Lezione #9

Ladder Diagram

Sommario

- Introduzione e generalità
- Elementi del LD (bobine, contatti)
- Esempi
- Altre istruzioni (controllo di flusso, terminazione…)

Ladder Diagram

- Il **LD** fu introdotto per rendere la transizione ai nuovi controllori programmabili più naturale per i tecnici dell'epoca

- I primi "componenti" disponibili erano
  - **Contatti** (aperti o chiusi)
  - **Bobine** (di eccitazione di un relè)
  - Temporizzatori
  - Contatori

Ladder Diagram

- Il nome deriva dalla sua forma **a scala**:
  - i montanti rappresentano le **linee di alimentazione**
  - i pioli (*rung*) alimentano una **bobina** se una certa combinazione di **contatti** abilita il flusso di energia

Componenti del LD
Contatti

- I **contatti** possono essere associati a variabili booleane (bit)
  - **interne**
  - **esterne** (e.g. un pulsante)

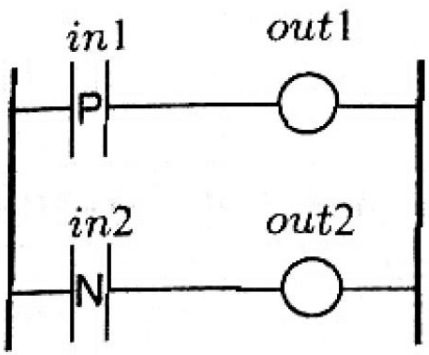
- La **chiusura** di un contatto nella zona di test **consente il flusso di energia** verso la zona di azione relativa

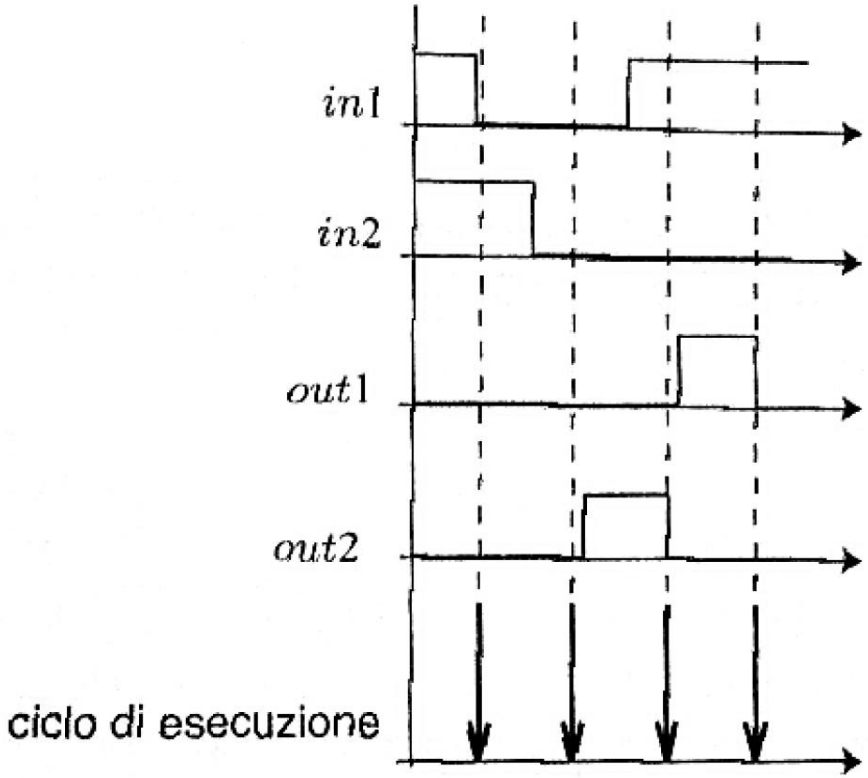
Componenti del LD
Contatti

- Contatti disponibili
  - `| |` **normalmente aperto**
    - Si chiude se il bit associato è 1
  - `|/|` **normalmente chiuso**
    - Si chiude se il bit associato è 0
  - `|P|` **a fronte di salita**
    - Si chiude quando il bit associato passa da 0 a 1
  - `|N|` **a fronte di discesa**
    - Si chiude quando il bit associato passa da 1 a 0

Componenti del LD
Contatti

- Contatti a fronte di salita/discesa





Componenti del LD
Bobine

- Le **bobine** possono essere usate per operare su un bit
- Anche in questo caso, il bit può corrispondere a una **variabile interna** o a un'**uscita**
- Se la bobina è **alimentata,** agisce sulla variabile associata

Componenti del LD
Bobine

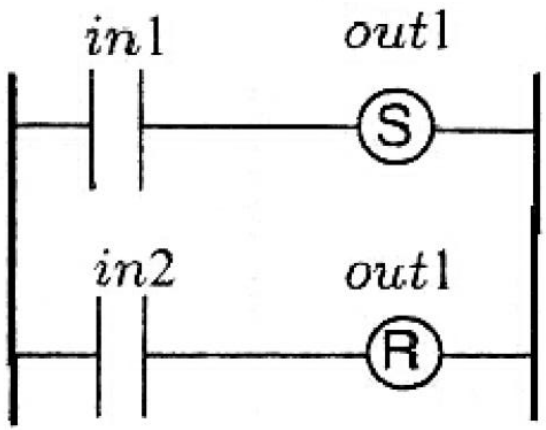
- Bobine disponibili
  - `( )` **semplice**
    - Se alimentata il bit associato è 1, altrimenti 0
  - `(/)` **negata**
    - Se alimentata il bit associato è 0, altrimenti 1
  - `(S)` **a memorizzazione (SET)**
    - Quando viene alimentata mette il bit associato a 1
    - Il bit resta 1 anche quando la bobina non è più alimentata
  - `(R)` **a memorizzazione (RESET)**
    - Quando viene alimentata mette il bit associato a 0
    - Il bit resta 0 anche quando la bobina non è più alimentata

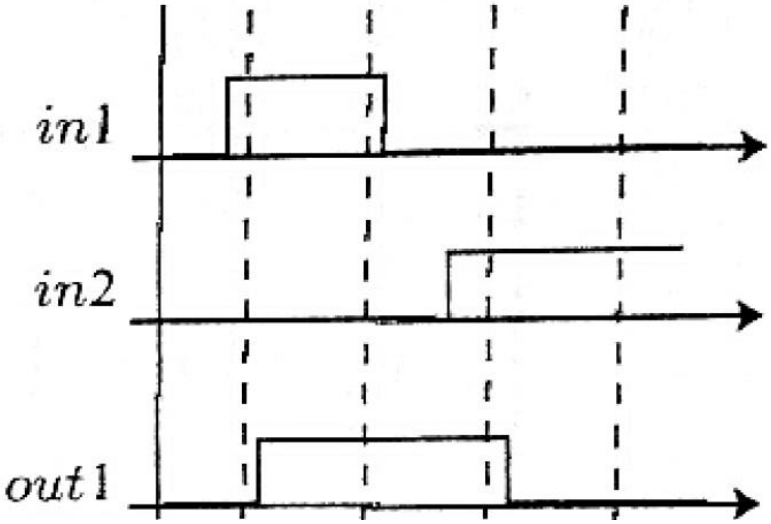
Componenti del LD
Bobine

- Bobine disponibili
  - `(P)` **a fronte di salita**
    - Il bit associato è 1 quando la bobina passa da non alimentata ad alimentata
    - Resta a 1 per l'intervallo tra due esecuzione consecutive
  - `(N)` **a fronte di discesa**
    - Il bit associato è 1 quando la bobina passa da non alimentata ad alimentata
    - Resta a 0 per l'intervallo tra due esecuzione consecutive
  - `(M)` `(SM)` `(RM)` **a ritenuta**
    - Corrispondono a `( )`, `(S)`, `(R)` ma il valore viene conservato in caso di mancata alimentazione del PLC
    - Equivalente a dichiarare la variabile associata come `RETAIN`

Componenti del LD
Bobine

- Bobine Set/Reset

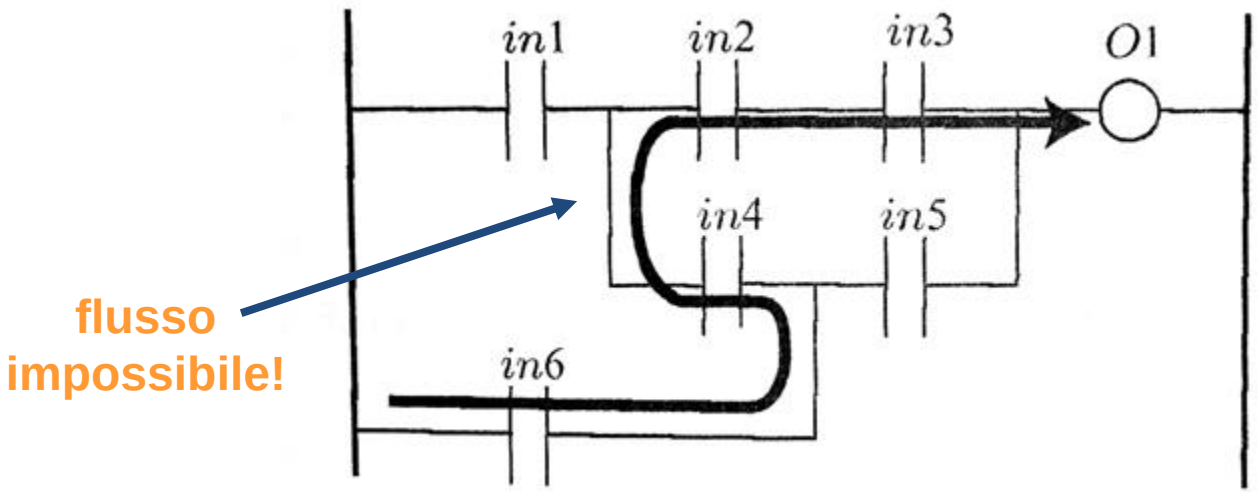




Componenti del LD

- Il flusso di energia procede **dall'alto verso il basso e da sinistra** (*zona di test*) **verso destra** (*zona di azione*)

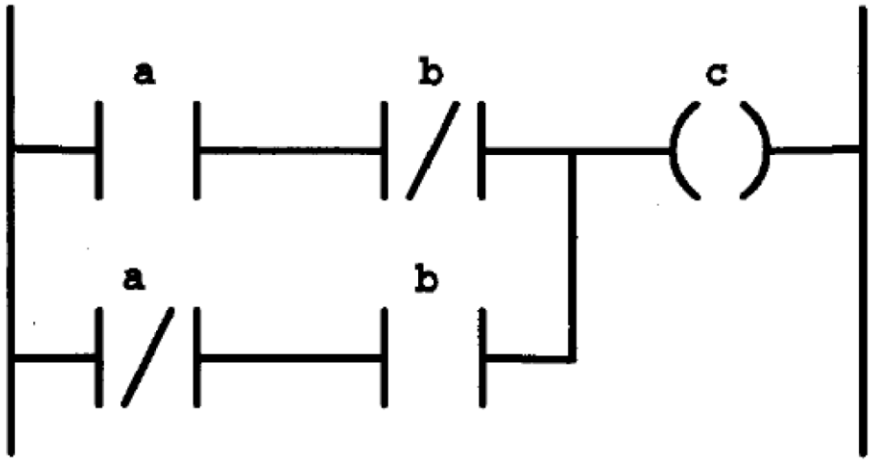
  → questo elimina possibili ambiguità

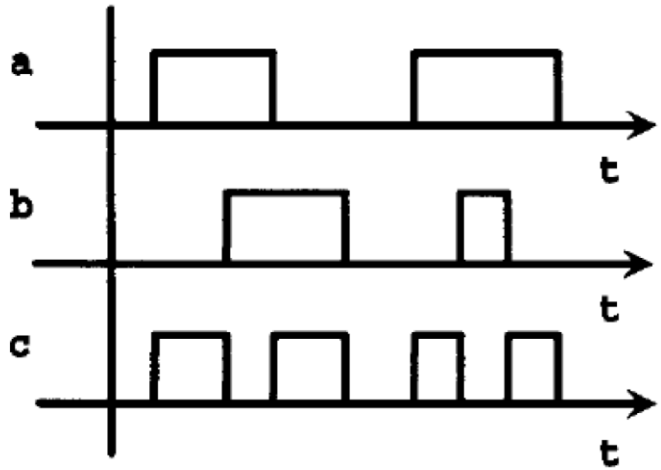


Esempi

XOR

- `c:= (a AND NOT(b)) OR (NOT(a) AND b)`

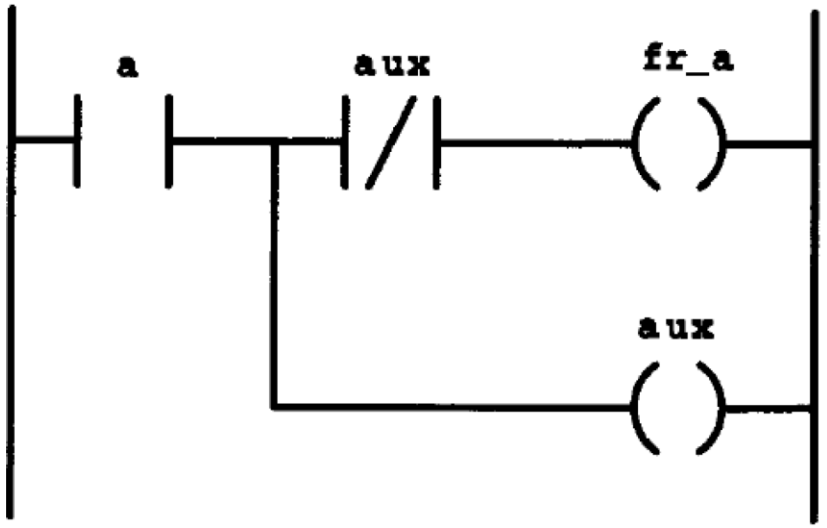


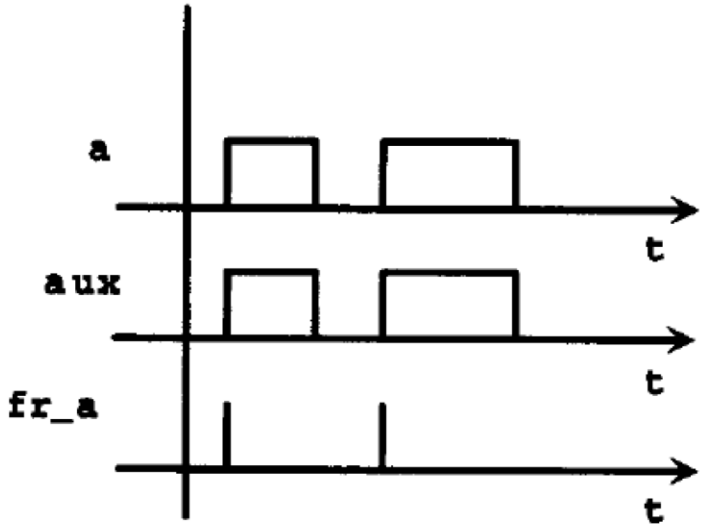


Esempi

Riconoscimento di un fronte di salita

- `fr_a := NOT(aux) AND a`
- `aux := a`

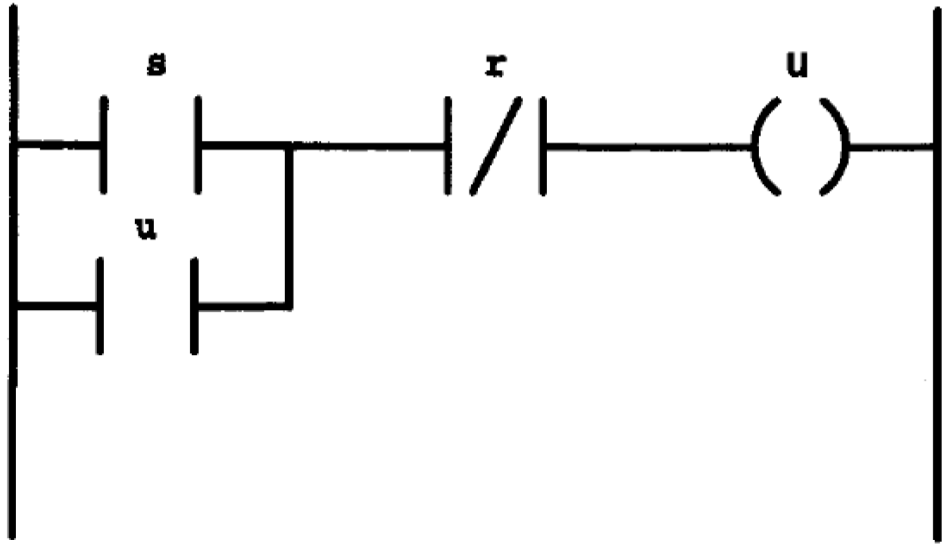




Esempi

Flip flop RS

- `u:= NOT(r) AND (s OR u)`
- La u al secondo membro è relativa all'esecuzione precedente (rete eseguita in modalità ciclica o periodica)



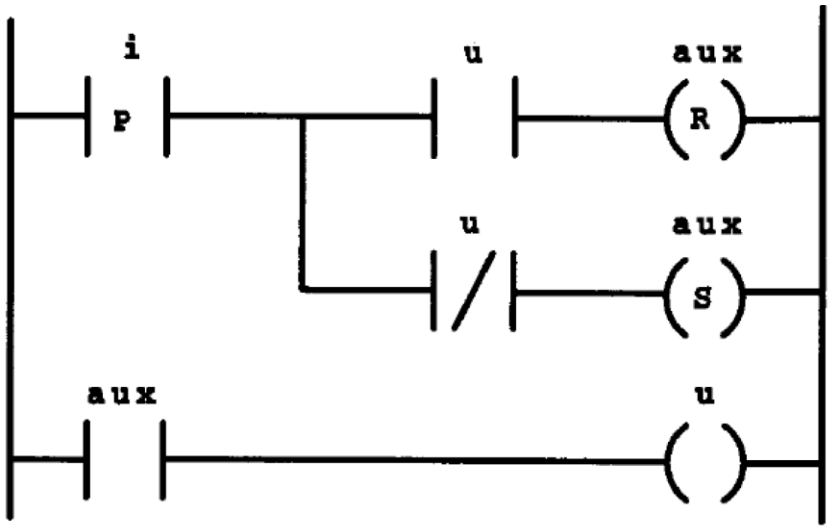
Esempi

Flip flop a commutazione

```
fr_i := NOT(i_precedente) AND i;
i_precedente := i;
IF fr_i THEN
   IF u THEN
      u := FALSE;
   ELSE
      u := TRUE;
   END_IF
END_IF
```

Riconoscimento fronte di salita

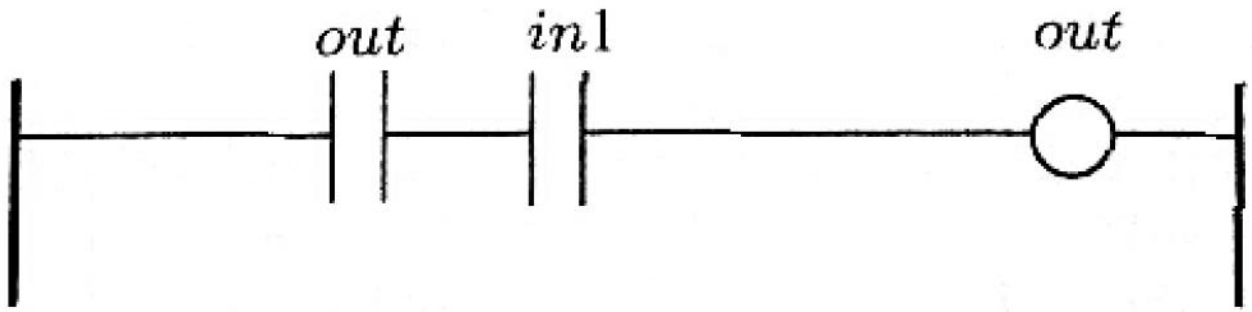
In ST l'istruzione condizionata consente di fare a meno della variabile ausiliaria



Rung con feedback

- Il **funzionamento ciclico** che caratterizza l'esecuzione di un programma in un PLC permette assegnamenti sequenziali come *feedback* senza ambiguità

- Es. `out(k) := in1(k) AND out(k-1)`



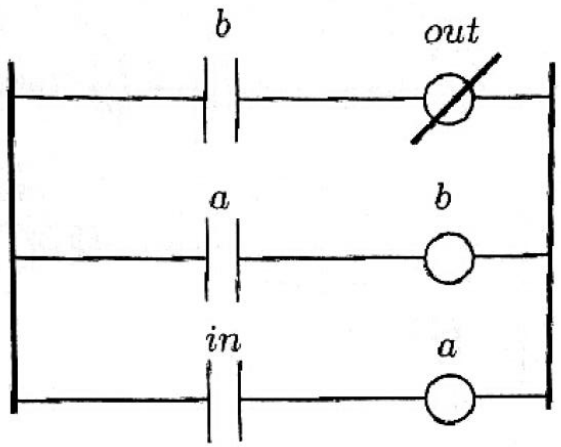
Esempio: ritardo

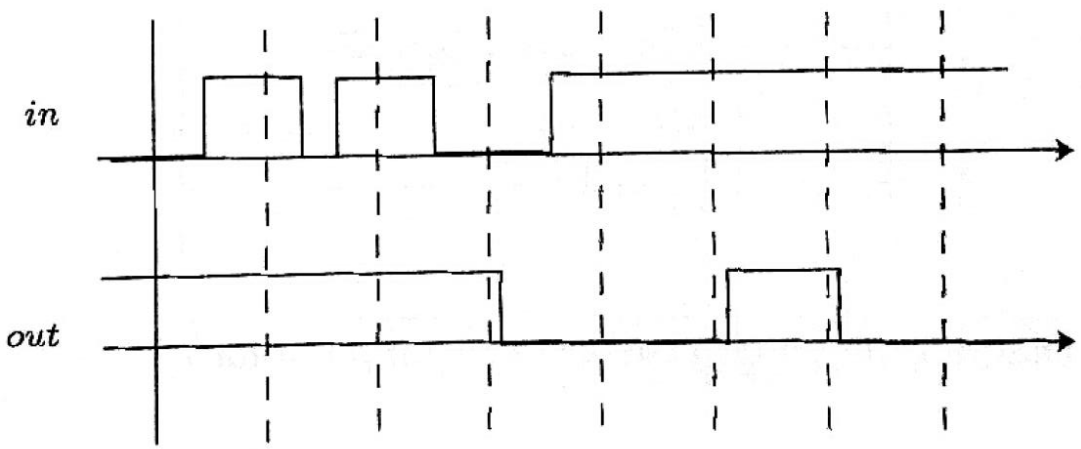
- Es. `NOT(out(k)) := b(k-1)`

`b(k) := a(k-1)`

`a(k) := in(k)`

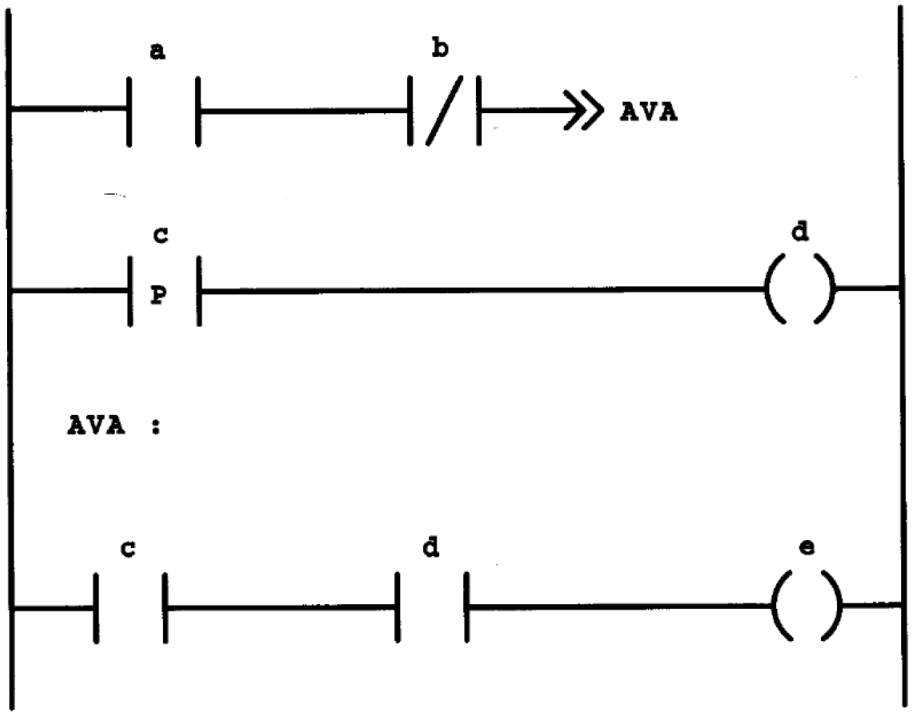
➔ `out(k) = NOT(in(k-2))`





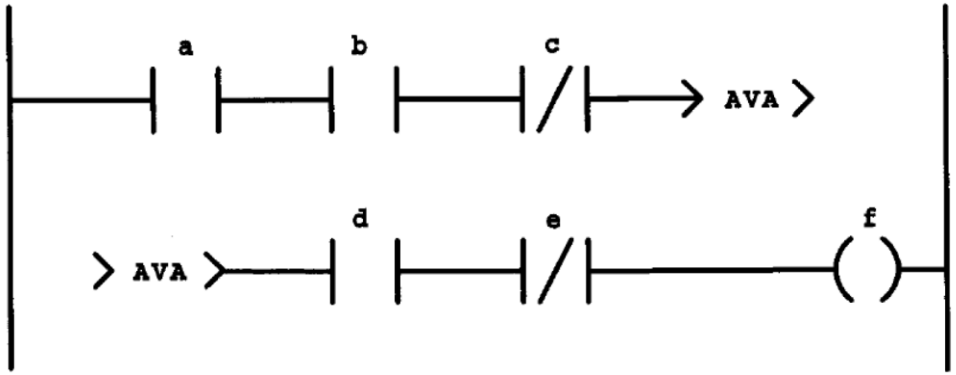
LD: istruzioni aggiuntive

- Controllo di flusso
  (per realizzare salti e istruzioni condizionali)



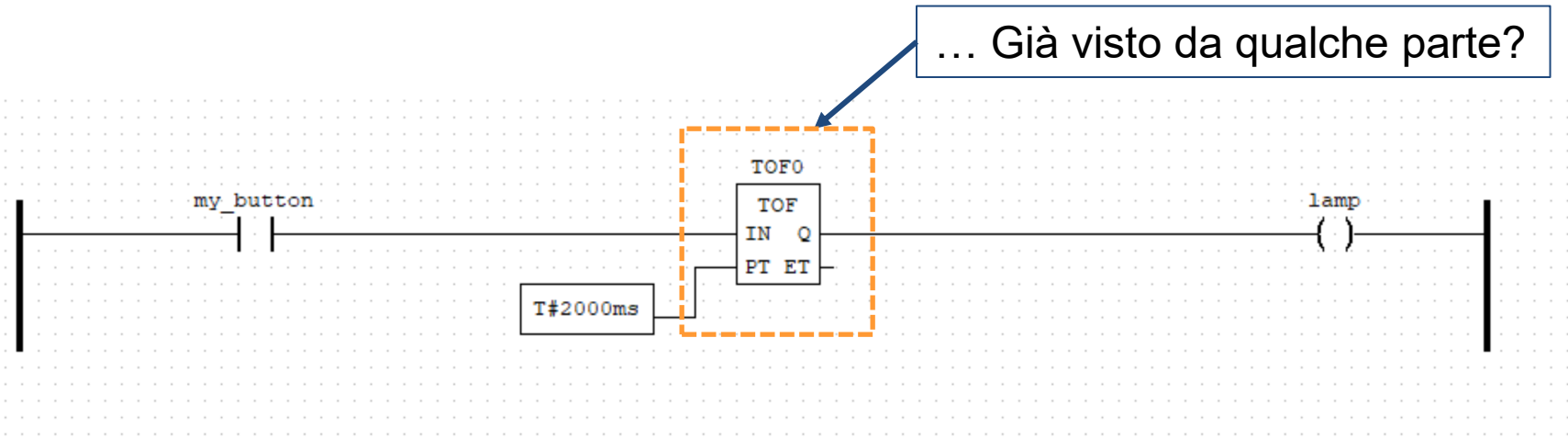
LD: istruzioni aggiuntive

- Terminazione del flusso
  - `---<RETURN>---`
- Connessione
  (per spezzare *rung* troppo lunghi)



LD: istruzioni aggiuntive

- È infine possibile chiamare blocchi funzionali

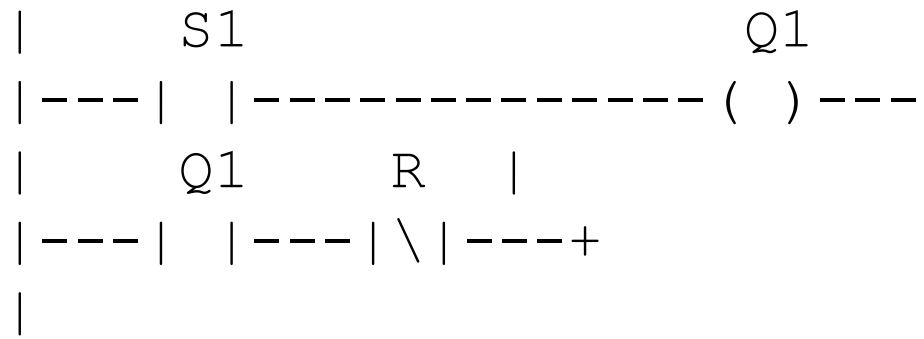


Esercizio

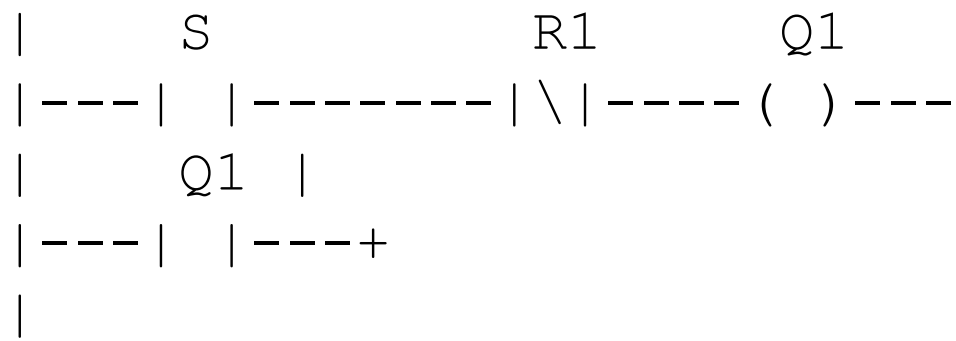
- Provare a realizzare in ladder diagram il corpo dei blocchi funzionali presentati nella lezione precedente

Esercizio

Flip-flop SR



Flip-flop RS



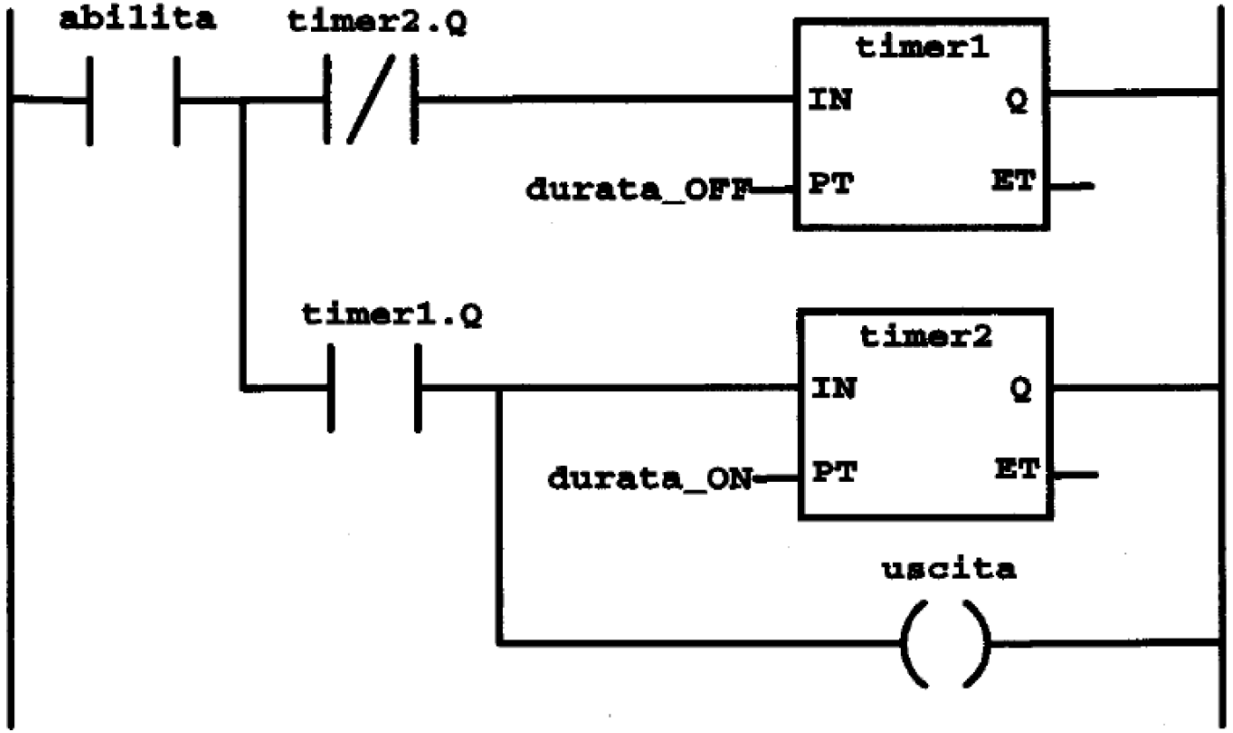
Esercizio

F_TRIG

```
FUNCTION_BLOCK F_TRIG (*fronte di discesa *)
        VAR_INPUT
                CLK : BOOL;
        END_VAR
        VAR_OUTPUT
                Q : BOOL;
        END_VAR
        VAR RETAIN
                AUX : BOOL := 1;
        END_VAR
        |                (*corpo in linguaggio a contatti *)
        |   CLK        AUX     Q
        |---|\|---+---|\|---( )---
        |   |          AUX
        |   +---------( )---
        |
END_FUNCTION_BLOCK
```

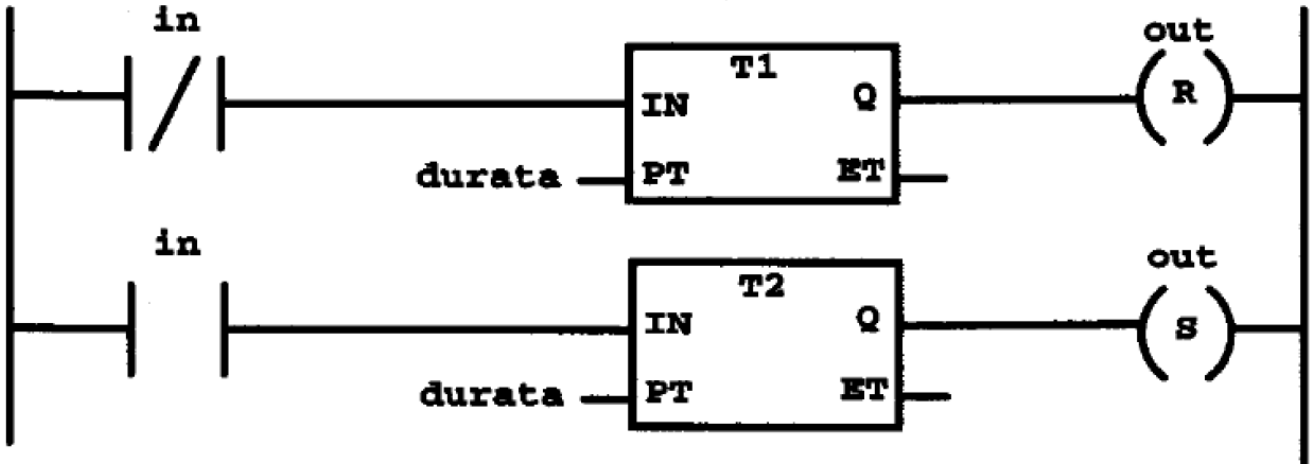
Esercizio

- Generatore onda quadra



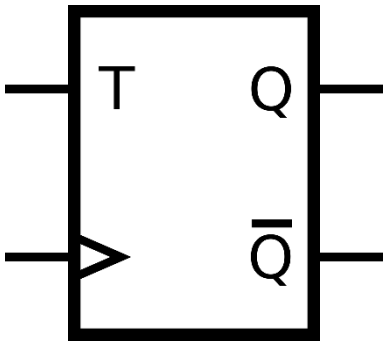
Esercizio

- Antirimbalzo

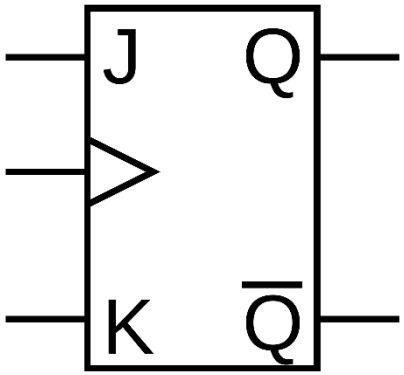


Esercizi

- Provare a realizzare (in OpenPLC):
  - Funzioni logiche: OR, AND, XOR…
  - Riconoscimento di un fronte di salita o discesa
  - Flip-flop (Toggle, e JK)
    → realizzare anche i diagrammi orari



T	Q	Qn
1	Q	Qn
0	Qn	Q



J	K	Q	Qn
0	0	Q	Qn
0	1	0	1
1	0	1	0
1	1	Qn	Q

Risorse e Riferimenti

- [1] Cap. 6
- [3] Cap. 2

- What is ladder logic @ RealPars
- OpenPLC introduction to ladder logic

Fine Lezione #9

Ladder Diagram