



Corso di Automazione industriale

Lezione 8

PLC – Testo strutturato

Esercizi

Esercizio 1

Si consideri un sistema di trasporto di pietre per mezzo di un carrello.

L'operatore determina l'inizio del ciclo tramite un apposito pulsante START. Il carrello percorre per intero il binario da sinistra a destra e si arresta in attesa di essere caricato. Le pietre, dopo essersi accumulate in un serbatoio, vengono meccanicamente trasferite nel carrello, il quale deve automaticamente muoversi lungo il binario da destra a sinistra.

Esercizio 1

Come **INPUT** vi sono a disposizione sei sensori:

S START

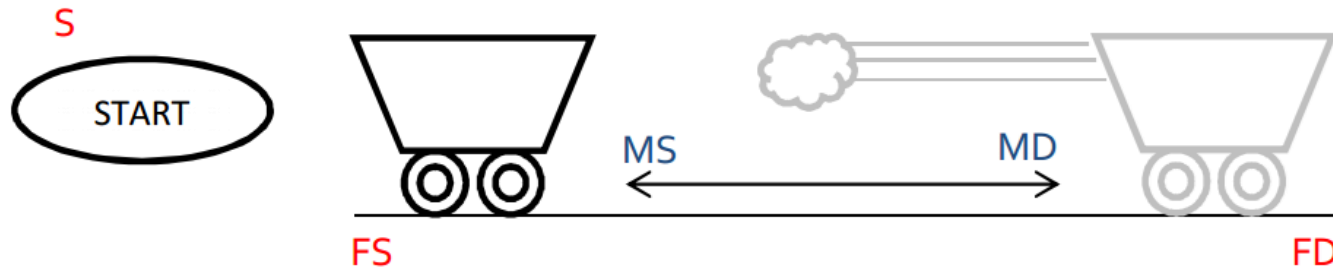
FS Fondo binario sx

FD Fondo binario dx

SV Serbatoio vuoto

FSG Fondo serbatoio giù

FSS Fondo serbatoio su



Come **OUTPUT** troviamo:

MD Motore carrello verso dx

MS Motore carrello verso sx

SG Motore serbatoio in giù

SS Motore serbatoio in su

Esercizio 1

La scelta dell'implementazione, in testo strutturato è molto più libera rispetto a ladder e SFC.

Essendo un linguaggio più di alto livello, da la possibilità di organizzare, in base alle esigenze del programmatore, il software.

N.B.: è un'arma a doppio taglio: è possibile fare confusione e rendere tutto incomprensibile!

Esercizio 1

Soluzione «ladder-based»

```
PROGRAM _INIT
```

```
MS := 0;  
MD := 0;  
SG := 0;  
SS := 0;
```

```
END_PROGRAM
```

```
PROGRAM _CYCLIC
```

```
IF Start AND FS AND NOT SV THEN
```

```
MD := 1;
```

```
END_IF;
```

```
IF FD THEN
```

```
MD := 0;
```

```
END_IF;
```

```
IF FD AND NOT FS AND FSS AND NOT FSG AND NOT SV THEN
```

```
SG := 1;
```

```
END_IF;
```

```
IF FD AND NOT FS AND FSG AND NOT FSS THEN
```

```
SG := 0;
```

```
END_IF;
```

```
IF FD AND NOT FS AND NOT FS AND FSG AND NOT FSS AND SV  
THEN
```

```
SS := 1;
```

```
END_IF;
```

```
IF FD AND NOT FS AND FSS AND NOT FSG THEN
```

```
SS := 0;
```

```
END_IF;
```

```
IF FD AND NOT FS AND FSS AND NOT FSG AND SV THEN
```

```
MS := 1;
```

```
END_IF;
```

```
IF FS AND NOT FD THEN
```

```
MS := 0;
```

```
END_IF;
```

```
END_PROGRAM
```

Esercizio 1

Soluzione «SFC-based»

```
PROGRAM_INIT
```

```
MS := 0;
```

```
MD := 0;
```

```
SG := 0;
```

```
SS := 0;
```

```
END_PROGRAM
```

```
PROGRAM_CYCLIC
```

```
CASE State OF
```

```
0:
```

```
MS := 0;
```

```
MD := 0;
```

```
SG := 0;
```

```
SS := 0;
```

```
IF Start AND NOT SV THEN
```

```
State := 1;
```

```
END_IF;
```

```
1:
```

```
MD := 1;
```

```
IF FD THEN
```

```
MD := 0;
```

```
State := 2;
```

```
END_IF;
```

```
2:
```

```
SG := 1;
```

```
IF FSG THEN
```

```
SG := 0;
```

```
State := 3;
```

```
END_IF;
```

```
3:
```

```
IF SV THEN
```

```
SS := 1;
```

```
State := 4;
```

```
END_IF;
```

```
4:
```

```
IF FSS THEN
```

```
SS := 0;
```

```
State := 5;
```

```
END_IF;
```

```
5:
```

```
MS := 1;
```

```
IF FS THEN
```

```
MS := 0;
```

```
State := 0;
```

```
END_IF;
```

```
END_CASE
```

```
END_PROGRAM
```

Esercizio 1

Quale soluzione è meglio?

Dipende dal tipo di macchina da controllare: come vedremo in seguito, nel caso del controllo dell'autolavaggio la soluzione «ladder-based» è la più semplice e intuitiva.

Nel caso, invece, del controllo per la stazione di lavorazione, la soluzione migliore è quella basata su un automa a stati finiti.

Esercizio 1.1

Aggiungiamo all'esercizio precedente una fermata per manutenzione ogni 100 cicli.

Serve aggiungere:

FM (Fermo Manutenzione) come output

RM (Reset Manutenzione) come input

N.B.: Servirà anche creare una variabile contatore!

Analizziamo solo la soluzione con macchina a stati

Esercizio 1.1

PROGRAM _INIT

```
MS := 0;  
MD := 0;  
SG := 0;  
SS := 0;  
n := 0;
```

END_PROGRAM

PROGRAM _CYCLIC

CASE State OF

0:

```
MS := 0;  
MD := 0;  
SG := 0;  
SS := 0;  
IF Start AND NOT SV THEN  
  State := 1;  
END_IF;
```

1:

```
MD := 1;
```

IF FD THEN

```
MD := 0;  
State := 2;
```

END_IF;

2:

```
SG := 1;
```

IF FSG THEN

```
SG := 0;  
State := 3;
```

END_IF;

3:

IF SV THEN

```
SS := 1;  
State := 4;
```

END_IF;

4:

IF FSS THEN

```
SS := 0;  
State := 5;
```

END_IF;

5:

```
MS := 1;
```

IF FS THEN

```
MS := 0;  
n := n + 1;
```

IF n >= 100 THEN

```
FM := 1;  
State := 6;
```

ELSE

```
State := 0;
```

END_IF;

END_IF;

6:

IF RM THEN

```
FM := 0;  
n := 0;  
State := 0;
```

END_IF;

END_CASE

END_PROGRAM

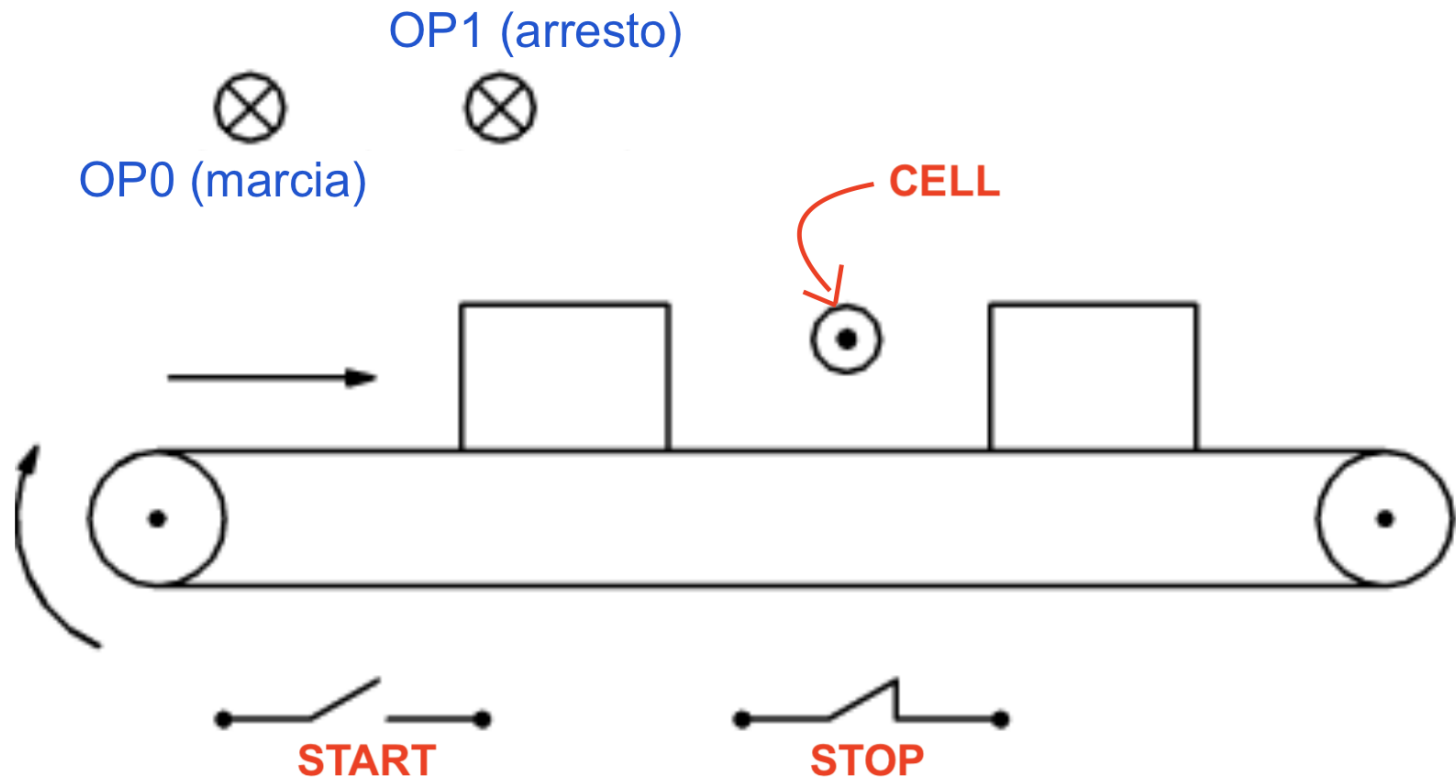
Esercizio 2

Si consideri un sistema conta-pezzi con nastro trasportatore. Il nastro viene azionato da un motore, comandato da due pulsanti START e STOP. Lo stato di marcia e/o di arresto deve essere mostrato tramite due lampade distinte.

Ciascun pezzo viene posizionato all'inizio del nastro e, ad ogni passaggio di un pezzo davanti alla fotocellula, deve essere gestito il conteggio. Il sistema si deve fermare in automatico ogni 50 pezzi.

Il nastro può essere fermato manualmente in qualunque momento premendo il tasto STOP. Il riavvio può avvenire tramite il tasto START, ma il conteggio deve riprendere da dove si era fermato.

Esercizio 2



Esercizio 2

- All'avvio, il motore che aziona il nastro deve essere fermo, quindi dovrà essere accesa solamente *OP1*.
- Premendo il tasto *START*, il motore si avvia: si deve accendere *OP0* e si deve spegnere *OP1*.
- Ogni volta che il motore è in funzione e si rileva un pezzo davanti alla fotocellula *CELL*, deve essere incrementato il conteggio.
- Quando il conteggio raggiunge il valore 50, il nastro deve essere fermato: si deve accendere *OP1* e si deve spegnere *OP0* (*ad un successivo riavvio il conteggio riparte da 0*).
- Fermando il nastro prima del valore limite il conteggio deve riprendere dal valore a cui era arrivato.

Esercizio 2

PROGRAM _INIT

OP1 := 1;

OP0 := 0;

State := 0;

N := 0;

END_PROGRAM

IF N >= 50 THEN

N := 0;

State := 0;

END_IF;

END_CASE;

END_PROGRAM

PROGRAM _CYCLIC

CASE State OF

0: (* Stop *)

OP1 := 1;

OP0 := 0;

IF START AND NOT STOP THEN

OP0 := 1;

OP1 := 0;

State := 1;

END_IF;

1: (* Movimento *)

IF EDGEPOS(CELL) THEN

N := N+1;

END_IF;

IF STOP THEN

State := 0;

END_IF;

Esercizio 3

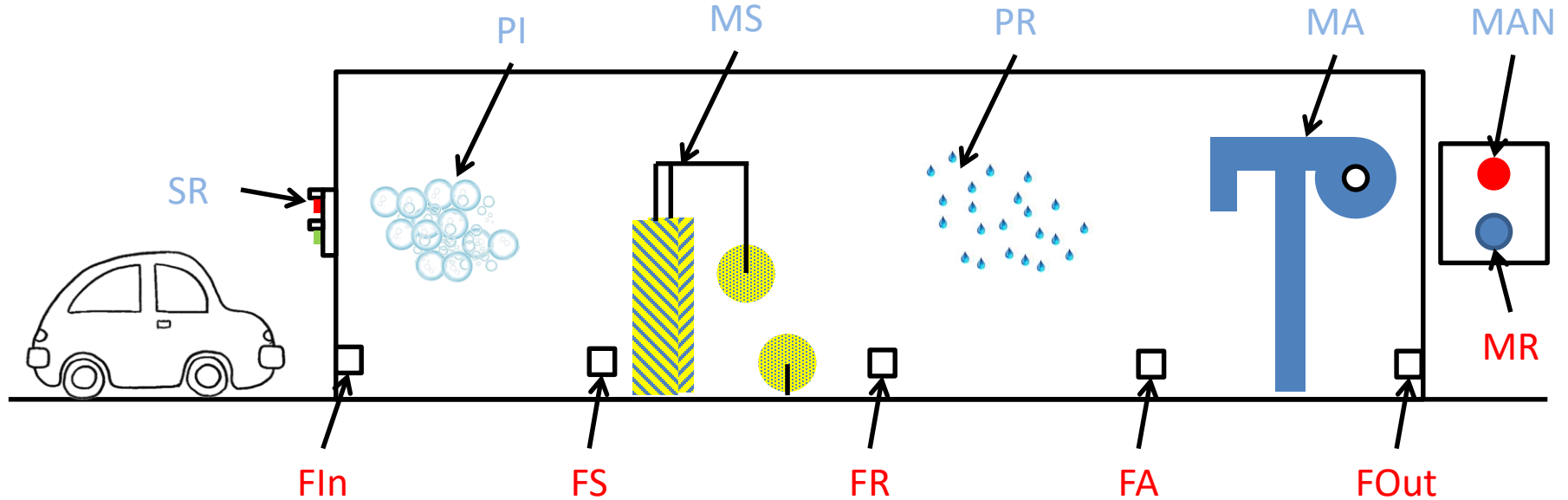
Si consideri un sistema di lavaggio automatico per autovetture.

Il cliente si avvicina al nastro movimentatore quando il semaforo è verde. Le fasi di lavaggio sono: insaponatura, spazzolatura, risciacquo e asciugatura.

Tutte le fasi sono precedute da una fotocellula che rileva l'arrivo della vettura in quella sezione dell'impianto.

Ogni 1000 lavaggi l'autolavaggio deve bloccarsi in attesa di manutenzione effettuata da un operatore.

Esercizio 3



Input

Output

Fin Fotocellula in ingresso

SR Semaforo stop (0=VERDE, 1=ROSSO)

FS Fotocellula spazzolatura

PI Pompa insaponatura

FR Fotocellula risciacquo

MS Motore spazzolatura

FA Fotocellula asciugatura

PR Pompa risciacquo

FOut Fotocellula uscita

MA Motore asciugatura

MR Manutenzione reset (anche out)

MAN Stop per manutenzione

Esercizio 3

L'impianto può essere visto come un insieme di singoli impianti:

- Insaponatura
- Spazzolatura
- Risciacquo
- Asciugatura

Ognuno di questi «impianti» deve avviarsi quando la fotocellula davanti ad essi si attiva e spegnersi quando la fotocellula dopo essi si disattiva.

Esercizio 3

Come mostrato nella lezione 4, la soluzione a questo esercizio prevede una gestione «distribuita» di ogni singola parte dell'impianto.

Abbiamo visto nella lezione 7 come sia possibile, in modo molto veloce, creare Function Block in ST.

Siccome abbiamo 3 sezioni che si comportano nel medesimo modo, andiamo, innanzitutto a creare il Function Block che controlla una sezione dell'impianto.

Esercizio 3

```
FUNCTION_BLOCK PlantSection
  IF NOT Activation THEN
    IF EDGEPOS(Fingresso) THEN
      Activation := 1;
    END_IF;
  ELSE
    IF EDGENEG(Fuscita) THEN
      Activation := 0;
    END_IF;
  END_IF;
END_FUNCTION_BLOCK
```

Ogni sezione si attiva sul fronte di salita della fotocellula in ingresso e viene disattivata sul fronte di discesa della fotocellula in uscita.

Ingressi:
Fingresso, Fuscita

Uscite:
Activation

Esercizio 3

```
FUNCTION_BLOCK FirstSection
  IF n<1000 THEN
    IF NOT SemaforoStop THEN
      IF EDGEPOS(Fingresso) THEN
        SemaforoStop := 1;
        Activation := 1;
      END_IF;
    ELSE
      IF EDGENEG(Fuscita) THEN
        SemaforoStop := 0;
        Activation := 0;
        n := n + 1;
      END_IF;
    END_IF;
  ELSE
    SemaforoStop := 1;
    Activation := 0;
    ManutenzioneRichiesta := 1;
    IF ManReset THEN
      n := 0;
      SemaforoStop := 0;
      ManutenzioneRichiesta := 0;
    END_IF;
  END_IF;
END_FUNCTION_BLOCK
```

La prima sezione (Insaponatura) dovrà anche gestire il semaforo e la manutenzione programmata. Per questo motivo è necessario utilizzare un diverso Function Block.

Ingressi:

Fingresso, Fuscita, ManReset

Uscite:

SemaforoStop, ManutenzioneRichiesta, Activation

Esercizio 3

```
PROGRAM_INIT  
END_PROGRAM
```

```
PROGRAM_CYCLIC
```

```
  Insaponatura(Fingresso := FIn, Fuscita := FS, ManReset := MR);  
  MAN := Insaponatura.ManutenzioneRichiesta;  
  SS := Insaponatura.SemaforoStop;  
  PI := Insaponatura.Activation;  
  Spazzolatura(Fingresso := FS, Fuscita := FR);  
  MS := Spazzolatura.Activation;  
  Risciacquo(Fingresso := FR, Fuscita := FA);  
  PR := Risciacquo.Activation;  
  Asciugatura(Fingresso := FA, Fuscita := FOut);  
  MA := Asciugatura.Activation;  
END_PROGRAM
```

Il file del programma è molto semplice: si effettuano richiami in successioni alle sezioni dell'impianto, passando gli ingressi relativi.

N.B.: Perché non abbiamo usato una sola entità del Function Block Plant Section?

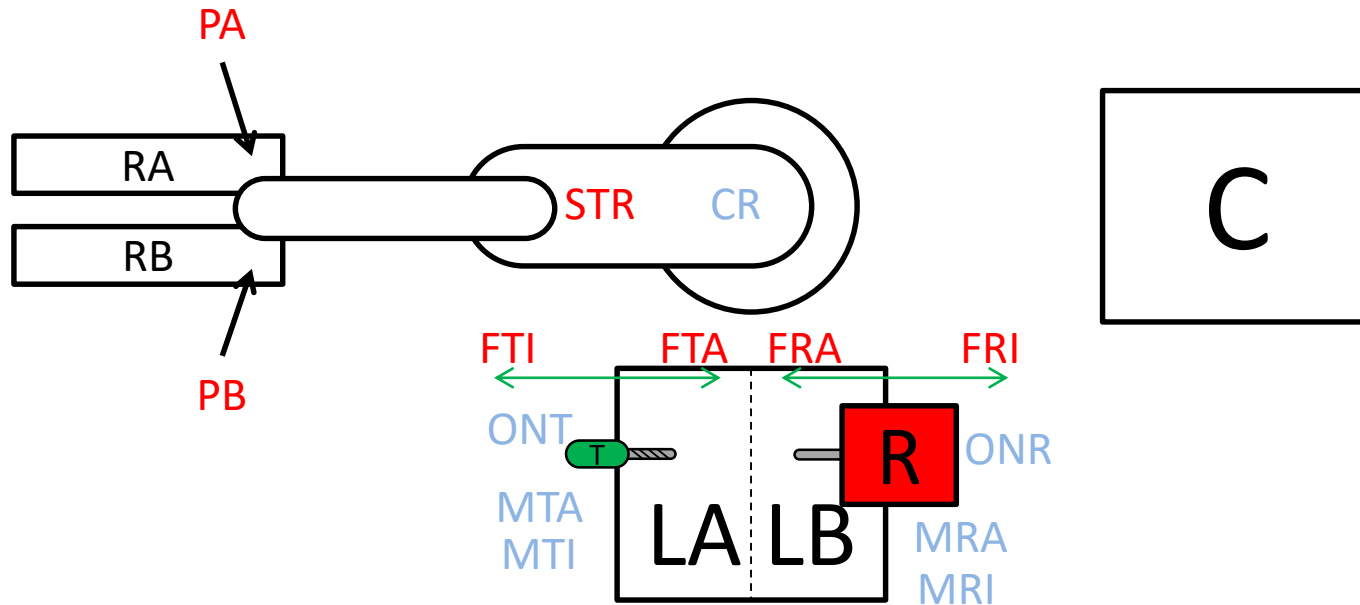
Esercizio 4

Si consideri un sistema di foratura e rivettatura automatica per lamiera.

All'arrivo dei due pezzi un robot effettua il prelievo dei componenti (in successione uno all'altro) e li posiziona nella maschera di montaggio. Al termine del posizionamento avviene la foratura per mezzo di trapano automatico (5 sec) e la rivettatura (10 sec).

Al termine della lavorazione il robot sposta il pezzo unito in un contenitore.

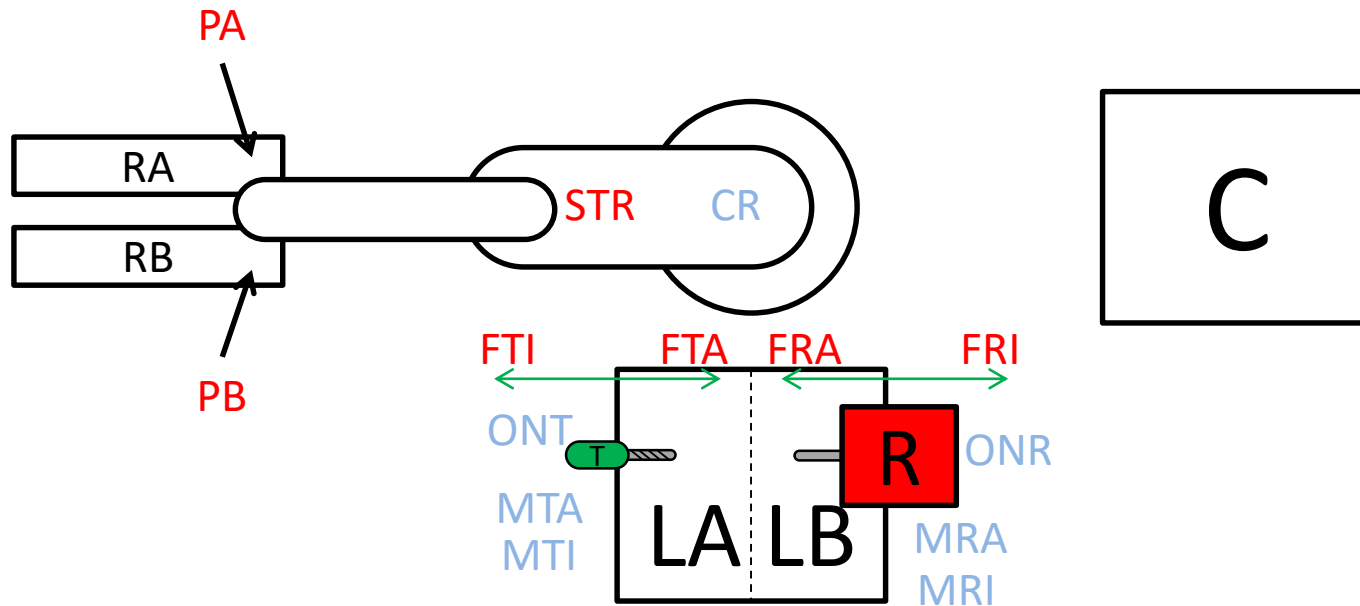
Esercizio 4



Input

- | | | | |
|------------|------------------------------------|------------|---------------------------------|
| PA | Sensore presenza nastro A | FTA | Finecorsa trapano avanti |
| PB | Sensore presenza nastro B | FRI | Finecorsa rivettatrice indietro |
| STR | Stato robot (0=FINE, 1=ESECUZIONE) | FRA | Finecorsa rivettatrice avanti |
| FTI | Finecorsa trapano indietro | | |

Esercizio 4



Output

- | | | | |
|------------|---|------------|---------------------------------|
| CR | Comando robot (0=STOP, 1=prelievo RA deposito LA, 2=prelievo RB deposito LB, 3=prelievo L deposito C) | ONT | On trapano |
| MTA | Movimento avanti trapano | MRA | Movimento avanti rivettatrice |
| MTI | Movimento indietro trapano | MRI | Movimento indietro rivettatrice |
| | | ONR | On rivettatrice |

Esercizio 4

Ragioniamo sui passi da seguire per realizzare un prodotto finito:

- 1)Attendere che PA e PB vadano a 1
- 2)Comando 1 al robot e attendere la fine dell'esecuzione
- 3)Comando 2 al robot e attendere la fine dell'esecuzione
- 4)Muovere il trapano avanti fino a raggiungere il suo finecorsa avanti
- 5)Azionare il trapano per 5 secondi
- 6)Muovere il trapano indietro fino a raggiungere il suo finecorsa indietro
- 7)Muovere la rivettatrice avanti fino a raggiungere il suo finecorsa avanti
- 8)Azionare la rivettatrice per 10 secondi
- 9)Muovere la rivettatrice indietro fino a raggiungere il suo finecorsa indietro
- 10)Comando 3 al robot e attendere la fine dell'esecuzione
- 11)Comando 0 al robot

Esercizio 4

```
PROGRAM_INIT
  State := 0;
END_PROGRAM

PROGRAM_CYCLIC
CASE State OF
  0: (* Stop *)
    IF PA AND PB THEN
      CR := 1;
      State := 1;
      STR := 1;
    END_IF;
  1: (* Prelievo A *)
    IF NOT STR THEN
      CR := 2;
      State := 2;
      STR := 1;
    END_IF;
  2: (* Prelievo B *)
    IF NOT STR THEN
      State := 3;
    END_IF;
  3: (* Avanti trapano *)
    MTA := 1;
    IF FTA THEN
      MTA := 0;
      State := 4;
    END_IF;
  4: (* Lavorazione trapano *)
    ONT := 1;
    t := t + dt;
    IF t >= T#5s THEN
      t := T#0s;
      ONT := 0;
      State := 5;
    END_IF;
  5: (* Indietro trapano *)
    MTI := 1;
    IF FTI THEN
      MTI := 0;
      State := 6;
    END_IF;
  6: (* Avanti rivettatrice *)
    MRA := 1;
    IF FRA THEN
      MRA := 0;
      State := 7;
    END_IF;
  7: (* Lavorazione rivettatrice *)
    ONR := 1;
    t := t + dt;
    IF t >= T#10s THEN
      t := T#0s;
      ONR := 0;
      State := 8;
    END_IF;
  8: (* Indietro rivettatrice *)
    MRI := 1;
    IF FRI THEN
      MRI := 0;
      CR := 3;
      State := 9;
      STR := 1;
    END_IF;
  9: (* Deposito pezzo *)
    IF NOT STR THEN
      State := 0;
    END_IF;
END_CASE;
END_PROGRAM
```

NB: STR viene settato a 1 per comprendere meglio l'esercizio. Dovrebbe essere settato e resettato in automatico

Conclusioni

Considerazioni Testo Strutturato

È il linguaggio di più alto livello tra quelli disponibili nella norma IEC 61131.

Mette in luce una serie di problematiche (presenti anche negli altri linguaggi) relative all'ingegnerizzazione del software.

Nel mondo industriale non è molto usato: sta entrando ora grazie ad alcuni tool che consentono la generazione automatica del codice (ne vedremo un esempio nelle lezioni successive)