Introduzione alla norma IEC 61131-3

Marco Mauri

Politecnico di Milano, Dipartimento di Meccanica

Via La Masa 34

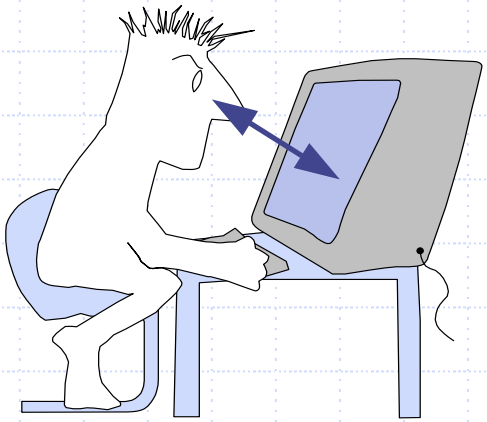
Tel. 02-2399 8377 Fax: 02-2399 8492

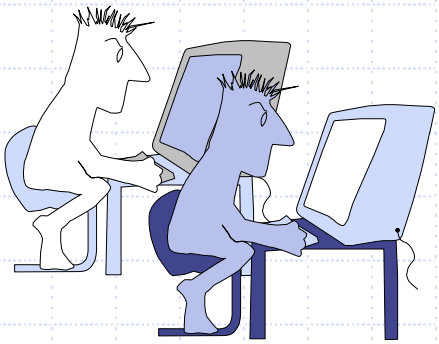
E-mail: marco.mauri@polimi.it

Struttura della norma

- Part 1: General Information (1992)
- Part 2: Equipment requirements and tests (1992)
- ***Part 3: Programmable Languages (1993)***
- Part 4: User Guidelines (1995)
- Part 5: Messaging Service Specification (1998)
- Part 6: Communication via Fieldbus (-)
- Part 7: Fuzzy Control Programming (1997)
- Part 8: Guidelines for the Implementation of (1998)
  Languages for Programmable Controllers

IEC 1131-3 Programming languages



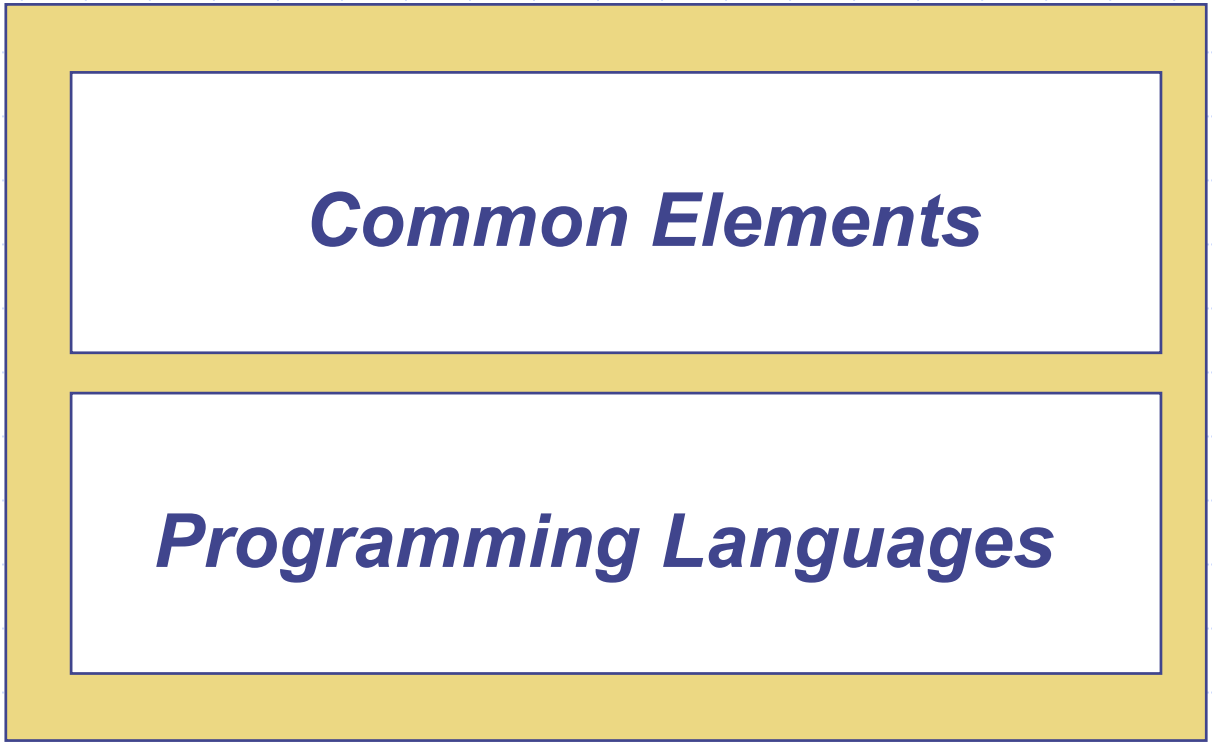


Un linguaggio di programmazione rappresenta l'interfaccia tra il programmatore e il sistema di controllo. In altre parole quello che vede sullo schermo

Persone con formazione differente usano differenti approcci al controllo.

La norma IEC 1131-3 fornisce il supporto adeguato per team con persone di differenti livelli o background

La norma IEC 61131-3



Elementi introduttivi

- Il set di caratteri che è possibile usare è definito nello standard ISO 646 "Basic Code Table"
- Gli identificatori che devono essere usati nel linguaggio:
  - Non ammettono come primo carattere un numero
  - Non ci sono 2 o più caratteri di sottolineatura consecutivi
  - Non sono ammessi spazi

✓ W123_PW W12_3PV aTemp1 _PROG1

✗ W123__PW W12 3PV 1aTemp Q%TY12

Tipi di dato

SINT	Short int.	8 bit	-128 a +127
INT	Integer	16 bit	-32768 a 32767
DINT	Double int.	32 bit	-2^{31} a $+2^{31}-1$
LINT	Long int.	64 bit	-2^{63} a $+2^{63}-1$
USINT	Unsigned sh. Int	8 bit	0 a 255
UINT	Unsigned Int	16 bit	0 a $2^{16}-1$
UDINT	Unsigned double Int	32 bit	0 a $2^{32}-1$
ULINT	Unsigned long Int	64 bit	0 a $2^{64}-1$

Se un tipo di dato è ambiguo si può far precedere da un prefisso

Esempio: **INT#12**

Tipi di dato

REAL	real number	32 bit	$\pm 10^{\pm 38}$
LREAL	Long real number	64 bit	$\pm 10^{\pm 308}$

STRING	Stringa di caratteri	(i.d.)

- Stringhe di bit

BOOL	stringa di 1 bit
BYTE	stringa di 8 bit
WORD	stringa di 16 bit
DWORD	stringa di 32 bit
LWORD	stringa di 64 bit

Tipi di dato

- Durata

TIME	time duration (i.d.) es: T#12d3h3s36ms

- Data e orario

DATE	data (i.d.)
TIME_OF_DAY	ora (i.d)
DATE_AND_TIME	data e ora (i.d.) es: D#1994-06-10 TOD#10:10:30 DT#2000-10-12-15:15:55.40

Tipi di dato

- La norma definisce per ogni tipo di dato un valore iniziale
- Per i tipi standard:
  - 0 per tutti i valori numerici
  - stringa vuota per le stringhe
  - 0001-01-01 per le date
- Nella dichiarazione di una variabile è comunque possibile sovrascrivere il valore iniziale di default del tipo

Tipi di dato

- Si possono definire alcuni tipi derivati
  - Semplici
  - Structured data types
  - Enumerated data types
  - Sub-ranges data types
  - Array data types
- Per i tipi derivati si possono definire valori iniziali che modificano il valore di default definito dalla norma.

Tipo di dato - DERIVATI

```
TYPE PRESSURE: REAL := 1.0
END_TYPE
```

Tipo derivato semplice

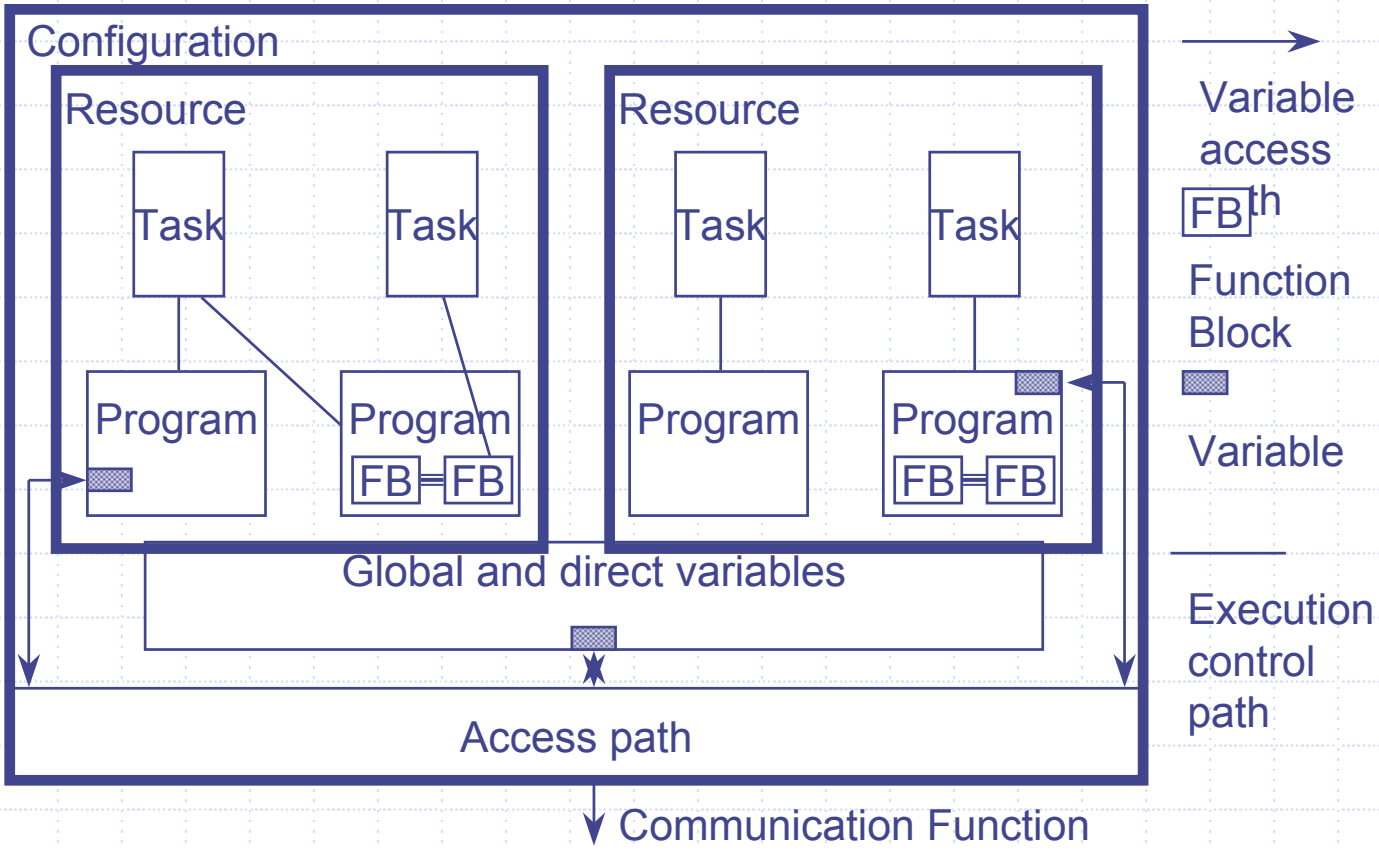
```
TYPE PRESSURE SENSOR:
    STRUCT
        INPUT:          PRESSURE := 2.0;
        STATUS:         BOOL:= 0;
        CALIBRATION:    DATE := DT#1994-01-10;
        HIGH_LIMIT:     REAL := 30.0;
        ALARM_COUNT:    INT := 0;
    END_STRUCT
END_TYPE
```

Tipo derivato strutturato

Tipo di dato - DERIVATI

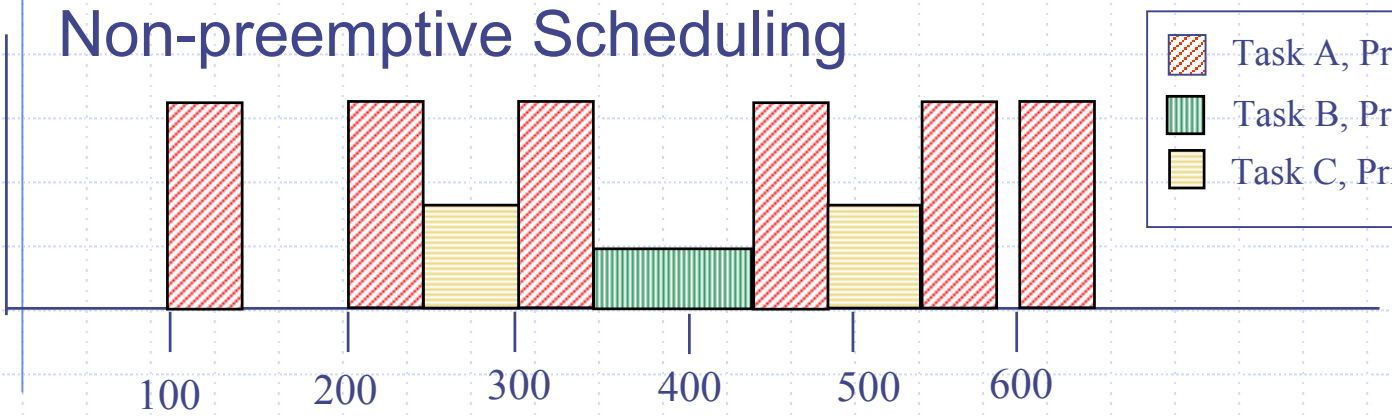
```
TYPE DEVICE_MODE:                                   Tipo derivato enumerativo
    (INITIALISING, RUNNING, STANDBY, FAULTY):= STANDBY;
END_TYPE

TYPE MOTOR_VOLTS:                                   Tipo derivato subrange
    INT(-6..+12);
END_TYPE

TYPE VESSEL_PRESS_DATA:                             Tipo derivato array
    ARRAY[1..20] OF PRESSURE
END_TYPE
```

Il modello software - FB

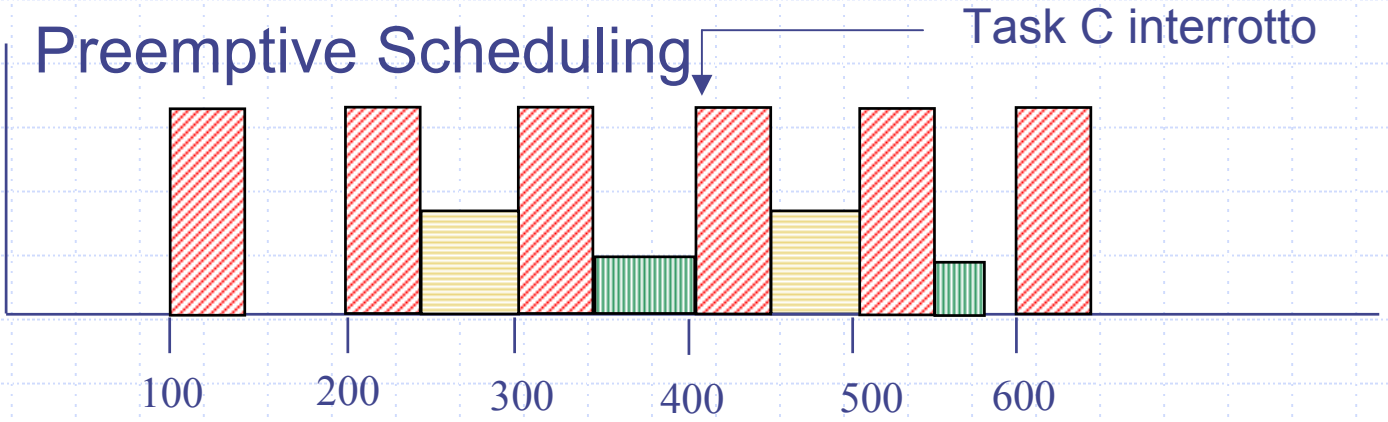


Il modello software - SCHEDULING

Non-preemptive Scheduling



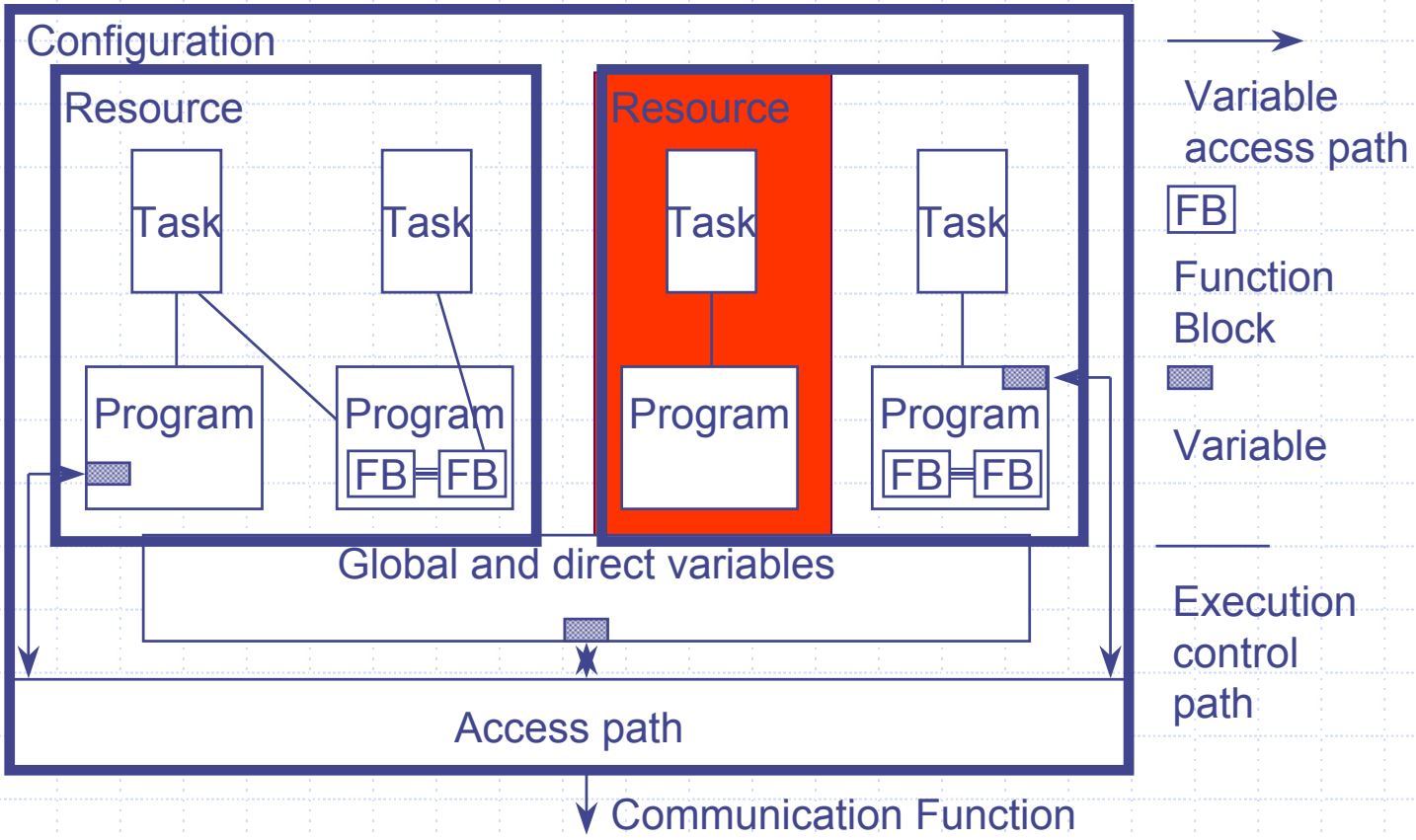
Preemptive Scheduling



Il modello software - POU

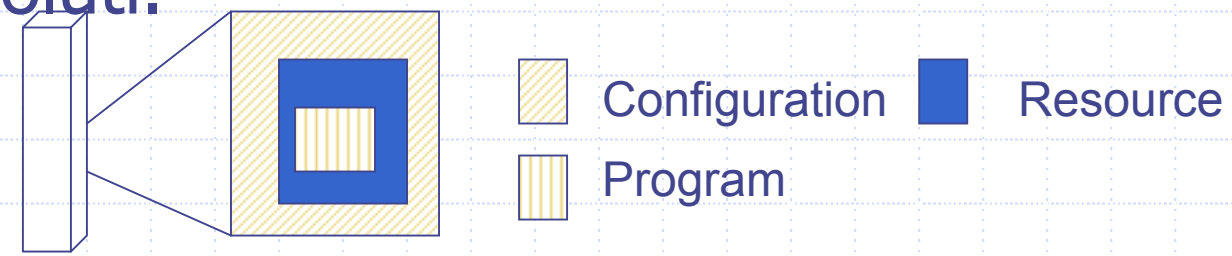
- Lo standard definisce program, function blocks e function come *Program Organisation Units (POU).*
- Le POU sono componenti dal comportamento ripetitivo che possono essere usate in differenti parti di una applicazione
- Le POU incoraggiano la riusabilità del codice dal macro-livello con i program al micro-livello con le function
- A differenza dei linguaggi di alto livello, la IEC 1131-3 impedisce l'utilizzo di POU ricorsive. Questo perché è difficile testare il software ricorsivo e non è predicibile il suo comportamento real-time.
- Program e Function blocks possono essere descritte con ST, IL, FBD, LD, SFC. Le function con ST, IL, FBC, LD.

Il modello software – DIFFERENZE



Il modello software – DIFFERENZE

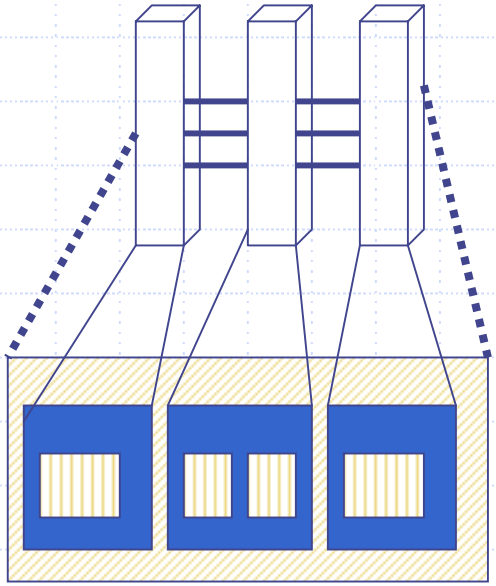
- La struttura tradizionale di un PLC era composta da una unica risorsa, un unico task e un unico programma.
- La struttura proposta dalla norma è molto più generale e si può applicare anche a sistemi molto evoluti.



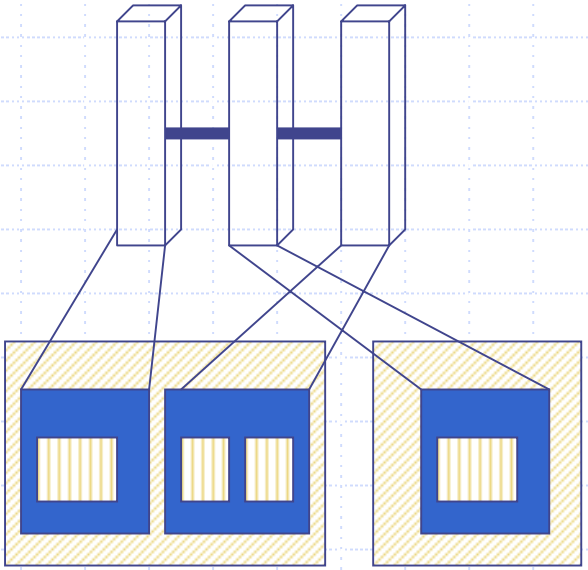
PLC singolo

Il modello software - MAPPING

Multi processor PLC



PLC distribuiti su una rete ad alta velocità



Configuration

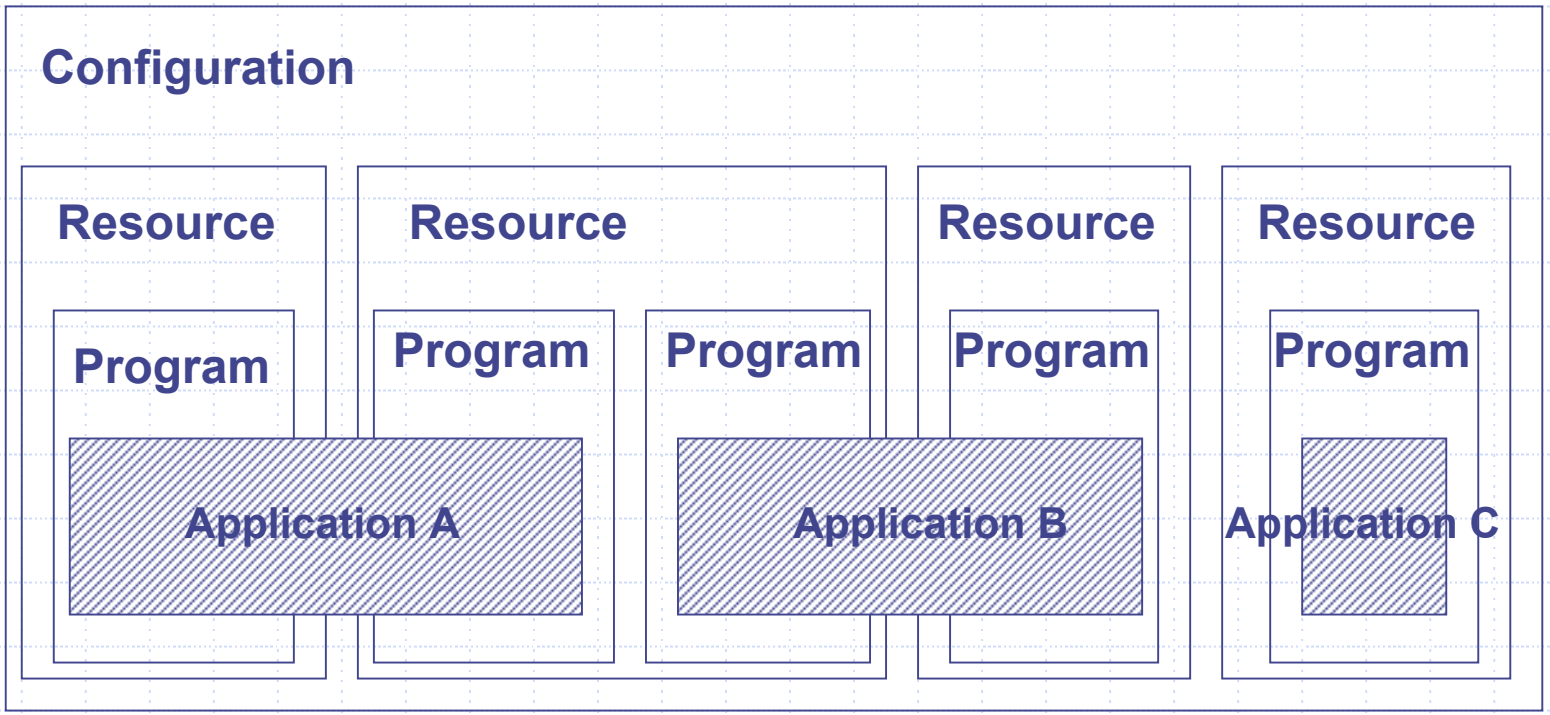
Resource

Program

Il modello software – APPLICAZIONE

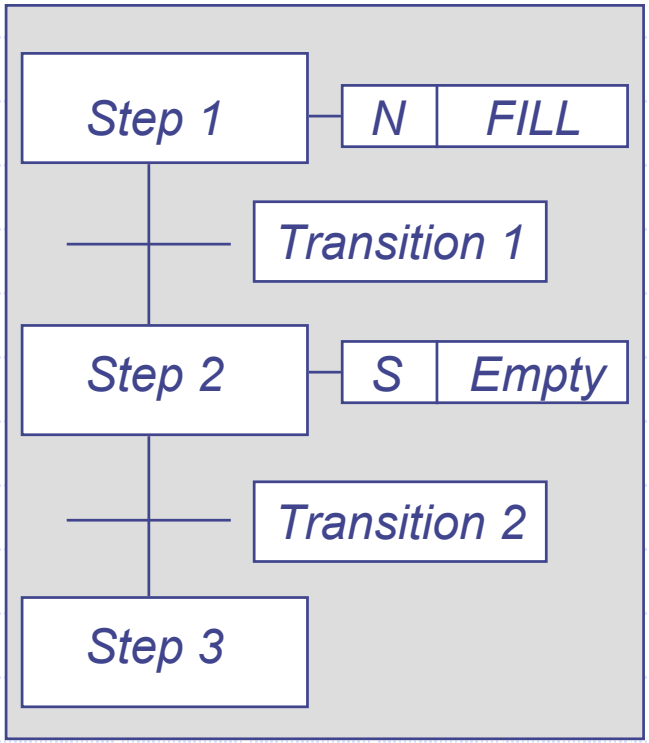
- Il concetto di **APPLICAZIONE** non è definito nella norma, ma è una importante parte del modello software se consideriamo PLC che sono in grado di controllare diversi parti del sistema.
- Un applicazione ad esempio potrebbe includere tutti i controlli necessari per lo start-up o lo shut-down di una parte del sistema
- Le applicazioni possono essere eseguite indipendentemente solo se caricate su differenti **RESOURCE**.

Il modello software – APPLICAZIONE

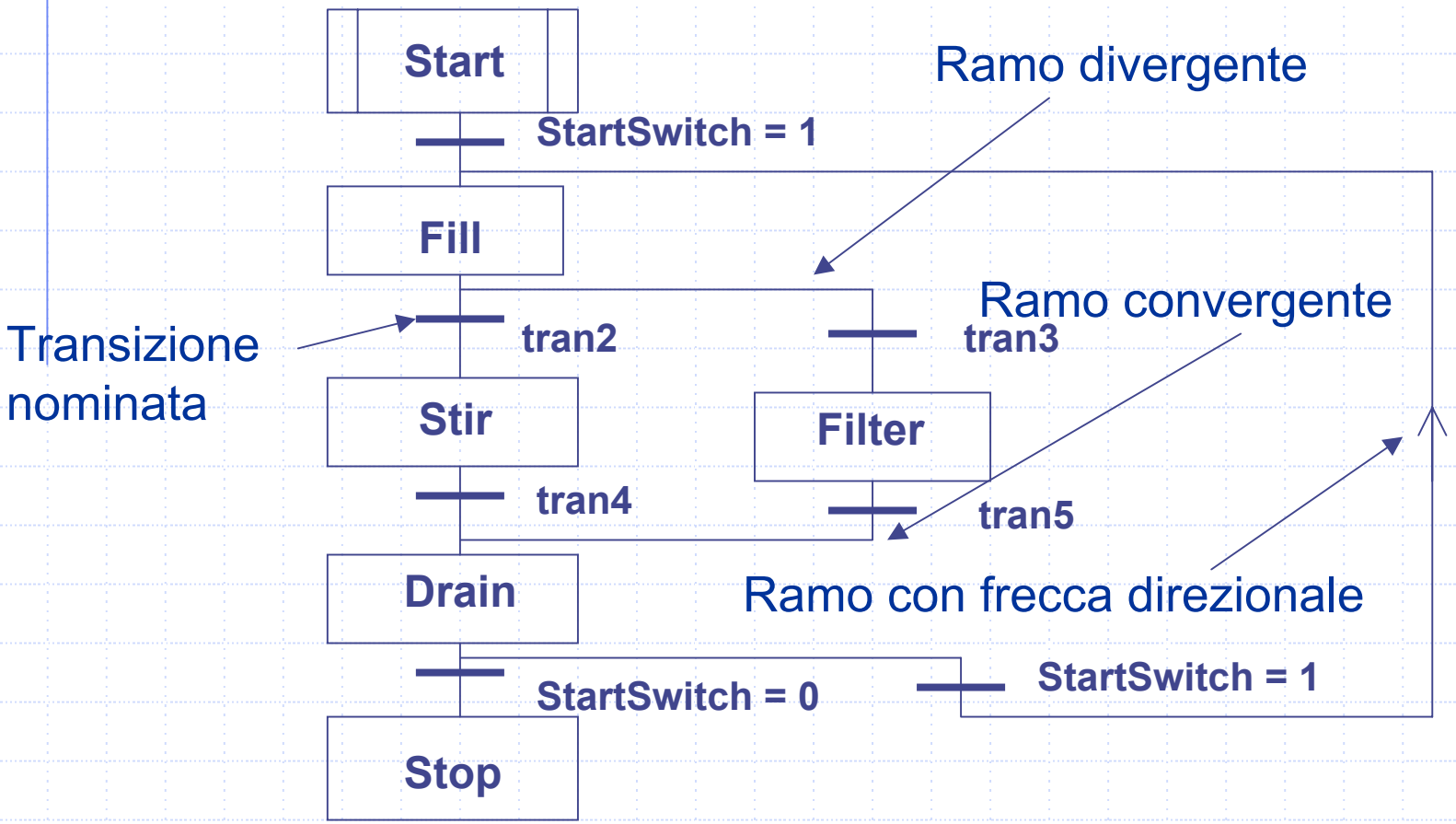


Elementi base

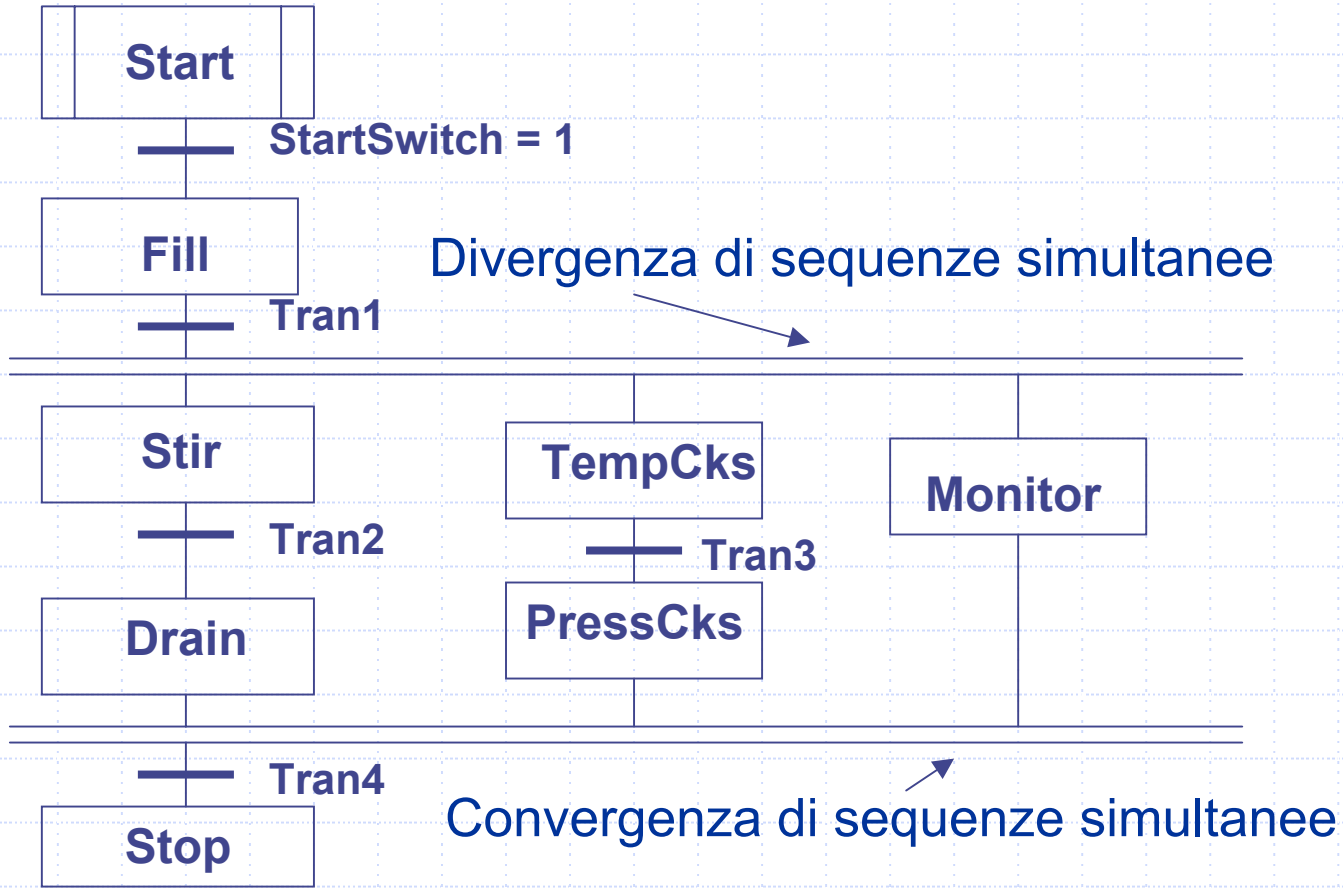
- SFC fornisce una potente tecnica grafica per descrivere il comportamento sequenziale di un programma di controllo.
- Utile per partizionare il problema di controllo
- Mostra una visione di insieme utile per una rapida diagnostica



Struttura del linguaggio



Sequenze simultanee

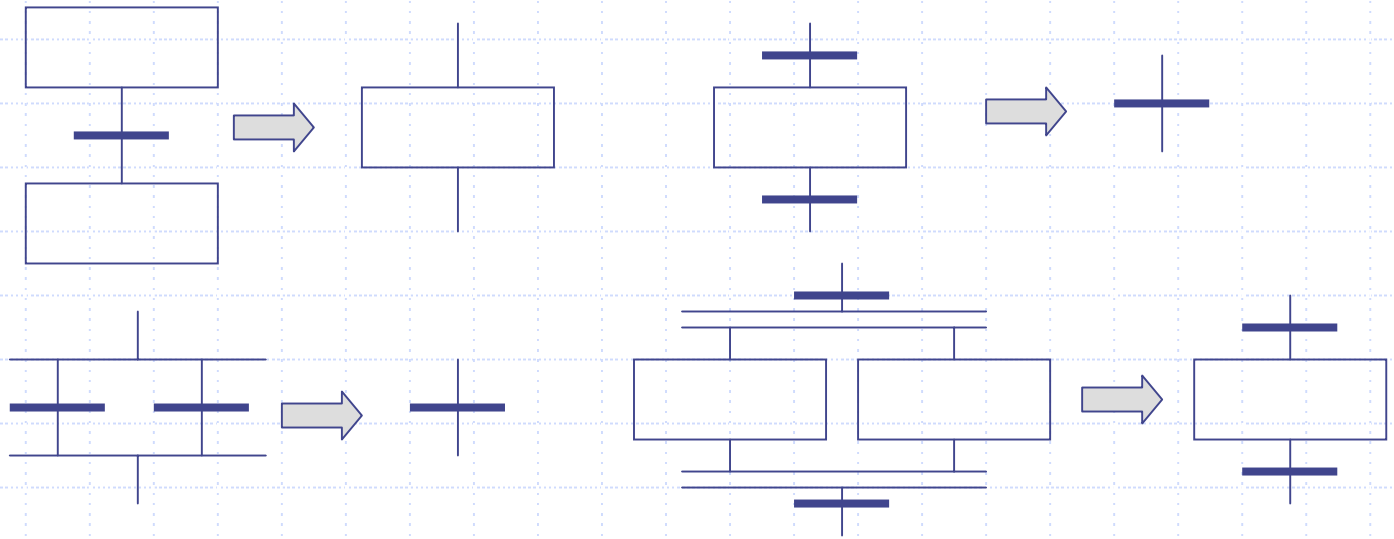


Sequenze simultanee

- Per passare al passo successivo dopo una convergenza è necessario che:
  - **tutte** le sequenze simultanee abbiano raggiunto **l'ultimo passo**
  - **sia verificata la condizione** della transizione posta dopo la convergenza

Scoprire SFC mal progettati

- E’ possibile scoprire in maniera automatica delle situazioni erronee applicando le seguenti regole:



Applicando queste regole deve essere possibile risolvere il diagramma con un singolo passo

IEC 1131-3: I linguaggi di programmazione

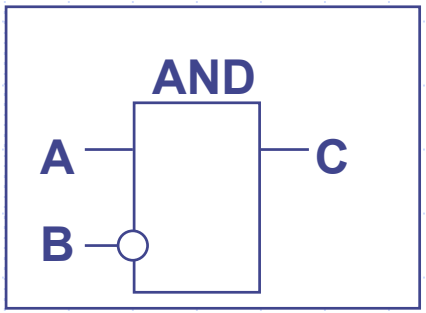
Instruction List

```
LD      A
ANDN    B
ST      C
```

Structured Text

```
C:= A   AND  NOT B
```

Function Block Diagram

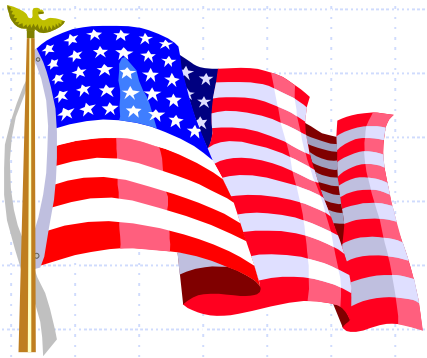


Ladder Diagram

```
A   B                  C
-| |--|/|---------------( )
```

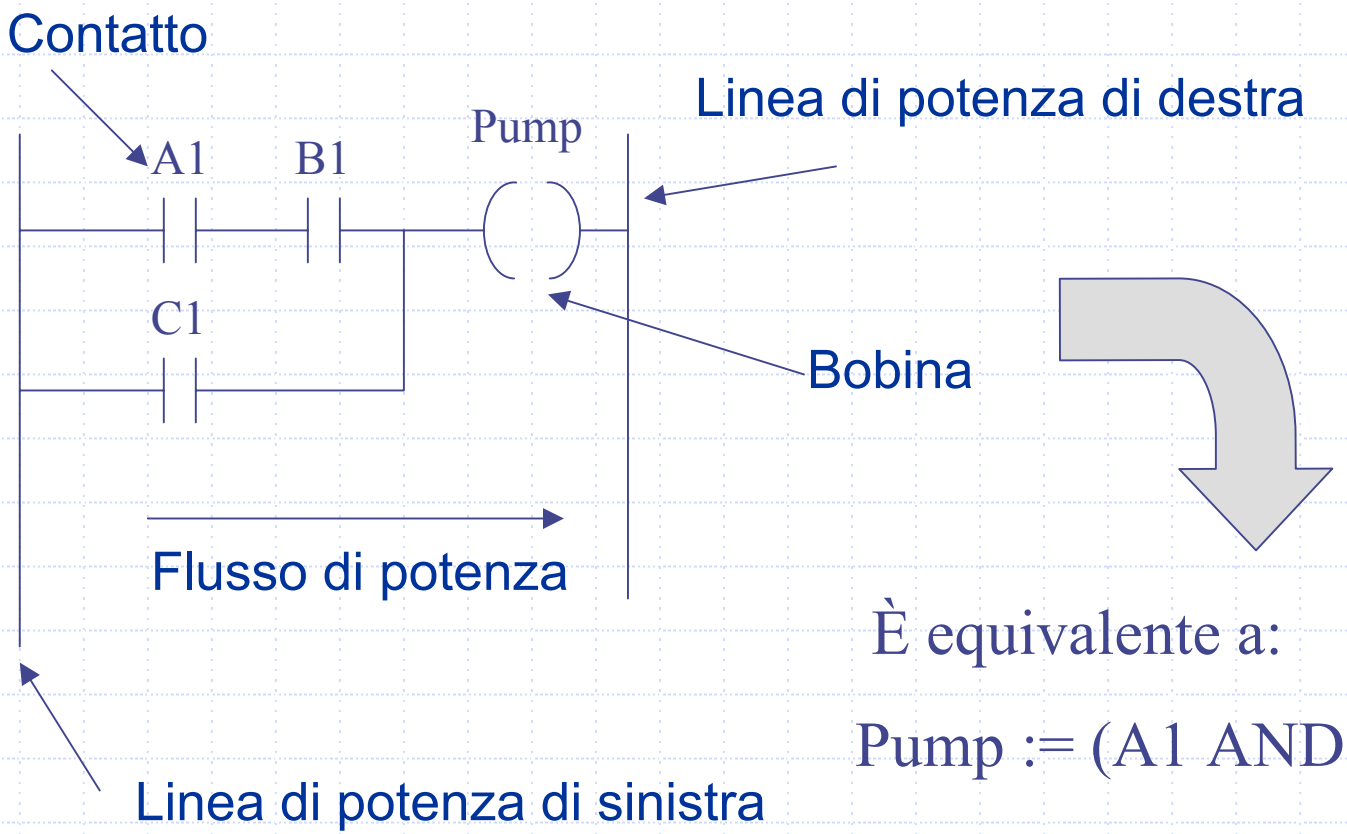
Ladder Diagram (LD)

- Basato su una diffusa notazione di origine Nord Americana che utilizza simboli elettrici standard.



```
A   B              C
-| |--|/|----------( )-
```

Struttura del linguaggio



È equivalente a:

Pump := (A1 AND B1) OR C1

Instruction List (IL)

- Modello di esecuzione basato su un singolo registro accumulatore.
- Basato sul linguaggio tedesco ‘Anweisungsliste’, AWL
- E’ consentita una sola operazione per linea

```
LD    A
ANDN  B
ST    C
```

La struttura

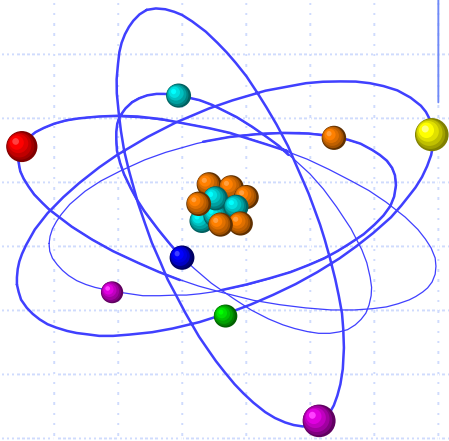
Carica il valore della variabile speed nel registro risultato

```
                LD      Speed     (* Load Speed and *)
                GT      1000      (* Test if > 1000  *)
                JMPCN   VOLTS_OK  (* Jump if not     *)
                LD      Volts     (* Load Volts and  *)
                SUB     10        (* Reduce by 10    *)
                ST      Volts     (* Store Volts     *)
VOLTS_OK:       LD      1         (* Load 1 and Store *)
                ST      %Q75      (* in output 75 *)
```

Label **Operatore** **Operando** **Commenti**

Structured Text (ST)

- Linguaggio strutturato di alto livello
- Presenta una sintassi simile al PASCAL
- Sono possibili strutture complesse e annidate
- Fornisce supporto per:
  - Cicli iterativi (REPEAT-UNTIL; WHILE-DO)
  - Esecuzioni condizionali (IF-THEN-ELSE; CASE)
  - Functions (SQRT(), SIN())



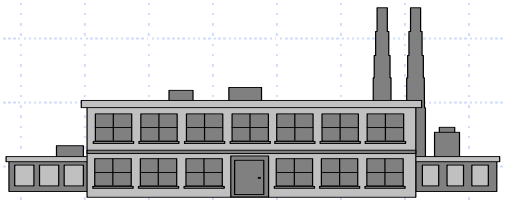
C:= A AND NOT B

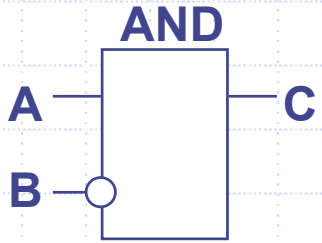
Esempio di ST

```
FUNCTION_BLOCK INTEGRAL
    VAR_INPUT
        RUN : BOOL;  (* 1 Integrate, 0 hold *)
        R1 : BOOL ;  (* Reset *)
        XIN : REAL;  (* Input per l'integrale *)
        X0 : REAL;   (* Valore Iniziale *)
        CYCLE : TIME;  (* Tempo di ciclo *)
    END_VAR
    VAR_OUTPUT
        Q : BOOL;   (* 1 = Not Reset *)
        XOUT : REAL; (* In, Out *)
    END_VAR
    Q := NOT R1;
    IF R1 THEN
        XOUT := X0;
    ELSEIF RUN THEN
        XOUT := XOUT+XIN*TIME_TO_REAL(CYCLE);
END_IF; END_FUNCTION_BLOCK
```

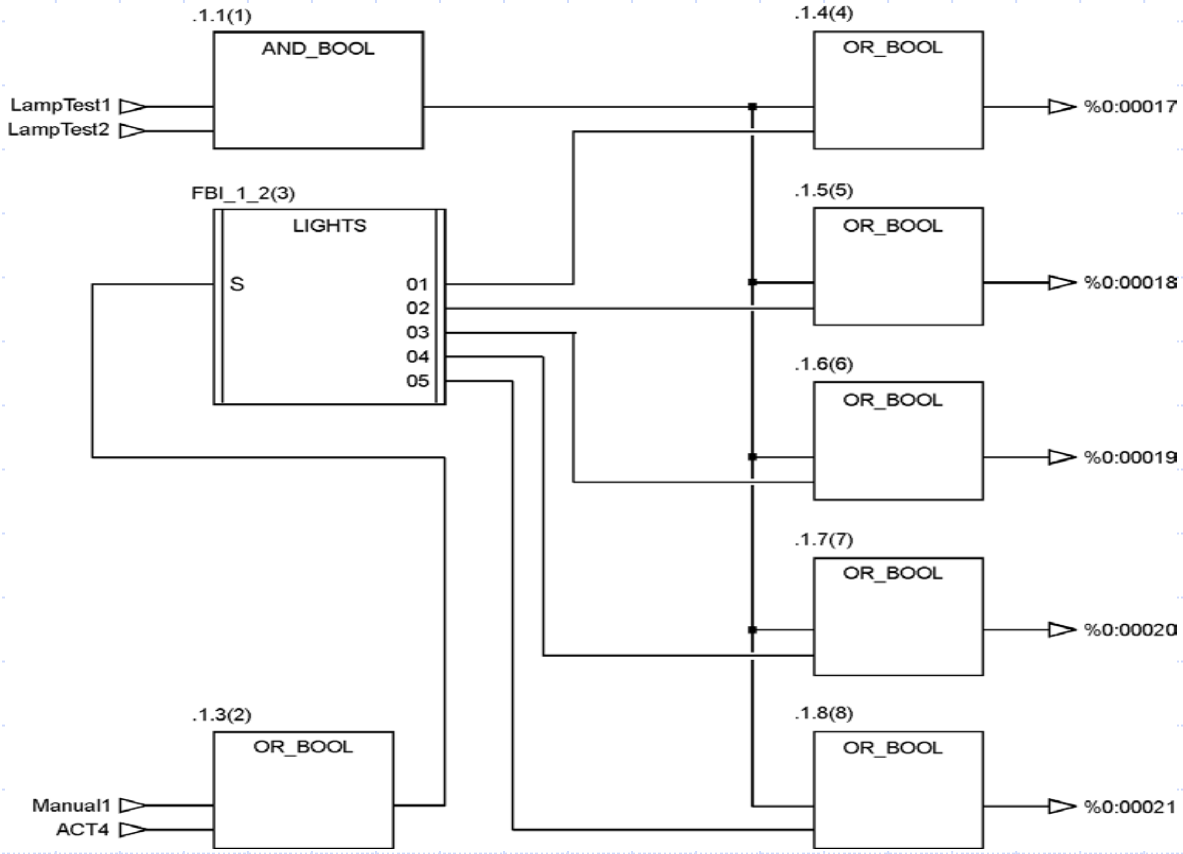
Function Block Diagram (FBD)

- Linguaggio grafico ampiamente usato in Europa
- Permette di programmare con elementi che appaiono come blocchi che possono essere connessi in maniera analoga a quello che si fa con un diagramma circuitale
- E' usato in molte applicazioni in cui è necessario evidenziare il flusso di informazioni o dati tra i diversi componenti del controllo

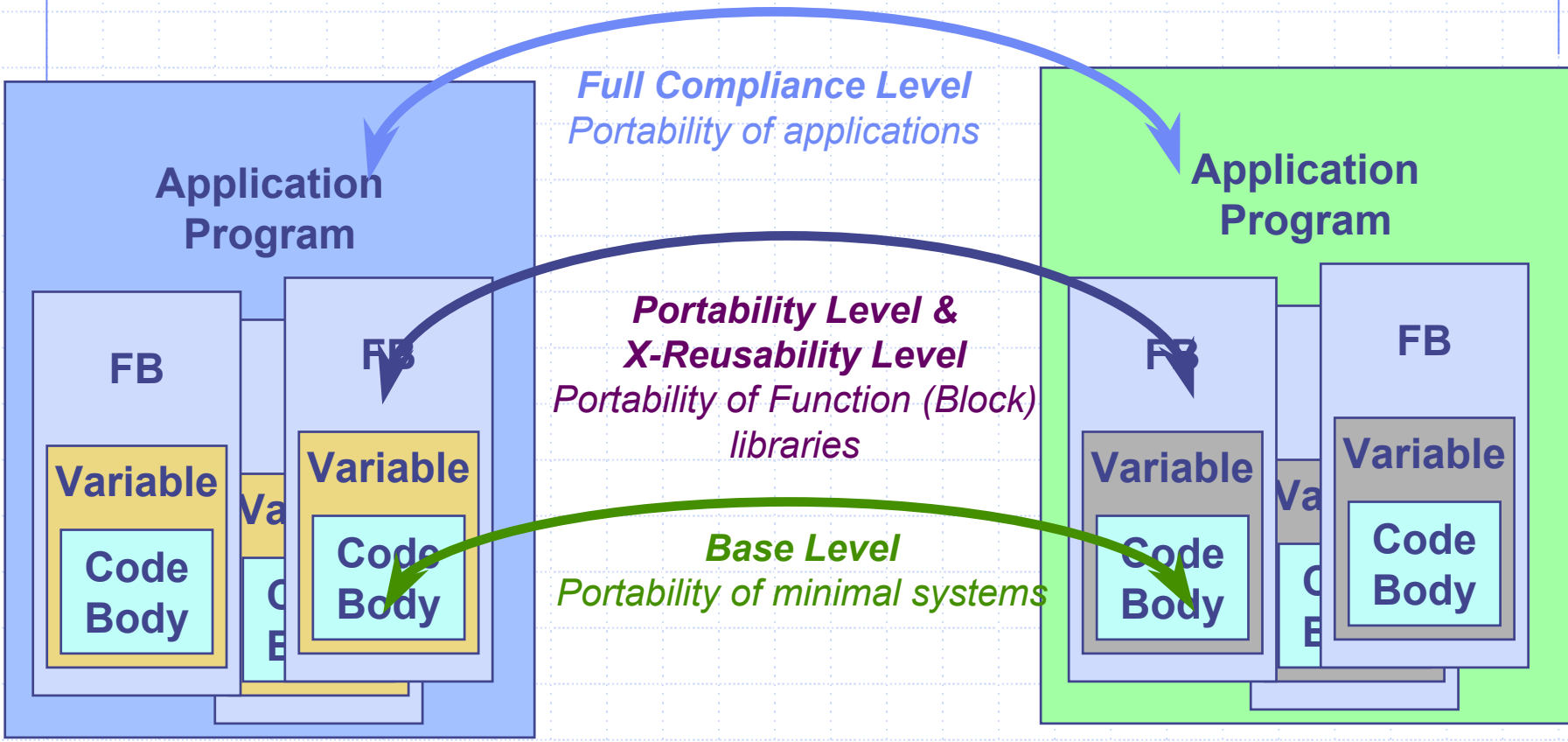




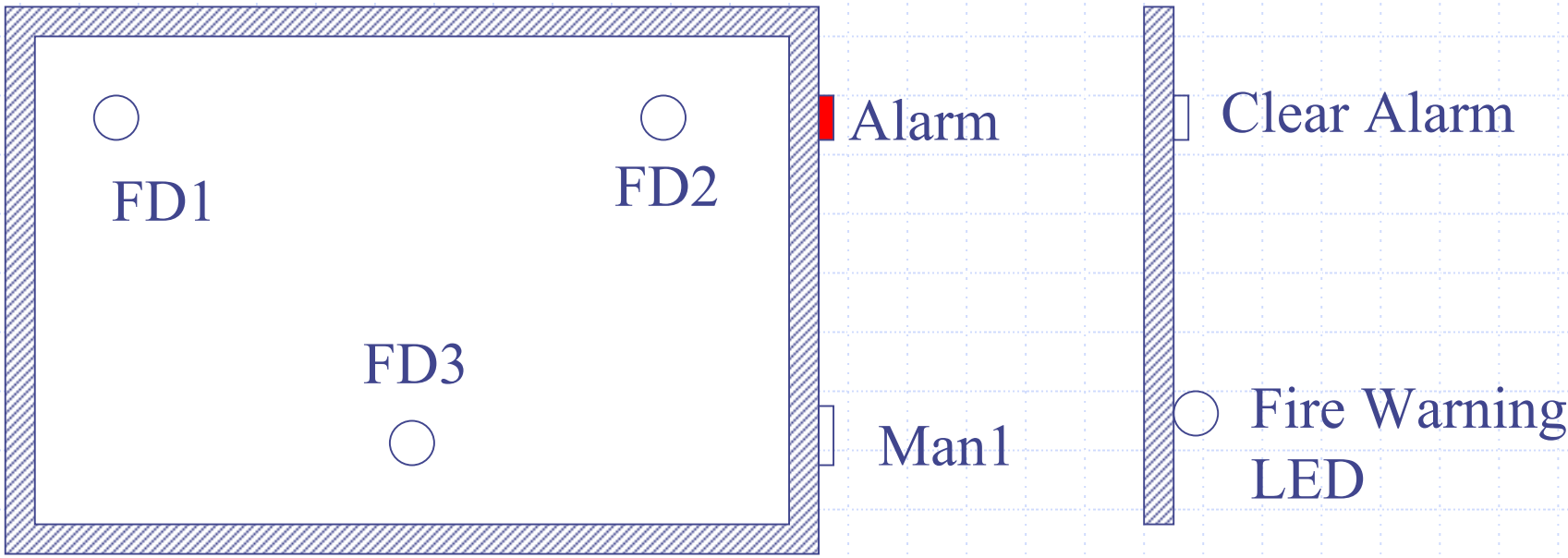
Esempio di FBD



Compliancy Levels



Esempio LD



Esempio LD

- Ipotizziamo di dover progettare un sistema di allarme incendi.
- Tre sensori di presenza fuoco FD1, FD2, FD3.
- C'è un bottone MAN1 che può essere usato per attivare l'allarme
- Per prevenire eventuali segnalazioni errate devono essere attivi almeno 2 sensori su 3 per attivare l'allarme. In caso sia abilitato solo un sensore viene acceso solo il led di segnalazione
- Il segnale d'allarme può essere resettato solo premendo il pulsante ClearAlarm

Esempio LD – Il Programma