**
  - Vanno valutate solo azioni e transizioni corrispondenti alle fasi attive

Strutture classiche di programmazione

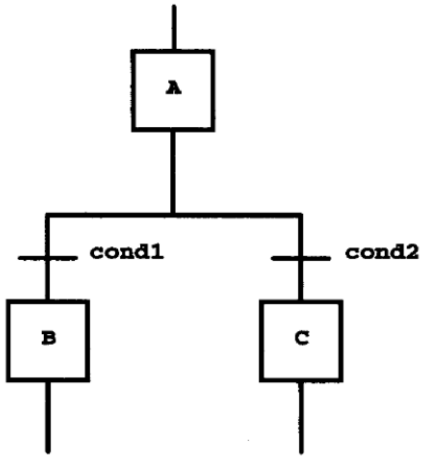
- Scelta e convergenza
- Parallelismo e sincronizzazione

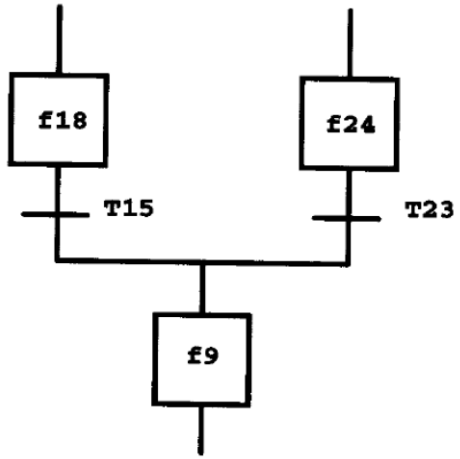
- Sincronizzazione locale
- Watchdog timer
- Semaforo

Strutture classiche di programmazione

Scelta e convergenza

- **Scelta e convergenza** possono essere utilizzate per realizzare costrutti del tipo `IF-THEN-ELSE-END_IF`
- Avrò un costrutto caratterizzato da **una fase** seguita/preceduta da **più transzioni**

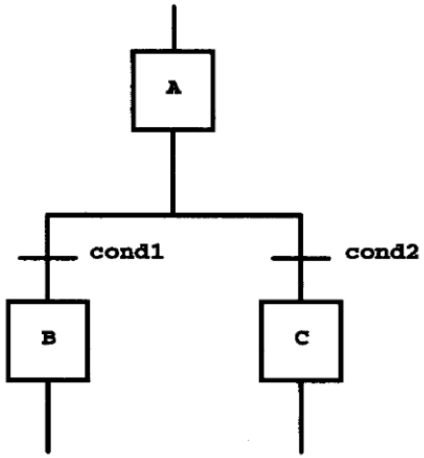


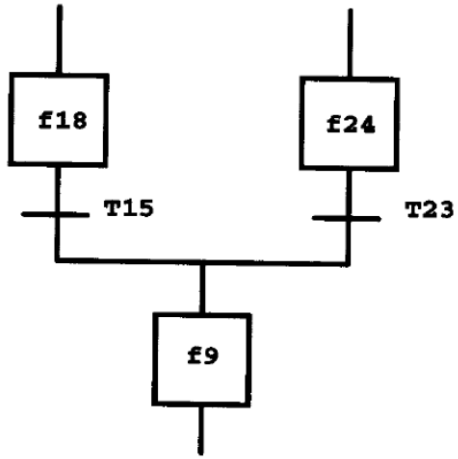


Strutture classiche di programmazione

Scelta e convergenza

- Le condizioni per la scelta devono garantire la **mutua esclusività**
  - **Naturale:** le condizioni non si verificano mai insieme
  - **Imposta**: è possibile numerare i rami in ordine di priorità o agire sulle condizioni (e.g. `cond2 AND NOT(cond1)`)

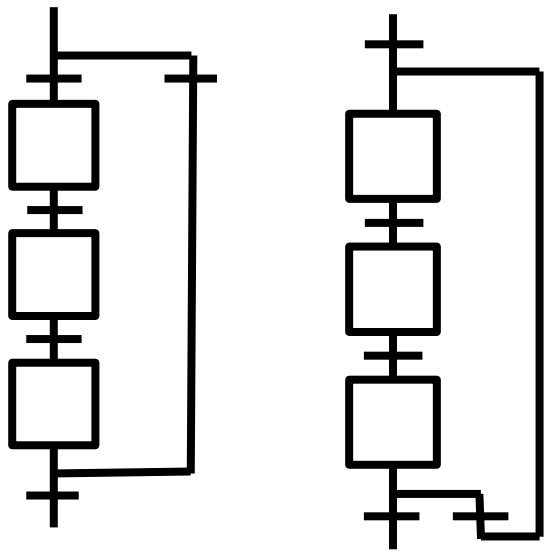




Strutture classiche di programmazione

Scelta e convergenza

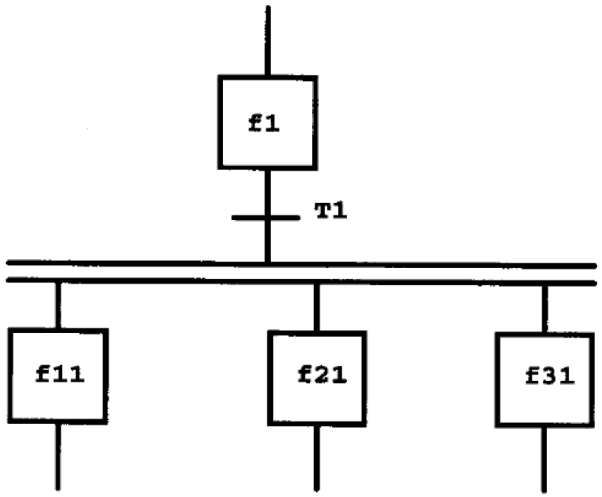
- La **convergenza** è la chiusura naturale della scelta
- Casi particolari di strutture scelta/convergenza sono il **salto** o il **ciclo** di una sequenza

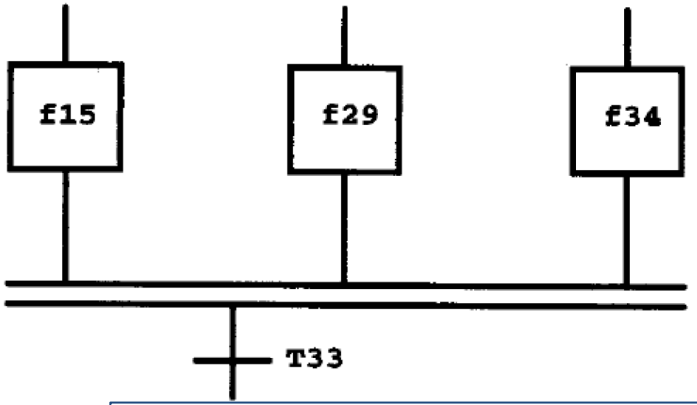


Strutture classiche di programmazione

Parallelismo e sincronizzazione

- Invece di scegliere tra due o più rami, posso farli eseguire **in parallelo**
- In questo caso avrò **più fasi** che seguono/precedono **un'unica transizione**
- In genere si utilizza una **doppia linea orizzontale**



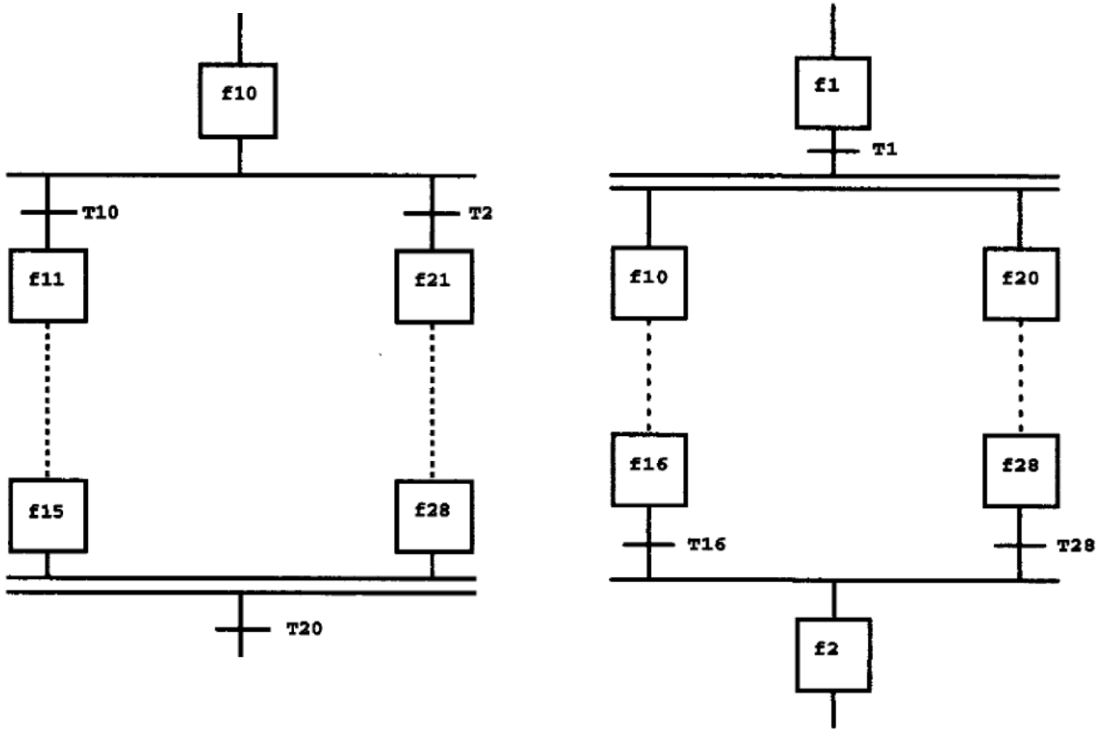


NB: la transizione è abilitata se tutte le fasi a monte sono attive!

Strutture classiche di programmazione

Errori da evitare

- Errori da evitare sono
  - **scelta seguita da sincronizzazione**
  - **parallelismo seguito da convergenza**

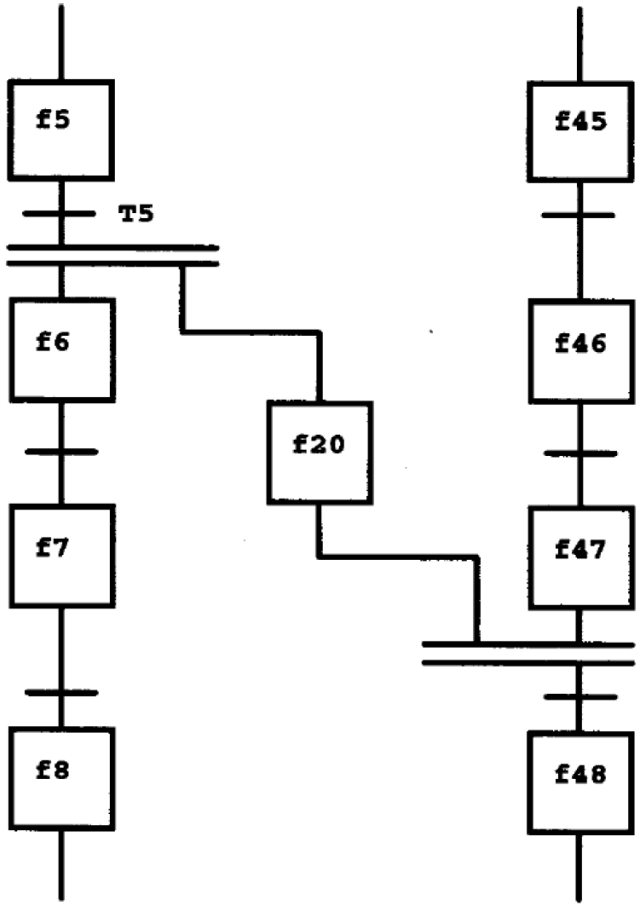


Strutture classiche di programmazione

Sincronizzazione locale

- Una **sincronizzazione locale** può essere utilizzata quando è necessario sincronizzare flussi di esecuzione indipendenti

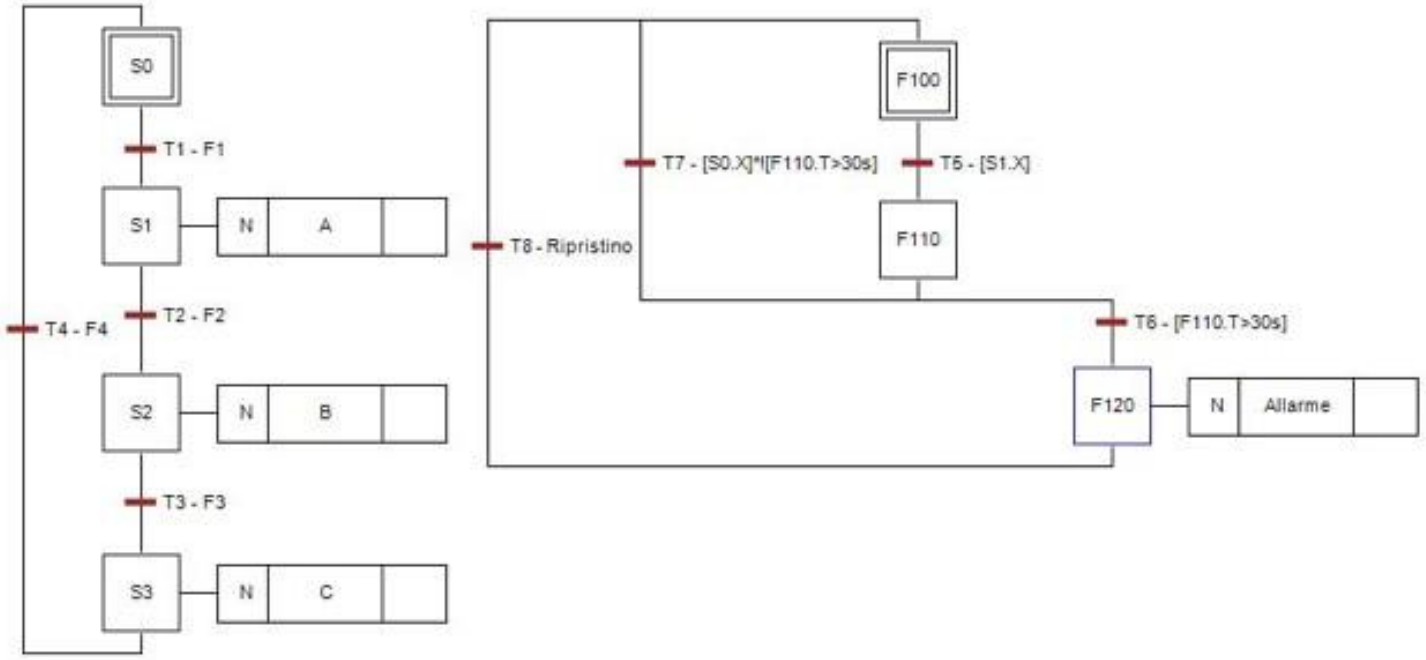
- es. la sequenza di destra, una volta raggiunta la fase `f47`,deve attendere che la sequenza di sinistra superi la transizione `T5` (`f20` è necessaria: senza, avrei due transizioni senza nessuna fase a separarle)



Strutture classiche di programmazione

Watchdog timer

- Un **watchdog timer** può essere utilizzato quando un'azione va eseguita entro un tempo prestabilito
- es. se F110 resta attiva per più di 30s, viene dato un allarme

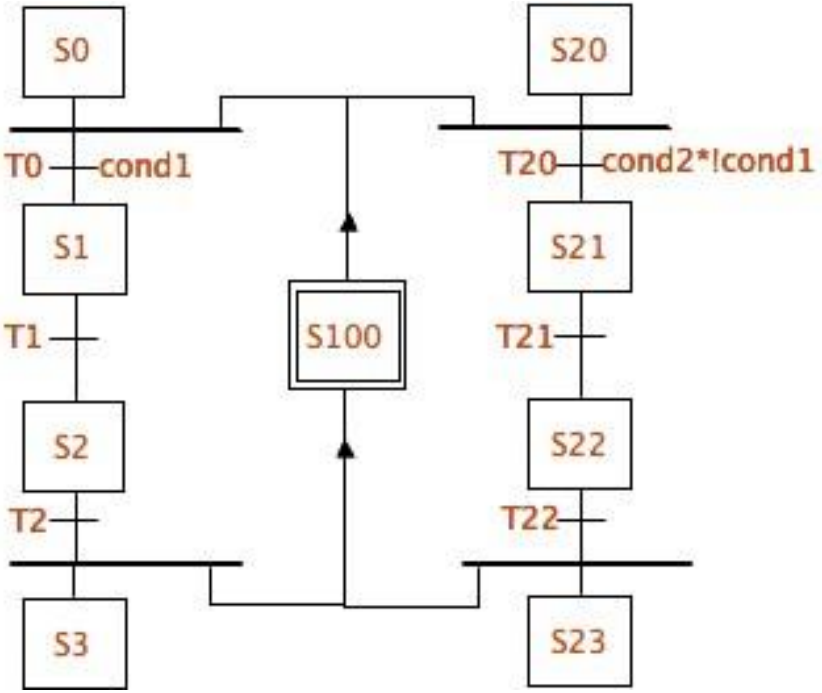


Strutture classiche di programmazione

Semaforo

- Un **semaforo** può essere utilizzato per rendere due sequenze **mutuamente esclusive**

- es. per superare le transizioni `T20` e `T0`, è necessario che `S100` sia attiva. `S100` è inizializzata a 1. Le transizioni `T0` e `T20` devono avere condizioni mutuamente esclusive



Macroazioni

- Le **macroazioni** consentono di realizzare un **controllo gerarchico** tra diversi SFC

- Una macroazione è un'azione tramite la quale un SFC modifica la condizione di un altro SFC di **livello gerarchico inferiore** (modificandone i marker di fase)

Macroazioni

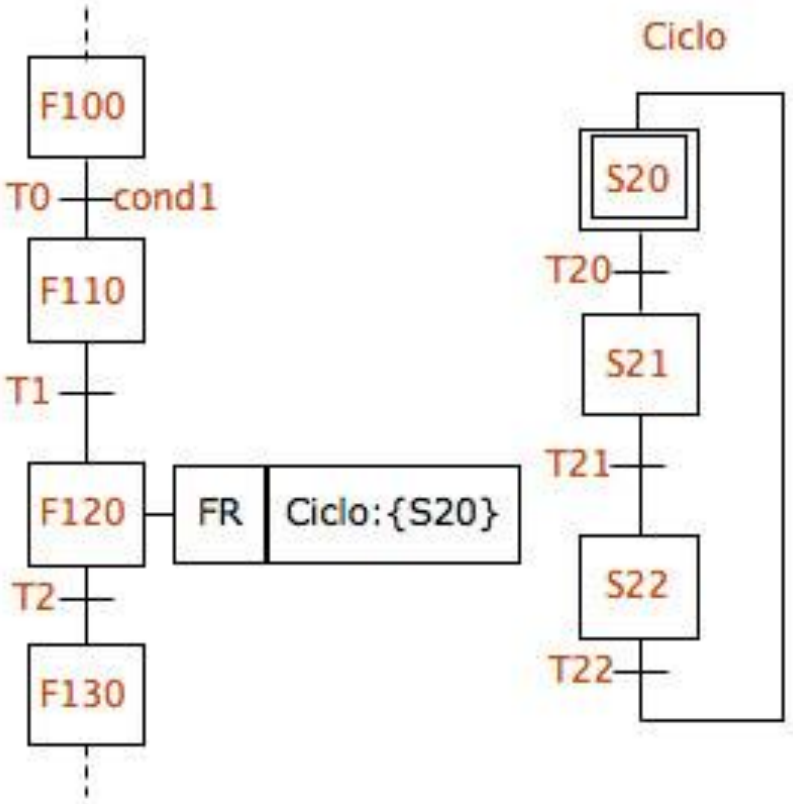
- Le macroazioni sono utili per gestire **situazioni critiche o di emergenza**

- Vedremo
  - Forzatura
  - Sospensione

Macroazioni
Forzatura

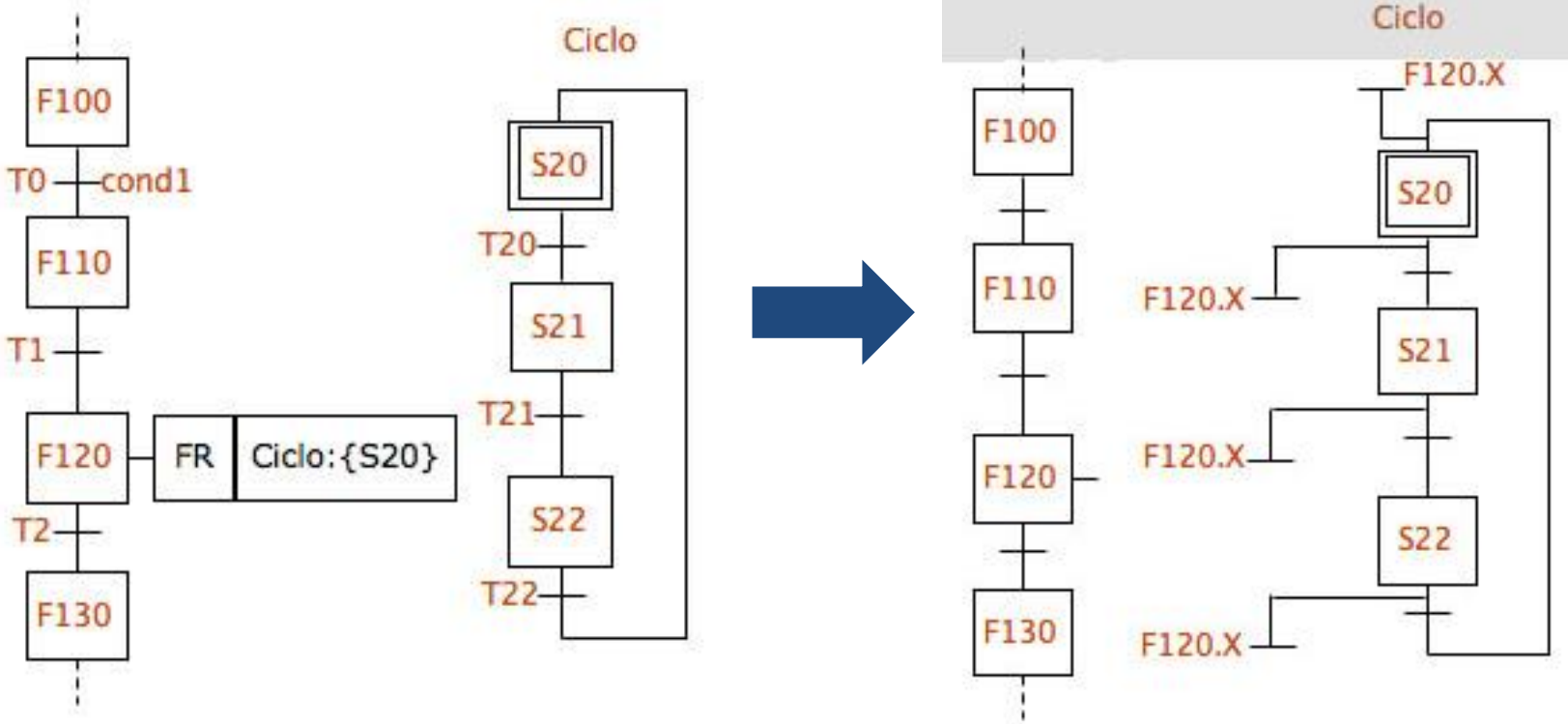
- Con la **forzatura**, un SFC impone una certa condizione a un altro

- es. l'SFC `Ciclo` viene forzato ad assumere la condizione in cui è attiva solo `S20`



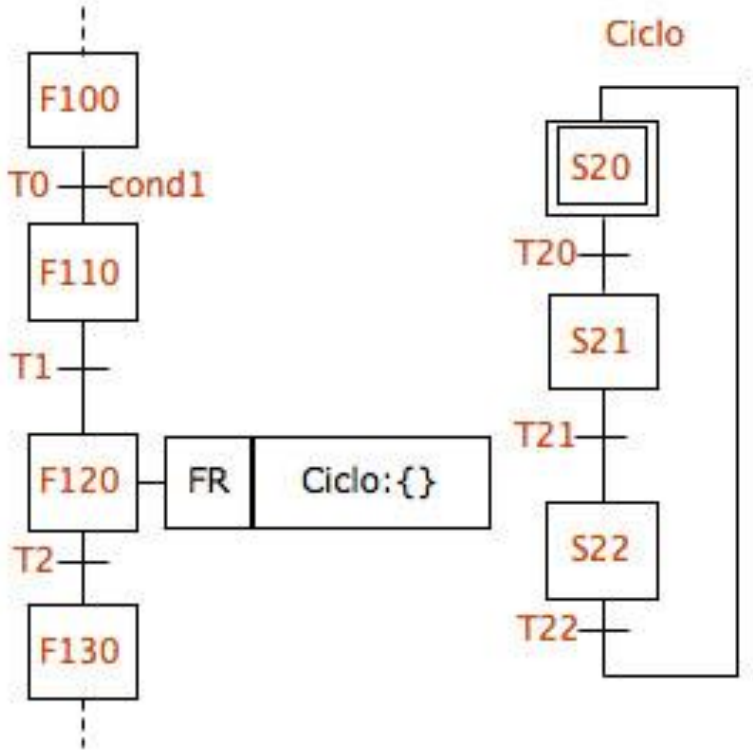
Macroazioni
Forzatura

- La forzatura può essere sostituita da un **insieme di transizioni** opportune



Macroazioni
Sospensione

- Con la **sospensione**, le fasi di un SFC vengono rese tutte inattive



Macroazioni
Sospensione

- Anche la sospensione può essere ottenuta con delle **transizioni** opportune
- Per riprendere l'esecuzione, sarà necessaria una **forzatura** che riattivi qualche fase

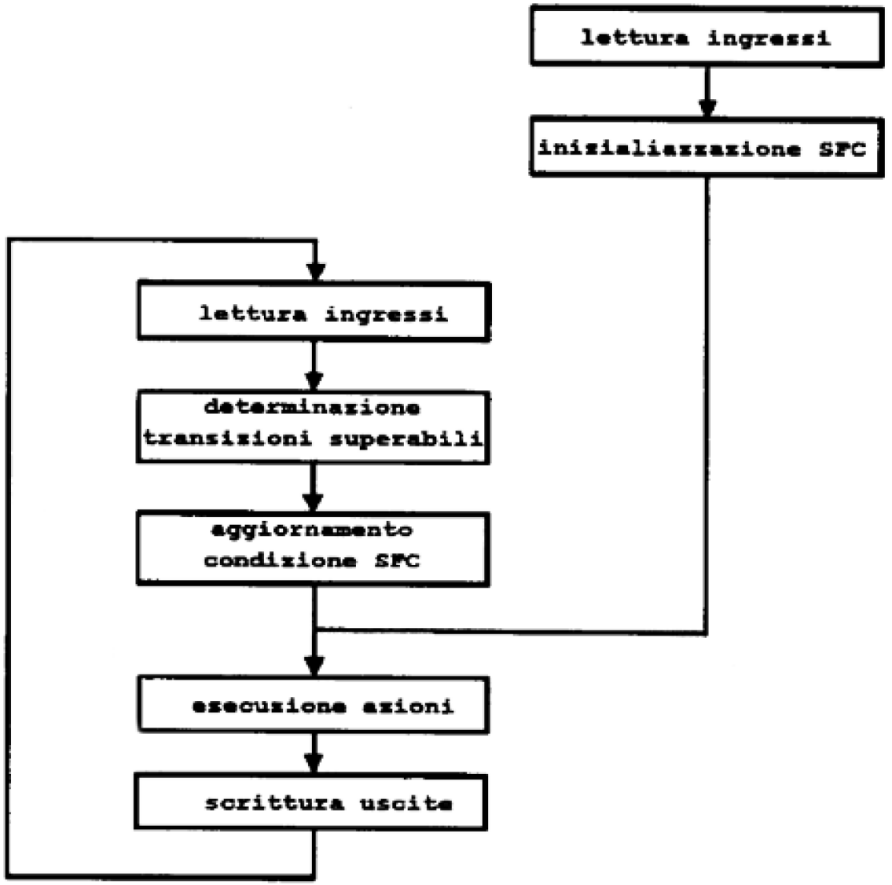
Traduzione in ladder

- Alcuni dispositivi potrebbero non prevedere l'SFC tra i possibili linguaggi
- Questo è vero, ad esempio, per dispositivi molto vecchi o piccoli e molto semplici
- Vedremo come **tradurre gli SFC in ladder** (uno dei linguaggi più diffusi)

Traduzione in ladder

- Bisogna innanzitutto identificare l'**algoritmo di evoluzione** dell'SFC

- È molto simile al **ciclo a copia massiva di ingressi e uscite**



Traduzione in ladder

Tale algoritmo prevede

- Una fase di **inizializzazione**
  - Lettura degli ingressi
  - Attivazione delle fasi iniziali
  - Esecuzione delle azioni associate
- Un **ciclo iterativo**
  - Aggiornamento degli ingressi
  - Superamento delle transizioni
  - Aggiornamento della condizione dell'SFC
  - Esecuzione delle azioni
  - Aggiornamento delle uscite

Traduzione in ladder

- Dobbiamo a questo punto **codificare l'algoritmo di evoluzione**

- Il programma ladder risultante avrà **quattro sezioni**:
  1. inizializzazione
  2. valutazione delle transizioni
  3. aggiornamento della condizione
  4. esecuzione delle azioni

Traduzione in ladder

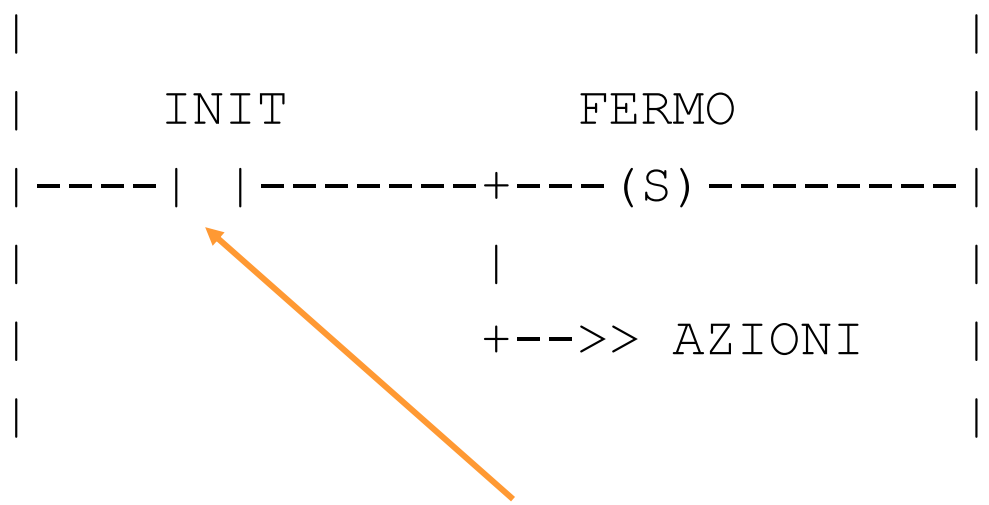
- A ogni fase sarà associata una variabile che funga da **marker di fase**
- A ogni transizione sarà associata una variabile che funga da **segnalatore della transizione**

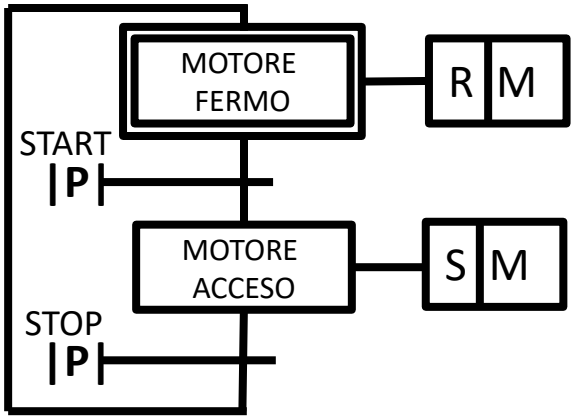
Traduzione in ladder

- **Inizializzazione**
  - Eseguita un'**unica volta**
  - Vanno messi a 1 i bit associati ai marker delle **fasi iniziali** (→ bobine (S))
  - Al termine, avrà un salto condizionato alla **sezione delle azioni**

Traduzione in ladder
Esempio

Inizializzazione





Assumerò che il dispositivo metta a disposizione una variabile INIT che è vera solo al primo ciclo di scansione

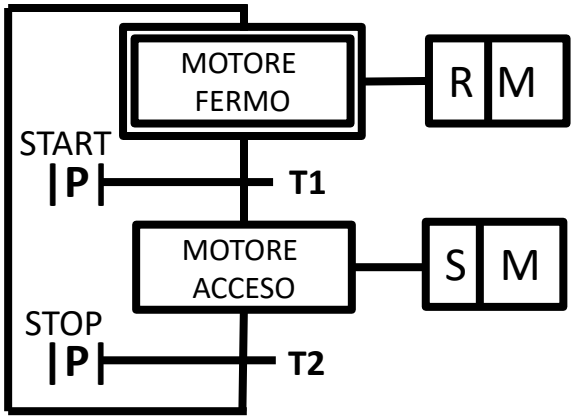
Traduzione in ladder

- **Valutazione delle transizioni**
  - Valutazione dello **stato di superabilità delle transizioni** e aggiornamento dei relativi segnalatori
  - È necessaria **un'istruzione per ogni transizione**

Traduzione in ladder
Esempio

Valutazione delle transizioni

```
|                                          |
|  FERMO     START        T1               |
|--|  |-------|  |--------(  )------|
|                                          |
|  ACCESO     STOP        T2               |
|--|  |-------|  |--------(  )------|
|                                          |
```



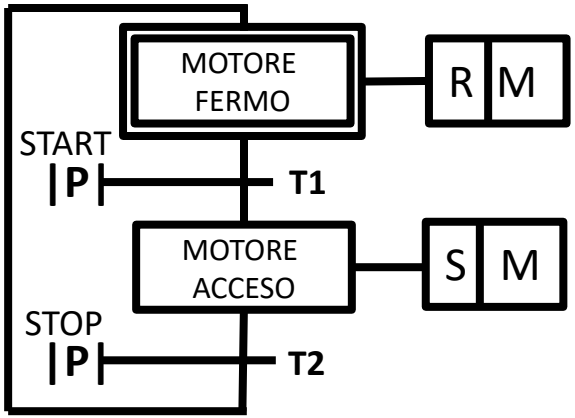
Traduzione in ladder

- **Aggiornamento della condizione**
  - Aggiornamento della **condizione dell'SFC**, disattivando le fasi a monte delle transizioni superabili e attivando quelle a valle
  - Se la transizione è superabile:
    - un'istruzione per ogni transizione per **disattivare le fasi a monte**
    - **poi** un'istruzione per ogni transizione per **attivare le fasi a valle**

Traduzione in ladder
Esempio

Aggiornamento della condizione

```
|  (*  Disabilitazioni *)  |
|  T1              FERMO   |
|--|  |-----------(R)-----  |
|                          |
|  T2              ACCESO  |
|--|  |-----------(R)-----  |
|                          |
|    (*  Abilitazioni *)   |
|  T1              ACCESO  |
|--|  |-----------(S)-----  |
|                          |
|  T2              FERMO   |
|--|  |-----------(S)-----  |
|                          |
```

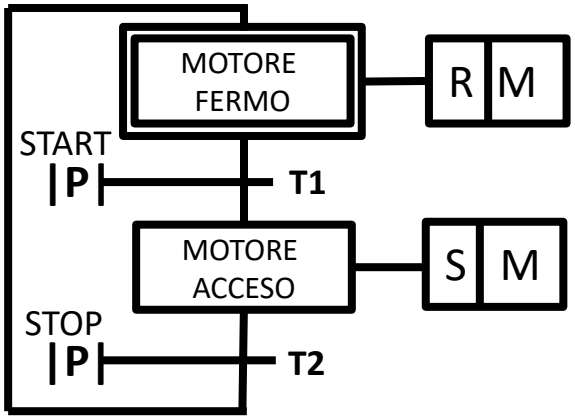


Traduzione in ladder

- **Esecuzione delle azioni**
  - Esecuzione delle **azioni delle fasi attive**
  - Preceduta dall'etichetta a cui punta il **salto nella sezione di inizializzazione**
  - Si useranno i **marker delle fasi attive come condizione** per eseguire una determinata azione

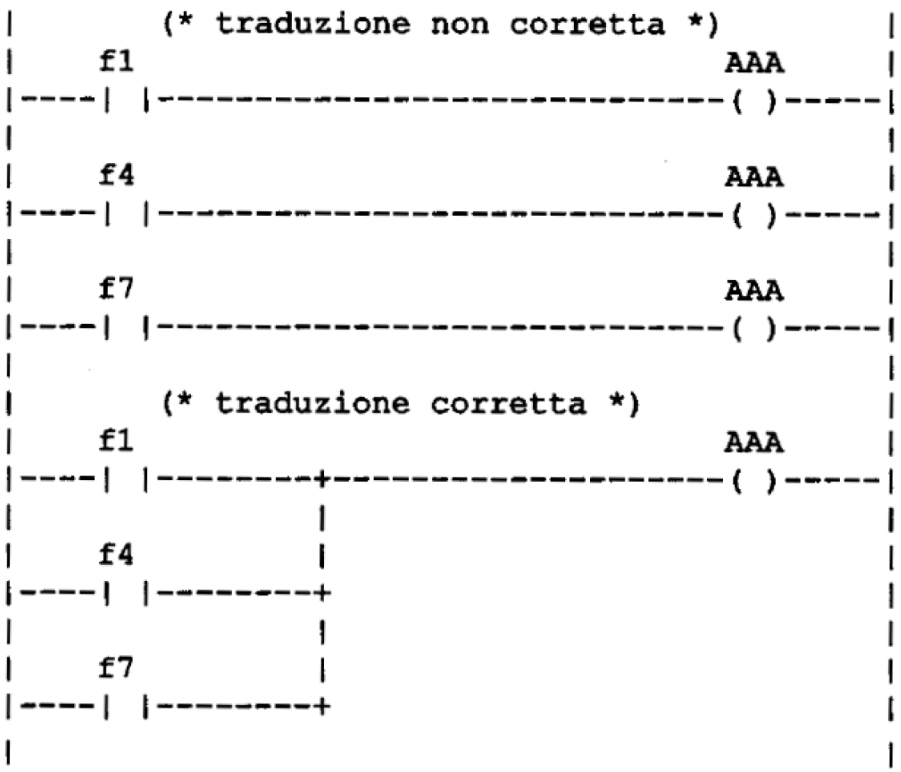
Esecuzione delle azioni

```
| AZIONI:                          |
|                                  |
|    FERMO                  M      |
|-----|  |-------------(R)-----|
|                                  |
|    ACCESO                 M      |
|-----|  |-------------(S)-----|
|                                  |
```



Traduzione in ladder

- Se più fasi **agiscono su una stessa variabile**, questo comportamento va realizzato con un **OR dei relativi marker** nella fase di esecuzione delle azioni
- Nel primo esempio vale solo la condizione espressa nell'ultimo rung!



Traduzione in ladder

- **Azioni temporizzate** (qualificatori L o D) possono essere realizzate con **blocchi funzionali di temporizzazione**

- Se un SFC è composto da più **grafi non connessi**, si può tradurre ciascun grafo e poi mettere insieme le sezioni corrispondenti in maniera opportuna

Esercizi

I testi [1] e [3] forniscono vari esempi di progettazione di SFC.

- Provare a realizzarne qualcuno in **OpenPLC**
- Provare a tradurre qualcuno degli esempi in **ladder**

Risorse e Riferimenti

- [1] Cap. 7
- [3] Cap. 4 e Appendice A

- TSA@federica [lezioni 10-12]
- SFC @Bedrock Automation
- https://slideplayer.it/slide/1008828/

Fine Lezione #10

Sequential Functional Chart