Automation Robotics and System CONTROL

Università degli Studi di Modena e Reggio Emilia

Università degli Studi di Ferrara

PLC
CodeSys
Example 2

Francesca Fanfoni

francesca.fanfoni@unimore.it

Indice

Esercitazione 1

- Esempio in Ladder Diagram (LD)

Esercitazione 2

- Esempio in Structured Text (ST)
- Allocazione dei task

Un progetto

- **Progetto** si intendono tutti gli oggetti necessari per la stesura del programma per PLC
- Gli **oggetti** sono:
  - POU (Program Organization Unit),
  - tipi di dato definiti dall'utente,
  - la parte di visualizzazione,
  - risorse
  - librerie.

  POU (Program Organization Unit) può essere *Function*, *Function Block* o *Program*.
- Un progetto in CodeSys è identificato dal nodo che raccoglie tutti gli oggetti che sono necessari alla **Application** definizione di una particolare istanza del programma PLC su un determinato dispositivo hardware (PLC, controller).

The First Project

Task 1: Handle a dummy traffic signal unit.

The red/green phases of both traffic signals alternate and we have to handle the request of pedestrian crossing to stop the traffic.

Task 2: We will insert a yellow transitional phase.

Step by step procedure

1. Create a schema solution, a new project and a new program
2. Compile a project
3. Run the simulator and transfer the project to the target
4. Configure the visualization
5. Testing the program sequence

ARSControl@unimore.it

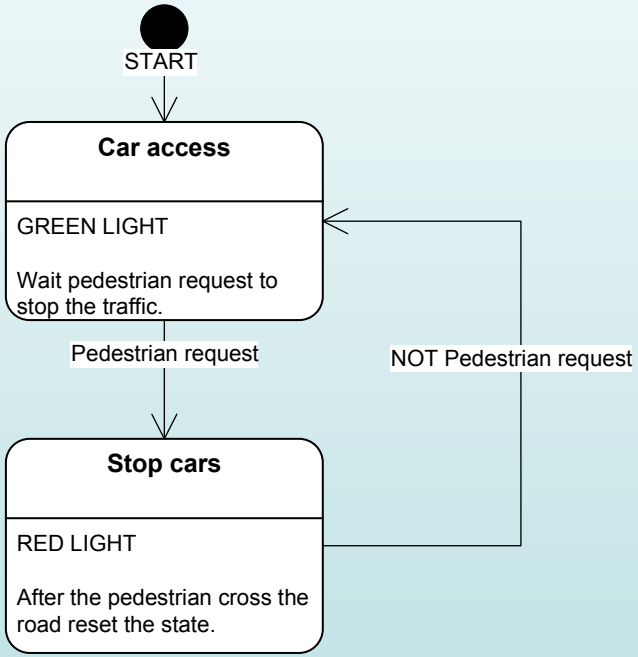
The First Project

Create a schema solution

The red/green phases of both traffic signals alternate and we have to handle the request of pedestrian crossing to stop the traffic.

We have to think and design a solution draw a simple diagram that is directly connected to the PLC program.

In the schema have to define:
- States
- Variables



ARSControl@unimore.it

Ladder diagram (LD)

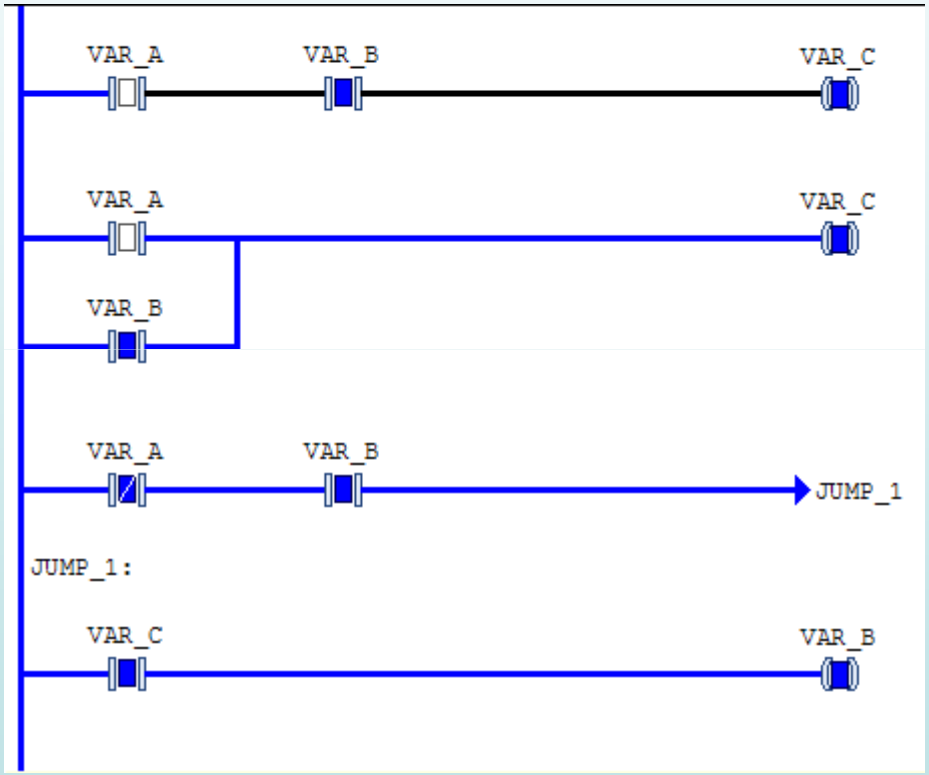
- **LOGIC**

AND serial connection of two or more contacts

OR parallel connection of two or more contacts

Negations

Jumps to skip over networks in the program, using a symbolic name of a jump label

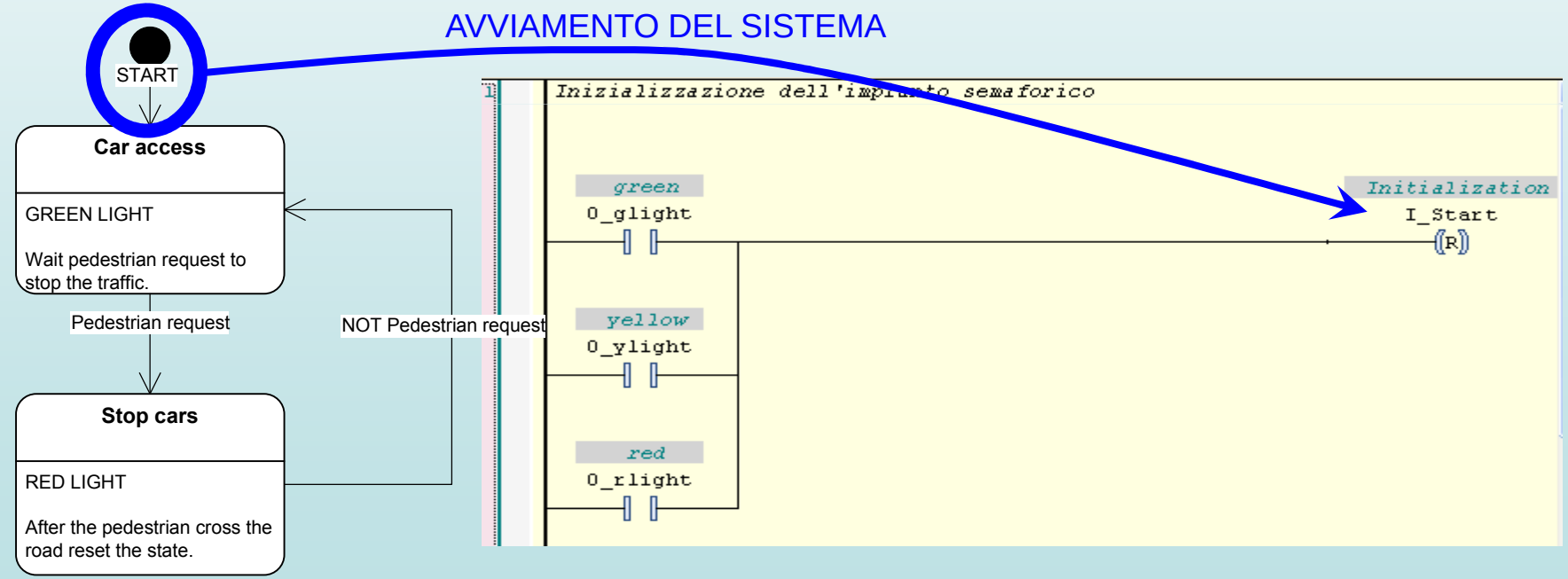


ARSControl@unimore.it

The First Project

Compiling a project and trasferring the project

- La soluzione viene implementata con 3 network
- Uno stato corrisponde ad una network.
- La prima network dell'Application gestisce l'inizializzazione dell'intero sistema.

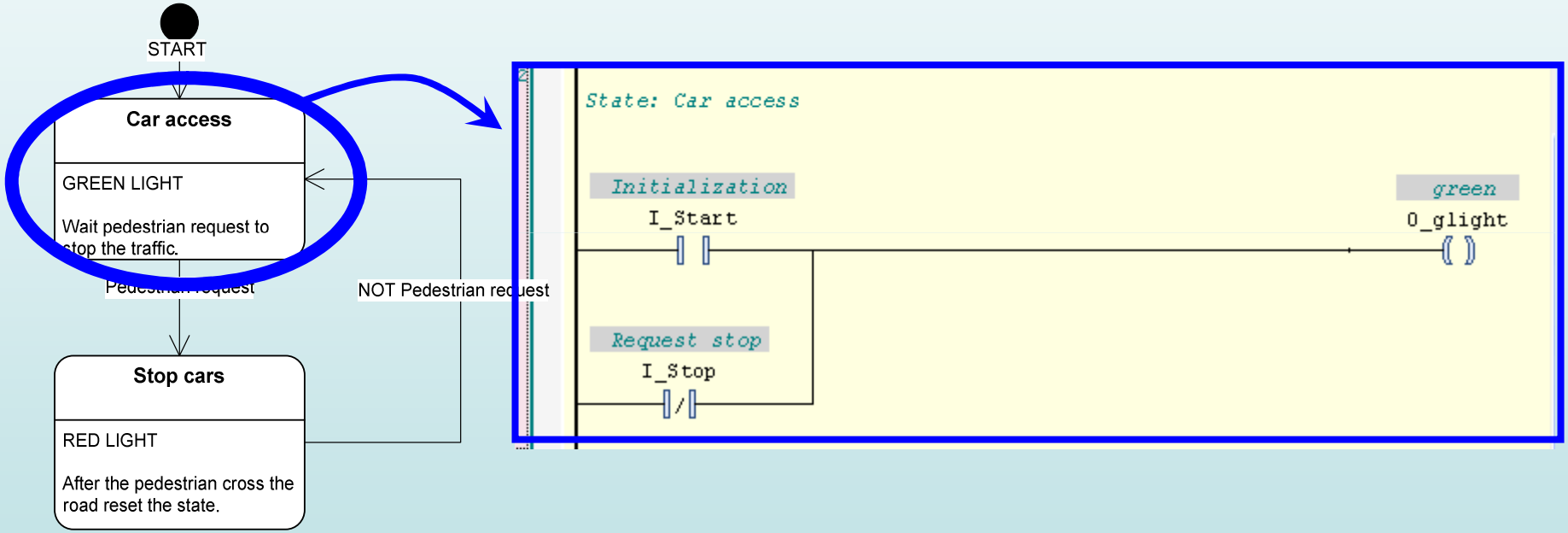


ARSControl@unimore.it

The First Project

Compiling a project and trasferring the project

- Ladder diagram program is dived in ***Network***
- Each network can be linked to a part of the solution schema

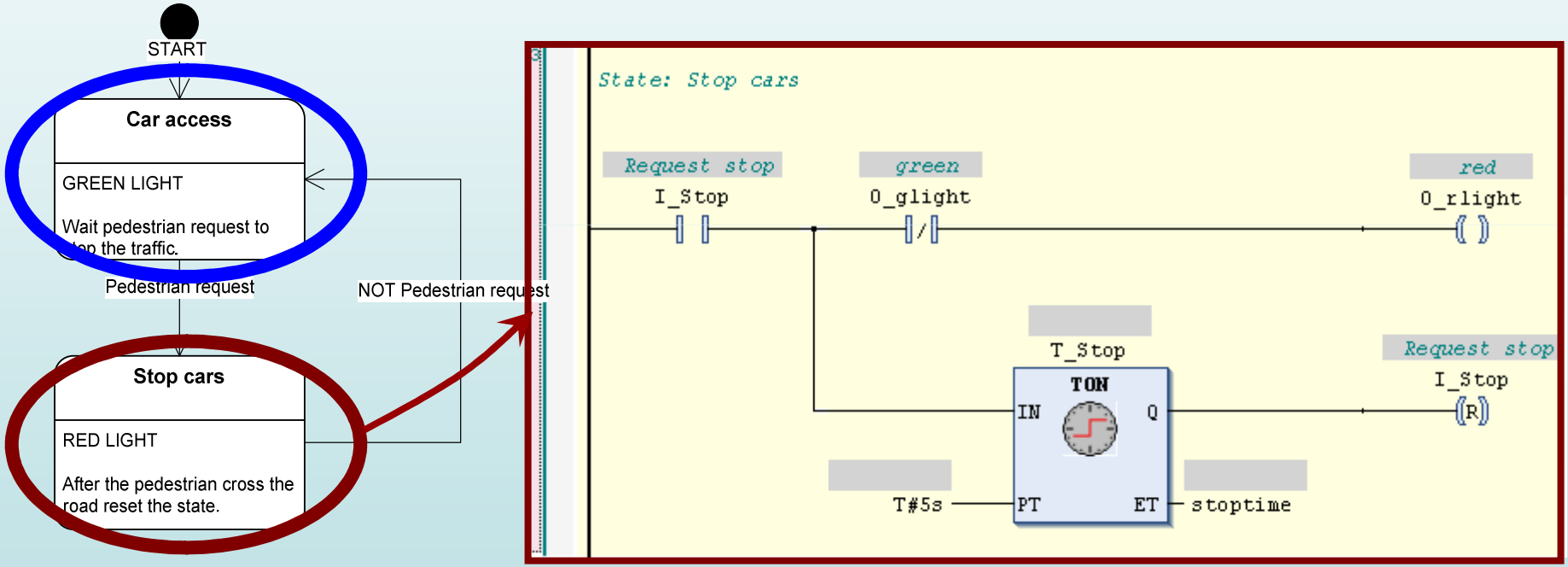


ARSControl@unimore.it

The First Project

Compiling a project and trasferring the project

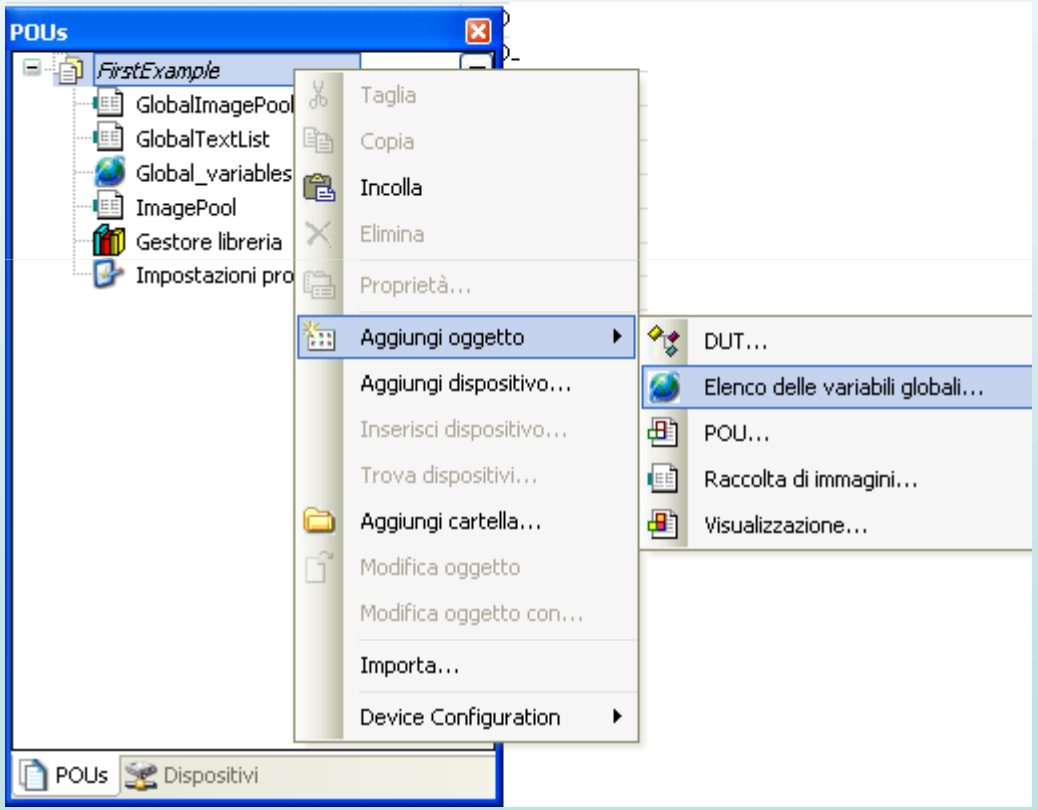
- Ladder diagram program is dived in ***Network***
- Each network can be linked to a part of the solution schema



ARSControl@unimore.it

Global variables

- Dichiarazione di un insieme di variabili globali per il progetto.



Global Variables List (GVL)

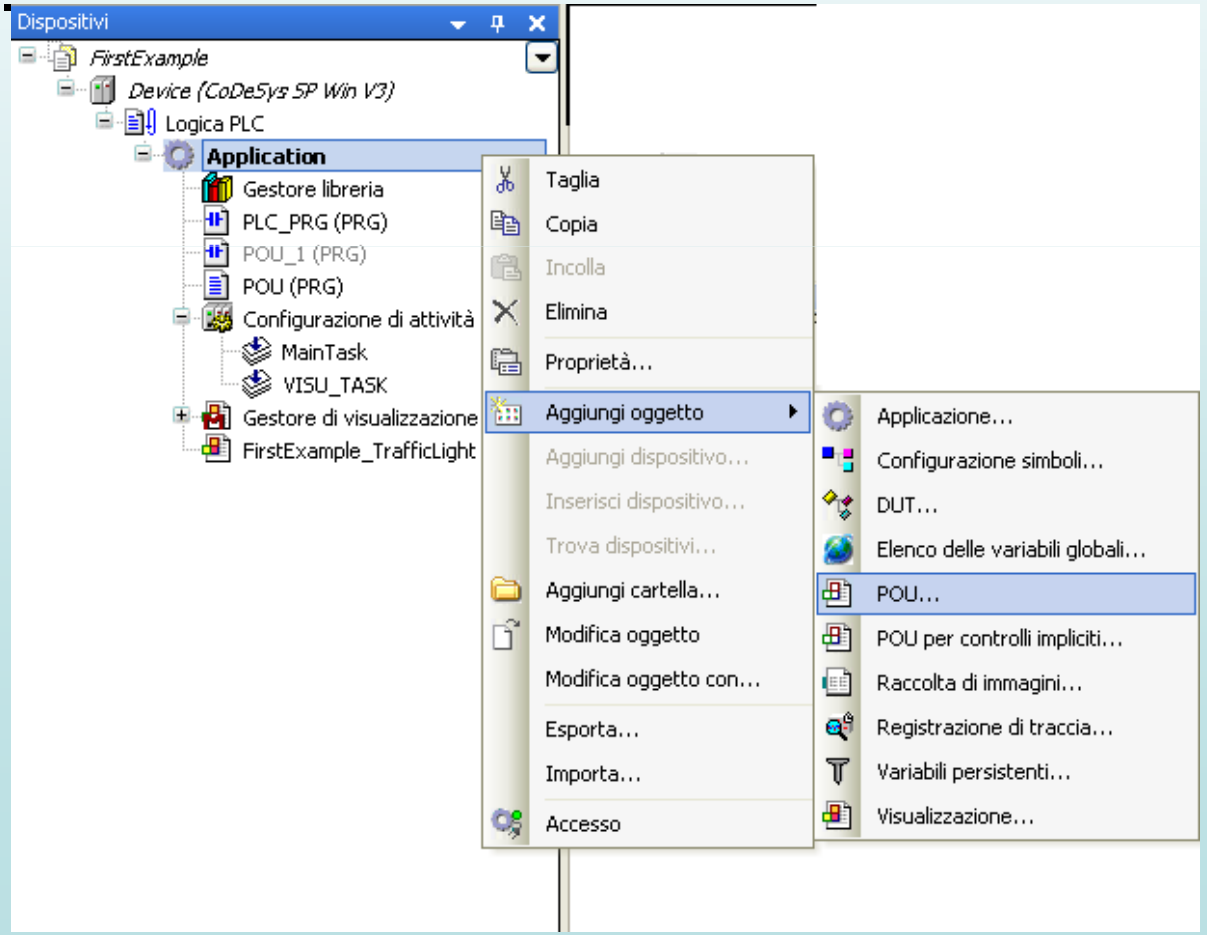
- Dichiarazione di un insieme di variabili globali per il progetto.
- Elenco delle variabili:

Global_variables_list

```
1  VAR_GLOBAL
2
3      I_Start  :BOOL := TRUE;
4      I_Stop   :BOOL := FALSE;
5
6      O_gligth    :BOOL ; // green
7      O_rligth    :BOOL;  // red
8      O_yligth    :BOOL;  // yellow
9      T_Stop: TON;        // timer
10     stoptime: TIME;     //elapsed time
11
12 END_VAR
```

Dichiarazione di una POU in ST

- Aggiungere **un programma** in testo strutturato all'applicazione.



Structured Text (ST)

- ST is a textual language similar to “C”, or (for who might remember it) PASCAL.
- ST can be successfully used to develop complex algorithm, data structure handling, etc.
- ST has the syntactical structure of the procedural programming languages:
  - *Assignment*
  - *Choices*
  - *Iteration*

Assignment

- The variable on the left side should be of the same type of the result of the expression of the right side.
- on contrary, the ST compiler will introduce a *variable casting* to set all the variables to the same type.

<variabl := <expression>;

Choices

```
IF <Boolean_Expr_1> THEN
    <code>;
ELSIF <Boolean_Expr_2> THEN
    <code>;
ELSE
 <code>;
END_IF

CASE <integer_expression> OF
<integer_value_1> : <code>;
<integer_value_2> : <code>;
...
ELSE
        <code>;
END_CASE
```

Iteration

```
REPEAT
        <code>;
UNTIL <Boolean_Expr>
END_REPEAT;

 WHILE <Boolean_Expr> DO
        <code>
 END_WHILE;

 FOR <integer_Variable>:= <initial_value> TO
                                  <final_value> BY <step> DO
        <code>
 END_FOR;
```

Warning

- The iteration structure may violate the real time concerns of the program.
- An iteration can’t be done to wait for a external variable changes.

ST Operators (a)

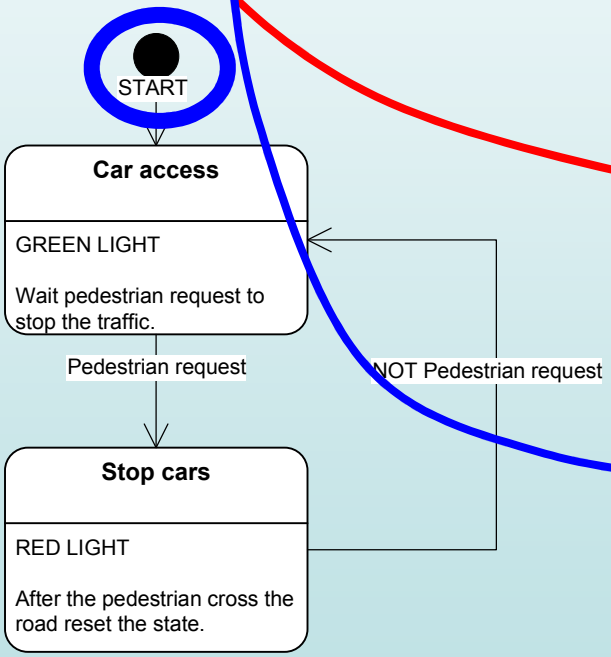
Operation[a]	Symbol	Example	Precedence
Parenthesization	(expression)	(A+B/C), (A+B)/C, A / (B+C)	10 (HIGHEST)
Negation	-	-A, - A	8
Unary Plus	+	+B, + B	8
Complement	NOT	NOT C	8
Exponentiation[b]	**	A**B, B ** B	7
Multiply	*	A*B, A * B	6
Divide	/	A/B, A / B / D	6
Modulo	MOD	A MOD B	6

ST Operators (b)

Operation[a]	Symbol	Example	Precedence
Add	+	A+B, A + B + C	5
Subtract	-	A-B, A - B - C	5
Comparison	< , > , <= , >=	A<B	4
Equality	=	A=B, A=B & B=C	4
Inequality	<>	A<>B, A <> B	4
Boolean AND	&	A&B, A & B, A & B & C	3
Boolean AND	AND	A AND B	3
Boolean Exclusive OR	XOR	A XOR B	2
Boolean OR	OR	A OR B	1 (LOWEST)

POU in testo strutturato

- **Variabili del programma per gestire un diagramma a stati.**
- **Inizializzazione del primo stato**

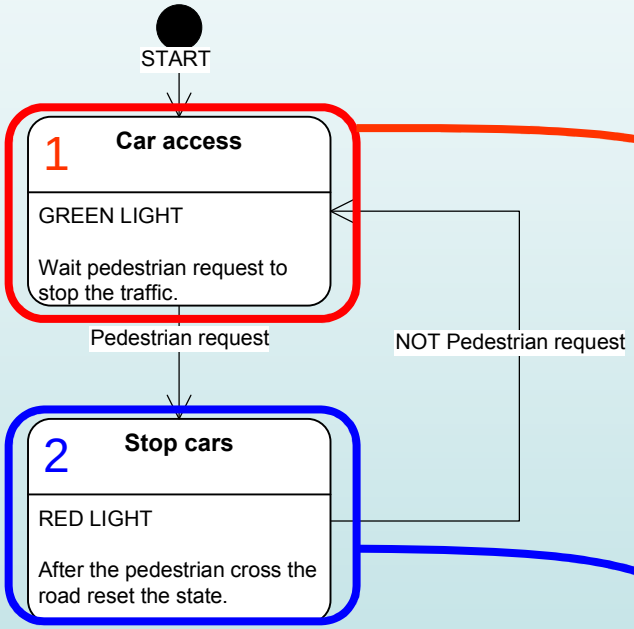



```
PROGRAM POU

VAR
    CurrentState :INT; //the actual state
    NextState    :INT; // the next state

END_VAR
```

```
//Init first state
IF I_Start THEN
    CurrentState := 1;
    NextState := 1;
    I_Start := FALSE;
END_IF
```

POU in testo strutturato

- Gli stati del sistema vengono gestiti con una struttura CASE … OF




```
1  //Init first state
2  IF I_Start THEN
3      CurrentState := 1;
4      NextState := 1;
5      I_Start := FALSE;
6  END_IF
7
8  CASE CurrentState OF
9      1: IF NOT (I_Stop) THEN
10         O_glight := TRUE;
11         O_rlight := FALSE;
12         END_IF
13         // Test the transition:
14         IF (I_Stop) THEN // request red light
15             NextState := 2;
16         END_IF
17
18     2:  O_glight := FALSE;
19         O_rlight := TRUE;
20         T_Stop.IN := I_Stop;     //TIMER input
21         T_Stop.PT := T#10S;      // TIMER period
22         stoptime := T_Stop.ET; // elapsed time
23         // Test the transition:
24         IF (T_Stop.Q)  THEN  //TIMER output
25             NextState := 1;
26             //Reset the request to stop
27             I_Stop := FALSE;
28         END_IF
29 END_CASE
30
31 CurrentState := NextState;
32 T_Stop();  (* call the timer funtion (you can view the function properties
33               in the HELP documentation of CodeSys) *)
```

- **Aggiornamento dello stato →**

Allocazione dei TASK

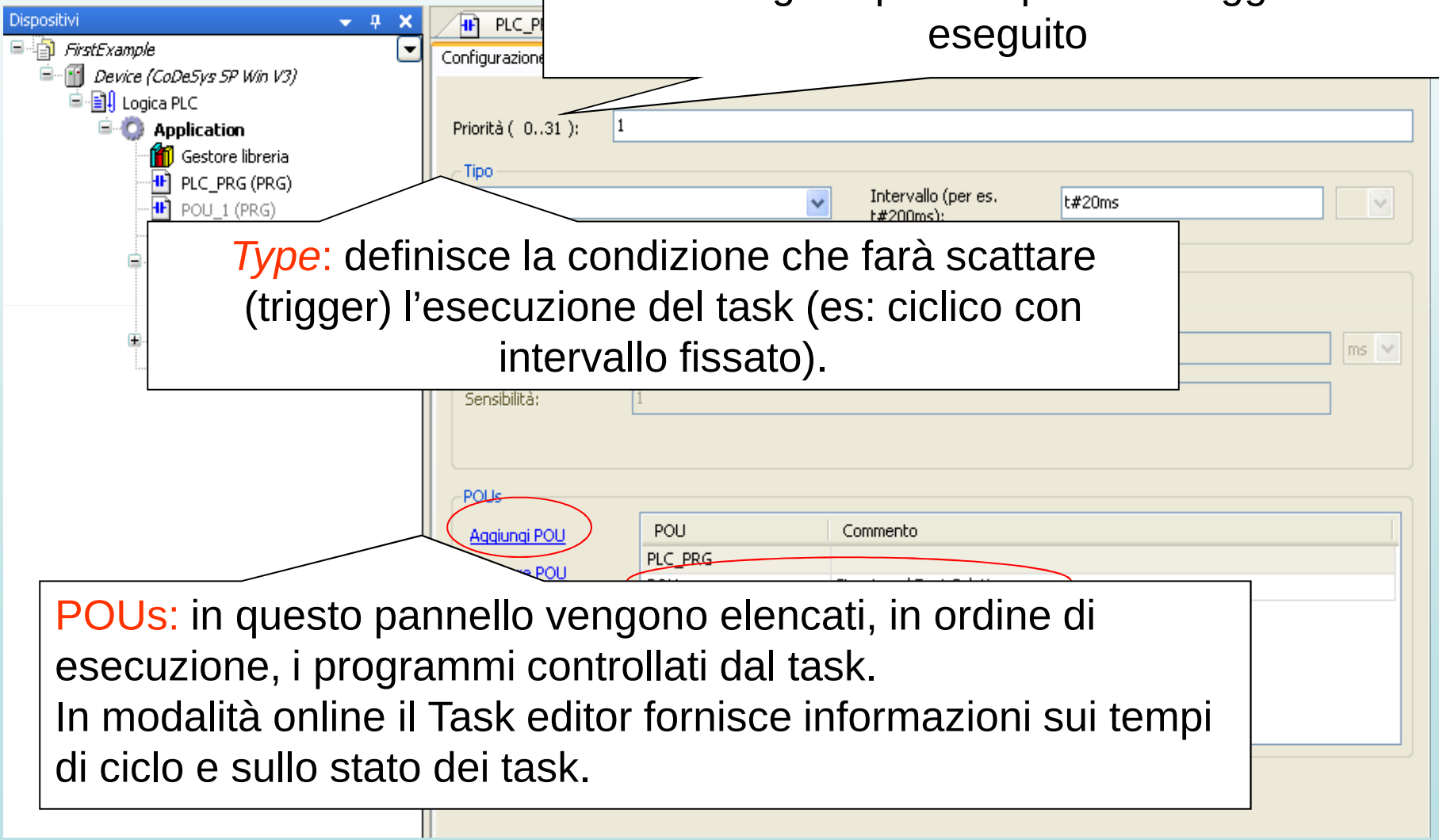
- Il progetto principale è PLC_PRG e viene sempre eseguito.
- POU sono eseguite se sono allocate a un task periodico
- Allocazione della nostra POU al task

- Per definire un Task è necessario aver inserito, sotto il nodo application, l'oggetto Task configuration (con il comando "Add object"> "Configurazione attività"). Selezionando quest'ultimo ed eseguendo nuovamente il comando "Add object" sarà possibile inserire il Task.

Allocazione dei TASK

- Allocazione della nos

Priority: se più task soddisfano i criteri per essere eseguiti quello a priorità maggiore viene eseguito

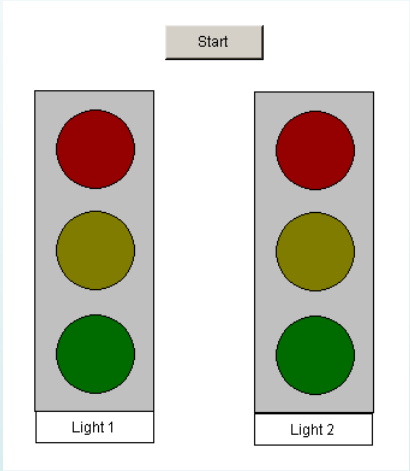


Type: definisce la condizione che farà scattare (trigger) l'esecuzione del task (es: ciclico con intervallo fissato).

POUs: in questo pannello vengono elencati, in ordine di esecuzione, i programmi controllati dal task.
In modalità online il Task editor fornisce informazioni sui tempi di ciclo e sullo stato dei task.

Esercizio 2

Esercizio 2: Controllo semaforo



Descrizione dell‘ applicazione:

Funzione: premendo il pulsante start inizia il ciclo semaforico.

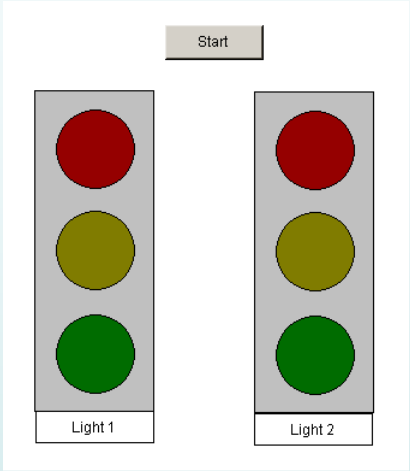
Azione: All’attivazione del pulsante il sistema esce dallo stato “standby” ed esegue il ciclo semaforico. Il ciclo prevede 4 stati, a ciascuno dei quali corrispondono opportune uscite per i due semafori: “verde”, “verde-giallo”, “giallo-rosso”, “rosso”.

Il passaggio da uno stato all’altro è determinato da un timer

Dopo sette cicli il sistema torna nello stato di standby, nel quale tutte le luci sono spente

Esercizio 2

Esercizio 2: Controllo semaforo



Descrizione dell‘ applicazione:

Funzione: premendo il pulsante start inizia il ciclo semaforico.

Azione: All’attivazione del pulsante il sistema esce dallo stato “standby” ed esegue il ciclo semaforico. Il ciclo prevede 4 stati, a ciascuno dei quali corrispondono opportune uscite per i due semafori: “verde”, “verde-giallo”, “giallo-rosso”, “rosso”.

Il passaggio da uno stato all’altro è determinato da un timer

Dopo sette cicli il sistema torna nello stato di standby, nel quale tutte le luci sono spente

PLC
CodeSys
Example 2

Francesca Fanfoni

francesca.fanfoni@unimore.it