

UNITÀ DIDATTICA **3.3**

SOFTWARE (ELEMENTI E LINGUAGGI DI PROGRAMMAZIONE)

- 3.3.1** ELEMENTI SOFTWARE
- 3.3.2** CARATTERISTICHE PRINCIPALI E CLASSIFICAZIONE DEI LINGUAGGI DI PROGRAMMAZIONE
- 3.3.3** LINGUAGGIO (SCHEMA) LADDER
- 3.3.4** CONVERSIONE DEGLI SCHEMI ELETTRICI FUNZIONALI IN SCHEMI LADDER
- 3.3.5** TECNICHE DI PROGRAMMAZIONE
- 3.3.6** LINGUAGGIO CON DIAGRAMMA FUNZIONALE (CEI 3-35)
- 3.3.7** LINGUAGGIO A BLOCCHI LOGICI
- 3.3.8** LINGUAGGIO IN LISTA DI ISTRUZIONI
- 3.3.9** ALTRI TIPI DI LINGUAGGI
- 3.3.10** PROGRAMMAZIONE LINEARE E STRUTTURATA
- 3.3.11** RIFERIMENTI NORMATIVI
- 3.3.12** CONSIDERAZIONI FINALI

3.3.1 - Elementi software.

A differenza di quelli hardware, visti nella precedente unità didattica, si intendono per *elementi software* quelli *non* «tangibili», specificati dal costruttore nei dati tecnici del PLC. Sono da considerare tali:

- i *temporizzatori*;
- i *contatori*;
- i *relè interni*;
- l'*accumulatore*.

I **temporizzatori** ed i **contatori** equivalgono ai corrispondenti elettromeccanici ed elettronici.

I **relè interni**, che possono essere volatili (non ritentivi) e non volatili (ritentivi), assumono varie terminologie e, a seconda del tipo di PLC, prendono il nome di: *memorie interne*, *indicatori*, *relè di appoggio*, *variabili interne*, *flag*, *marker*, ecc.. Equivalgono ai relè ausiliari elettromeccanici e sono praticamente aree di memoria riservate a risultati intermedi utilizzabili al momento opportuno.

Un elemento estremamente importante è l'**accumulatore** o *registro temporaneo*, che assume varie terminologie ma ha la stessa funzione in tutti i PLC. È costituito da un cella di memoria a 1 bit ed ha la funzione di memorizzare, in modo *parziale*, il risultato logico di ogni riga di istruzioni (sottounità 3.3.8). Parecchi PLC possiedono più registri temporanei con il risultato di avere istruzioni più potenti con minor uso di capacità della memoria utente.

Si considerano elementi software oltre che hardware, anche tutte le *uscite fisiche* in quanto possono essere utilizzate come contatti ausiliari, allo stesso modo degli ingressi.

Tutti gli elementi possono venire indirizzati, interrogati (test), combinati tra loro in qualsiasi modo ed in qualunque punto del programma.

3.3.2 - Caratteristiche principali e classificazione dei linguaggi di programmazione.

Le particolarità dei linguaggi devono corrispondere ad un serie di parametri essenziali, quali:

- facilità di stesura anche per il personale non altamente specializzato;
- essere sufficientemente potenti;
- essere accessibili a tecnici provenienti da esperienze in settori diversi.

Una prima classificazione di carattere generale dei linguaggi di programmazione comune a tutti i controllori programmabili, è fornita nello schema di figura 3.3.1. Per detti linguaggi la Norma IEC 1131 (CEI 65-40), ha codificato segni grafici ed alcune istruzioni di base, ma la maggior parte del repertorio di istruzioni di livello superiore è ancora piuttosto diversificato anche se, tuttavia, presenta molte similitudini.

Tutto ciò comporta una certa limitazione per i tecnici che operano su controllori diversi, poiché per loro si rende necessaria la conoscenza sia dell'hardware che del software dei vari tipi di PLC.

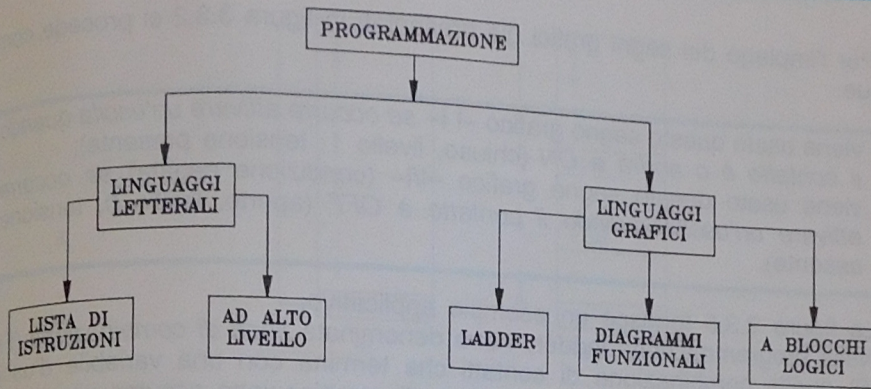


Fig. 3.3.1 - Schema a blocchi dei linguaggi di programmazione per PLC.

3.3.3 - Linguaggio (schema) ladder.

Il nome deriva dall'inglese *ladder* (scala) per il particolare disegno e viene usualmente denominato «schema a contatti»; a tutt'oggi è quello maggiormente utilizzato in campo internazionale per la programmazione dei PLC.

Si compone di una serie molto limitata di segni grafici e risulta estremamente congeniale ai programmatori con conoscenza di logica elettromeccanica, perché richiama in un certo senso lo schema elettrico funzionale. I segni grafici sono rappresentati nella figura 3.3.2 e spesso vengono riportati anche sulle unità di programmazione portatili.

Questi particolari segni grafici normalizzati IEC 1131, derivano da una simbologia americana già in uso per gli schemi funzionali.

Lo schema ladder strutturalmente è composto da due linee verticali e da linee orizzontali (scala), sulle quali vengono disegnati gli elementi costituenti il sistema da controllare (Fig. 3.3.3).

Si distinguono:

- la barra d'alimentazione (riga verticale sinistra), cui fanno capo tutti gli elementi di *input*;
- il ritorno comune o massa (riga verticale destra che spesso viene omessa), cui fanno capo tutte le *variabili d'uscita (output)*;
- la *zona di test* (o zona di input): in questa parte dello schema vengono disegnati i vari input in serie e in parallelo, in chiusura e in apertura, seguendo la logica di evoluzione del processo; a seconda del loro stato verrà stabilita, attraverso il test eseguito dal microprocessore, in funzione del programma utente, la condizione della variabile d'uscita (output);
- la *zona di azione* (o zona di output), destinata alle variabili d'uscita abilitate o meno dalla zona precedente;
- i segni + stanno ad indicare che non vi è collegamento fisico tra le due zone, ma solo la logica di processo dettata dalla programmazione; pure non vi è collegamento fisico tra i vari input, ma solo il *test* (interrogazione), che il microprocessore esegue sul loro livello logico 1 o 0;

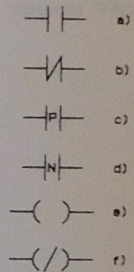


Fig. 3.3.2 - Segni grafici usati per il linguaggio ladder (a contatti). Da Norma IEC 1131 (CEI 65-40).

- a) contatto ON,
b) contatto OFF,
c) contatto rivelatore di transizione positiva (fronte di salita),
d) contatto rivelatore di transizione negativa (fronte di discesa),
e) variabile esterna o interna (output),
f) variabile esterna o interna negata (output not).

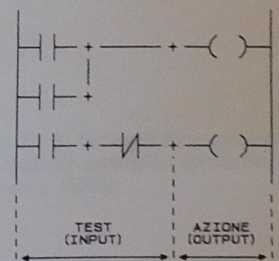


Fig. 3.3.3 - Esempio di schema ladder reale.

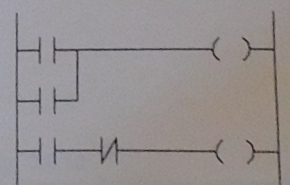
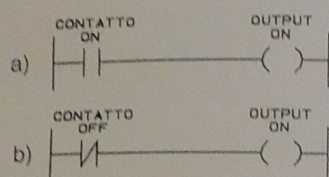


Fig. 3.3.4 - Esempio di schema ladder semplificato (abituale usato).

Di conseguenza non sarà più l'installatore a cablare fisicamente i vari contatti in serie o parallelo ma si limiterà soltanto al collegamento di ogni contatto singolarmente sul modulo di ingresso del PLC. Sarà il microprocessore stesso a livello software, in base al risultato ottenuto dal test ed al programma steso dall'utente, a combinare gli ingressi in serie o in parallelo per abilitare o meno l'uscita. In base al test verrà stabilito il livello 1 (tensione presente) se il contatto sul punto di ingresso del modulo è chiuso, livello 0 se il contatto è aperto.

- per semplificare il disegno usualmente il segno + si omette; lo schema prende quindi la forma di figura 3.3.4; la lettura dello schema va fatta sempre da sinistra verso destra e dall'alto verso il basso.

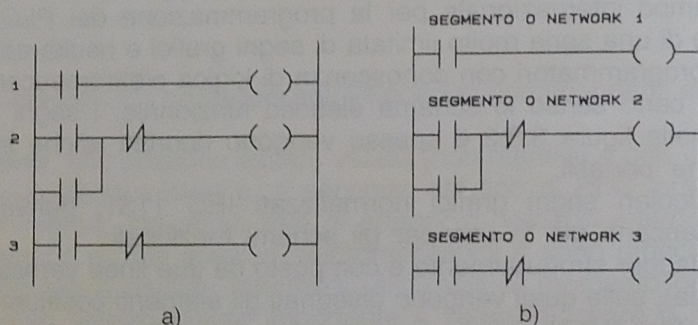
Per l'impiego dei segni grafici già presentati in figura 3.3.2 si procede come segue:



- viene usato questo segno grafico $-|I|$ se occorre attivare un'uscita quando il contatto è o andrà a ON (chiuso, livello 1, tensione presente);
- viene usato questo segno grafico $-|/I|$ (condizione negata) se occorre attivare un'uscita quando il contatto è OFF (aperto, livello 0, tensione assente).

La figura 3.3.5 fornisce un esempio applicativo.

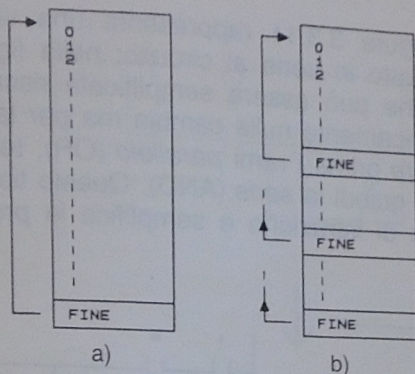
Nella programmazione ladder, viene denominata *linea di combinazione* (Fig. 3.3.6) quella combinazione di contatti che termina con una variabile d'uscita singola o in parallelo con altre. La linea di combinazione assume varie terminologie a seconda del tipo di controllore: ad esempio per i PLC SIEMENS o OMRON viene denominata «network» o «segmento».



3.3.4 - Conversione degli schemi elettrici funzionali in schemi ladder.

Lo schema elettrico funzionale può essere convertito in schema ladder osservando alcune indicazioni di carattere generale per tutti i tipi di PLC:

- negli schemi elettrici funzionali il flusso logico dei segnali va da *sinistra verso destra*, viceversa negli schemi ladder dall'*alto verso il basso*;
- nessun output può essere collegato *direttamente* alla barra di sinistra; per fare ciò occorre interporre un contatto di un relè interno. Tale contatto è inesistente fisicamente, ma per il PLC è un elemento da controllare come gli altri ingressi effettivi;
- non esiste limite al numero dei contatti da inserire nelle linee di combinazione (memoria permettendo);
- tutti gli output possono essere utilizzati come contatti input (via software);
- nessun contatto può essere disposto a destra dell'output;
- è possibile collegare in parallelo più output (come già accennato);
- il programma viene eseguito ciclicamente dal primo indirizzo (a cui corrisponde la prima istruzione) fino all'istruzione di fine programma. Questa caratteristica può essere utilizzata per «testare» un programma lungo nel quale risulta difficile la ricerca di eventuale errori; il programma stesso viene suddiviso in una serie di blocchi terminanti con l'istruzione di «Fine», rendendo più facile l'operazione (Fig. 3.3.7). Quando il primo blocco è stato testato si cancella la prima istruzione di «Fine», eseguendo tale procedura fino al completamento del test dell'intero programma.



3.3.5 - Tecniche di programmazione.

Molti circuiti elettrici funzionali non possono essere *convertiti* in schemi ladder semplicemente cambiandone i segni grafici e la disposizione. Si forniscono di seguito alcuni esempi di conversione di schemi funzionali in schemi ladder, per la programmazione dei PLC. Per semplicità i contatti vengono disegnati tutti in chiusura (NO).

Nelle figure 3.3.8 e 3.3.9 la conversione schema funzionale-ladder è immediata e non necessita di commenti; viceversa la figura 3.3.10a mostra la conversione reale che può essere semplificata riscrivendo lo schema come in figura 3.3.10b. Dal punto di vista elettrico il funzionamento non cambia, ma per la programmazione si preferisce spostare il contatto singolo (1) di un ramo parallelo nella parte in basso del ramo stesso, per evitare di usare istruzioni di operazioni tra blocchi, risparmiando memoria e tempo di scansione.

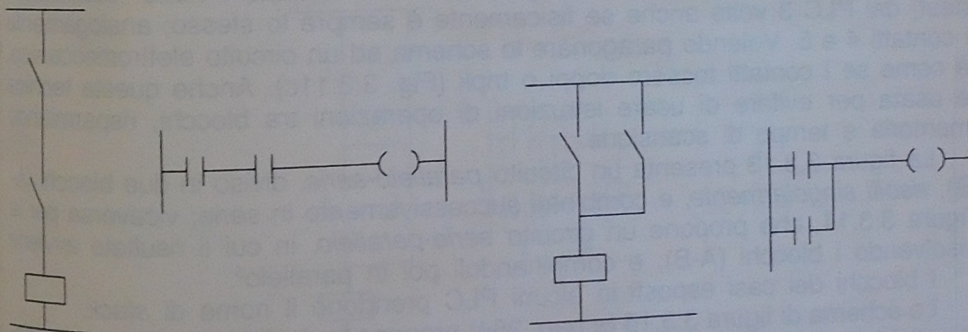


Fig. 3.3.8

Fig. 3.3.9

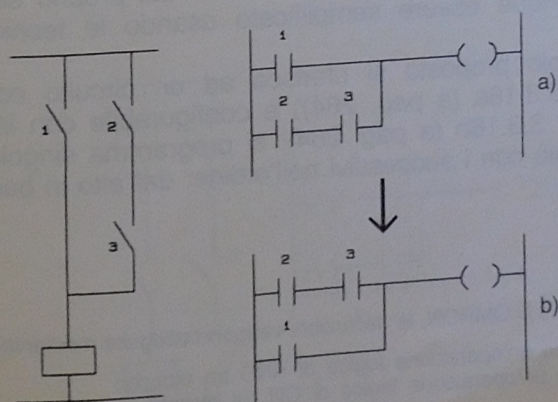


Fig. 3.3.10 -
 a) Schema funzionale con contatti in serie e parallelo, convertito in schema ladder;
 b) stesso schema semplificato.

Lo schema di figura 3.3.11 rappresenta una variante al precedente con aggiunta di un contatto in serie al circuito; nella figura 3.3.11a è descritta la conversione reale che può essere semplificata riscrivendo il circuito come in figura 3.3.11b. Elettricamente nulla cambia ma per la programmazione è preferibile risolvere *sempre* prima i rami *parallelo* (OR), tenendo presenti le considerazioni precedenti, e quindi la *serie* (AND). Questo tipo di soluzione è finalizzata al risparmio di righe di istruzione e semplifica la programmazione.

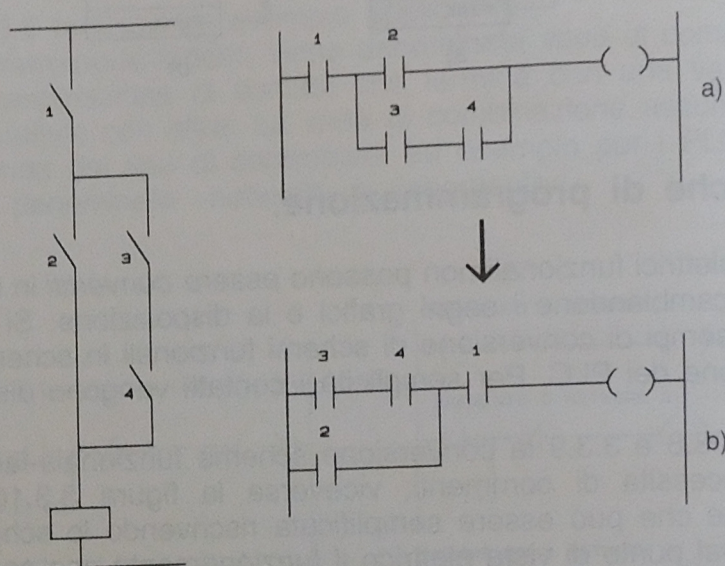


Fig. 3.3.11 -

a) Schema funzionale complesso convertito in schema ladder;
b) stesso schema semplificato.

La figura 3.3.12 propone un circuito complesso con rami in serie ed in parallelo, risolto come da figura 3.3.12a; anche in questo caso conviene *riscrivere* il circuito come esemplificato nella figura 3.3.12b. Il contatto 1 viene «esaminato» (test) dal PLC 3 volte anche se fisicamente è sempre lo stesso; analogamente i contatti 4 e 5. Volendo paragonare lo schema ad un circuito elettromeccanico, è come se i contatti fossero doppi o tripli (Fig. 3.3.11c). Anche questa tecnica è usata per evitare di usare istruzioni di operazioni tra blocchi, risparmiando memoria e tempo di scansione.

La figura 3.3.13 presenta un circuito *parallelo-serie*, diviso in due blocchi (A-B), *risolti singolarmente*, e combinati successivamente in serie; viceversa per la figura 3.3.14, che propone un circuito *serie-parallelo*, in cui il risultato avviene risolvendo i blocchi (A-B), e combinandoli poi in parallelo⁵.

I blocchi dei casi esposti in alcuni PLC prendono il nome di *stack*.

Lo schema di figura 3.3.15 (a pag. 264) propone la risoluzione di blocchi serie-parallelo combinati: nel caso esposto si divide il circuito in due blocchi (A-B), si programma ciascun blocco combinandoli successivamente in un unico circuito. Per la risoluzione ogni PLC applicherà i codici del proprio set di istruzioni. Il circuito può comunque essere semplificato usando le tecniche già studiate precedentemente.

Un altro esempio proposto si riferisce ad un circuito complesso, la cui conversione (Fig. 3.3.16a (a pag. 264)) è configurabile con la *suddivisione in blocchi* della figura 3.3.16b (a pag. 264): si programma singolarmente ciascun blocco combinandolo con i successivi nell'ordine: dall'alto in basso e da sinistra verso destra.

⁵ Con riferimento ai PLC OMRON, le risoluzioni vengono eseguite mediante istruzioni particolari:

- AND-LOAD, che indica un'operazione logica di AND tra blocchi;
- OR-LOAD, che indica un'operazione logica di OR tra blocchi.

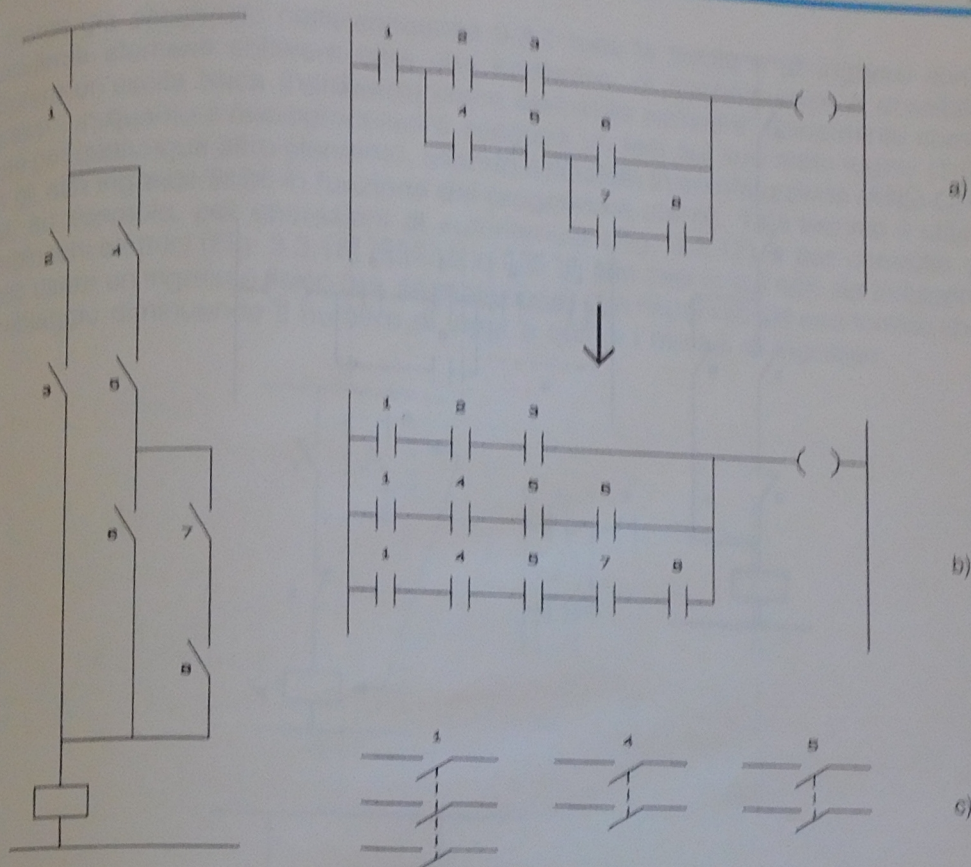


Fig. 3.3.12 -

a) Conversione in ladder di uno schema funzionale complesso;
 b) medesimo schema riscritto per semplificarne la programmazione;
 c) equivalente elettromeccanico dei contatti 1-4-5.

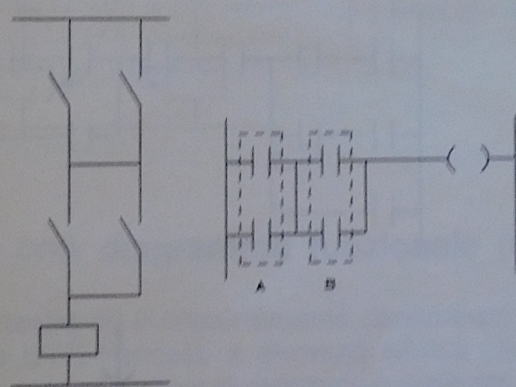


Fig. 3.3.13 - Schema funzionale con contatti parallelo-serie, convertito in schema ladder.

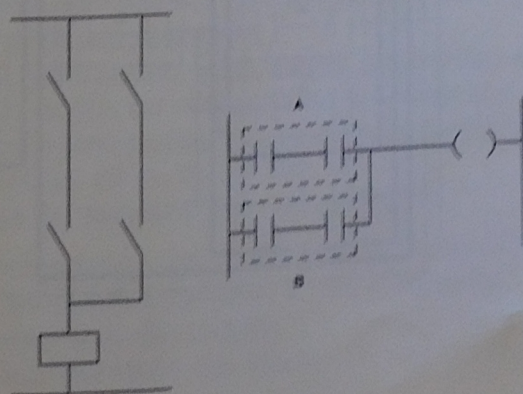


Fig. 3.3.14 - Schema funzionale con contatti serie-parallelo convertito in schema ladder.

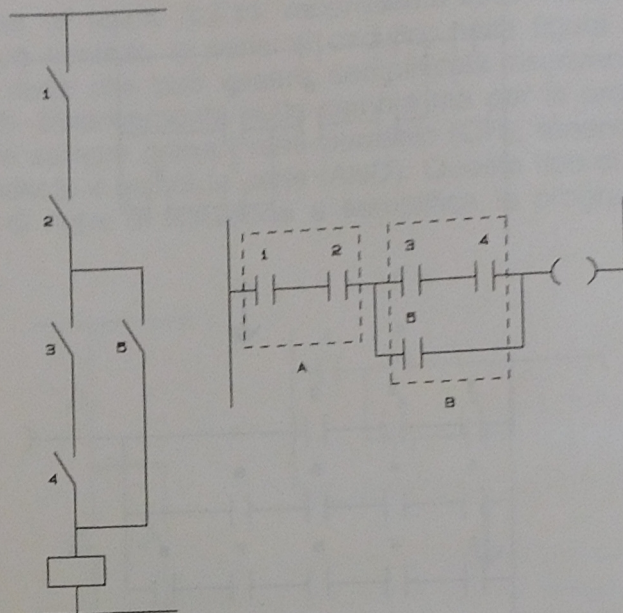


Fig. 3.3.15 - Risoluzione di blocchi circuitali serie-parallelo.

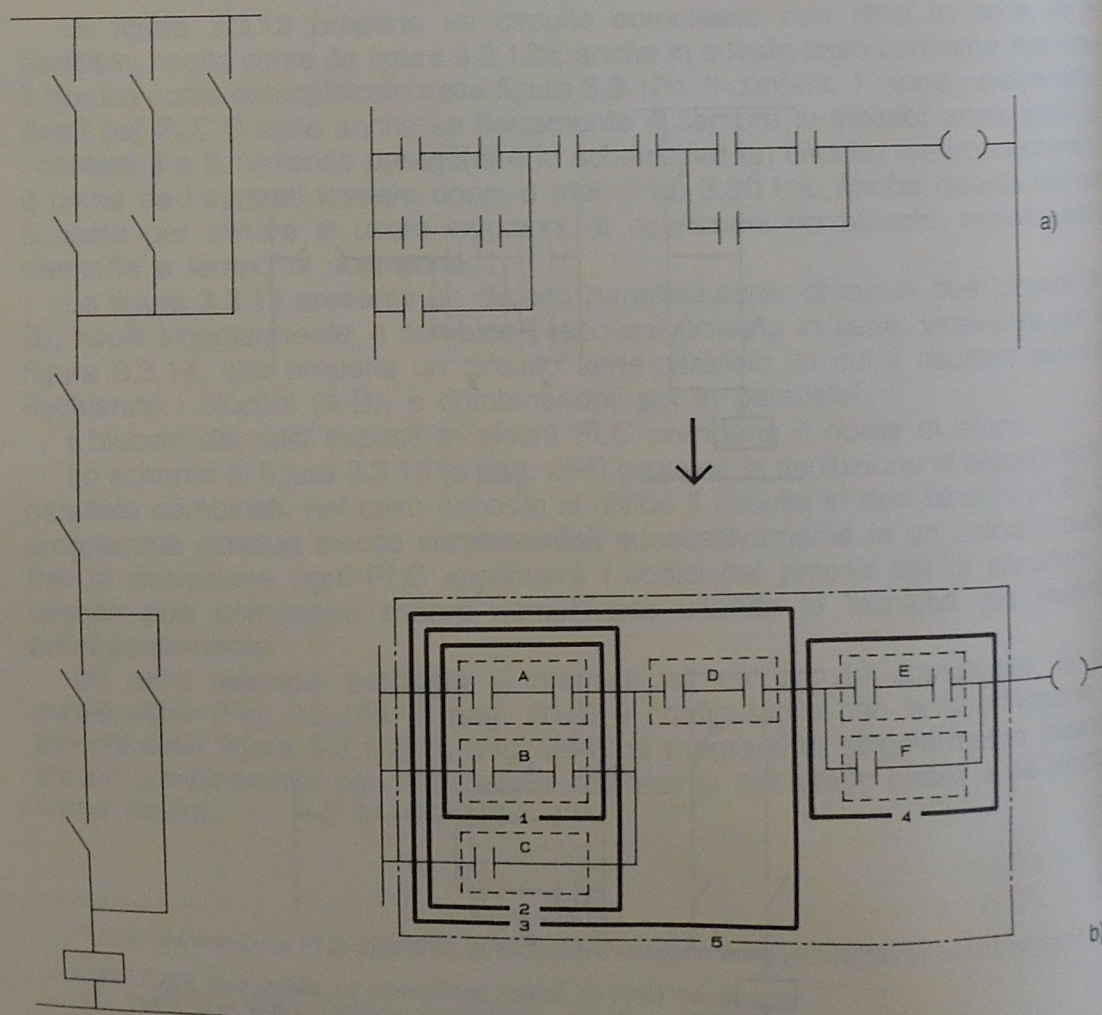


Fig. 3.3.16 -
a) schema funzionale con contatti in serie e in parallelo convertito in schema ladder;
b) soluzione suddividendo il circuito in blocchi.

Come già accennato nella sottounità 3.3.1 tutte le uscite e gli ingressi sono considerati elementi software oltre che hardware; di conseguenza è possibile utilizzare un'uscita fisica (hardware) come elemento *software*, tipicamente come ingresso, in quanto il microprocessore eseguirà un test sul suo stato logico (1-0) come per qualunque altro elemento, usando il risultato in combinazione (AND-OR) con gli altri ingressi *fisici*, in funzione del programma utente. Tale tecnica è utilizzata, ad esempio, per operazioni di autoritenuta (Fig. 3.3.17) o per controllo di interblocchi elettrici (Fig. 3.3.18) oppure in tutti gli altri casi in cui non sia indispensabile usare un ingresso fisico (es. segnalazione) con risparmio sia economico che di cablaggio diminuendo il numero di input e quindi i moduli di ingresso.

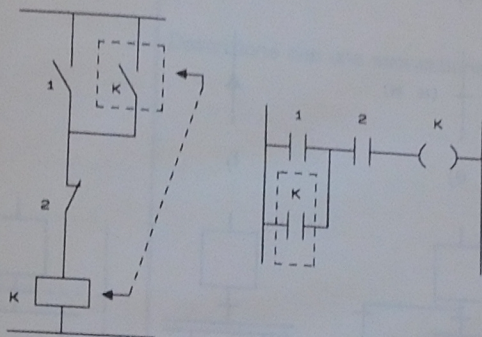


Fig. 3.3.17 - Esempio di uscita usata come ingresso in un circuito di autoritenuta.

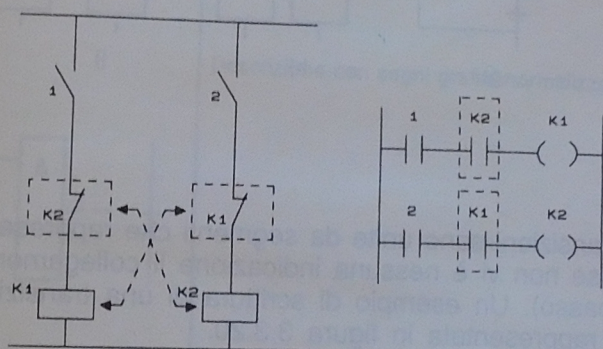


Fig. 3.3.18 - Esempio di uscita usata come ingresso per operazioni di interblocco.

3.3.6 - Linguaggio con diagramma funzionale (CEI 3-35).

Nasce nel 1977 in Francia ed è comunemente denominato *GRAFCET*; prima normalizzato con norme UTE francesi, è divenuto Norma CEI 3-35 (IEC 848).

Detto linguaggio descrive le funzioni di comando e controllo legate al processo senza tenere conto del tipo di tecnologia usata, quindi nessun campo viene escluso: è un tipo di programmazione strutturata (vedi sottounità 3.3.10) destinata allo svolgimento di tipo sequenziale e denominato anche «passo-passo»; presenta quindi molte affinità con il flow-chart (schema di flusso) e può essere applicato anche a sistemi non elettrici (pneumatici, idraulici, meccanici, ecc.), o in abbinamento a due o più tecnologie.

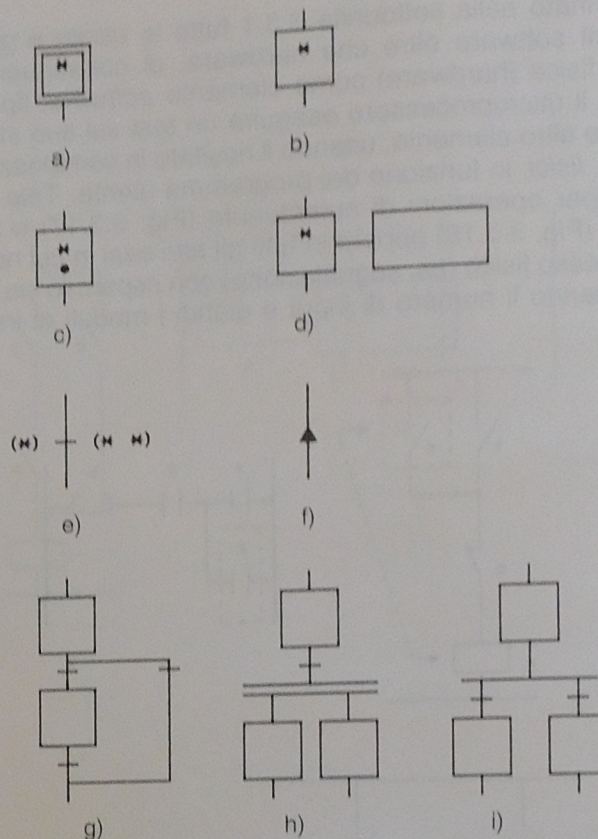
Il linguaggio grafico, utilizzando solo segni geometrici ed una simbologia alfanumerica, può essere scritto ed interpretato da tecnici con esperienze diverse che non abbiano una conoscenza specifica nel settore elettrico-elettronico. I segni grafici adottati sono descritti nella figura 3.3.19.

Il linguaggio presenta le seguenti caratteristiche:

- ad ogni *passo* corrispondono una o più *comandi od azioni*,
- ad ogni *transizione* corrispondono una o più *condizioni*,
- ogni azione è rappresentata da *output*,
- ogni transizione è rappresentata da *input*,

Fig. 3.3.19 - Segni grafici usati per la preparazione dei diagrammi funzionali per sistemi di comando e controllo (CEI 3-35).

- a) Passo (fase) iniziale, l'asterisco deve essere sostituito dal numero assegnato al passo;
 b) passo (fase), l'asterisco deve essere sostituito dal numero assegnato al passo;
 c) passo attivo;
 d) passo associata ad un'azione;
 e) transizione; l'asterisco deve essere sostituito dal numero assegnato alla transizione, i due asterischi vanno sostituiti con le condizioni associate alla transizione: es. testo, grafico o espressione booleana;
 f) collegamento orientato;
 g) esempio di salto di passo;
 h) esempio di inizio sequenze simultanee;
 i) esempio di divergenza (di selezione di sequenza).



- passi e transizioni sono unite da segmenti che rappresentano i *collegamenti orientati* (se non vi è nessuna indicazione il collegamento è diretto dall'alto verso il basso). Un esempio di scrittura di una transizione in quattro modi diversi è rappresentata in figura 3.3.20.

Durante l'evoluzione del processo i passi possono essere:

- *attivi*,
- *inattivi*.

Per passo **attivo** si intende che i comandi o le azioni ad esso associati sono esecutivi (es. eccitare la bobina di una elettrovalvola). Il comando *non è memorizzato*.

Per passo **inattivo** si intende che i comandi o le azioni ad esso associati devono terminare o mantenere il loro stato (es. eccitare la bobina di una elettrovalvola e mantenerla eccitata). Il comando *è memorizzato*.

I vari passi-azioni che si susseguono uno dopo l'altro (Fig. 3.3.21 a pag. 268), durante il funzionamento di un sistema automatico, iniziano da una situazione di riposo e vengono eseguiti solo dopo il superamento delle condizioni necessarie (transizioni); terminato il ciclo il programma riparte dal passo iniziale.

Nella stesura del programma occorre rispettare alcune regole evolutive (CEI 3-35):

- si dovrà avere almeno un *passo iniziale*;
- deve sempre esistere l'*alternanza passo-transizione* (Fig. 3.3.22);
- il superamento di una transizione determina la *disattivazione* delle azioni associate ai passi precedenti, a meno che non sia diversamente e obbligatoriamente espresso, e l'*attivazione* delle azioni associate ai passi successivi;
- la transizione viene superata quando tutte le condizioni associate sono *vere* e tutti i passi immediatamente precedenti sono *attivi*.

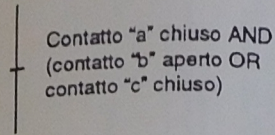
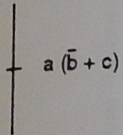
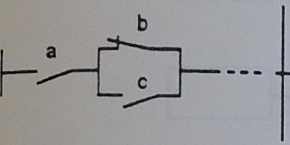
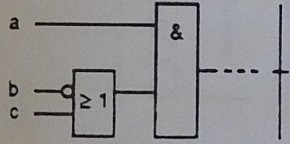
No.	Segno grafico	Descrizione
3.2		Descrizione della condizione di transizione associata con un testo
3.3		Descrizione con una espressione booleana
3.4		Descrizione con segni grafici normalizzati
3.5		

Fig. 3.3.20 - Modi di rappresentazione delle condizioni di transizione (da Norma CEI 3-35).

Il linguaggio con diagramma funzionale può essere rappresentato secondo due livelli:

- 1° livello, è denominato anche letterale, descrittivo o funzionale e il compito degli elementi che compongono il sistema viene descritto in modo dettagliato; è comprensibile da qualsiasi tecnico.
- 2° livello, è denominato anche tecnologico e descrive gli elementi che concorrono alla risoluzione del sistema con specifici riferimenti tecnici; a questa soluzione va sempre abbinata una legenda di interpretazione delle sigle.

A volte è utile descrivere il sistema utilizzando entrambi i livelli; si parla allora di «descrizione tecnologica (2° livello) con didascalie funzionali (1° livello)».

La tavola 3.3.1 a pag. 267 riporta un esempio di linguaggio di programmazione in diagramma funzionale, riferito al comando di una unità operatrice con i livelli descritti, per un migliore confronto.

Molti PLC hanno la possibilità di essere programmati direttamente in diagramma funzionale (Grafcet), utilizzando software dedicati, oppure da unità di programmazione che permetta l'inserimento diretto del programma in forma grafica.

La figura 3.3.23 fornisce un esempio di applicazione del software per PLC Simatic SIEMENS.

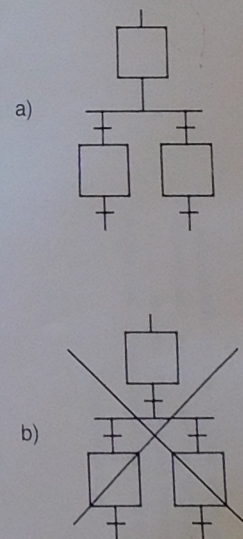


Fig. 3.3.22 - Descrizione dell'alternanza passo-transizione; a) disposizione corretta, b) disposizione errata.

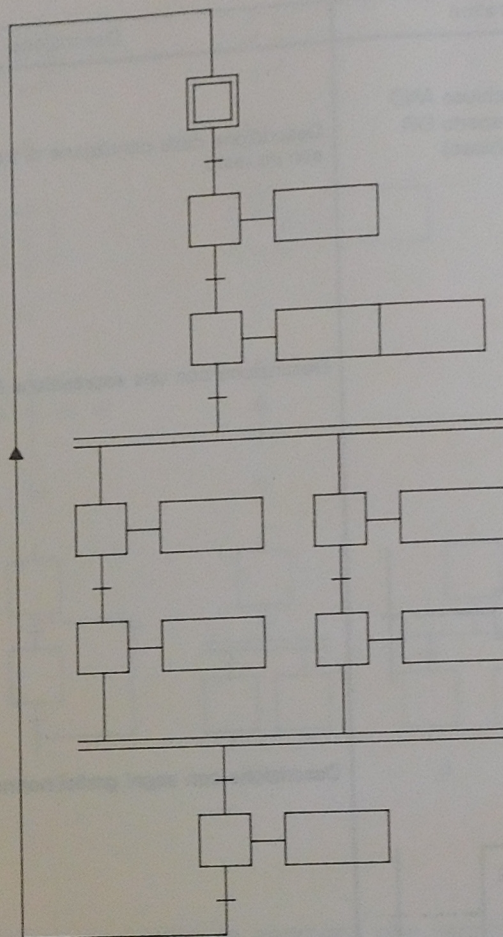


Fig. 3.3.21 - Esempio di diagramma funzionale.

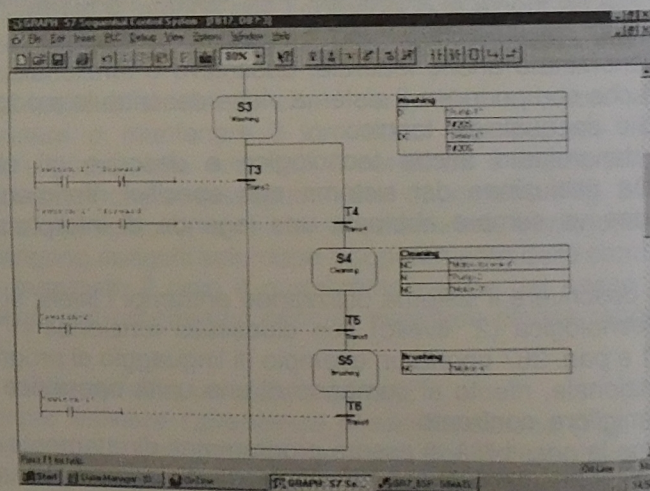
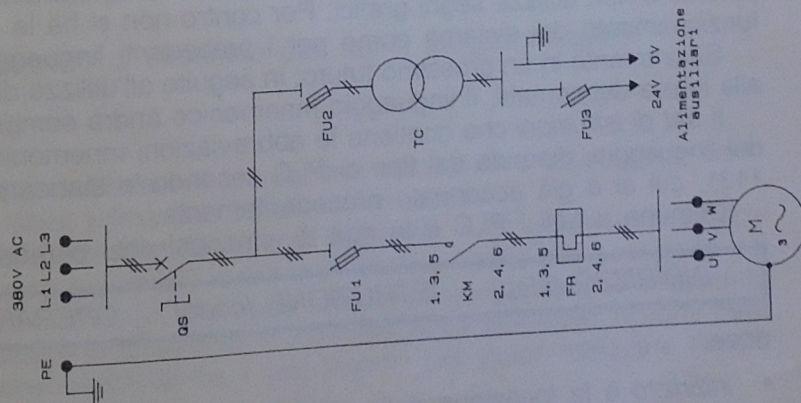


Fig. 3.3.23 - Esempio di diagramma funzionale su video con software S7-Graph (da catalogo SIEMENS). Si noti che le transizioni sono indicate secondo il linguaggio ladder.

COMANDO DI UNA UNITA' OPERATRICE

SCHEMA DI POTENZA UNIFILARE



LEGENDA

- KM - Contattore movimento unita' operatrice
- FR - Protezione termica del motore
- SB1 - Pulsante di arresto unita' operatrice
- SB2 - Pulsante di marcia unita' operatrice

SCHEMA DI COMANDO

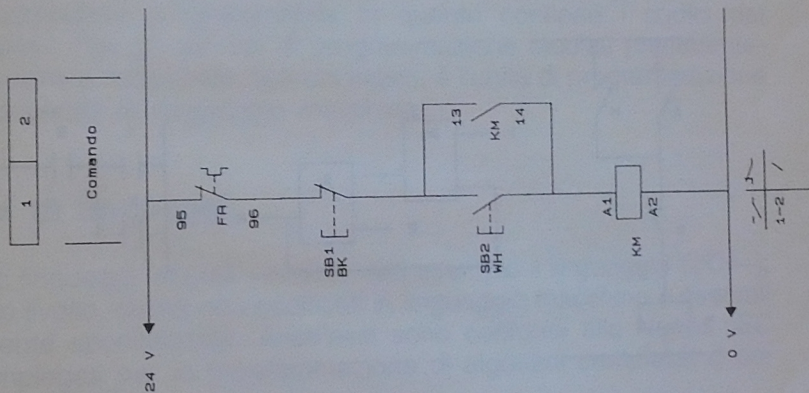


DIAGRAMMA FUNZIONALE DI 1° LIVELLO

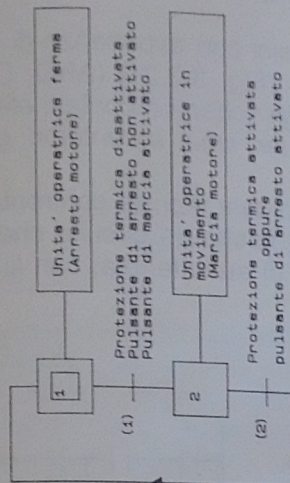


DIAGRAMMA FUNZIONALE DI 2° LIVELLO

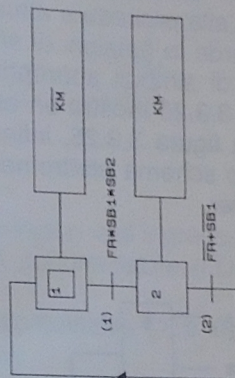
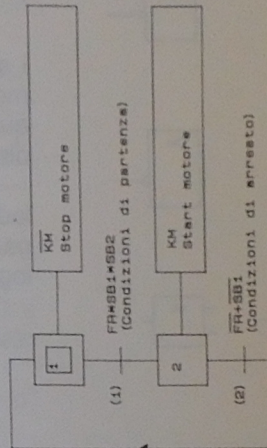


DIAGRAMMA FUNZIONALE TECNOLOGICO CON DIDASCALIE FUNZIONALI



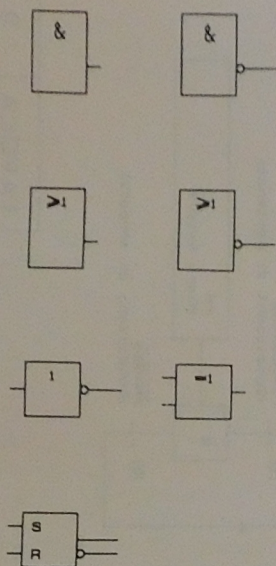


Fig. 3.3.24 - Esempio di segni grafici per linguaggio a blocchi logici (da Norma CEI 3-26).

3.3.7 - Linguaggio a blocchi logici.

Particolarmente congeniale per i tecnici con esperienze di elettronica, utilizza i segni grafici della logica combinatoria (Fig. 3.3.24) (AND, OR, XOR, ecc.), in modo simile alla procedura attuata per gli schemi a contatti elettromeccanici. Per quanto riguarda le funzioni di *shift register*, *timer* e *contatori*, gli stessi controllori dispongono di simboli appropriati.

La figura 3.3.25 mostra un esempio di tale tecnica confrontata con lo schema ladder. Nella figura 3.3.26, infine, viene risolto un circuito che ha come base di partenza uno schema elettromeccanico di autoritenuta, con linguaggio a blocchi logici e ladder.

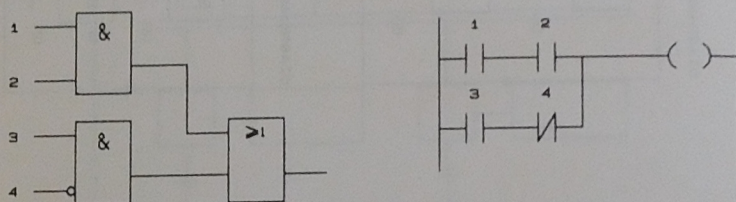


Fig. 3.3.25 - Confronto tra linguaggio ladder e a blocchi logici.

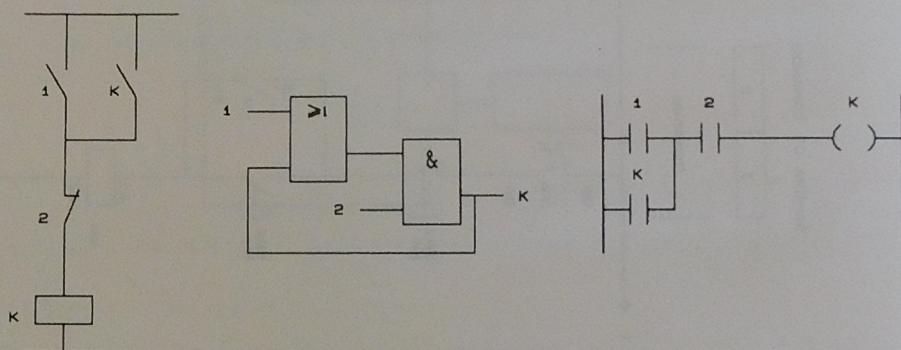


Fig. 3.3.26 - Circuito di autoritenuta a contatti risolto con linguaggio a blocchi logici e ladder.

3.3.8 - Linguaggio in lista di istruzioni.

È un linguaggio di tipo letterale denominato anche «*mnemonico*», risulta molto potente e non utilizza segni grafici. Per contro non si ha la «visione» rapida del funzionamento del sistema come per i precedenti linguaggi.

Sicuramente in un prossimo futuro, in seguito all'utilizzo di personale estraneo alla logica dei contatti, il linguaggio mnemonico andrà sempre più diffondendosi.

Il *set di istruzioni* che contiene le abbreviazioni mnemoniche o alfanumeriche del linguaggio, dipende dal tipo di PLC secondo lo standard internazionale IEC 1131, cui si è già accennato precedentemente.

Comune a tutti i PLC è la riga di istruzioni così caratterizzata:

INDIRIZZO (step)	ISTRUZIONE (code)	OPERANDO (elemento)
------------------	-------------------	---------------------

dove:

- *indirizzo* è la locazione della memoria utente;
- *istruzione* (operatore) è il comando per il PLC (cioè che cosa deve eseguire);
- *operando* è la conseguenza del comando (cioè su quale elemento hardware o software deve essere eseguito il comando stesso).

In pratica il PLC viene istruito riga per riga sull'operazione che deve compiere. Con riferimento al set di istruzioni dei PLC della ditta OMRON, si fornisce un semplice esempio di lista di istruzioni:

INDIRIZZO	ISTRUZIONE	OPERANDO
00000	LD	
00001	OUT	00000
00002	END	00101

la cui *chiave di lettura* risulta:

- all'indirizzo 0, esegui un test (LD) sullo stato logico dell'ingresso n° 00000 e memorizza il risultato nel registro (memoria) *accumulatore* (LD viene usato all'inizio di ogni combinazione logica);
- all'indirizzo 1, trasferisci il contenuto dell'accumulatore (1 o 0) all'uscita n° 00101 (che sarà quindi attivata o disattivata);
- all'indirizzo 2, leggi il comando di fine programma (END) e ritorna all'indirizzo 00000.

Ogni riga di istruzioni termina quindi con un'azione; a volte necessitano *due righe* per l'azione completa come ad esempio per l'attivazione di un temporizzatore: la prima riga è dedicata al comando di attivazione vero e proprio, la seconda per impostare il valore di conteggio.

È bene precisare che il linguaggio mnemonico descritto è l'unico comprensibile direttamente dal controllore programmabile in quanto contiene i codici del «linguaggio macchina». Per gli altri tipi di programmazione studiati precedentemente (ladder, diagramma funzionale, blocchi logici), è l'unità di programmazione che provvede a convertirli in linguaggio macchina.

3.3.9 - Altri tipi di linguaggi.

Sono i cosiddetti linguaggi letterali «evoluti» (ad esempio il linguaggio C/C++), sono del tipo ad alto livello, quindi non codificati in linguaggio macchina e riservati a personale altamente specializzato. Anch'essi sono conformi alla Norma IEC 1131 e vengono impiegati per la programmazione di algoritmi complessi o per compiti di elaborazione dati.

Non si è ritenuto approfondire ulteriormente l'argomento nel contesto riservato alle finalità del corso.

3.3.10 - Programmazione lineare e strutturata.

Alcuni PLC consentono di effettuare due tipi di programmazione:

- lineare,
- strutturata.

Nella programmazione **lineare** il programma viene memorizzato in un unico blocco ed eseguito dalla prima fino all'ultima istruzione.

La programmazione **strutturata** invece suddivide il programma in *blocchi* distinti gerarchicamente connessi, in cui ogni blocco costituisce una parte del processo (Fig. 3.3.27). I blocchi che assumono terminologie diverse a seconda del PLC, possono essere programmati e collaudati separatamente; occorre però descrivere la relazione esistente tra i vari blocchi per poter realizzare l'intero processo e cioè:

- la sequenza di esecuzione dei blocchi;
- le condizioni che determinano l'esecuzione o meno del blocco;
- l'integrazione dei blocchi.

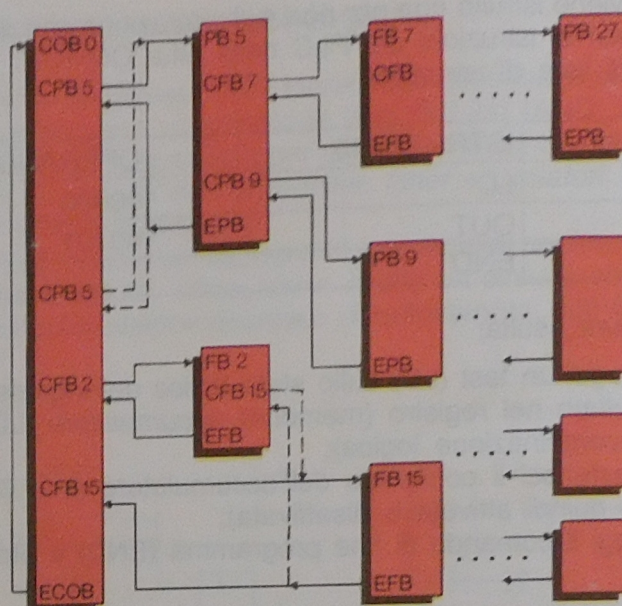


Fig. 3.3.27 - Organizzazione strutturata a blocchi suddivisi secondo le diverse sigle (da fascicolo tecnico SAIA PLC).

I blocchi possono essere salvati in modo indipendente e successivamente uniti (linkati) al fine di ottenere il programma principale. Si comprende quindi come si possa intervenire modificando un singolo blocco mantenendo inalterato il resto del programma.

Nella struttura deve esistere almeno un blocco principale (COB0.....ECOB nella figura 3.3.27), conseguentemente, a seconda della potenza della CPU, è possibile organizzarne altri (COB1, COB2, ecc.) in modo da ottenere una *elaborazione parallela* con tanti programmi indipendenti. Ciò contribuisce notevolmente ad una maggiore comprensibilità, per software complessi e facilità di manutenzione degli stessi.

Ogni blocco principale può essere successivamente suddiviso in altri blocchi secondari (PB5, FB7, PB9, ecc. nella figura 3.3.27), che svolgono determinate funzioni e vengono richiamati, mediante una procedura condizionata o incondizionata, dal blocco principale o dai blocchi secondari se la struttura prevede vari stadi di annidamento.

Un tipo di software strutturato è già stato presentato nella sottounità 3.3.6, nello studio della programmazione dei diagrammi funzionali (Grafcet).

3.3.11 - Riferimenti normativi.

Le convenzioni riguardo la normalizzazione dei controllori programmabili sono di recente costituzione. Si occupano dei PLC le Norme IEC 1131 tradotte dalle EN 61131 e classificate per il CEI dal *Comitato Tecnico n° 65*; in esse vengono trattati argomenti riguardanti l'hardware ed il software dei controllori programmabili. A questo proposito ogni linguaggio per la programmazione fissato dalla norma sopracitata assume la seguente *terminologia*:

- linguaggio a *lista di istruzioni* (mnemonico) = IL;
- linguaggio con *schema a contatti* (ladder) = LD;
- linguaggio a *blocchi logici* (funzionali) = FBD;
- linguaggio *testuale strutturato* (schema di flusso) = ST;
- linguaggio con *diagramma funzionale sequenziale*, già presente come Norma CEI 3-35 = SFC.

La tavola 3.3.2 riassume i principali linguaggi sopracitati.

Si ritiene opportuno citare alcuni termini, tratti dal catalogo SIEMENS e non *normalizzati*, in quanto sono entrati nell'uso comune e rappresentano alcuni dei linguaggi di programmazione sopracitati; in particolare:

SEGNI GRAFICI PER SCHEMI LADDER (LD)	SEGNI GRAFICI PER DIAGRAMMI FUNZIONALI SEQUENZIALI (SFC) (VEDERE ANCHE CEI 3-35)	SEGNI GRAFICI PER BLOCCHI LOGICI (FBD)	LISTA DI ISTRUZIONI (IL)
			Indirizzo
Contatto di chiusura (ON)			Operatore (codice)
			Operando
Contatto di apertura (OFF)			Significato
Contatto rilevatore di transizione positiva (fronte di salita)			
Contatto rilevatore di transizione negativa (fronte di discesa)			
Uscita interna o esterna			
Uscita interna o esterna con ritenuta SET			
Uscita interna o esterna non ritenuta RESET			
Barra di alimentazione sinistra			
Barra di alimentazione destra o ritorno comune (spesso viene omessa)			
Contatto rilevatore di transizione positiva (fronte di salita)			
Contatto di chiusura (Come sopra)			
Contatto rilevatore di transizione negativa (fronte di discesa)			
Uscita interna o esterna			
Uscita interna o esterna negata			
Fine programma			

NOTA 1	NOTA 2
Oltre agli operatori possono essere presenti modificatori del tipo: N - NOT (negazione booleana dell'operando)	Gli operandi dipendono dal tipo di PLC considerato. Consultare quindi la consultazione di relativo manuale

TAVOLA	3.3.2
--------	-------

NOTA 1

Oltre agli OPERATORI possono essere presenti MODIFICATORI del tipo:
N - NOT (negazione booleana dell'operando)

NOTA 2

Gli OPERANDI dipendono dal tipo di PLC considerato; si rimanda quindi in consultazione al relativo manuale.

- il termine *KOP* (KOntactPlan) significa *schema a contatti (ladder)*;
- il termine *FUP* (FUnktionsPlan) è destinato alla programmazione con *blocchi logici*;
- il termine *AWL* (AnWeisungListe) identifica il *linguaggio mnemonico (lista di istruzioni)*.

È pure molto in uso il termine *GRAFCET*, tratto dalle norme *UTE* francesi per rappresentare il diagramma funzionale sequenziale.

3.3.12 - Considerazioni finali.

Tenuto presente il linguaggio previsto per il tipo di PLC scelto, il programmatore deciderà quello a lui più congeniale secondo le proprie esperienze.

È indispensabile per il programmatore *decidere* la condizione (NC-NO) dei vari input, tenuto conto delle imposizioni normative, e per l'installatore *conoscere* la medesima condizione, per non creare successivamente alterazioni del processo.

Se ad esempio il programmatore ha deciso il test di un input considerato in chiusura (NO), codificando quindi l'istruzione in modo appropriato e l'installatore collega erroneamente in apertura (NC) quel contatto, durante l'esecuzione (RUN), il PLC opererà in maniera errata, cioè in modo esattamente opposto a quella desiderata.