

Linguaggio di Programmazione Ladder

- ❖ E' il più vecchio linguaggio di programmazione per PLC
- ❖ Si basa su simboli di provenienza "elettrica":
 - binari di potenza (power rail), contatti elettrici e avvolgimenti magnetici (coil)
- ❖ Si articola in linee orizzontali dette “rung”
- ❖ Ciascun “rung” può contenere contatti, coil, Function Block e Funzioni
- ❖ Ciascun “rung” deve essere connesso necessariamente al binario di potenza sinistro (left power rail), mentre il collegamento con quello destro è opzionale

Elementi di Base del Linguaggio Ladder

❖ Power Rail



❖ Linee Elettriche Orizzontali



❖ Connessioni ai Power Rail



❖ Contatto Normalmente Aperto



❖ Contatto Normalmente Chiuso



❖ Coil

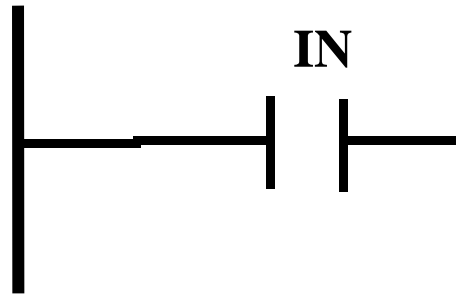


❖ Negated Coil

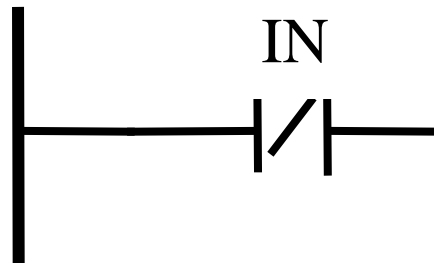


Utilizzo dei Contatti

- ❖ Ad ogni contatto viene associata una variabile binaria. Tale variabile viene solamente letta (può coincidere con un ingresso).
- ❖ **Contatto Normalmente Aperto:** la corrente fluisce da sinistra a destra se la variabile IN è 1. La corrente fluisce a destra per qualunque scansione del Programma Ladder fino a quando la variabile IN diviene 0

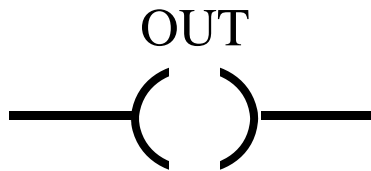


- ❖ **Contatto Normalmente Chiuso:** la corrente fluisce da sinistra a destra se la variabile IN è 0. La corrente fluisce a destra per qualunque scansione del Diagramma Ladder fino a quando la variabile IN diviene 1

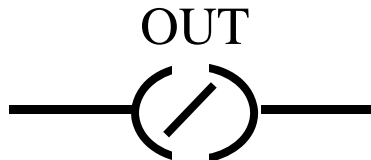


Utilizzo dei Coil

- ❖ Ad ogni coil viene associata una variabile binaria. La variabile viene scritta (può coincidere con una uscita fisica)
- ❖ **Coil:** la variabile OUT associata al **Coil** è posta a 1 se vi è una corrente che fluisce da sinistra. La variabile rimane a 1 per qualunque scansione del Programma Ladder fino a quando la corrente cessa di fluire da sinistra.

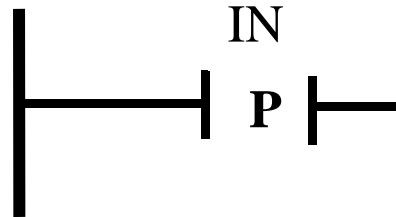


- ❖ **Negated Coil:** la variabile OUT associata al **Negated Coil** è posta a 0 se vi è una corrente che fluisce da sinistra. La variabile rimane a 0 per qualunque scansione del Programma Ladder fino a quando la corrente cessa di fluire da sinistra.



Altri Contatti del Linguaggio Ladder

- ❖ Contatto sensibile alla transizione 0-1 (Positive Transition-Sensing Contact)

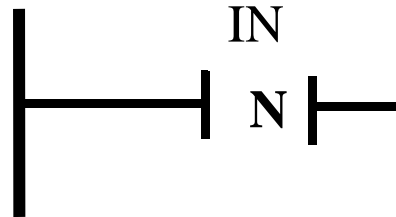


- ❖ La corrente fluisce da sinistra a destra del **Positive Transition-Sensing Contact**, se la variabile IN passa da 0 a 1. La corrente fluisce a destra solo per una scansione del Programma Ladder (quella relativa alla transizione).

Scansione	Valore di IN quando viene valutato il rung	Corrente alla Destra
1	OFF	OFF
2	ON	ON
3	ON	OFF
4	ON	OFF
5	OFF	OFF

Altri Contatti del Linguaggio Ladder

- ❖ Contatto sensibile alla transizione 1-0 (Negative Transition-Sensing Contact)

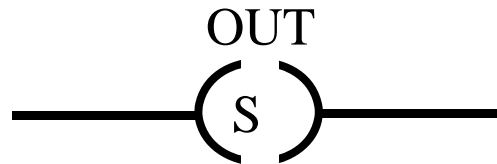


- ❖ La corrente fluisce da sinistra a destra del **Negative Transition-Sensing Contact**, se la variabile IN passa da 1 a 0. La corrente fluisce a destra **solo per una scansione del Diagramma Ladder** (quella relativa alla transizione)

Scansione	Valore di IN quando viene valutato il rung	Corrente alla Destra
1	ON	OFF
2	OFF	ON
3	OFF	OFF
4	OFF	OFF
5	ON	OFF

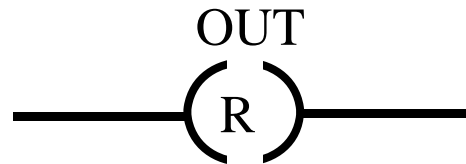
Altri Coil del Linguaggio Ladder

❖ Set Coil



- ❖ La variabile OUT associata al **coil** e' posta a 1 se vi e' una corrente che fluisce da sinistra. La variabile rimane a 1 per qualunque scansione del Diagramma Ladder fino a quando viene utilizzato un coil RESET.

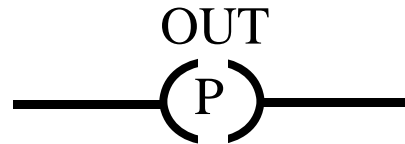
❖ Reset Coil



- ❖ La variabile OUT associata al **coil** e' posta a 0 se vi e' una corrente che fluisce da sinistra. La variabile rimane a 0 per qualunque scansione del Diagramma Ladder fino a quando viene utilizzato un coil SET.

Altri Coil del Linguaggio Ladder

❖ Positive Transition-Sensing Coil

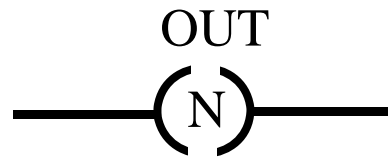


- ❖ La variabile OUT associata al **Positive Transition-Sensing Coil** è posta a 1 se la corrente che fluisce da sinistra passa da un valore FALSE ad un valore TRUE. La variabile rimane a 1 solo per una scansione del Diagramma Ladder (quella relativa alla transizione dello stato della corrente).

Scansione	Valore della corrente alla sinistra del coil quando viene valutato il rung	Valore di OUT
1	OFF	OFF
2	ON	ON
3	ON	OFF
4	OFF	OFF

Altri Coil del Linguaggio Ladder

❖ Negative Transition-Sensing Coil

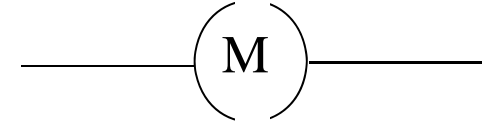


- ❖ La variabile OUT associata al **Negative Transition-Sensing Coil** è posta a 1 se la corrente che fluisce da sinistra passa da un valore TRUE ad un valore FALSE. La variabile rimane a 1 solo per una scansione del Diagramma Ladder (quella relativa alla transizione dello stato della corrente).

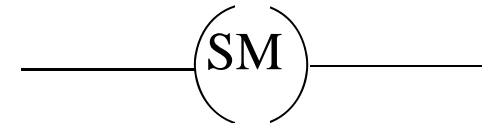
Scansione	Valore della corrente alla sinistra del coil quando viene valutato il rung	Valore di OUT
1	ON	OFF
2	OFF	ON
3	OFF	OFF
4	ON	OFF

Altri Coil del Linguaggio Ladder

❖ Retentive Memory Coil



❖ SET Retentive Memory Coil



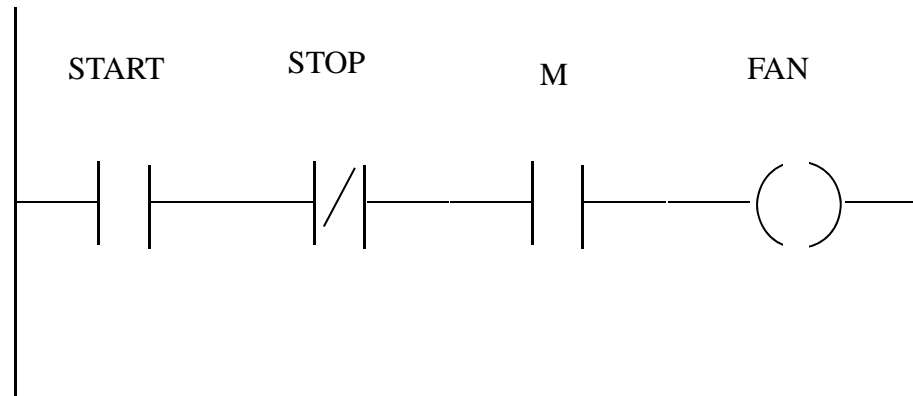
❖ RESET Retentive Memory Coil



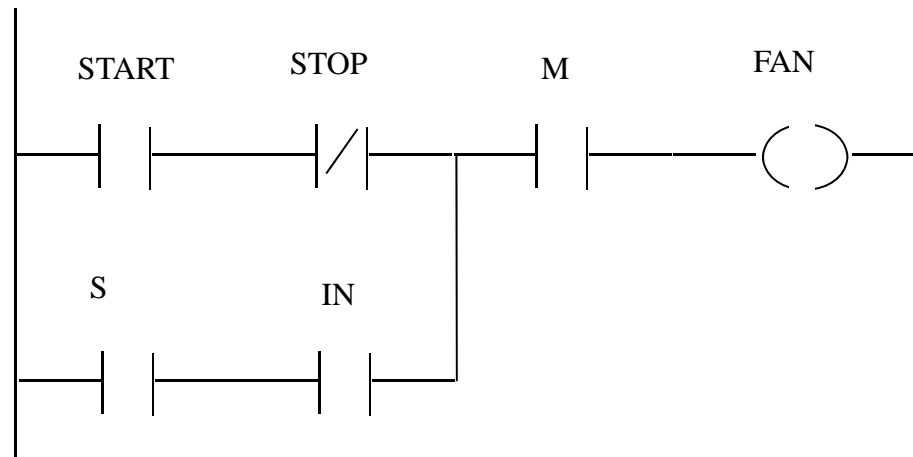
❖ Hanno lo stesso comportamento del coil, del SET coil e del RESET coil, ad eccezione del fatto che la variabile binaria associata ad essi viene dichiarata in modo automatico di tipo RETENTIVE

Logiche di Base Realizzabili con gli Elementi del Linguaggio Ladder

❖ Logica AND



❖ Logica OR

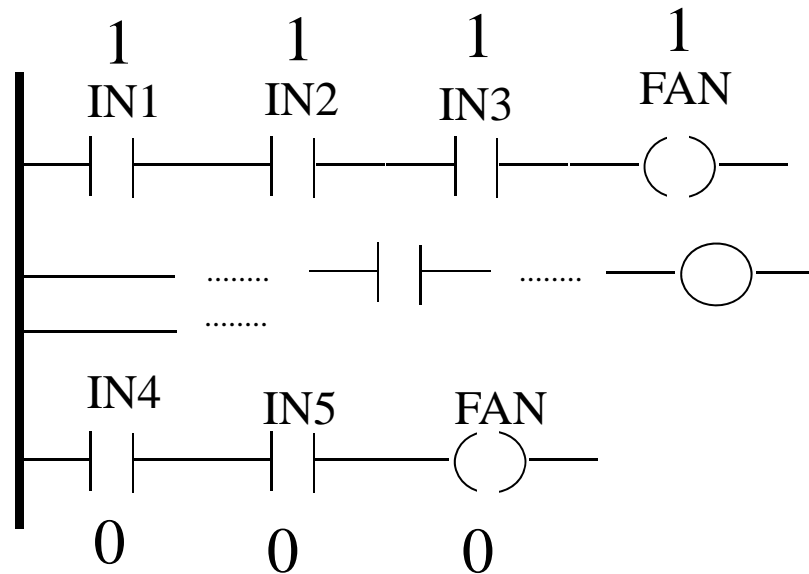


Regole di Esecuzione dei Rung

- ❖ Un programma scritto in linguaggio Ladder viene eseguito valutando un rung alla volta
- ❖ L'ordine di valutazione dei rung è quello che procede dal primo rung in alto verso l'ultimo rung in basso
- ❖ Quando l'ultimo rung viene valutato, si inizia nuovamente a valutare il primo rung (dopo aver aggiornato le uscite e letti gli ingressi)

Effetti Collaterali delle Regole di Esecuzione dei Rung

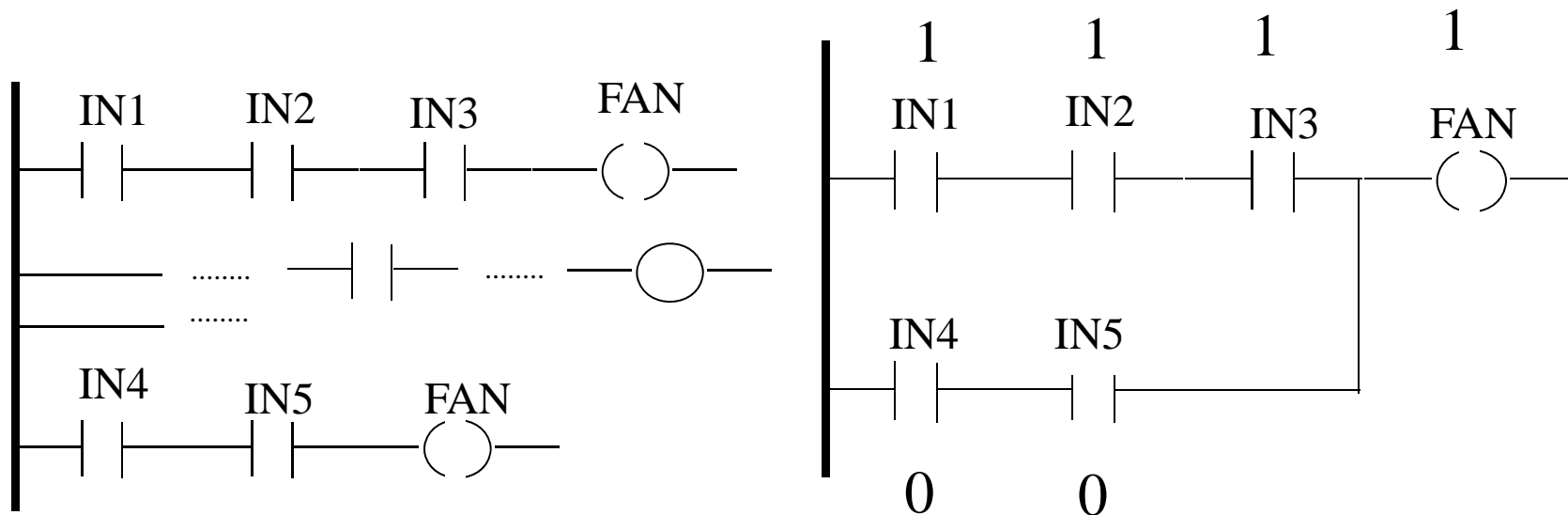
- ❖ L'ordine di valutazione comporta la necessità di riunificare i rung che operano delle modifiche (Write) sulle stesse **uscite** (**Attenzione: vale solo per i coil "normali"**)
- ❖ Esempio:



- L'effetto di IN1, IN2, e IN3 sull'**uscita** reale collegata alla variabile FAN è nullo. L'uscita reale può essere modificata solo da IN4 e IN5, a causa della posizione del rung che li contiene

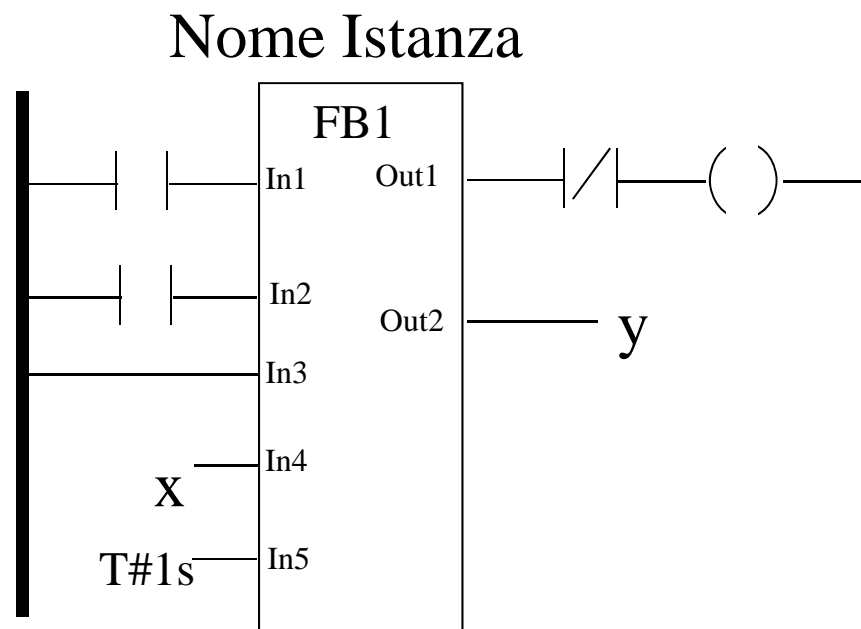
Effetti Collaterali delle Regole di Esecuzione dei Rung

- ❖ Riunificando i rung, l'uscita reale collegata alla variabile FAN verrà aggiornata solo dopo aver valutato il rung composto dagli ingressi IN1, IN2, IN3, IN4 e IN5



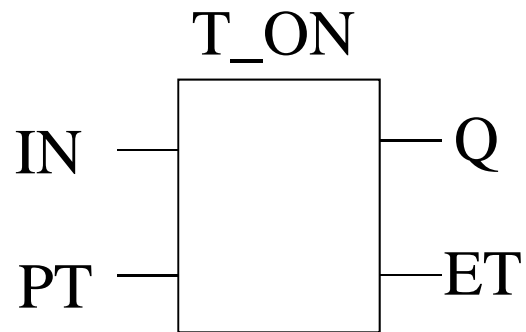
Uso di Istanze di Function Block e di Funzioni

- ❖ E' possibile connettere contatti con gli ingressi dell'istanza del FB o della funzione, purché essi siano binari
- ❖ E' possibile connettere coil con le uscite dell'istanza del FB o della funzione, purché esse siano binarie
- ❖ Nel caso in cui una Istanza di FB o una funzione richieda un ingresso binario sempre TRUE, è possibile collegare tale ingresso direttamente al power rail di sinistra
- ❖ Eventuali variabili analogiche o valori analogici (interi, reali, temporali, etc.) possono essere connessi direttamente ai corrispondenti ingressi dell'istanza del FB o della funzione

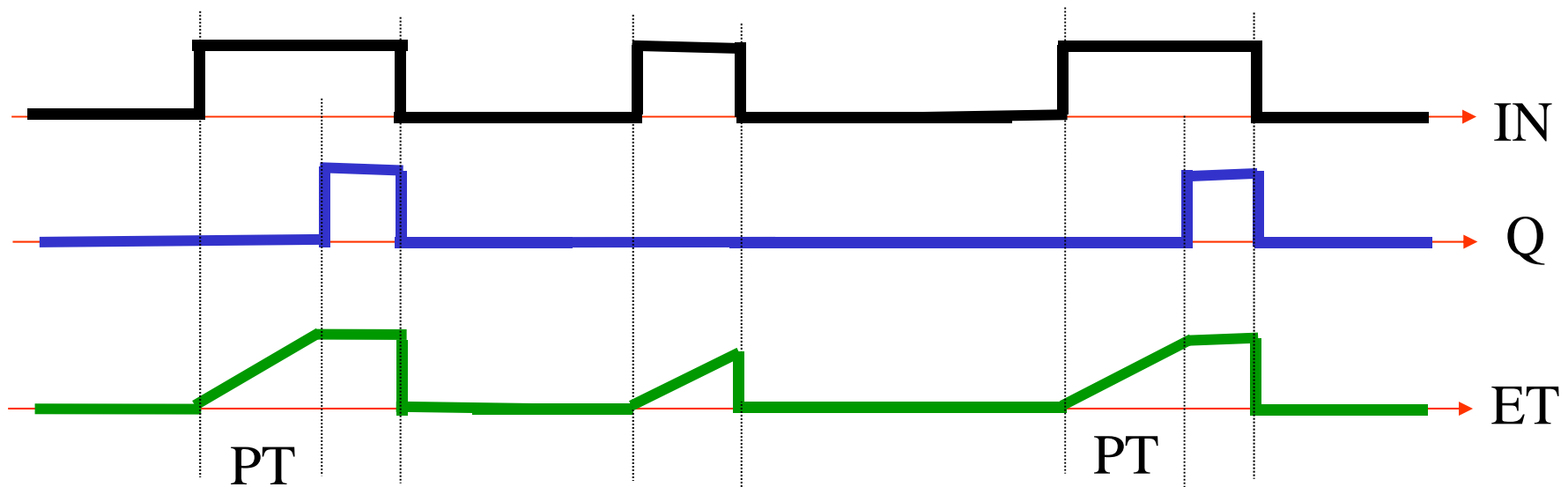


Function Block di Uso Comune

Timer T_ON

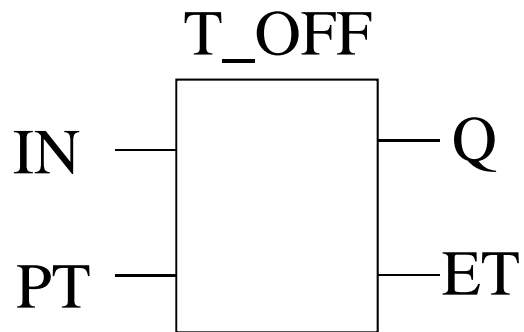


- ❖ IN: BOOL. If a rising edge is detected, the on-delay timing is started
- ❖ PT: TIME. Preset time interval for the delay
- ❖ Q: BOOL. Output
- ❖ ET: TIME. Elapsed time interval

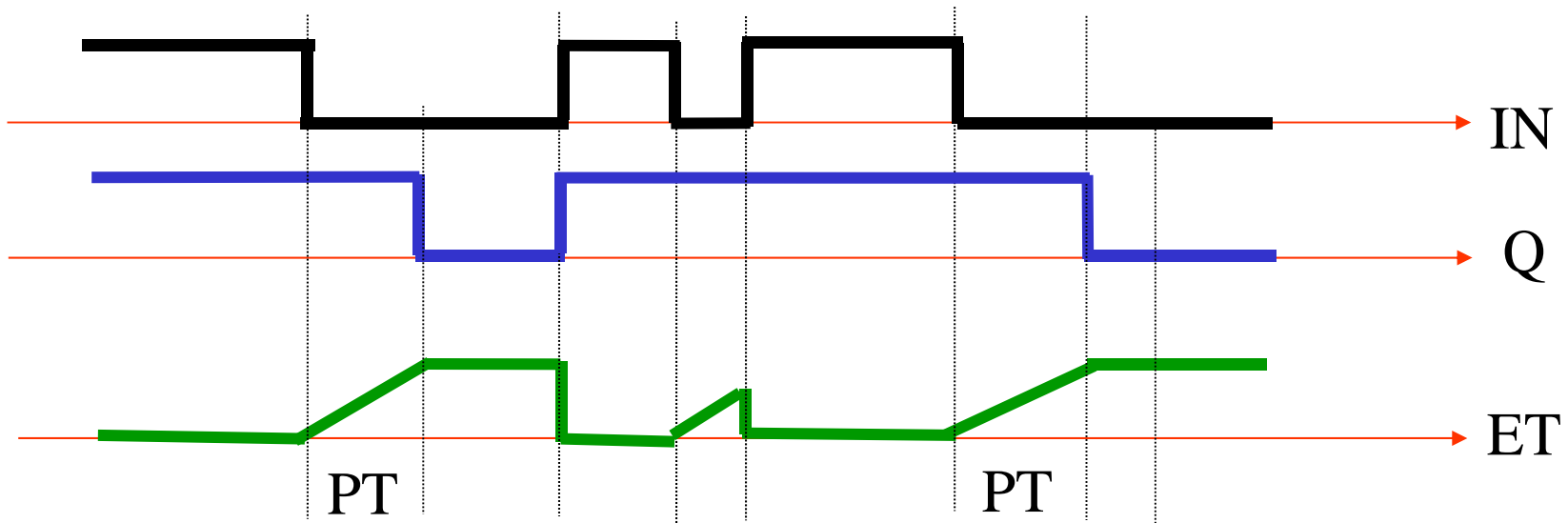


Function Block di Uso Comune

Timer T_OFF



- ❖ IN: BOOL. If a falling edge is detected, the off-delay timing is started.
- ❖ PT: TIME. Preset time interval for the delay
- ❖ Q: BOOL. Output
- ❖ ET: TIME. Elapsed time interval



Function Block di Uso Comune

❖ Bistabili

➤ SR, RS

❖ Bitwise Boolean

➤ AND, OR, NOT, XOR

❖ Comparison

➤ EQ, LE, LT, GE, GT, NE

❖ Counters

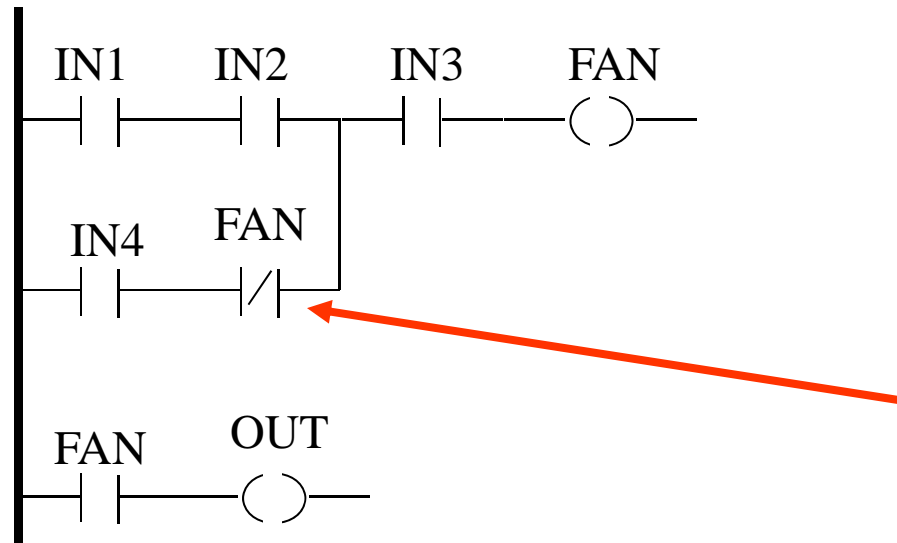
➤ CTD (down), CTU (up)

❖ Altri (disponibili su alcuni PLC)

➤ PID

Feedback Paths

- ❖ E' possibile che un rung presenti dei contatti e dei coil ai quali vengono associate le stesse variabili



- ❖ Il valore della variabile associata al contatto FAN è quello valutato nell'ultima valutazione (rung precedente)
- ❖ **NOTA:** L'uso dei Feedback può essere pericoloso, in quanto può portare ad una instabilità delle uscite del PLC.
- ❖ Ad esempio, l'uscita FAN diviene instabile se inizialmente FAN=0 e nel tempo gli ingressi IN1, IN2, IN3 e IN4 si mantengono costanti e pari a IN1=0, IN2=0, IN3=1, IN4=1

Tecnica di Programmazione con il Ladder

❖ Tecnica basata su Relazioni I/O

- La tecnica di programmazione più istintiva e naturale che è possibile applicare al linguaggio Ladder è quella che permette di esprimere tramite i rung le relazioni causa effetto, che legano le uscite da comandare agli ingressi o a particolari condizioni logiche interne.

❖ Tecnica basata sulla Macchina a Stati

- Deve essere applicata a problemi più complessi, in cui l'attivazione delle uscite non dipende esclusivamente dagli ingressi (o da variabili interne, quali bits, contatori, timers) ma è legata al concetto di "stato".

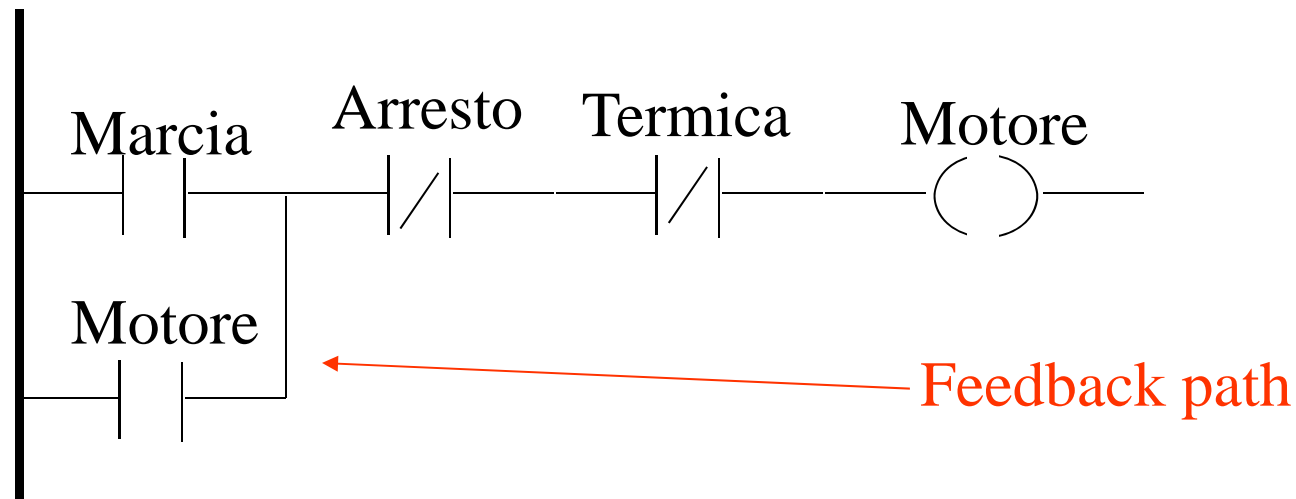
Tecnica basata su Relazioni I/O

- ❖ La scrittura di un programma in Ladder deve prevedere:
 - l'esplicitazione delle relazioni tra le uscite da comandare e gli ingressi o particolari condizioni logiche interne
 - la loro traduzione utilizzando i simboli del linguaggio Ladder.
- ❖ Nel seguito verranno mostrati due esempi che permettono di comprendere meglio quanto detto.

Tecnica basata su Relazioni I/O

Marcia Arresto Motore

- ❖ Classico problema di gestione dell'accensione/spegnimento di un motore
- ❖ L'accensione del motore è comandata da un pulsante di Marcia, che deve essere attivato (normalmente aperto). Una volta attivato, la sua posizione non è più rilevante per il mantenimento dell'accensione del motore.
- ❖ Per la gestione di eventuali emergenze, è previsto l'utilizzo di un pulsante di Arresto (normalmente chiuso).
- ❖ Infine il motore deve essere interrotto nel caso di surriscaldamento, ossia a seguito dell'intervento della termica (normalmente chiuso).



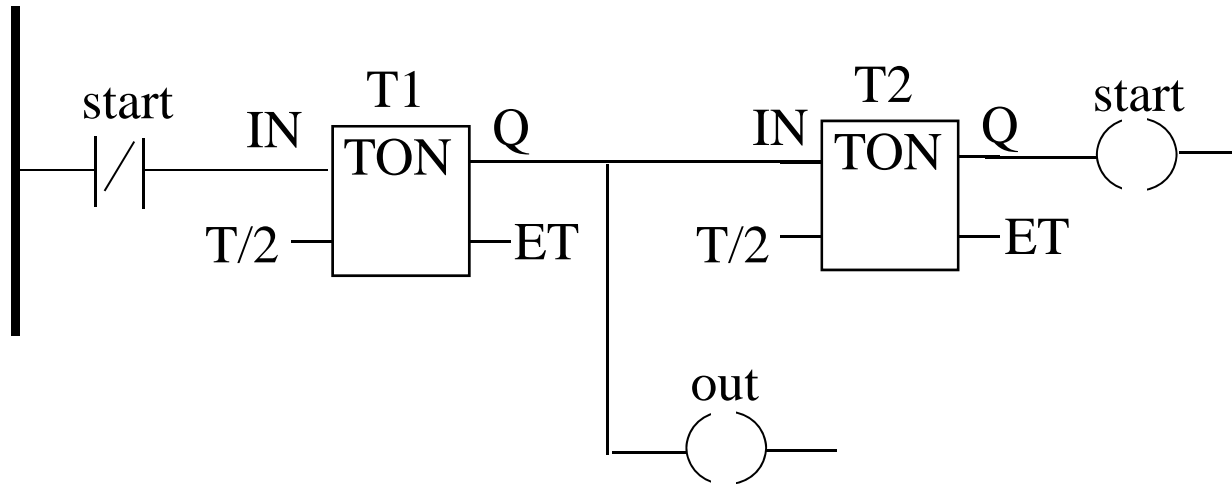
Tecnica basata su Relazioni I/O

Circuito di Clock

- ❖ Si supponga di voler realizzare un programma che permetta di fornire in uscita al PLC un segnale periodico ad onda quadra.
- ❖ Sia T il periodo del segnale.
- ❖ Sia out la variabile binaria alla quale viene associata l'uscita fisica del PLC per la quale si vuole produrre il segnale periodico.
- ❖ Si consideri nella soluzione del problema una variabile binaria interna ($start$), inizializzata a OFF (0).
 - Feedback sulla variabile $start$
- ❖ Si considerino, infine, due function block timer TON, denominati $T1$ e $T2$, ciascuno caratterizzato dal valore del PT pari a $T/2$ (semiperiodo).

Tecnica basata su Relazioni I/O

Circuito di Clock



Scansione	start contatto	T1		T2		out	start coil
		Q	ET	Q	ET		
1..n	0	0	<T/2	0	0	0	0
n+1	0	1	T/2	0	0	1	0
n+2...m	0	1	T/2	0	<T/2	1	0
m+1	0	1	T/2	1	T/2	1	1
m+2	1	0	0	0	0	0	0
m+3	0	uguale alla scansione 1					
	0	stessa sequenza precedente					

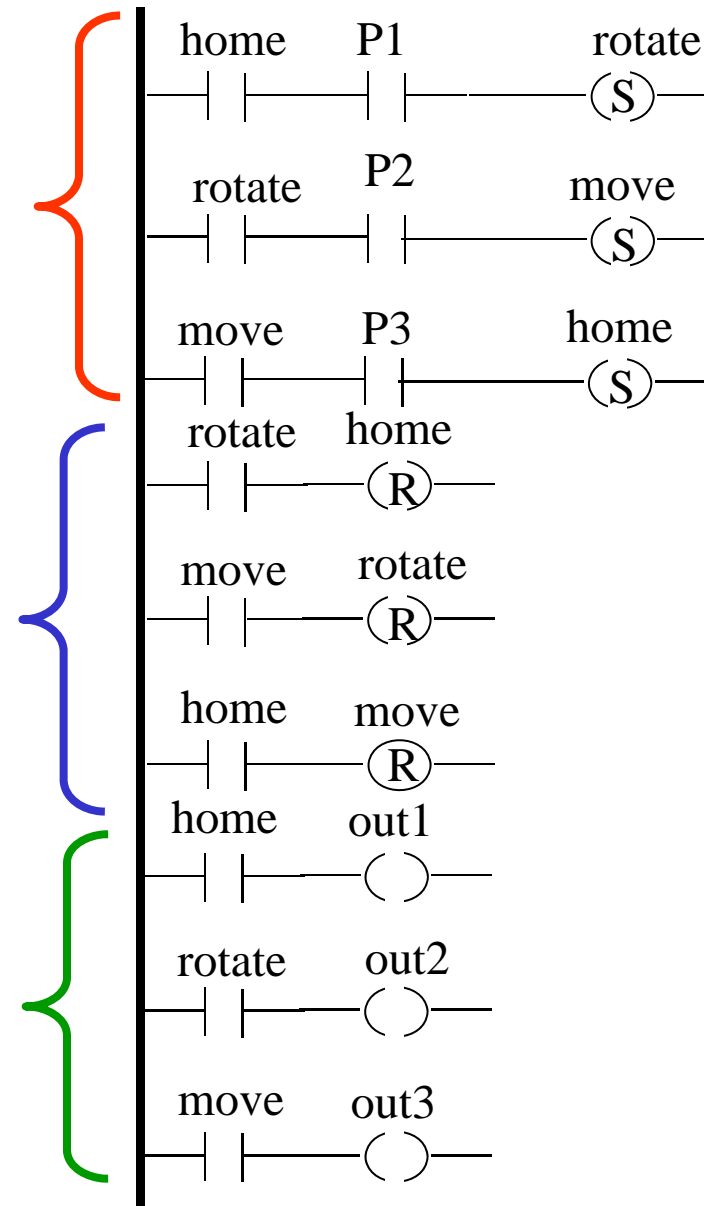
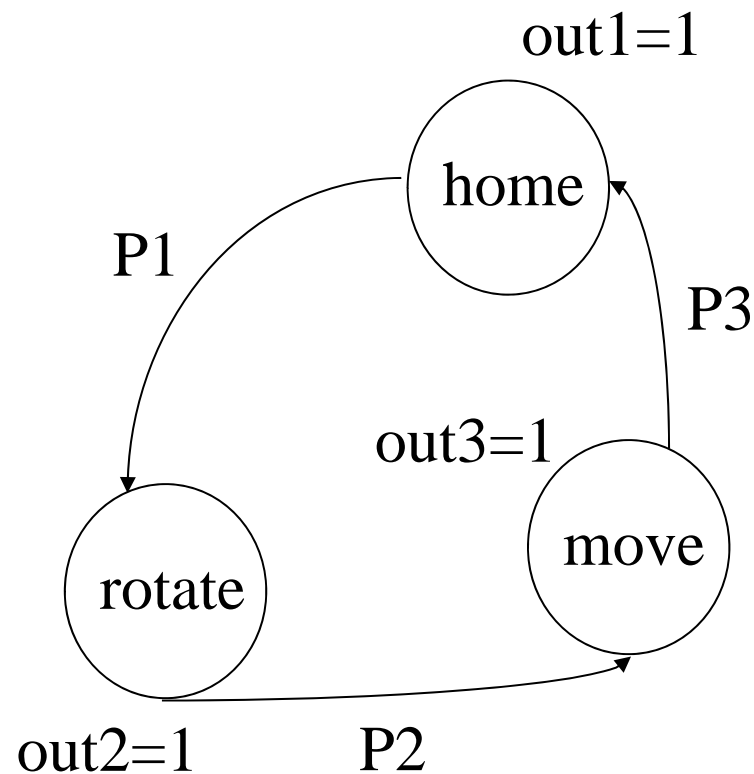
Tecnica basata sulla Macchina a Stati

- ❖ Esistono problemi in cui l'attivazione delle uscite (ad esempio i comandi agli attuatori) non dipende esclusivamente dagli ingressi (o da variabili interne, quali bits, contatori, timers) ma è legata al concetto di "stato".
- ❖ Tali problemi sono caratterizzati da soluzioni che prevedono l'evoluzione del sistema da uno stato ad un altro, a partire da uno stato iniziale per far ritorno, spesso, a tale stato.
- ❖ Per tali problemi, il comando di uno o più attuatori avviene in corrispondenza di uno stato, e può verificarsi che lo stesso attuatore venga attivato in due o più stati differenti anche in corrispondenza di ingressi diversi.
- ❖ L'evoluzione del sistema da uno stato ad un altro avviene in corrispondenza di valori assunti da particolari ingressi, oppure in base a valori di timers o di contatori, ovvero da valori di opportune espressioni logiche.

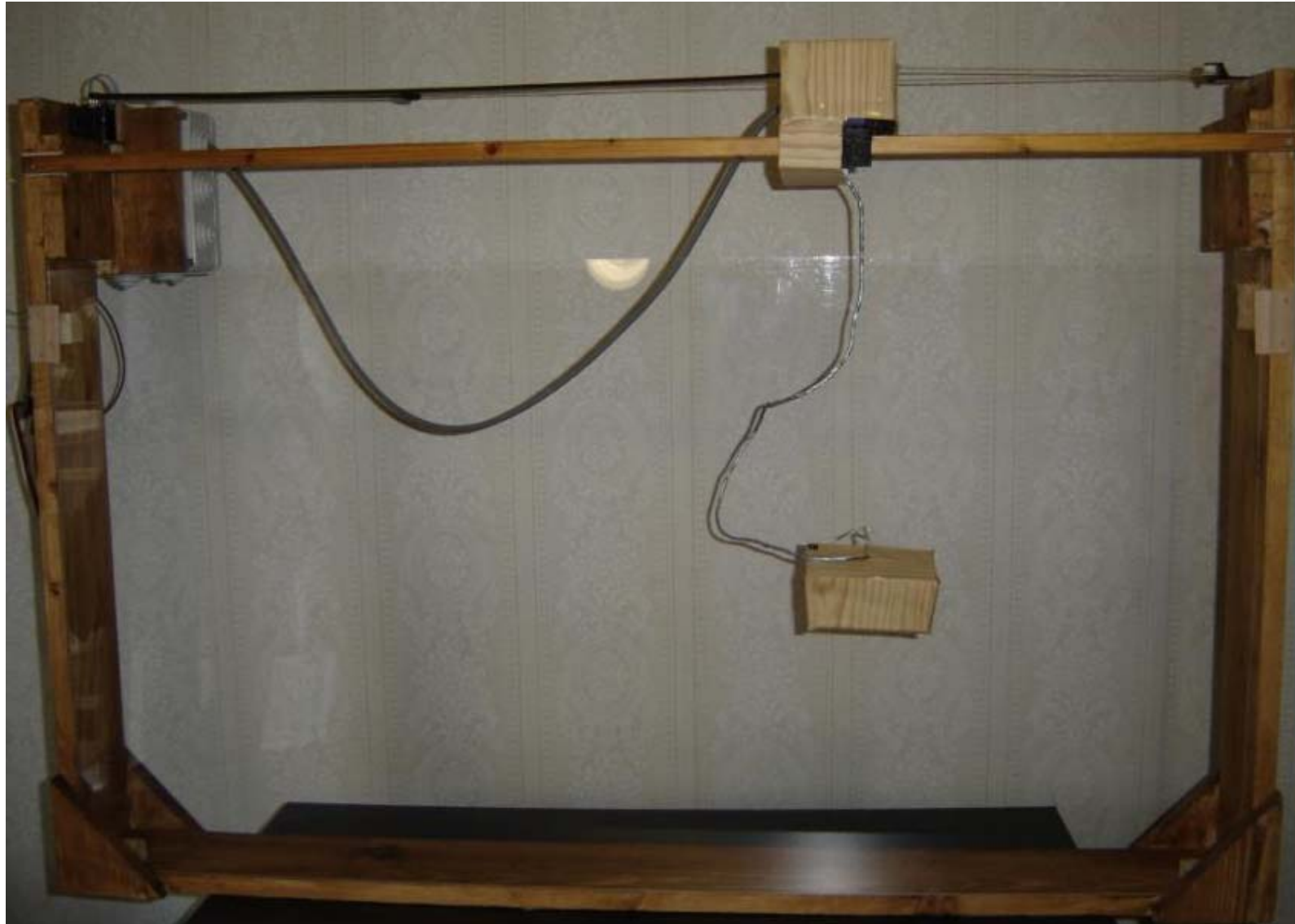
Tecnica basata sulla Macchina a Stati

- ❖ Rappresentare la soluzione del problema con una macchina a stati
- ❖ Ogni stato viene rappresentato da una variabile binaria
- ❖ Per ogni stato vengono identificate le azioni da eseguire
- ❖ Vengono identificati gli eventi che producono il passaggio di stato.
- ❖ Ciascun evento dovrà essere rappresentato da una variabile binaria
- ❖ Il programma in Ladder si compone di tre porzioni:
 - Rappresentazione dell'attivazione di un nuovo stato a partire dallo stato corrente a seguito di un determinato evento
 - Rappresentazione della disattivazione dello stato precedente a causa dell'attivazione di un nuovo stato
 - Rappresentazione delle azioni eseguite in ciascuno degli stati

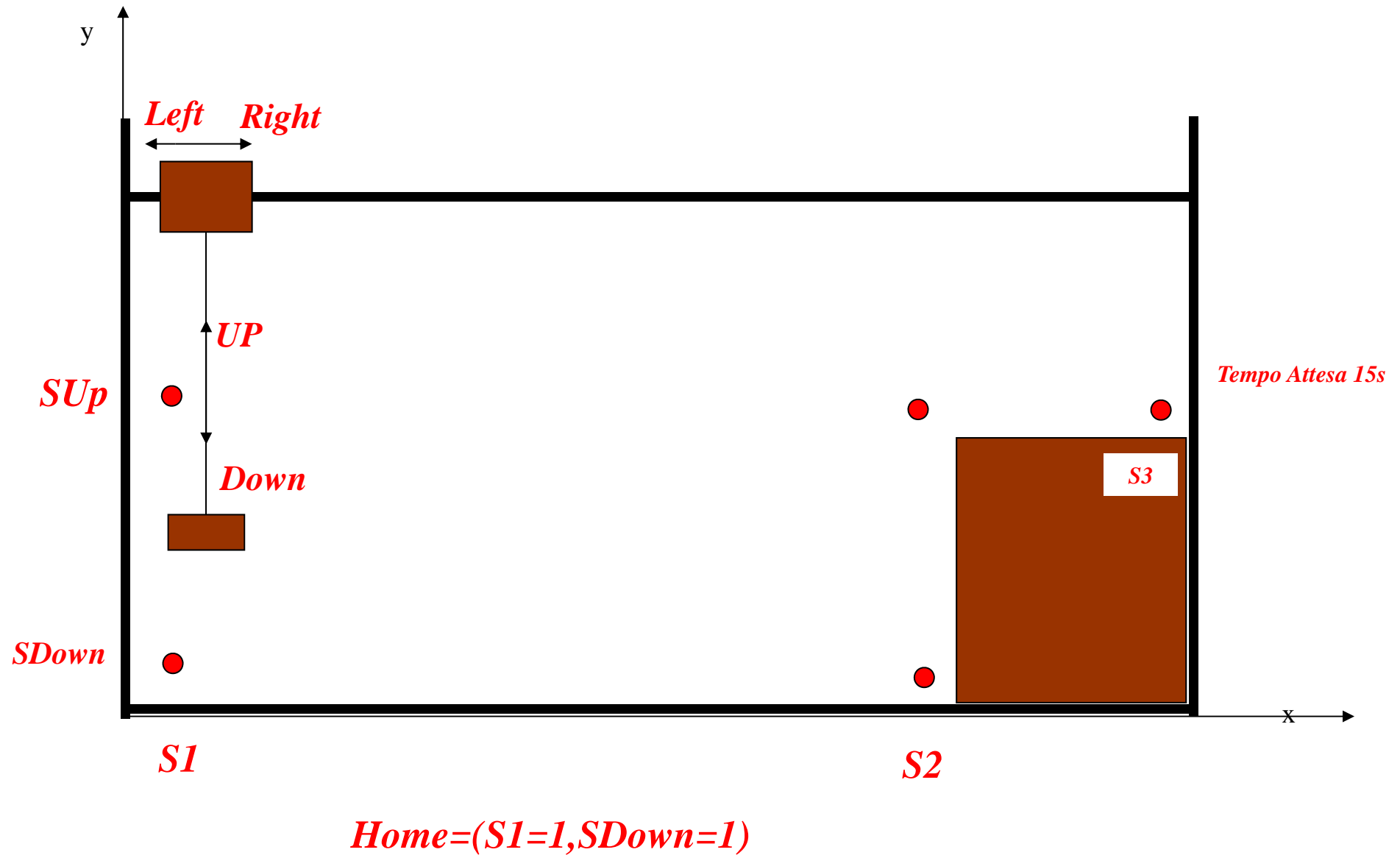
Esempio di Tecnica di Programmazione basata sulla Macchina a Stati



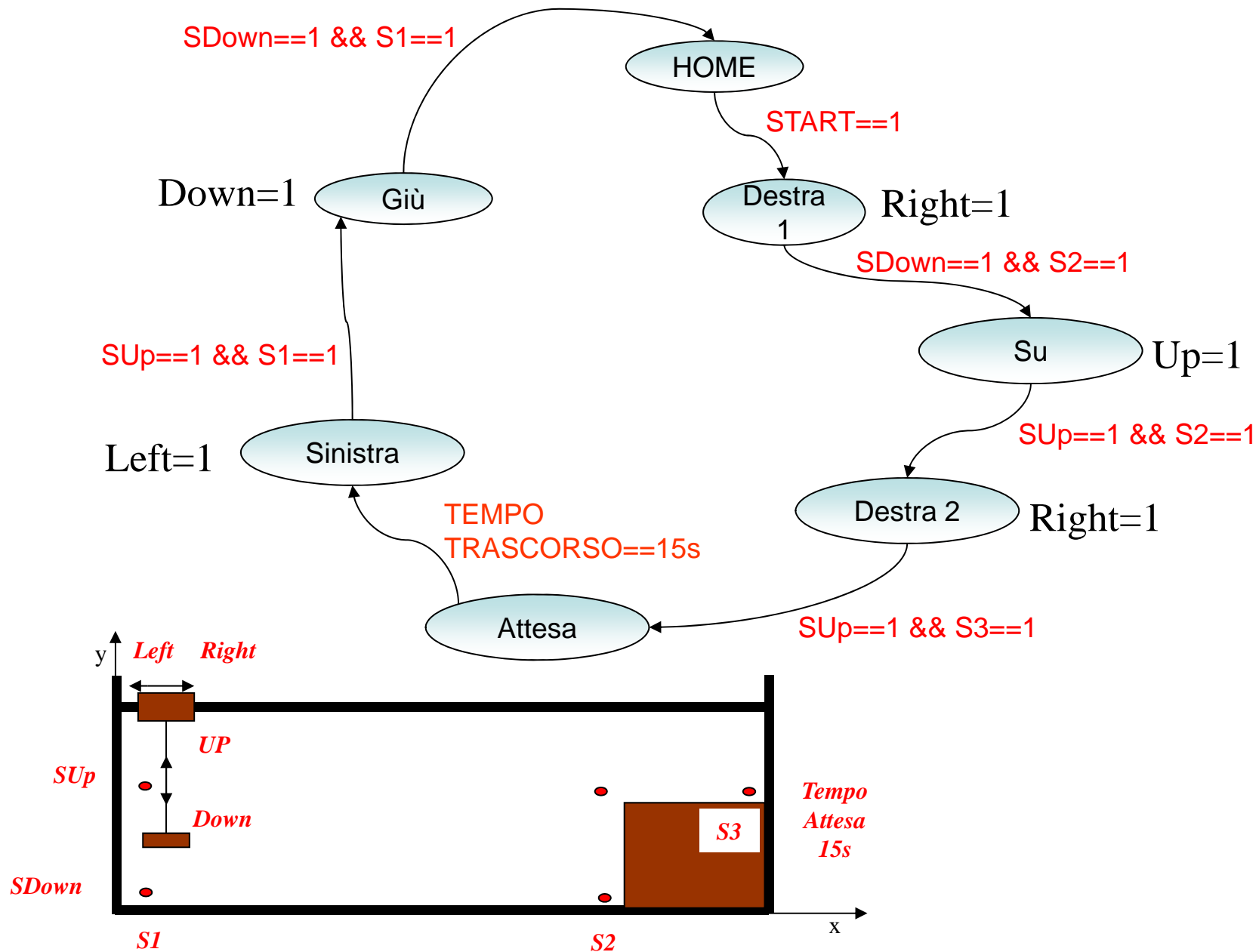
Esempio: Carroponte



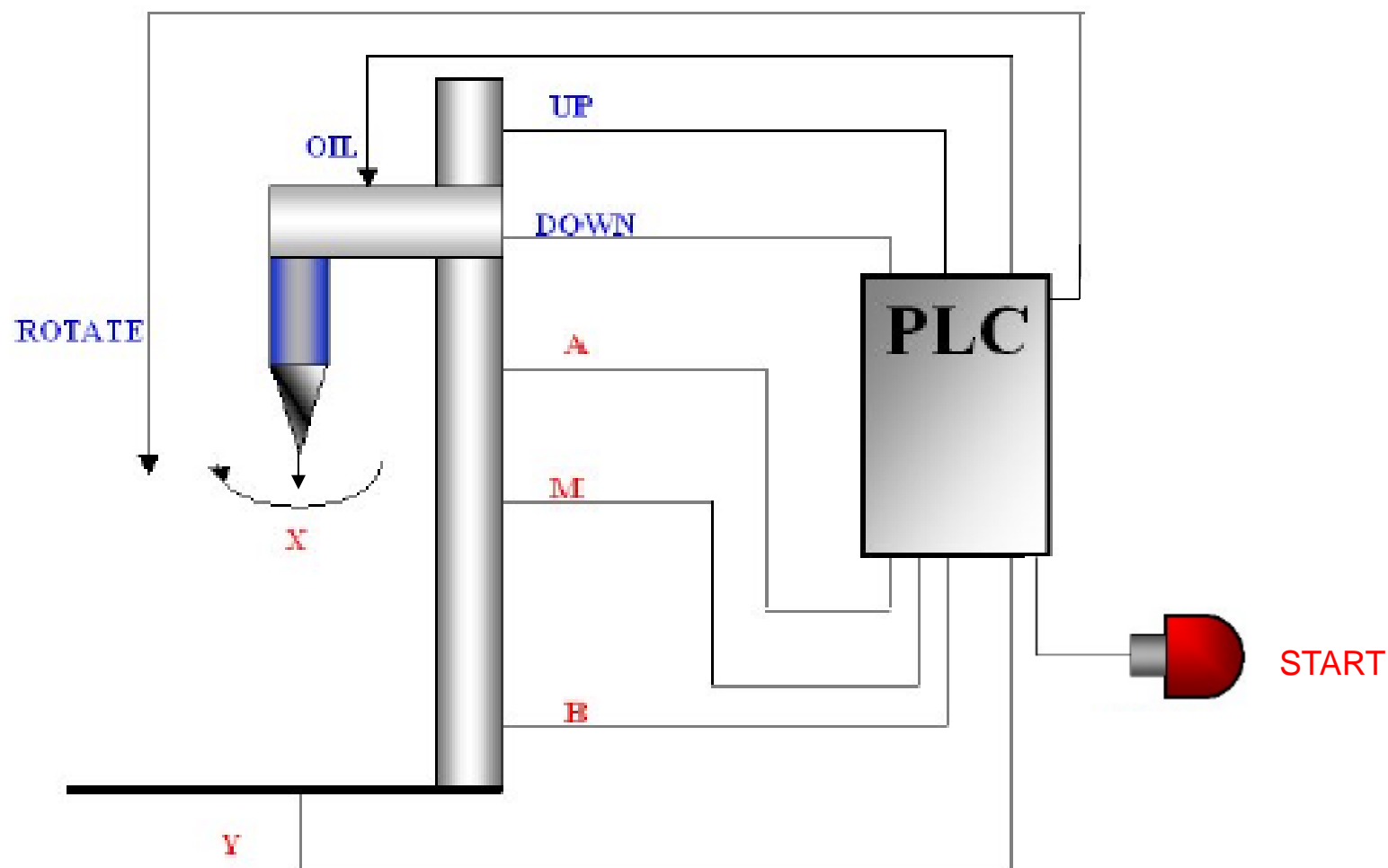
Esempio: Carroponte

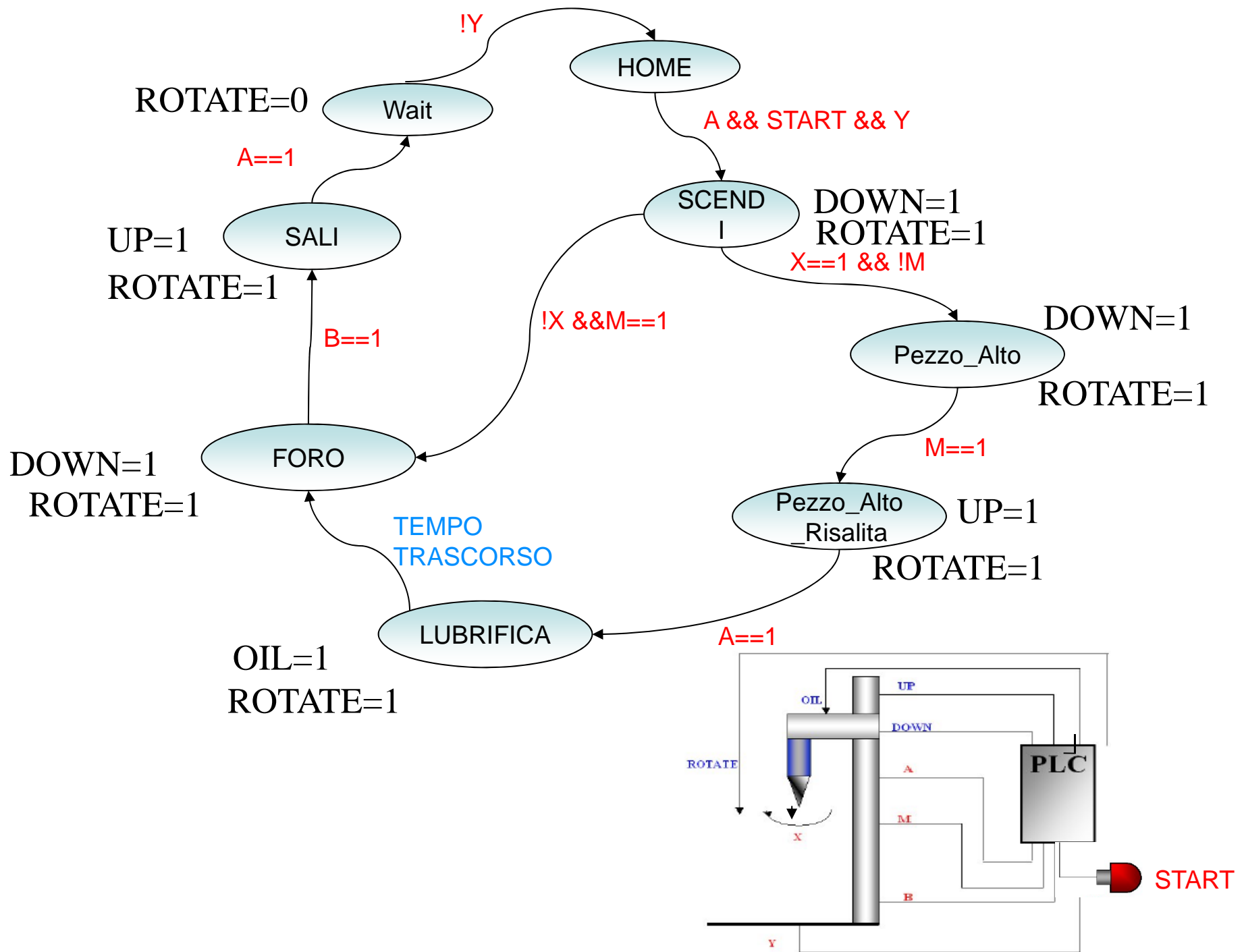


Esempio: Carroponte



Esercizio: Trapano Automatico





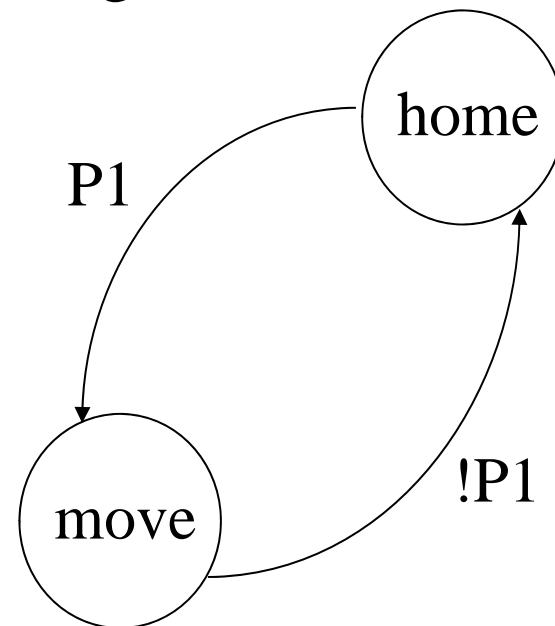
Esempio di Cattiva Programmazione

❖ La tecnica di programmazione vista prima non funziona quando:

- La macchina a stati ha solo due stati
- Gli eventi che determinano il passaggio da uno stato all'altro sono mutuamente esclusivi

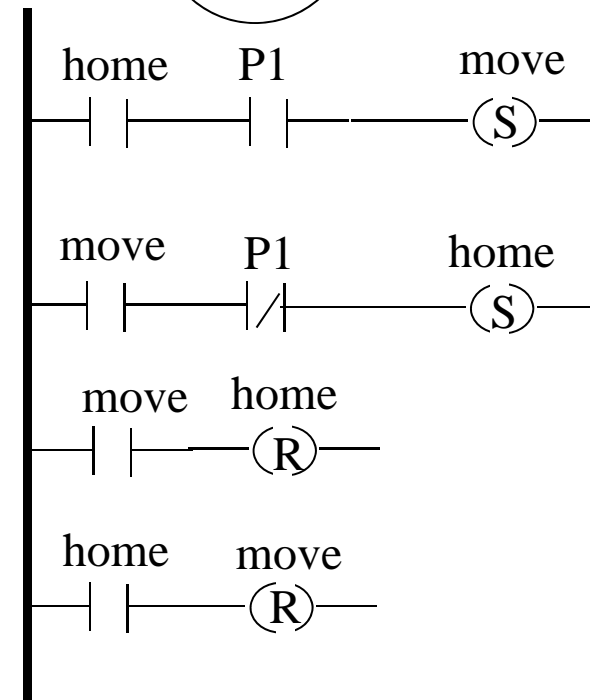
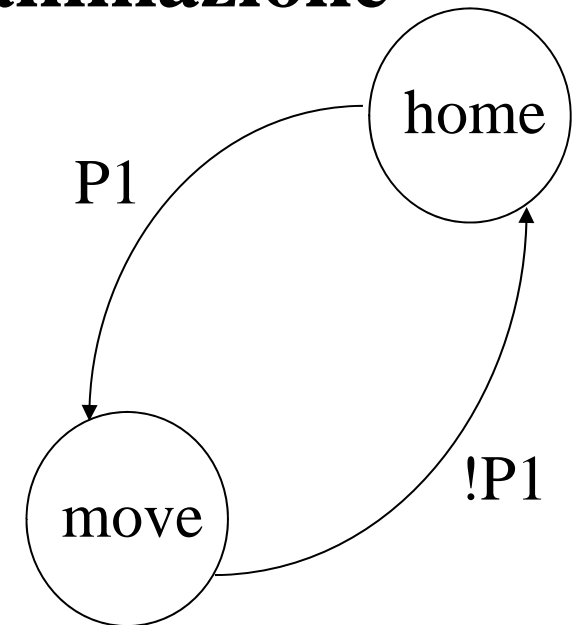
❖ Esempio: due stati (home e move) e gli eventi P1 e !P1

- Se $P1=0$, $home=1$ e $move=0$
- Se $P1=1$, $home=0$ e $move=1$

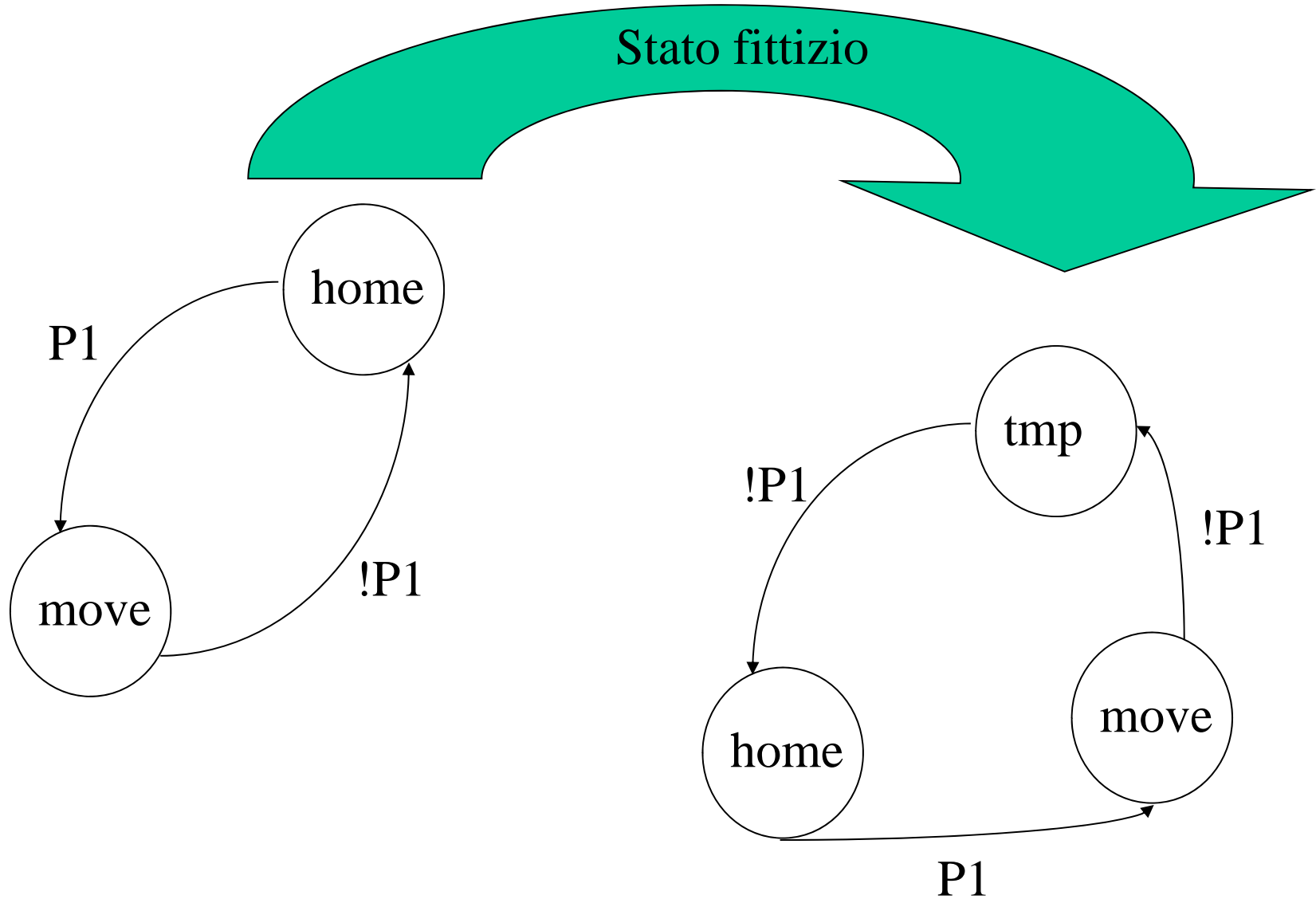


Esempio di Cattiva Programmazione

- ❖ Start: $P1=0$ e $home=1$
 - Ciclicamente $home=1$, $move=0$
- ➡ Si supponga: $P1=1$
 - $move=1$
 - Ciclicamente $home=0$, $move=1$
- ❖ Si supponga: $P1=0$
 - $home=1$
 - $home=0$
 - Ciclicamente $home=0$, $move=1$



Esempio di Buona Programmazione



Esempio di Buona Programmazione

❖ Start: $P1=0$ e $tmp=1$

➤ Ciclicamente $tmp=0$, $home=1$, $move=0$

➡ Si supponga: $P1=1$

➤ $move=1$

➤ $home=0$

➤ Ciclicamente $tmp=0$, $home=0$, $move=1$

❖ Si supponga: $P1=0$

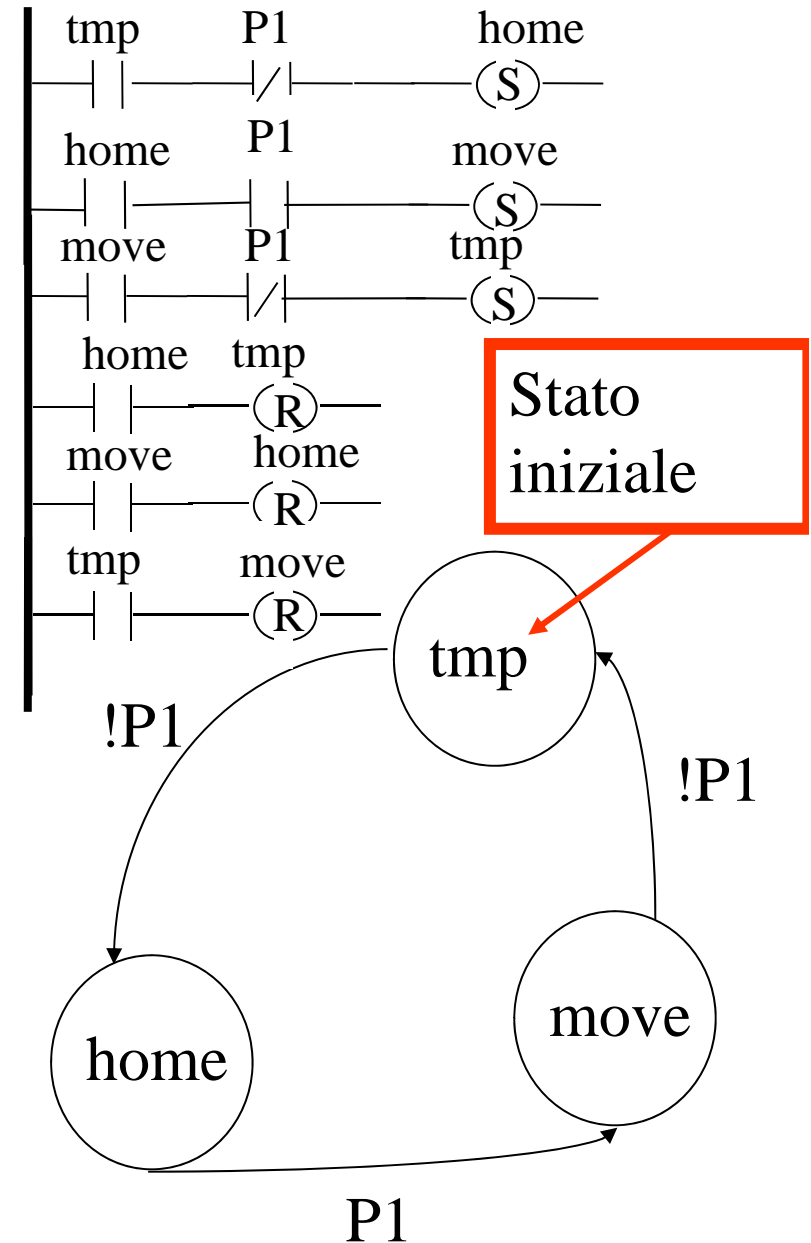
➤ $tmp=1$

➤ $move=0$

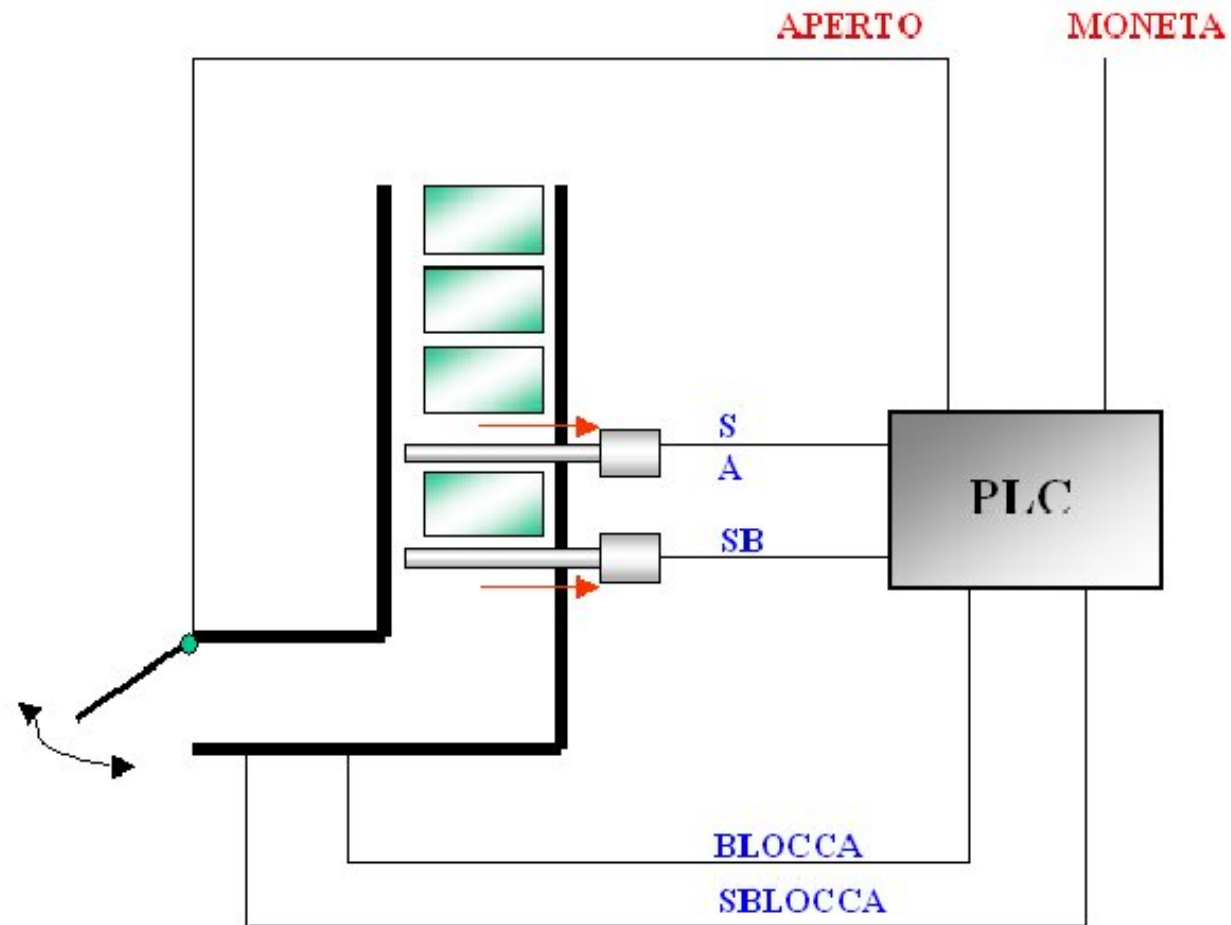
➤ $home=1$

➤ $tmp=0$

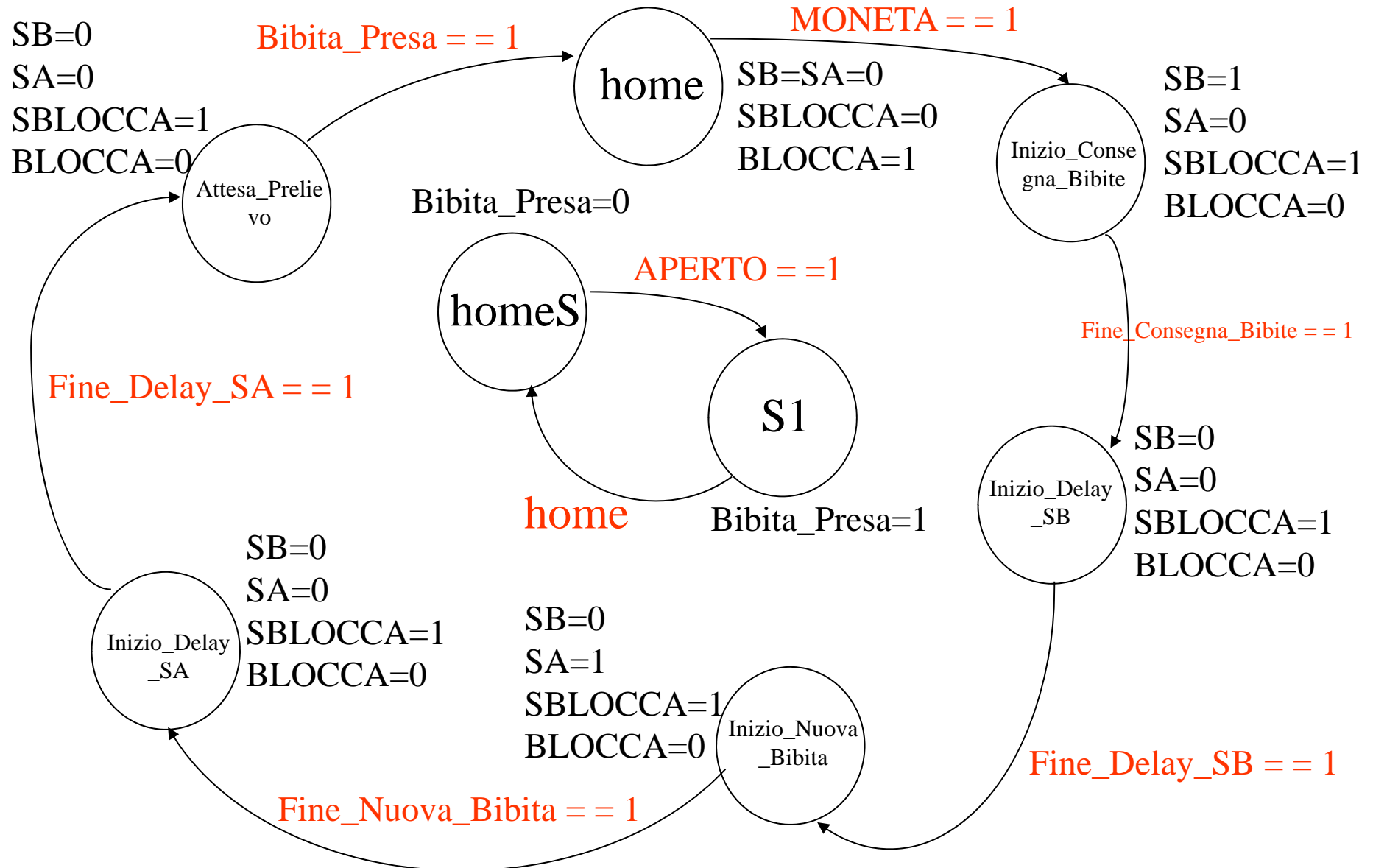
➤ Ciclicamente $tmp=0$, $home=1$, $move=0$



Esercizio: Distributore di Bibite



Esercizio: Distributore di Bibite



Due Soluzioni:

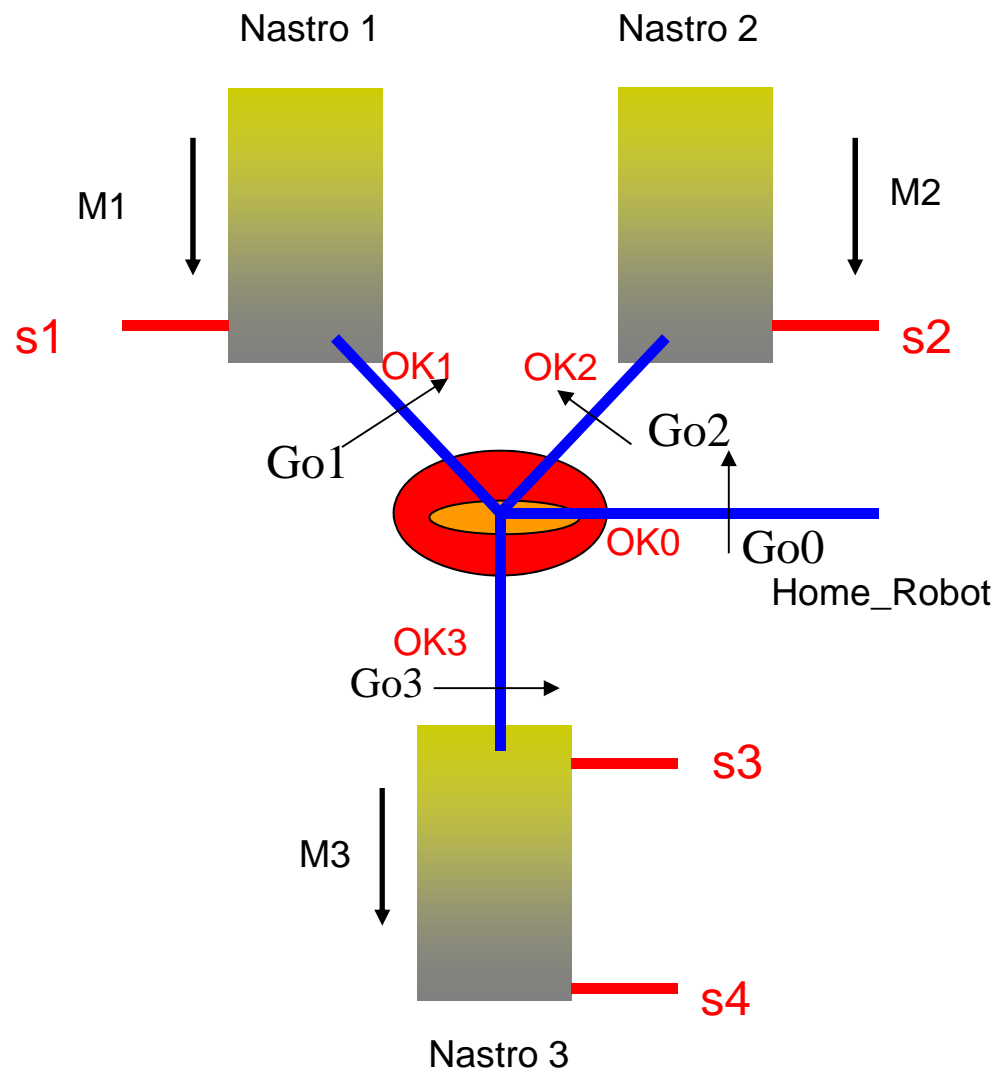
❖ Realizzazione delle due macchine a stato in un unico Programma con task ciclico

➤ **Limite: Frequenza di campionamento segnale "Aperto" non compatibile con il program scan**

❖ Realizzazione delle due macchine a stato con due Programmi controllati da due task ciclici con frequenze diverse

➤ **Attenzione alla codifica della macchina con due stati!**

Esercizio: 3 Nastri



Esercizio: 3 Nastri

