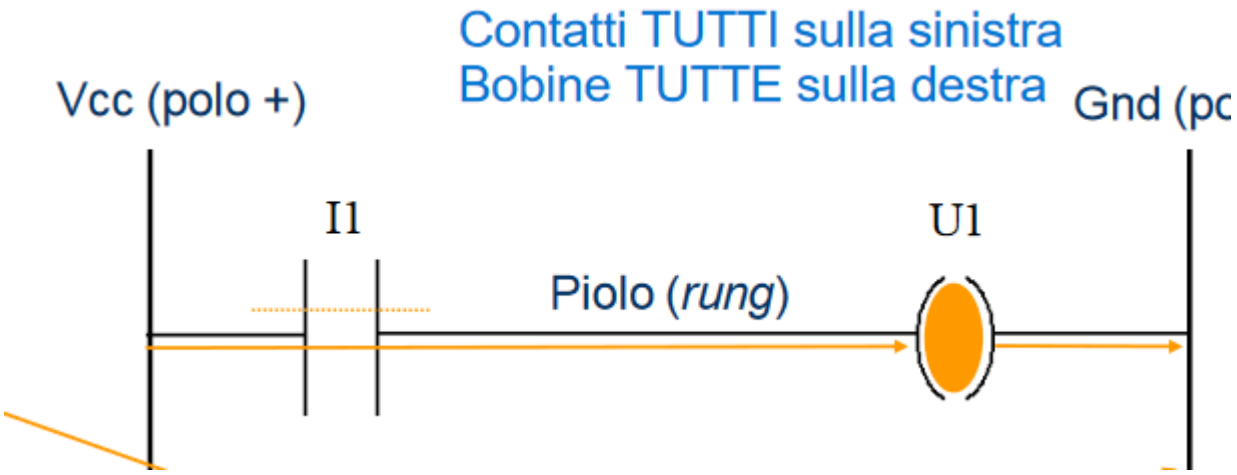
Ladder Diagram

Pagina Iniziale: Automazione Industriale

Indice: Index

Linguaggio grafico che si basa sulla trasposizione di una rete elettrica molto semplice in logica di programmazione(Uso di Contatti e Bobine)
Presenza di ingressi e uscite, sulle uscite posso scrivere e leggere mentre sulle entrate posso solo leggere

Elementi Base



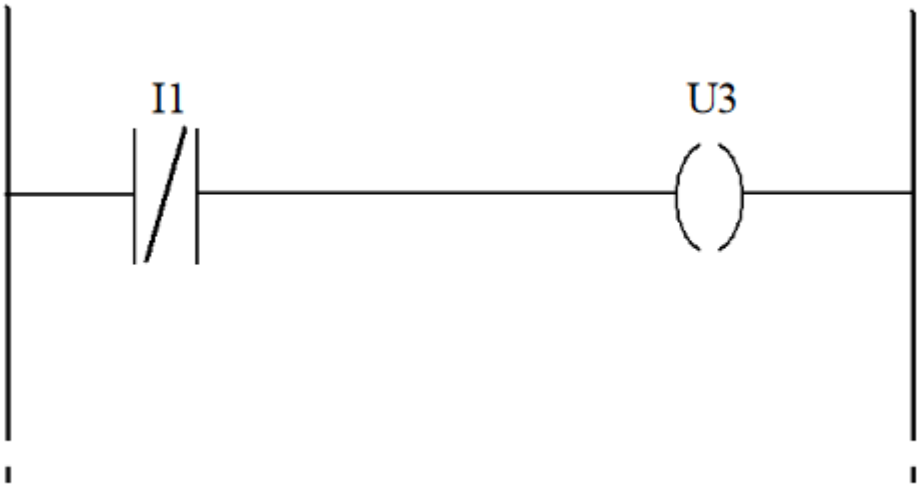
Contatti
Rappresentano gli ingressi e vengono inseriti sulla sinistra del piolo, normalmente un contatto senza una sbarra all'interno rappresenta un Contatto aperto e dunque il bit associato risulta 0 se aperto 1 se chiuso. Il Contatto chiuso è rappresentato con una sbarra all'interno e ha un comportamento inverso rispetto al Contatto aperto
Bobine
Rappresentano le uscite e sono presenti alla destra del piolo
si attivano al passare della corrente, quindi il bit associato vale 0 se le condizioni logiche alla sua sinistra non sono verificate, 1 altrimenti
Esistono anche le bobine Set(Mantiene lo stato logico alto anche quando le condizioni di attivazione vengono a mancare) e Reset(Riportano lo stato logico a 0)
Programma
Insieme di pioli letti dall'alto verso il basso e da sinistra verso destra
Il flusso di energia in un piolo va solo in una direzione senza possibilità di invertirsi

Ciclo a Coppia Massiva

Ad ogni ciclo vengono letti gli ingressi ed aggiornate le variabili del programma, viene eseguita la sequenza di pioli e vengono scritte le uscite in accordo con i valori del programma
Il valore delle variabili lette in ingresso rimane costante per tutto il ciclo del programma dunque l'invertire delle istruzioni che leggono e scrivono su un'uscita in sequenza porterà a risultati differenti

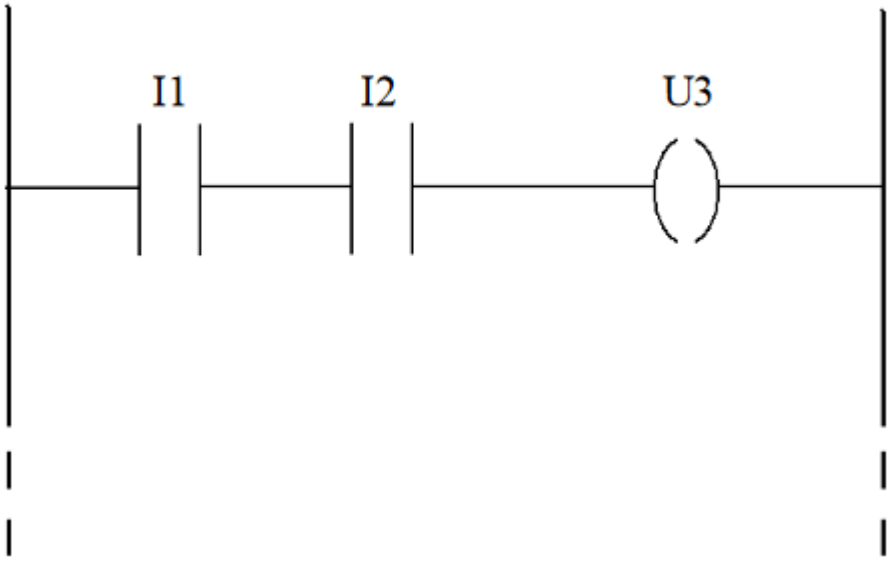
Funzioni Logiche

NOT



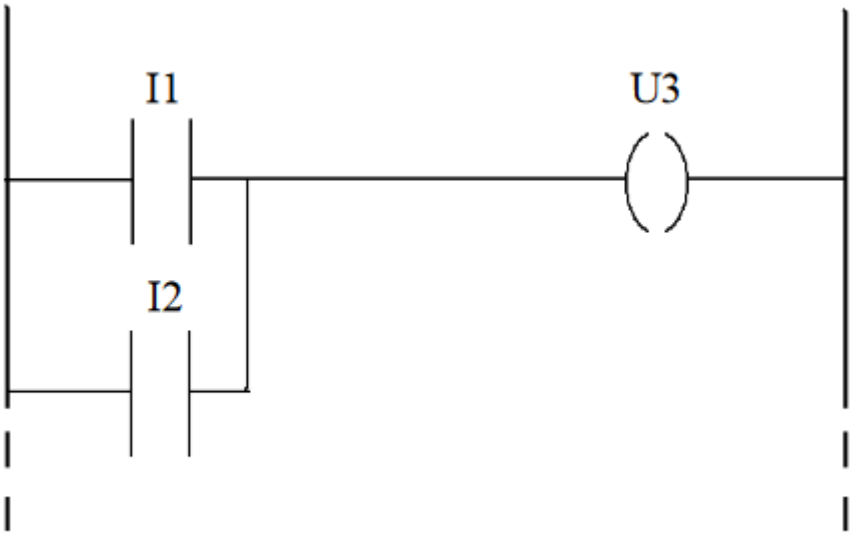
U3 = NOT (I1)

AND



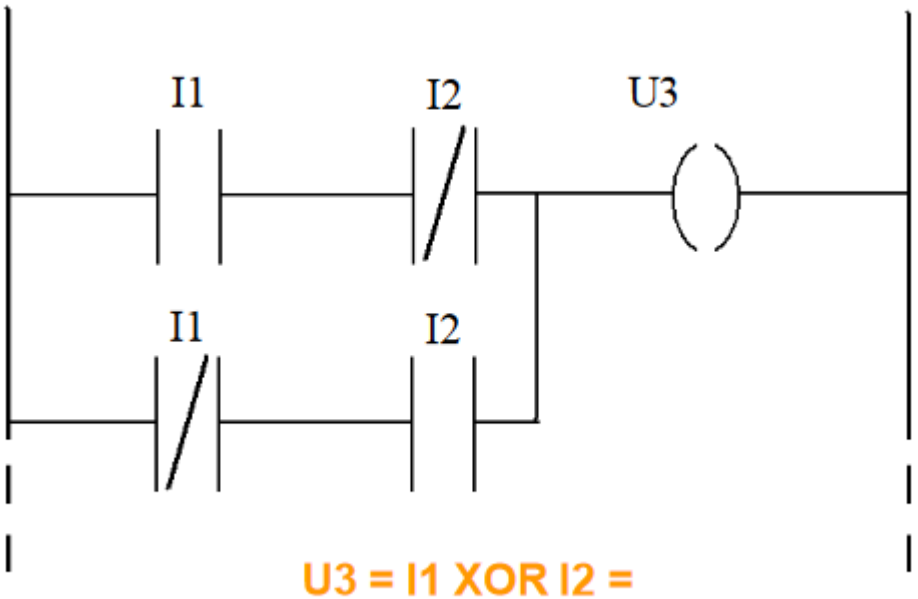
U3 = I1 AND I2

OR



U3 = I1 OR I2

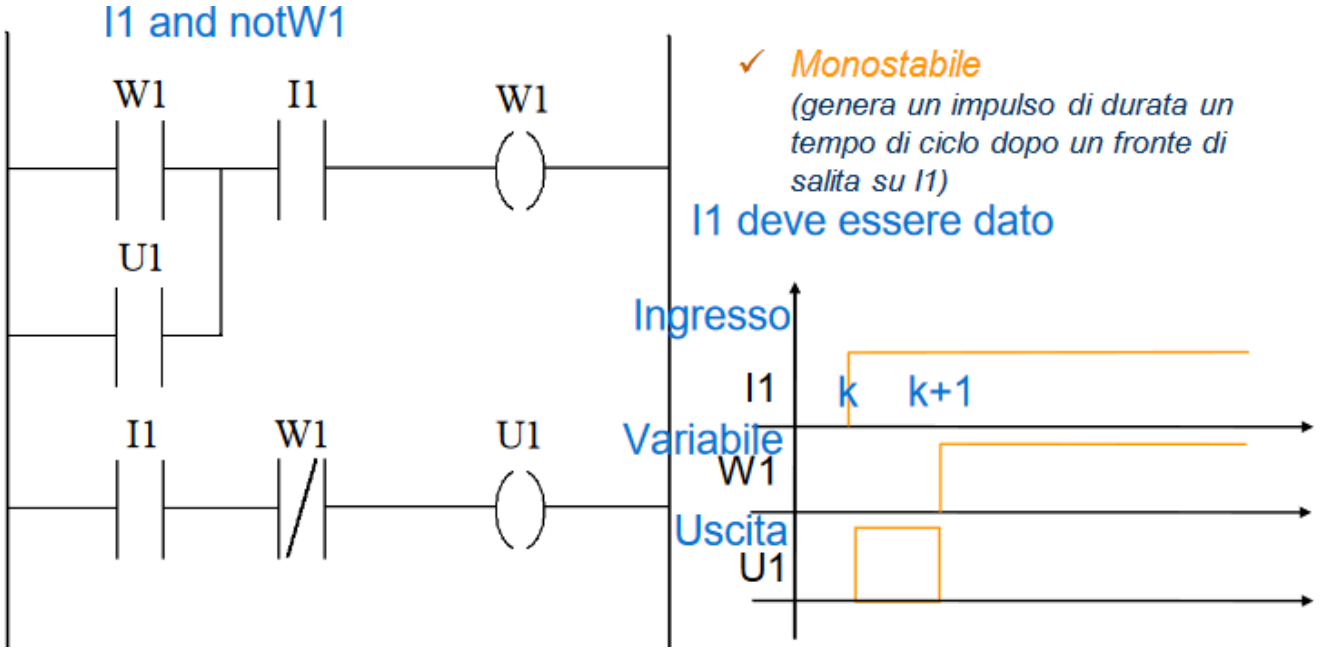
XOR

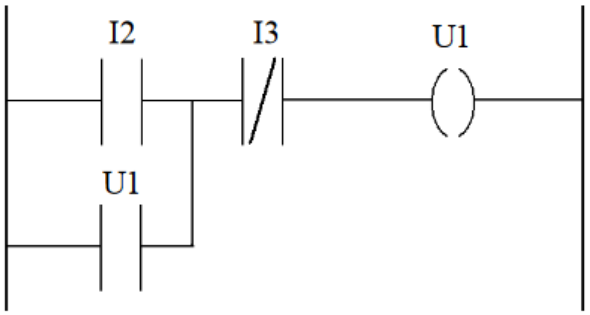


U3 = I1 XOR I2 =
[I1 AND NOT(I2)] OR [NOT(I1) AND I2]

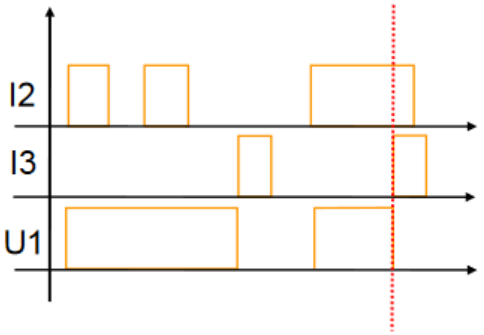
Elementi Dinamici

Possibile associare ad un contatto anche una variabile di uscita che insieme all'esecuzione sequenziale e ciclica di uno schema LD permette la creazione di elementi dinamici(o con memoria) e l'identificazione di variazioni di una variabile
Una variabile una volta calcolata non cambia il suo valore fino al ciclo successivo dove potrà essere resettata o mantenuta

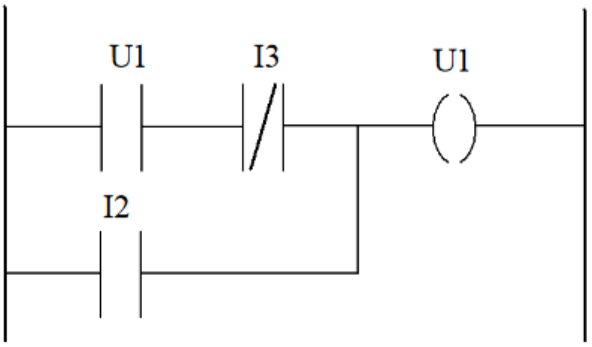




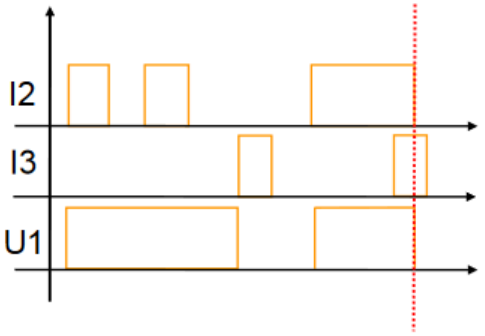
✓ *Flip-Flop a reset vincente*
I2 set
I3 reset
Il reset determina se la mia uscita sia alta o bassa

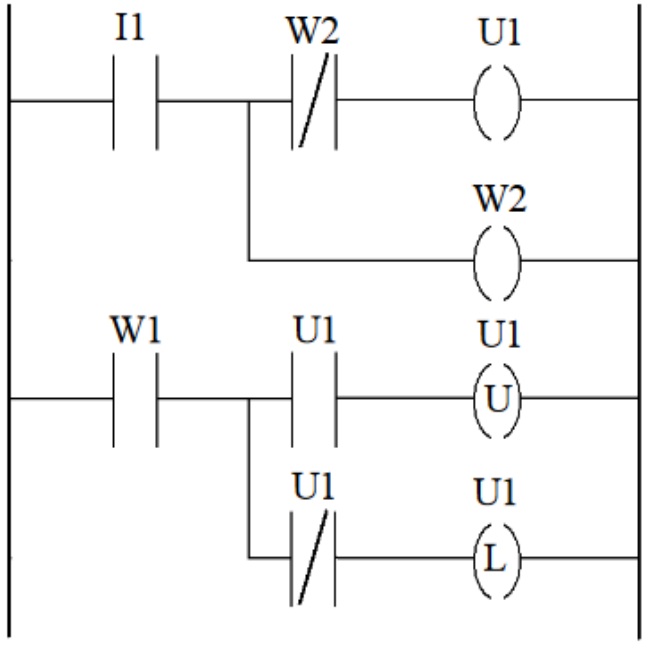


Utile in una sequenza(Dire che sono qui)

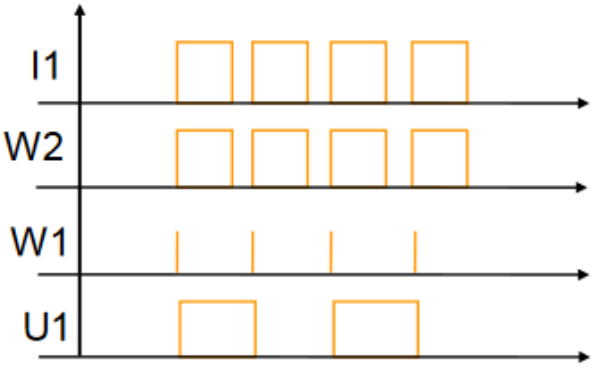


✓ *Flip-Flop a set vincente*
I2 set
I3 reset
Quando il set si alza forza l'uscita ad alto

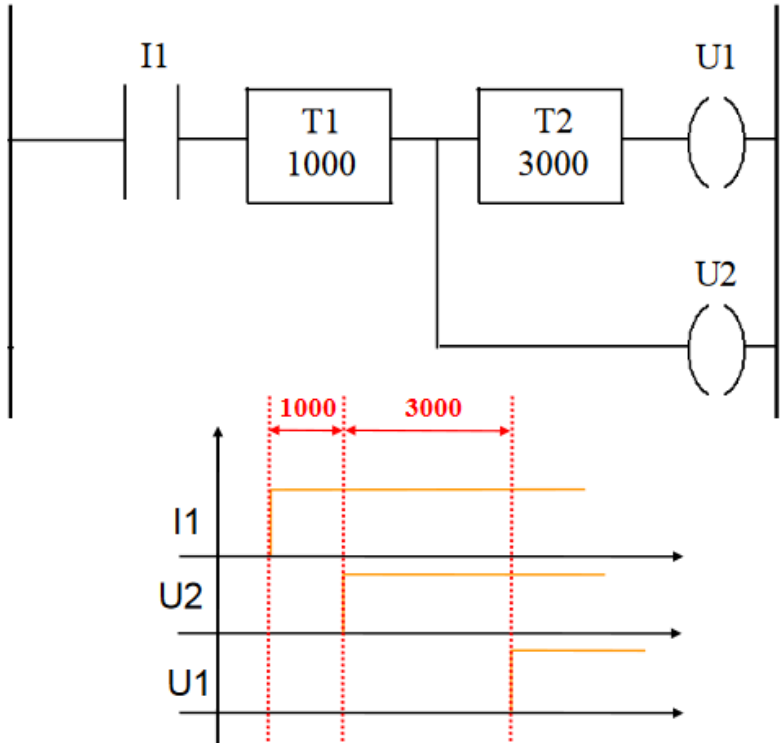




✓ *Flip-flop di tipo D*
(un impulso su I1 fa commutare l'uscita)

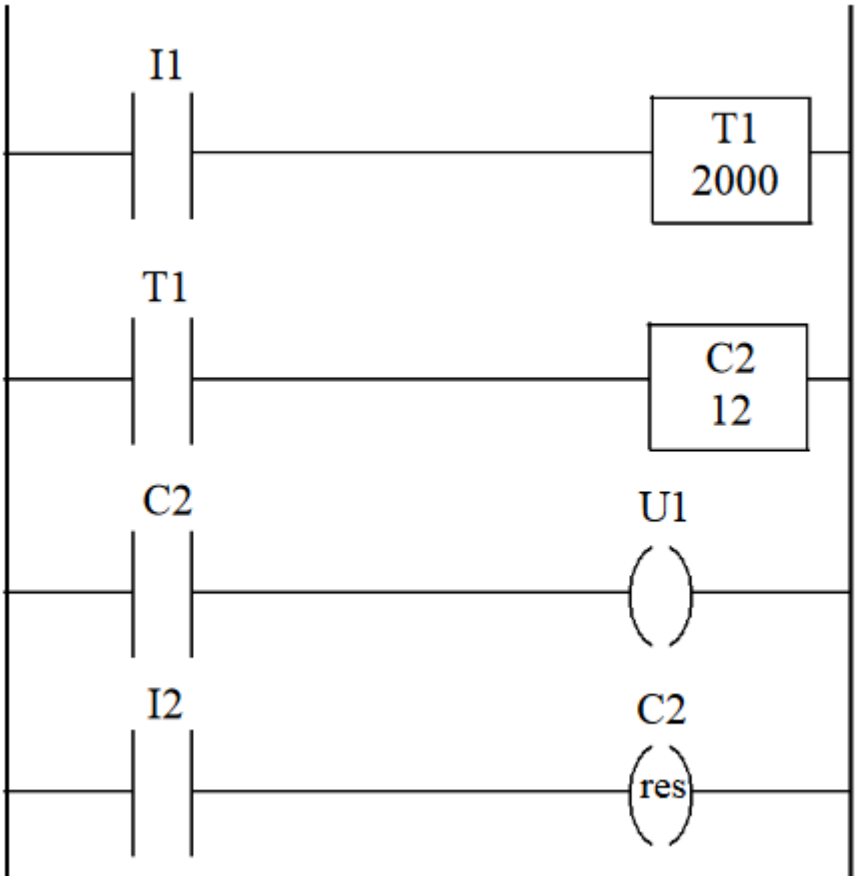


Istruzioni di temporizzazione



Il timeout può essere inserito sia in dentro il blocco che tramite un filo collegato al blocco

Sono compiute o da temporizzatori o da contatori:
Temporizzatore(Tx): se il piolo consente il fluire della corrente conta il trascorrere del tempo fino ad un valore preimpostato(Presenta una soglia, è resettabile), allo scadere del tempo Tx diventa vero. Se il piolo si disattiva prima dello scadere del timer esso si disattiva. IN Tx.acc si può leggere il tempo trascorso
Un temporizzatore a ritenuta(TxR) continua a contare anche se il piolo si disattiva
Reset del temporizzatore(RES) ferma il temporizzatore e lo inizializza a 0
Contatore ad incremento incrementa il valore solo sul fronte di salita(piolo subisce una transizione falso-vero), Cx.acc contiene il valore attuale del contatore e Cx diventa vero quando il contatore raggiunge il valore preimpostato
Il segnale di reset(RES) riporta a 0 il contatore Cx



Funzioni e Blocchi Funzione

Istruzioni complesse simili alle funzioni di un linguaggio di programmazione di alto livello vengono racchiuse in blocchi collegati con un piolo e il loro uso è immediato

Traduzioni Schema SFC in Ladder

Si segue l'algoritmo di evoluzione senza ricerca di stabilità
Ad ogni passo si associa un bit di memoria rappresentante lo stato del passo
Ad ogni transizione si associa un bit di memoria rappresentante la superabilità della transizione
Scomposizione programma in quattro sezioni:

1. Sezione di inizializzazione
2. Sezione di esecuzione delle azioni
3. Sezione di valutazione delle transizioni
4. Sezione di aggiornamento dello stato
   Non permette una buona rappresentazione del superamento di più transizioni contemporaneamente
   Per eseguire più schemi SFC in Ladder basta aggiungere gli schemi tradotti dei relativi schemi SFC uno sotto l'altro(Ladder è intrinsecamente parallelo)
   Ladder è migliore per algoritmi molto semplici
   SFC più adatti per programmi con sequenze/parallelismi e operazioni algebriche