PLC e standard IEC 1131-3

PLC e standard IEC 1131-3
- introduzione al PLC -

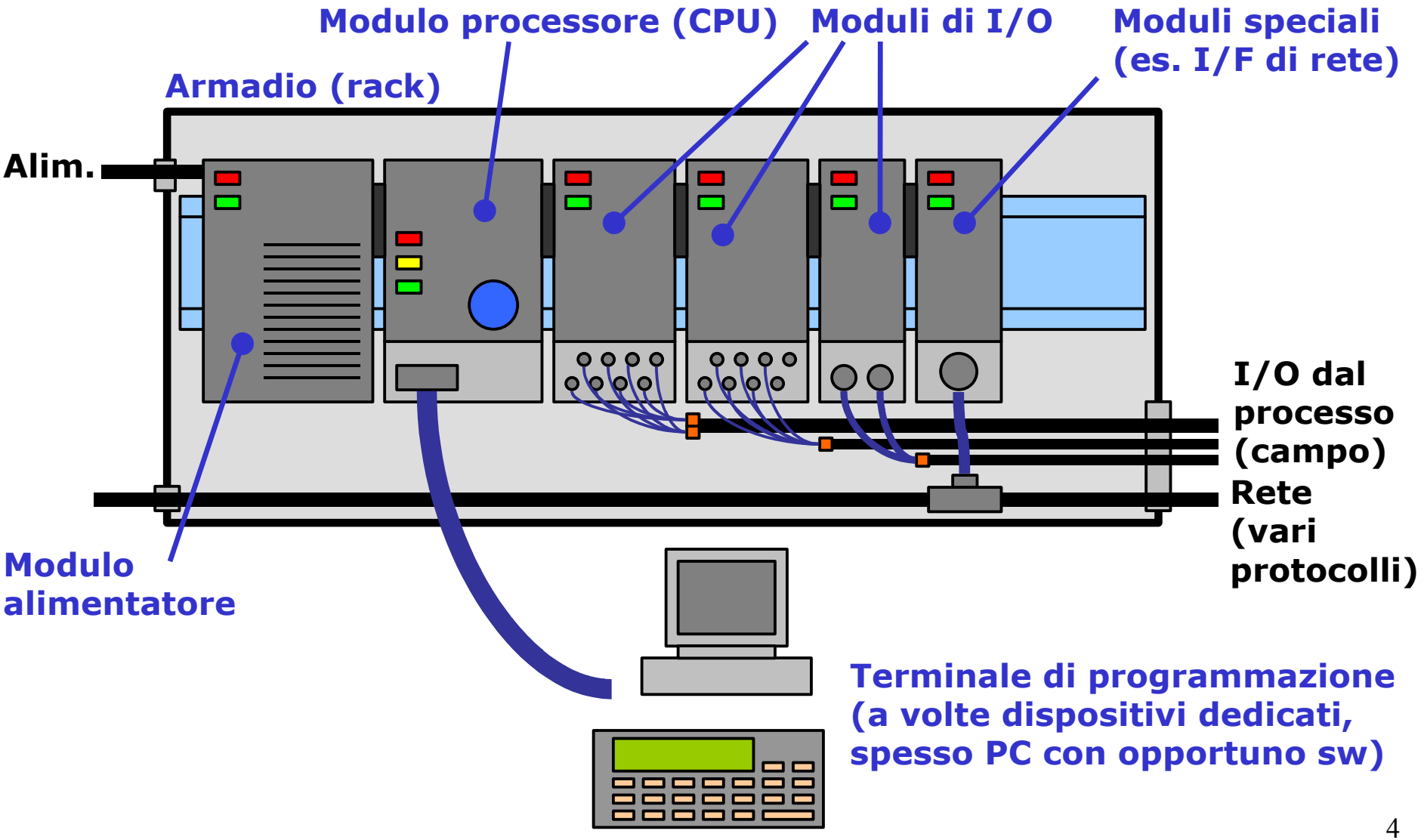
Definizioni secondo lo standard IEC 1131.

PLC (Programmable Logic Controller): sistema elettronico a funzionamento digitale, destinato all'uso in ambito industriale, che utilizza una memoria programmabile per l'archiviazione interna di istruzioni orientate all'utilizzazione per l'implementazione di funzioni specifiche, come quelle logiche, di sequenziamento, di temporizzazione, di conteggio e di calcolo aritmetico, e per controllare, mediante ingressi ed uscite sia digitali che analogici, vari tipi di macchine e processi.

Sistema PLC: configurazione realizzata dall'utilizzatore, formata da un PLC e dalle periferiche associate, necessaria al sistema automatizzato previsto.

PLC e standard IEC 1131-3
- introduzione al PLC -

Componenti fondamentali di un sistema PLC.



PLC e standard IEC 1131-3
- introduzione al PLC -

Armadio.

Contiene i vari moduli assicurandone la connessione meccanica ed elettrica (tramite bus) e la schermatura. Le sue caratteristiche fondamentali sono il numero di slot, il grado di protezione, le dimensioni e il tipo di fissaggio.

Modulo alimentatore.

Fornisce l'alimentazione stabilizzata ai moduli del rack. Le sue caratteristiche principali sono la potenza massima erogabile, la connettibilità in parallelo (per aumentare la potenza o per motivi di ridondanza), la possibilità di inviare al PLC un segnale di shutdown in caso di mancanza di alimentazione, la presenza di batterie tampone e di indicatori di stato.

Moduli di ingresso/uscita (I/O).

Il PLC comunica con il campo attraverso moduli di I/O digitali e analogici, che assicurano l'isolamento galvanico per salvaguardare l'elettronica interna. La trasmissione avviene in tensione o - più spesso - in corrente, modulando l'assorbimento sulle linee di alimentazione (il che richiede un solo cavo ed è preferibile in presenza delle alte cadute resistive date dai collegamenti lunghi). I moduli di I/O analogici realizzano anche le conversioni D/A e A/D.

PLC e standard IEC 1131-3
- introduzione al PLC -

Moduli speciali.

Ne esistono di molti tipi. I principali sono

- moduli di I/O remoto (posti in un rack diverso da quello del PLC),
- moduli per connessione in rete (per bus di campo, ethernet,...),
- moduli per controllo PID,
- moduli per la lettura di sensori particolari (termocoppie, encoder,…),
- moduli d'interfaccia operatore (tastierini, display,…),
- moduli di backup (CPU di riserva sincronizzate con quella principale, che le subentrano in caso di malfunzionamento).

Terminale di programmazione.

Vi sono terminali di tipo dedicato che si collegano direttamente al PLC tramite una porta di comunicazione e sono dotati di una tastiera per l'inserimento delle istruzioni e di un display per il controllo del programma.
Sono sempre più diffusi sistemi di sviluppo basati su PC, con cui si effettua off-line la programmazione del codice da memorizzare sul PLC. Si utilizzano dei pacchetti software appositi. I terminali PC sono connessi al PLC direttamente o via rete. Spesso consentono anche il monitoraggio del PLC durante il suo normale funzionamento.

PLC e standard IEC 1131-3
- introduzione al PLC -

Modulo processore.

Il modulo processore (CPU) contiene uno o più microprocessori, che eseguono i programmi del sistema operativo e quelli sviluppati dall'utente, la memoria dove questi programmi sono conservati e l'hardware per l'interfacciamento con gli altri moduli del sistema.

La sua modalità di funzionamento standard consiste nell'eseguire periodicamente il ciclo di copia massiva degli ingressi e delle uscite, che consiste dei seguenti passi:

- Lettura degli ingressi fisici e aggiornamento coi valori così ottenuti di un'area specifica della memoria;
- Esecuzione del programma utente, che opera sui valori in memoria e in memoria pone i risultati;
- Esecuzione dei programmi di gestione del sistema (ad es. di diagnostica);
- Scrittura sulle uscite fisiche dei valori corrispondenti conservati nell'area di memoria riservata a questo scopo.

In casi particolari (tipicamente guasti o emergenze) una CPU può eseguire operazioni con accesso immediato ai punti di ingresso/uscita, tipicamente in risposta ad interrupt. E' bene limitare l'uso degli interrupt il più possibile, dal momento che un loro utilizzo esagerato riduce la leggibilità del codice.

PLC e standard IEC 1131-3
- introduzione al PLC -

La velocità di elaborazione di una CPU è misurata dal tempo di scansione, cioè dal tempo che intercorre tra due attivazioni successive della stessa porzione del programma utente.

Il tempo di scansione dipende da quanti ingressi e uscite bisogna aggiornare e dalle dimensioni e dalla complessità del programma utente. E' definito in millisecondi per kiloword di programma (con word di 8 o 16 bit).

Il produttore di un PLC indica tipicamente un valor medio del tempo di scansione per programmi di media complessità (non tutte le istruzioni hanno la stessa durata). I tempi di scansione tipici sono dell'ordine di qualche frazione di ms/kw.

Il tempo di risposta del PLC è, invece, il massimo intervallo di tempo che passa tra la rilevazione di un certo evento e l'esecuzione dell'azione di risposta per esso programmata.

Il tempo di risposta, quindi, tiene conto anche dei ritardi introdotti dai moduli di I/O.

PLC e standard IEC 1131-3
- introduzione al PLC -

Il sistema operativo di un PLC è un insieme di programmi memorizzati in modo permanente, che si occupano (tra l'altro) di

- controllo delle attività del PLC,
- elaborazione dei programmi utente,
- comunicazione,
- diagnostica interna, ovvero
  - watchdog timer (controllo del tempo di esecuzione di alcune funzionalità e generazione di un allarme se esso supera una soglia assegnata),
  - controlli di parità sulla memoria e sulle linee di comunicazione,
  - controllo della tensione di alimentazione e dello stato delle batterie tampone.

Un generico PLC può trovarsi in tre modalità operative:

- esecuzione: si eseguono i programmi utente aggiornando ingressi e uscite;
- validazione: si eseguono i programmi ma l'aggiornamento delle uscite è disabilitato (questa modalità serve tipicamente a verificare la correttezza del codice);
- programmazione: questa è la modalità utilizzata per caricare il codice nella memoria del PLC.

PLC e standard IEC 1131-3
- introduzione al PLC -

La memoria di un PLC è organizzata per aree distinte:

- area del sistema operativo (ROM),
- area di lavoro del sistema operativo (RAM),
- area di I/O (RAM),
- area dei programmi utente (RAM durante lo sviluppo, poi PROM o EPROM),
- area dei dati utente (RAM).

La memoria a disposizione dei programmi utente varia tipicamente da circa mezzo kiloword a qualche centinaio di kiloword, con word di 8 o 16 bit.

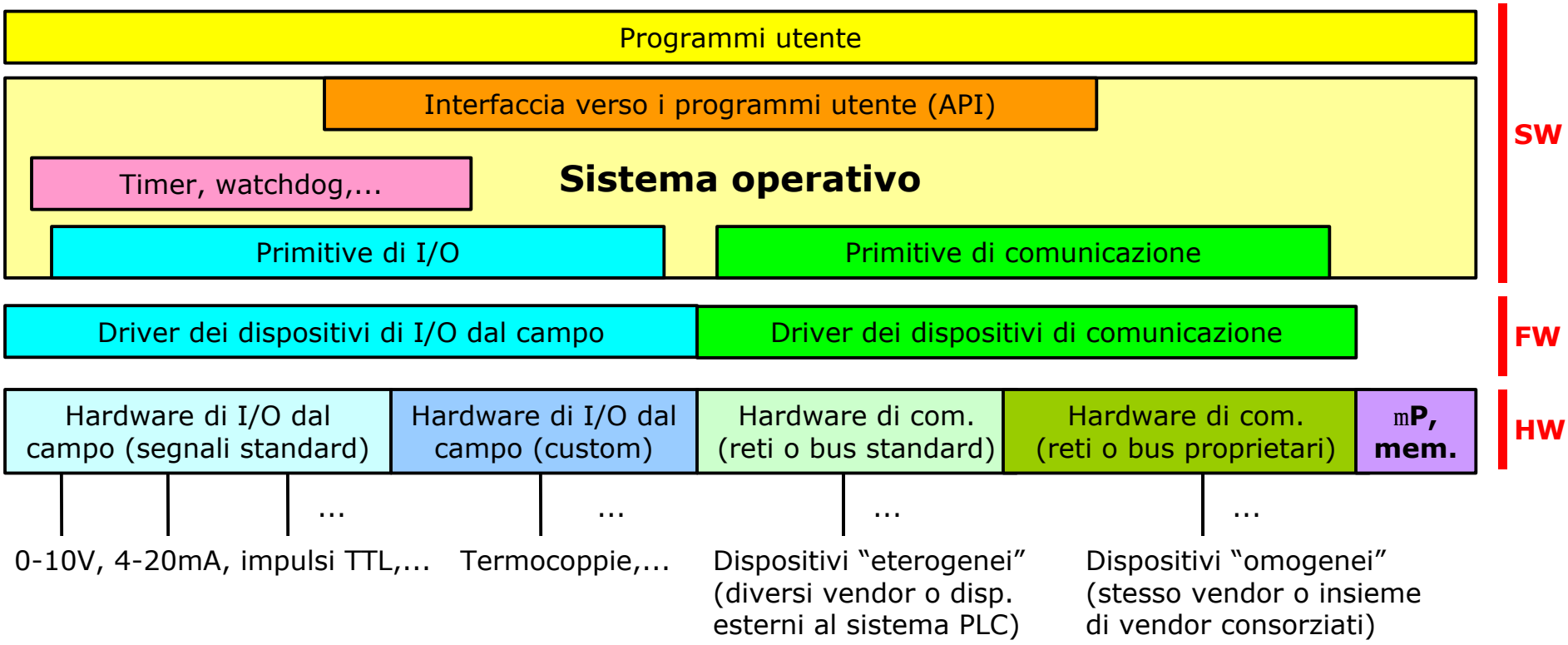
Altre caratteristiche importanti di un PLC sono

- l'espandibilità della memoria,
- il numero di moduli di I/O collegabili direttamente o in remoto,
- il numero e il tipo di porte di comunicazione disponibili (seriali, parallele, di rete),
- i linguaggi supportati (torneremo su questo punto),
- le capacità ed eventualmente le modalità di multitasking,
- la possibilità di gestire interrupt.

PLC e standard IEC 1131-3
- introduzione al PLC -

Nonostante abbiano una struttura apparentemente semplice, dal punto di vista hardware/firmware/software i PLC sono oggetti piuttosto complessi.

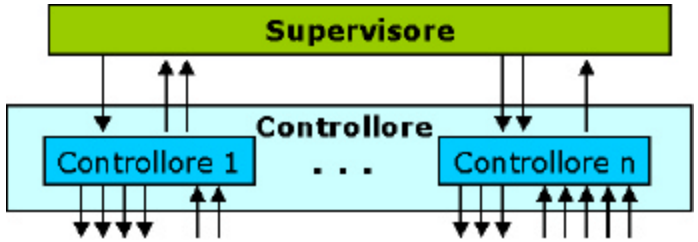
Proviamo a tracciare uno schema **molto semplificato** della situazione, limitando il campo ad una CPU.



E' ovvio che le cose sarebbero ancora più complesse se considerassimo (cosa che però esula dal corso) un sistema composto da più rack con varie CPU, dispositivi di rete, di I/O remoto, d'interfaccia operatore, di programmazione e così via.

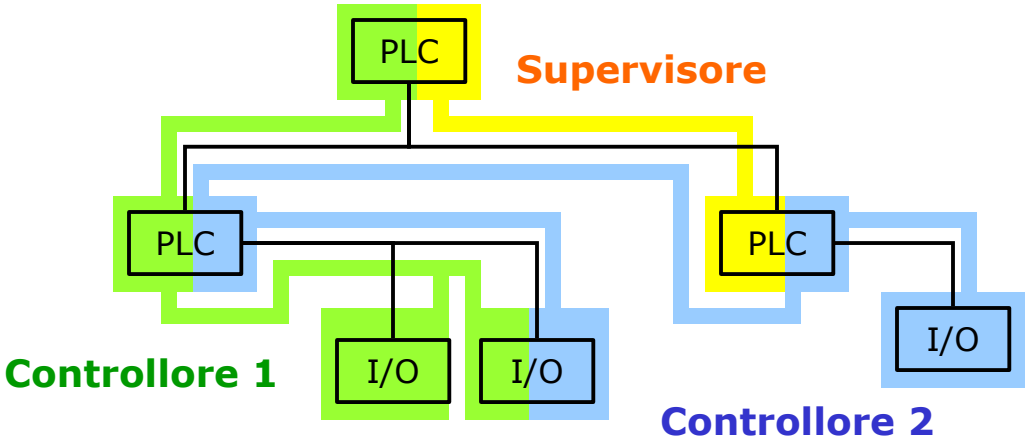
PLC e standard IEC 1131-3
- lo standard IEC 1131 -

Abbiamo già delineato uno schema **concettuale** di com'è organizzato il controllo logico di un generico impianto, il che ci ha condotto in modo naturale ad una struttura gerarchica che d'ora in poi chiameremo struttura logica del controllo.



Parallelamente, però, esiste un'altra struttura, ovvero quella che connette tra di loro tutti i dispositivi formanti il controllore. Chiameremo questa struttura fisica del controllo(re).

Possiamo quindi dire che la struttura logica si mappa su quella fisica e viceversa.



Palesemente, la stessa struttura logica si può implementare con strutture fisiche differenti in tutto o quasi, dal numero di CPU al tipo di rete e così via.
Occorre quindi una buona dose di sistematicità nel correlare tra loro le due strutture, perché la fase di dimensionamento e successiva configurazione della struttura fisica ha grande importanza.

PLC e standard IEC 1131-3
- lo standard IEC 1131 -

Senz'alcuna pretesa di esaustività né alcun ordine particolare, proviamo ad immaginare quali possono essere alcuni dei problemi più importanti.

A volte non si ha libertà completa nello scegliere la struttura fisica, tipicamente quando alcuni "pezzi" sono già installati nelle macchine dai loro produttori, oppure quando si affronta il revamping di un impianto già esistente.

A volte è necessario cambiare fornitore di tutto o parte del sistema hw/sw usato.

Quasi sempre il sistema deve comunicare con dispositivi che non sono PLC, tipicamente per integrarsi col sistema informativo.

Spesso i programmatori non sono specialisti del controllo, per cui nel dar loro le specifiche occorre essere assolutamente univoci.

Spesso occorre iniziare a verificare la funzionalità del sistema quando la struttura fisica non è ancora del tutto definita, o se lo è potrebbe cambiare.

Si capisce che c'è bisogno di uno standard…
…anche se non tutti i produttori sono completamente d'accordo.

Tuttavia, da alcuni anni (molto dopo la nascita del PLC, quindi) uno standard c'è.

PLC e standard IEC 1131-3
- lo standard IEC 1131 -

La normativa IEC 1131 definisce uno standard per il controllo logico.

Essa riguarda lo sviluppo di sistemi di controllo basati su PLC ed ha come obiettivi primari la correttezza, la qualità e il contenimento del costo dei sistemi medesimi.

La normativa si sviluppa secondo alcune linee guida:

- definire modelli, concetti e terminologia comuni;
- definire un riferimento per la realizzazione di strumenti di sviluppo, verifica e simulazione dei sistemi di controllo;
- facilitare l'interazione tra progettisti e il riuso di elementi dei progetti;
- consentire la sopravvivenza dei progetti sviluppati a diverse generazioni tecnologiche dei prodotti (hw/sw) usati per implementarli.

PLC e standard IEC 1131-3
- lo standard IEC 1131 -

La normativa IEC 1131si occupa di vari aspetti relativi ai sistemi di controllo basati su PLC:

- le specifiche dei dispositivi,
- i linguaggi di programmazione,
- i protocolli per la comunicazione, …

In questo corso approfondiremo la parte **3** della normativa, ovvero quella che introduce il modello concettuale del software costituente un'applicazione di controllo ed i linguaggi di programmazione per PLC.

La standardizzazione dei linguaggi di programmazione è il punto cruciale soprattutto per quanto attiene alla portabilità dei progetti e all'interoperabilità dei prodotti.

Le linee guida della normativa a tale riguardo sono quindi le seguenti:

- favorire il progresso verso metodi moderni di sviluppo incoraggiando (e in un certo senso costringendo) gli sviluppatori ad applicare concetti di programmazione strutturata e modularità;
- favorire la portabilità del software;
- facilitare la verifica e il riuso del codice;
- ridurre costi e tempi di sviluppo e di adattamento degli sviluppatori a possibili nuovi sistemi.

PLC e standard IEC 1131-3
- lo standard IEC 1131-3 -

La normativa IEC 113-3 definisce cinque linguaggi di programmazione per i PLC, di cui tre grafici e due testuali.

Linguaggi grafici:

diagramma funzionale sequenziale	(SFC, Sequential Functional Chart),
linguaggio a contatti	(LD, Ladder Diagram),
diagramma a blocchi funzionali	(FBD, Function Block Diagram).

Linguaggi testuali:

lista di istruzioni	(IL, Instruction List),
testo strutturato	(ST, Structured Text).

Vi sono oggi diversi ambienti di sviluppo che supportano più di un linguaggio IEC 1131-3.

Alcuni di questi ambienti consentono anche di "mescolare" i linguaggi entro un progetto, ossia d'implementare alcune parti del progetto con un linguaggio ed altre con un altro.

L'aderenza allo standard è indice di portabilità del codice (anche se non implica la compatibilità del formato dei file).

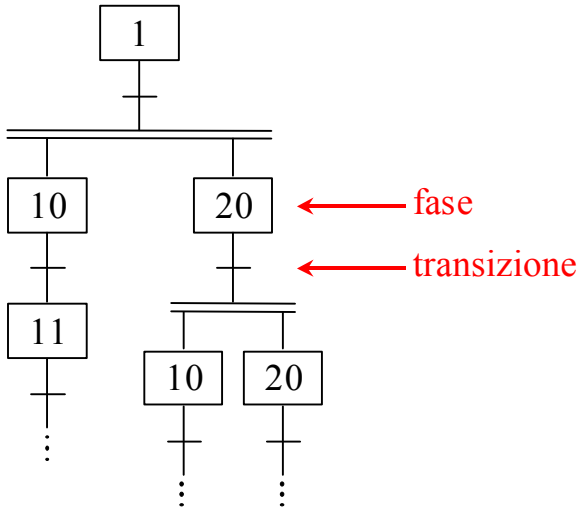
PLC e standard IEC 1131-3
- i linguaggi per la programmazione dei PLC -

Sequential Functional Chart.

Il Diagramma Funzionale Sequenziale (Sequential Functional Chart, SFC) è basato sui concetti di fase (entro cui si eseguono azioni) e di transizione (con cui, al verificarsi di certe condizioni, si passa da un certo insieme di fasi attive ad un altro).

E' un linguaggio palesemente derivato dalle reti di Petri e gerarchicamente superiore agli altri quattro, nel senso che le azioni possono essere programmate in uno degli altri linguaggi e che spesso SFC è usato come strumento di specifica.

Esempio di codice SFC:



PLC e standard IEC 1131-3
- i linguaggi per la programmazione dei PLC -

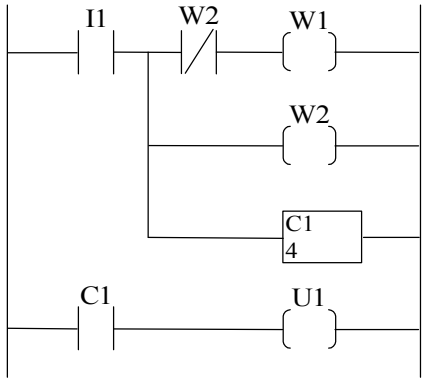
Ladder Diagram.

Il Linguaggio a Contatti o Diagramma a Scala (Ladder Diagram, LD) è derivato dai disegni dei sistemi di controllo realizzati relé elettormeccanici, il che lo rende facilmente accettabile a tecnici "di vecchia scuola".

Si basa sui concetti di contatto e bobina ed è stato inizialmente pensato per funzioni di logica binaria; poi è stato esteso per trattare anche numeri interi e/o reali.

E' un linguaggio di basso livello e poco strutturato, non molto adatto a sistemi complessi. Tuttavia è importante perché è stato il primo linguaggio grafico per PLC, è presente in tutti i PLC industriali ed è uno standard di fatto del mercato americano.

Esempio di codice LD:



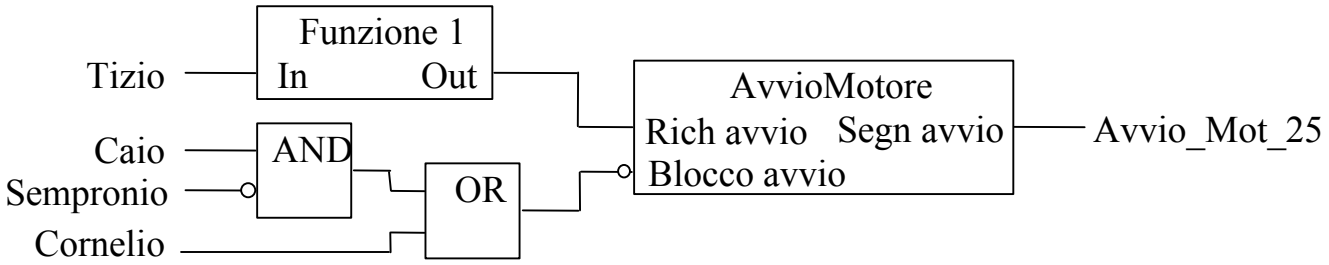
PLC e standard IEC 1131-3
- i linguaggi per la programmazione dei PLC -

Function Block Diagram.

Il Diagramma a Blocchi Funzionali (Function Block Diagram, FBD) può essere visto come analogo ai diagrammi circuitali, in cui le connessioni rappresentano i percorsi dei segnali tra i componenti.

Un blocco funzionale ha due caratteristiche principali, ovvero la definizione dei dati (ingressi e uscite) e un algoritmo che processa i valori correnti degli ingressi e delle variabili interne (locali o globali) e produce i nuovi valori delle uscite.

Esempio di codice FBD:



PLC e standard IEC 1131-3
- i linguaggi per la programmazione dei PLC -

Instruction List.

La Lista Istruzioni (Instruction List, IL) è un linguaggio di basso livello molto simile all'assembler. E' adatto per compiti molto specifici quali l'interfacciamento di hardware particolare. Anch'esso, come il LD, è disponibile per tutti i PLC.

Structured Text.

Il Testo Strutturato (Structured Text, ST) è un linguaggio testuale ad alto livello, simile al PASCAL o ad alcuni BASIC.

In questo corso studieremo il linguaggi LD e SFC, ponendo particolare attenzione a come essi sostanziano il modello del controllore espresso come rete di Petri e proveniente dai metodi di progetto studiati.

Il linguaggio LD (Ladder Diagram)

Il linguaggio LD
- introduzione -

Il linguaggio LD è un linguaggio grafico che si pone l'obiettivo di riprodurre in un paradigma di programmazione, per quanto una siffatta riproduzione è possibile, il funzionamento di una rete elettrica in cui gli utilizzatori (o bobine) sono o non sono alimentati a seconda dello stato di interruttori (o contatti).

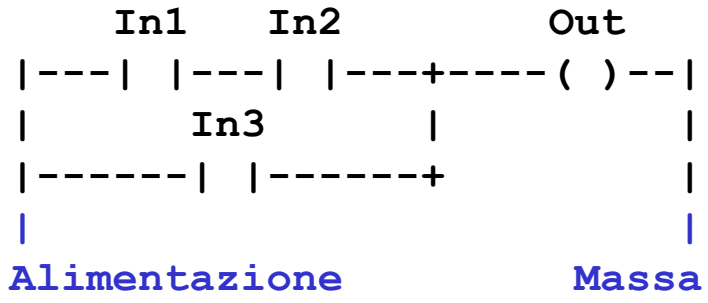
La motivazione alla base della nascita di LD (precedente a IEC1131-3) è "storica": LD è nato per far accettare l'dea di "programmare" (e quindi l'uso del PLC) a chi era abituato a fare i sistemi di controllo logico con i relé elettromeccanici.

Gli elementi fondamentali di LD sono quindi derivati proprio dagli schemi di controllo a relé elettromeccanici, e sono

- due linee verticali laterali dette montanti perché ricordano appunto i montanti di una scala a pioli (donde il nome LD) e che rappresentano un'alimentazione elettrica (il montante di sinistra è il polo positivo, quello di destra è la massa), e
- dei collegamenti orizzontali tra i montanti, detti pioli o rung, che contengono a sinistra dei contatti e a destra delle bobine.

Il linguaggio LD
- concetti fondamentali -

Il principio alla base di LD è elementare e lo si comprende con un semplice esempio:



In questo rung, la bobina **Out** è alimentata se sono chiusi i contatti **In1** ed **In2**, oppure se è chiuso **In3**.

Quindi, assumendo la convenzione che "alimentato" significhi "vero" e "non alimentato" significhi "falso", il rung corrisponde all'istruzione di assegnamento

```
Out = (In1 and In2) or In3
```

dove le variabili in gioco sono tutte booleane.

Come s'intuisce, LD è un modo di programmare vicino a chi è abituato a ragionare coi relé, anche se non è certo questo (come vedremo) il motivo per cui lo trattiamo nel corso.

Il linguaggio LD
- concetti fondamentali -

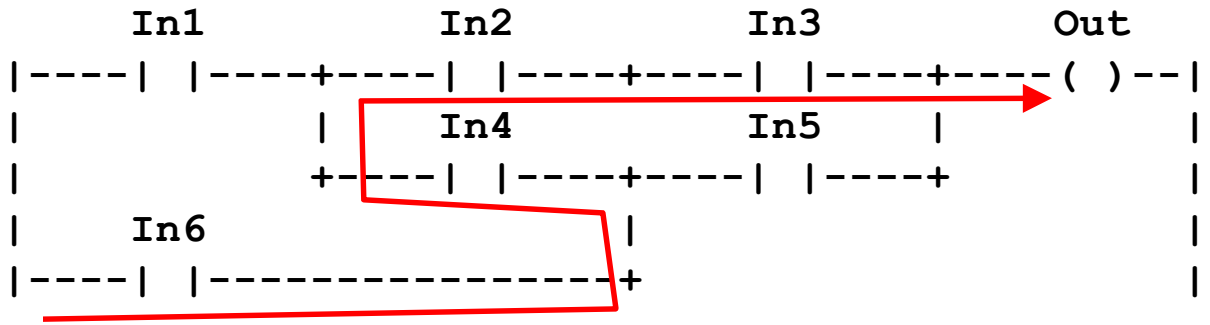
Tuttavia, la corrispondenza di un programma LD (per definizione sequenziale) con una rete elettrica (per natura sede di fenomeni simultanei) non può essere totale.

Per eliminare ogni possibile ambiguità a ciò conseguente, si stabiliscono quindi nella definizione di LD le regole enunciate nel seguito.

Regola 1.

La corrente può fluire nei contatti e nelle bobine soltanto da sinistra verso destra.

Esempio:



In questo rung il percorso di corrente indicato dalla freccia è vietato, col che il rung medesimo corrisponde senza possibili ambiguità all'istruzione

```
Out = (In1 and ((In2 and In3) or (In4 and In5))) or (In6 and In5)
```

Il linguaggio LD
- concetti fondamentali -

Regola 2.

I rung vengono esplorati dal PLC dal primo in alto all'ultimo in basso, e giunti all'ultimo si ricomincia dal primo.
Di conseguenza, l'ordine dei rung è rilevante (come lo è quello delle istruzioni in un programma e come non lo è quello di collegamenti elettrici in parallelo).

Regola 3.

La sincronizzazione delle variabili del programma con ingressi e uscite avviene secondo il principio della copia massiva:

- si leggono gli ingressi (che quindi ai fini del programma restano costanti per tutto il ciclo);
- si eseguono tutti i rung (a meno di salti, che vedremo poi) e si scrivono (cioè si assegna un valore a) tutte le bobine normali (ci sono anche quelle a ritenuta e anch'esse le vedremo poi), col che ogni bobina conserva il suo valore fino a che non viene riscritta in un ciclo successivo;
- si aggiornano le uscite;
- si ricomincia tutto da capo.

Osservazione importante.

LD è un linguaggio che descrive il ciclo operativo del PLC, nel senso che "in un programma LD è scritta la sequenza delle cose che il PLC deve fare ad ogni ciclo".

Il linguaggio LD
- istruzioni base -

Contatto normalmente aperto.

Simbolo: `---| |---`

Può essere associato ad un bit di ingresso (che si indica con **Ix:y**, intendendo il bit **y** della word **x**), ad un bit di uscita (**Ux:y**), ad un bit costituente una variabile booleana interna (**Wx:y**), oppure ad un bit di stato di un temporizzatore o di un contatore (che vedremo poi). Il nome del bit associato si pone sopra il contatto.

Quasi tutti i sistemi consentono di dare ai bit anche dei nomi simbolici oltre ai nomi legati all'indirizzo fisico di memoria come quelli sopra, per migliorare la leggibilità dei programmi.

Se il bit associato al contatto vale 1 ("vero") il contatto è chiuso e c'è continuità logica (elettrica), altrimenti il contatto è aperto e non c'è continuità.

Contatto normalmente chiuso.

Simbolo: `---|/|---`

E' del tutto analogo al contatto normalmente aperto, con la sola (ovvia) differenza che se il bit associato vale 1 il contatto è aperto, altrimenti è chiuso.

Il linguaggio LD
- istruzioni base -

Bobina (normale).

Simbolo: `---( )---`

Va inserita sempre alla fine del rung e può essere associata ad un bit di uscita (**Ux:y**) o interno (**Wx:y**), non ad un ingresso (il che non avrebbe senso).

La bobina si attiva quando vi passa corrente. Quindi, il bit ad essa associato vale 1 ("vero", "ON") se le condizioni logiche alla sua sinistra sono verificate, altrimenti vale 0 ("falso", "OFF"). Il nome del bit associato si pone sopra la bobina.

Bobina di tipo latch.

Simbolo: `---(L)---`

Quando si attiva, il bit associato va a 1 e vi resta finché non si attiva una bobina associata allo stesso bit e di tipo unlatch.

Bobina di tipo unlatch.

Simbolo: `---(U)---`

Quando si attiva, il bit associato va a 0 e vi resta finché non si attiva una bobina associata allo stesso bit e di tipo latch.

L'insieme di una bobina latch e di una unlatch associate allo stesso bit costituisce di fatto un flip-flop di tipo set-reset.

Il linguaggio LD
- istruzioni base -

Contatto a riconoscimento di fronte.

Simbolo: `---|P|---` (fronte positivo), `---|N|---` (fronte negativo)

Il contatto a riconoscimento di fronte **P** si chiude per un solo ciclo quando il bit ad esso associato passa da 0 ad 1; resta aperto in tutti gli altri casi.

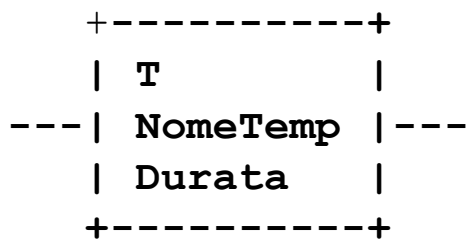
Il contatto a riconoscimento di fronte **N** si chiude per un solo ciclo quando il bit ad esso associato passa da 1 a 0; resta aperto in tutti gli altri casi.

Questi contatti semplificano a volte la programmazione, ma non sono disponibili in tutte le implementazioni del linguaggio LD.

Il linguaggio LD
- istruzioni di temporizzazione -

Temporizzatore (normale).

Simbolo:



Durata è un parametro, espresso di solito in centesimi o millesimi di secondo, e indica ovviamente quanto tempo il temporizzatore deve contare.

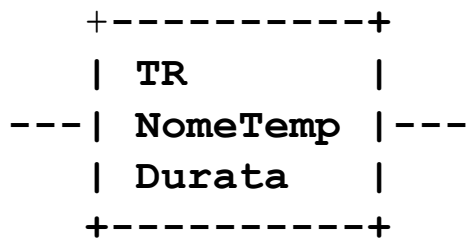
Se al temporizzatore giunge corrente (da sinistra), allora esso conta il tempo fino a raggiungere la durata impostata. A questo punto la variabile **NomeTemp** va a 1 e così rimane fino al reset del temporizzatore, che si ha quando cessa la continuità elettrica verso di esso. In ogni altro caso, **NomeTemp** vale 0.

In ogni istante, il tempo contato dal temporizzatore è accessibile con un apposito nome simbolico automaticamente definito dal sistema, di solito **NomeTemp.acc** (**acc**umulated time).

Il linguaggio LD
- istruzioni di temporizzazione -

Temporizzatore a ritenuta.

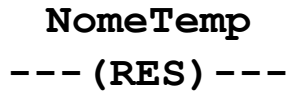
Simbolo:



E' del tutto analogo al temporizzatore normale, con la sola differenza che se viene meno la continuità (da sinistra) il temporizzatore a ritenuta non si resetta, bensì ferma il conteggio del tempo al valore raggiunto in quel momento. Quando la continuità ritorna, il conteggio riprende quindi dal valore raggiunto in precedenza.

Anche qui il tempo contato è accessibile in ogni istante con un apposito nome simbolico, di solito **NomeTemp.acc**.

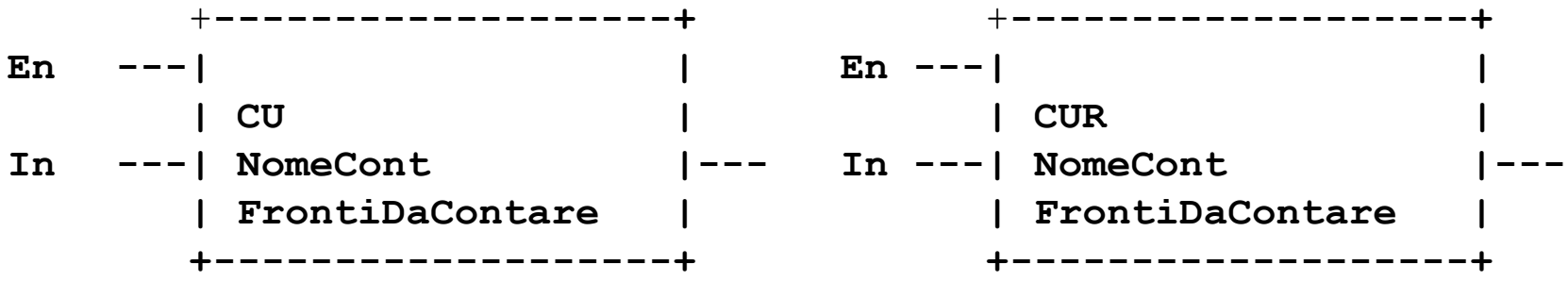
Per resettare il temporizzatore si usa l'apposito comando



Il linguaggio LD
- istruzioni di conteggio -

Contatore (normale) e contatore a ritenuta ad incremento (Up).

Simbolo:

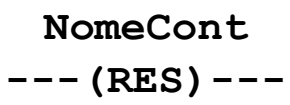


Sono analoghi ai temporizzatori, con la sola differenza che vengono contati i fronti di salita (da 0 ad 1) dell'ingresso **In**, fino al numero **FrontiDaContare**, se c'è continuità elettrica all'ingresso **En** (Enable, ossia abilitazione del conteggio).

Il numero di fronti contati è accessibile in ogni istante con un apposito nome simbolico, di solito **NomeCont.acc**.

Ci sono anche i contatori a decremento (indicati con **CD**, **CDR**), che contano da **FrontiDaContare** a zero.

Per resettare i contatori a ritenuta al valore iniziale del conteggio si usa l'apposito comando



Il linguaggio LD
- istruzioni di controllo del flusso -

Salto (ad etichetta).

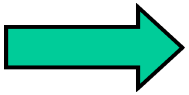
Simbolo: `---(JMP)---` e `---|LBL|---`

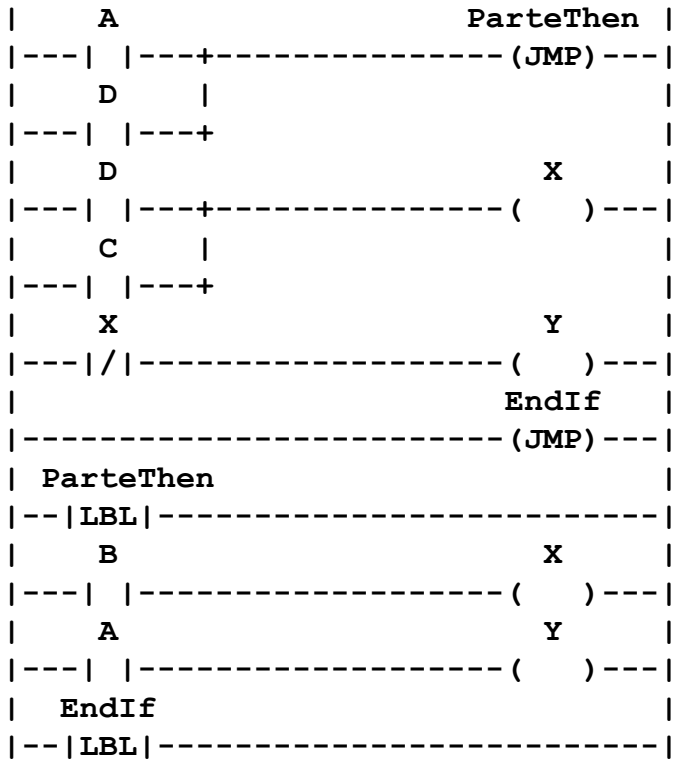
Se la bobina **JMP** è alimentata, il PLC salta al rung successivo a quello che contiene il solo elemento **LBL**.

Esempio.

Realizzazione in LD del codice

```
if (A or D) {
    X = B; Y = A;}
else {
    X = D or C; Y = not X;}
```





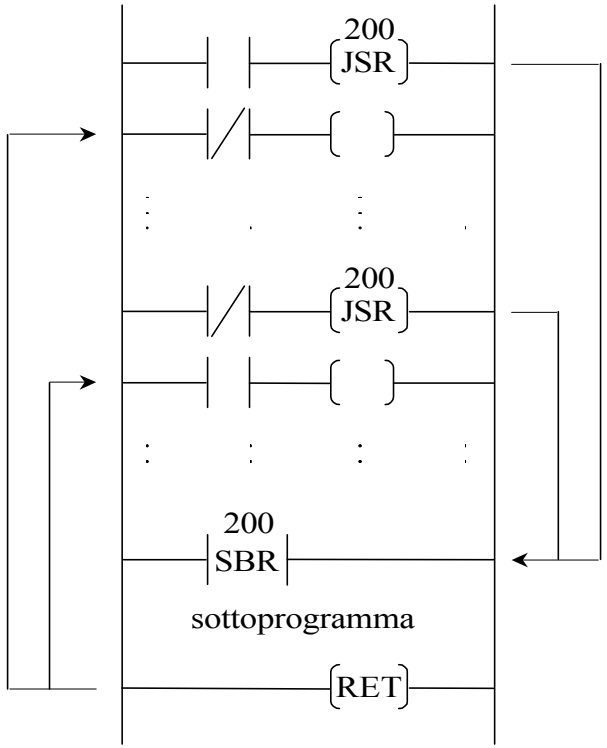
Il linguaggio LD
- istruzioni di controllo del flusso -

Salto a subroutine.

Simbolo: `---(JSR)---`, `---|SBR|---` e `---(RET)---`

Se la bobina **JSR** è alimentata, il PLC salta al rung successivo a quello che contiene il solo elemento **SBR**, prosegue fino a che incontra una bobina **RET** alimentata e quindi torna al rung successivo a quello di **JSR**.

Esempio.



NOTE

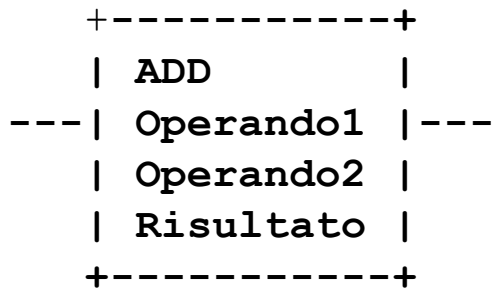
La porzione di codice tra **SBR** e **RET** non viene eseguita durante la normale esecuzione del programma, ma soltanto in seguito ad un salto a subroutine.

Si parla di "salto" e non di "chiamata" perché non c'è un meccanismo per il passaggio di parametri: il programma chiamante e la subroutine comunicano tramite le variabili, che sono globali al programma.

Il linguaggio LD
- istruzioni aritmetico/logiche -

Operazioni aritmetiche e logiche.

Simbolo (addizione):



Se al blocco giunge corrente, le variabili **Operando1** ed **Operando2** vengono sommate e il risultato è posto nella variabile **Risultato**.

Ogni ambiente di programmazione LD consente di dichiarare l'elenco delle variabili usate nel programma. I loro nomi, quindi, corrispondono a indirizzi di memoria che il compilatore risolve al momento di creare il codice eseguibile.

Naturalmente oltre ad **ADD** esistono **SUB** (sottrazione), **MUL** (moltiplicazione), **DIV** (divisione), **AND** (moltiplicazione binaria bit a bit), **OR** (addizione binaria bit a bit).

Alcuni PLC consentono di trattare solo numeri interi, altri anche i reali. Il tipo di una variabile, ovviamente, si decide quando la si dichiara, e le successive istruzioni che la coinvolgono si comportano di conseguenza (ad esempio eseguendo una divisione intera). Su questo aspetto però ci sono differenze tra i prodotti ed è bene prestavi attenzione, dichiarare sempre esplicitamente i tipi e non mescolarli.

Il linguaggio LD
- istruzioni di comparazione e trasferimento -

Istruzioni di comparazione.

Simbolo (maggiore di):

```
    +-----------+
    | GRT       |
---| Operando1 |---
    | Operando2 |
    +-----------+
```

Il blocco si comporta come un contatto, che è chiuso se **Operando1** è maggiore di **Operando2** e aperto altrimenti.

Naturalmente oltre a **GRT** esistono **EQU** (uguale a), **NEQ** (diverso da), **GEQ** (maggiore o uguale a), **LEQ** (minore o uguale a), **LES** (minore di).

Istruzione di trasferimento.

Simbolo:

```
    +--------+
    | MOV    |
---| OpSorg |---
    | OpDest |
    +--------+
```

Se al blocco giunge corrente, il contenuto di **OpSorg** viene trasferito (ovvero copiato) in **OpDest**.

Il linguaggio LD
- miscellanea -

Istruzioni di comunicazione via rete.

Esistono istruzioni di **SEND** e **GET**, rispettivamente per inviare e ricevere dati (ovvero blocchi di word in memoria) tra PLC. La sintassi di tali istruzioni è una delle parti meno standardizzate di LD, anche perché fa uso della rete ed è proprio nei relativi protocolli che si concentrano molti degli aspetti proprietari dei diversi prodotti.

Ai fini di questo corso non ci diffondiamo oltre, salvo notare che non tutte le primitive di comunicazione implicano la sincronizzazione dei PLC coinvolti: si può mettersi in attesa che qualcuno spedisca (e allora la sincronizzazione c'è) oppure semplicemente leggere dalla memoria di quel qualcuno (e allora si trova quel che al momento vi è scritto, senz'alcuna sincronizzazione).

Sempre più diffuso, per favorire la chiarezza, è dunque il cosiddetto modello a memoria condivisa (shared memory). In esso lo spazio d'indirizzamento di tutti i PLC in rete è unico, e quindi è come se più CPU condividessero la stessa memoria. Questo consente, assunto perfetto il meccanismo fisico con cui lo si implementa (ipotesi che ovviamente noi facciamo) di trattare i problemi di comunicazione come accesso a memoria condivisa.

Nei moderni sistemi di sviluppo per PLC questo è il modello di comunicazione di gran lunga più diffuso. Di conseguenza, la dichiarazione delle variabili avviene a livello di sistema e ognuna di esse, nel suo indirizzo, ha anche l'indirizzo di rete del PLC cui logicamente appartiene.

Il linguaggio LD
- osservazioni -

Il linguaggio LD, come si è detto, è stato creato assai prima dell'introduzione della normativa IEC1131-3.

Come facilmente s'immagina, quindi, di LD esistevano vari "dialetti", come accade per tutti i linguaggi che incontrano il successo prima di entrare a far parte di uno standard (si pensi al C con gli standard ANSI, K&R e così via).

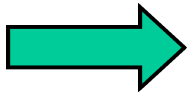
Conseguenze.

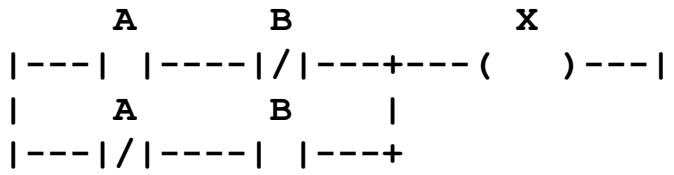
- Essendo l'introduzione di uno standard sempre e comunque il risultato di un negoziato, soprattutto poi se l'oggetto normato è qualcosa su cui molte aziende già fondano un grande business, lo standard (nel nostro caso la normativa IEC 1131-3) è concepito in modo da fissare i principi senza impatto sui dettagli, che i diversi soggetti negozianti lo standard medesimo possono già aver realizzato in modi eterogenei.
- Quindi tuttora esistono vari dialetti LD, che differiscono per la morfologia e per qualche elemento sintattico ma non per la semantica.
- Nel corso si adotta quindi, per gli elementi dove esistono piccole differenze quali ad esempio i contatori, una notazione convenzionale che non è quella di nessun produttore specifico, ma che in una qualsiasi di tali notazioni si trasforma in modo assolutamente ovvio.

Il linguaggio LD
- alcuni esempi semplici -

Esempio 1.

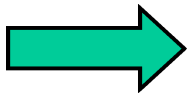
Realizzare in LD la funzione logica combinatoria **X = A xor B**.

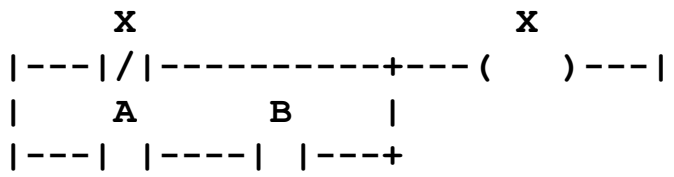




Esempio 2.

Realizzare in LD la funzione logica sequenziale **X = (not X) or (A and B)**, dove ovviamente (trattandosi di un assegnamento come è vero per qualsiasi istruzione di programma) la **X** a destra del segno di uguale indica il valore precedente di **X**, essendo quindi questo il fatto che rende sequenziale la funzione.

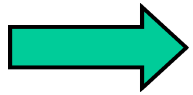
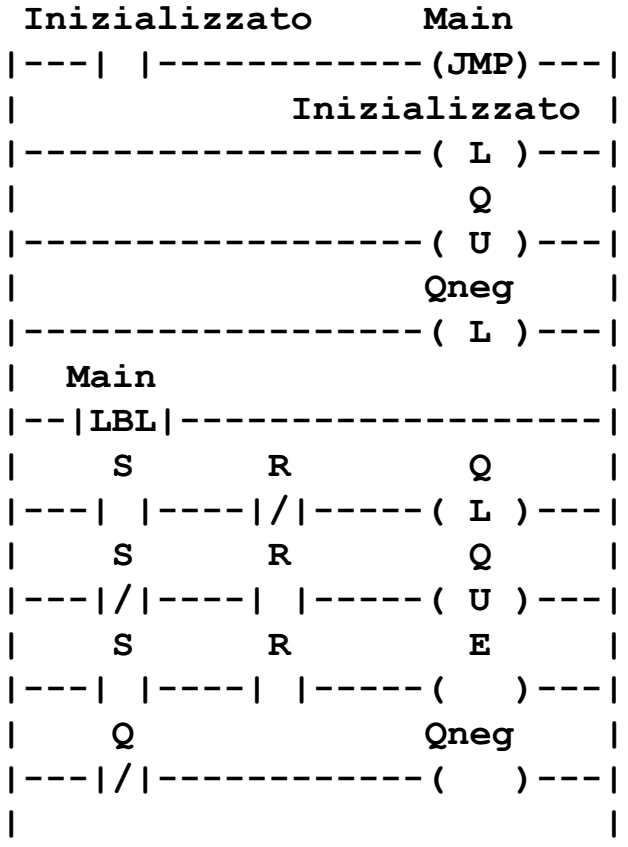




Il linguaggio LD
- alcuni esempi semplici -

Esempio 3.

Scrivere il programma LD che implementa un flip-flop con due ingressi **S** e **R**, due uscite dati complementari **Q** e **Qneg** ed un'uscita di errore **E**, descritto dalla tabella della verità seguente:

S	R	Q	Qneg	E
0	0	Q	Qneg	0
1	0	1	0	0
0	1	0	1	0
1	1	Q	Qneg	1

Inizialmente, **Q** dev'essere posto a 0 e **Qneg** a 1. Si assuma che alla prima esecuzione del programma tutte le variabili booleane siano a 0, come è per default in tutti i sistemi di sviluppo.