# La norma IEC 61499

**Salvatore Cavalieri**

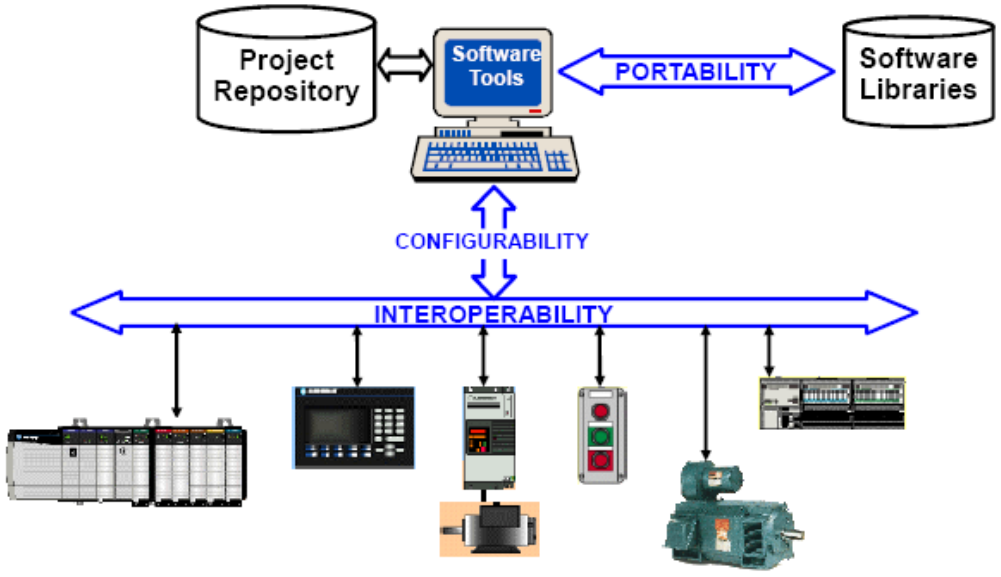
**Università di Catania, DIEEI**

**Documenti di Riferimento:**

- **Libro:** "IEC 61499: Uno standard per sistemi distribuiti di automazione industriale", Luca Ferrarini e Carlo Veber, Pitagora Editrice, Bologna, 2004, ISBN 88-371-1493-1
- **Dispensa sul sito: "**Workbook for Designing Distributed Control Applications using Rockwell Automation's HOLOBLOC Prototyping Software", John Fischer and Thomas O. Boucher

# Il Software nell'Automazione

- **Alcuni requisiti specifici per il software nell'automazione:**
  - Utilizzo di **modelli software indipendenti da SO e HW** al fine di una drastica riduzione dei costi di progettazione
  - **Integrazione di system tools** (design tool, programming tool, configuration tool)
  - **portabilità**: Ogni tool software può accettare e correttamente interpretare librerie prodotte da altri tool software
  - **configurabilità**: tutti i dispositivi e i loro componenti software devono poter essere configurati da diversi tool software (produttori diversi)



# Perché un nuovo standard ?

- IEC 61131-3 è troppo limitato a sistemi single-processor o poco distribuiti (variabili globali e pochi FB di comunicazione)
- Utilizza strumenti di modellazione basati su FB troppo legati al concetto di program scan (time-triggered e non event-triggered)
- Di fatto nel tempo la realizzazione di architetture aperte basate su IEC 61131-3 non è stata realizzata (vedi portabilità e configurabilità)

# IEC 61499: Concetti Generali

- Lo standard propone una **metodologia di progettazione/modellazione** di sistemi di controllo distribuiti

- Lo standard permette la creazione di dati e modelli **portabili e interscambiabili**, sia attraverso un linguaggio proprio definito source code sia attraverso XML

- Lo standard enfatizza i concetti e i metodi formali alla base di UML e degli approcci OO.
  - **riuso di codice**

# IEC 61499: Concetti Generali

- E’ basato su un modello event triggered.
  - Si basa sulla netta separazione tra eventi, dati e algoritmi.
  - L’esecuzione di un algoritmo è iniziata all’arrivo di eventi e l’algoritmo usa i valori correnti dei dati in ingresso disponibili all’istante in cui accadono gli eventi.
- La programmazione è basata su Function Blocks, che è un formalismo già definito da IEC 61131-3, ma che viene esteso per una migliore integrazione.
- Vengono potenziati i modelli di comunicazione in ambiente distribuito (client/server, publisher/subscriber)

# IEC 61499 vs. IEC 61131-3

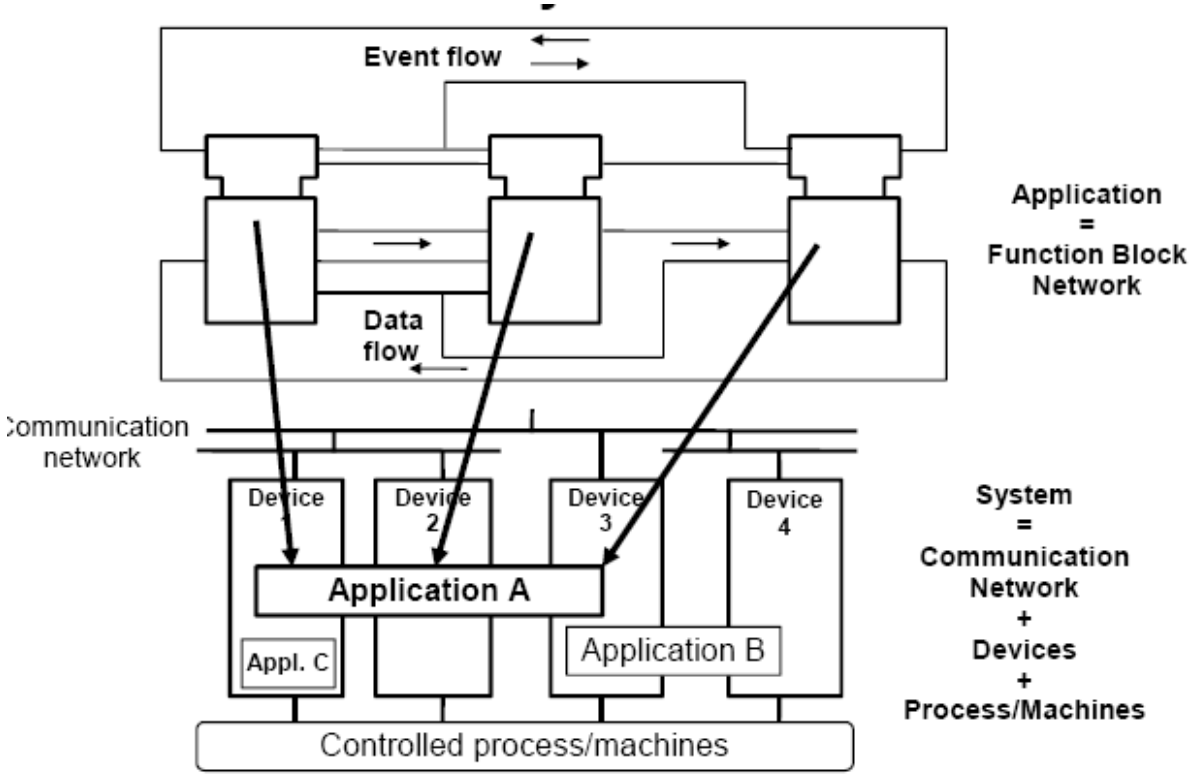
- Entrambi sono basati su Function Block
- Lo standard IEC 61131-3 non distingue tra eventi e dati.
  - L'arrivo di un nuovo dato (ad esempio un sensore) è esso stesso considerato un evento.
  - La scansione del PLC esegue continuamente l'algoritmo associato al function block.
- Lo standard IEC 61499 è event driven e non necessariamente esegue continuamente ogni function block, ma solo all'occorrenza di un evento.
- Lo standard IEC 61499 permette una più efficiente distribuzione di applicazioni su risorse distribuite (non per forza PLC).

# IEC 61499

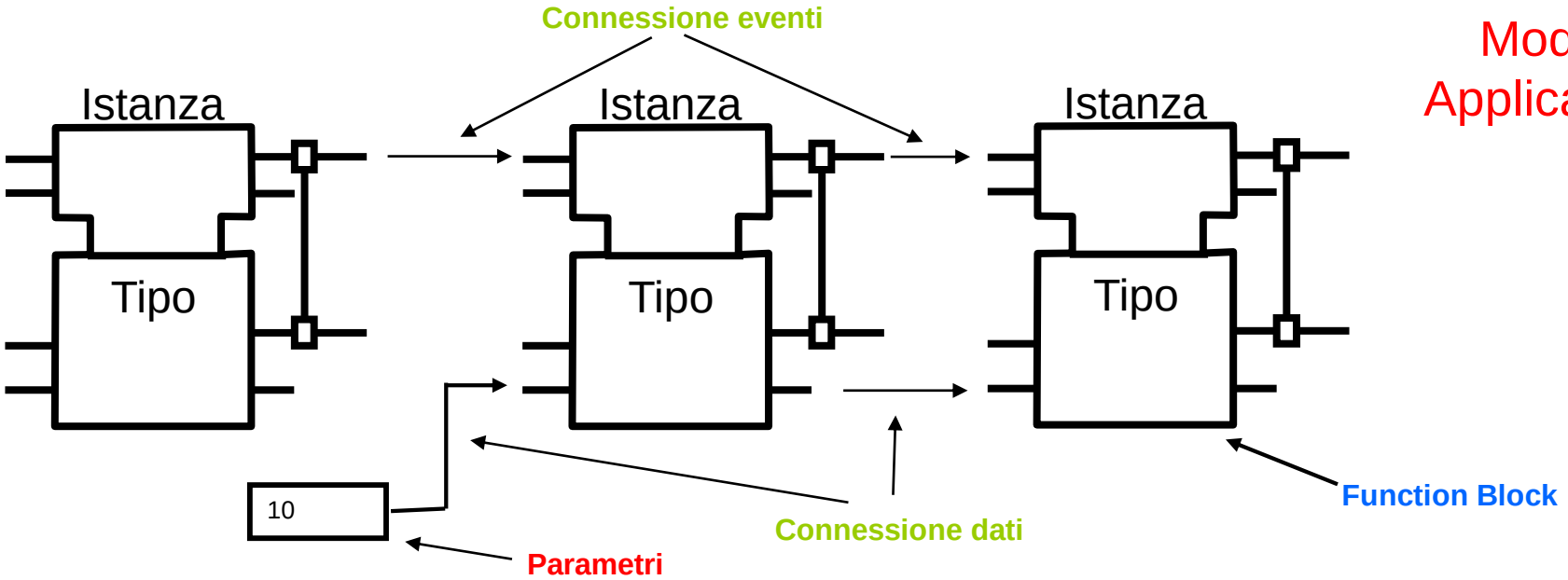
- **Part 1, Architecture**
  - IEC Standard, January 2005
- **Part 2, Software Tool Requirements**
  - IEC Standard, January 2005
- **Part 4, Rules for Compliance Profiles**
  - IEC Standard, May 2005
- **Part 3, Tutorial Information**
  - ritirato (obsolete), 2007

# IEC 61499: Concetti Generali

- La metodologia di progettazione di un sistema distribuito si basa sulla definizione di diversi modelli:
  - Modello dell'Applicazione
    - ✓ Modello del Blocco Funzionale
  - Modello del Sistema
    - ✓ Modello del Dispositivo
    - ✓ Modello della Risorsa



# IEC 61499: Modello di Applicazione

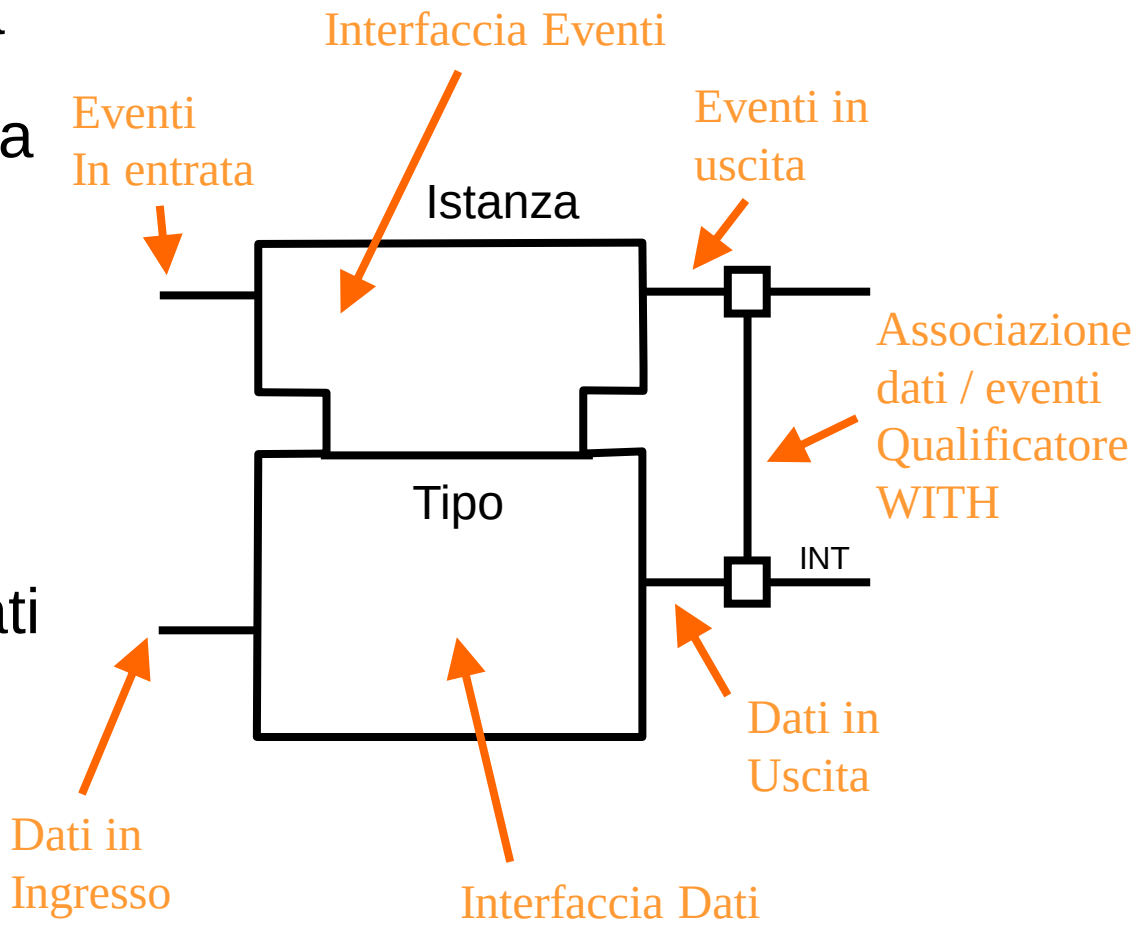



Modello di Applicazione

- Un'applicazione viene definita in termini di:
  - insieme di istanze di Blocchi Funzionali
  - insieme di Parametri. Un Parametro è un'entità dotata di tipo e specifica un valore costante
  - Insieme di connessioni: eventi e dati. Una connessione è caratterizzata da direzionalità
- L'applicazione viene descritta nel suo insieme; essa può essere distribuita su più risorse del sistema, anche su più dispositivi (vedi modello di sistema, dispositivo, risorsa)
- L'unità più piccola che può essere distribuita è il **Blocco Funzionale**

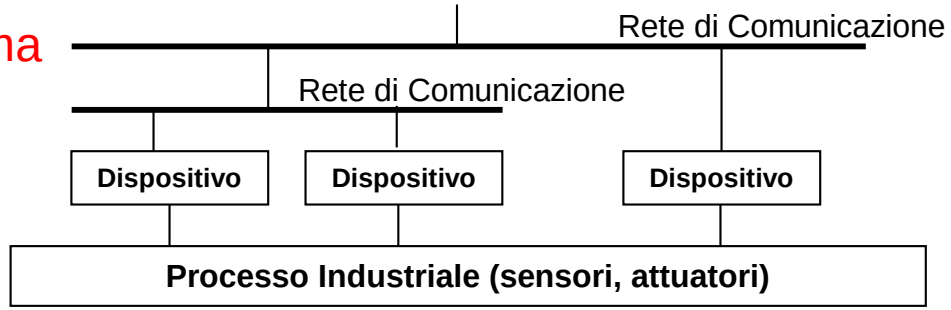
# IEC 61499: Modello del Blocco Funzionale

- E’ un’unità software che riceve dati ed eventi da altri blocchi, esegue algoritmi e genera a sua volta dati e eventi
- E’ un’estensione di un blocco funzionale classico, quale IEC 61131
- Basato su OO
- Interfaccia: Eventi e Dati



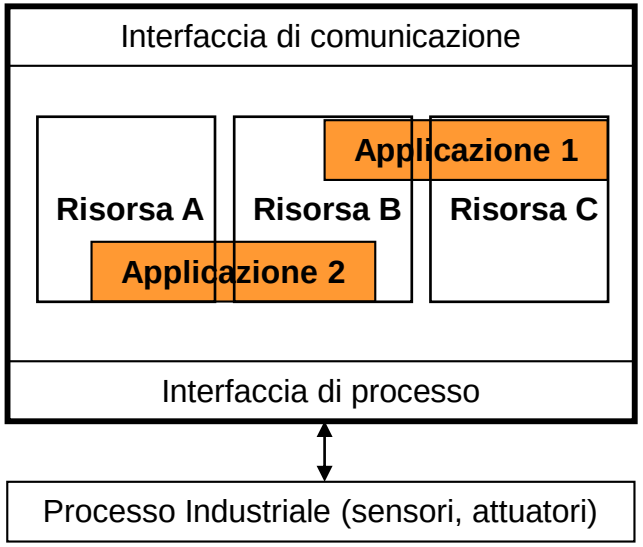
# IEC 61499: Modello di Sistema

Modello di Sistema




- Un sistema di controllo e misura per un processo industriale è un insieme di dispositivi interconnessi tramite una o più reti di comunicazione
- I **dispositivi** possono essere:
  - Dispositivi di controllo (es. PLC)
  - Dispositivi di supporto (es. Sistemi di supervisione)
- Le **reti** possono essere:
  - Reti di controllo: collegano i sistemi di controllo
  - Reti di Informazione: permettono lo scambio dati tra sistemi informativi e acquisizione dati

# IEC 61499: Modello di Dispositivo

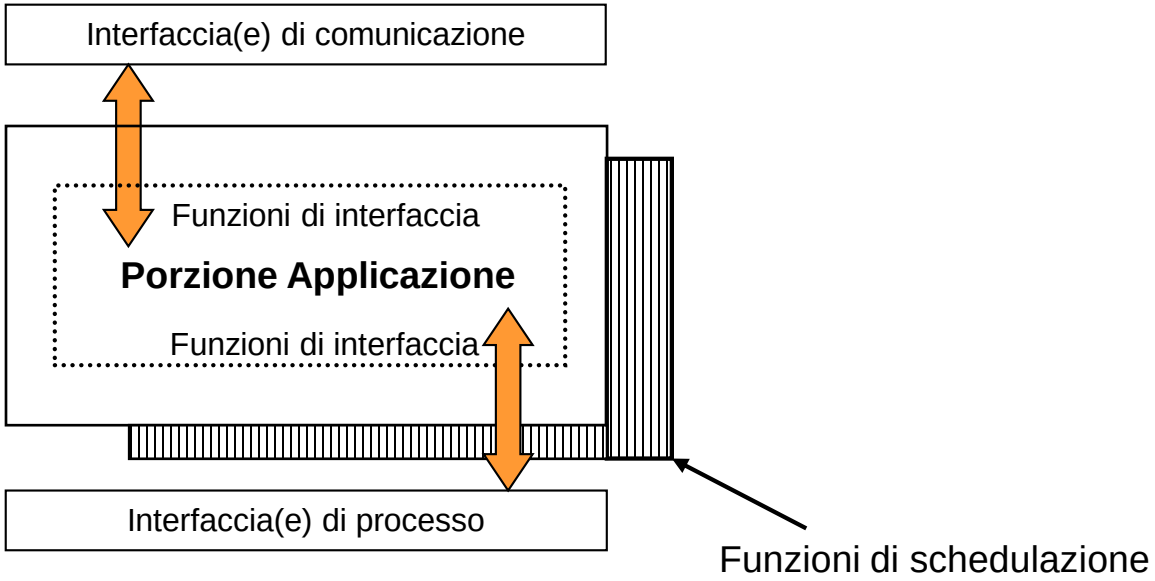



**Modello di Dispositivo**

- Un dispositivo è definito in termini di Risorse ed interfacce
- Le risorse forniscono dei servizi indispensabili per eseguire le applicazioni
- Le interfacce sono:
  - di processo (tra il processo industriale e la risorsa)
  - di comunicazione (tra la rete di comunicazione e la risorsa)

# IEC 61499: Modello di Risorsa

**Modello di Risorsa**




- Una risorsa acquisisce dati dalle interfacce, elabora i dati e genera dati verso le interfacce
- Una risorsa è caratterizzata:
  - Una o più porzioni di un'applicazione
  - Funzioni per l'interfacciamento con il processo fisico
  - Funzioni per l'interfacciamento con la rete di comunicazione
  - Funzioni di schedulazione, che permette l'esecuzione di un'applicazione o di una sua porzione

# Tipi di Blocchi Funzionali

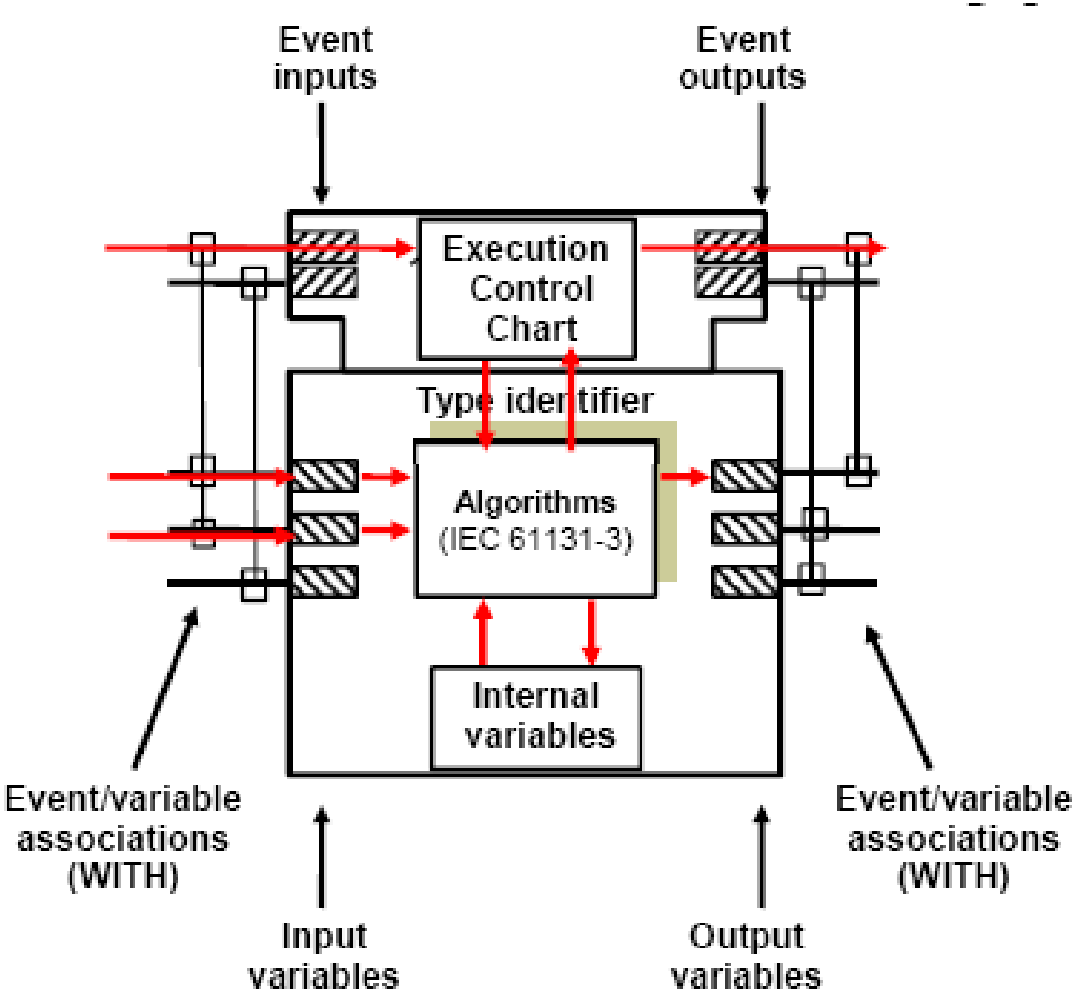
- Esistono 4 tipi di blocchi funzionali, graficamente simili:
  - **Blocchi di base:** comportamento definito da ECC e algoritmi.
  - **Blocchi composti**: unione di più blocchi.
  - **Blocco Sottoapplicazione:** di recente definizione
  - **Blocchi di interfaccia di servizio:** realizzano interfacciamento con hardware/software della risorsa e con dispositivi di comunicazione.

- E’ incoraggiata la composizione fra blocchi. Da pochi blocchi base è possibile avere comportamenti complessi (la composizione favorisce il riutilizzo e l’eleganza).

# Blocco Funzionale di Base

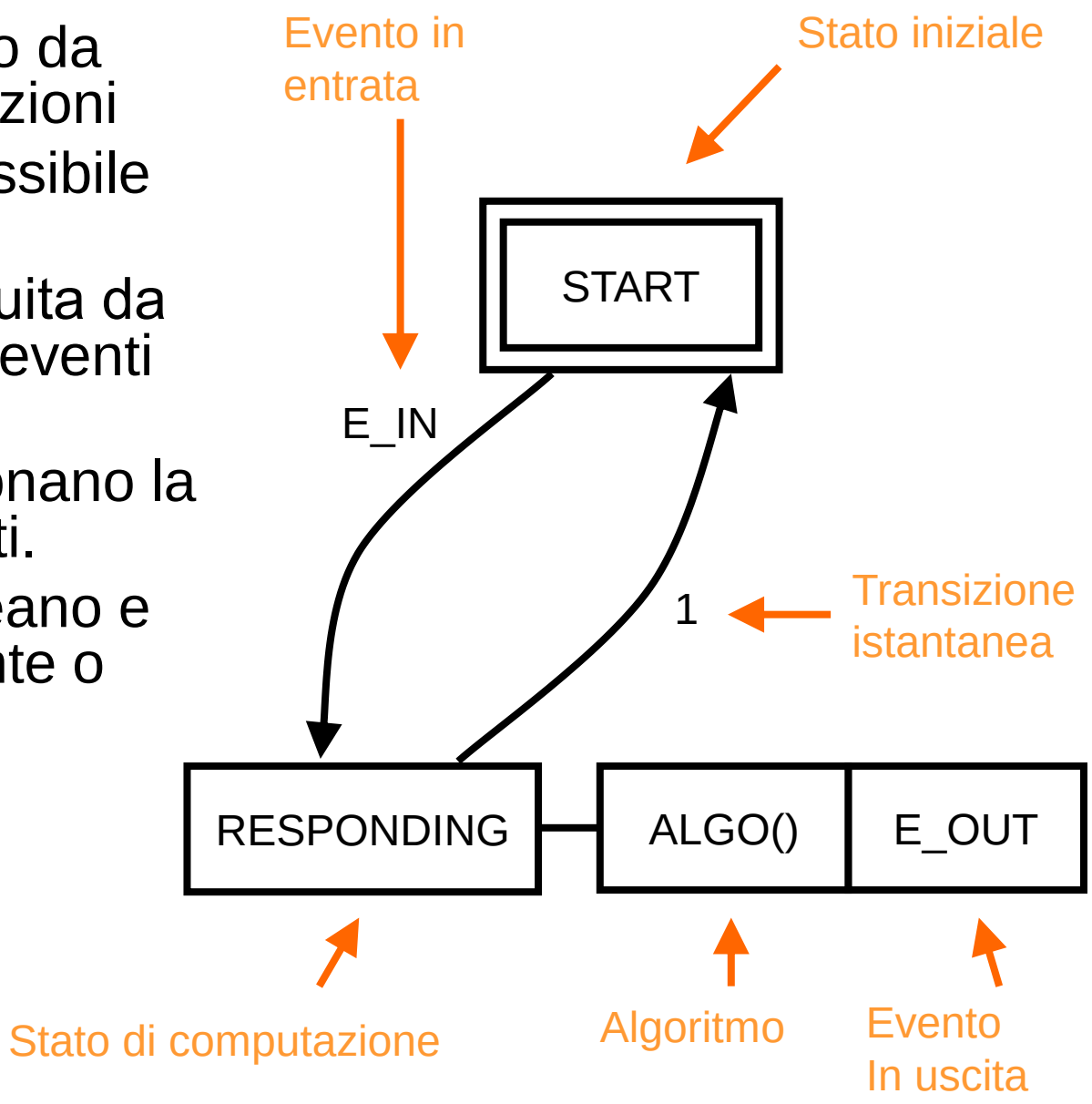
- Il blocco funzionale di base è caratterizzato da:
  - variabili interne, equiparate ad attributi dell'OO
  - algoritmi,
    - ✓ Possono essere formalizzati in un qualunque linguaggio anche se lo standard suggerisce IEC 61131-3
    - ✓ Elaborano i dati in ingresso e le variabili interne e producono nuovi valori per i dati in uscita
  - un modello che descrive l'esecuzione del blocco
    - ✓ Viene scritto nella forma di un Execution Control Chart (ECC)

# Blocco Funzionale di Base



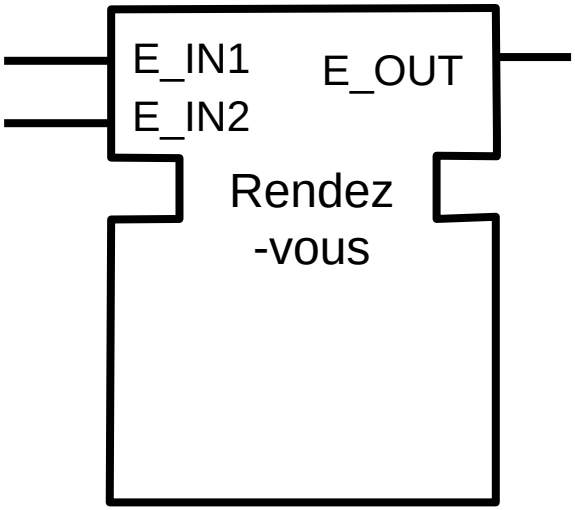
# Execution Control Chart

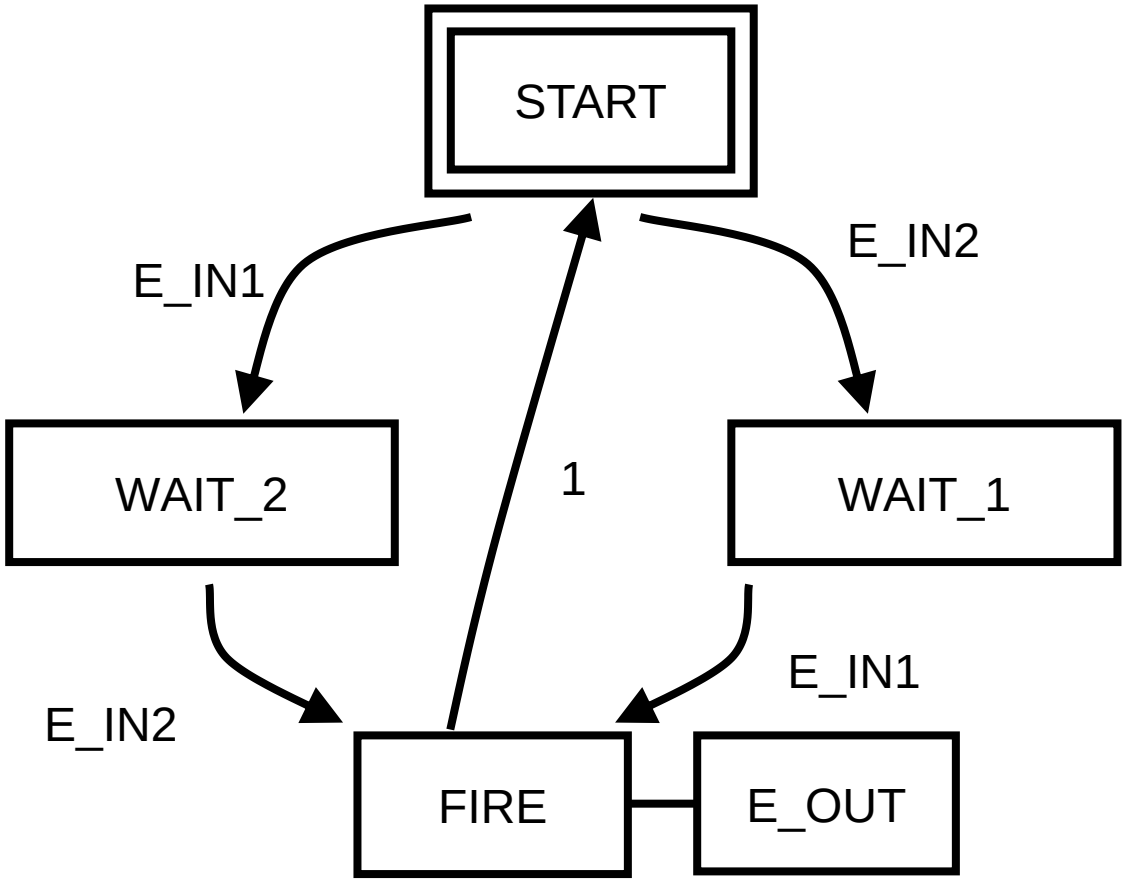
- ECC: modello fatto da stati, azioni, transizioni
- Ad uno stato è possibile associare azioni
- Un'azione è costituita da un algoritmo e da eventi in uscita
- Gli eventi condizionano la transizione tra stati.
- Un evento è booleano e può essere costante o un'espressione



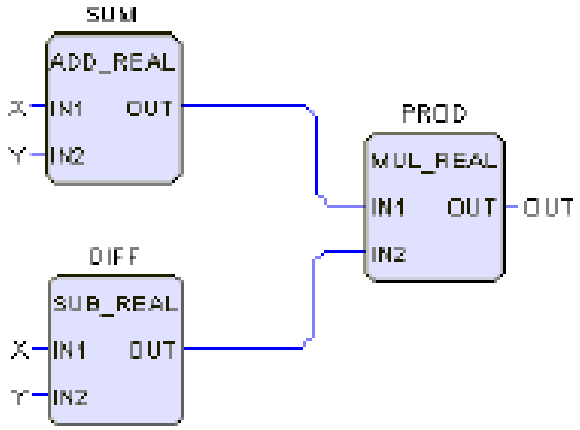
# Esempio di ECC: elemento *rendez-vous*

- Il componente aspetta E_IN1 e E_IN2 prima di far partire E_OUT.

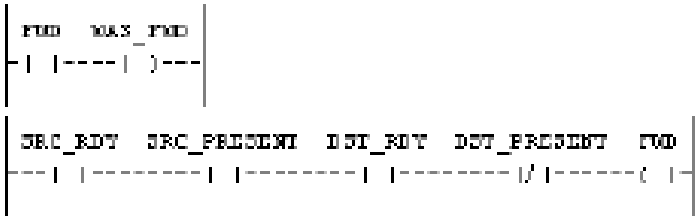


# Algoritmi: Linguaggi di Programmazione




**Function Block Diagram (FBD)**



**Ladder Diagram (LD)**

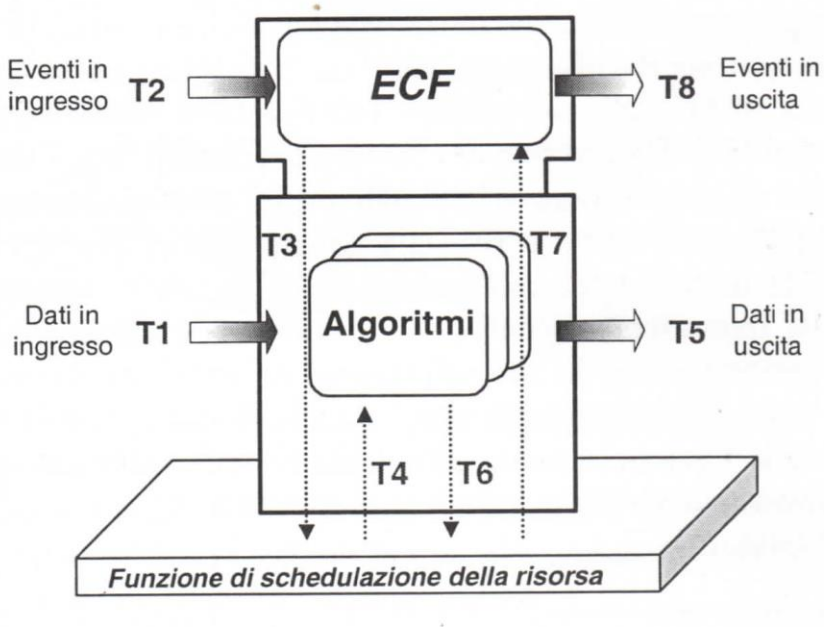
```
OUT := (X-Y)*(X+Y);
```

**Structured Text (ST)**

- Also Java, C++, etc, depending on software tool support

# Interazione tra Blocco Funzionale e Risorsa

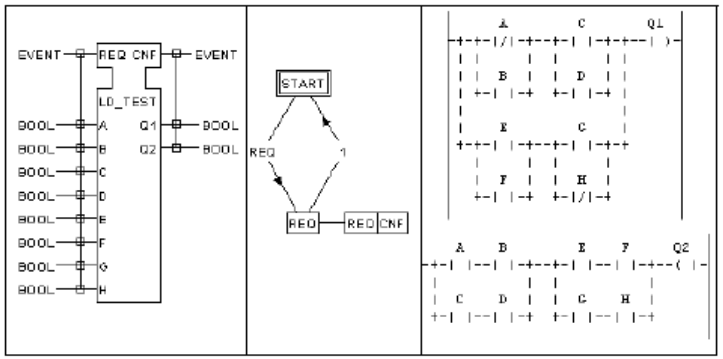
- Ogni istanza di FB possiede una porzione dedicata al controllo dell'esecuzione dell'ECC: La Funzione di Controllo di Esecuzione (ECF) utilizza l'ECC e controlla l'esecuzione degli algoritmi




- T1: i dati sono disponibili e stabili
- T2: occorrenza evento legato ai dati (WITH)
- T3: ECF richiede alla funzione di schedulazione della risorsa di eseguire l'algoritmo associato allo stato attuale dell'ECC
- T4: la funzione di schedulazione rende disponibile la risorsa e l'algoritmo viene eseguito
- T5: algoritmo calcola i dati in uscita
- T6: il blocco funzionale comunica alla funzione di schedulazione che ha terminato l'algoritmo e che rilascia la risorsa
- T7: la funzione di schedulazione comunica all'ECF che l'esecuzione è finita
- T8: ECF genera gli eventi in uscita

- Alcuni Parametri Prestazionali:
  - Tsetup: T2-T1, Tstart: T4-T2
  - Talg: T6-T4, Tfinish:T8-T6

# Mapping con XML




A corresponding XML document would be:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE FBType SYSTEM "../LibraryElement.dtd" >
<FBType Name="LD_TEST" Comment="LD Algorithm Example" >
  <Identification Standard="61499-2-C.1" Description="LD Algorithm Example" />
  <VersionInfo Organization="IEC TC65/WG6" Version="0.2" Author="JHC" Date="2000-11-
16" Remarks="Corrected Identification" />
  <VersionInfo Organization="IEC TC65/WG6" Version="0.1" Author="JHC" Date="2000-06-
20" Remarks="Tested Sun compiler" />
  <VersionInfo Organization="IEC TC65/WG6" Version="0.0" Author="JHC" Date="2000-02-
01" />
  <CompilerInfo header="package fb.rt.part2;" >
    <Compiler Language="Java" Vendor="IBM" Product="VisualAge" Version="3.0" />
    <Compiler Language="Java" Vendor="Sun" Product="JDK" Version="1.1.8" />
  </CompilerInfo>
  <InterfaceList>
    <EventInputs>
      <Event Name="REQ" >
        <With Var="A" />
        <With Var="B" />
        <With Var="C" />
        <With Var="D" />
        <With Var="E" />
        <With Var="F" />
        <With Var="G" />
        <With Var="H" />
      </Event>
    </EventInputs>
    <EventOutputs>
      <Event Name="CNF" Comment="Execution Confirmation" >
        <With Var="Q1" />
        <With Var="Q2" />
      </Event>
    </EventOutputs>
    <InputVars>
      <VarDeclaration Name="A" Type="BOOL" />
      <VarDeclaration Name="B" Type="BOOL" />
      <VarDeclaration Name="C" Type="BOOL" />
      <VarDeclaration Name="D" Type="BOOL" />
      <VarDeclaration Name="E" Type="BOOL" />
      <VarDeclaration Name="F" Type="BOOL" />
      <VarDeclaration Name="G" Type="BOOL" />
      <VarDeclaration Name="H" Type="BOOL" />
    </InputVars>
    <OutputVars>
      <VarDeclaration Name="Q1" Type="BOOL" />
      <VarDeclaration Name="Q2" Type="BOOL" />
    </OutputVars>
  </InterfaceList>
  <BasicFB>
    <ECC >
      <ECState Name="START" Comment="Initial State" x="341.1765" y="105.8824" >
      </ECState>
```

# Blocco Funzionale di Base

- Esistono numerosi FB di base di libreria
- La maggior parte presentano degli eventi comuni:
  - REQ (EI). Causa attivazione algoritmo
  - INIT (EI). Serve per inizializzare il FB
  - CNF (EO). Si associa alla conclusione dell'algoritmo a seguito di una REQ.
  - INITO (EO). Server per far propagare il segnale di inizializzazione
  - IND (EO). Evento in uscita prodotto dal FB, legato in genere ad un cambiamento occorso internamente nel FB
  - Se sono presenti dei particolari dati booleani in ingresso QI e in uscita QO, essi influenzano il significato degli eventi in ingresso/uscita
  - Ad esempio si avrà INIT+ e INIT- a seconda se QI=1 o QI=0, e i due eventi avranno diversi effetti
  - Ad esempio si avrà CNF+ e CNF- a seconda se QO=1 o QO=0, e i due eventi avranno diversi significati

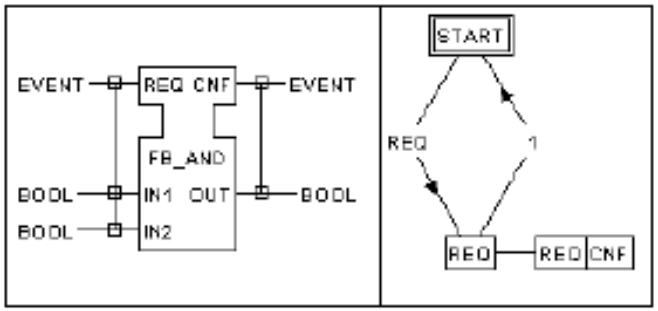
# Blocco Funzionale di Base derivato da IEC 61131

- Funzioni e Function Blocks IEC 61131-3 possono essere convertite in Function Blocks IEC 61499, secondo le seguenti regole:
  - Il nome del FB IEC 61499 è composto da FB_Nome, dove Nome è il nome della Funzione o Function Block IEC61131 (ad esempio FB_ADD_INT).
  - Le Variabili di Ingresso/Uscita e i corrispondenti Data Type sono gli stessi
  - Gli eventi INIT, INITO vengono usati solo per i FB IEC 61499 derivati da FB IEC 61131, ma non esistono nei FB IEC 61499 derivati da semplici funzioni IEC 61131.
  - Si usano gli eventi **REQ e CNF** (**quasi sempre**)
  - Esempi: FB_AND, FB_OR, FB_NOT

# Blocco Funzionale di Base: Esempio

- FB_AND

The interface and ECC would appear graphically as follows.

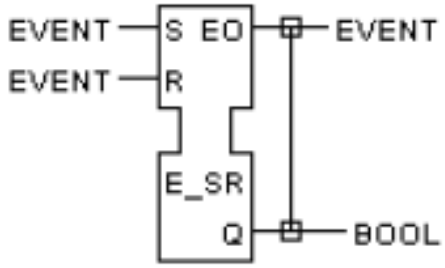


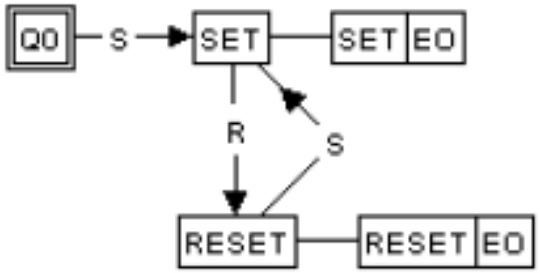

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE FBType SYSTEM "../LibraryElement.dtd" >
<FBType Name="FB_AND" Comment="Boolean AND" >
  <Identification Standard="61499-1-D.1" Classification="Math"
ApplicationDomain="Any" Function="AND" Type="Boolean" />
  <VersionInfo Organization="IEC TC65/WG6" Version="0.1" Author="JHC"
Date="2000-06-10" Remarks="Tested Sun compiler." />
  <VersionInfo Organization="IEC TC65/WG6" Version="0.0" Author="JHC"
Date="2000-01-29" Remarks="Simple Boolean AND" />
  <CompilerInfo header="package fb.rt.part2;" >
    <Compiler Language="Java" Vendor="Sun" Product="JDK" Version="1.1.8"
/>
    <Compiler Language="Java" Vendor="IBM" Product="VisualAge"
Version="3.0" />
  </CompilerInfo>
  <InterfaceList>
    <EventInputs>
      <Event Name="REQ" >
        <With Var="IN1" />
        <With Var="IN2" />
      </Event>
    </EventInputs>
    <EventOutputs>
      <Event Name="CNF" >
        <With Var="OUT" />
      </Event>
    </EventOutputs>
    <InputVars>
      <VarDeclaration Name="IN1" Type="BOOL" />
      <VarDeclaration Name="IN2" Type="BOOL" />
    </InputVars>
    <OutputVars>
      <VarDeclaration Name="OUT" Type="BOOL" Comment="IN1&amp;#38;IN2" />
    </OutputVars>
  </InterfaceList>
  <BasicFB>
    <ECC >
      <ECState Name="START" Comment="Initial State" x="200" y="105.8824" >
      </ECState>
      <ECState Name="REQ" Comment="Normal execution" x="205.8824"
y="676.4706" >
        <ECAction Algorithm="REQ" Output="CNF" />
      </ECState>
      <ECTransition Source="START" Destination="REQ" Condition="REQ"
x="370.5882" y="405.8824" />
      <ECTransition Source="REQ" Destination="START" Condition="1"
x="52.9412" y="429.4117" />
    </ECC>
  <Algorithm Name="REQ" >
    <ST Text="  OUT := (IN1 &amp;#38; IN2);&amp;#10;" />
  </Algorithm>
  </BasicFB>
</FBType>
```

Algoritmo AND

# Blocco Funzionale di Base: Esempio

- E_SR , E_RS (Event)

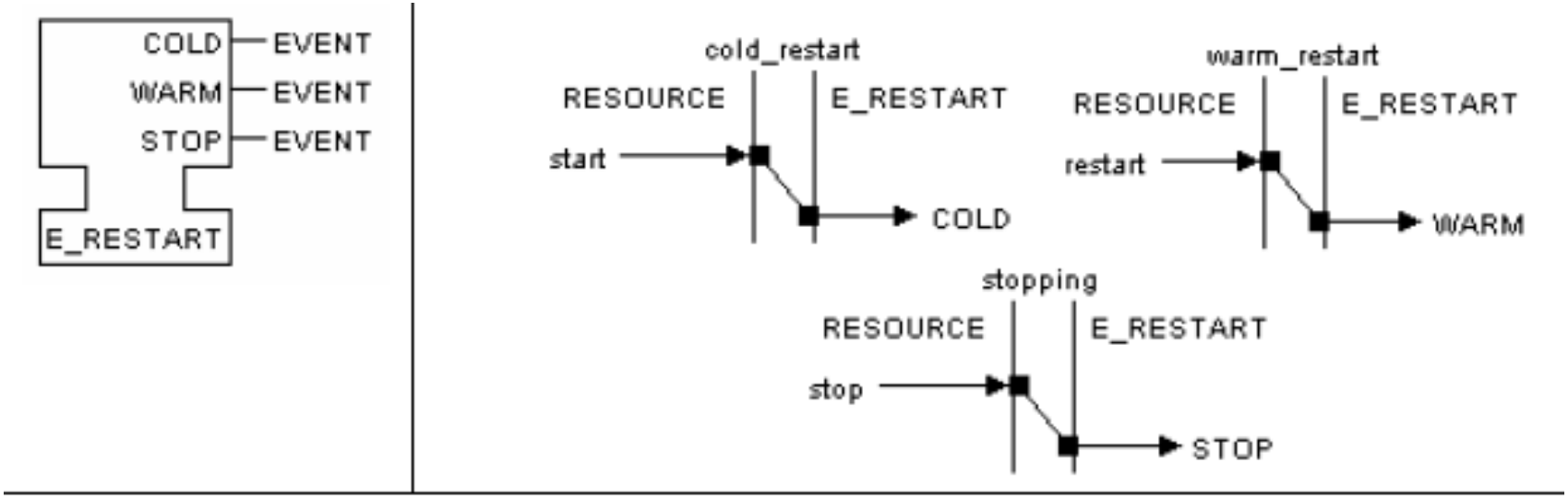



The output Q is set to 1 (TRUE) upon the occurrence of an event at the *S* input, and is reset to 0 (FALSE) upon the occurrence of an event at the R input. An event is issued at the EO output when the value of Q changes.

# Blocco Funzionale di Base: Esempio

- E_Restart (Event)




1) An event is issued at the COLD output upon "cold restart" of the associated resource.

2) An event is issued at the WARM output upon "warm restart" of the associated resource.

3) An event is issued at the STOP output (if possible) prior to "stopping" of the associated resource.

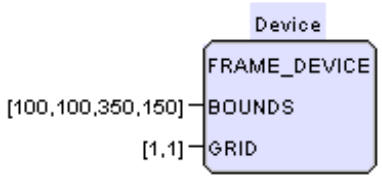
NOTE 1 See IEC 61131-3 for a discussion of "cold restart" and "warm restart".

# IEC 61499: Ambiente Holobloc

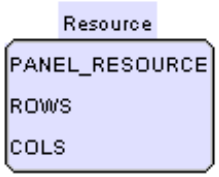
- Esistono molti tools di modellazione e programmazione in ambiente IEC 61499.
- Molti offrono anche il run-time (su PC e/o PLC)
- Alcuni sono gratuiti e sviluppati in ambito accademico, altri commerciali:
  - Function Block Development Kit (FBDK), Rockwell Automation / Holobloc Inc. (www.holobloc.com)
  - FBench - Open Source FB Workbench, The University of Auckland (NZ)/ o3neida
  - CORFU / Archimedes Engineering Support System, University of Patras (Greece)
  - ODCE - Open Distributed Control Environment, Technical University of Vienna (Austria)
  - ISaGRAF, ICS Triplex (Canada) (www.icstriplex.com), software commerciale
  - NxtOne, (www.nxtcontrol.com), software commerciale

# Device e Risorse in Ambiente Holobloc

- Le applicazioni possono essere mappate su:
  - Device type FRAME_DEVICE. Crea una finestra di una certa ampiezza sullo schermo.




  - Resource PANEL_RESOURCE. Crea un pannello rettangolare entro la finestra; entro questo pannello vengono:
    - ✓ visualizzati tutti gli output su schermo prodotti da eventuali function blocks HMI (ad esempio OUT_BOOL).
    - ✓ immessi gli ingressi da FB HMI (ad esempio IN_BOOL)



# Linguaggi IEC 61131-3 in Ambiente Holobloc

- Uso di Ladder e ST per la definizione di algoritmi

- Nel caso di Ladder, i rung servono solo per tradurre:

  Uscita := espressione

  L'espressione viene specificata tramite la Notazione Polacca Inversa. Esempi booleani:

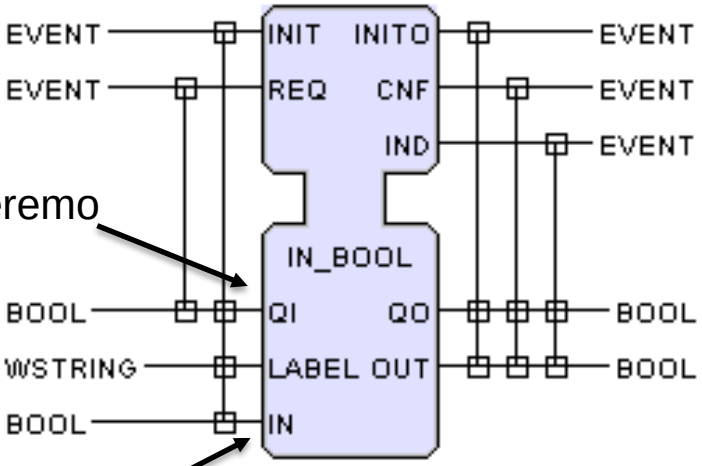
  | | |
  |---|---|
  | IN1 IN2 & | (IN1 AND IN2) |
  | IN1 ! IN2 \| | (Not IN1 OR IN2) |

- Nel caso di ST, useremo solo espressioni del tipo:

  Variabile := espressione

# Blocchi Funzionali di I/O in Holobloc

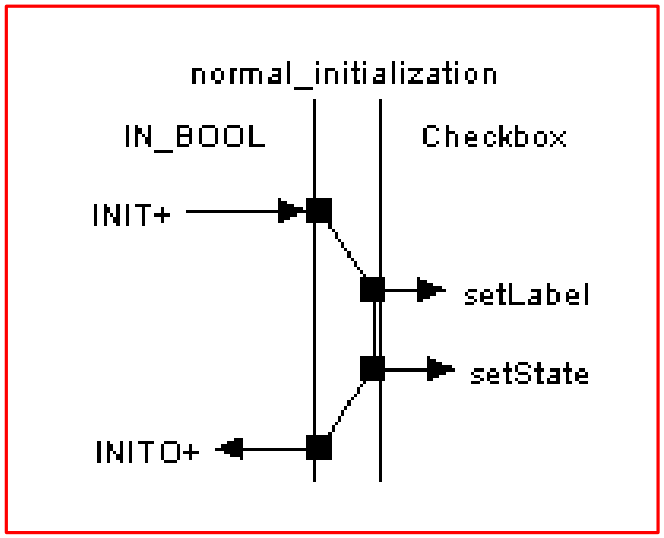
## IN_BOOL (HMI)



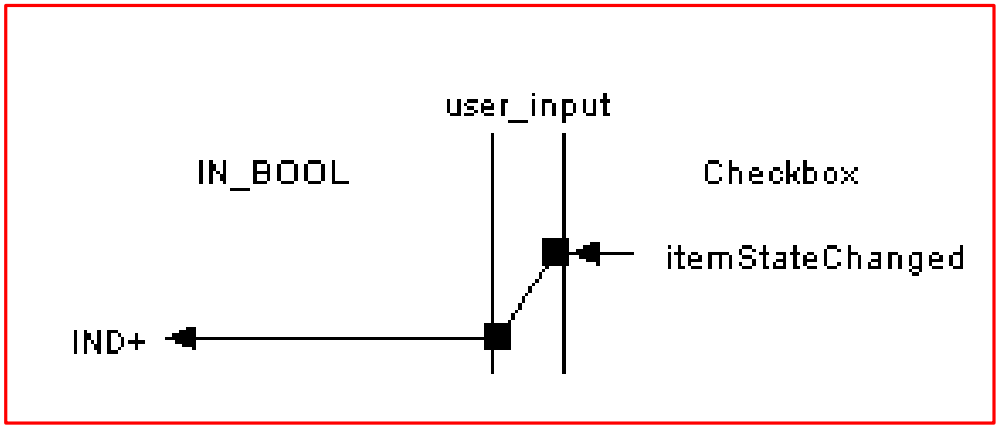

Lo considereremo sempre 1

Selezione box.

Lo considereremo sempre 0 (non selezionato)

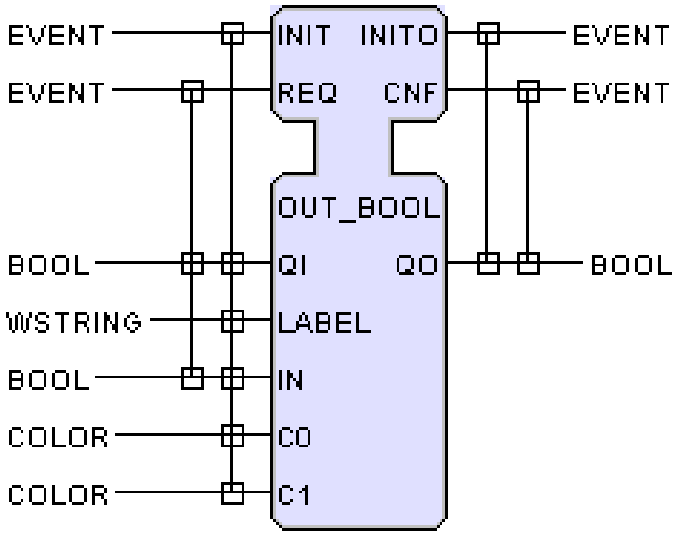



In OUT viene prodotto il dato inserito dall'utente



# Blocchi Funzionali di I/O in Holobloc

## OUT_BOOL (HMI)



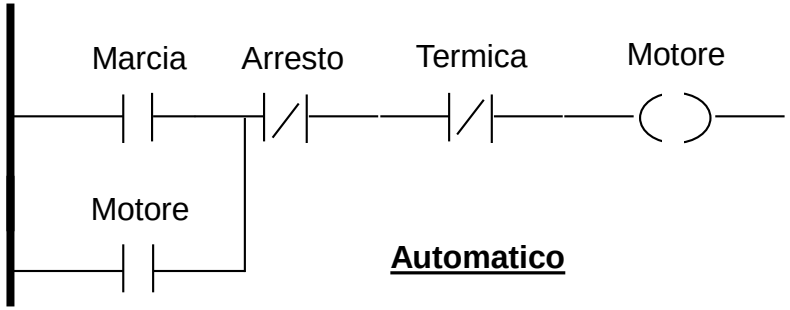

- INIT+ (QI=1), setta la Label e lo Stato (i colori ON/OFF)
- IN viene letto per ogni REQ+ (Q=1)

# Esercizi in Holobloc su Blocco Funzionale di Base

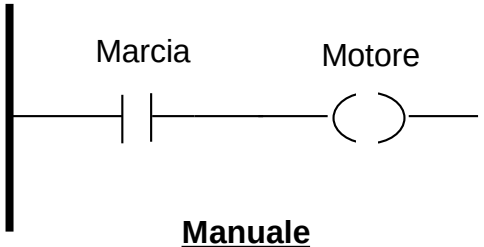
- Esercizio 0: Realizzazione di una Applicazione usando FB di Base standard: AND
- Esercizio 1: Realizzazione di una Applicazione usando FB di Base standard: Xor (!A && B || A && !B)
  - A! B & A B! & |
- Esercizio 2: Realizzazione di un FB di Base ex-novo: Xor + Applicazione che lo utilizza
- Attenzione: quando si creano FB di Base e Composti e si vogliono testare (tasto play), bisogna chiudere e riaprire l'ambiente Holobloc!

# Esercizi in Holobloc su Blocco Funzionale di Base

- Esercizio 3: Realizzazione di un FB MarciaArrestoMotore + Applicazione

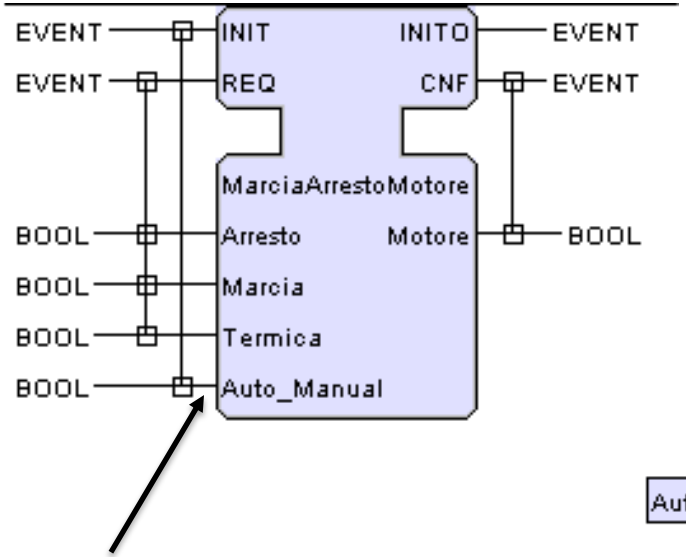



**Automatico**

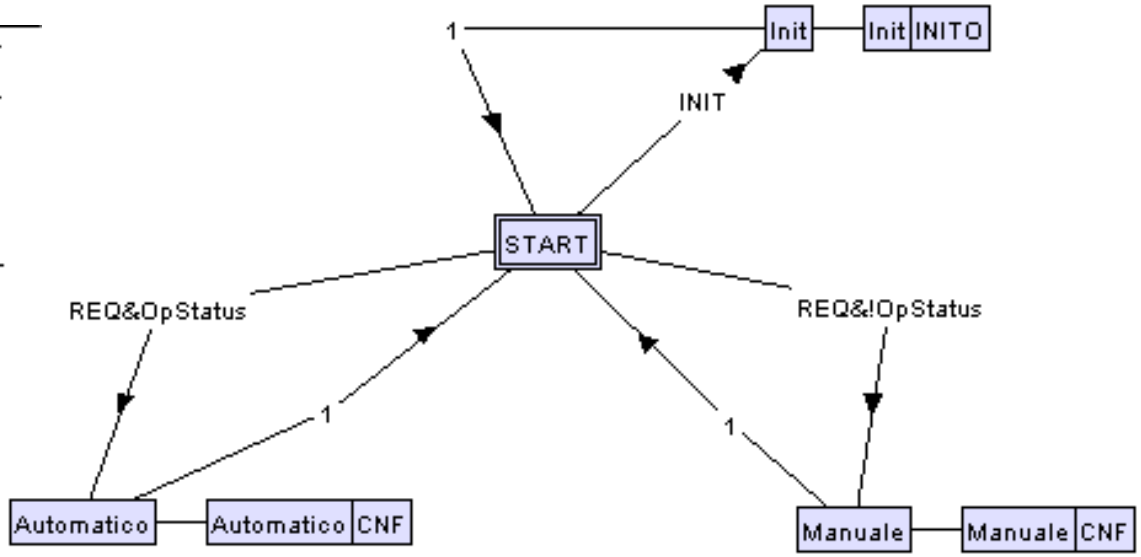



**Manuale**

Notazione Polacca Inversa=Marcia Motore | Arresto ! & Termica ! &

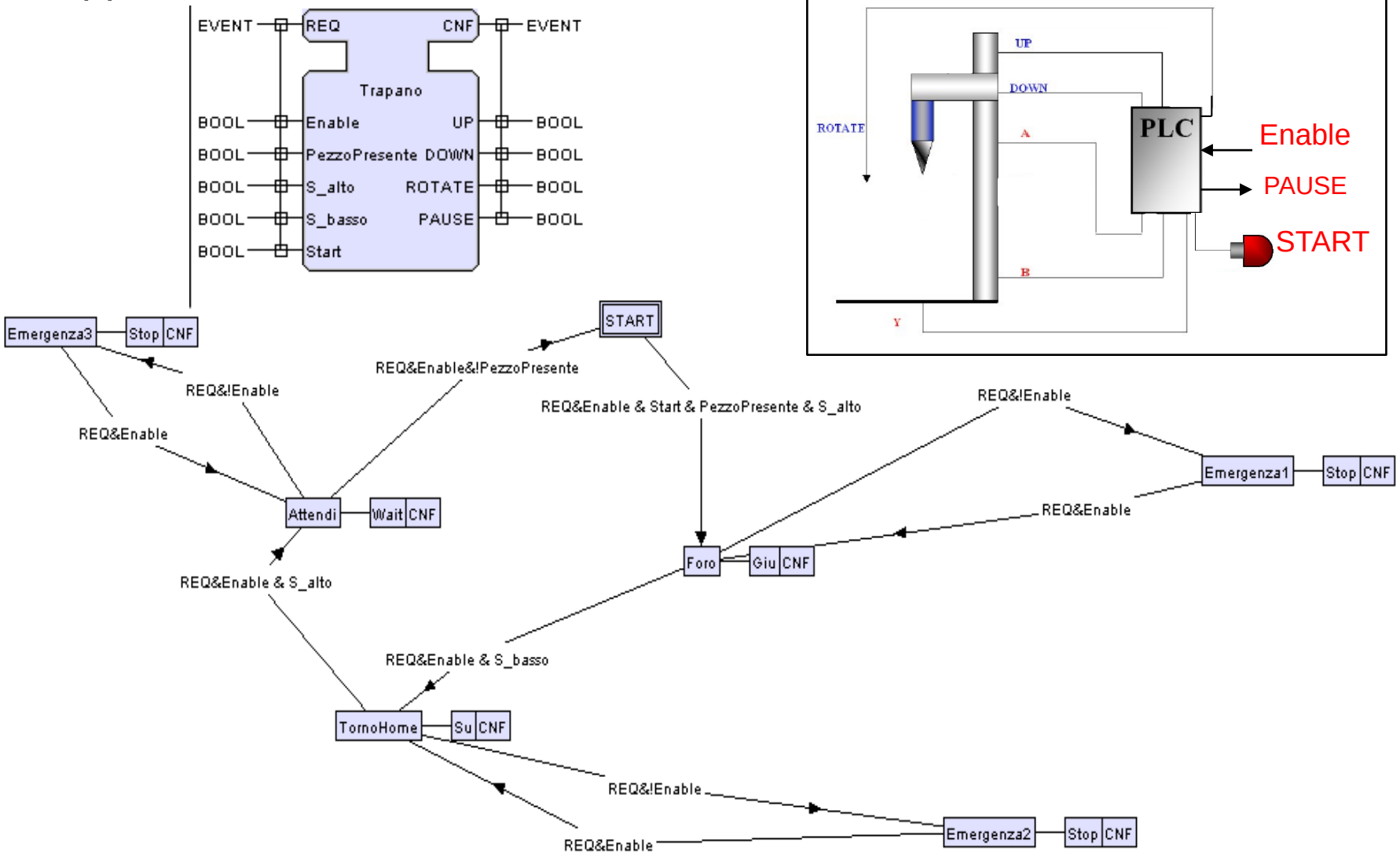



Si suppone Parametro Statico



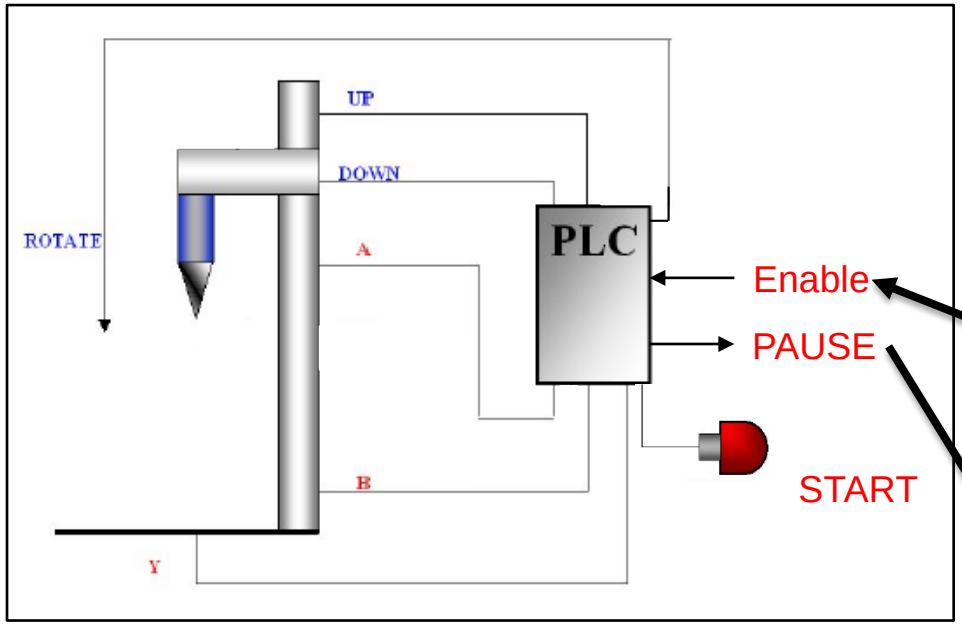
# Esercizi in Holobloc su Blocco Funzionale di Base

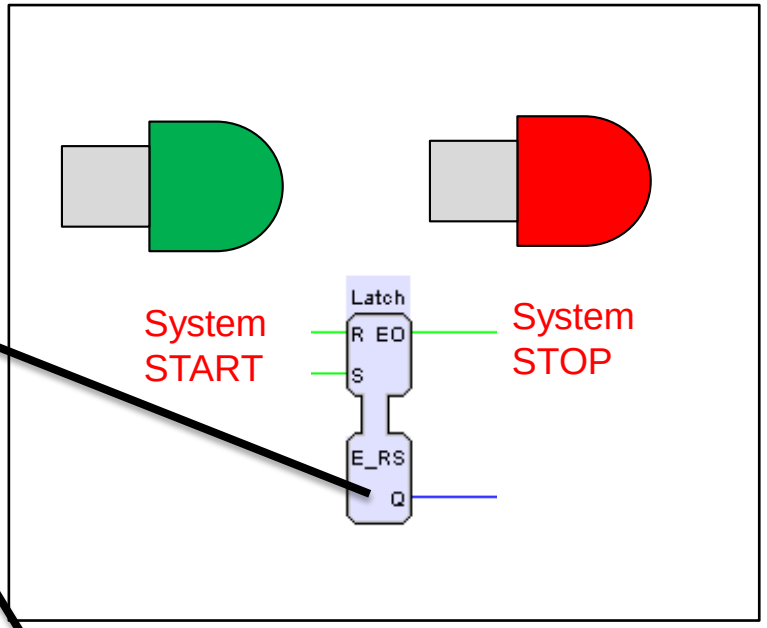
- Esercizio 4: Realizzazione di un FB TrapanoAutomatico + Applicazione

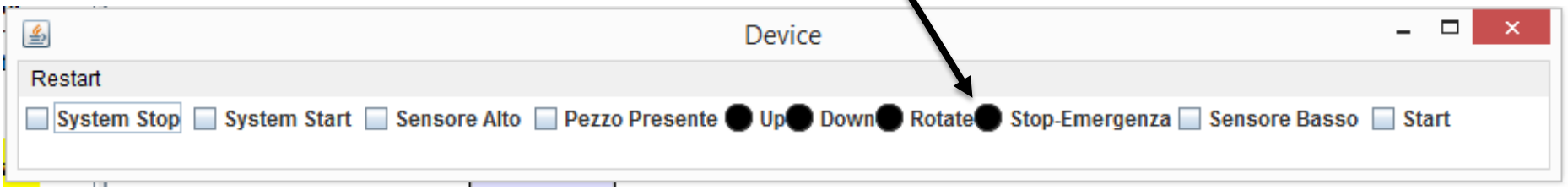


# Esercizi in Holobloc su Blocco Funzionale di Base

- Esercizio 4: Realizzazione di un FB TrapanoAutomatico + Applicazione

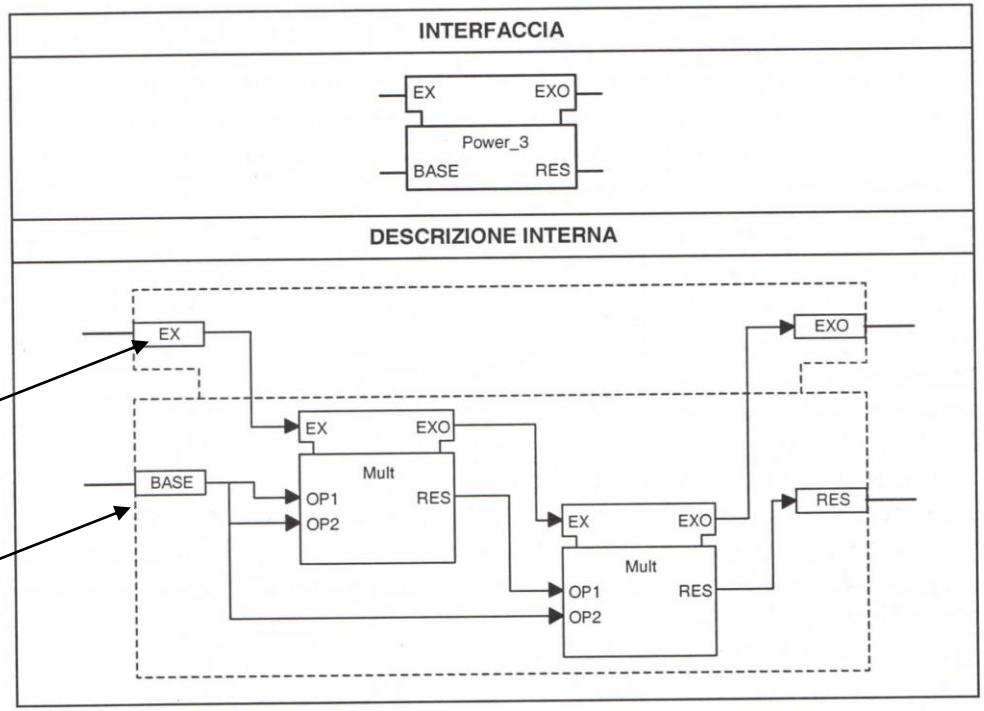


# Il Blocco Funzionale Composto

- Sono blocchi in cui il funzionamento è espresso in termini di una sotto-rete di blocchi funzionali
- Sono caratterizzati da:
  - Interfaccia esterna con **Variabili di Eventi e Dati** (EI, DI, EO, DO)
  - Funzionamento interno non dipende da un ECC, ma dall'interazione dei blocchi che lo compongono
  - Non è possibile definire variabili interne, ma tutto il funzionamento deve basarsi unicamente sul comportamento di ciascun blocco.

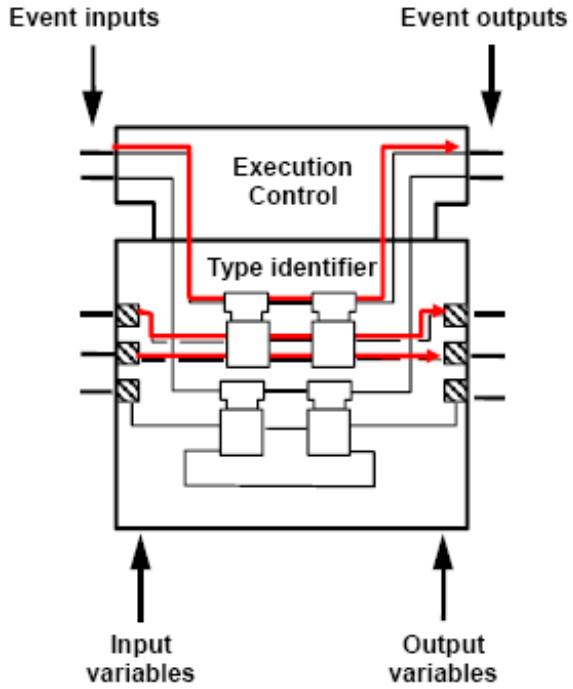



Variabile di evento

Variabile di dato

# Il Blocco Funzionale Composto

- **Esecuzione di un blocco composto:**
  - Se un evento in ingresso è connesso direttamente ad un evento in uscita, l'occorrenza evento determina l'evento in uscita
  - Se un evento in ingresso è connesso ad un blocco interno, si attiva tale evento
  - Se l'evento di uscita di un blocco interno è connesso con un evento di ingresso di un altro blocco interno, allora tale evento viene considerato nel funzionamento di quest'ultimo blocco interno
  - Se l'evento di uscita di un blocco interno è connesso con un evento di uscita del blocco composto, allora l'evento di uscita viene attivato



# Esercizi su Blocco Funzionale Composto

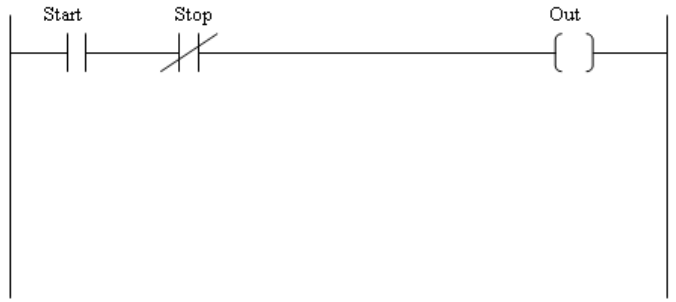
- Esercizio 5: Realizzazione di un Blocco Funzionale Composto + Applicazione che lo utilizza

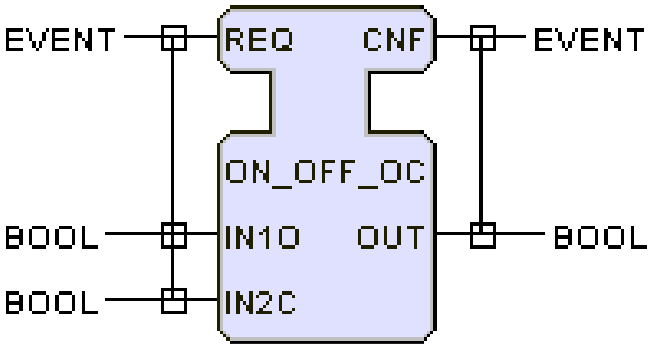
| Language | Vendor | Product | Version |
|---|---|---|---|
| Java | Sun | JDK | 1.6.0 |

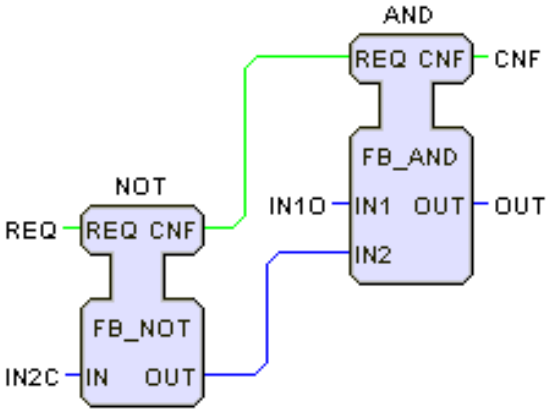
OK Cancel

| Attribute | Value |
|---|---|
| header | package fb.rt.esercizi; import fb.rt.math.*; |
| classdef | |

OK Cancel



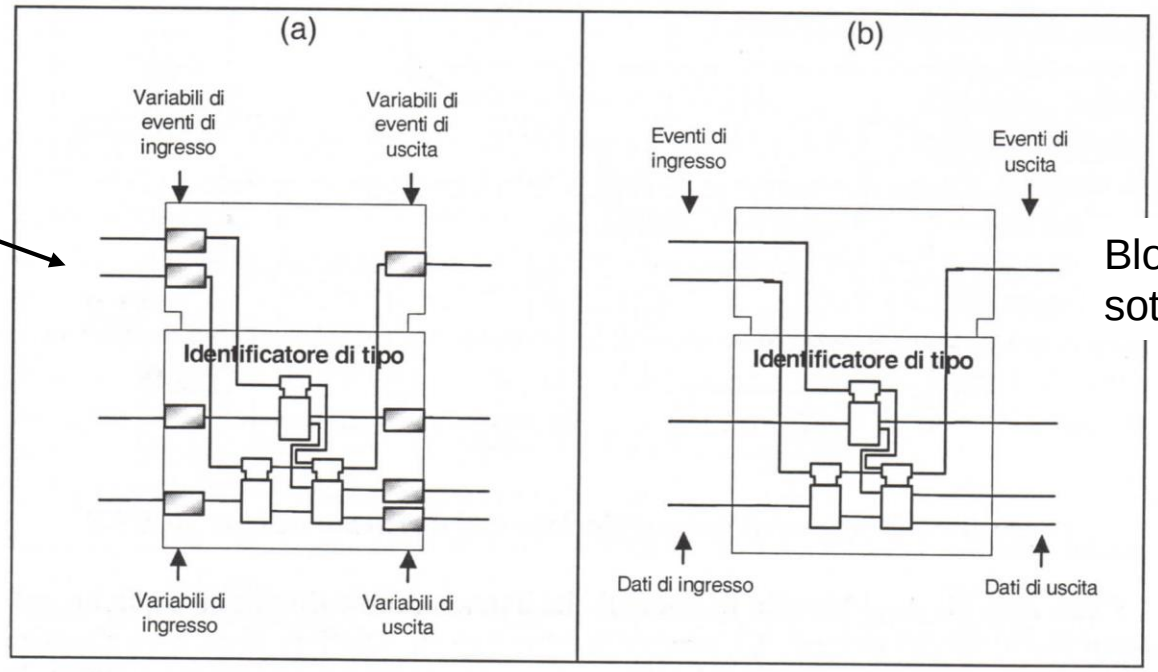



# Il Blocco Funzionale Sottoapplicazione

- E’ stato introdotto recentemente
- Differenze con blocco composto:
  - Non esistono Variabili di Eventi e Dati in I/O
  - **Una sottoapplicazione può essere distribuita su più risorse**
  - Nell’interfaccia non compare la clausola WITH
  - All’interno è necessario utilizzare particolari blocchi (SPLIT e MERGE) per realizzare diramazioni e congiungimenti di eventi

Blocco composto



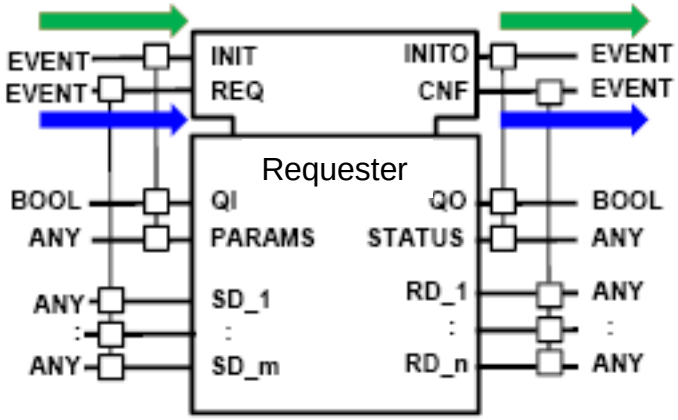

Blocco sottoapplicazione

# Il Blocco di Interfaccia di Servizio

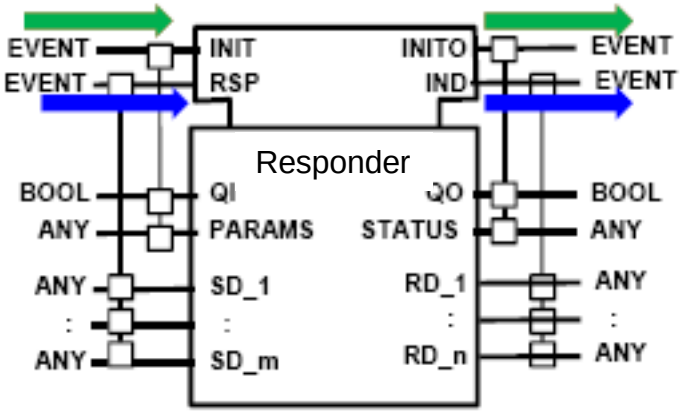
- I blocchi di interfaccia di servizio (SIFB) servono per utilizzare servizi offerti da una risorsa o da un dispositivo.

- Vantaggi:
  - Fornire un'interfaccia standard per l'utilizzo di funzioni private del sistema (calcolo o comunicazione)
  - Nascondere all'utente particolari di secondario interesse

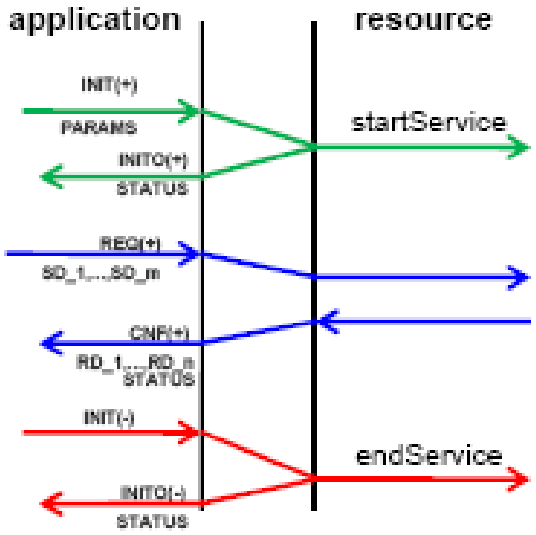
# Il Blocco di Interfaccia di Servizio

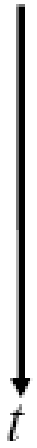


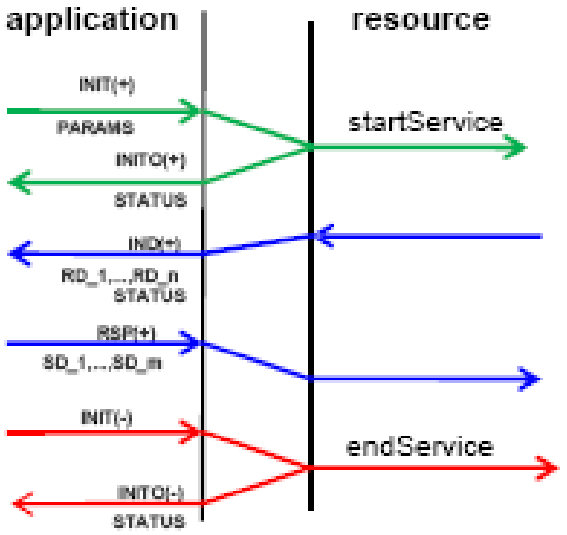

(application-initiated transactions)




(resource-initiated transactions)



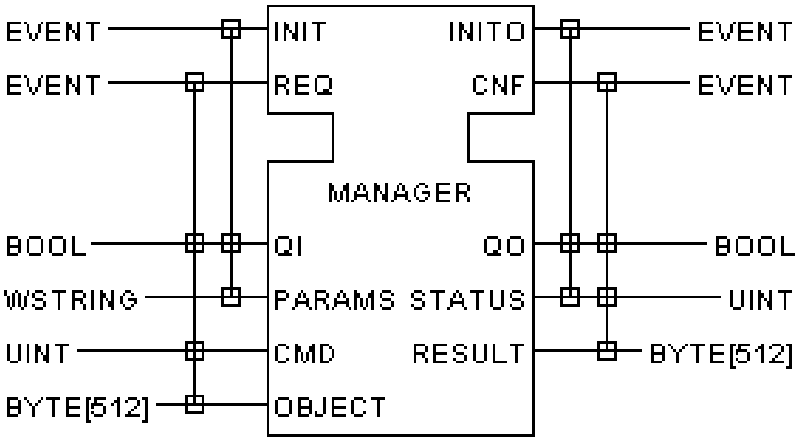



# Il Blocco di Interfaccia di Servizio

- Due tipi di blocchi di interfaccia di servizio:
  - Blocco di Gestione
  - Blocco di Comunicazione

# Il Blocco di Gestione

- Blocco di Gestione
  - Permettono di gestire ad alto livello le risorse e dispositivi
  - Risorsa:
    - » Agiscono su: tipi di dati, tipi di blocchi funzionali, istanze di blocchi funzionali, connessioni tra istanze di blocchi funzionali
    - » Permettono di: creare, inizializzare, iniziare, fermare, distruggere, chiedere l'esistenza e gli attributi, notificare il cambio di disponibilità e di stato
  - Dispositivo:
    - » Agiscono su: **risorse**
    - » Permettono di: creare, inizializzare, iniziare, fermare, distruggere, chiedere l'esistenza e gli attributi, notificare il cambio di disponibilità e di stato




| Value | Command | Semantics |
|---|---|---|
| 0 | CREATE | Create specified object |
| 1 | DELETE | Delete specified object |
| 2 | START | Start specified object |
| 3 | STOP | Stop specified object |
| 4 | READ | Read data from access path |
| 5 | WRITE | Write data to access path |
| 6 | KILL | Make specified object unrunnable |
| 7 | QUERY | Request information on specified object |

# Il Blocco di Comunicazione

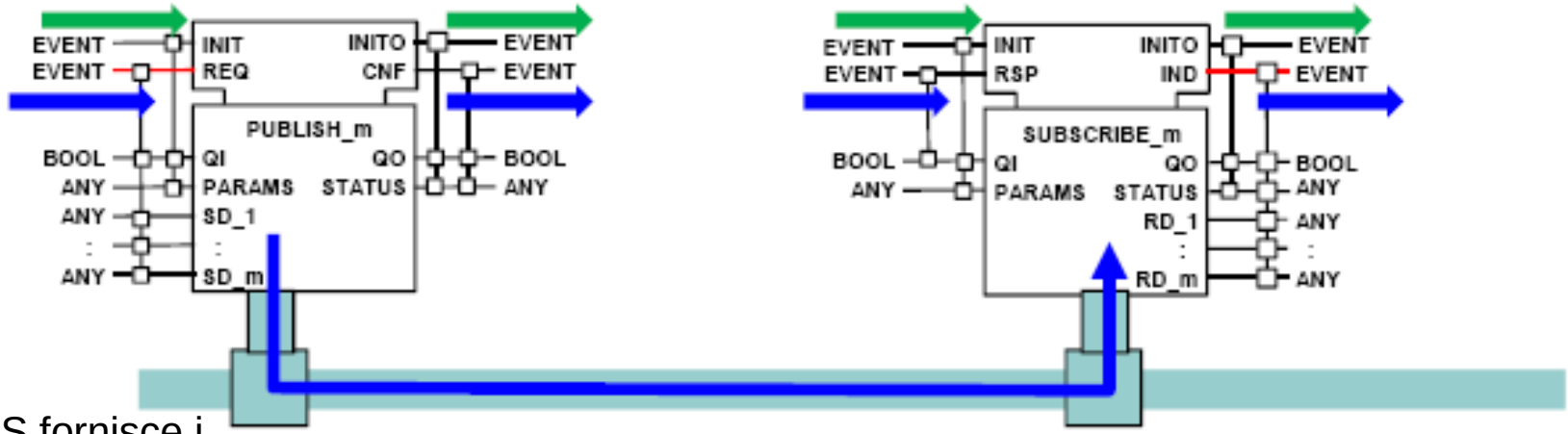
- Blocco di Comunicazione, Communication Function Block (COMFB)

- Offrono un'interfaccia STANDARD alle applicazioni per l'accesso alle risorse rappresentate da reti di comunicazione che connettono risorse e dispositivi del sistema di controllo

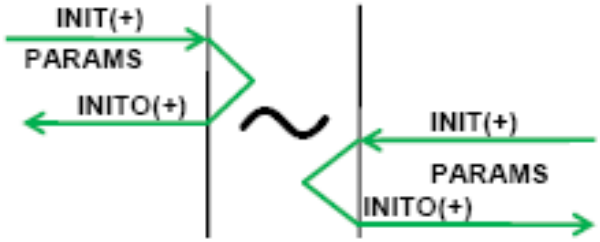
- La normativa sfrutta due modalità di comunicazione:
  - broadcast (unidirezionale) e
  - client-server (bidirezionale)

# Il Blocco di Comunicazione

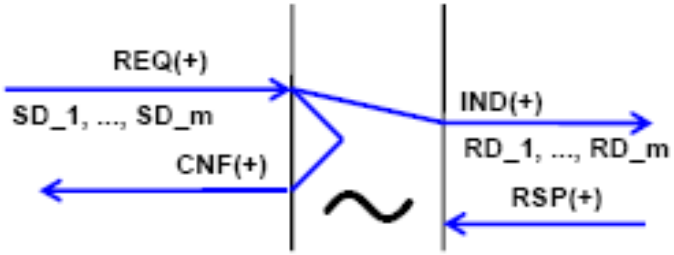
- Blocco di Comunicazione Unidirezionale Publish/Subscribe (non confermato)




- PARAMS fornisce i parametri necessari ad identificare il tipo di protocollo di comunicazione e i parametri necessari alla creazione della connessione




Publish

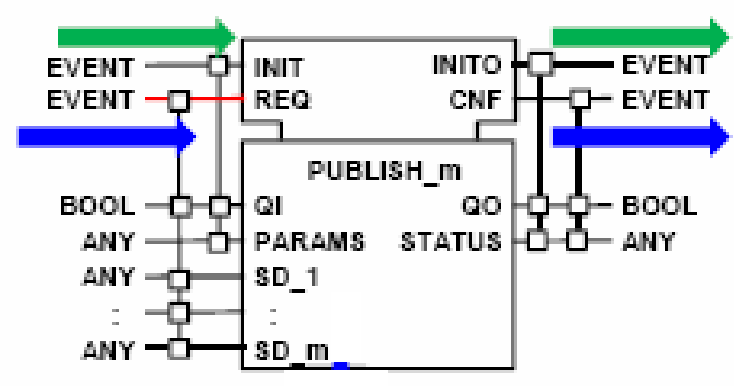


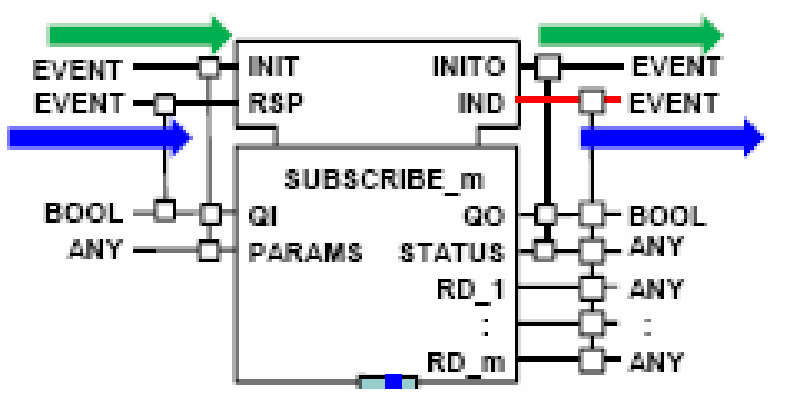

Subscribe

# Il Blocco di Comunicazione

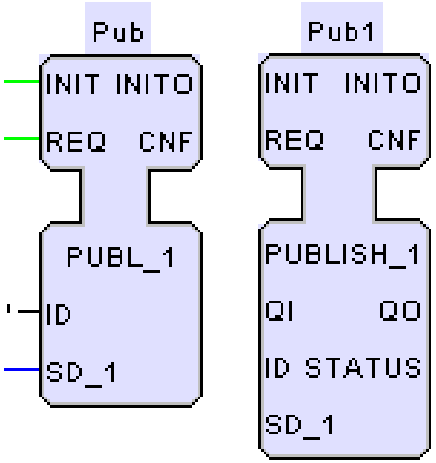
- Blocco di Comunicazione Unidirezionale Publish/Subscribe

**IEC 61499**



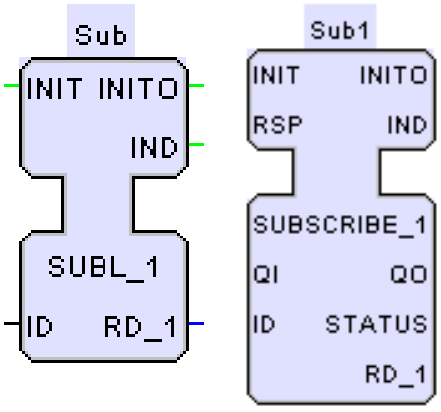




**Holobloc**




Usata solo per comunicazioni locali (stesso device)

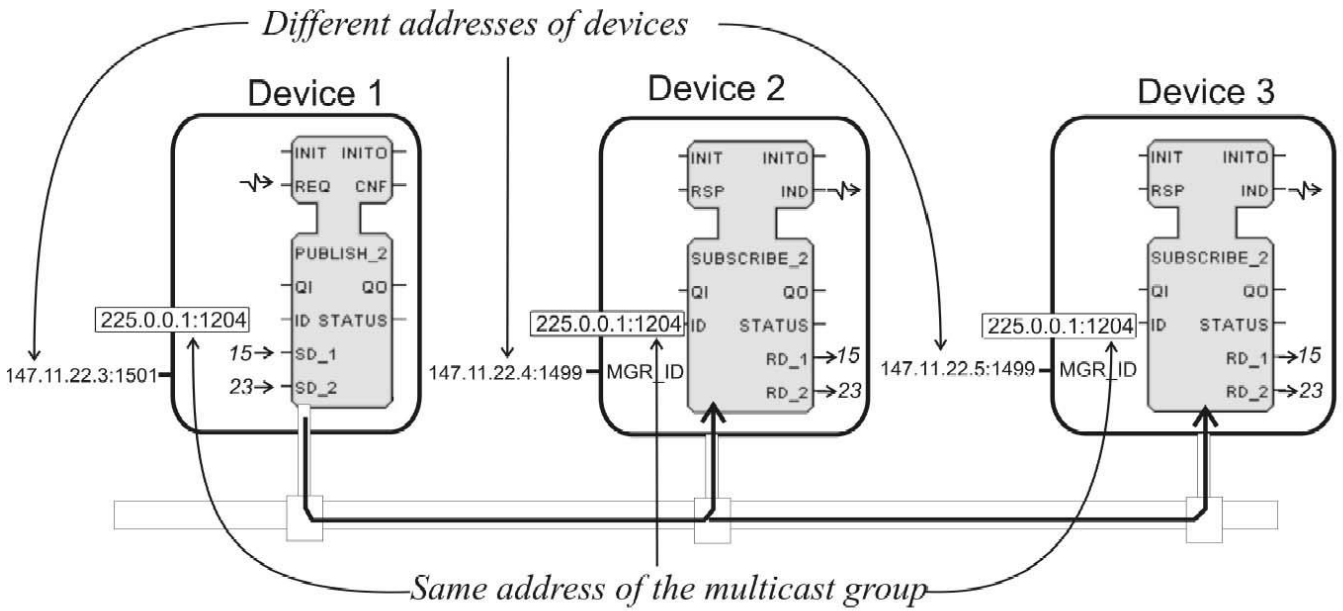
- L'ingresso PARAMS può essere anche indicato con ID




Usata solo per comunicazioni locali (stesso device)

# Il Blocco di Comunicazione in Holobloc

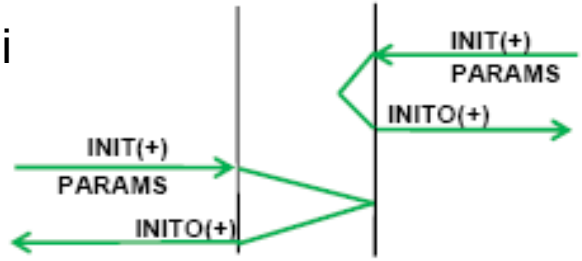
- I Function Block PUBLISH_n/SUBSCRIBE_n incapsulano UDP multicast services
  - ID (PARAMS) coincide con **IP address: port number**
  - Il range degli indirizzi IP ammessi è limitato a quello degli indirizzi IP multicast, da 224.0.0.1 a 239.255.255.255, mentre il numero di porta è limitato al range {1..65535}.
  - *Esempio*
    - ✓ PUBLISH che desidera pubblicare dati su UDP con ID 225.0.0.1:0001
    - ✓ ID del Publish verrà settato a "225.0.0.1:0001".
    - ✓ ID del Subscribe deve essere identico.



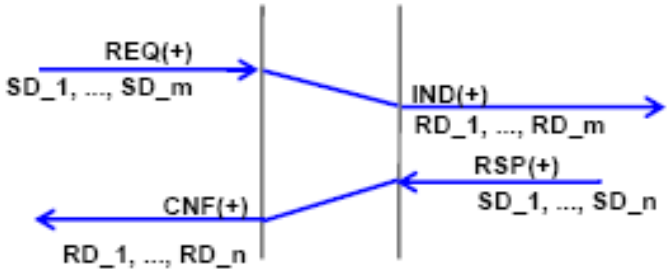
# Il Blocco di Comunicazione

- Blocco di Comunicazione Bidirezionale Client/Server


- PARAMS fornisce i parametri necessari ad identificare il tipo di protocollo di comunicazione e i parametri necessari alla creazione della connessione




Client

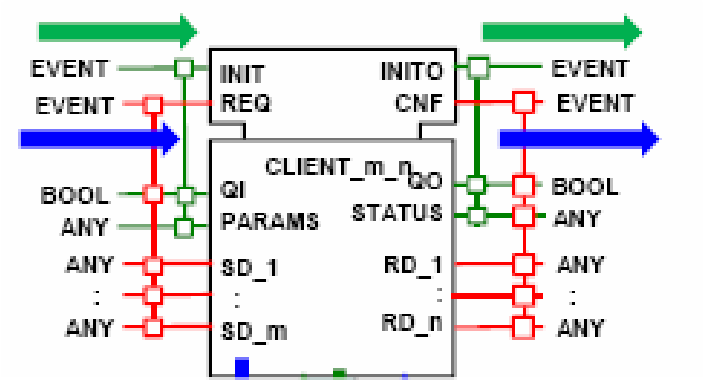


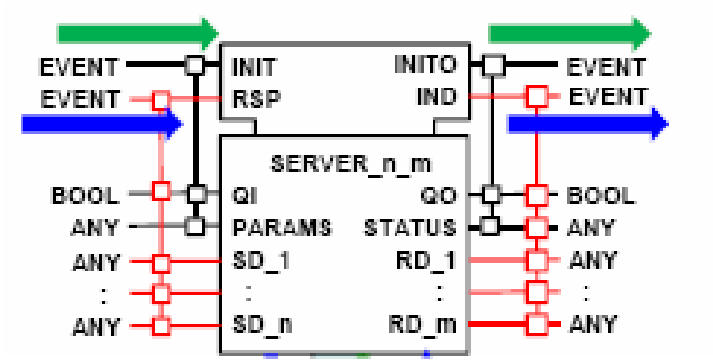

Server

# Il Blocco di Comunicazione

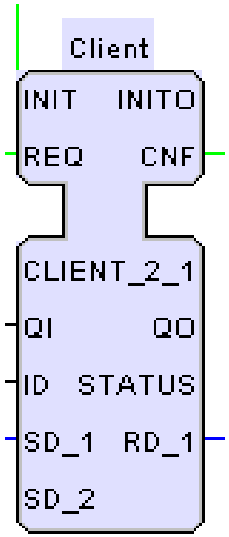
- Blocco di Comunicazione Bidirezionale Client/Server

**IEC 61499**

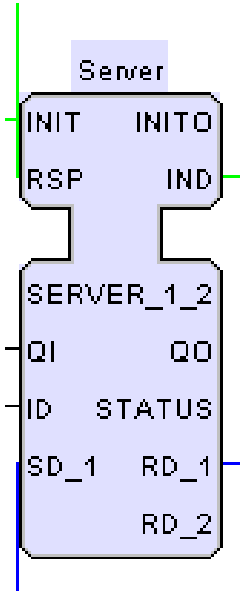



**Holobloc**

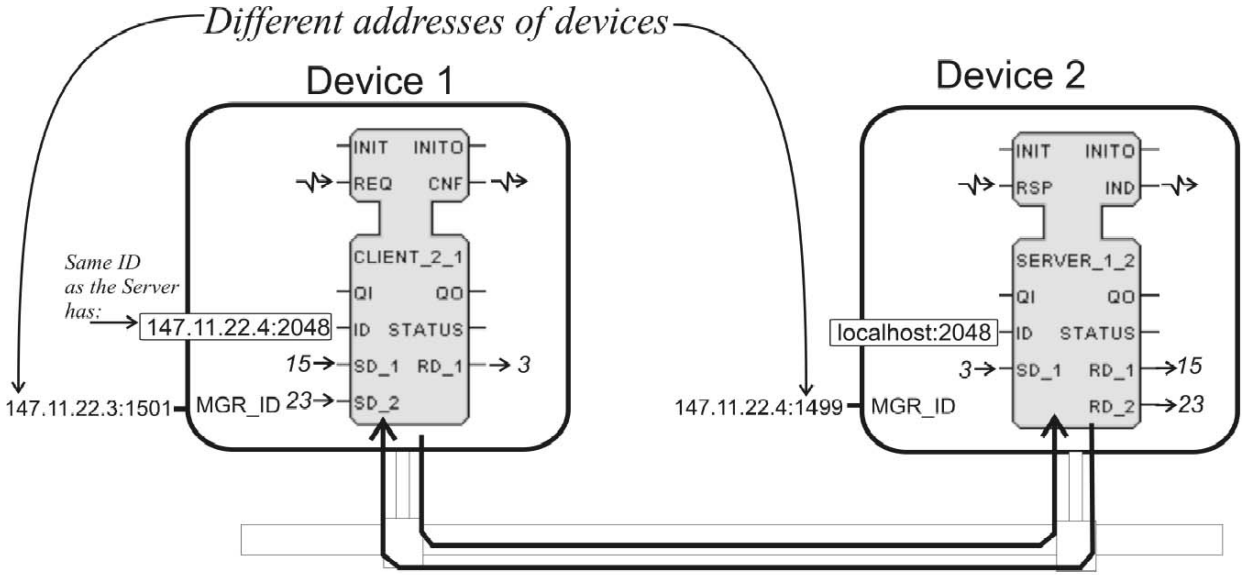



- L'ingresso PARAMS può essere anche indicato con ID



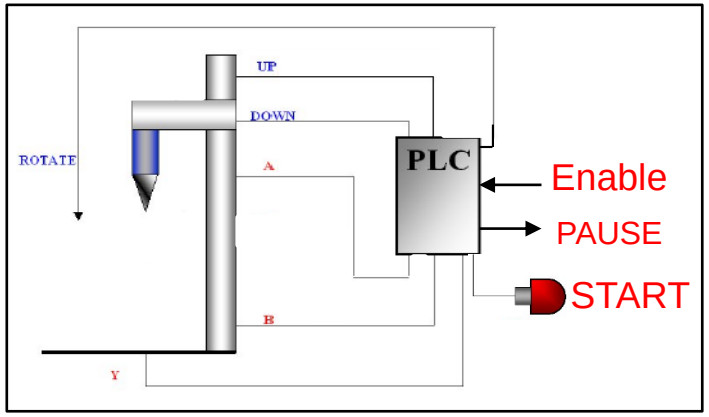
# Il Blocco di Comunicazione in Holobloc

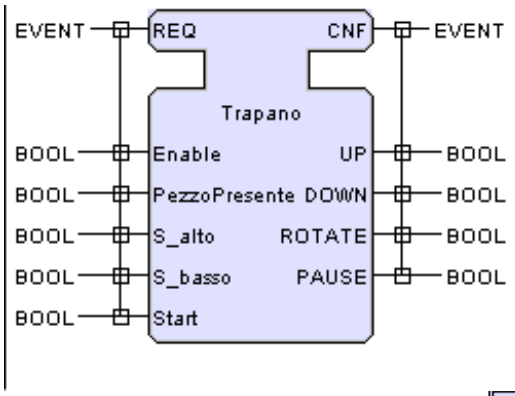
- I Function Block CLIENT_x_y/SERVER_y_x incapsulano TCP/IP client/server socket services
  - ID (PARAMS) coincide con **IP address: port number** o **localhost:port number**
  - *Esempio*
    - ✓ Il SERVER, che è ospitato dal Device 2, ha un ID compost da: indirizzo del Device 2 (localhost) e un numero di porta non utilizzata dal Device 2. Il CLIENT, ospitato nel Device 1 ha un ID composto da: indirizzo del Device 2 (SERVER) e lo stesso numero di porta usato nell'ID del SERVER
    - ✓ Client e Server risiedono nello stesso device, l'ID (per entrambi) verrà settato a: **"**localhost:0001**"**

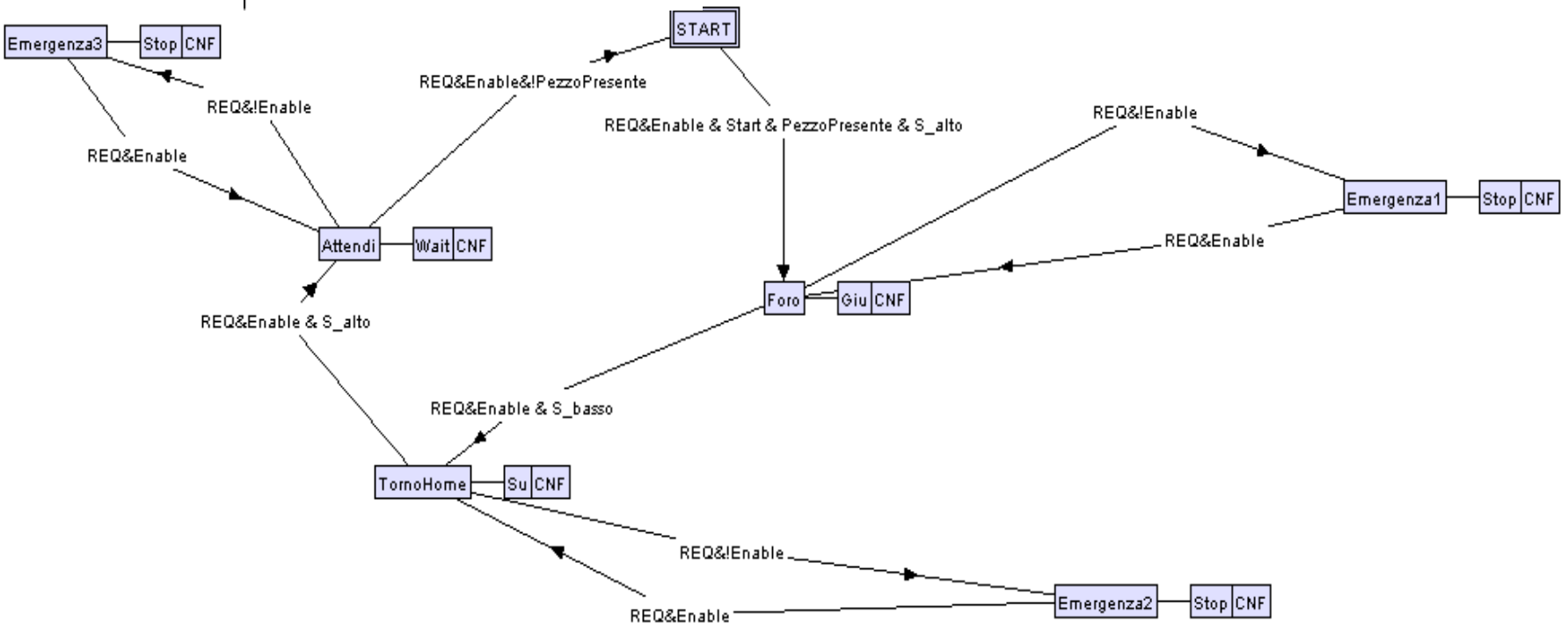


# Esercizi su Blocco di Comunicazione

- Esercizio 6-7: Utilizzo di un FB Basic



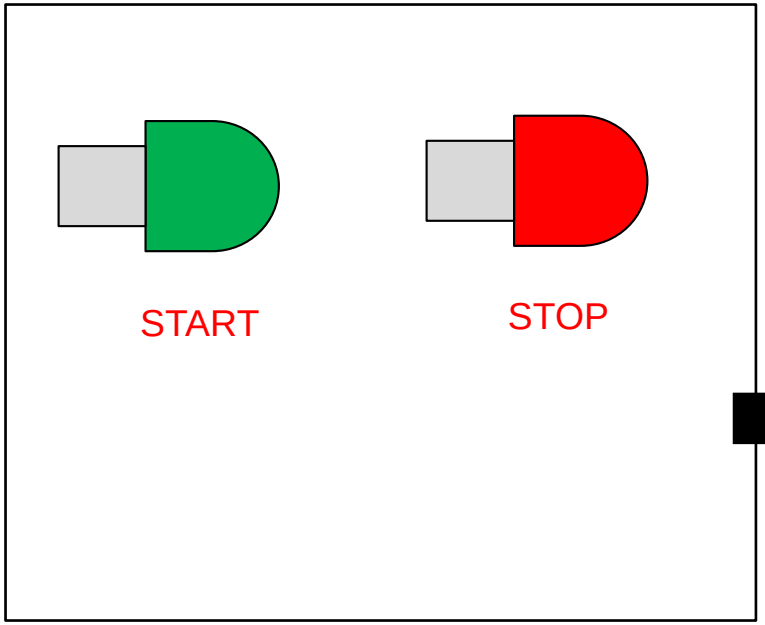



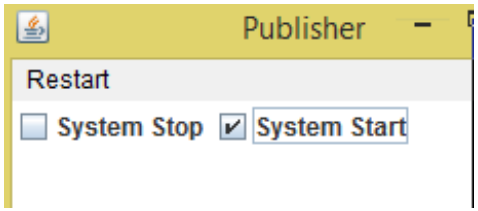



# Esercizi su Blocco di Comunicazione

- Esercizio 6: Realizzazione Applicazione Publish-Subscribe

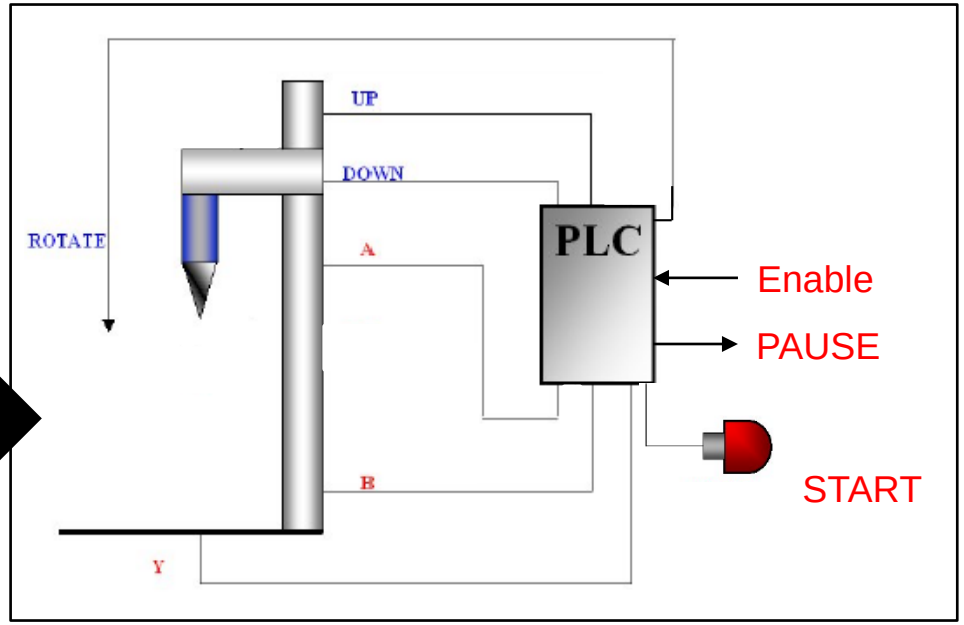
## Publisher

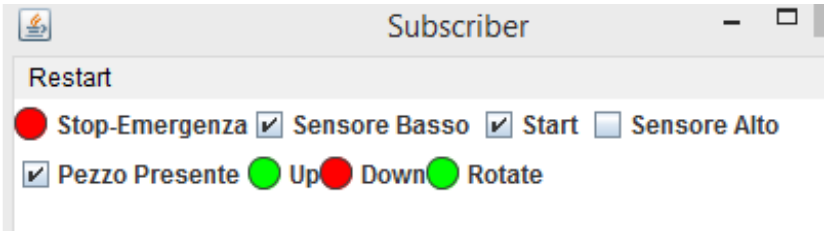



225.0.0.1:0001

## Subscriber



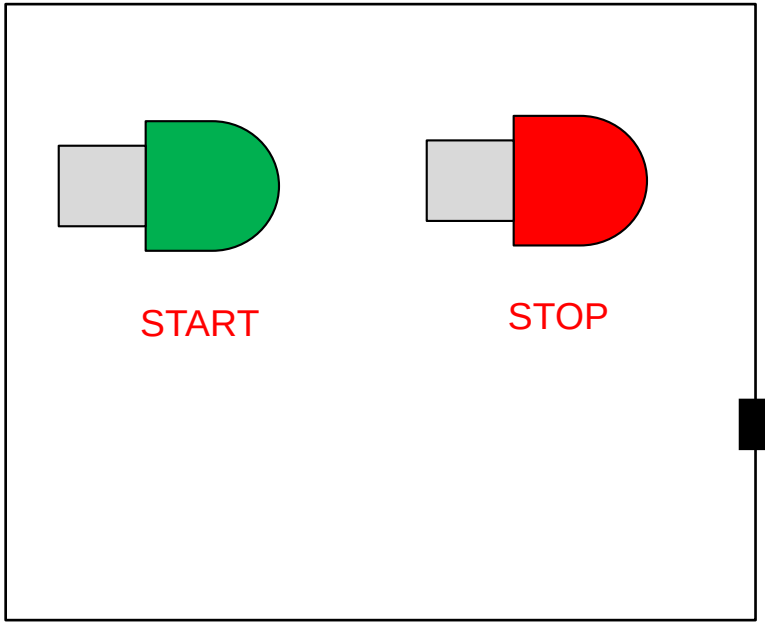




225.0.0.1:0001

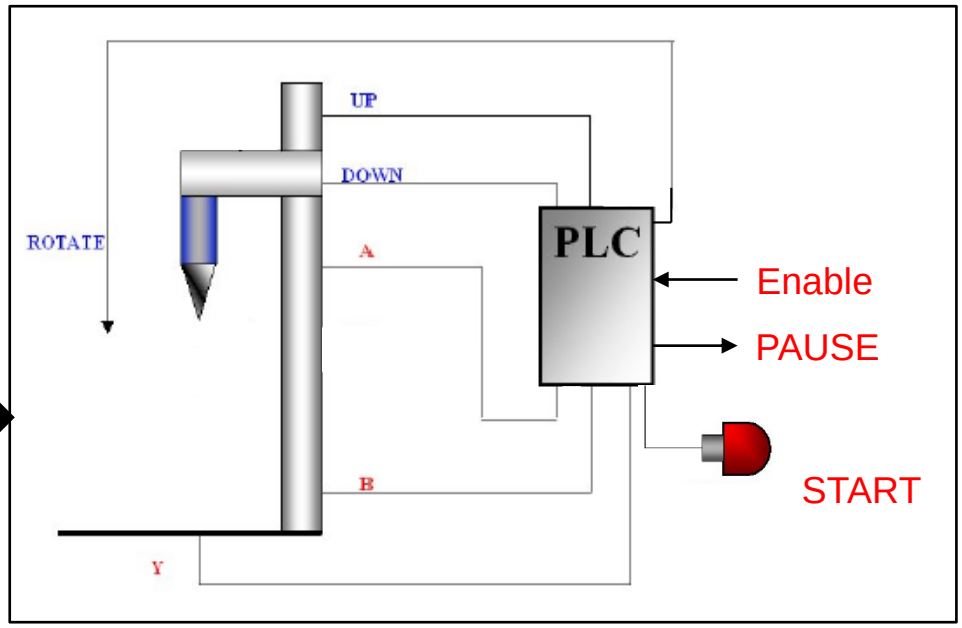
# Esercizi su Blocco di Comunicazione

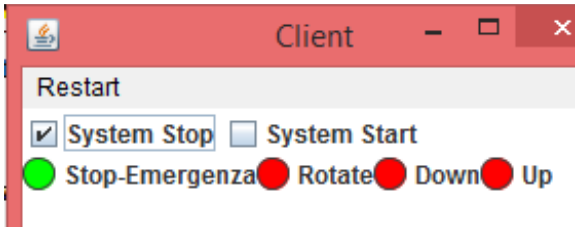
- Esercizio 7: Realizzazione Applicazione Client-Server

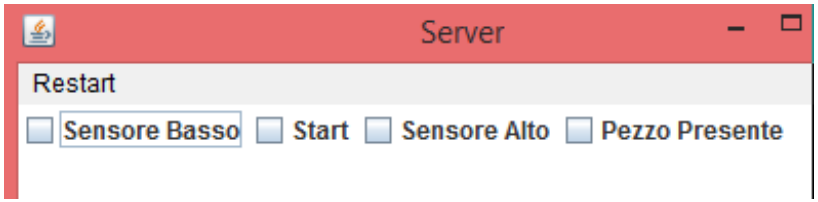
## Client




## Server




localhost:0001

localhost:0001