PLC E SCADA: RICHIAMI A CORSI PRECEDENTI

- **Appunti e selezione dei lucidi di Sistemi per l'Industria e PLC**
  - **Chi ha già sostenuto o frequentato Sistemi per l'Industria o PLC o Sistemi distribuiti e PLC troverà alcune duplicazioni rispetto al corso frequentato**
  - **Le informazioni nel presente documento che fanno riferimento al corso della triennale possono incidentalmente essere presenti nei temi d'esame**
  - **Per approfondimenti si faccia riferimento al materiale di Sistemi per l'Industria e PLC (http://alessandra-flammini.unibs.it/ -> SDPLC e SIPLC -> Sistemi per l'Industria e PLC)**

Nota: Le slide con asterisco * sono da considerarsi approfondimenti (non saranno presenti negli esami)

PLC: CARATTERISTICHE GENERALI

- **HW modulare o compatto**
  - **Single o multi-CPU**
  - **Diversi moduli**
    - **CPU**
    - **ingressi logici**
    - **uscite logiche**
    - **ingressi analogici**
    - **ingressi dedicati**
    - **....**
    - **moduli funzionali**

- **SW "semplice"**
  - **schemi a contatti**
  - **struttura ciclica**
  - **autodiagnostica**

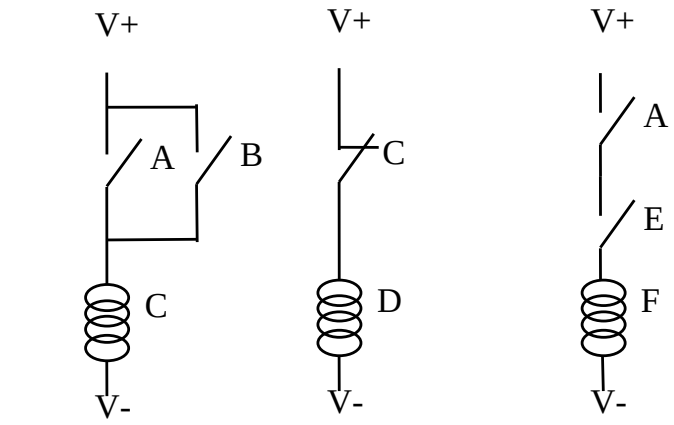
Programmazione: Configurazione, Tabella Simboli e Blocco Codice

- **Il PLC spesso è un sistema modulare, quindi devo dirgli di quali moduli è composto**
  - **Configurazione**
  - **Indirizzamento geografico delle risorse (se aggiungo due moduli uguali di ingressi logici, l'indirizzo degli ingressi sarà diverso e dipende dalla posizione del modulo rispetto alla CPU)**

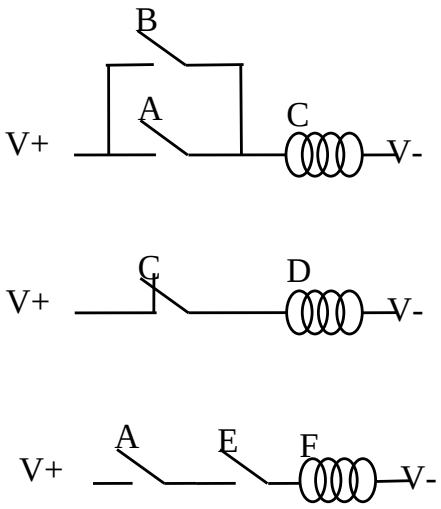
- **Il PLC ha ingressi e uscite che vengono collegate a sensori e attuatori secondo quanto descritto negli schemi funzionali**
  - **Lista delle attribuzioni o Tabella di I/O**
  - **Il PLC è una scatola nera e vedo tutti i segnali che entrano e che escono**

- **Le funzionalità e le relazioni tra ingressi e uscite sono descritte nel programma**
  - **Blocco di codice**

DAGLI SCHEMI A RELAIS AD UN “LINGUAGGIO”



C = A OR B D = NOT(C) F = A AND E



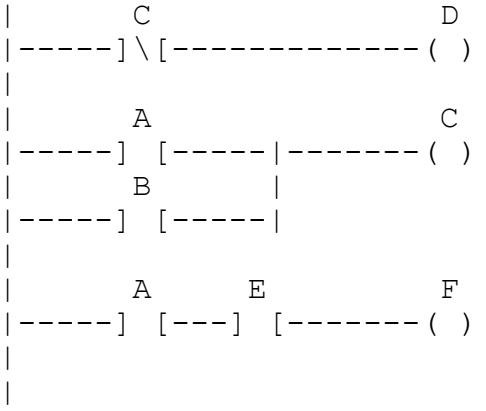

```
|       A                 C
|-----] [-----|-------( )
|       B       |
|-----] [-----|
|
|       C                 D
|-----]\[-------------( )
|
|       A     E           F
|-----] [---] [-------( )
|
|
                Programma 1
```

```
|       C                 D
|-----]\[-------------( )
|
|       A                 C
|-----] [-----|-------( )
|       B       |
|-----] [-----|
|
|       A     E           F
|-----] [---] [-------( )
|
|
                Programma 2
```

- **Routando a sinistra di 90° uno schema a relais si ottiene qualcosa che assomiglia ad una sequenza di istruzioni (programma)**
- **Nello schema a relais tutti i segmenti sono eseguiti contemporaneamente**
- **Nei programmi le istruzioni vengono eseguite in sequenza**
  - **I programmi 1 e 2 sono uguali?**

IL “LINGUAGGIO” LOGICO BOOLEANO

- **Questo nuovo “linguaggio”, derivato dagli schemi a relais per essere comprensibile agli operatori (anni 80). L’uscita era una bobina il cui valore è assegnato secondo la logica *if input then Out=1 else Out=0***
  - **NOT (si prende il contatto normalmente chiuso, come nei relais)**
  - **AND (si prende la serie dei contatti, come nei relais)**
  - **OR (si prende il parallelo dei contatti, come nei relais)**
  - **SET o RESET (si usa una memoria che viene vista come una “bobina speciale”, <u>è più semplice gestire la memoria rispetto alle autoritenute</u>) *if input then Out=1***
  - **Si possono creare altre funzioni (es. rilevatore di fronte) che con i relais erano molto più difficili da realizzare**



NOT

OR

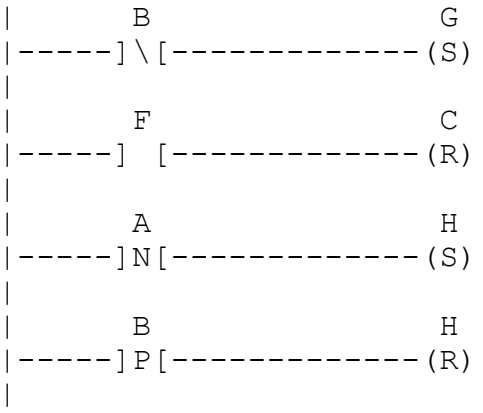
AND

SET

RESET

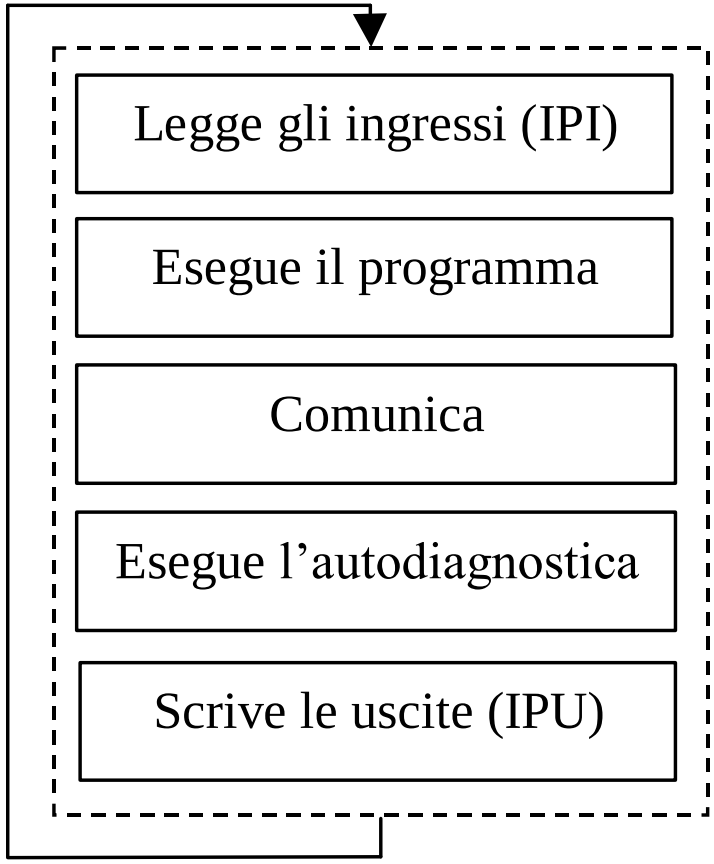
Rilevatore di fronte negativo

Rilevatore di fronte positivo



CICLO DI FUNZIONAMENTO (CICLO DI SCANSIONE)

- **Il PLC viene usato per effettuare ciclicamente una serie di istruzioni del tipo "se... allora..."**
- **Ciclo di scansione**
  - **T_ciclo_min ~ ms**
- **Esecuzione sequenziale del programma**
- **Architettura ciclica**
  - **no tempi di attesa**
  - **macchine a stati?**
- **Fase "comunica"**
  - **diagnostica via PC**
  - **Fase di background a T controllato**
- **Fase "autodiagnosi"**
  - **diagnostica locale (CPU)**
  - **diagnostica dei moduli di I/O**



IMMAGINI DI PROCESSO

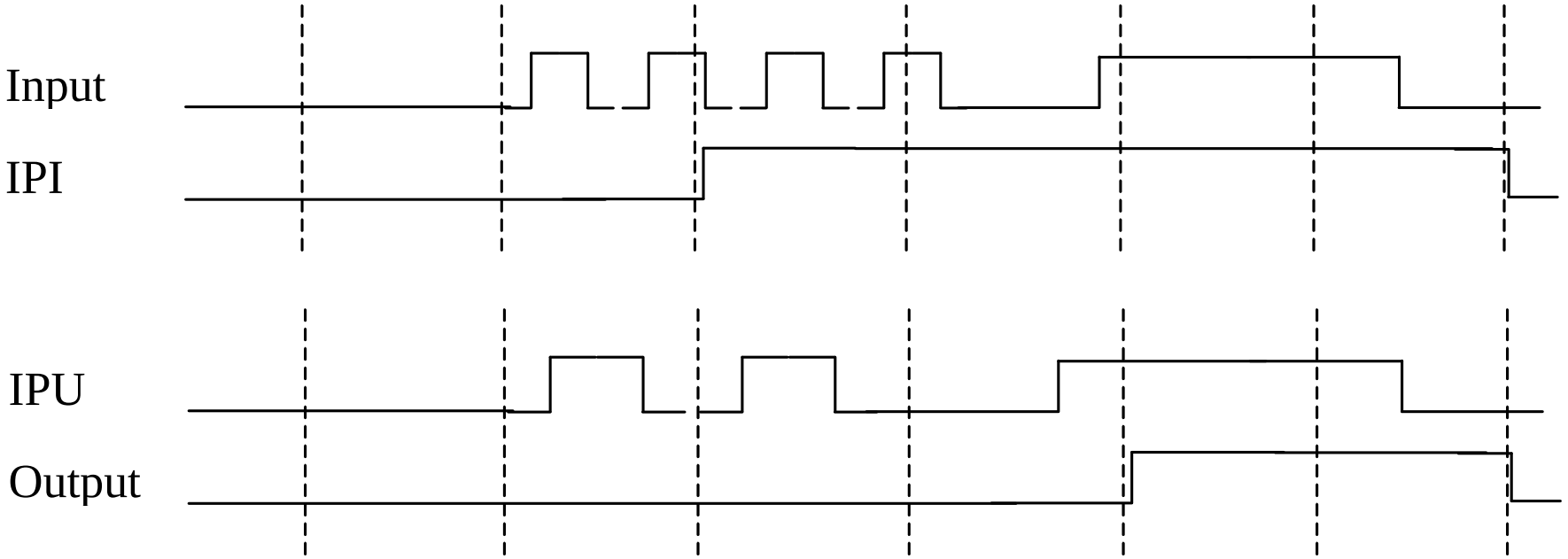
- **Immagini di processo degli ingressi (IPI):**
  - **variabili nelle quali viene memorizzato il valore degli ingressi logici all'inizio del ciclo**
  - **il programma applicativo (ciclo k) si svolge a ingressi congelati**
  - **è possibile accedere direttamente agli ingressi fisici senza modificare le IPI**

- **Immagini di processo delle uscite (IPU):**
  - **variabili nelle quali il programma applicativo scrive come se fossero le uscite logiche ma che vengono effettivamente scaricate sulle uscite alla fine del ciclo**
  - **l'ultima scrittura di un'uscita è la sola che ha effetto**
  - **sincronizzazione delle uscite fisiche**

- **Le immagini di processo non si applicano a:**
  - **I/O logico "veloce"**
  - **I/O legato a interrupt**
  - **I/O analogico**
  - **I/O logico di alcuni PLC (Es. SAIA)**

IMMAGINI DI PROCESSO

- **Immagini di processo:**
  - **Alcune variazioni degli ingressi possono andare perse (dipende da T_ciclo)**
  - **È consigliabile accedere alle uscite in un unico punto del programma**



PLC: MODALITA' DI FUNZIONAMENTO

- **Modalità STOP (/PROG,/TERM):**
  - **non esegue il programma applicativo**
  - **esegue funzioni del sistema operativo di dialogo con il sistema di programmazione (terminale o PC)**
  - **esegue funzioni di:**
    - **diagnostica**
    - **configurazione**
    - **programmazione del programma utente (download / upload)**

- **Modalità RUN:**
  - **esegue il programma applicativo sotto il controllo del sistema operativo residente (ciclo di funzionamento)**
  - **effettua autodiagnostica**
  - **se connesso al sistema di programmazione può:**
    - **visualizzare all'operatore lo stato di variabili**
    - **eseguire il programma applicativo solo per un certo numero di cicli**
    - **effettuare leggere modifiche al programma**

LINGUAGGI TRADIZIONALI

- **Ladder diagram (schemi a contatti, KOP per Siemens)**
  - **Orientato ai manutentori**
  - **Richiama gli schemi funzionali a relais**
  - **Istruzioni secondo il paradigma *if "input" then operation (elso do nothing)***

- **Instruction List (lista di istruzioni, AWL per Siemens)**
  - **Orientato al personale informatico**
  - **Pseudoassembler**

- **Function Block diagrams (Schemi funzionali, FUP per Siemens)**
  - **Orientato al personale elettronico**
  - **Segue lo standard ANSI/IEEE Std.91**

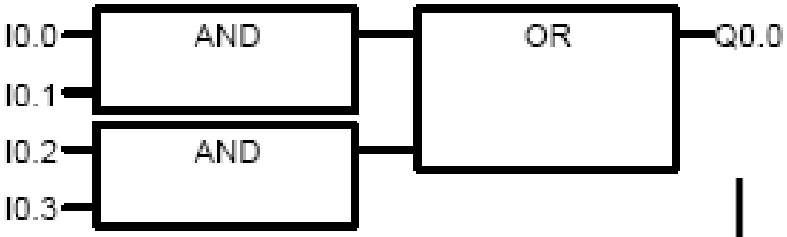
LINGUAGGI AWL, FUP, KOP

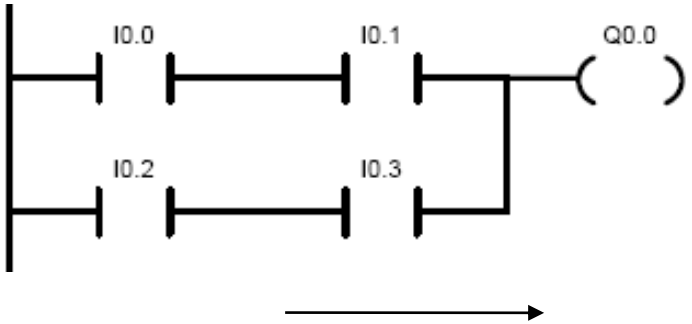
- **(Q0.0) = (I0.0)&(I0.1) + (I0.2)&(I0.3)** **nota: Q=uscita I=ingresso**

AWL **FUP** **KOP**
Lista di istruzioni **Schemi funzionali IEEE Std.91** **Ladder Diagram**

- **(A1.0) = (E0.0)&(E0.1) + (E0.2)&(E0.3)** **nota: A=uscita E=ingresso**

```
LD    I0.0
A     I0.1
LD    I0.2
A     I0.3
OLD
=     Q0.0
```





Ipotetico flusso di corrente

LINGUAGGIO LADDER

- **Dati (I dati possono essere condivisi da blocchi scritti con linguaggi diversi)**
  - **Il Ladder supporta tutti i tipi di dati semplici (bit, byte, word, integer, real,…)**
  - **Supporto dei dati relativi al tempo (Date&Time, Time)**
  - **Array difficilmente supportati (solo indirizzamento immediato o diretto)**

- **Istruzioni**
  - **Il codice sono sequenze di istruzioni con lo stesso costrutto (if "condition" then "operation"). NB: l'operatore bobina è l'eccezione che risponde al costrutto if "condition" then "set boolean operand" else "reset boolean operand"**
  - **Le condizioni sono test su singola variabile o contronti**
  - **Gli operatori comprendono varie tipologie (aritmetici, logici, trasferimento, abilitazione e configurazione oggetti quali timer, counter o blocchi di programma quali FC o FB, operatori di controllo di flusso del programma)**

- **Particolarità**
  - **Ordine delle istruzioni, data dependency, ecc. dipende dal ciclo di scansione e dal meccanismo delle immagini di processo e da meccanismi implementativi**

USO DEI DATI

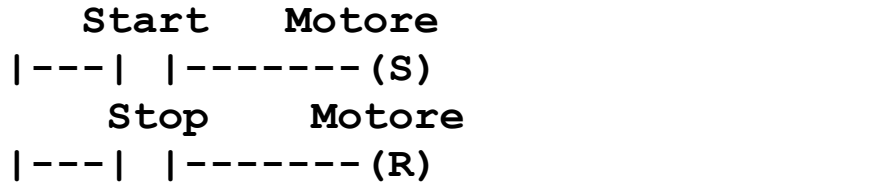
- **I dati (Immagini di I/O, variabili,..) sono allocati secondo l'indirizzo fisico (non deallocati come nella programmazione su PC)**
  - **La "fisicità" dell'indirizzo è un esigenza**
    - **Si pensi ad un PLC con 2 moduli identici di Ingressi Logici (32 Cad.): il cablatore utilizza il terzo contatto del secondo modulo (lo distingue perché lo vede) e lo collega ad un finecorsa; il programmatore sa che c'è un finecorsa ma come recupera l'informazione? I moduli hanno indirizzamento geografico (posizionale) fisso e analogamente è per le variabili**
    - **Il sistema operativo del PLC può non avere un MMU (o non potente e standard come gli OS)**
    - **Uno stesso indirizzo fisico può essere accessibile a due variabili logiche (Attenzione!)**
- **Allocare dati correlati contigui tra loro in un unico "blocco"**
  - **le sessioni di comunicazione trasferiscono tali "blocchi"**
  - **sovradimensionare i blocchi per tenere conto di eventuali modifiche**
- **I dati (Immagini di I/O, variabili,..) sono contenuti nella lista delle attribuzioni**
  - **dare nomi simbolici appropriati e utilizzare lo spazio "commento"**
  - **Utilizzare tutti i "tools" del sistema di sviluppo per una buona significatività e reperibilità del dato**

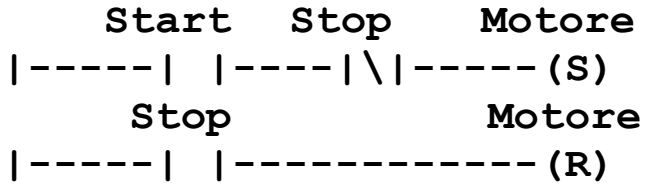
PLC: STRATEGIE DI PROGRAMMAZIONE

- **Ripartire il problema in più sottoproblemi**
- **Ogni segmento deve poter essere commentabile in modo chiaro e compiuto (strutturare il programma in sottoprogrammi e suddividere i segmenti complessi in più segmenti semplici facendo uso di merker come variabili intermedie)**
- **Seguire le logiche in sicurezza, ridondando gli interblocchi su più segmenti**
  - **Esempio: selettore RICETTE 1,2,3 (R1,R2,R3)**
    - **Senza interblocchi: Se R1 allora…**
    - **Con inerblocchi: Se R1&!R2&!R3 allora…**
- **Considerare condizioni di guasto tipico (Es. strappo cavi)**
- **Raggruppare le condizioni relative ad un certo stato di una o più uscite secondo la logica del "minimo impatto delle modifiche" (se si aggiungesse una condizione si dovrebbe modificare il programma nel minimo numero di punti)**
- **Uscite e merker (variabili) devono essere assegnati una sola volta all'interno di ogni ciclo di scansione**

PLC: il concetto di interblocco

- **Il semplice problema del Set Reset di un motore**

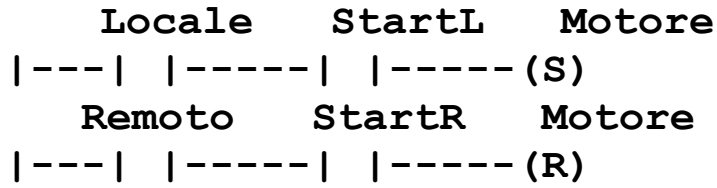


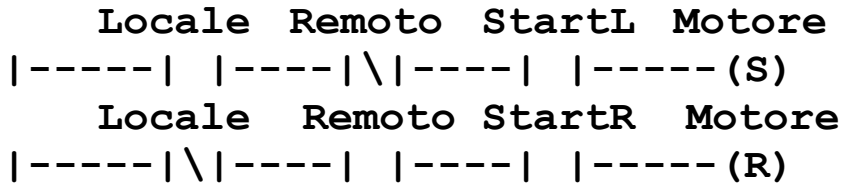


L'ordine dei segmenti Influisce sulle funzionalità

L'ordine dei segmenti non influisce -> più affidabile!

- **L'interblocco totale: il caso del selettore Locale Remoto**



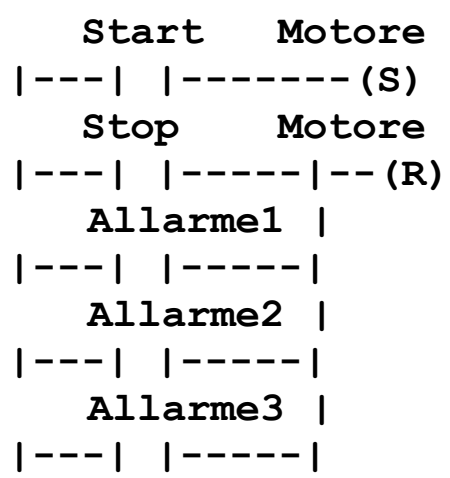


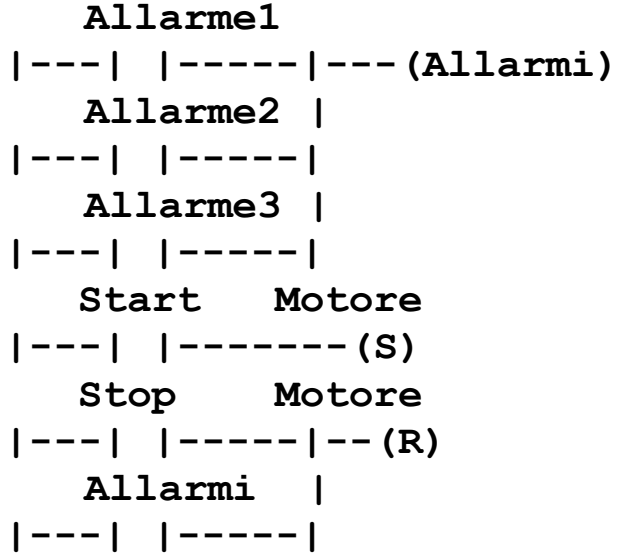
L'ordine dei segmenti Influisce sulle funzionalità

L'ordine dei segmenti non influisce -> più affidabile!

PLC: il concetto di variabile intermedia (merker)*

- **Merker = equivalente del relè ausiliario nei circuiti elettromeccanici**
  (Più semplicemente… una variabile di memoria)

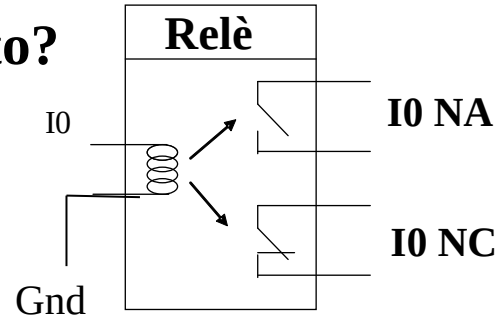


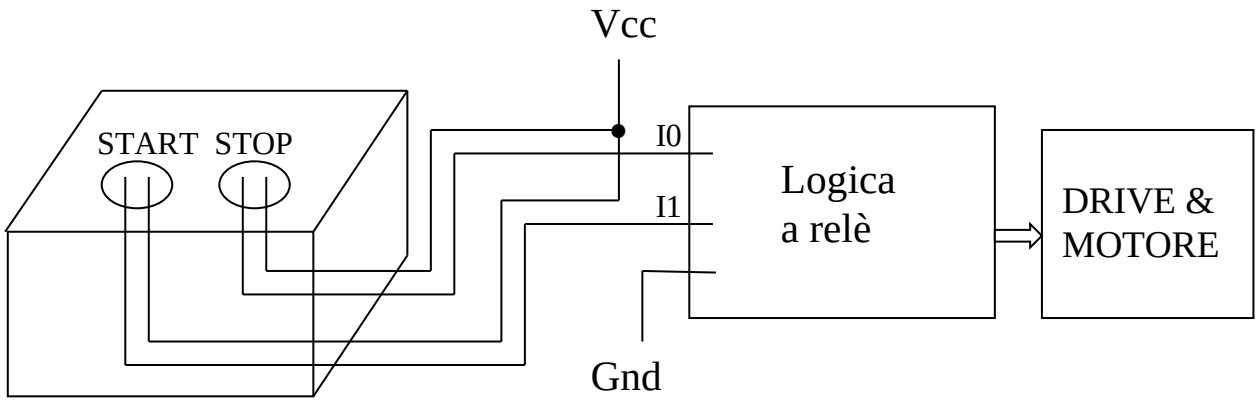


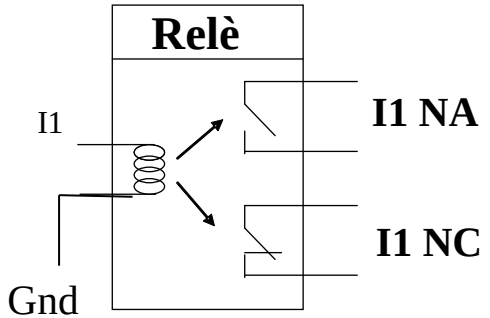
- **Il merker Allarmi (es. M2.3) aumenta la leggibilità**
- **I merker sono organizzati a byte (stringhe di 8 bit)**
  M<indirizzo byte>.<indirizzo bit> M2.3 = bit 3 del byte M2

Contatti NA e NC

- **Considerare condizioni di guasto tipico (Es. strappo cavi, sensore non alimentato)**
- **Cosa succede se strappo i cavi? Come si comporta il sistema?**
  - **Contatti NA per i segnali abilitanti (Es. START, Ripristino_Allarme,…)**
  - **Contatti NC per i segnali inibenti (Stop, Allarme,…)**
- **E se mi cambiano un NA con un NC devo ricablare tutto?**
- **Si usano relè di appoggio così da realizzare logiche che vedono sempre ingressi NA (E' il principio dei PLC)**

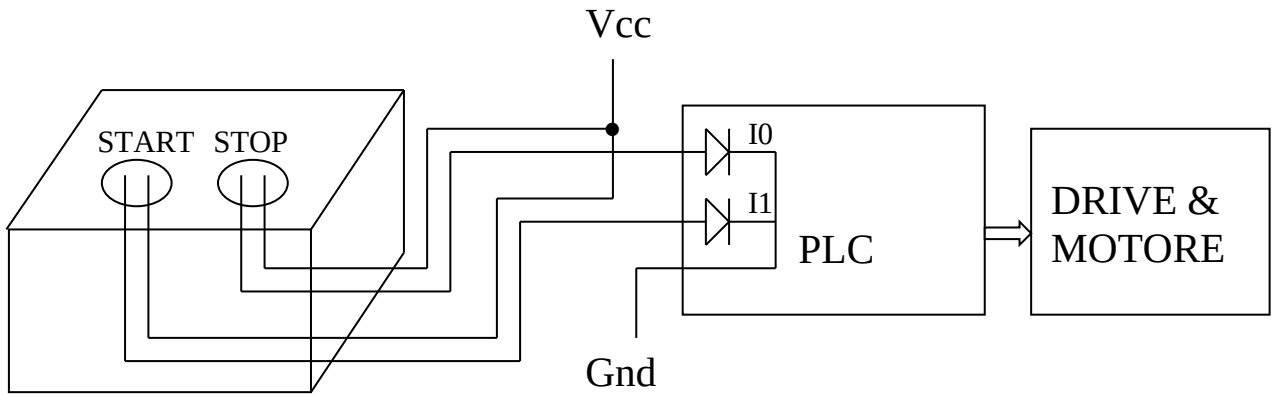






PLC: strategie di programmazione

- **Considerare guasti tipici, sostituzione di componenti, modifiche**
  - **Contatti NA per i segnali abilitanti (Es. START, Ripristino)**
  - **Contatti NC per i segnali inibenti (Es. STOP, Allarme)**
  - **Si usavano relè di appoggio per gestire sempre logiche in NA**
  - **L'ingresso del PLC è come un relè di appoggio**
  - **Se l'ingresso mi arriva NA l'interrogazione "è attivo?" corrisponde a --| |--**
  - **Se l'ingresso mi arriva NC l'interrogazione "è attivo?" corrisponde a --|\|--**



PLC: strategie di programmazione *

- **Il programma deve essere leggibile**
- **Vale la logica del "minimo impatto delle modifiche"**
  - **Un programma è fatto meglio di un altro se una modifica sulle funzionalità implica un numero minore di modifiche al programma**

```
   Start  Stop  Motore
|---| |----|\|---(S)
   Stop         Motore
|---| |----------(R)
```

Ingressi NA

```
   Start  Stop  Motore
|---| |----| |---(S)
   Stop         Motore
|---|\|----------(R)
```

Ingresso Stop NC

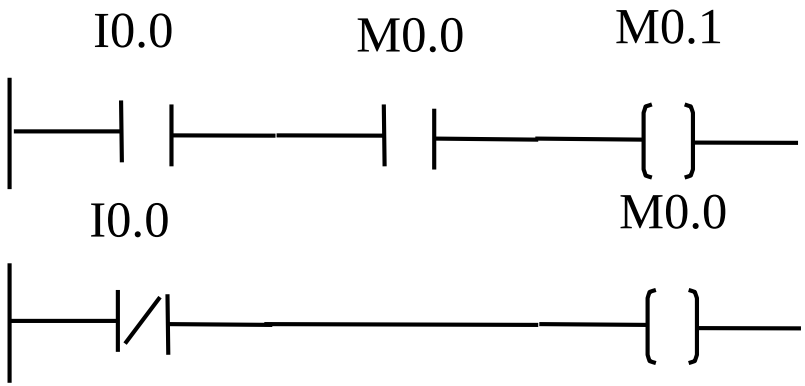
"Se c'è Start e Stop avvia Motore??"

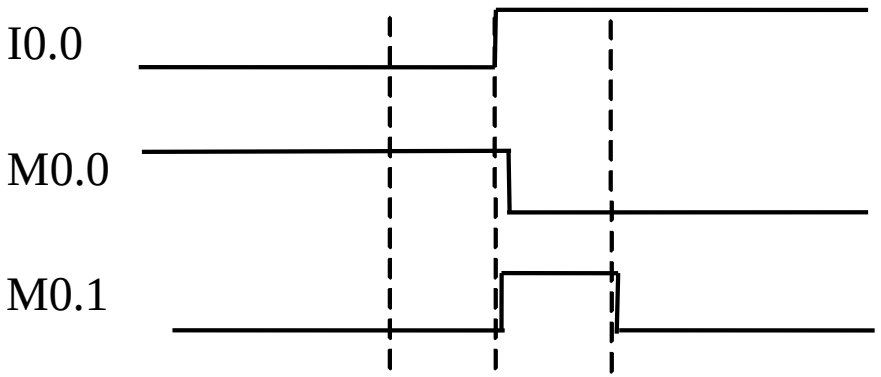
```
   Start  Start_M
|---| |-----( )
   Stop   Stop_M
|---|\|-----( )

   Start_M Stop_M  Motore
|---| |-----|\|-----(S)
   Stop_M           Motore
|---| |-------------(R)
```

Leggibile e con minimo impatto delle modifiche

LADDER: ORDINE DEI SEGMENTI

- **Si deve porre grande attenzione nell'ordine dei segmenti**
- **Se si prova a invertire la posizione dei due ladder, sul segnale M0.1 non si ha più la rilevazione del fronte di salita di I0.0**





Infatti, perché si abbia un impulso su M0.1, M0.0 deve essere ritardato rispetto a I0.0
NOTA: su molti PLC esiste l'operatore "rilevatore di fronte"

GESTIONE DI MOTORI: STRATEGIE

- **MODO JOG**

  Quando il motore viene cablato (o in caso di guasto) deve esserci localmente una modalità di prova

  - **Pulsante JOG -> il motore va (a velocità limitata) solo mentre il pulsante è premuto**

- **START-STOP**

  Esiste una modalità START-STOP (il pulsante avvia o arresta) su quadro (locale) e su pulpito di comando (remoto)

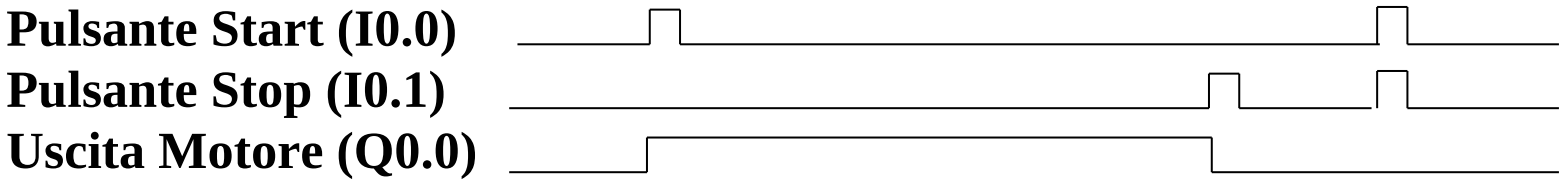
  - **Può esistere una modalità START-STOP da remoto**

- **AUTOMATICO**

  Ad impianto avviato, il motore è normalmente gestito in automatico

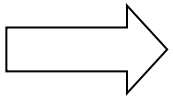
- **….**

STRUTTURE AD AUTORITENUTA: logiche avvio/arresto



- Deriva dalla logica a relè (autoritenuta Reset-prevalente)
- AUTORITENUTA Reset prevalente: Out = (Set+Out)&!Reset
- AUTORITENUTA Set prevalente: Out = Set+(Out&!Reset)
- E se il pulsante di Start si incastra?

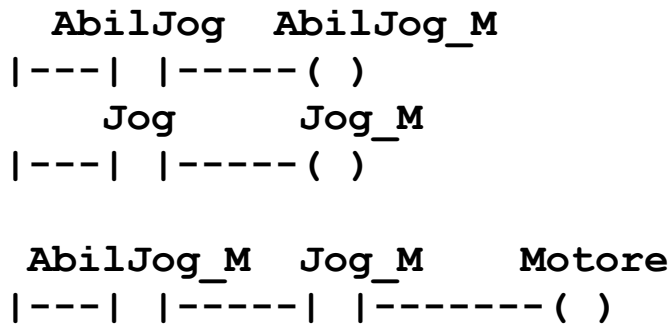
```
   Start  Start_M
|---| |-----( )
   Stop   Stop_M
|---|\|-----( )

   Start_M Stop_M  Motore
|---| |-----|\|-----(S)
   Stop_M          Motore
|---| |-------------(R)
```



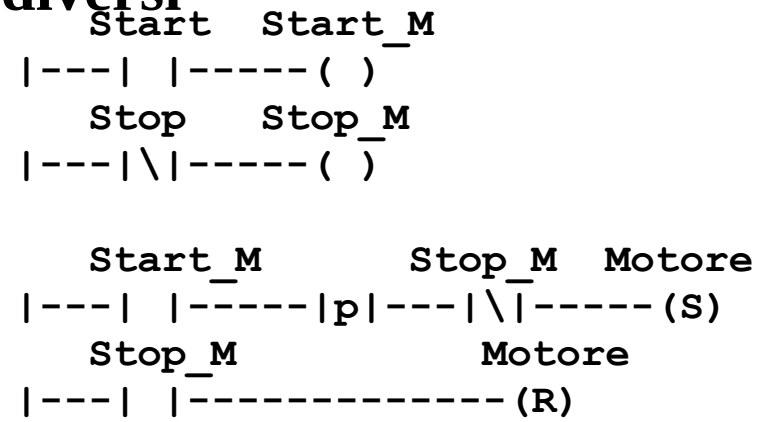
```
   Start  Start_M
|---| |-----( )
   Stop   Stop_M
|---|\|-----( )

   Start_M       Stop_M  Motore
|---| |-----|p|---|\|-----(S)
   Stop_M          Motore
|---| |-------------(R)
```

GESTIONE DI MOTORI: STRATEGIE

- **Il motore può essere gestito in modi diversi**

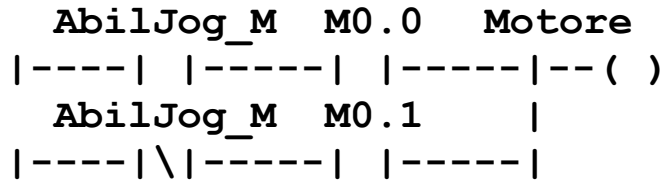


M0.0



M0.1

- **Presi singolarmente i due programmi funzionano, messi insieme no**
- **Istruzioni di Set-Reset e di Assegnazione non possono coesistere**
- **E' meglio creare delle memorie di appoggio (motori virtuali M0.0 e M0.1)**



GESTIONE DI MOTORI: STRATEGIE *

- **Progetto del programma = Segmentazione del programma in più parti (stazioni)**
  (Stazione = sottoparte omogenea e ben identificabile dell'impianto)
- **In ogni stazione c'è tipicamente un motore principale, ma possono esserci dei motori ausiliari (ventilatori, condizionatori,…)**
- **Il motore principale può essere gestito in più modi (locale, pulpito, remoto, automatico,…)**
- **Per ogni stazione vengono definite:**
  - **le procedure di inizializzazione**
  - **il programma, ripartendo tra sequenze "in manuale" e "in automatico"**
    (l'insieme dei comandi manuali è più vasto di quelli automatici)
  - **Cronologicamente si affrontano:**
    - **le sequenze "in manuale" (da locale, da pulpito,…in logica di sicurezza)**
    - **le sequenze "in automatico"**
      (ridondanza e timeout sui sensori che determinano la transizione)
    - **strutture dati per la tracciabilità e strutture dati condivise con SCADA**

GESTIONE DI MOTORI E LOGICHE DI ALLARME

❑Gestione logiche di allarme:

- **Segnalazione (lampade, sirene,…) sulla base dell'intervento di alcuni allarmi**
- **Allarmi a soglia**
  - **Valore sotto soglia di attenzione: lampada spenta**
  - **Valore sopra soglia di attenzione ma sotto soglia di allarme: lampada che lampeggia lenta**
  - **Valore sopra soglia di allarme: lampada che lampeggia veloce**
  - **Valore sopra soglia di allarme continuativamente da più di un tempo T: lampada accesa fissa. Tale condizione viene memorizzata**

❑Tipologie di allarme con memorizzazione:

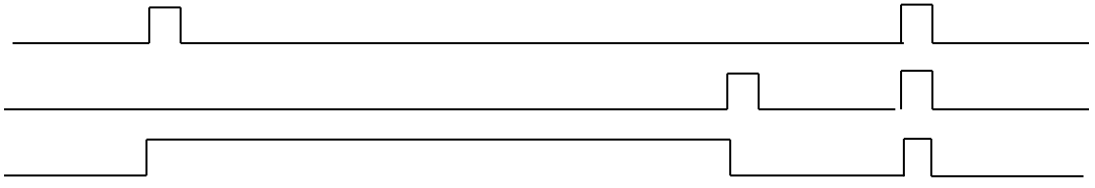
- **Allarme con memorizzazione: serve un reset generale**
- **Allarme con memorizzazione e riconoscimento: c'è un comando di Ripristino che sblocca l'allarme (vince il Ripristino)**
- **Allarme con memorizzazione, riconoscimento e rientro: c'è un comando di Ripristino che sblocca l'allarme (vince l'allarme)**

❑Funzioni correlate alla gestione di allarmi

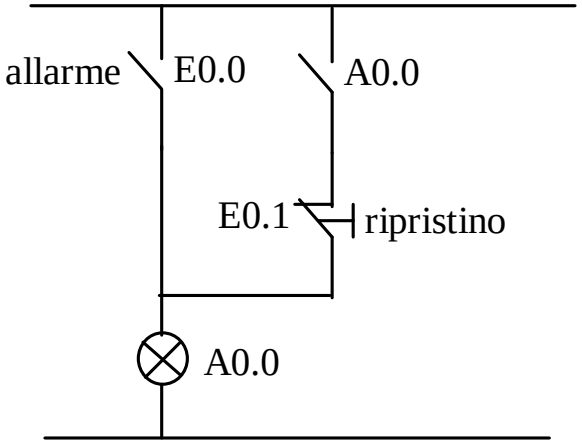
- **Riconoscimento del primo allarme intervenuto**
- **Priorità e riconoscimento degli allarmi**

STRUTTURE AD AUTORITENUTA: logiche di allarme *

Condizione di ALLARME E0.0
Pulsante di RIPRISTINO E0.1
Segnalazione di ALLARME A0.0



- **Memorizzazione dei pulsanti**
- **ALLARME prevalente**
- **ALLARME: contatti NC**
- **RIPRISTINO: contatti NA**

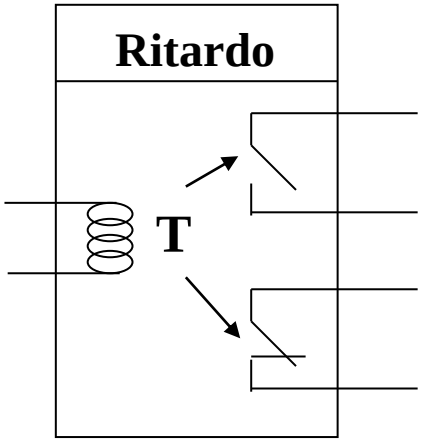


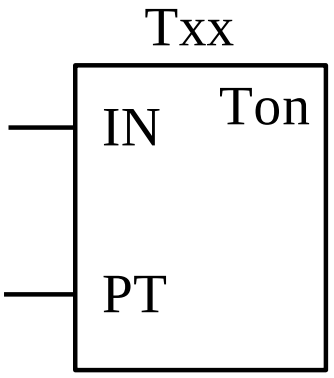

```
Ripristino  Ripristino_M
|-----| |----------( )
     Allarme   Allarme_M
|-----|\|----------( )

   Allarme_M     Lampada
|---| |------------(S)
   Ripristino_M  Lampada
|---| |------------(R)
```

Nota: il circuito usa relè d'appoggio
Nota: il programma è il migliore? Interblocchi? E se il pulsante si incastra?
Nota: sapremmo unirlo alla logica di Start/Stop?

USO DEI TIMER

- **Tutte queste funzioni si possono realizzare mediante funzioni logiche e un modulo "ritardo"**
- **Il PLC implementa un solo modulo SW "Timer" che agisce come ritardo**
- **Come esprimo il ritardo PT?**
- **Esiste uno standard IEC61131 -> T#0s**
- **L'ingresso IN corrisponde alla bobina del temporizzatore. Nel rele' (vecchi PLC)**
  - **se IN=falso ->Txx=0**
  - **se IN=vero ->Txx si incrementa (con saturazione)**
  - **se Txx>PT allora Txx(contatto)=vero**





- **Nel PLC std IEC e' il fronte di salita di IN che avvia il timer**

IL TIPO DI DATO TIME

- **Il contenuto di un operando del tipo di dati TIME viene interpretato in ms. La rappresentazione comprende le indicazioni di giorni (d), ore (h), minuti (m), secondi (s) e millisecondi (ms).**

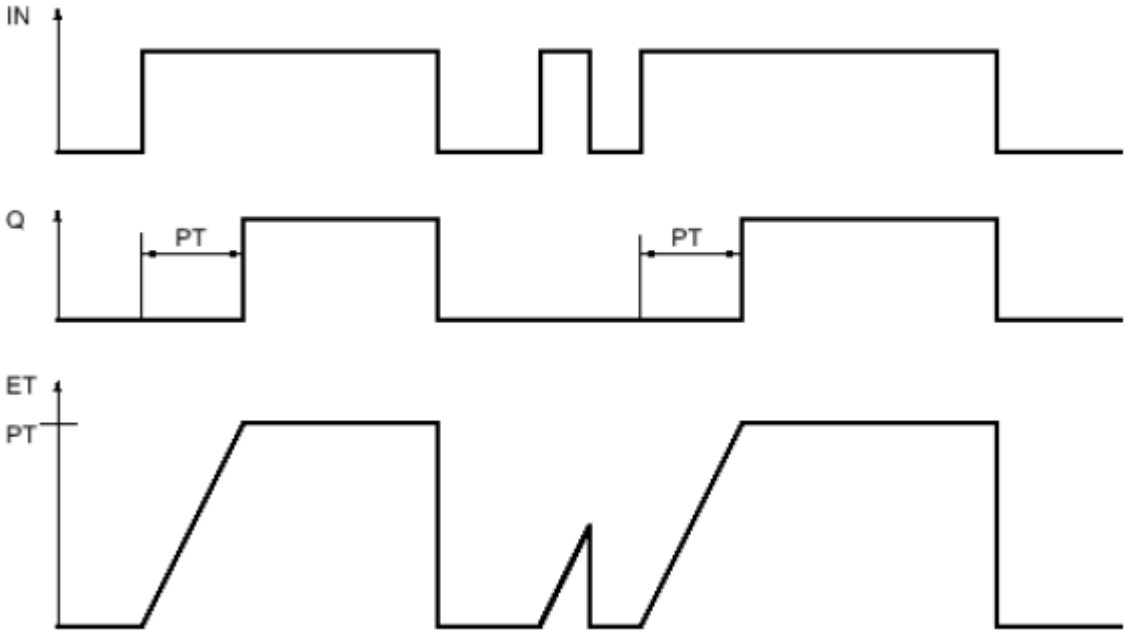
- **TIME viene espresso con 32 bit (integer, unità ms) nel formato con segno da T#-24d20h31m23s648ms a T#+24d20h31m23s647ms**

- **Esempi di rappresentazione:**
  - **T#10d20h30m20s630ms,**
  - **TIME#10d20h30m20s630ms**
  - **10d20h30m20s630ms**
  - **T#5h10s (Non è necessario indicare tutte le unità di tempo)**
  - **Non si devono eccedere i limiti (23 h, 59 m, 59 s o 999 ms).**

- **Un timer IEC è un oggetto con ingresso IN e uscita OUT booleane, il cui valore corrente ET e la cui costante di tempo PT sono espresso come TIME**

TIMER RITARDO ALL'INSERZIONE (es. filtro di abilitazioni)

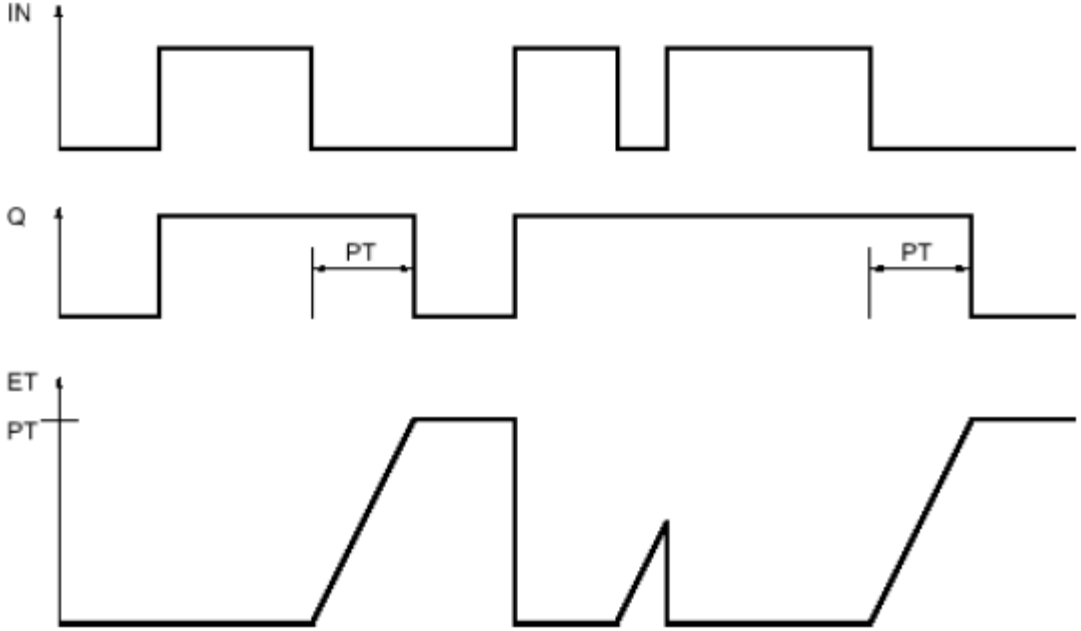
- **Se l'informazione presente su IN ha un fronte di salita, si attende PT quindi l'uscita va a 1 fino a quando IN non ha una commutazione a zero. La commutazione a zero di IN durante PT azzera il Timer**
  - **Istruzione "Avvia ritardo all'inserzione" –(TON)-**



Immagini prese da manuale Siemens

TIMER RITARDO DISINSERZIONE (es. “buchi di rete”)

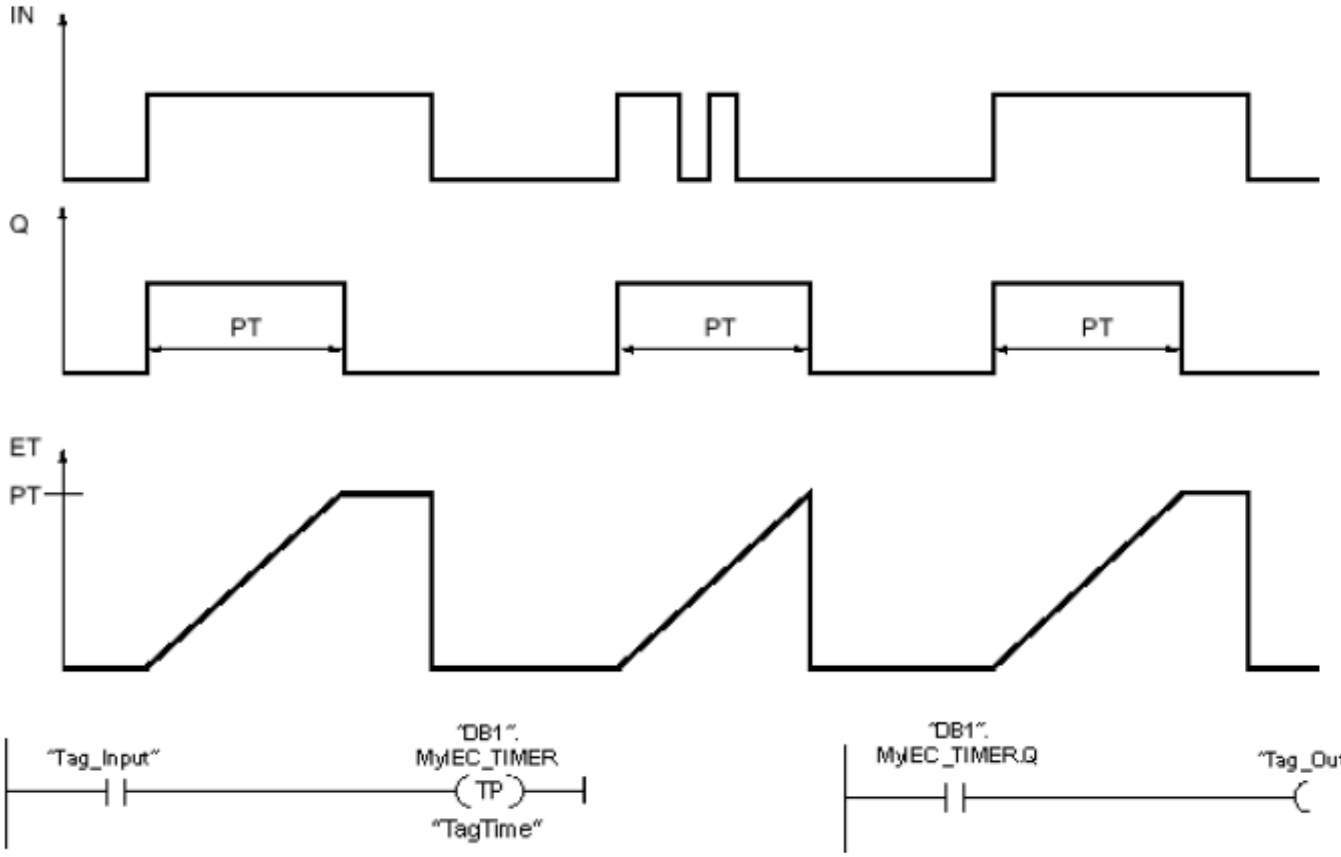
- **Se l’informazione presente su IN ha un fronte di salita l’uscita viene posta a 1; al fronte di discesa di IN, l’uscita rimane a 1 e vi rimane per un tempo PT quindi viene posta a 0. Un nuovo fronte di salita di IN fa ripartire il meccanismo**
  - **Istruzione “Avvia ritardo alla disinserzione” –(TOF)-**



Immagini prese da manuale Siemens

TIMER GENERAZIONE DI IMPULSI (es. timeout motore nastro)

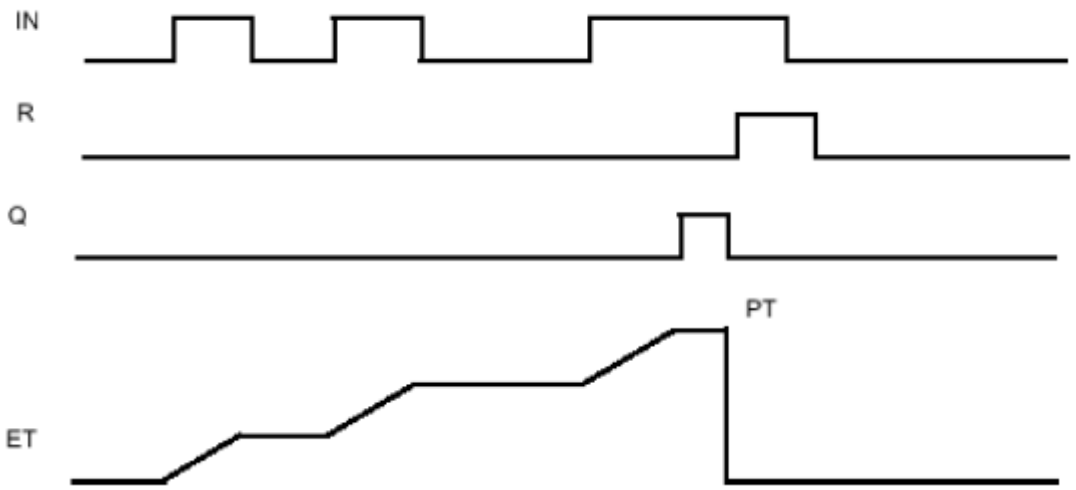
- **Se l'informazione presente su IN ha un fronte di salita, l'uscita viene impostata per una durata programmata PT indipendentemente da come evolve IN**
  - **Istruzione "Avvia temporizzazione quale impulso" –(TP)-**



Immagini prese da manuale Siemens

TIMER ACCUMULATORE TEMPORALE (es. durata utensile)

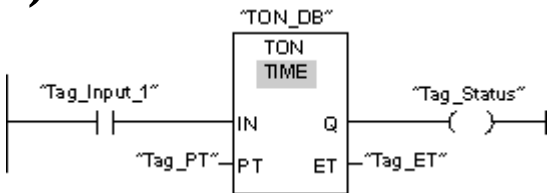
- **Se l'informazione presente su IN ha un fronte di salita, ET accumula i valori temporali in corrispondenza di IN=1 e l'uscita viene posta a 1 allo scadere di PT. Il segnale di reset R (booleano) resetta ET e Q**
  - **Istruzione "Avvia accumulatore temporale" –(TONR)-**

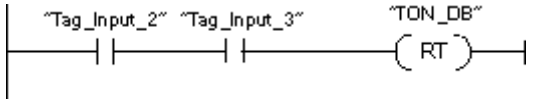


Immagini prese da manuale Siemens

OSSERVAZIONI SUI TIMER

- **I timer (e in particolare i timer con memoria) possono essere resettati mediante la bobina –(RT)-**





Immagini prese da manuale Siemens

- **La costante di tempo puo' essere sovrascritta –(PT)-**

- **I timer TON e TONR sono i piu' utilizzati**

- **Esercizio: il ritardo all'inserzione dei rele' corrisponde esattamente al timer TON?**
  - **Si. La rilevazione del fronte di salita di In di TON equivale alla rilevazione di In in quanto il fronte di salita attiva e il fronte di discesa disattiva (definizione dell'assegnazione)**

- **Esercizio: il ritardo alla disinserzione dei rele' corrisponde esattamente al timer TOF?**
  - **Si. Il fronte di discesa fa partire il timer e il fronte di salita lo resetta**

TIMER: RIASSUNTO

- **I timer dei PLC sono oggetti software (ce ne sono tanti, inutile risparmiare)**

- **Tutti i timer si attivano sul fronte di salita di In e si disattivano con il reset o con condizioni specifiche (*). L'uscita Out è a 0 se il timer non è attivo**

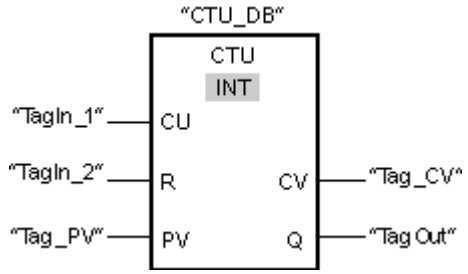
Tipo Timer	Il conteggio inizia	Il conteggio si ferma	Il conteggio si azzera (*)	Out è a 1 se timer attivo e	Il timer si disattiva
TON	Fronte salita In		Fronte discesa In	Se ET >= PT	Fronte discesa In
TOF	Fronte discesa In		Fronte salita In	Se ET < PT	Solo al reset
TP	Fronte salita In		Fronte discesa In & ET>=PT	Se ET < PT	Fronte discesa In & ET>=PT
TONR	Fronte salita In	Fronte discesa In	Solo al reset	Se ET >= PT	Solo al reset

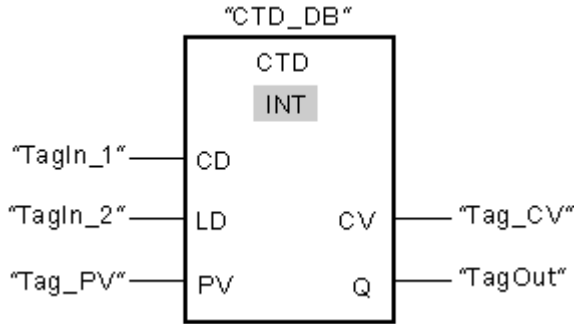
IL CONTATORE

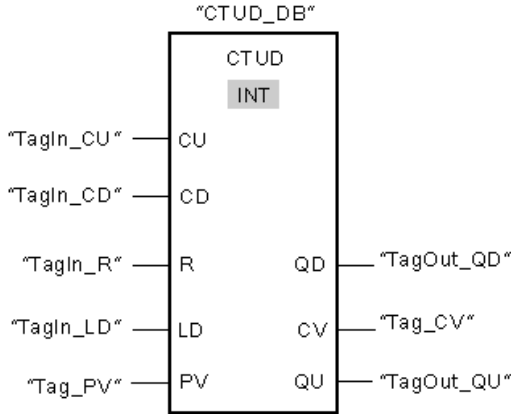
- **Il temporizzatore conta il tempo, il contatore conta eventi**

Immagini prese da manuale Siemens

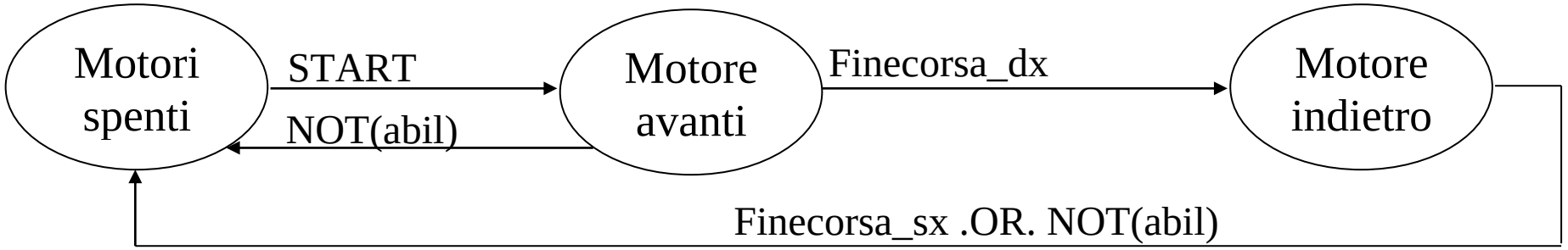
- **Esistono tre tipi di contatori: abanti, indietro, up/down (Es. CTU –avanti-)**
  - **Evento = fronte di salita su CU; PV = soglia del contatore (max INT = 32767)**
  - **Il valore corrente CV del contatore si incrementa fino a maxINT e si blocca**
  - **Applicando 1 all'ingresso R di reset il valore CV si azzera inibendo CU**
  - **L'uscita binaria Q indica se PV>=CV**
  - **Contatore down (CTD): CV si decrementa ad ogni evento fino a raggiungere la soglia (=0) (minINT=-32768). Applicando 1 su LD si carica CV al valore PV**
  - **Contatore up/down (CTUD): +1-1=0**







FUNZIONI SEQUENZE AUTOMATICHE



- **Metodo dei flag (implementabili con i merker):**
  - **flag1 (stato di attesa dello START)**
  - **flag2 (stato di motore_avanti e attesa del finecorsa_dx)**
  - **flag3 (stato di motore_indietro e attesa del finecorsa_sx)**
- **Si definiscono delle reti combinatorie per il passaggio tra gli stati**
  - **Se flag1 .AND. START .AND. abil -> res(flag1),set(flag2),res(flag3)**
  - **Se flag2 .AND. finecorsa_dx -> res(flag2), set(flag3), res(flag1)**
  - **Se flag3 .AND. finecorsa_sx -> res(flag3), set(flag1), res(flag2)**
  - **Se .NOT. abil -> res(flag2), res(flag3), set(flag1)**
  - **Se flag1 -> spegni motori**
  - **Se flag2 -> motore avanti**
  - **Se flag3 -> motore indietro**

PASSI

TRANSIZIONI

- **Nota: Potrebbe esserci più di una transizione alla volta, creando transizioni che non ci sono**

SEQUENZE AUTOMATICHE (2)

- **START:motore_avanti fino al finecorsa_dx, poi motore indietro fino al finecorsa_sx**
  - **Metodo dei flag (implementabili con i merker):**
    - **flag1 (stato di attesa dello START)**
    - **flag2 (stato di motore_avanti e attesa del finecorsa_dx)**
    - **flag3 (stato di motore_indietro e attesa del finecorsa_sx)**
  - **Metodo del semaforo (flag0):**

    Si definisce un flag (flag0) che indica un passaggio di stato
    - **Set flag0 // inizialmente il semaforo è verde**

    Transizioni
    - **Se flag0.AND.flag1.AND.START.AND.abil -> res(flag0,flag1,flag3), set(flag2)**
    - **Se flag0.AND.flag2.AND.finecorsa_dx -> res(flag0,flag1,flag2), set(flag3)**
    - **Se flag0.AND.flag3.AND.finecorsa_sx -> res(flag0,flag2,flag3), set(flag1)**
    - **Se .NOT. abil -> res(flag0,1,2,3)**

    Passi
    - **Se flag1 -> spegni motori**
    - **Se flag2 -> motore avanti**
    - **Se flag3 -> motore indietro**

SEQUENZE AUTOMATICHE (3)

- **START:motore_avanti fino al finecorsa_dx, poi motore indietro fino al finecorsa_sx**
  - **Metodo delle variabili (implementabili con i merker), simile al sistema IPI e IPU:**
    - **Stato (variabile che può assumere valori 0, 1, 2). Non varia per tutte le transizioni**
    - **Stato_futuro (variabile "immagine di stato" che aggiorno solo alla fine)**
    - **In questo modo eseguo solo una transizione per ogni ciclo**

  - **Metodo della variabile di appoggio (Stato_futuro):**

    Transizioni
    - **Se (Stato=0).AND.START.AND.abil -> Stato_futuro = 1**
    - **Se (Stato=1).AND.finecorsa_dx.AND.abil -> Stato_futuro = 2**
    - **Se (Stato=2).AND.finecorsa_sx.AND.abil -> Stato_futuro = 0**
    - **Se .NOT. abil -> Stato_futuro = 0 (transizione più prioritaria)**
    - **Stato = Stato_futuro**

    Passi
    - **Se (Stato=0) -> spegni motori**
    - **Se (Stato=1) -> motore avanti**
    - **Se (Stato=2) -> motore indietro**

PROGETTO DELLE SEQUENZE AUTOMATICHE

- **La vera fase di progetto è il disegno della sequenza e l'identificazione degli STATI (passi e relative azioni) e delle TRANSIZIONI**
  - **Due tipi di transizioni:**
    - **Condizionate: hanno uno stato di partenza e uno stato di arrivo**
    - **Incondizionate: hanno solo lo stato di arrivo e normalmente sono prioritarie rispetto a quelle condizionate (sono gli ultimi segmenti della parte "transizioni")**

- **Fasi realizzative a valle del progetto della sequenza in termini di passi e transizioni**

1. **Dichiarare le variabili che servono**
2. **Predisporre il programma con una parte di TRANSIZIONI, il passaggio di stato (allineamento tra Stato e Stato_futuro), e una parte di PASSI**
3. **Numerare gli N STATI e realizzare la relativa parte di programma instanziando gli N segmenti necessari e popolandoli con le azioni da fare per ogni stato**
4. **Numerare le M TRANSIZIONI e realizzare la relativa parte di programma instanziando gli N segmenti necessari e popolandoli con le interrogazioni necessarie per lo svolgimento della transizione**

METODOLOGIE DI PROGETTAZIONE: fasi 1 e 2*

- **1) Specifica dei requisiti (produzione di documenti)**
  - **Layout dell'impianto**
  - **Lista delle motorizzazioni**
  - **Schemi pneumatici e oleodinamici,..**

  ⟹ **lista sensori/attuatori (I/O)**
  lista moduli funzionali

  - **Definizione delle condizioni ambientali (grado IP, T, Rh, rischi,..)**

- **2) Progetto**
  - **Stesura topologico dell'impianto (def. di sottoparti funzionalmente omogenee)**
    Per ogni sottoparte vanno definiti:
    - **Le funzionalità**
    - **gli I/O**
    - **i moduli funzionali**

    ⟹ **scorte, margini** ⟹ **dimensionamento PLC**
  - **Layout dell'impianto -> Layout elettrico (fondamentale per i cablaggi)**
  - **Definizione dei PLC e della relativa configurazione (CPU, I/O, moduli,…)**

METODOLOGIE DI PROGETTAZIONE: fasi 3 e 4*

- **3) Progetto**
  - **Stesura dello schema elettrico**
    - **Progettazione della parte di potenza (avviamento motori, azionamenti,..)**
    - **Progettazione ausiliari (non passano dal PLC – funghi di emergenza -)**
    - **Progettazione della parte di comando (PLC, tabella I/O)**

  - **Lo schema elettrico, supportato da software specifici, consta di**
    - **rappresentazione grafica-funzionale dei dispositivi e dei collegamenti**
    - **numerazione e identificazione di dispositivi e collegamenti**
    - **gestione riferimenti incrociati ad altri schemi**
    - **distinta dei materiali**
    - **descrizioni in diverse lingue**

- **4) Progetto**
  - **Codifica ciclo PLC**
    - **Lista delle attribuzioni o tabella I/O -> tabella dei simboli**
    - **Progetto del programma**

METODOLOGIE DI PROGETTAZIONE: progetto del programma*

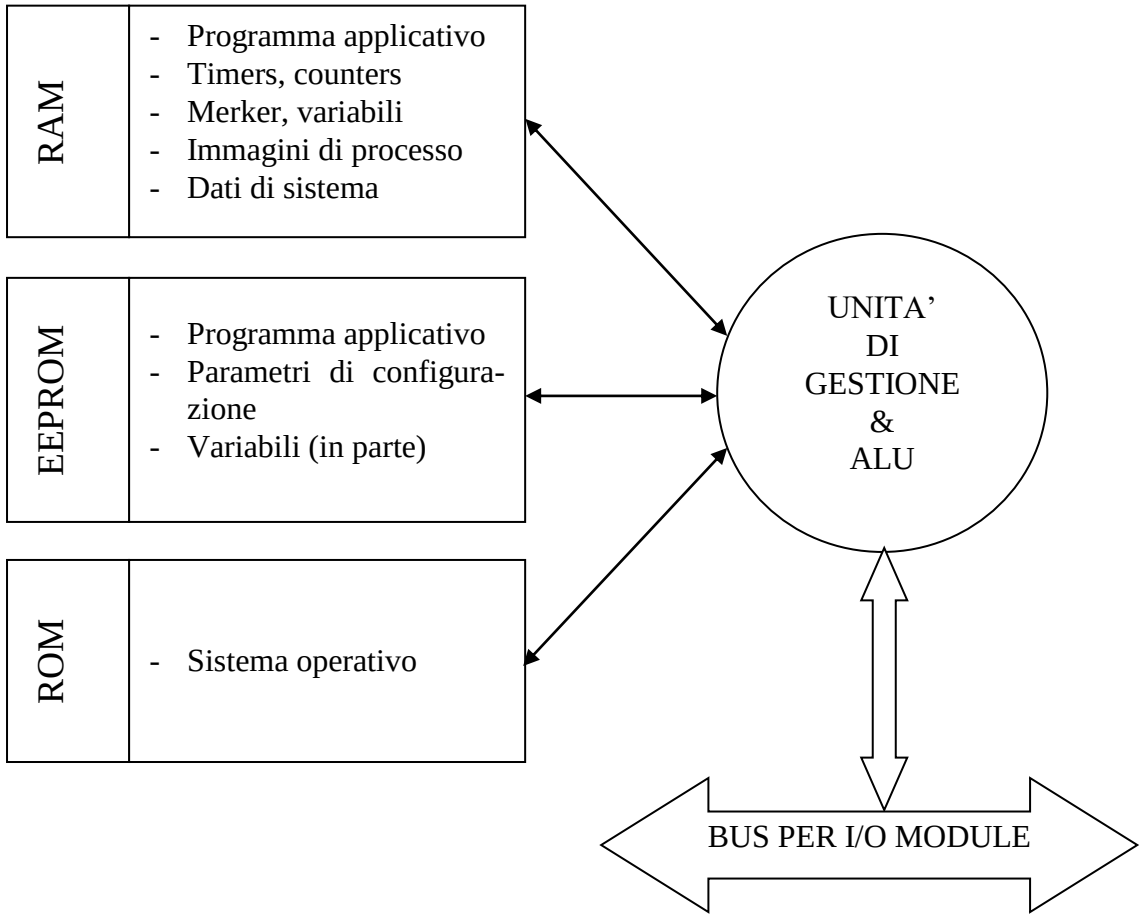
- **5) Implementazione**
  - **Segmentazione del programma in più parti (stazioni)**
  - **Stazione = sottoparte omogenea e ben identificabile dell'impianto che risulta il più possibile funzionalmente disaccoppiata dal resto**

  - **Per ogni stazione vengono definite:**
    - **le procedure di inizializzazione**
    - **il programma, ripartendo tra sequenze "in manuale" e "in automatico" (l'insieme dei comandi manuali è più vasto di quelli automatici)**

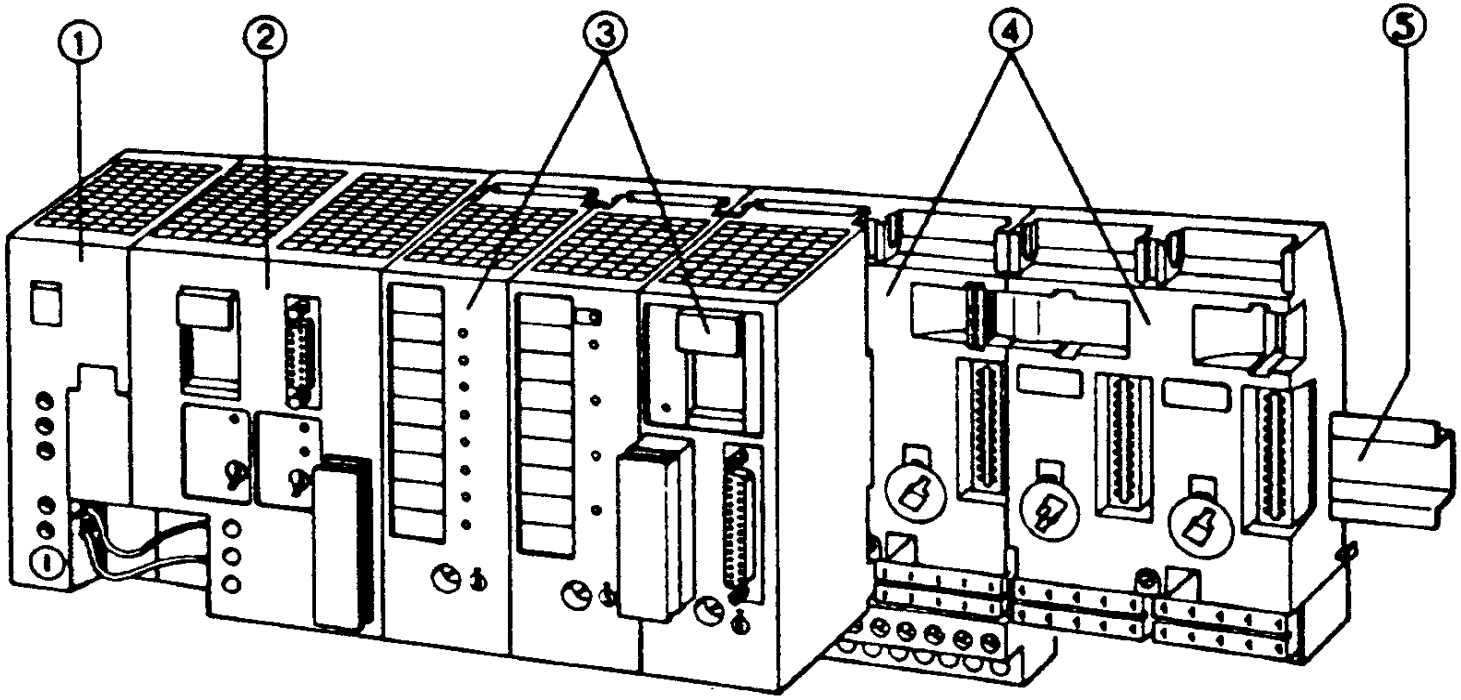
  - **Cronologicamente si affrontano:**
    - **le sequenze "in manuale" (da pulpito o da pannello di comando via SCADA, in logica di sicurezza)**
    - **le sequenze "in automatico" (ridondanza e timeout sui sensori che determinano la transizione)**
    - **strutture dati per la tracciabilità dei materiali**
    - **strutture dati condivise con SCADA**

IL MODULO CPU

- **ALU (Arithmetic Logic Unit)**
- **I/O bus generator**
- **Memorie**
- **Alimentazione**
- **I/O locale**
- **Interfacce di comunicazione**
- **Interfacce diagnostiche**



PLC: ELEMENTI INDISPENSABILI

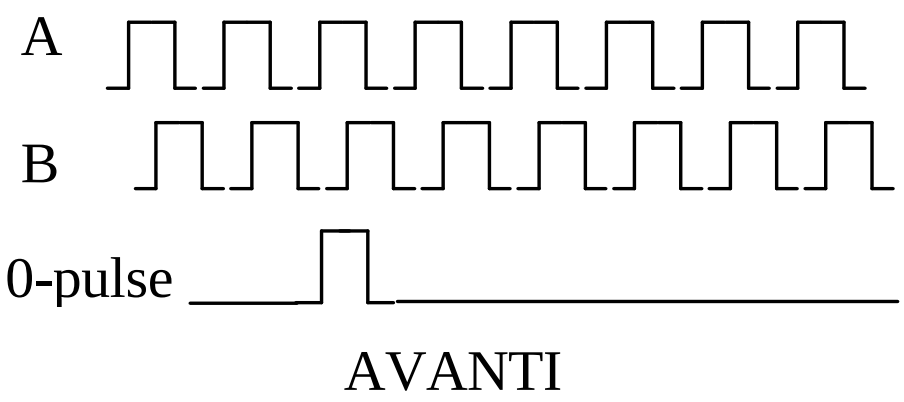


1) Alimentatore
2) CPU
3) Unita' periferiche (I/O logico, I/O analogico, controllo assi,…)
4) Moduli di bus con punto di attacco (indirizzamento geografico –posizionale- dei moduli)
5) Guide profilate normalizzate DIN

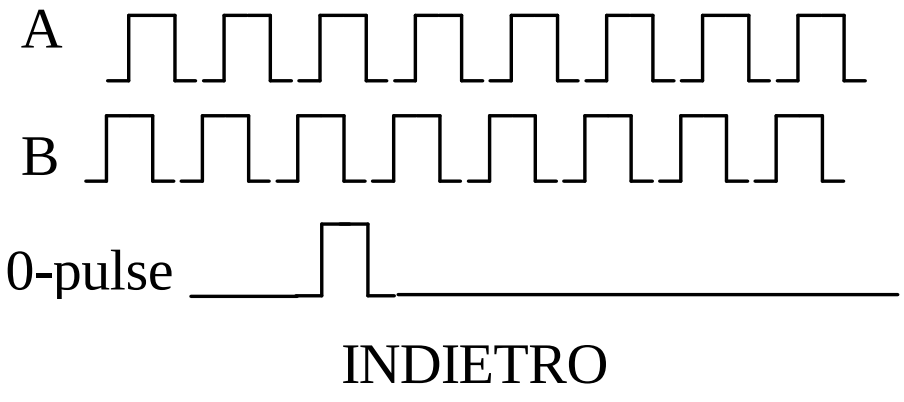
ENCODER OTTICI INCREMENTALI

- **Misure di posizione e velocità**

- **Dispositivi che forniscono un segnale ad onda quadra con *m* impulsi per ogni giro**
  - **velocità angolare = *w* -> frequenza di uscita = *m*w***
  - **3 segnali di uscita:**
    - **due segnali in quadratura per posizione, velocità e verso di rotazione**
    - **un impulso di zero per la ricostruzione della posizione assoluta**
    - **elettronica di condizionamento veloce (circuiti di conteggio)**



AVANTI



INDIETRO

- **<u>Elettronica di condizionamento</u>: contatori**

MODALITA' DI INTERFACCIAMENTO DI SENSORI/ATTUATORI

- **Sensori/attuatori discreti (ON/OFF)**
  - **Applico tensione e verifico passaggio di corrente**
  - **Facile da trasmettere, facile da isolare**
  - **Il relais è ingombrante, lento e dissipativo**
  - **L'isolatore galvanico isola (1500V) ma ha meno potenza**