UNIVERSAL
AUTOMATION.ORG

IEC 61499: primer course

Module 3: IEC 61499 standard

Valeriy Vyatkin
Luleå University of Technology and Aalto University

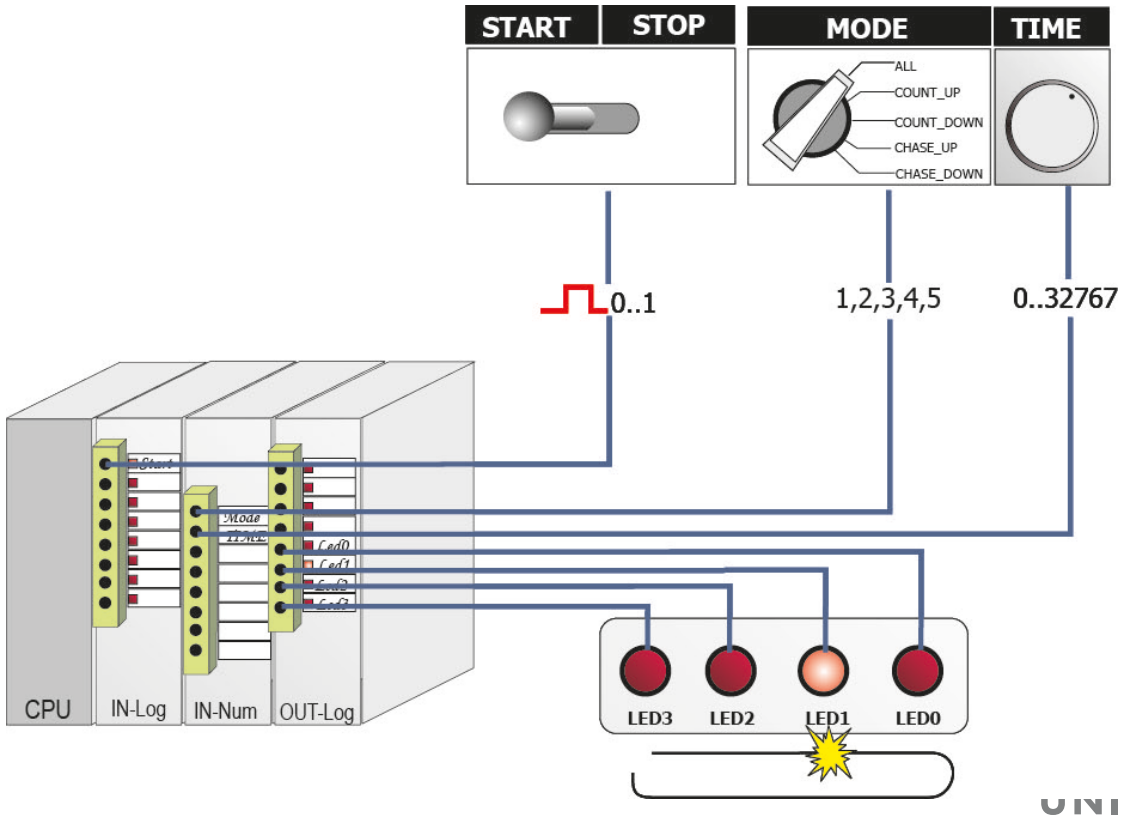
Example of a simple system FLASHER

Example: FLASHER

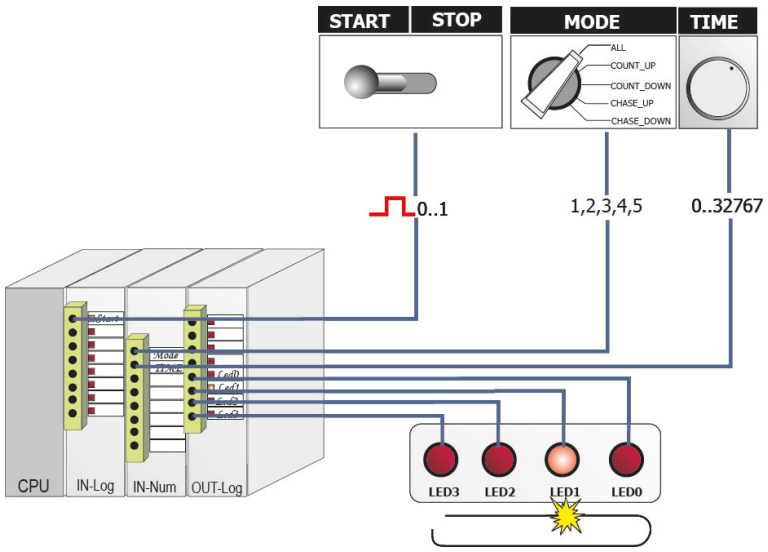
Consider a simple product Flashing Lights

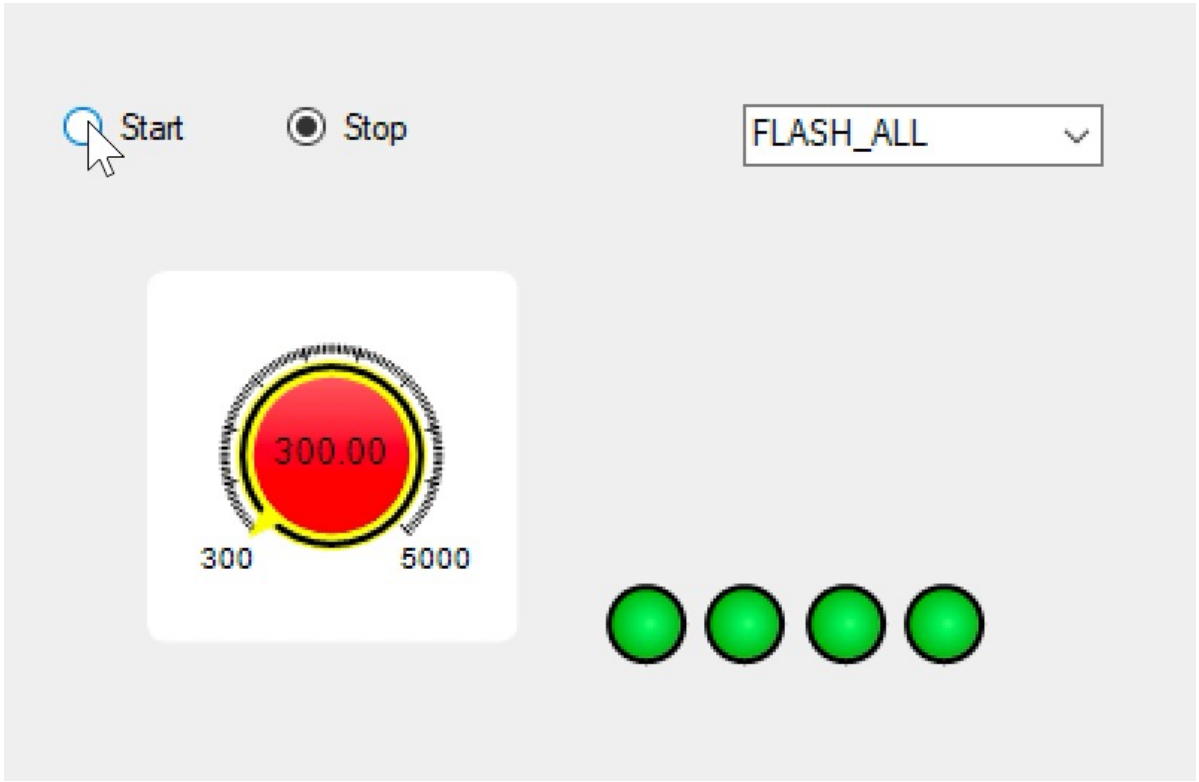
Components of the physical system:

1. PLC (CPU)
2. START/STOP Button
3. Mode Selection Switch
   - All lights on, count up, count down, chase up and chase down
4. Time Delay Nub
   - Delay between 2 consecutive light flashes
5. 4 LEDs

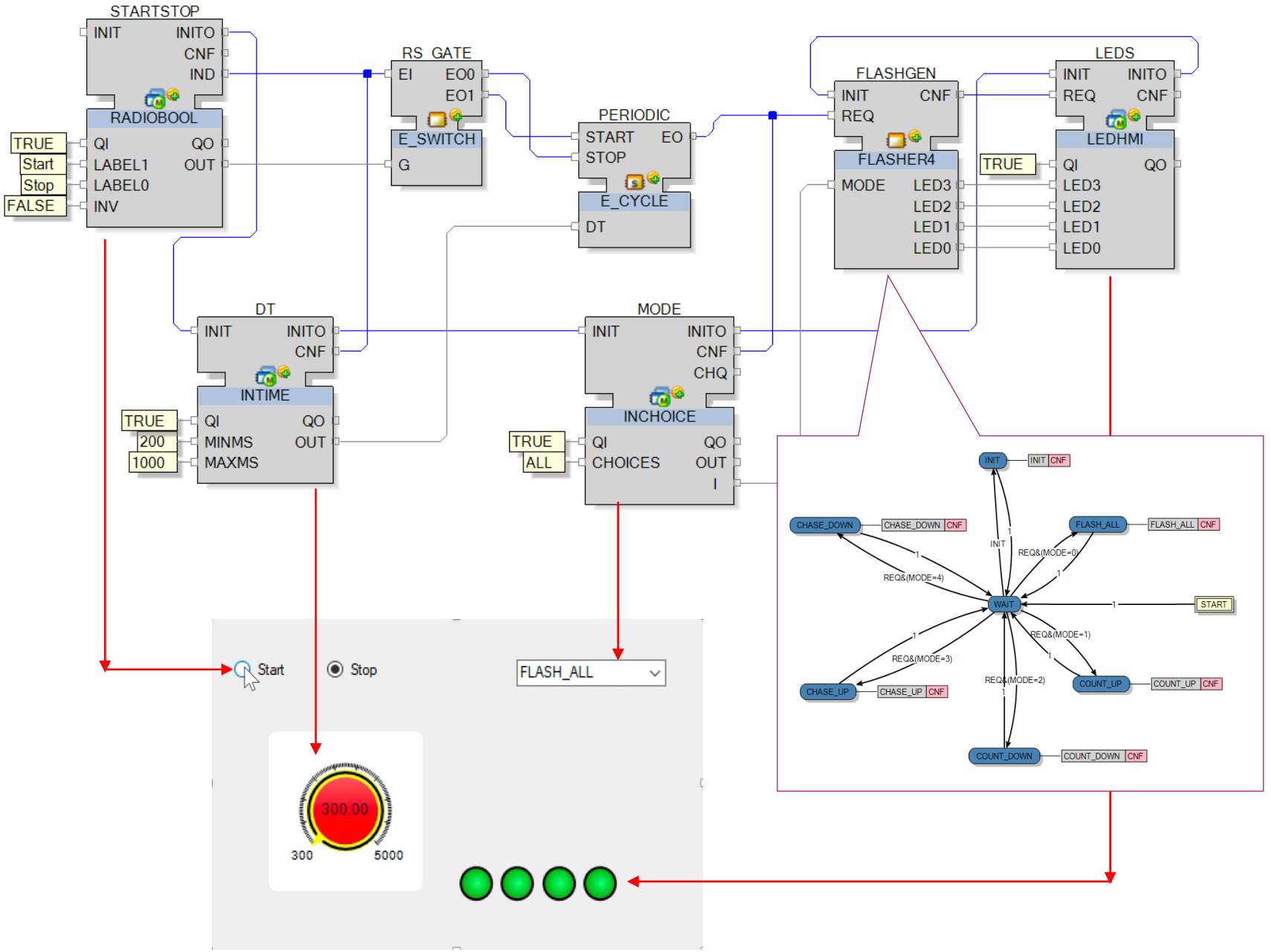


FLASHER modelled in software





FLASHER application

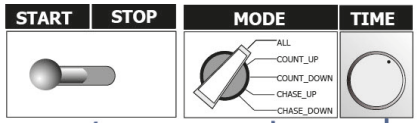


The core functionality is captured in the FLASHER4 FB in a form of state machine

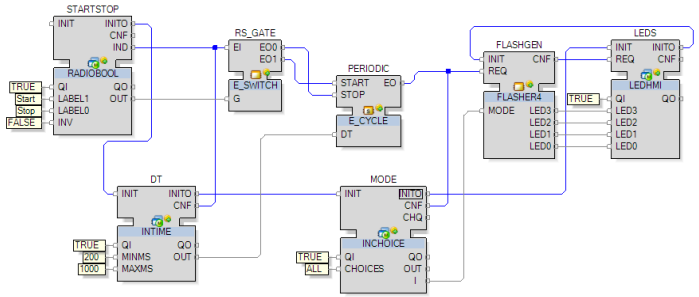
Model and simulate control logic with IEC 61499

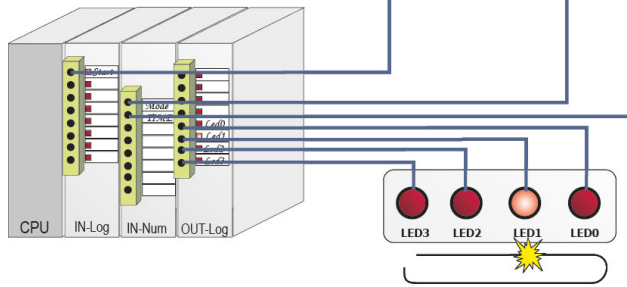
HMI
Runs on a computer

Physical system



IEC 61499 executable model: Runs on PLC or PC (Soft PLC)

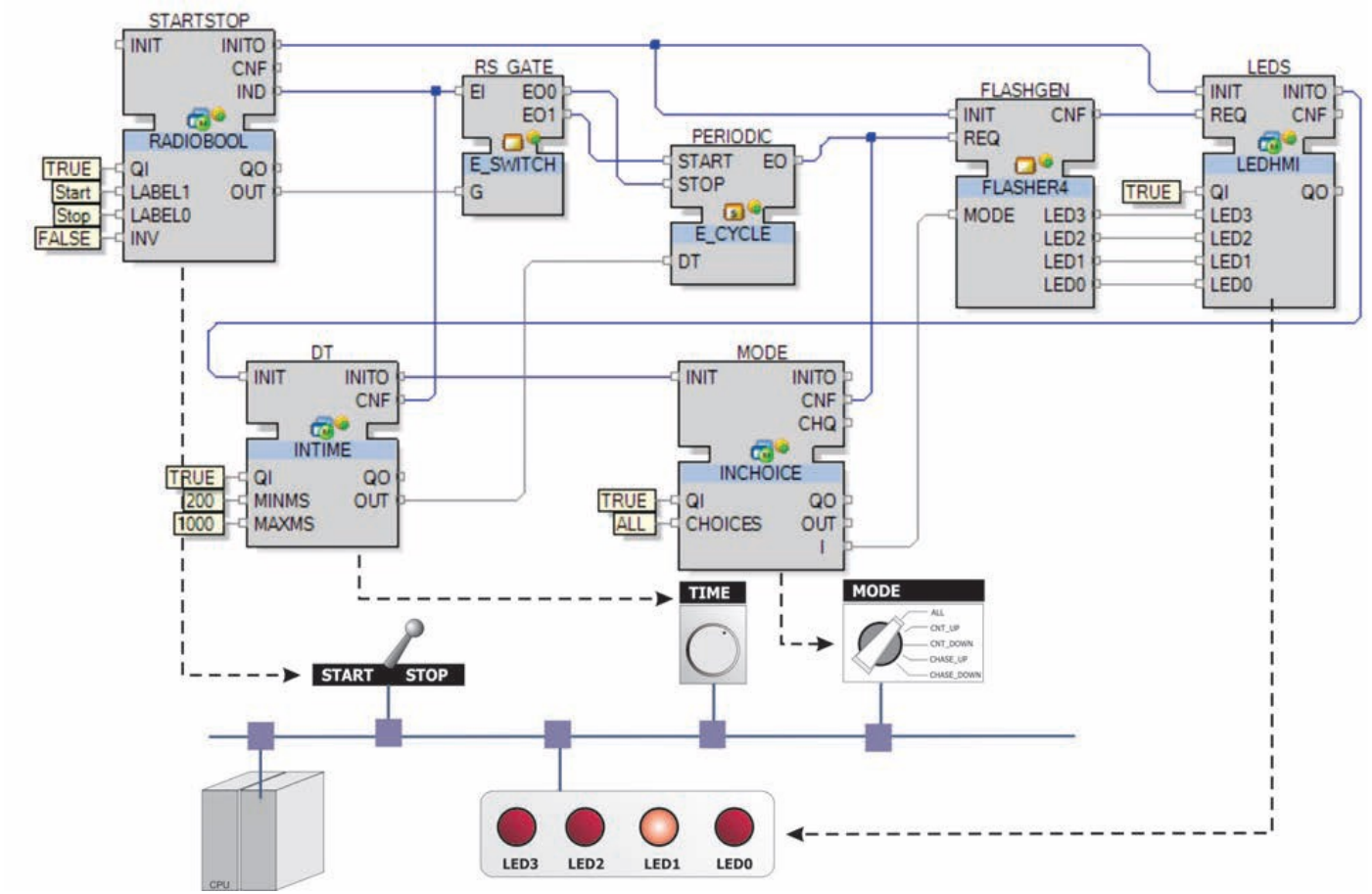




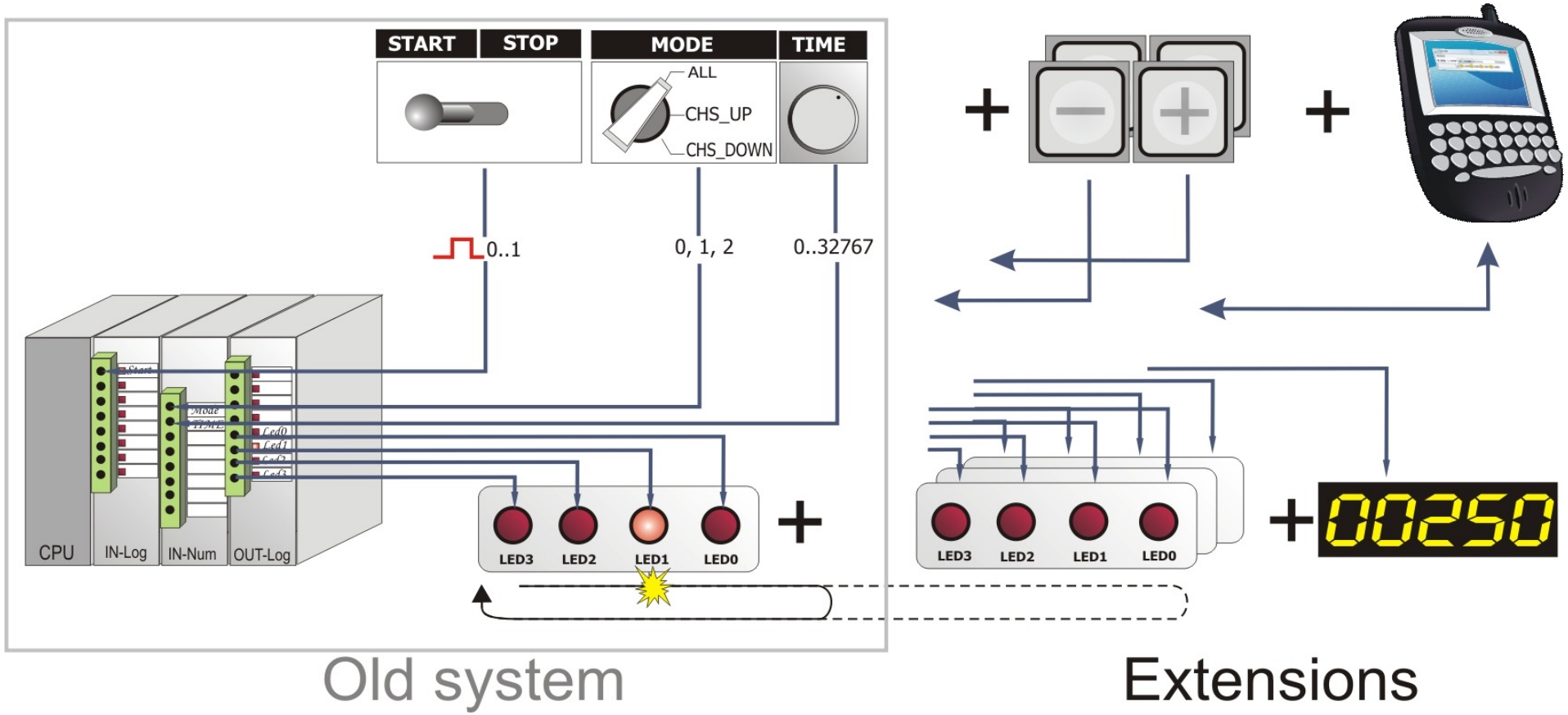
UNIVERSAL AUTOMATION.ORG

Distributed FLASHER

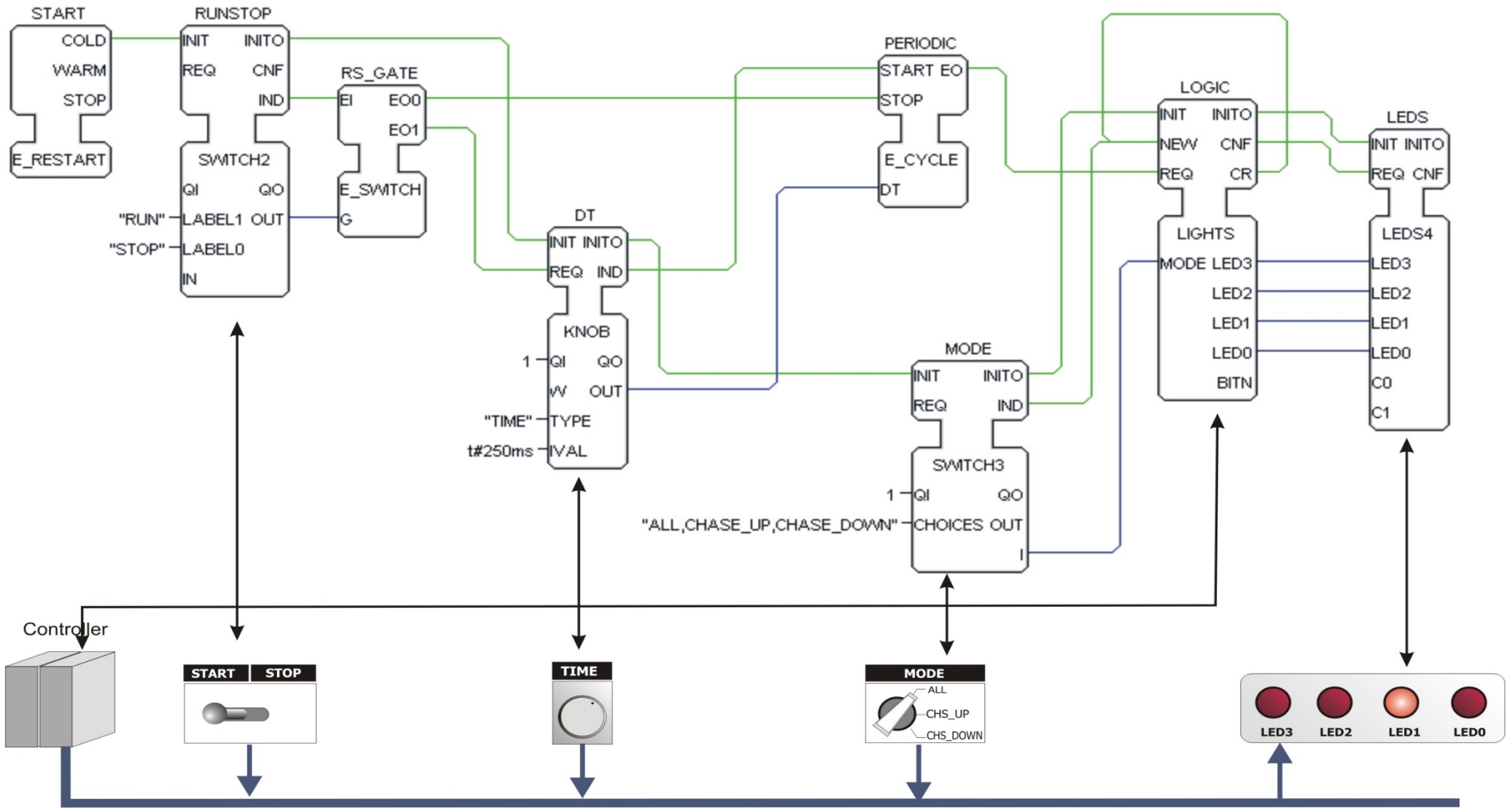
- It is beneficial to model the functionalities in a hardware independent way.
- Same block diagram can be executed on different types of controllers or even on a distributed hardware architecture (running on a network of PLCs).



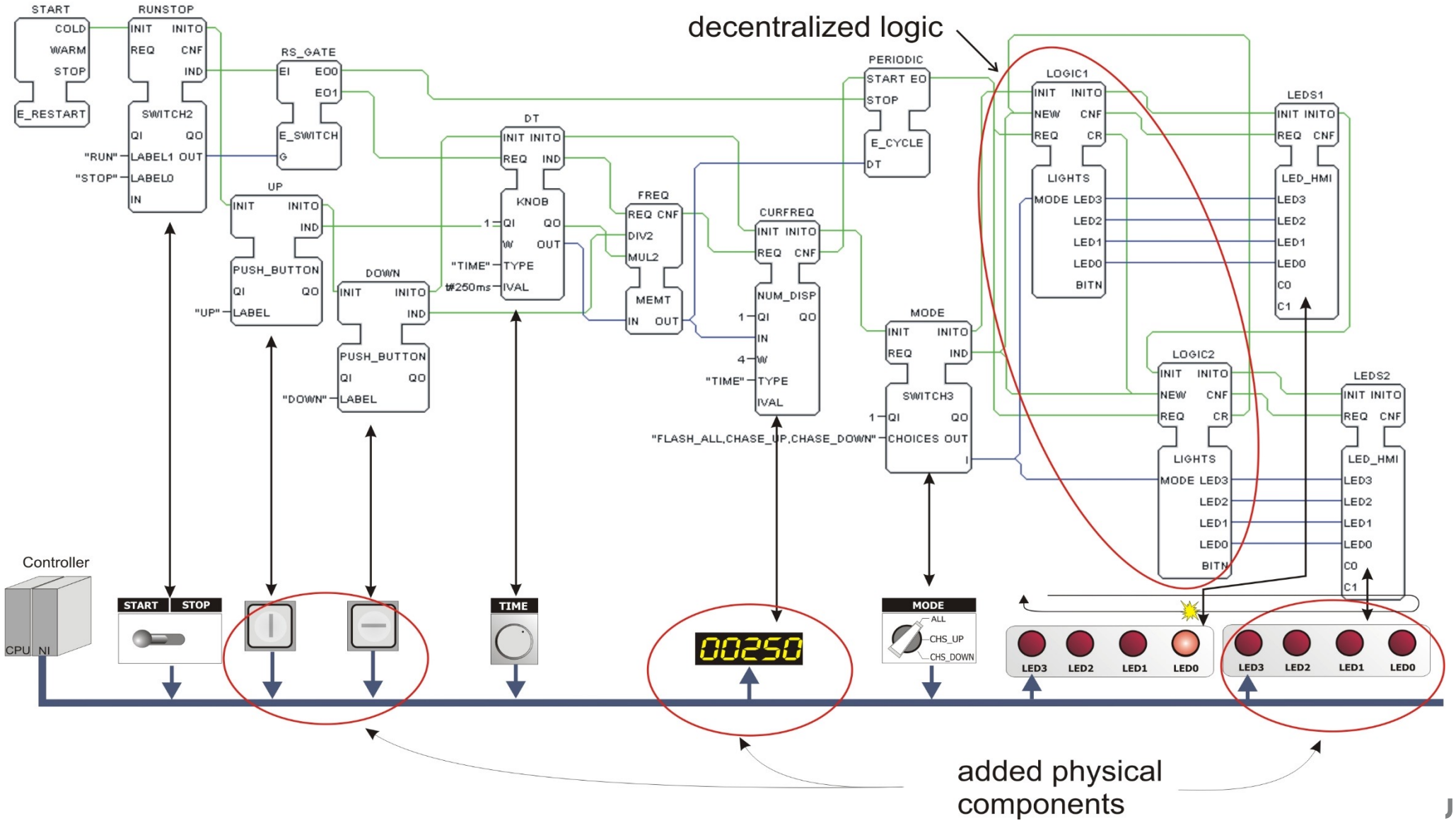
Motivation: Extensibility



Distributed deployment



Extended system

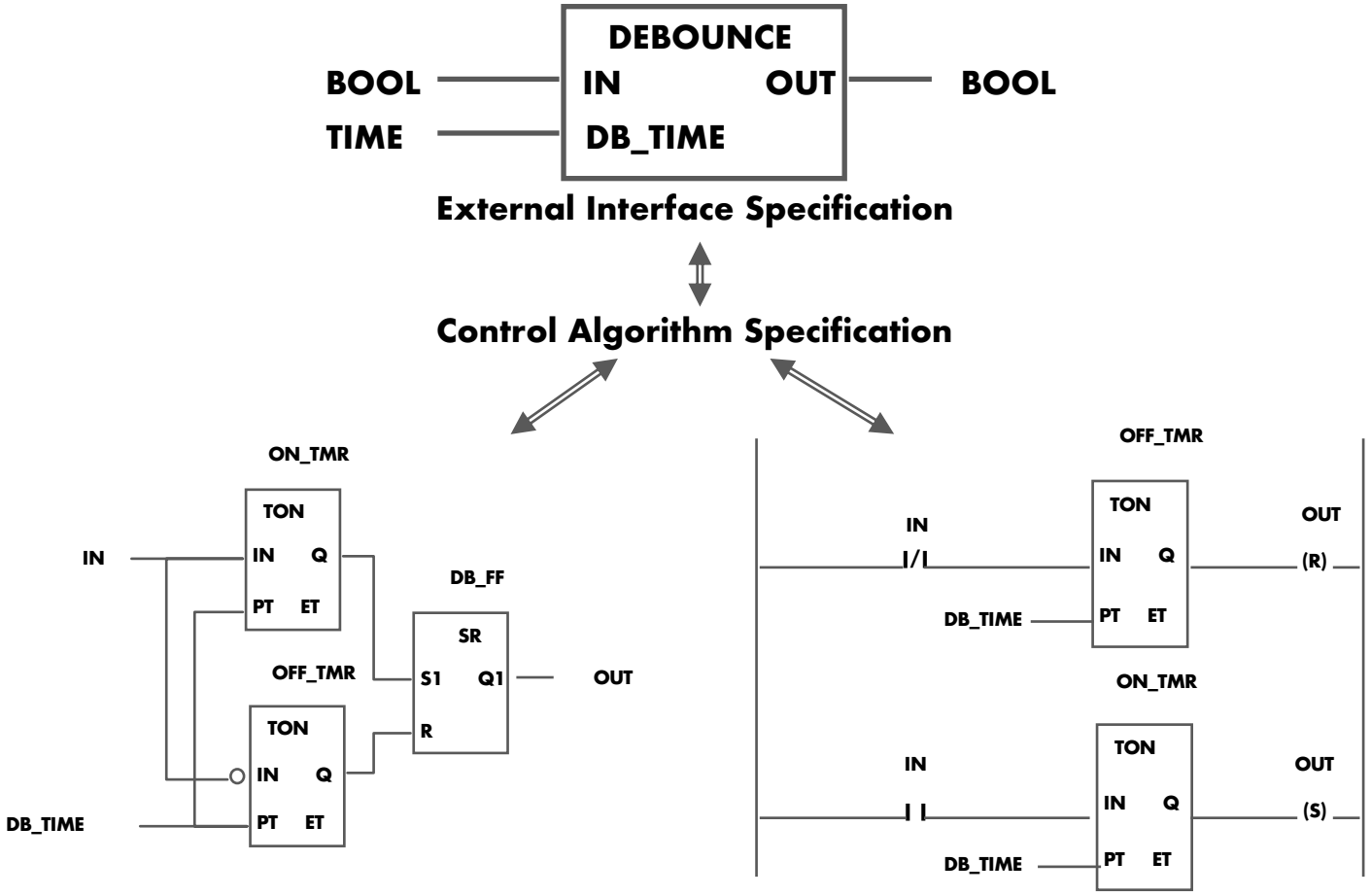


Common elements

Parts of IEC 61499 Standard

- IEC 61499-1:2012
  - Function blocks - Part 1: Architecture
- IEC 61499-2:2012
  - Function blocks - Part 2: Software tool requirements
- IEC TR 61499-3:2004
  - Function blocks - Part 3: Tutorial information (withdrawn)
- IEC 61499-4:2013
  - Function blocks - Part 4: Rules for compliance profiles

Function Block Model: IEC 61131-3

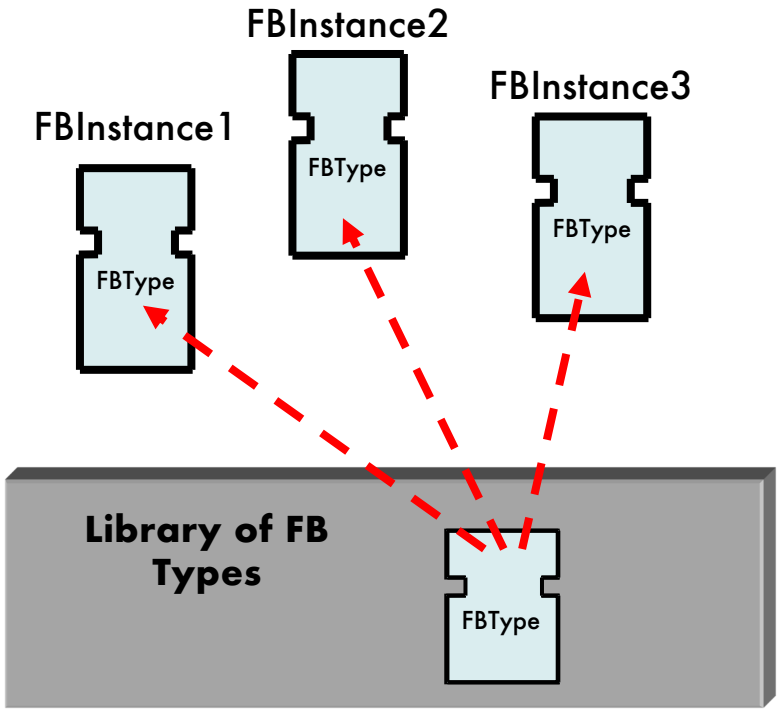


IEC 61131-3 and 61499 Data Types

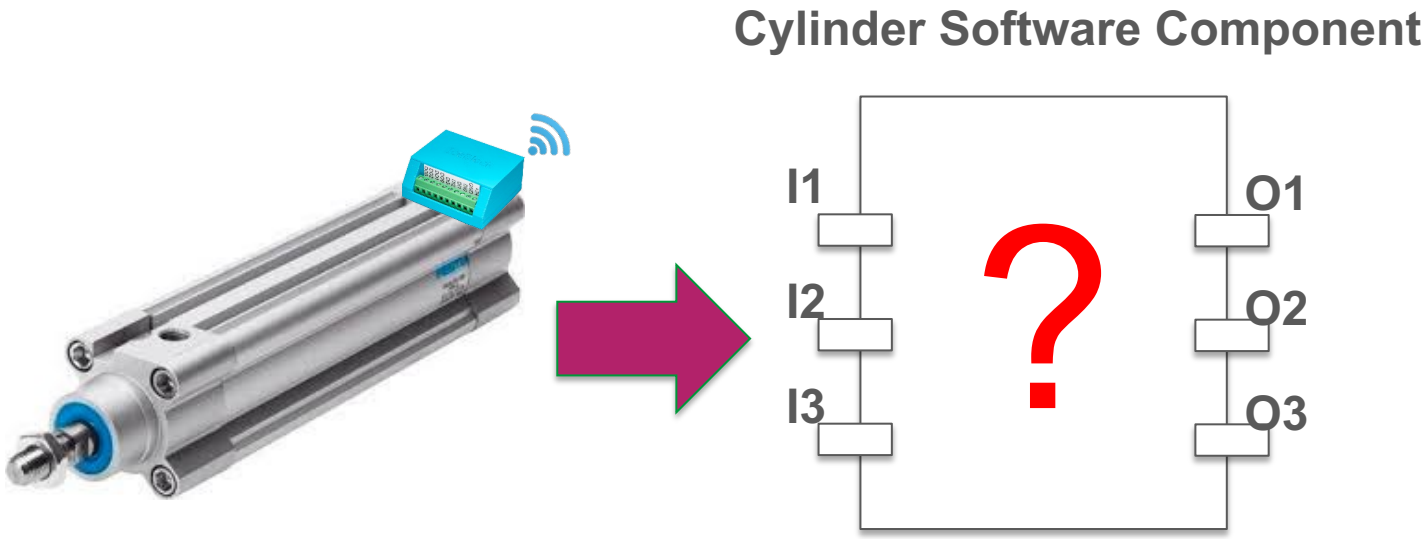
- Signed Integers: SINT(8), INT(16), DINT(32), LINT(64)
- Unsigned Integers: USINT(8), UINT(16), UDINT(32), ULINT(64)
- Floating Point: REAL(32), LREAL(64)
- Bit Strings: BOOL(1), BYTE(8), WORD(16), DWORD(32), LWORD(64) (TRUE, FALSE, 1, 0, 255, 16#FF, etc.)
- Character Strings: STRING(8)
- Duration: TIME (t#2s, t#1500ms, etc.)
- Time: TIME_OF_DAY or TOD DATE DATE_AND_TIME or DT (TOD#17:32:55.678 D#2005-06-07 DT#2005-06-07-17:32:55, etc.)
- Derived Data Types: array, enumerated, structured, …

Function Block Type System

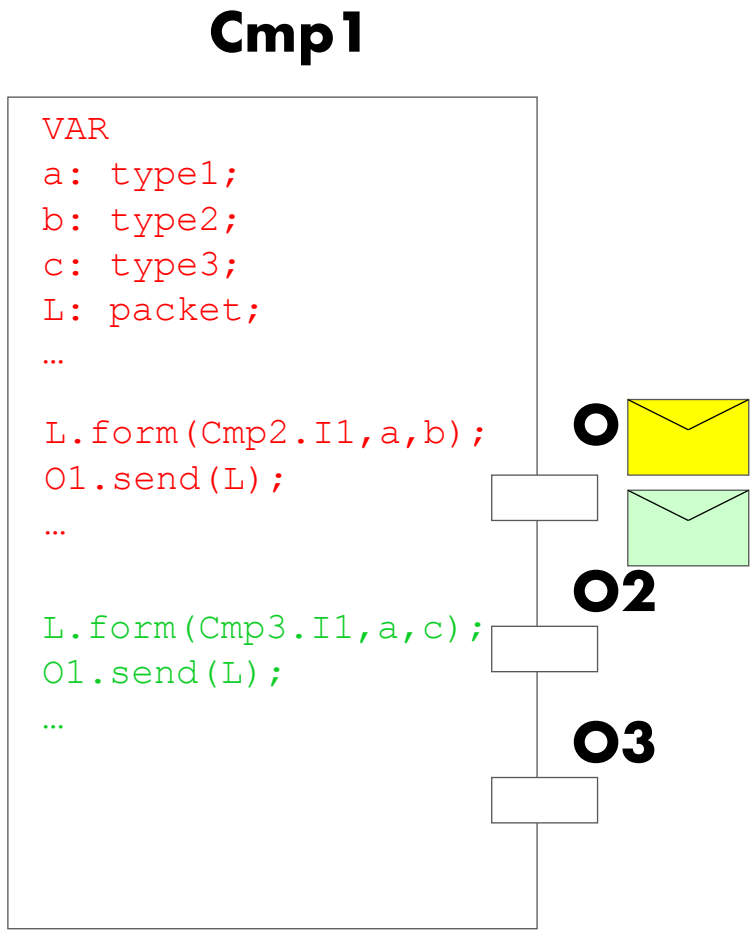
- Library contains FB type definition
- FB types can be instantiated later
- Each FB instance can have its own
- configuration/setting
- Changes in FB types cause automatic changes on all instances

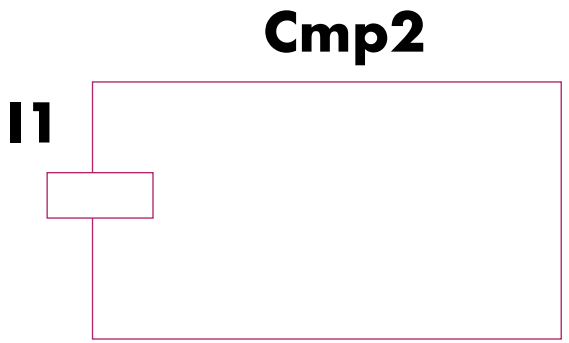


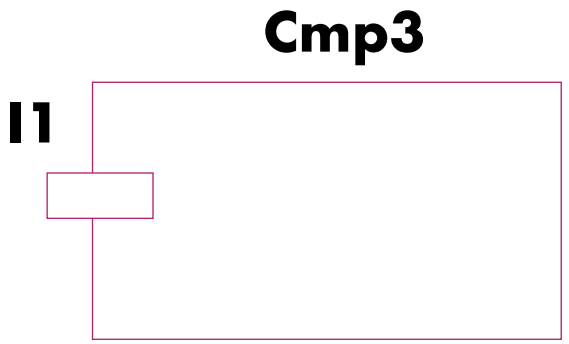
Motivation: Intelligent Automation Component



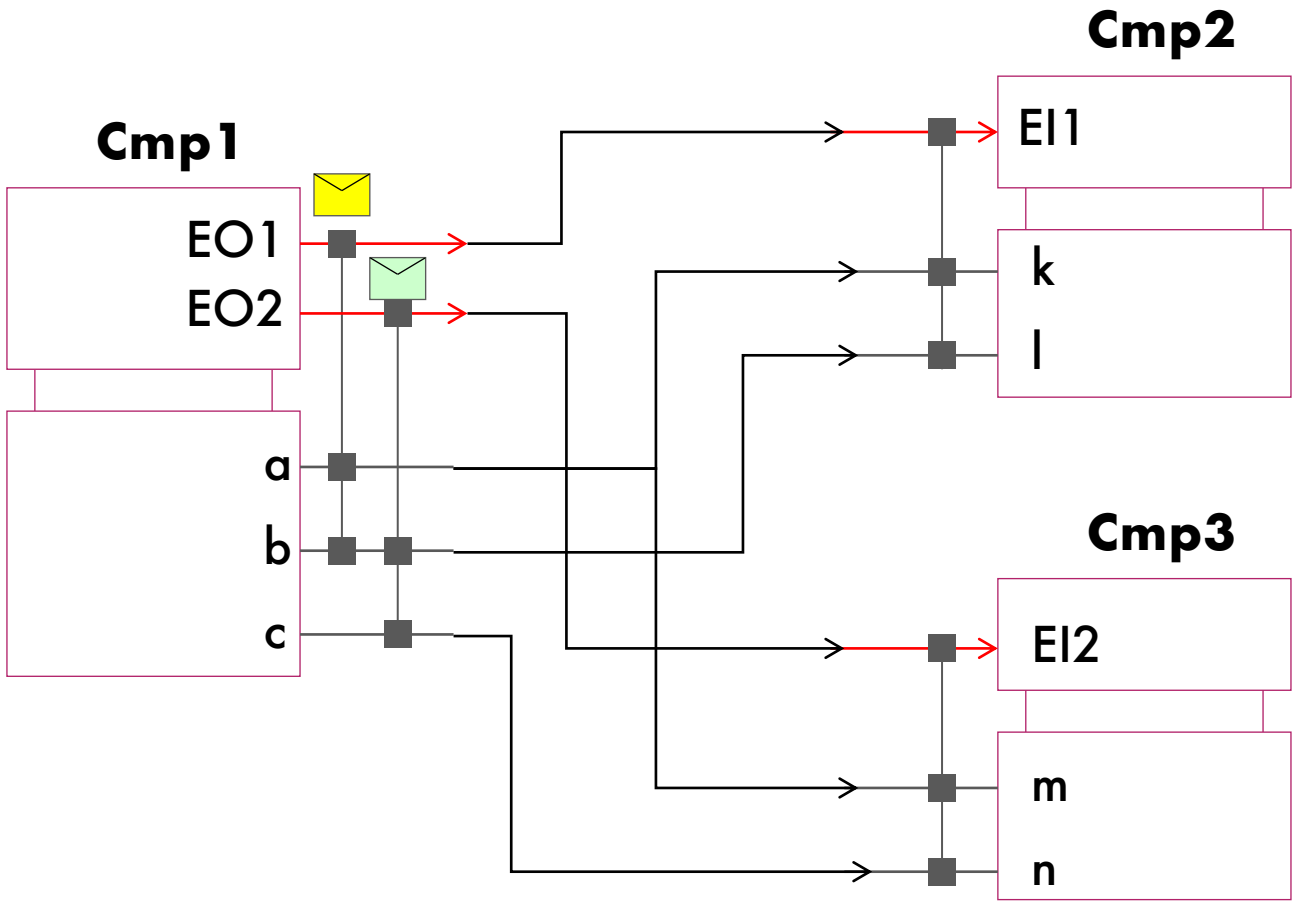
Communicating components







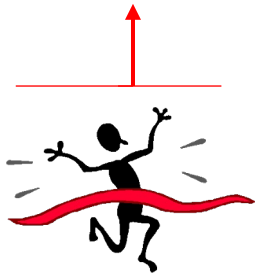
Message=Event + associated Data



Events

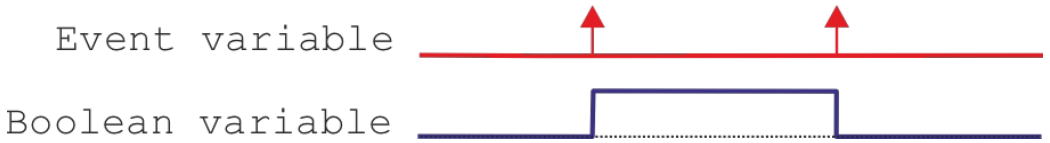
- Event
  - Event variables
  - Boolean
    - 0 and 1
  - No duration (short duration)

Event of crossing the line



- IEC 61499 function block

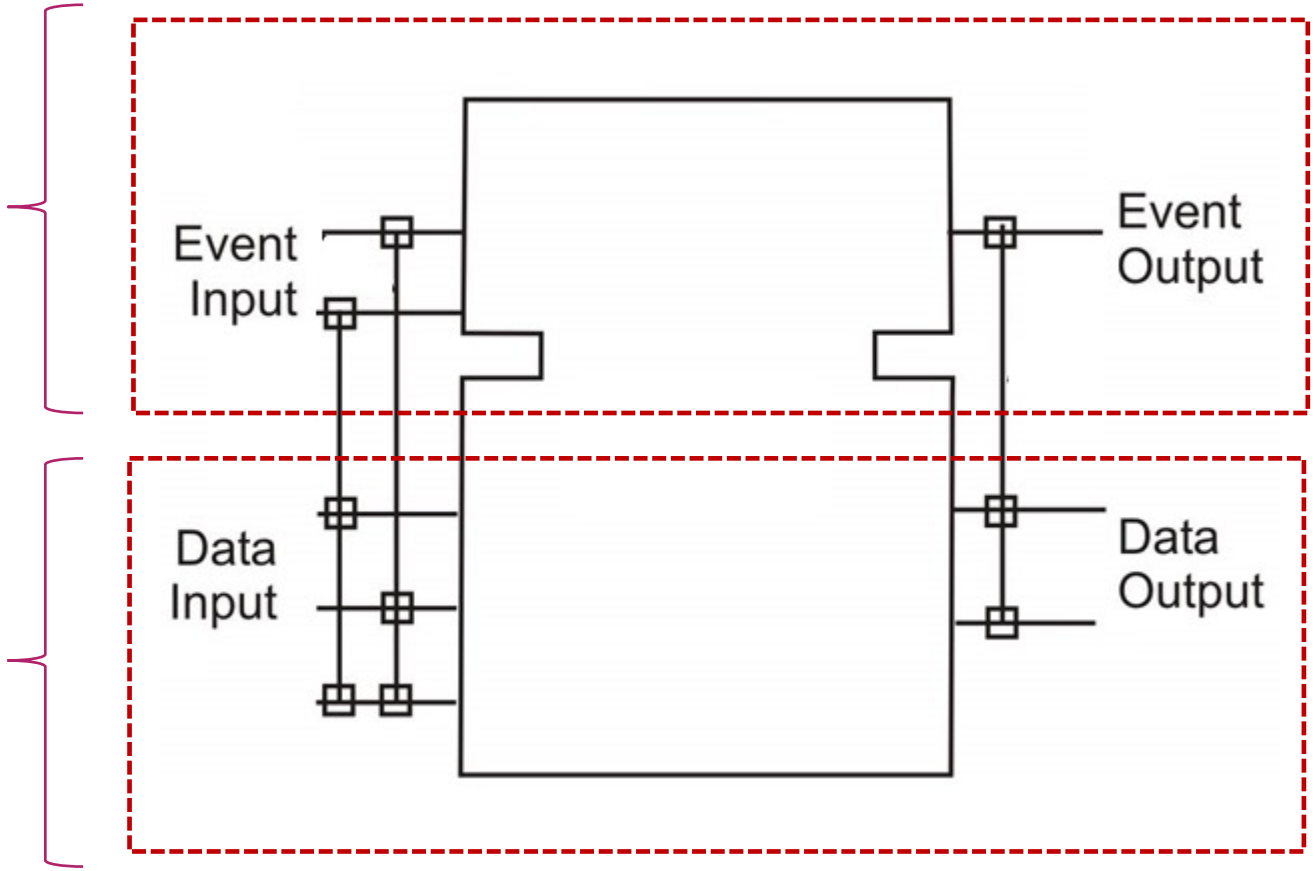
can only be activated by an event



Function Block: Header and Body

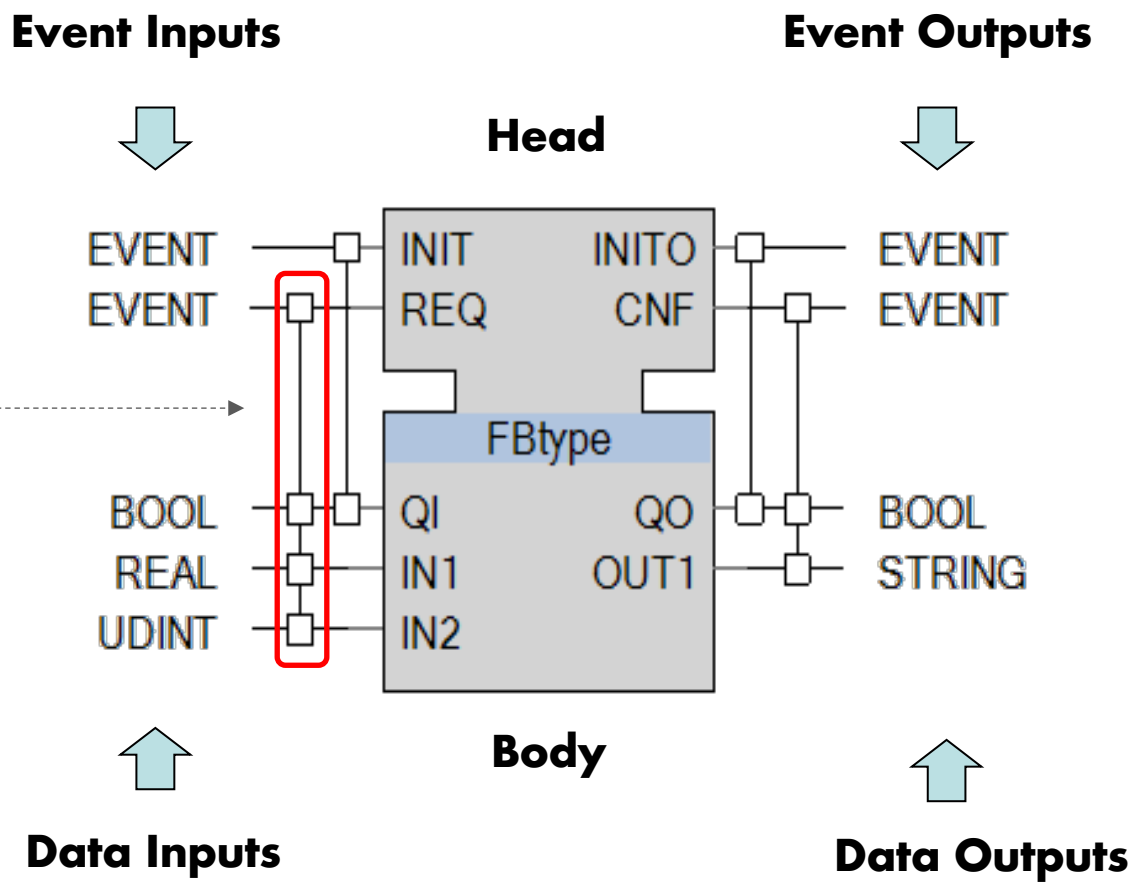
Upper part of FB is called **Header**

Lower part of FB is called **Body**



Function Block of IEC 61499: Interface

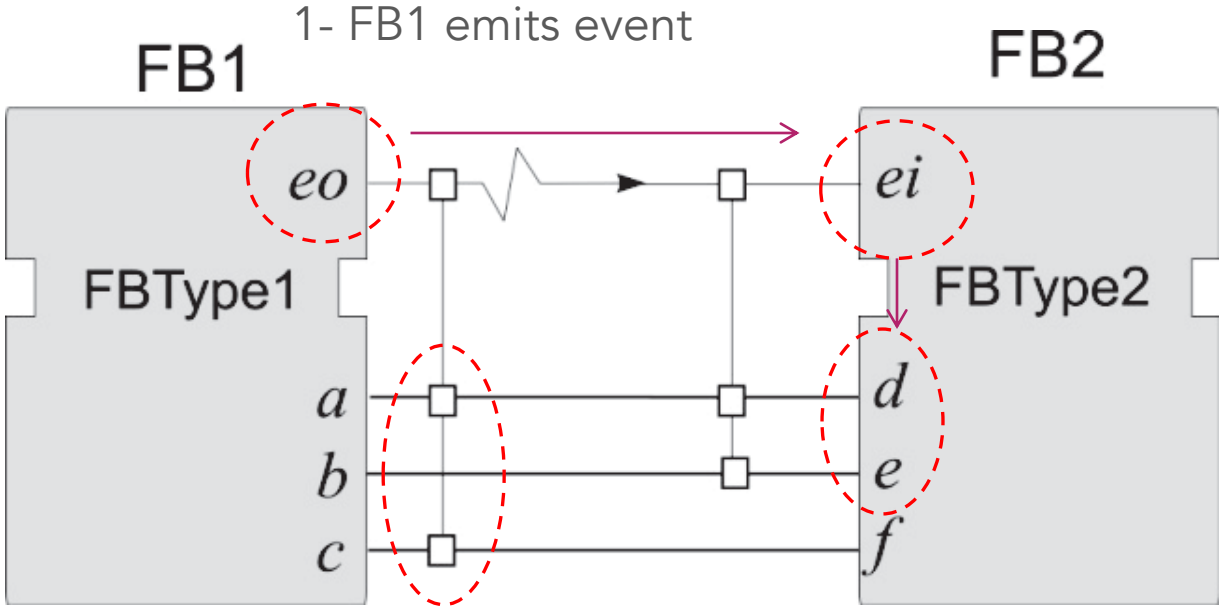
Event – data association: *if the event occurs, only the associated data will be updated.*



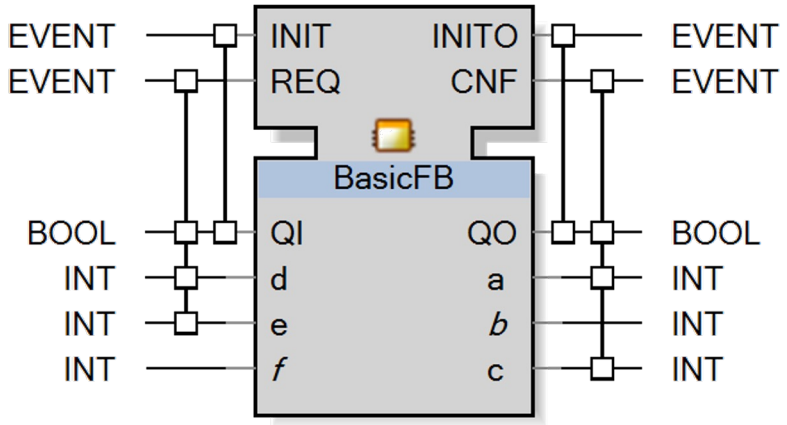
Function block: Event-Data Association

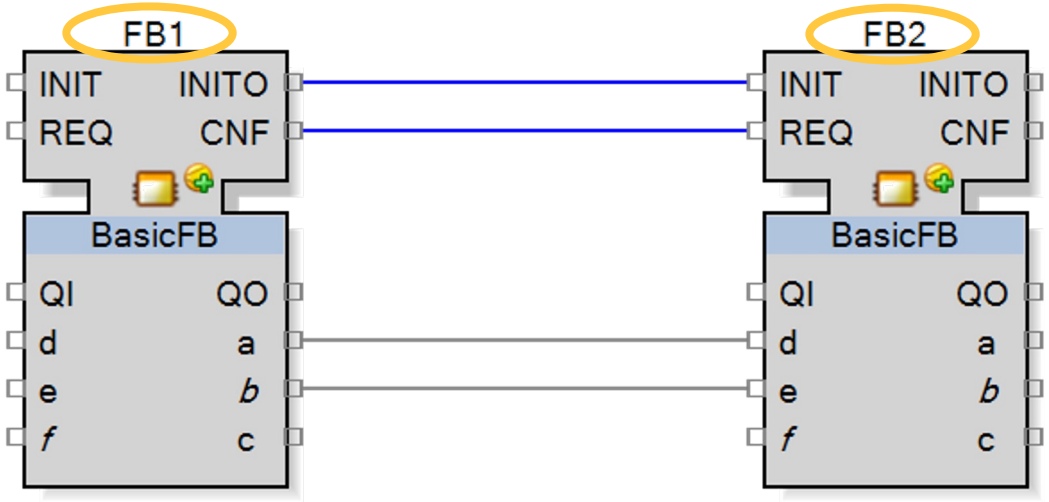
It is used to transfer data between FBs

- Event output "*eo*" of FB1 is connected by an association line to the event input "*ei*" of FB2.
- Once FB1 emits "*eo*", it triggers the execution of FB2.
- The values of input parameters "*d*" and "*e*" will be updated before the execution starts because they are associated with the event input "*ei*".

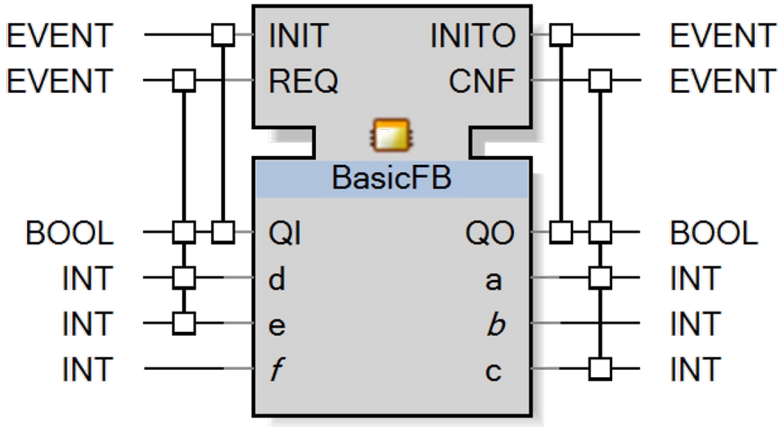


Event-data association

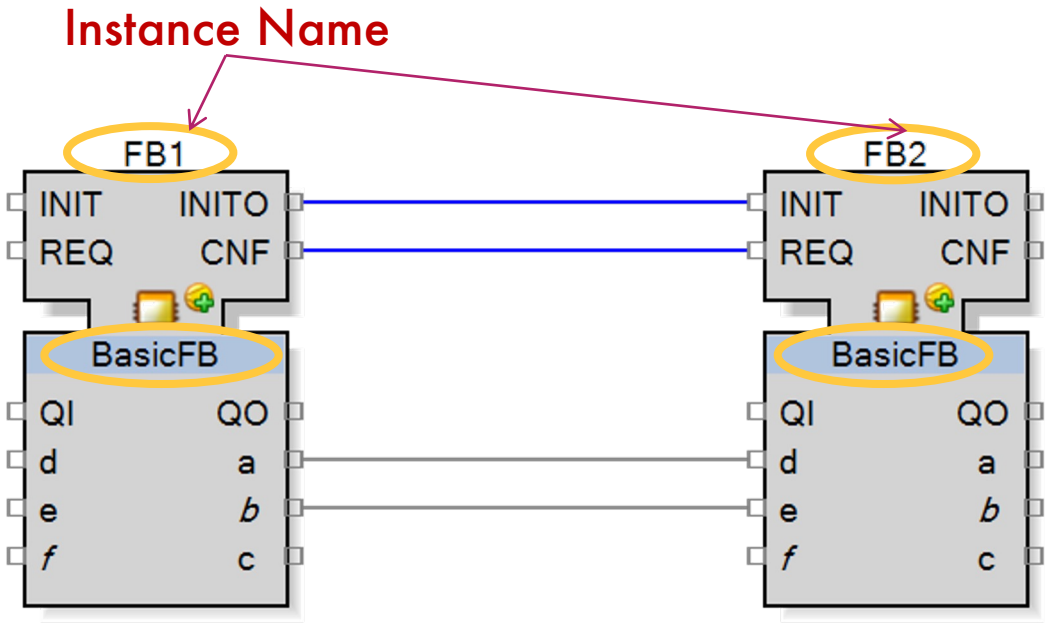




Event-data association

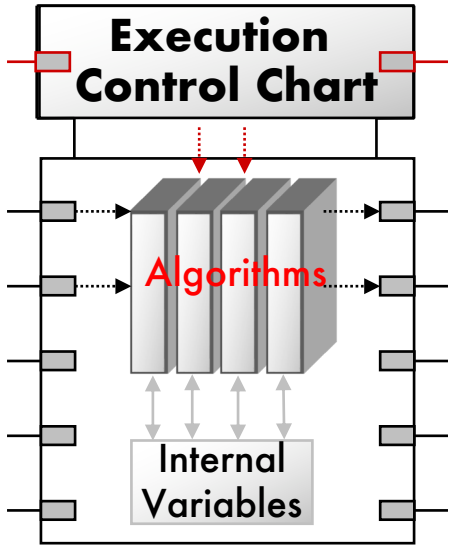


- Two instances FB1 and FB2 of the same FB type
  - Suppose event FB1.CNF is emitted
  - FB1.a is updated and value is sent to FB2.d.
  - FB2.d is sampled
  - FB1.b is not updated (no association). Therefore FB2.e will not receive the updated value.

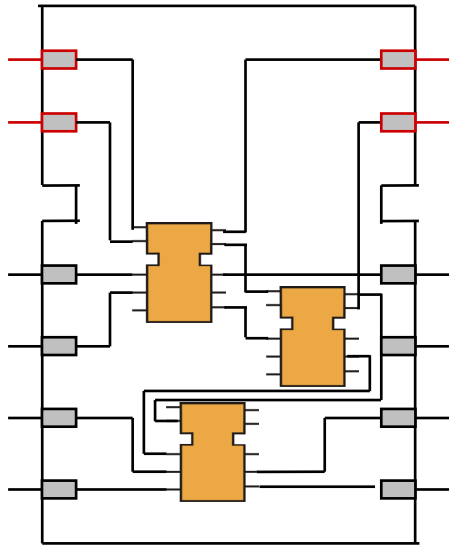


IEC 61499 Function Block Kinds

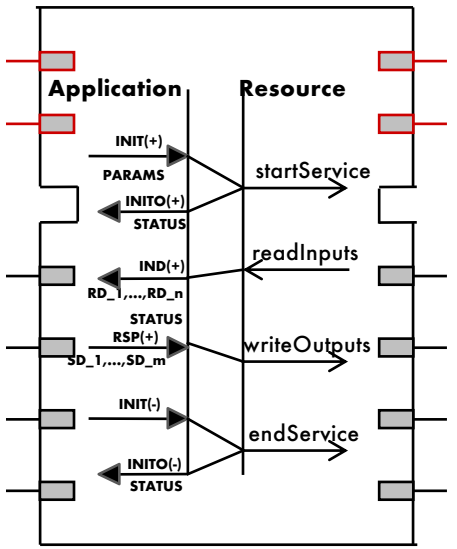
- Basic, Composite, and Service Interface Function Blocks



BFB

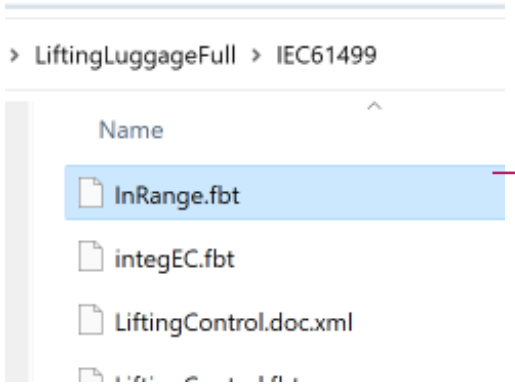


CFB



SIFB

XML Exchange Format



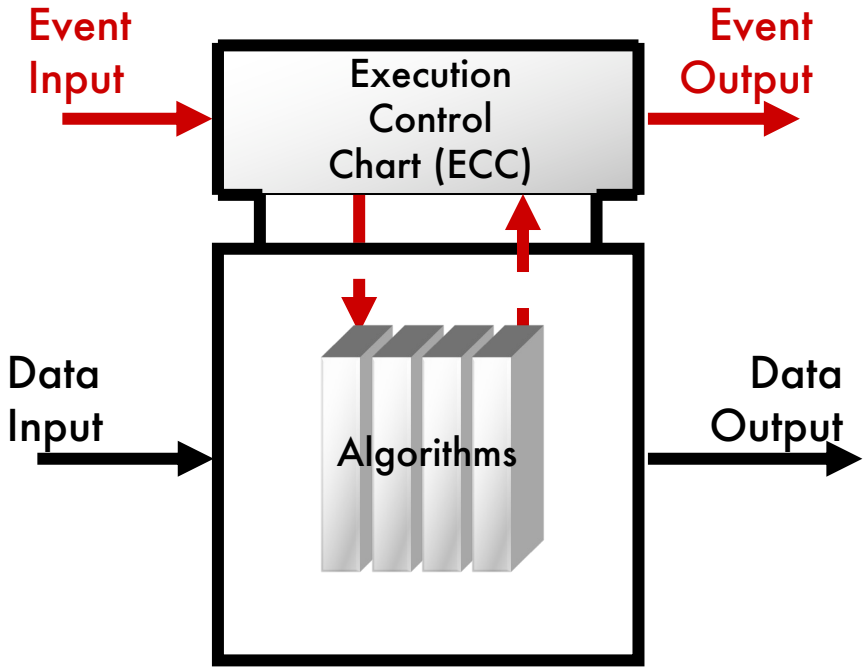
InRange.fbt

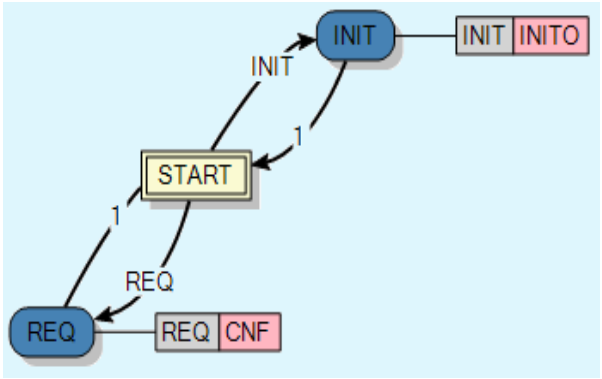
File Edit View

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE FBType SYSTEM "../LibraryElement.dtd">
<FBType GUID="997C1FE59E5FA6CD" Name="InRange" Comment="Basic Function Block
Type" Namespace="Main">
  <Attribute Name="Configuration.FB.IDCounter" Value="0" />
  <Identification Standard="61499-2" />
  <VersionInfo Organization="nxtControl GmbH" Version="0.0" Author="Horst
Mayer" Date="9/14/2008" Remarks="Template" />
  <InterfaceList>
    <EventInputs>
      <Event Name="INIT" Comment="Initialization Request">
        <With Var="pos" />
        <With Var="SensorPos" />
        <With Var="Range" />
      </Event>
      <Event Name="REQ" Comment="Normal Execution Request">
        <With Var="pos" />
        <With Var="SensorPos" />
        <With Var="Range" />
      </Event>
    </EventInputs>
    <EventOutputs>
      <Event Name="INITO" Comment="Initialization Confirm">
        <With Var="result" />
      </Event>
      <Event Name="CHG" Comment="Execution Confirmation">
        <With Var="result" />
      </Event>
    </EventOutputs>
    <InputVars>
      <VarDeclaration Name="pos" Type="REAL" />
```

Basic Function Block

Basic Function Blocks

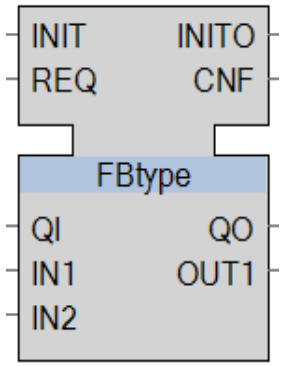


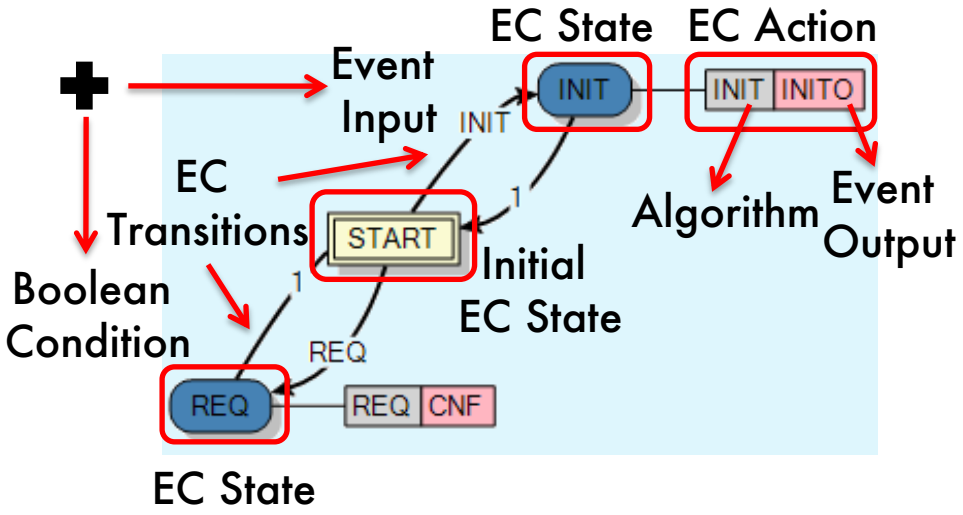


Algorithms can be programmed in PLC languages

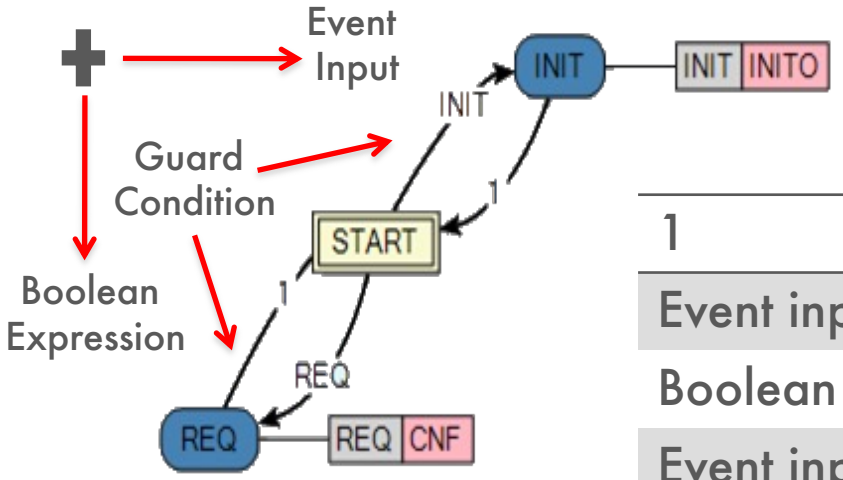
- Structured Text
- Ladder Logic
- etc.

Execution Control Chart (ECC)





Transition conditions



A transition condition can be one of the following:

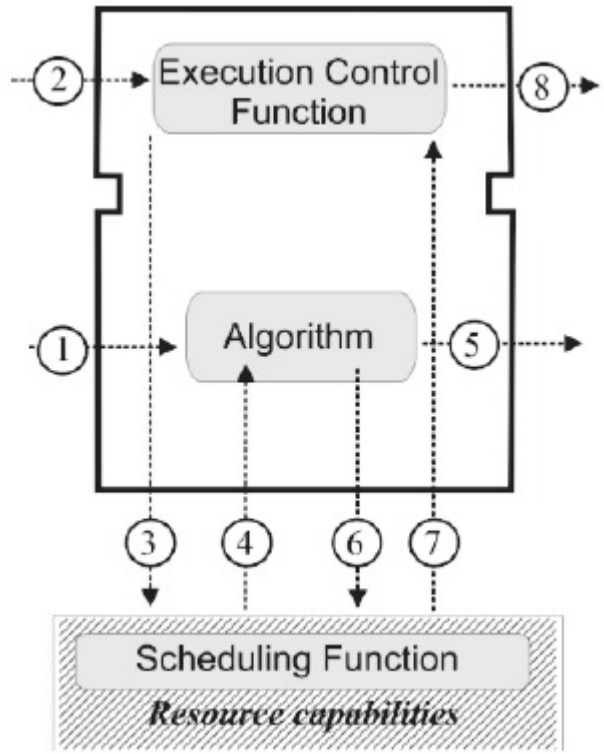
1
Event input
Boolean expression over data
Event input {&}[Boolean expression over data]

Examples:

1. REQ
2. ((Input_Var=1) OR (Internal_Var=0)) AND (NOT QO)
3. REQ AND (Input_Var=0)
4. REQ [Input_Var=0]

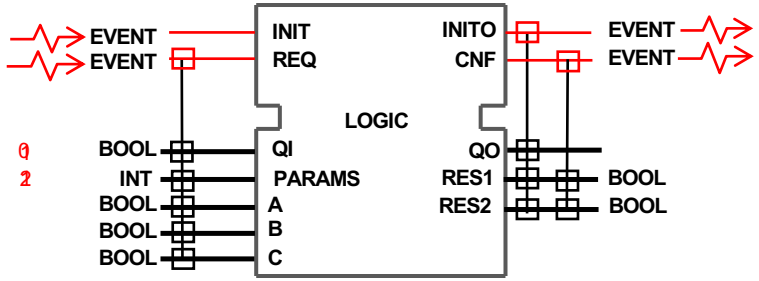
New syntax

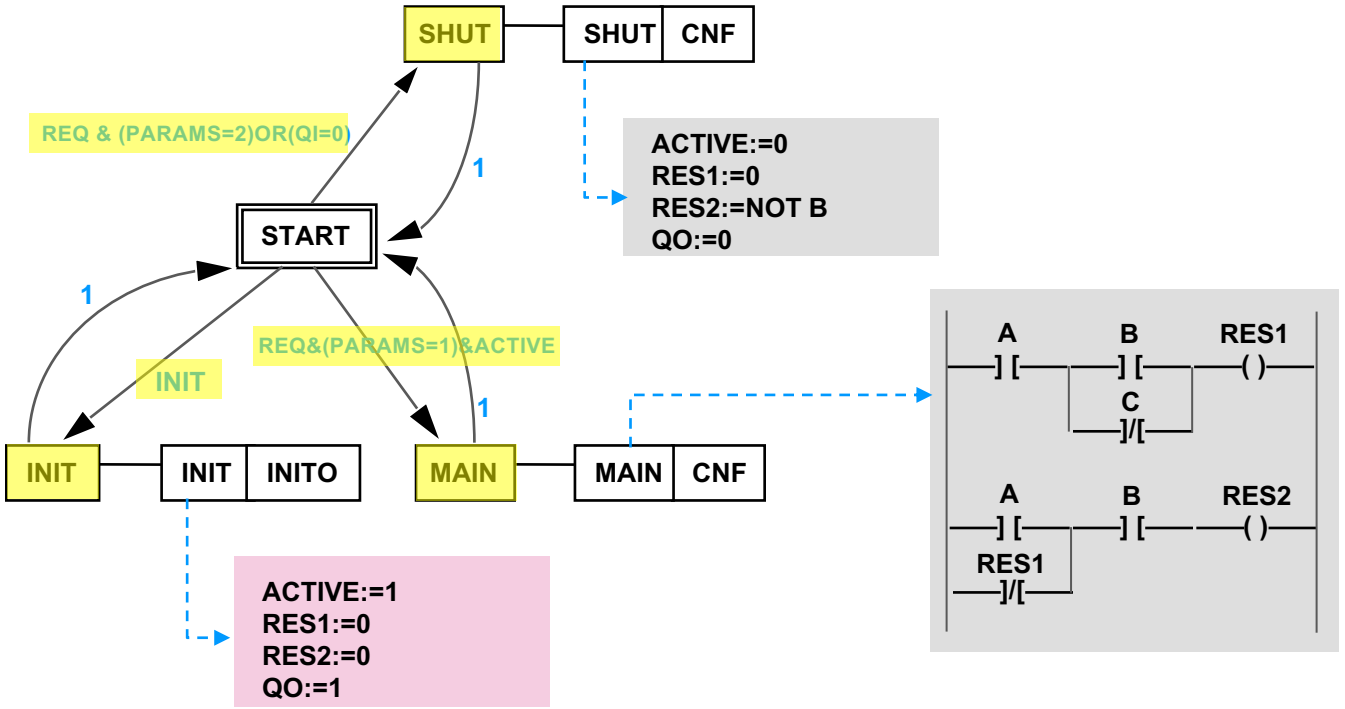
How does Basic Function Block work?



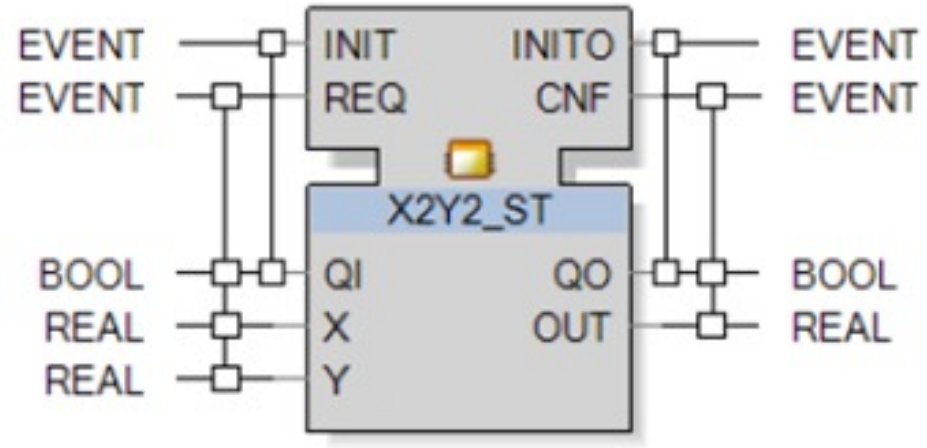
Step 1. The input variable values relevant to input event are made available
Step 2. The input event occurs, the execution control of the FB is triggered.
Step 3. The execution control function evaluates the ECC and notifies the scheduling function to schedule algorithm for execution.
Step 4. Algorithm execution begins.
Step 5. The algorithm completes the assignment of values for the output variables associated with the event output.
Step 6. The resource scheduling function is notified that algorithm execution has ended.
Step 7. The scheduling function invokes the execution control function.
Step 8. The execution control function signals event at the event output.

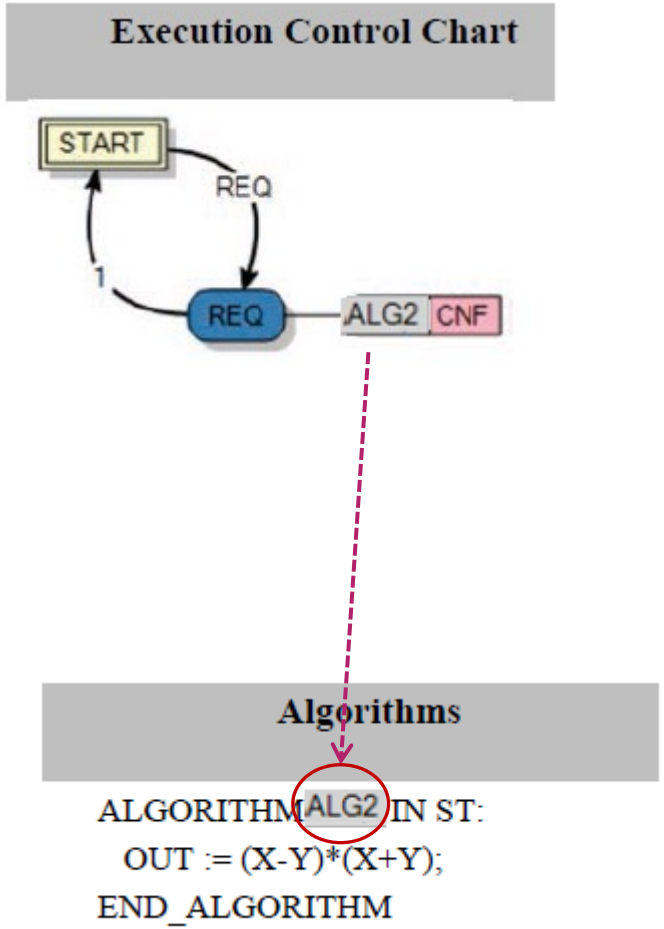
How does Basic Function Block work?





Basic FB – execution control chart (ECC)



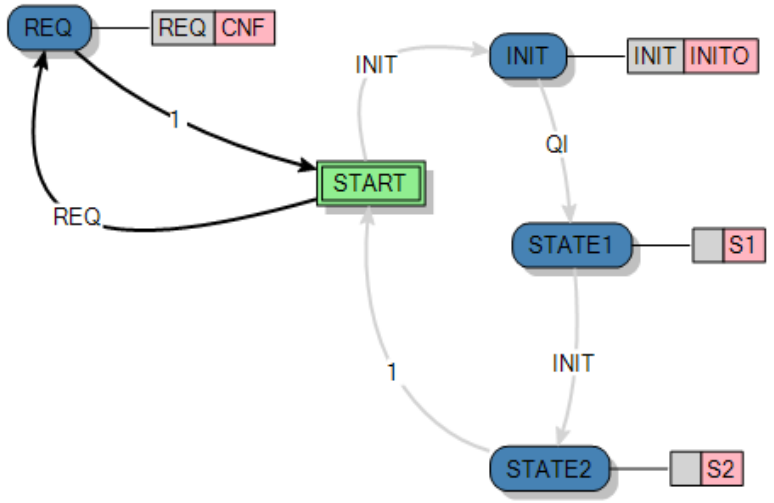


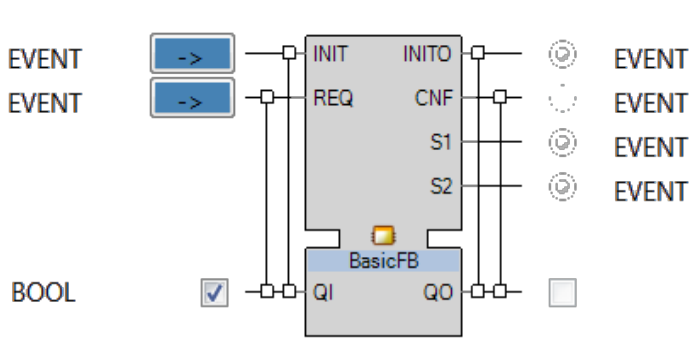
Each state of ECC can have one or more actions.
An action may have an algorithm call and an output event emission, or both.

Lifetime of Event Input

EAE stops ECC execution when:

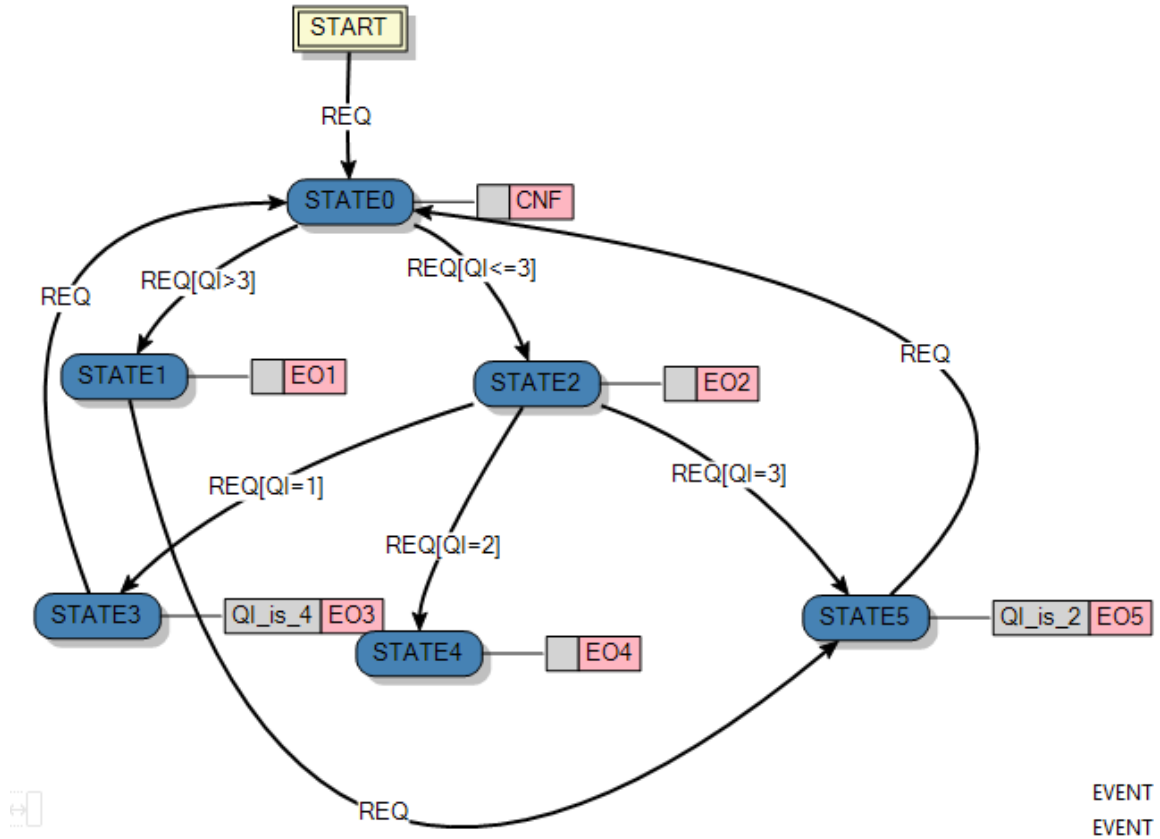
- There are no enabled transition conditions which have not been executed already in this run



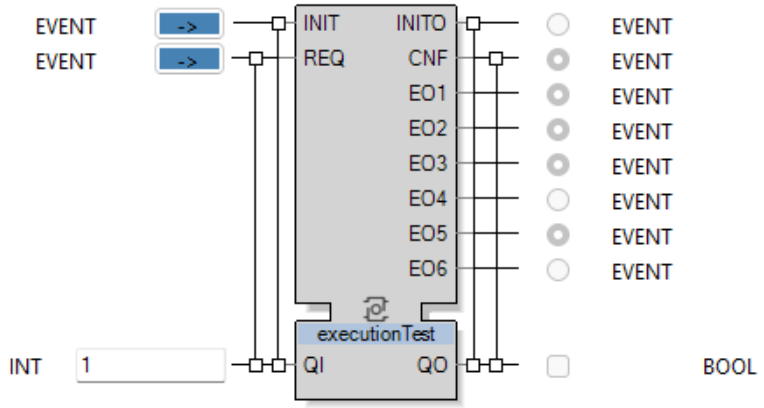


Trick: assign an algorithm, ALG1, to STATE1, within which set INIT:=False. This workaround is not portable.

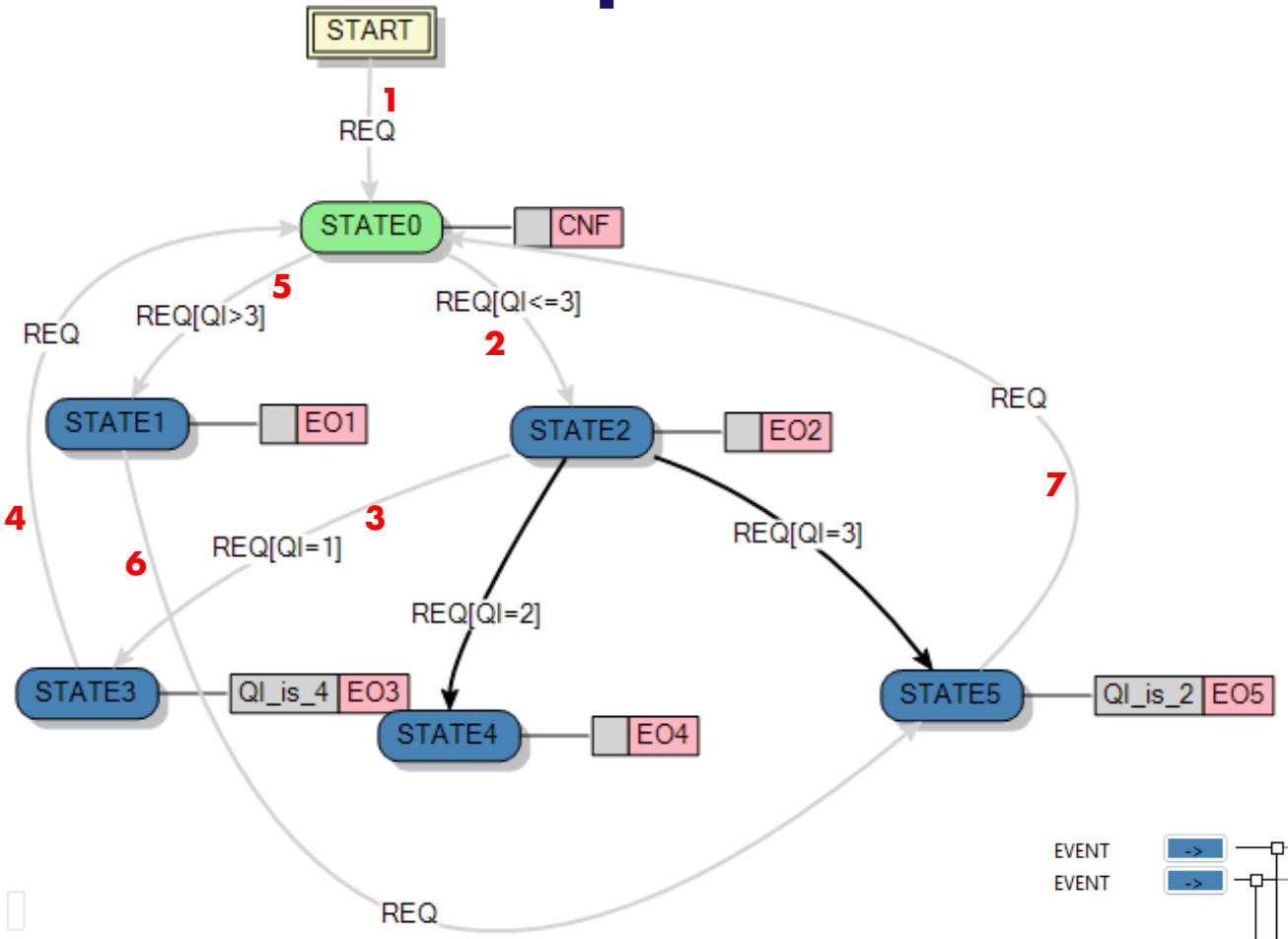
Lifetime of Event Input

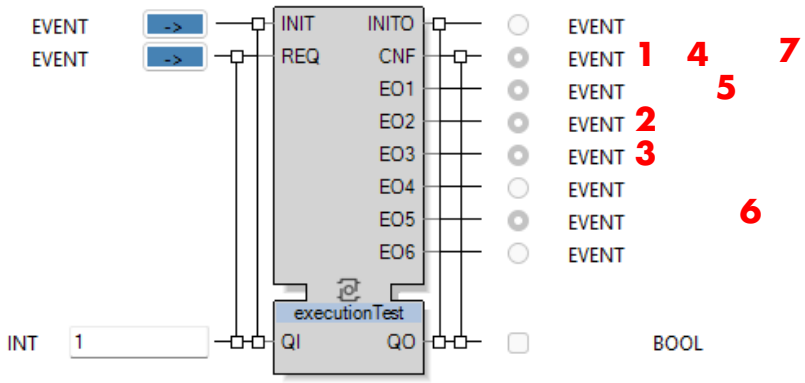


- The first transition with a true condition is executed and the corresponding ECC status of the block is occupied.
- The recently used transition is marked so that it cannot be used a second time.
- Once there are no transitions to other ECC states left to be fired, because they are already marked and thus executed, or none of the remaining, unmarked transition conditions proves to be true, the markings of the transitions is reset.
- FB is waiting for next input event

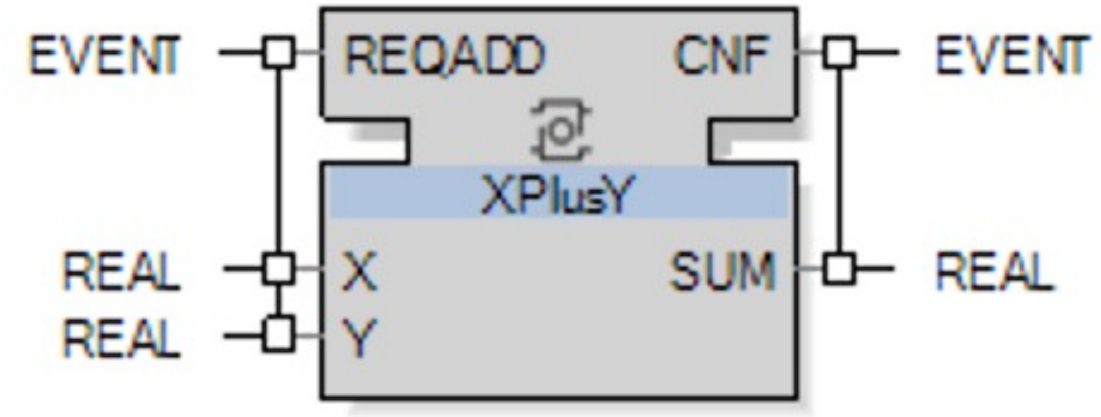


Lifetime of Event Input

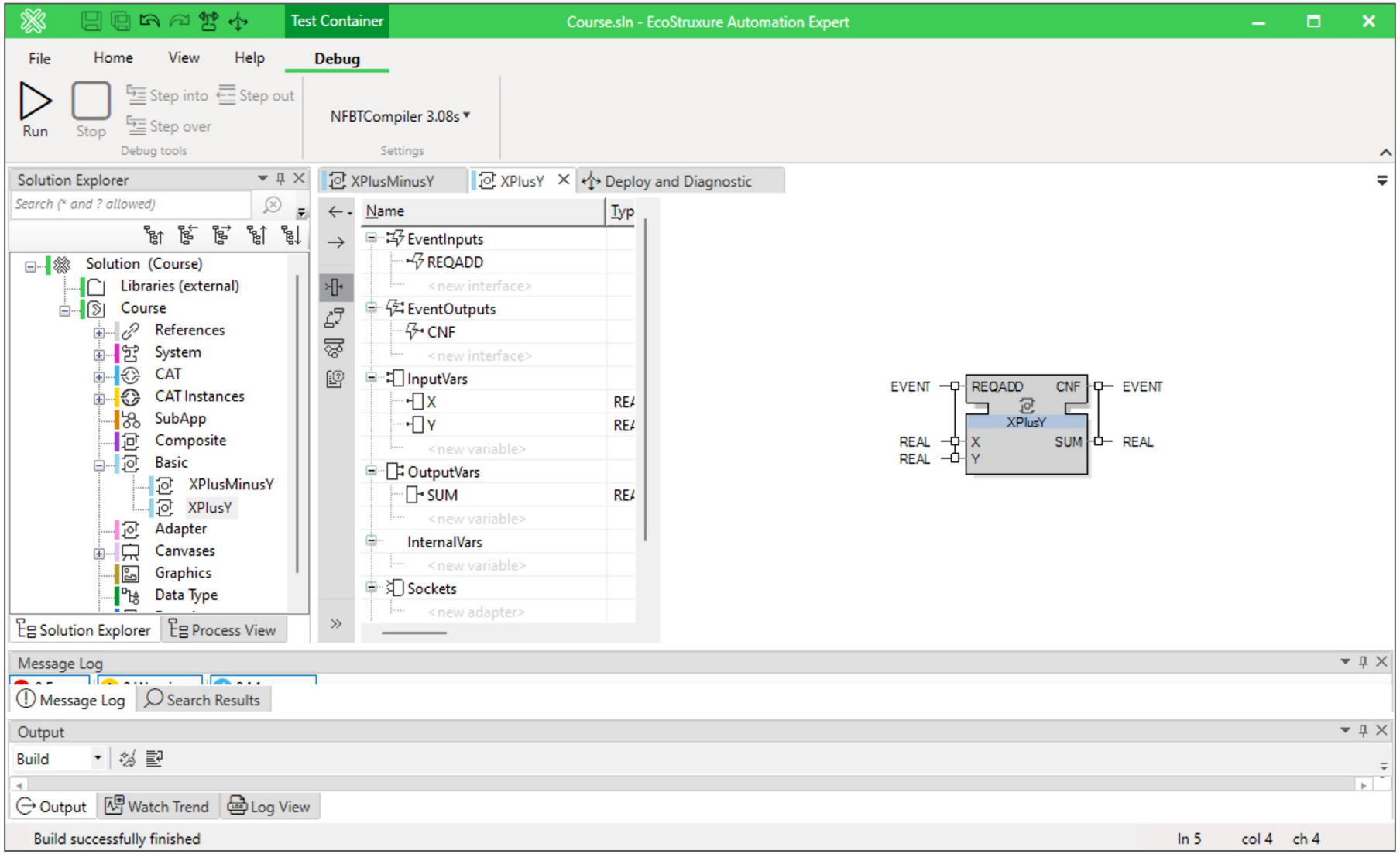




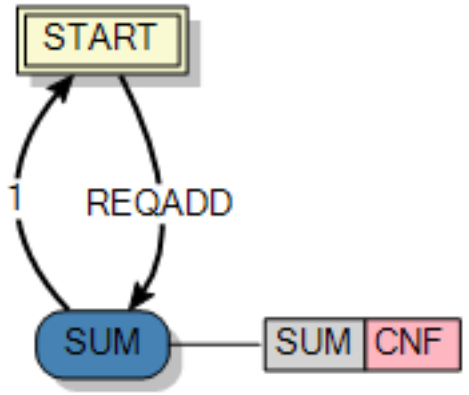
Example: A Basic FB that adds two real numbers



A Basic FB to add Two Real Numbers



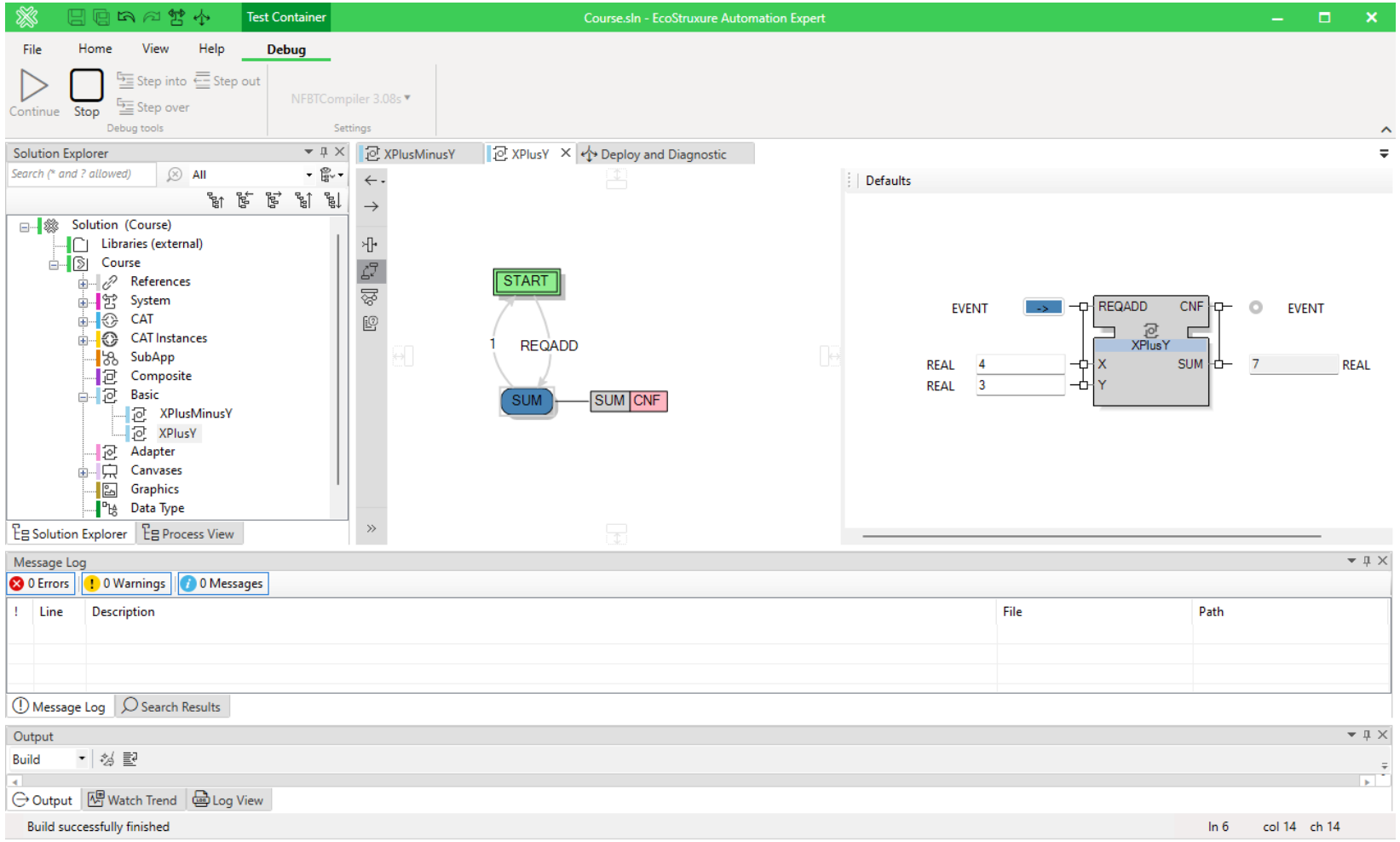
A Basic FB to Add Two Real Numbers



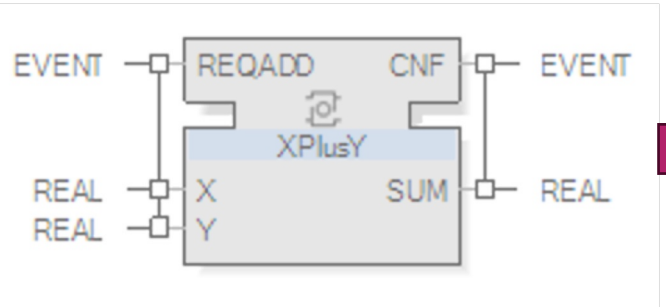
SUM

```
1  ALGORITHM SUM IN ST:
2  (* Add your comment (as per IEC 61131-3) here
3   Normally executed algorithm
4  *)
5   SUM:=X+Y;
6  END_ALGORITHM
```

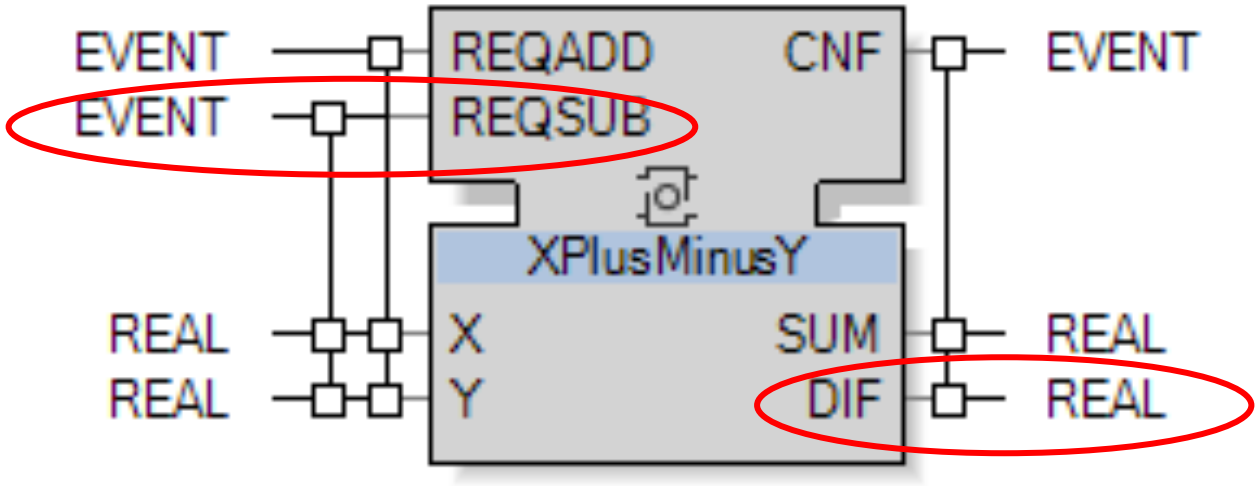
Running the basic FB to Add Two Real Numbers



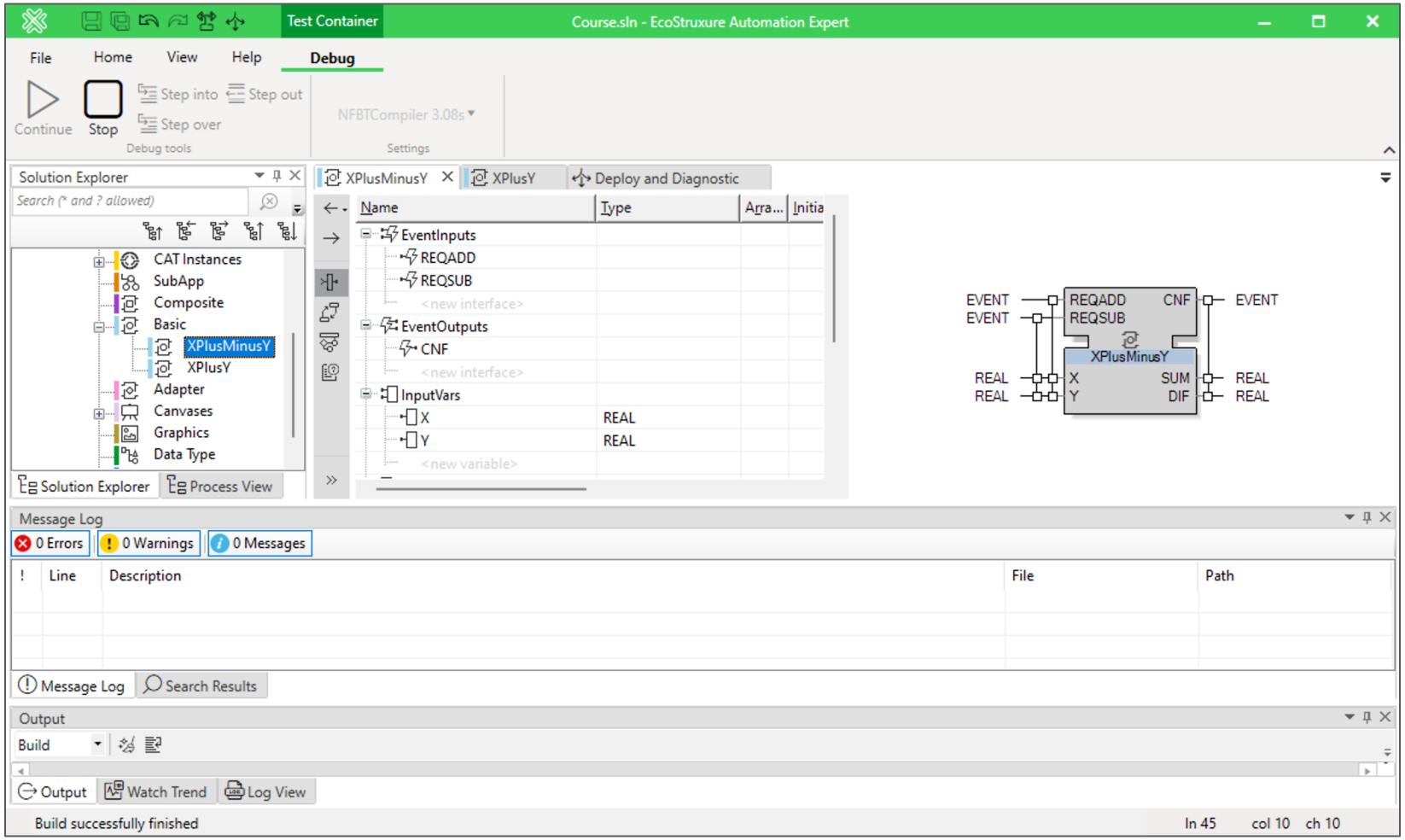
A Basic FB to Add and Subtract Two Real Numbers



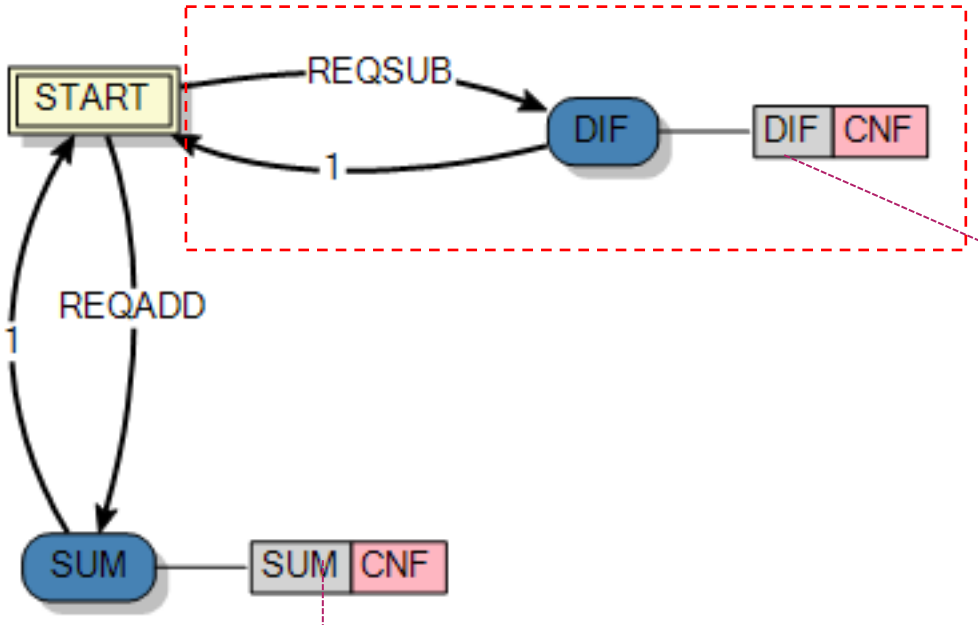
extend



A Basic FB to Add and Subtract Two Real Numbers



A Basic FB to Add and Subtract Two Real Numbers



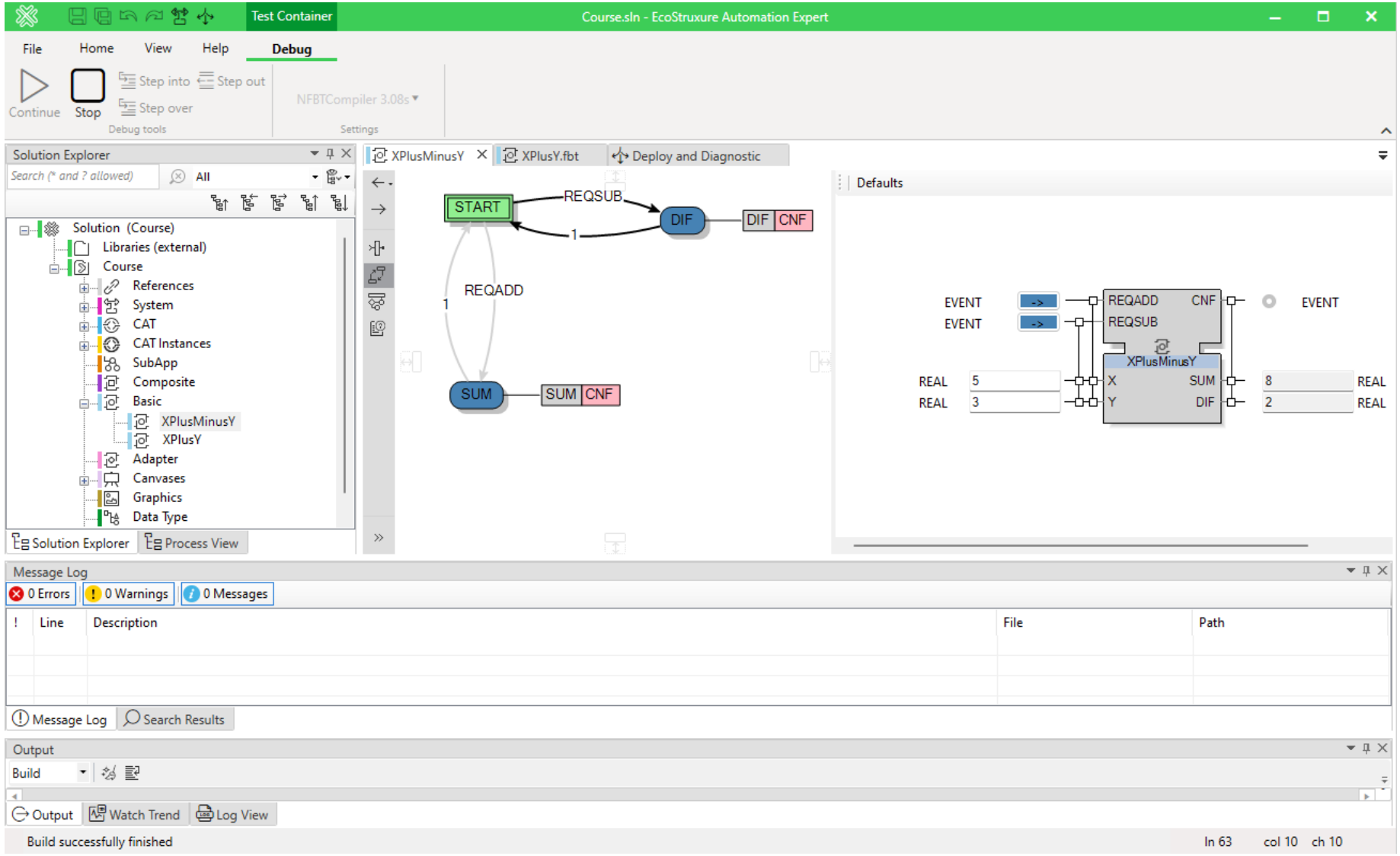
SUM | DIF

```
1  ALGORITHM DIF IN ST:
2  (* Add your comment (as per IEC 61131-3) here
3
4  *)
5  DIF:=X-Y;
6  END_ALGORITHM
```

SUM | DIF

```
1  ALGORITHM SUM IN ST:
2  (* Add your comment (as per IEC 61131-3) here
3  Normally executed algorithm
4  *)
5  SUM:=X+Y;
6  END_ALGORITHM
```

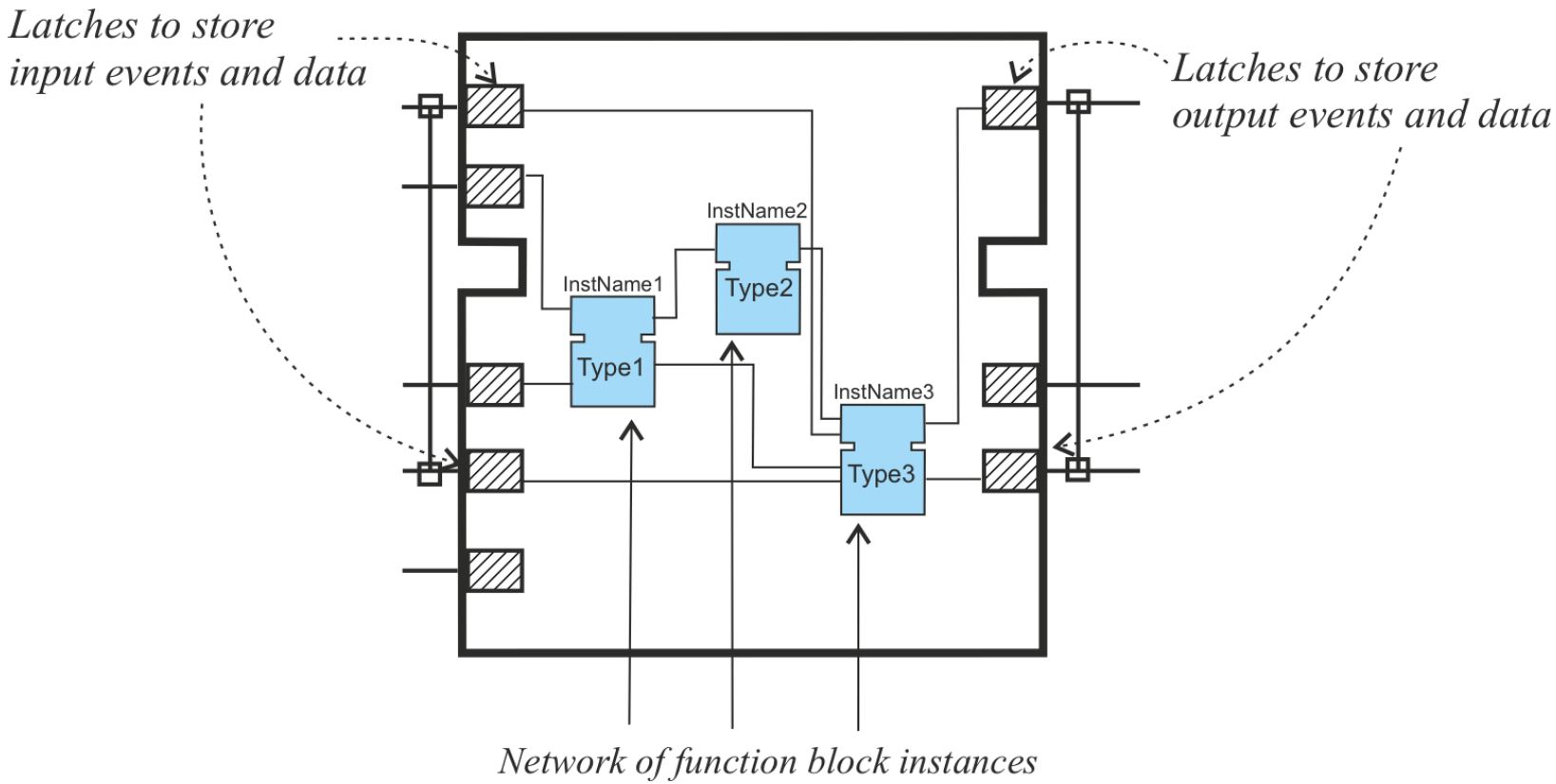
A Basic FB that Adds and Subtract 2 Real Numbers



Composite Function Block

Composite Function Block

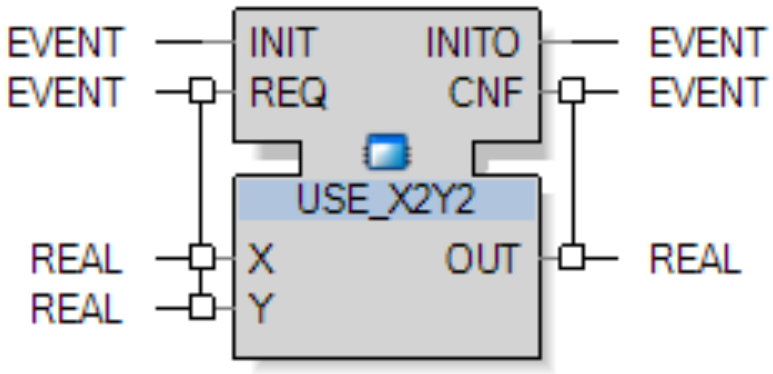
- Network of interconnected FBs
- No internal variables
  - Latches storing the values of input and output events and data
- Nested composite-in-composite



Composite Function Block: Example

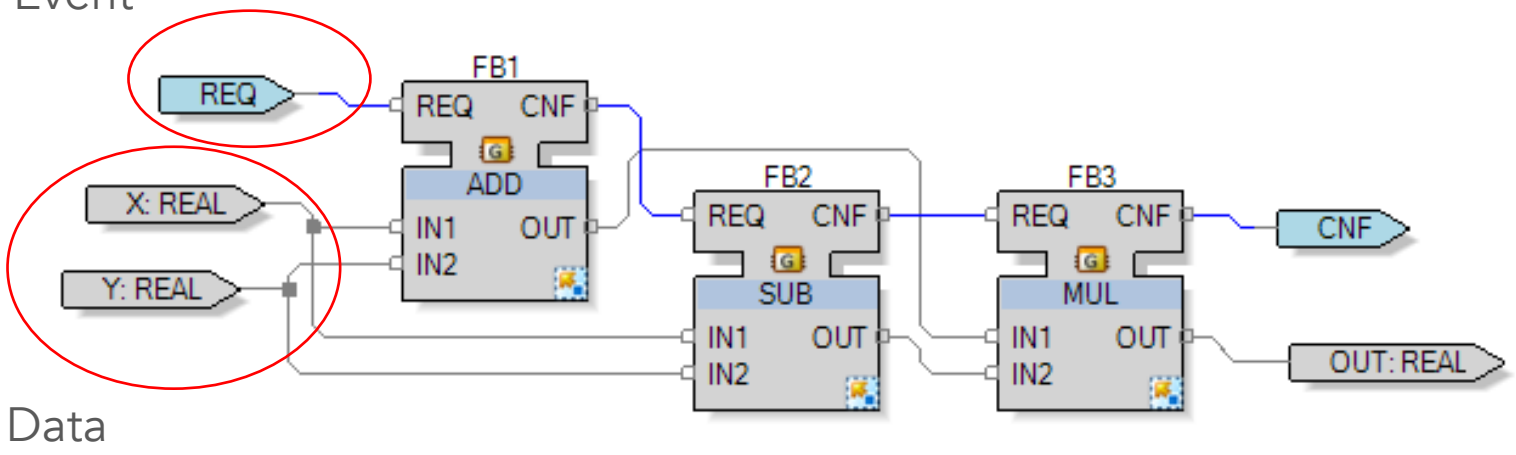
Computing OUT = $X^2 - Y^2$

Interface



Implementation

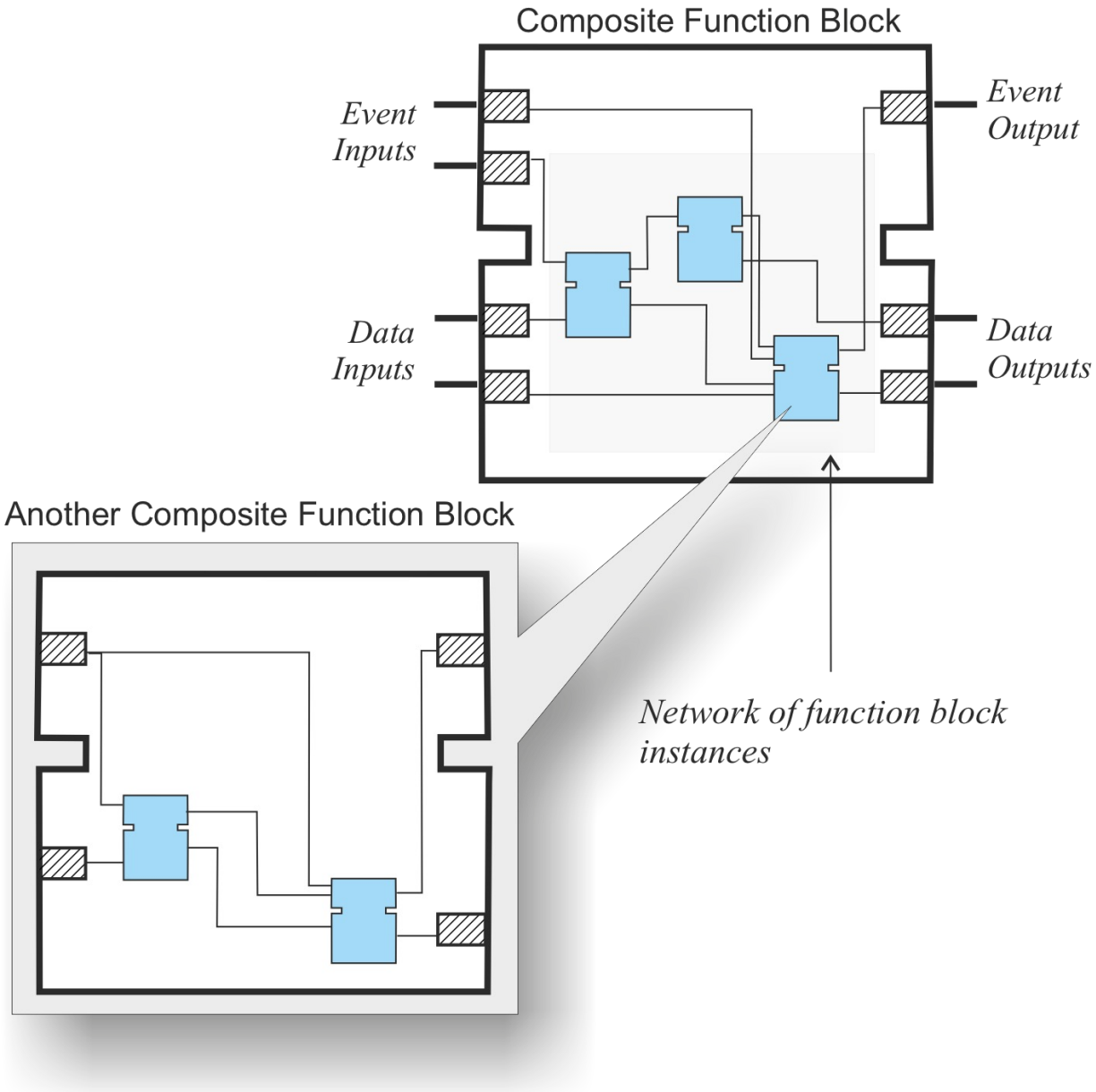
Event



Data

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

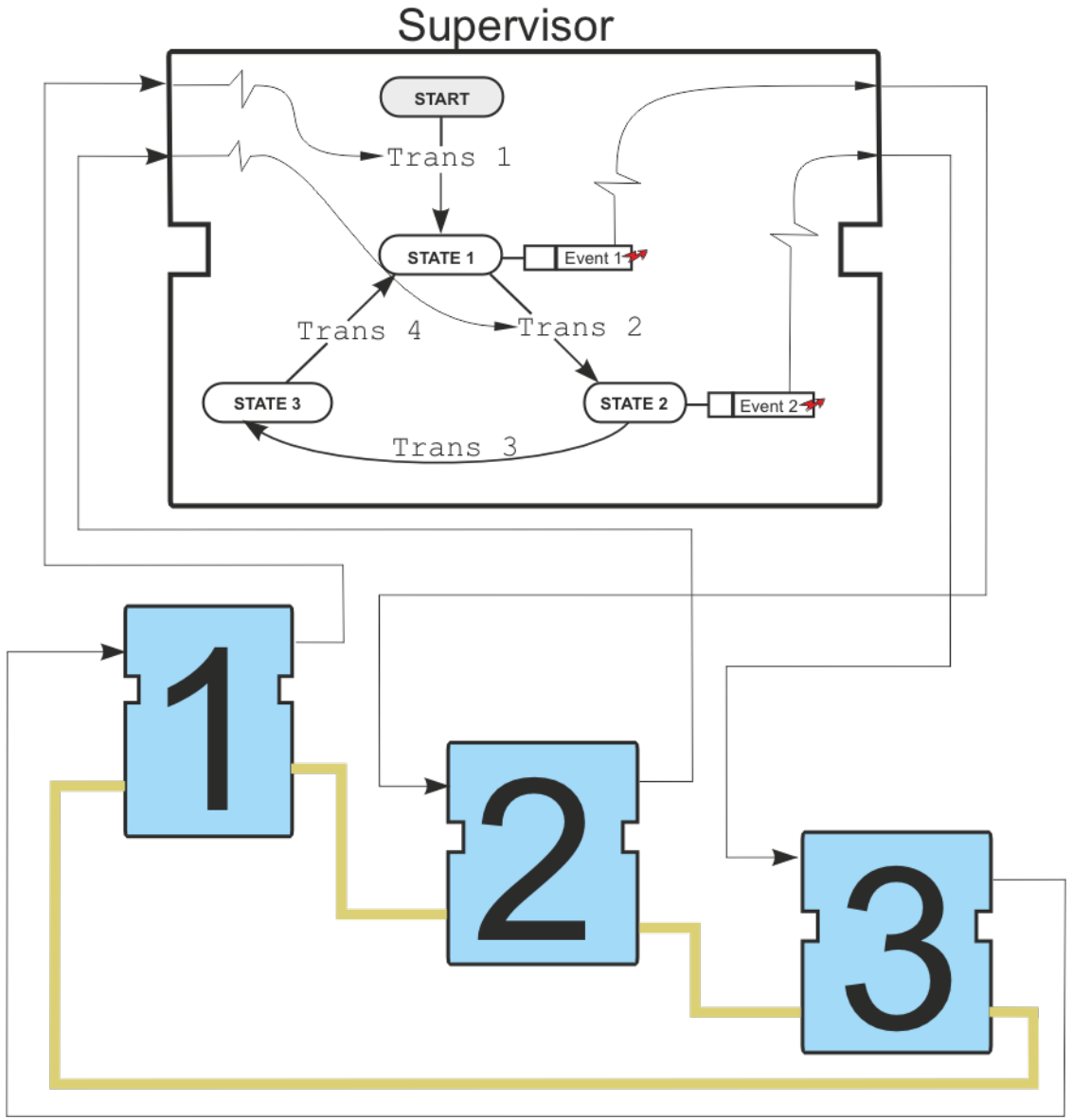
Hierarchical composition



Composite Function Block

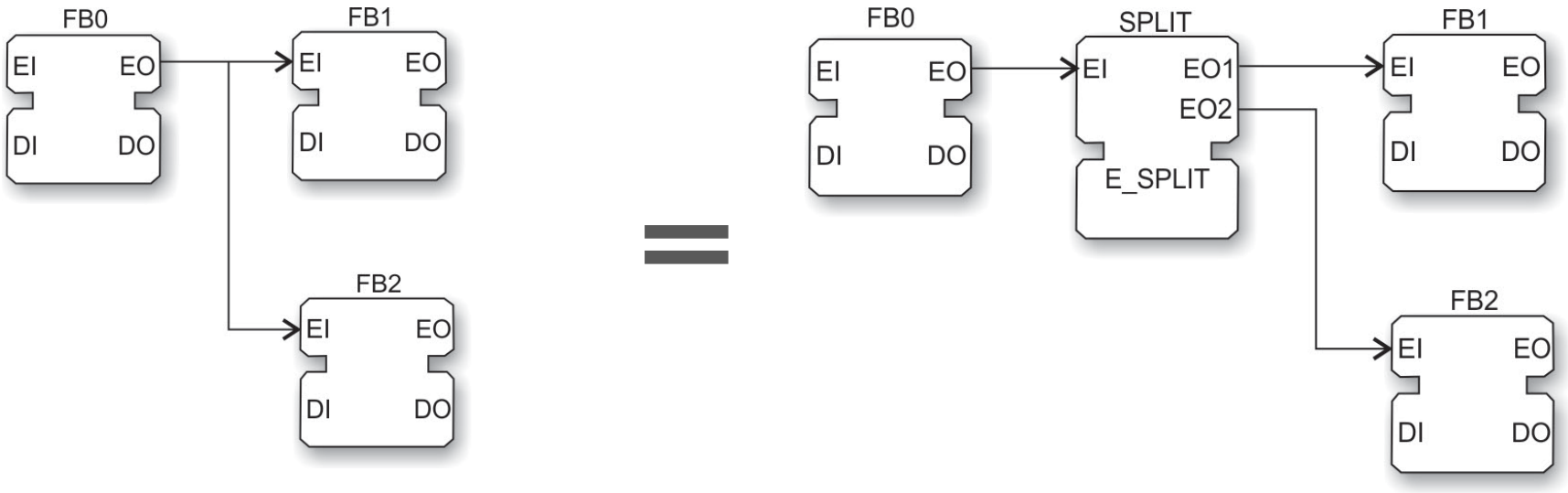
- Execution Control:
  - No ECC
  - No internal variables

- To ensure a particular order of execution, user can implement a supervisor basic function block



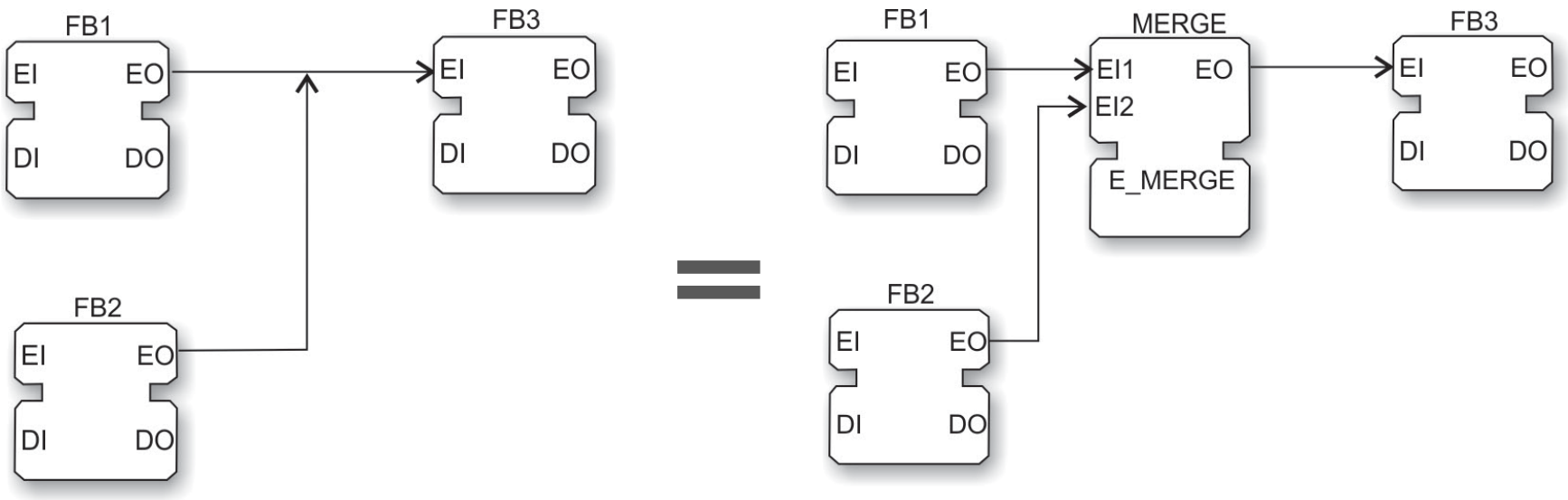
Rules for event connections

Event split is equivalent to using E_SPLIT standard library FB



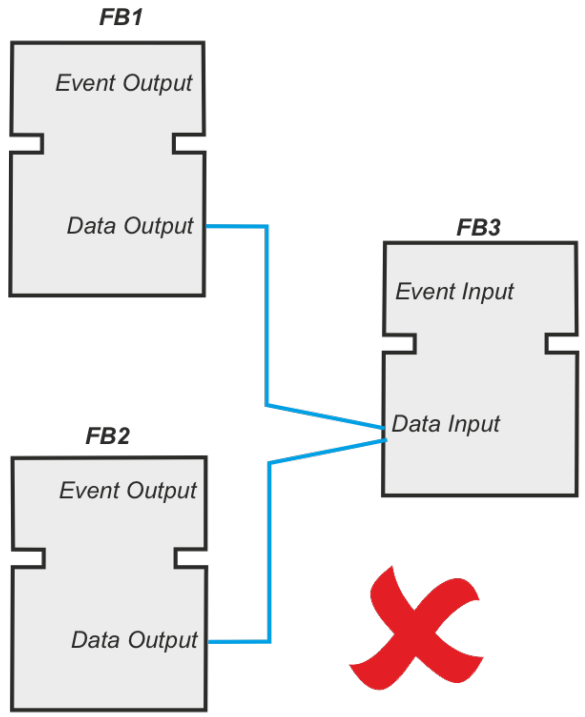
Event merge

Event merge is equivalent to using E_MERGE standard library FB

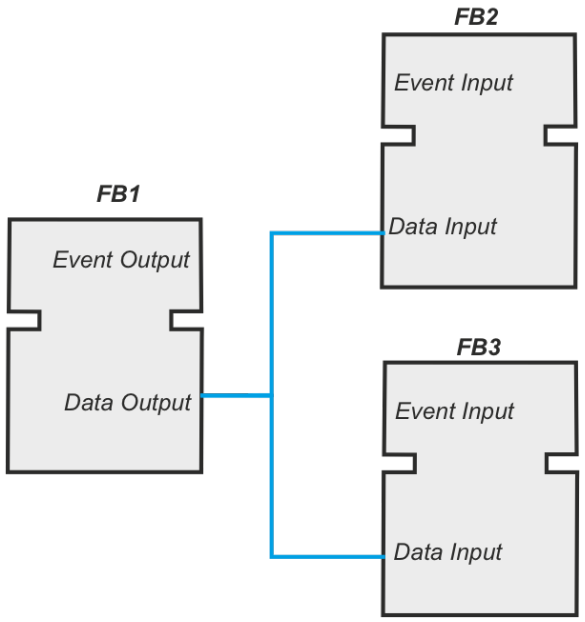


Data connections

Data connections – cannot be merged, but data split is allowed.



Incorrect

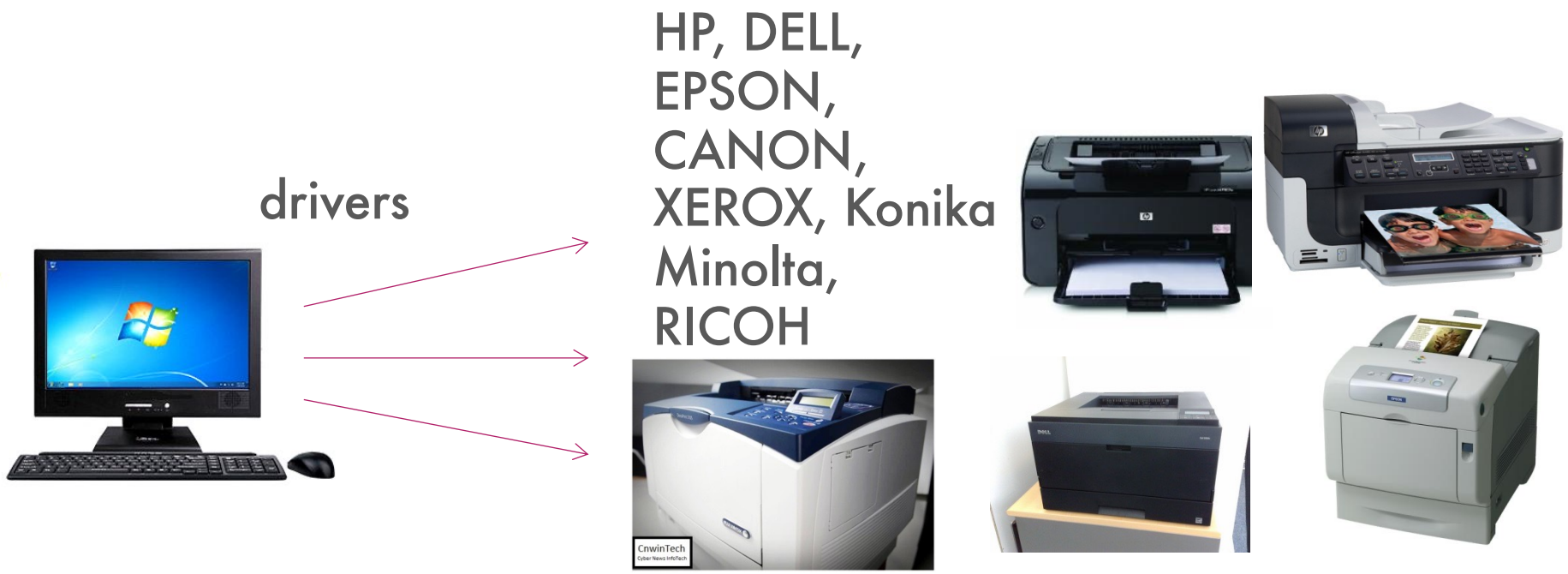


Allowed

Service Interface Function Block

Service interface function blocks (SIFB)

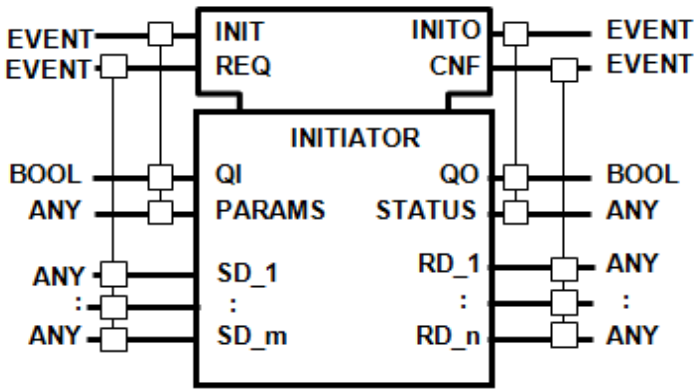
- Mechanisms for interacting with hardware resources
- PC and different vendor printers – need drivers to command printer to print and get status information
- Similar with **SIFB** – to get data from sensors and PLCs, and control actuators
- SIFB implementation requires low level knowledge of particular hardware
- Provided by vendor
- Encapsulation of IP



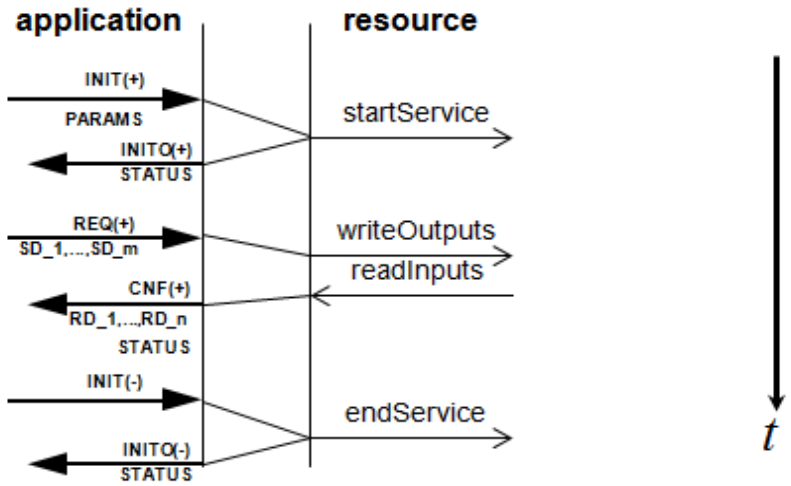
55

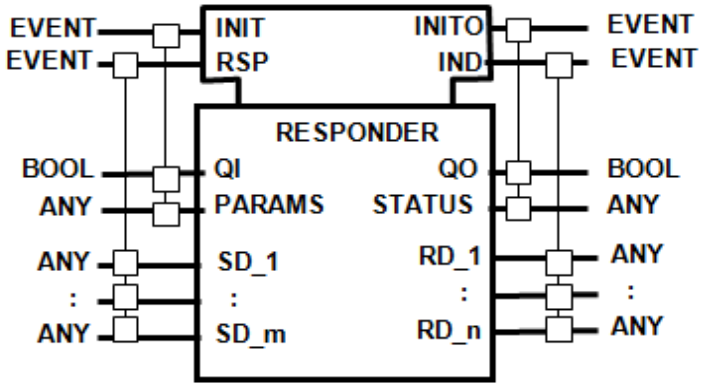
Service Interface Function Blocks (SIFB)

- Modelled as sequences of service primitives per ISO TR 8509

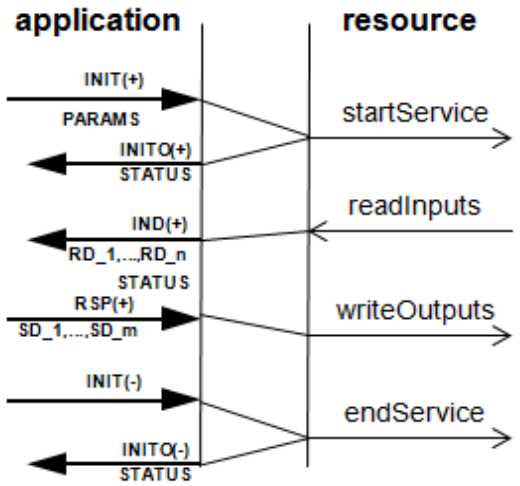


(application-initiated transactions)





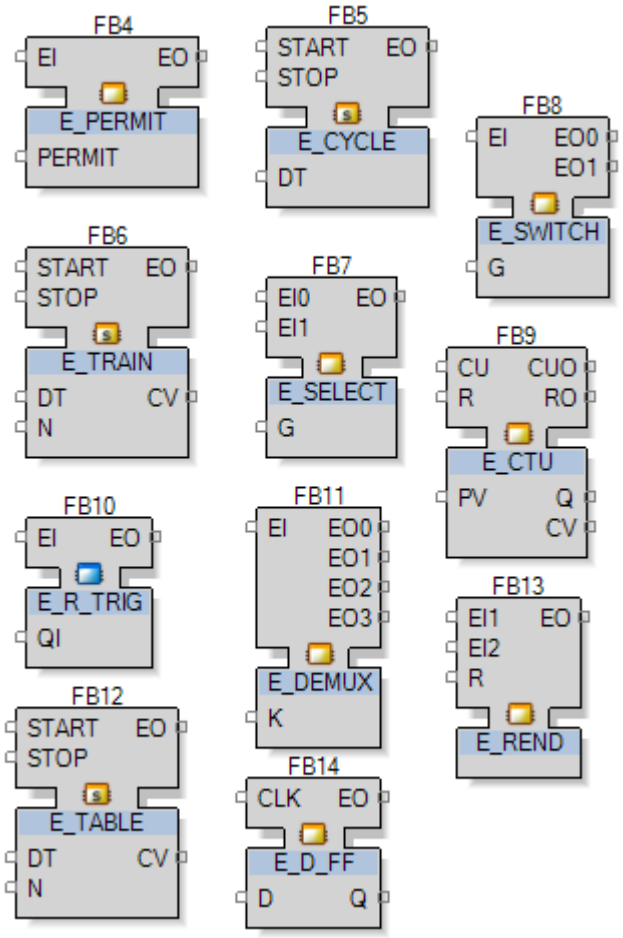
(resource-initiated transactions)



56

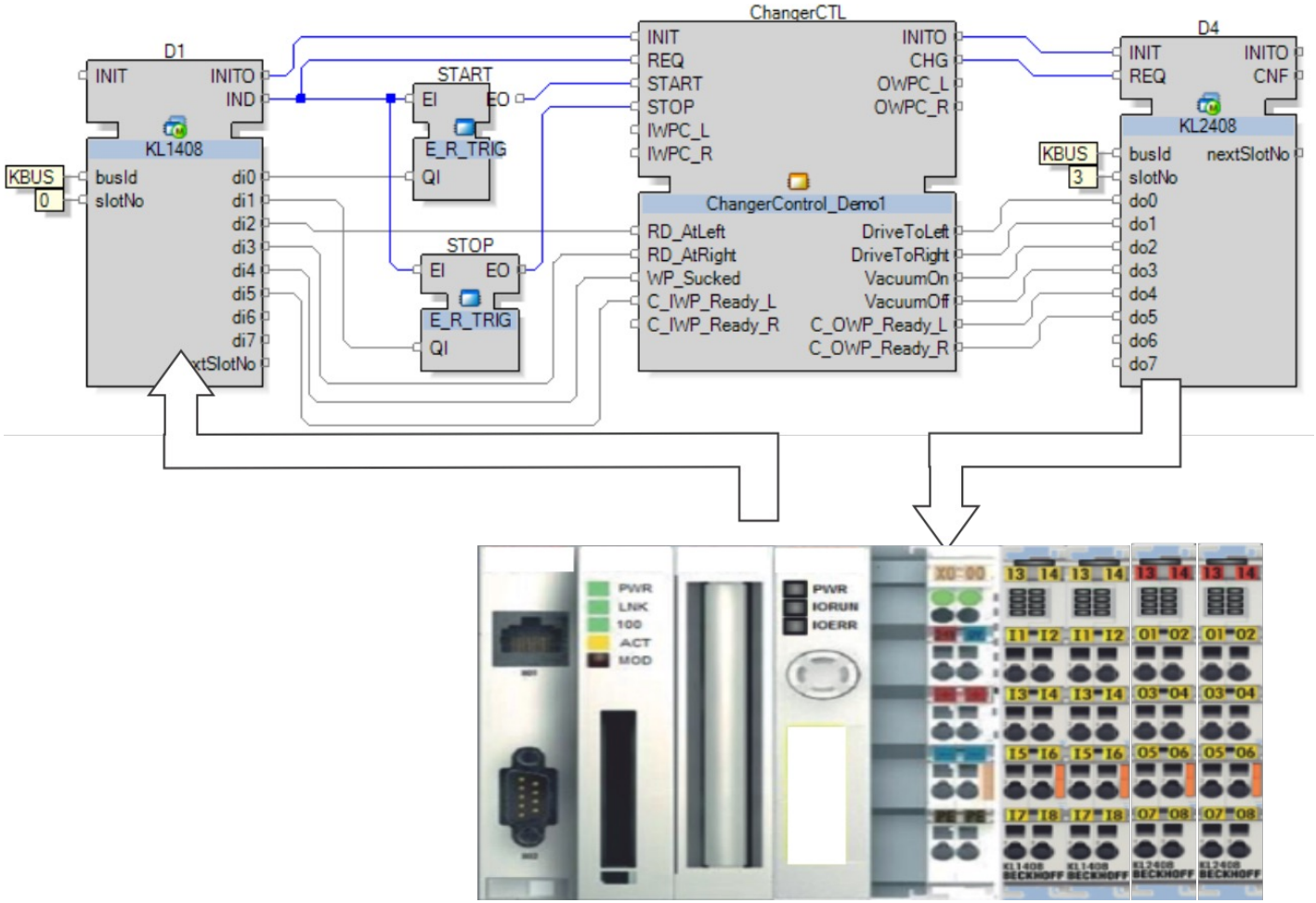
Operations with events

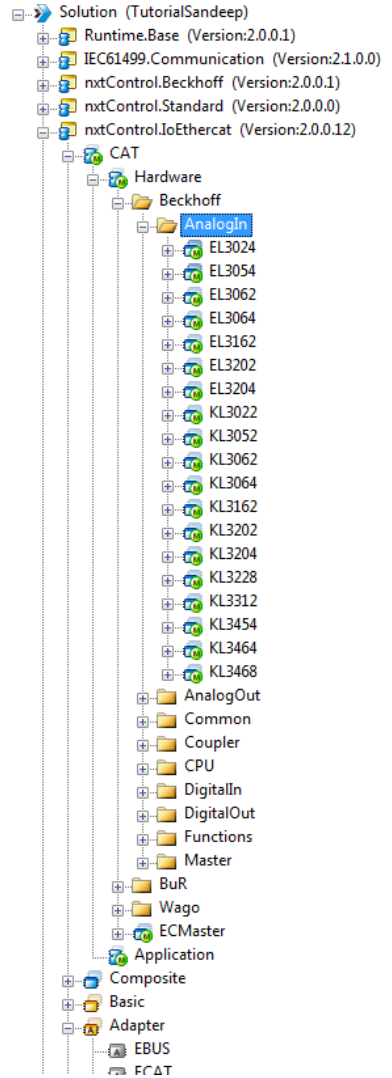
- **E_SPLIT/E_MERGE/E_REND**—Event split, merge, rendezvous;
- **E_PERMIT**—Permissive event propagation;
- **E_SELECT**—1 of 2 (Boolean) event selection;
- **E_SWITCH**—1 of 2 (Boolean) event demultiplexing;
- **E_DELAY**—Event delay (timer);
- **E_CYCLE**—Periodic event generation;
- **E_RESTART**—Generation of **COLD/WARM** restart, **STOP** events;
- **E_TRAIN/E_TABLE/E_N_TABLE**—Finite trains of events;
- **E_SR/E_RS/E_D_FF**—Event-driven bi-stables;
- **E_R_TRIG/E_F_TRIG**—Event-driven rising/falling edge detection;
- **E_SR/E_RS/E_D_FF**—Event-driven bi-stables;
- **E_CTU**—Event-driven up-counter.



57

Process interface: read inputs and write outputs



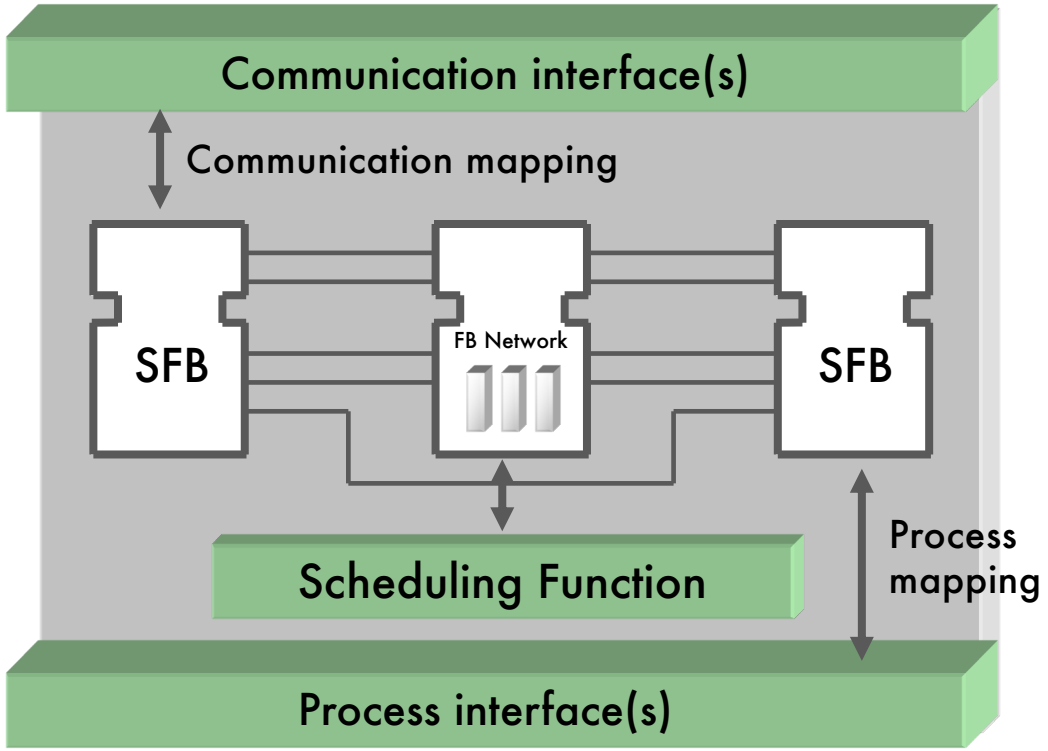


58

Elements of Distributed systems

Resource Model

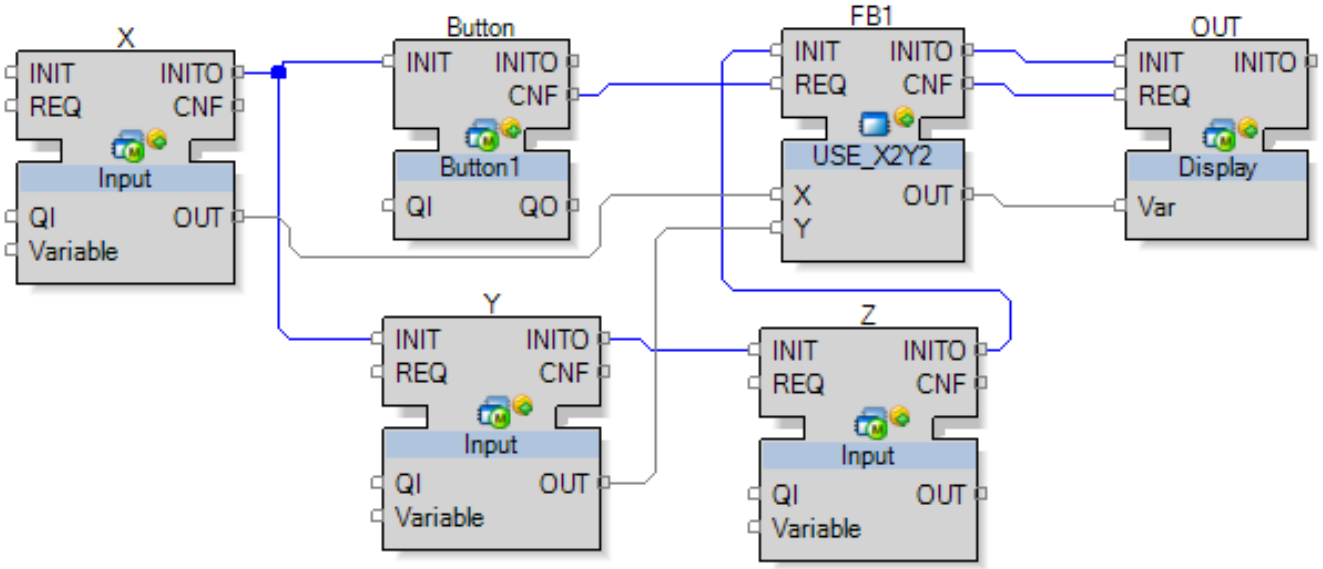
- The main execution container of FB network
- Each Resource is independent of other Resources in the device
- Access to Communication and Process interfaces via SIFBs
- Responsible for the scheduling of FBs

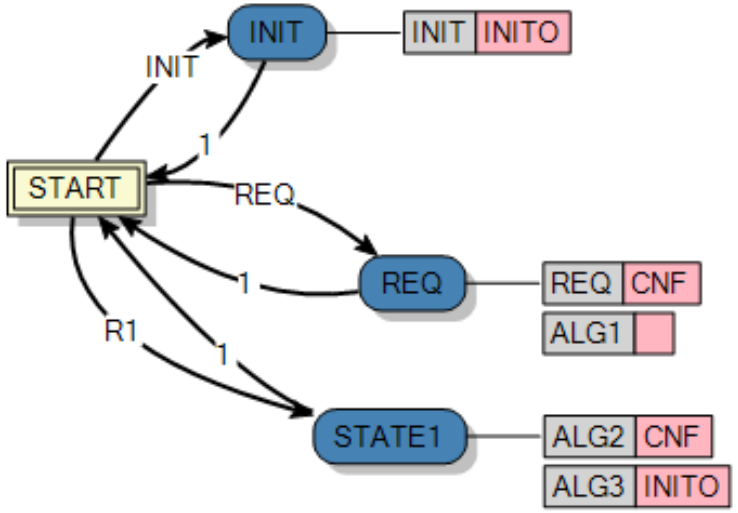


60

Resources – scheduling functions

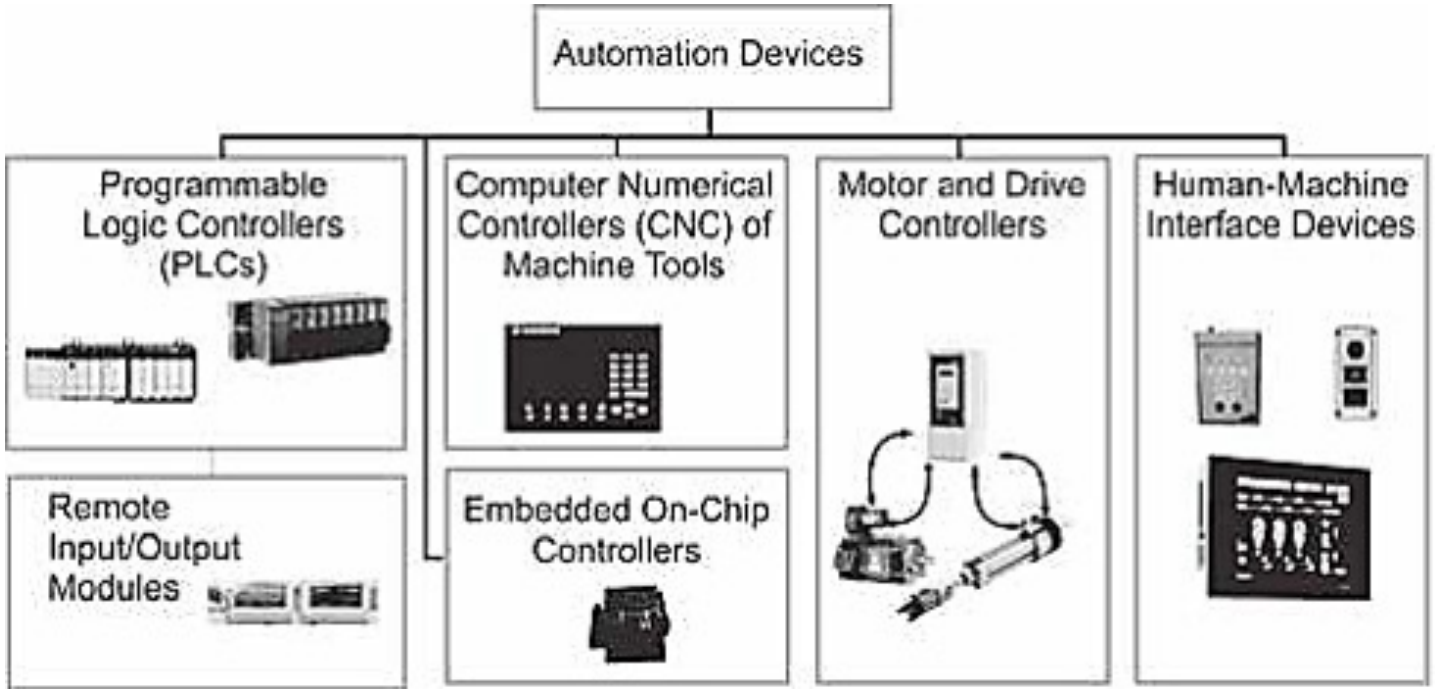
- Schedule algorithms for execution
- Determines sequence
  - Of FBs execution
  - Of algorithms execution





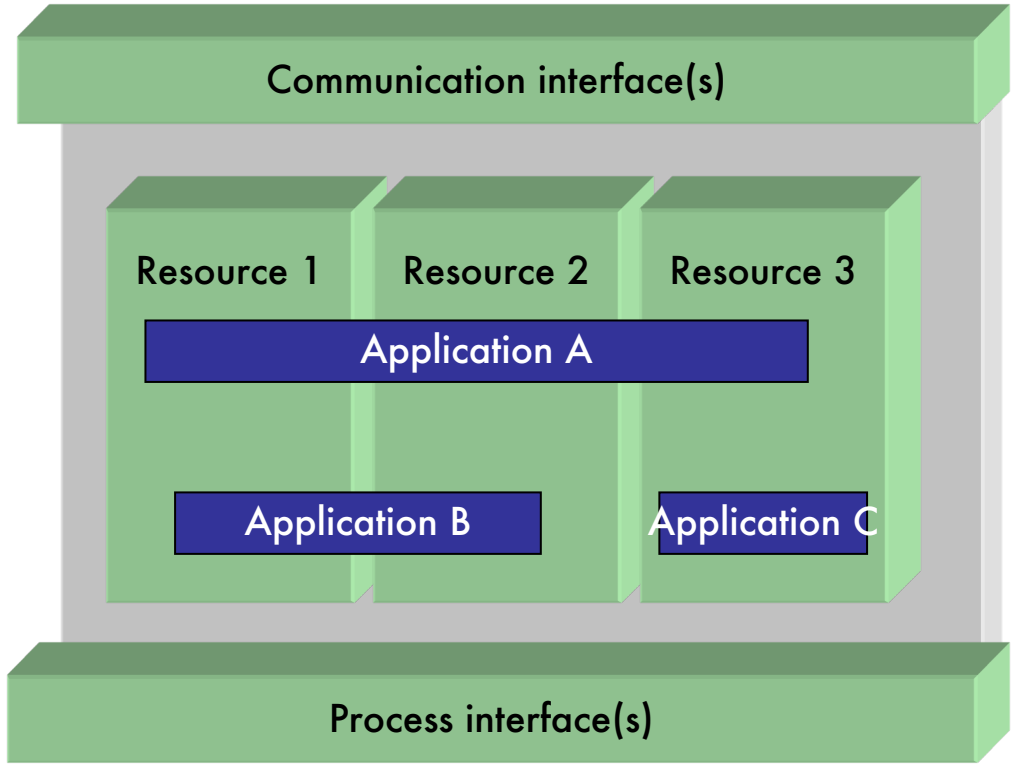
Device Model

- An abstract model represents a physical instrument interacting with automation systems or process information, e.g.
  - PLC, CNC, microcontroller, etc.



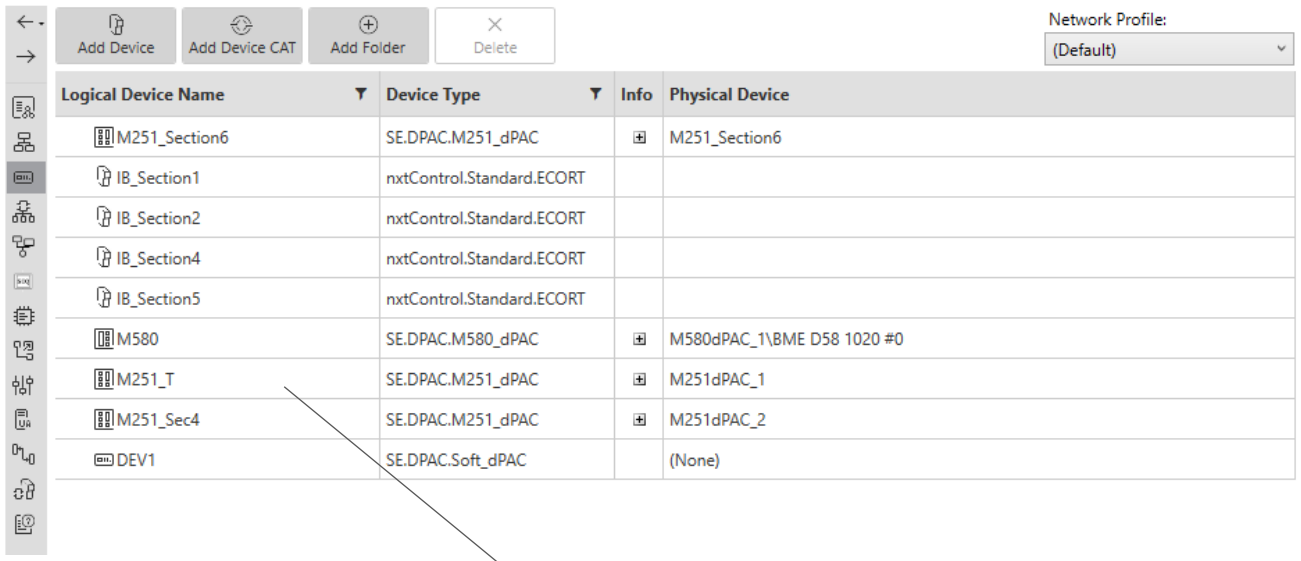
Device Model

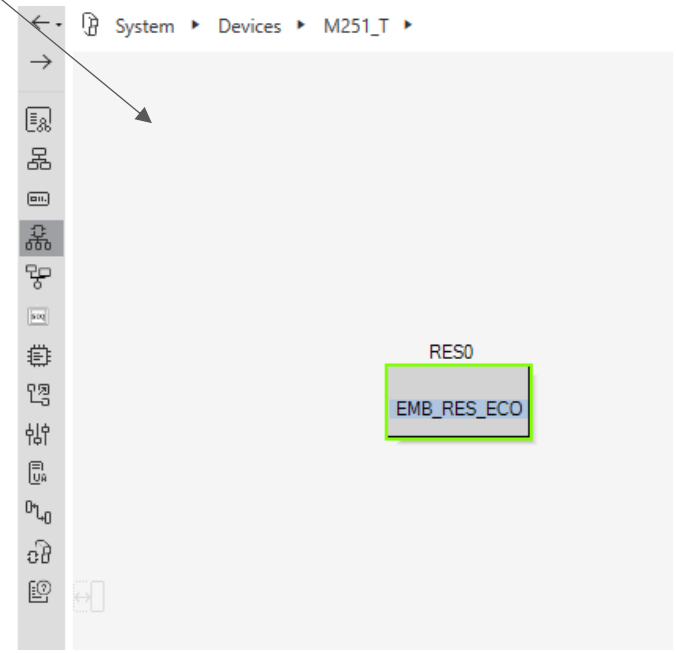
- A Device is specified by it process interfaces and communication interfaces
  - Process interface is the mapping between the physical entities (sensors, actuators) and the Resource
- Contains multiple Resources
- All Resources use the same Communication and Process interfaces of the Device

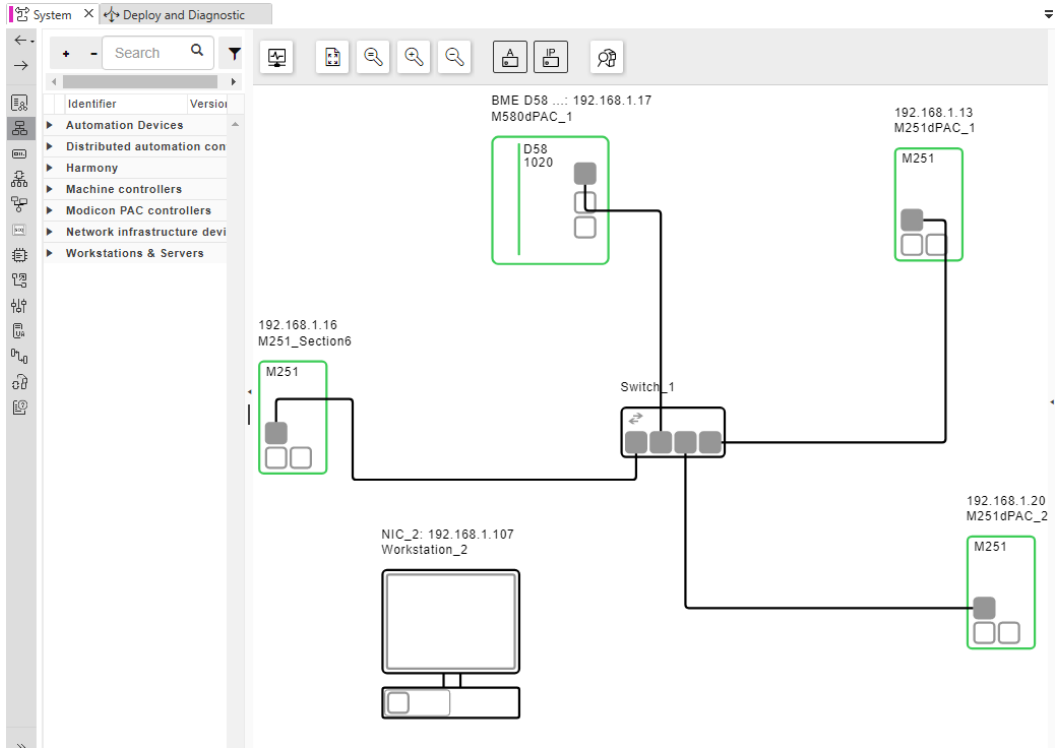


63

Device model: example of use







64

Device classes

Class 0:

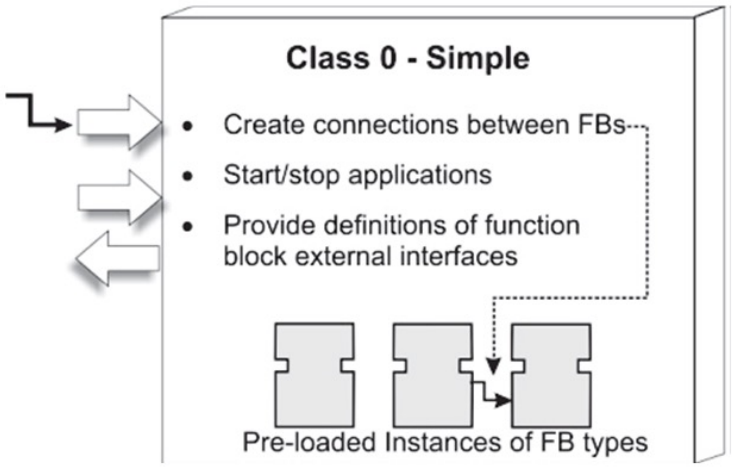
- Preprogrammed FB instances
- FBs connections can be changed

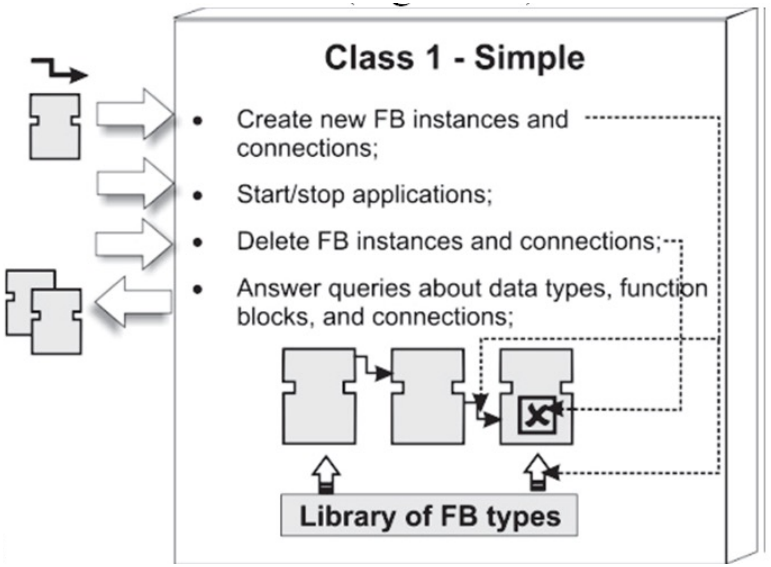
Class 1

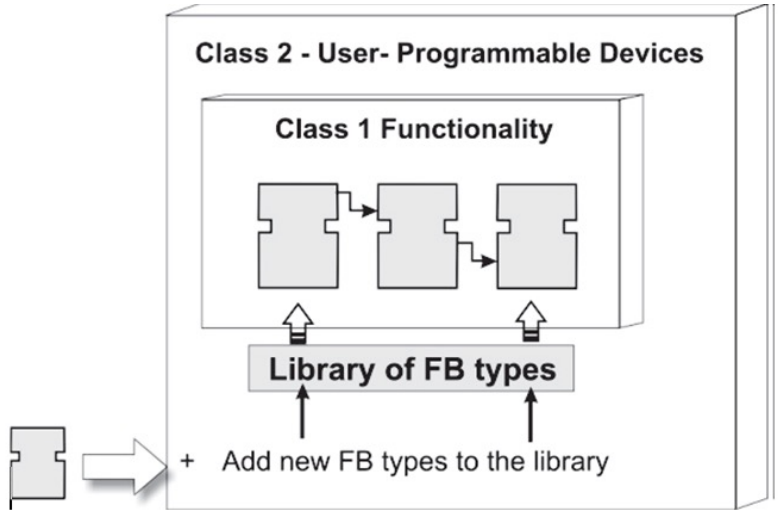
- New instance of FB library

Class 2

- Fully reprogrammable
- Can create FB types





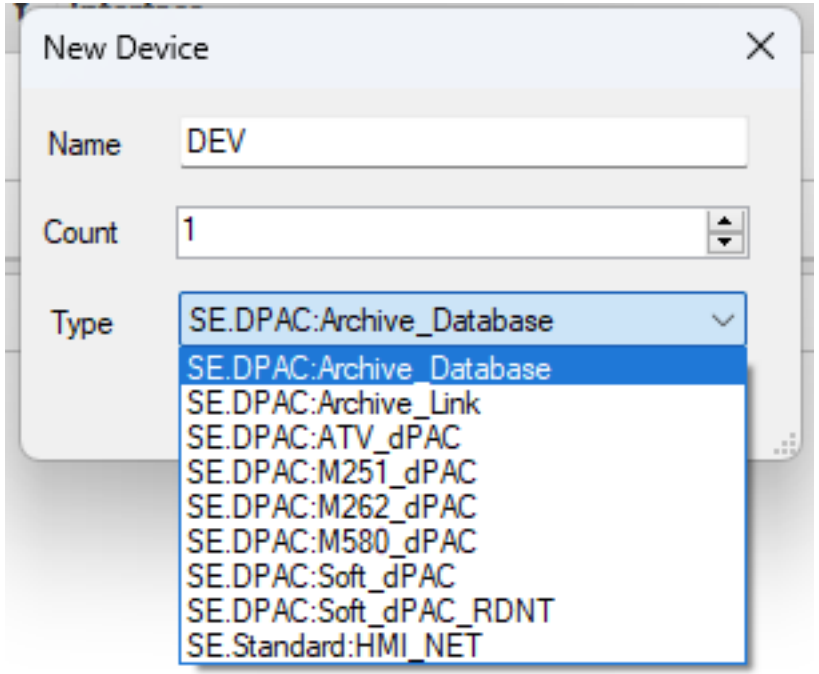


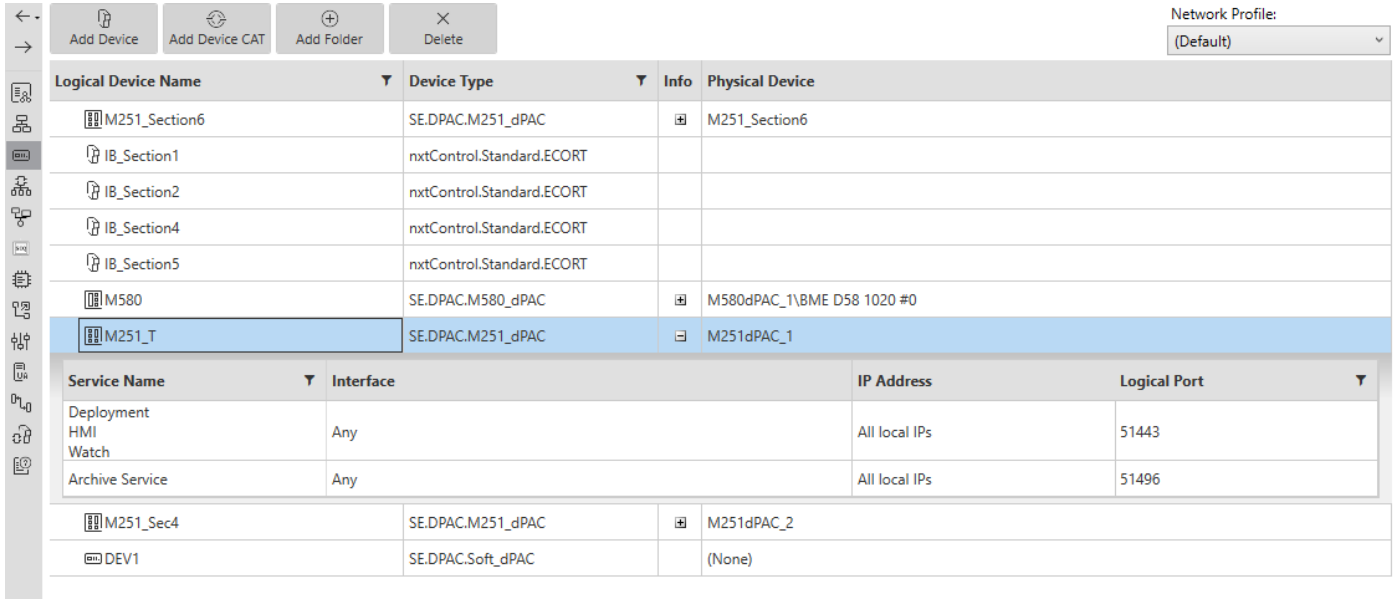
Device type definition

Interface of the device may have only data inputs.

Example: SE.DPAC.M251_dPAC

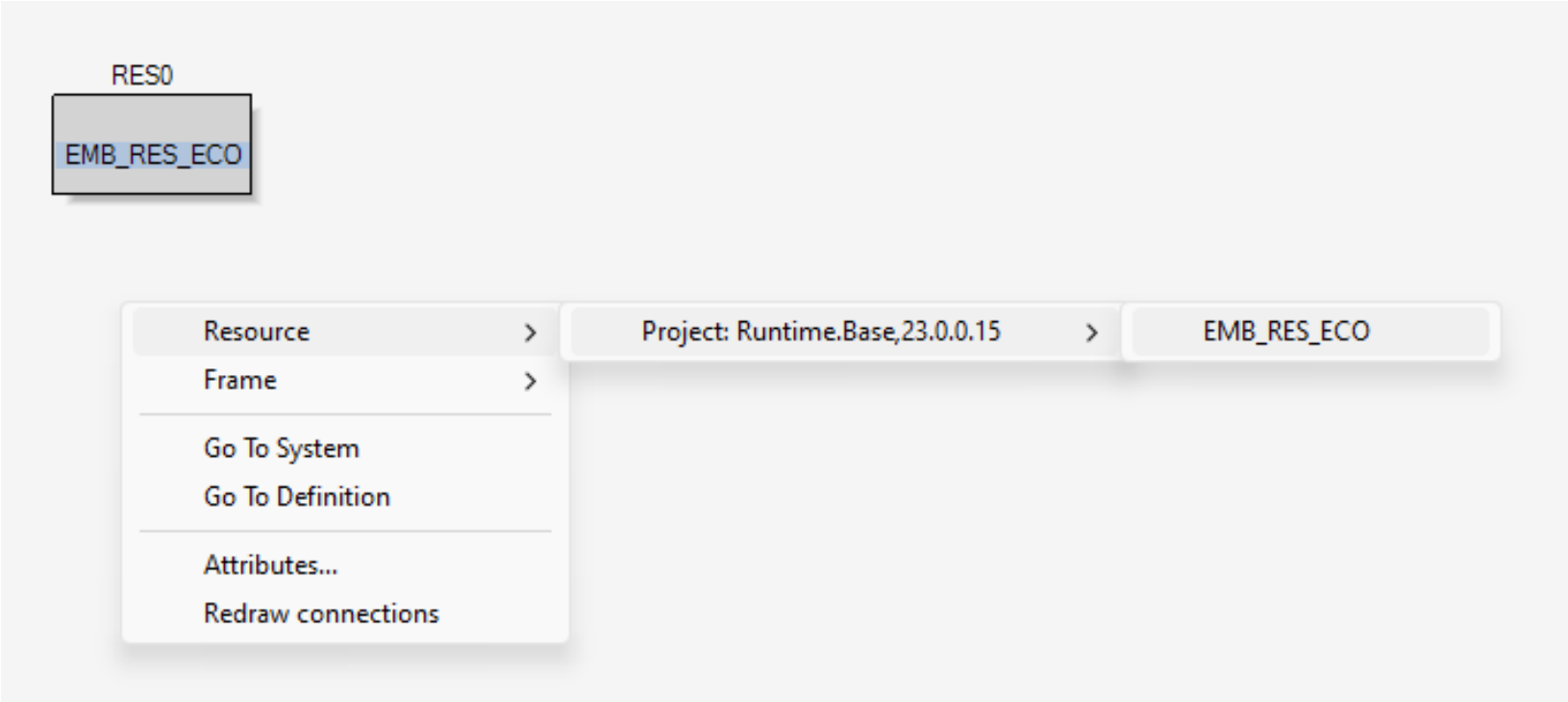
- Deployment IP Address:Port
- HMI IP Address:Port
- Watch IP Address:Port
- Archive Service IP Address:Port





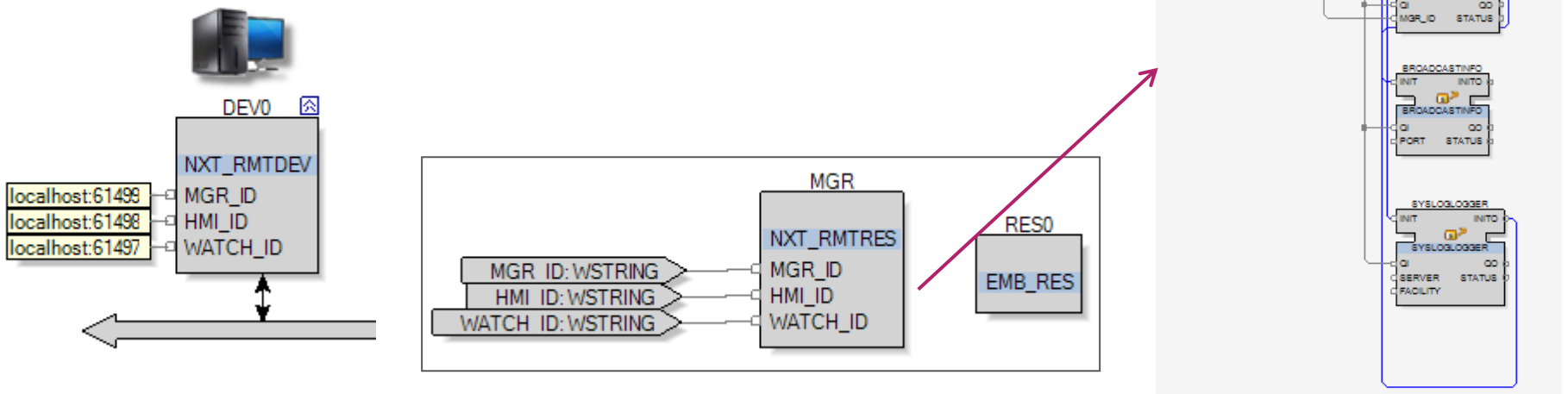
Resource type definition

- EMB_RES_ECO, PANEL_RESOURCE, VIEW_RES
  - Default function block START of E_RESTART



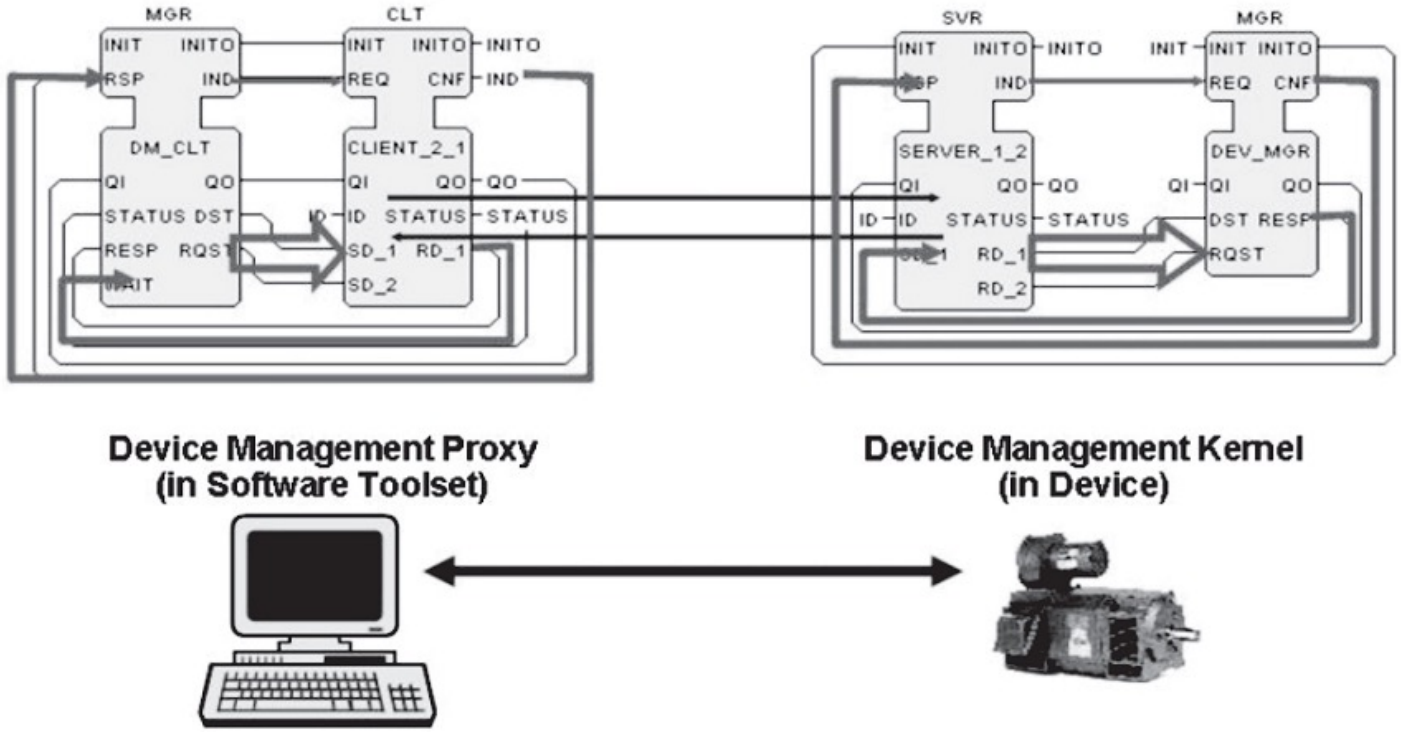
Device management

- By default, devices contain resource MGR
- KERNEL of TYPE DM_KRNL
- Executes received management commands
- Configuration tool manages the device



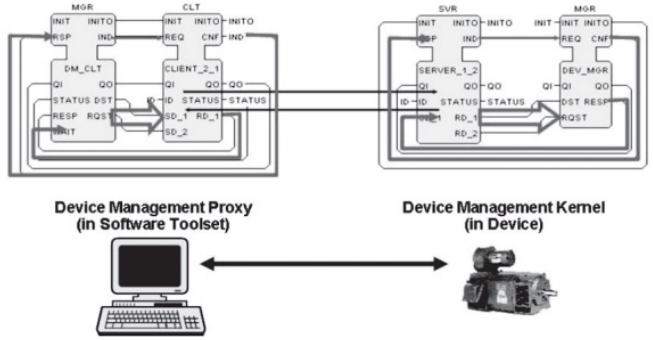
Device management

- Dev_MGR receives commands in XML
- Replies back with confirmation or errors

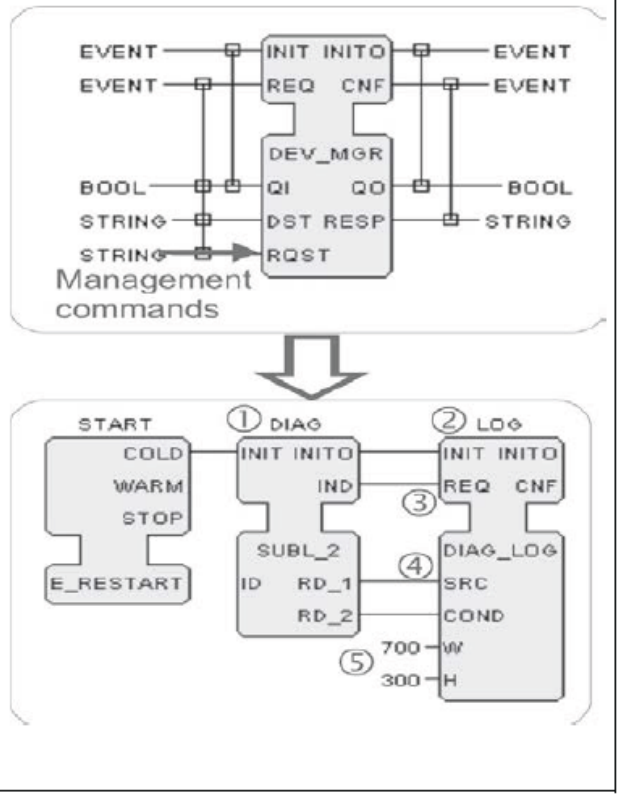


Device management

- Dev_MGR receives commands in XML
- Replies back with confirmation or errors

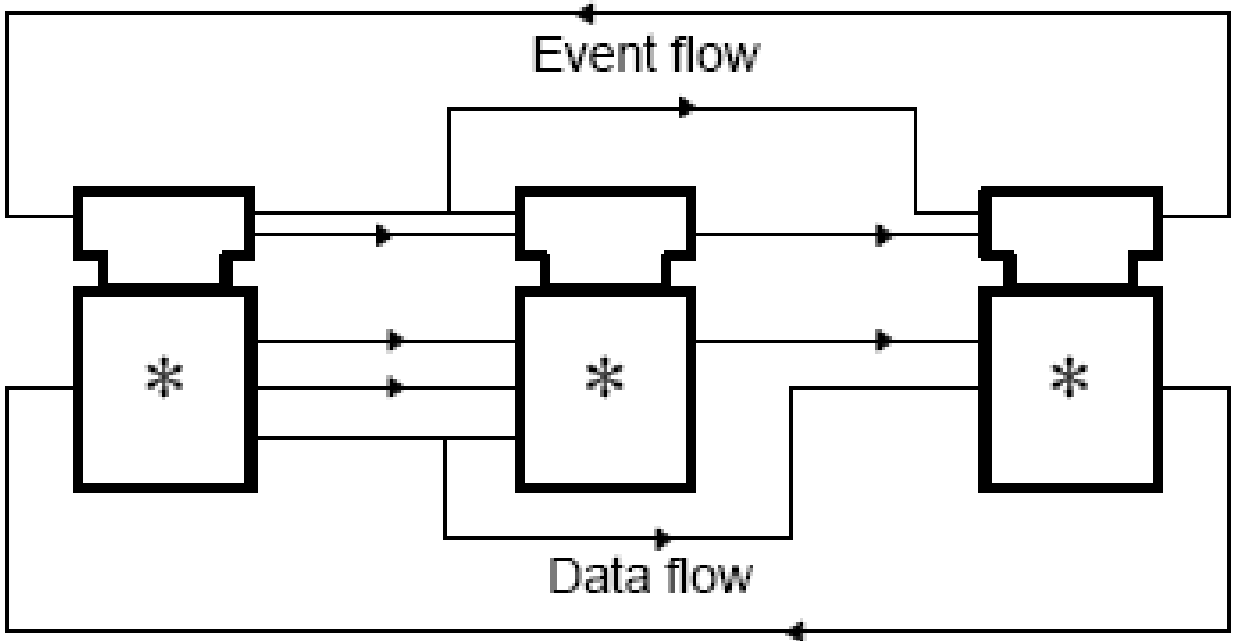


#	Management commands
1	<Request ID="3" Action="CREATE" > <FB Name="DIAG" Type="SUBL_2" /> </Request>
2	<Request ID="4" Action="CREATE" > <FB Name="LOG" Type="DIAG_LOG" /> </Request>
3	<Request ID="7" Action="CREATE"> <Connection Source="DIAG.IND" Destination="LOG.REQ" /> </Request>
4	<Request ID="8" Action="CREATE" > <Connection Source="DIAG.RD_1" Destination="LOG.SRC" /> </Request>
5	<Request ID="10" Action="WRITE" > <Connection Source="700" Destination="LOG.W" /> </Request>

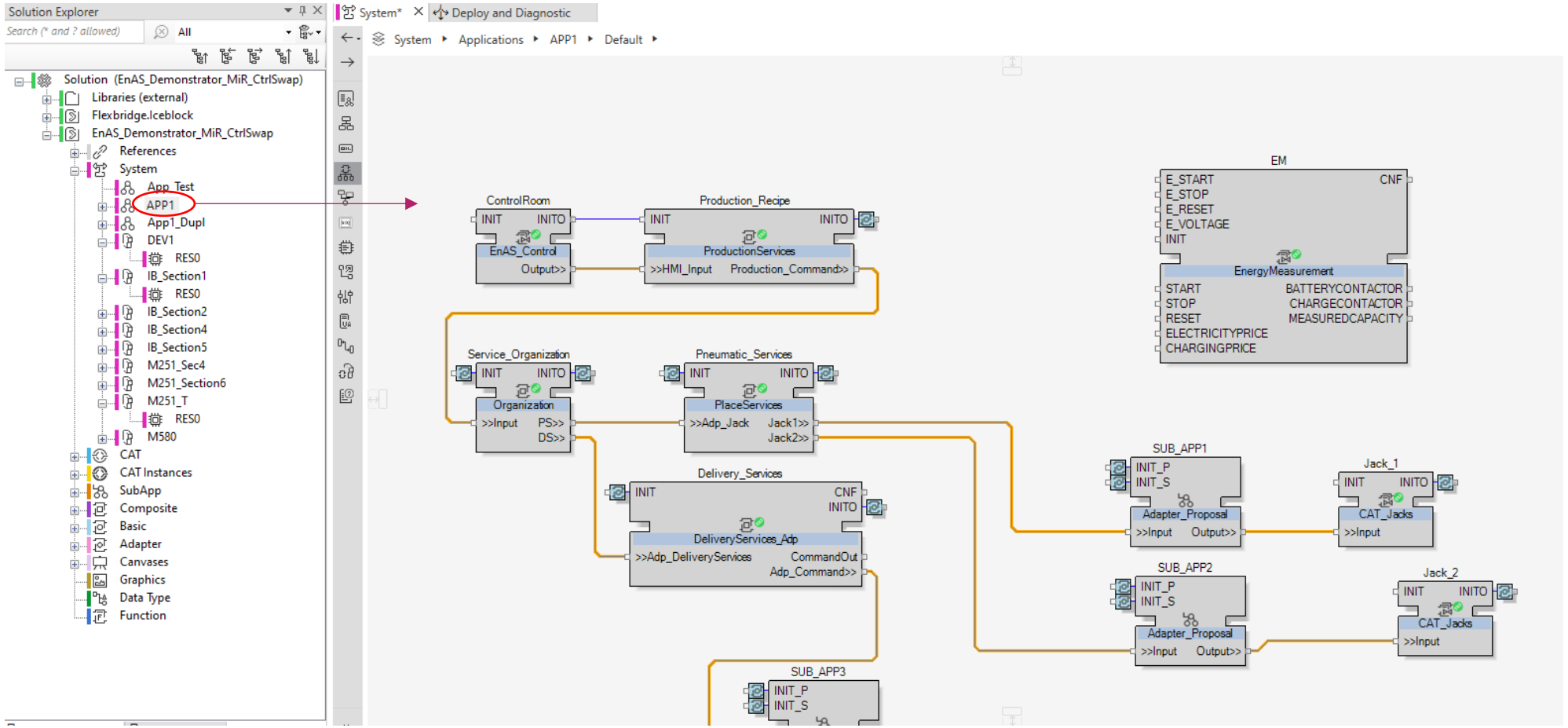


Application

- Application is a design artefact
- Consists of a network of FBs
- Designed independently from hardware where it will be deployed to
- Deployment: mapped to one or several devices

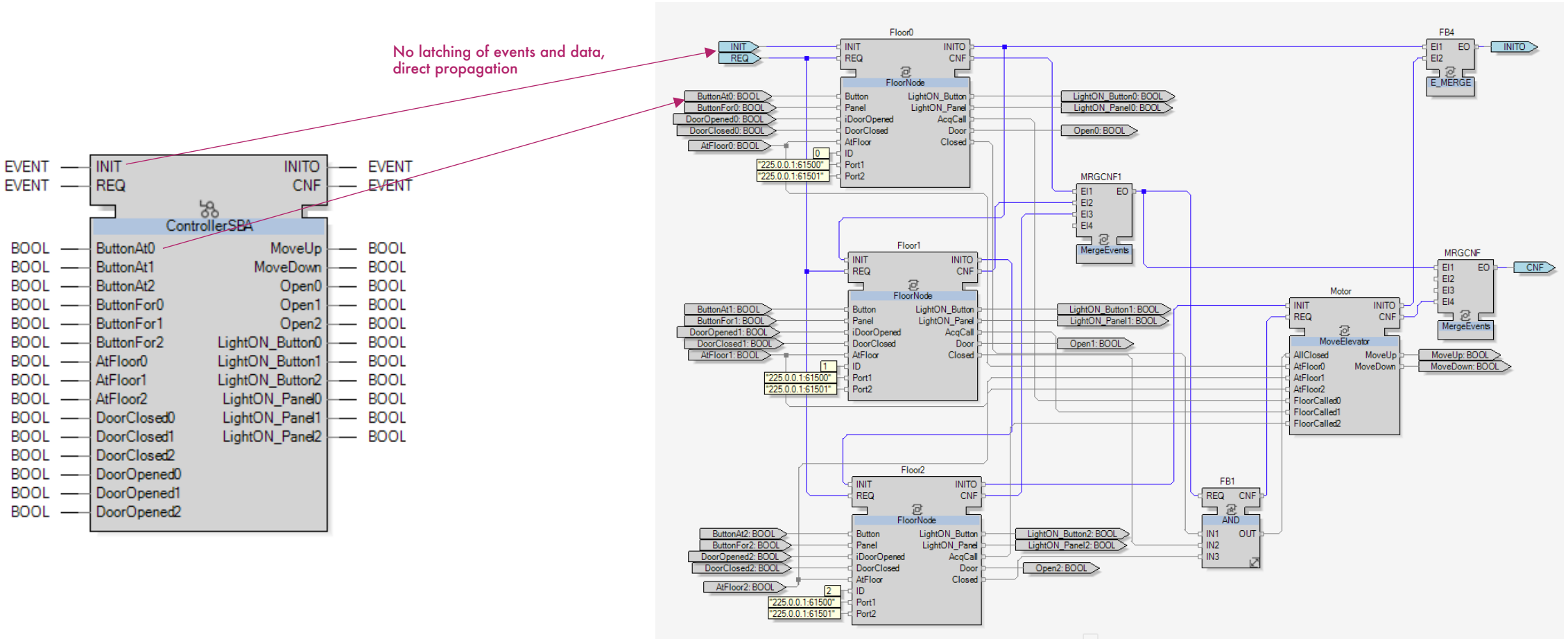


Application: Example



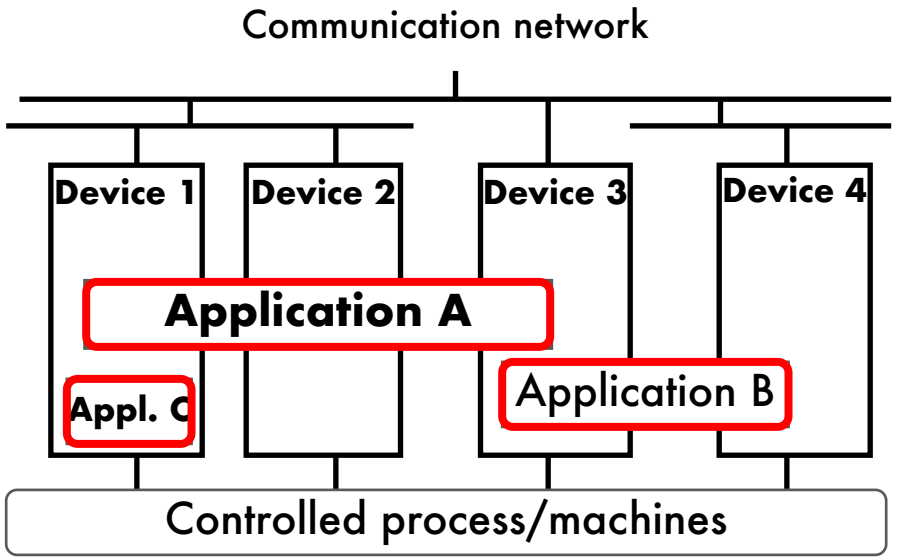
Subapplication

- Type definition, like for function blocks
- Interface, but without event-data associations
- Defined by a function block network, like Composite FB



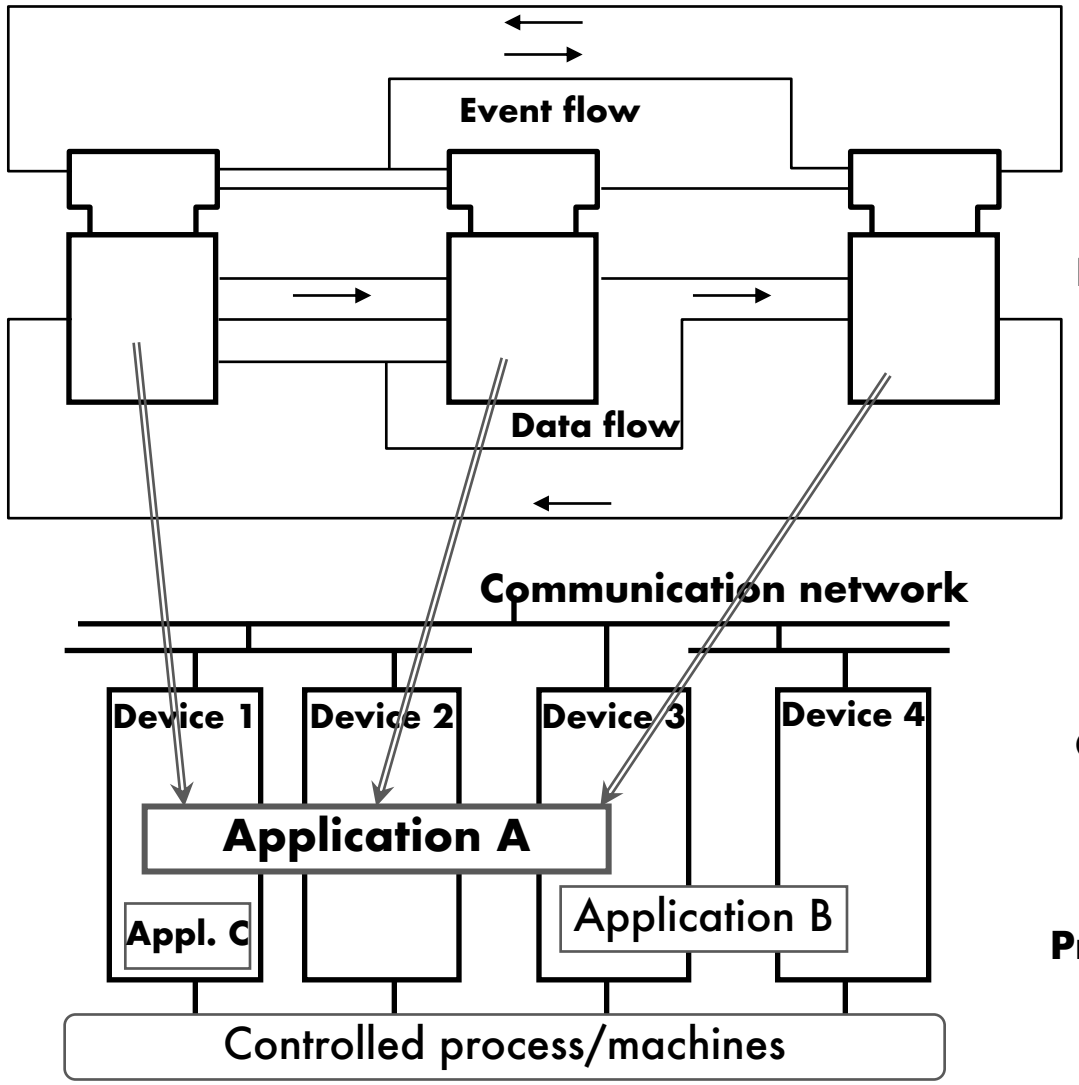
System Configuration

- Represents the physical deployment of the design
- Consists of:
  - Communication network
  - Devices
  - Controlled process and machines



Distribution of Application

- Application can be deployed to several devices
- A device in the system can contain and execute parts of different applications

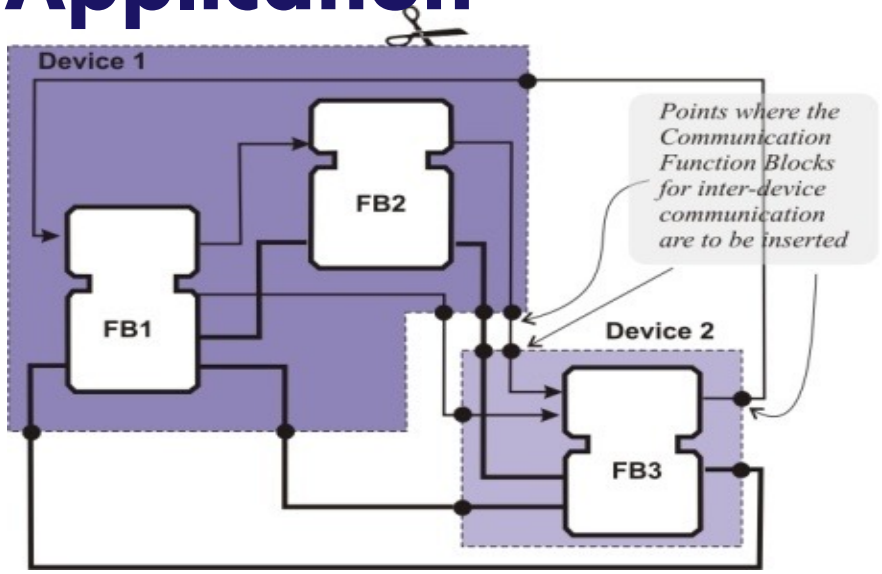


Application = Function Block Network

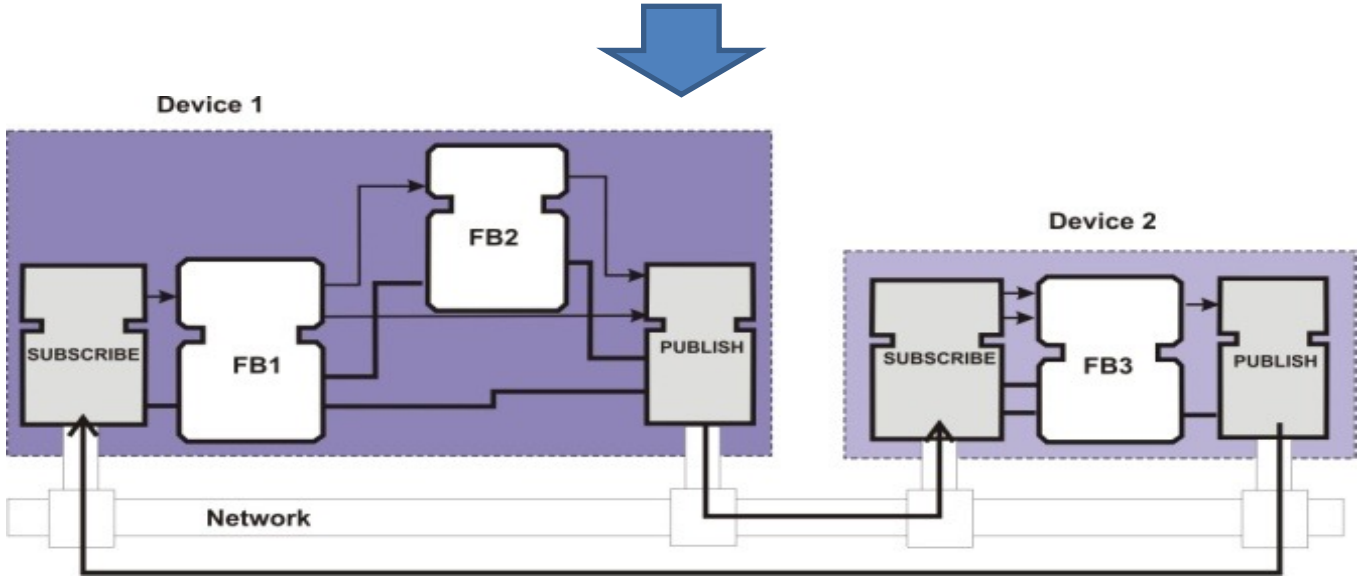
System = Communication Network + Devices + Process/Machines

75

Distribution of Application

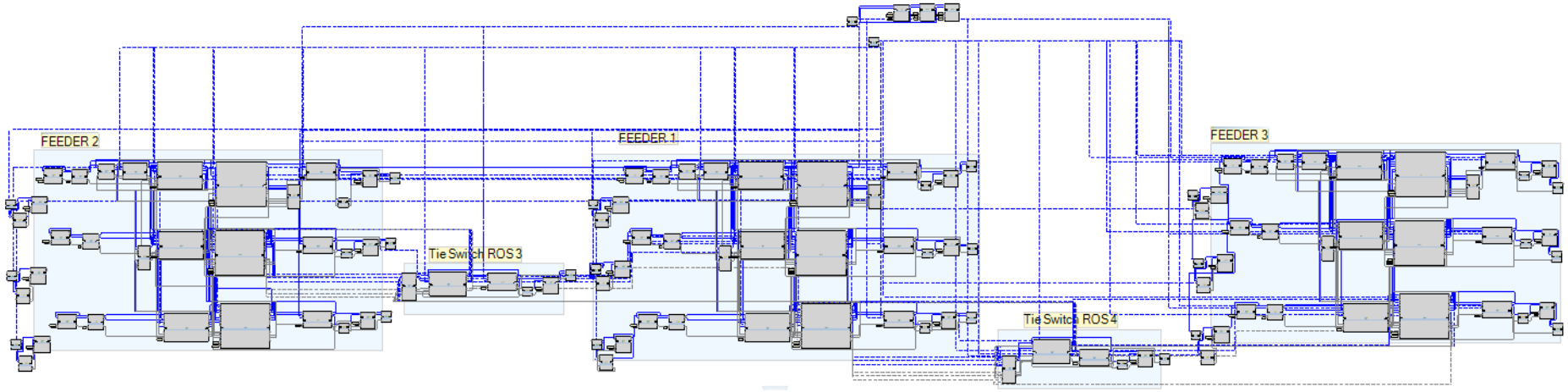


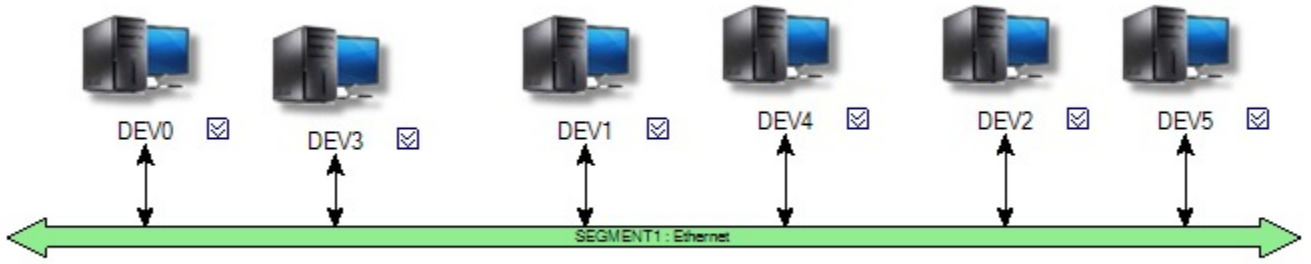
Tools usually insert the communication blocks automatically



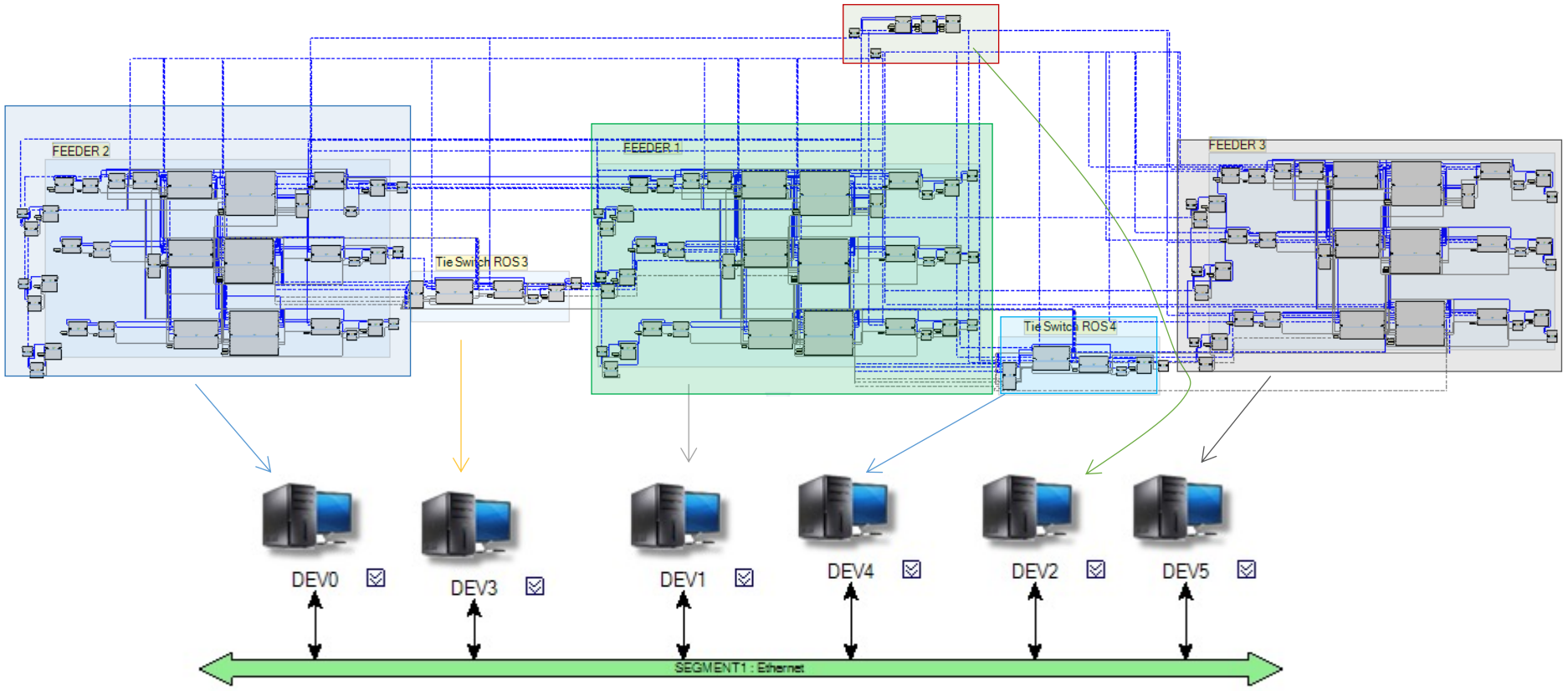
76

Example: Distributed systems configuration





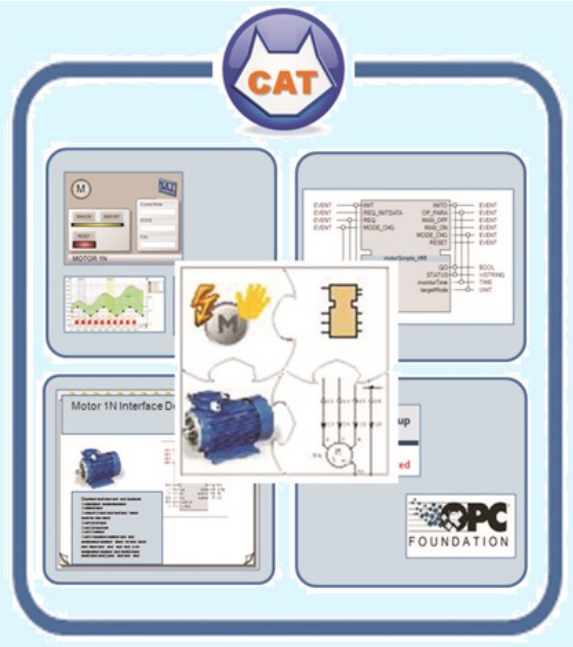
Example: Distributed systems configuration



Composite Automation Type (CAT)

Composite Automation Type (CAT)

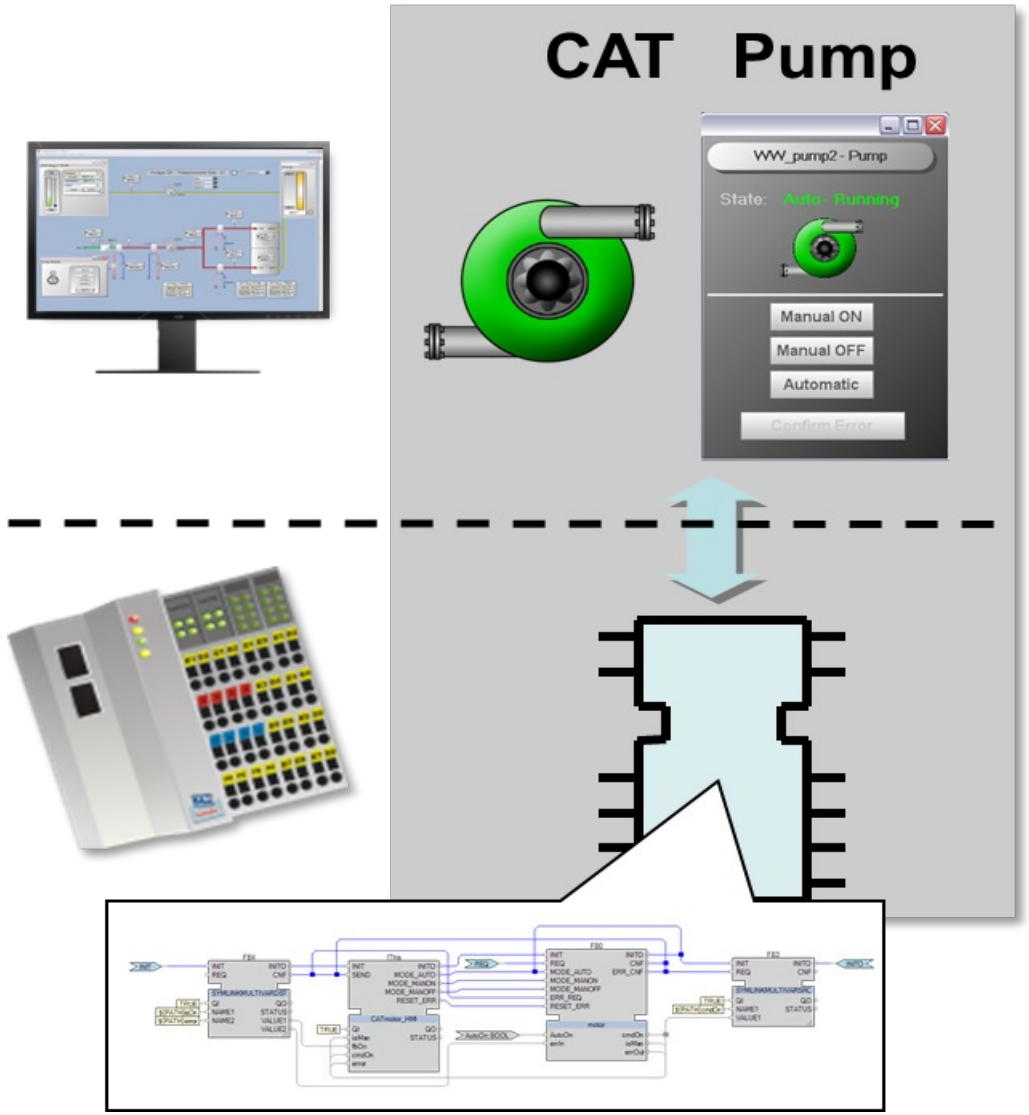
- Object-oriented mechanism of creating visualisation for automation projects
  - Object-oriented
  - Self-contained
  - Reconfigurable
- Each CAT consists of:
  - Visualization symbols
  - Control logic
  - Plant models
  - HMI and I/O connections
- Supports:
  - Direct hardware interaction
  - Automatic deployment of control logic



80

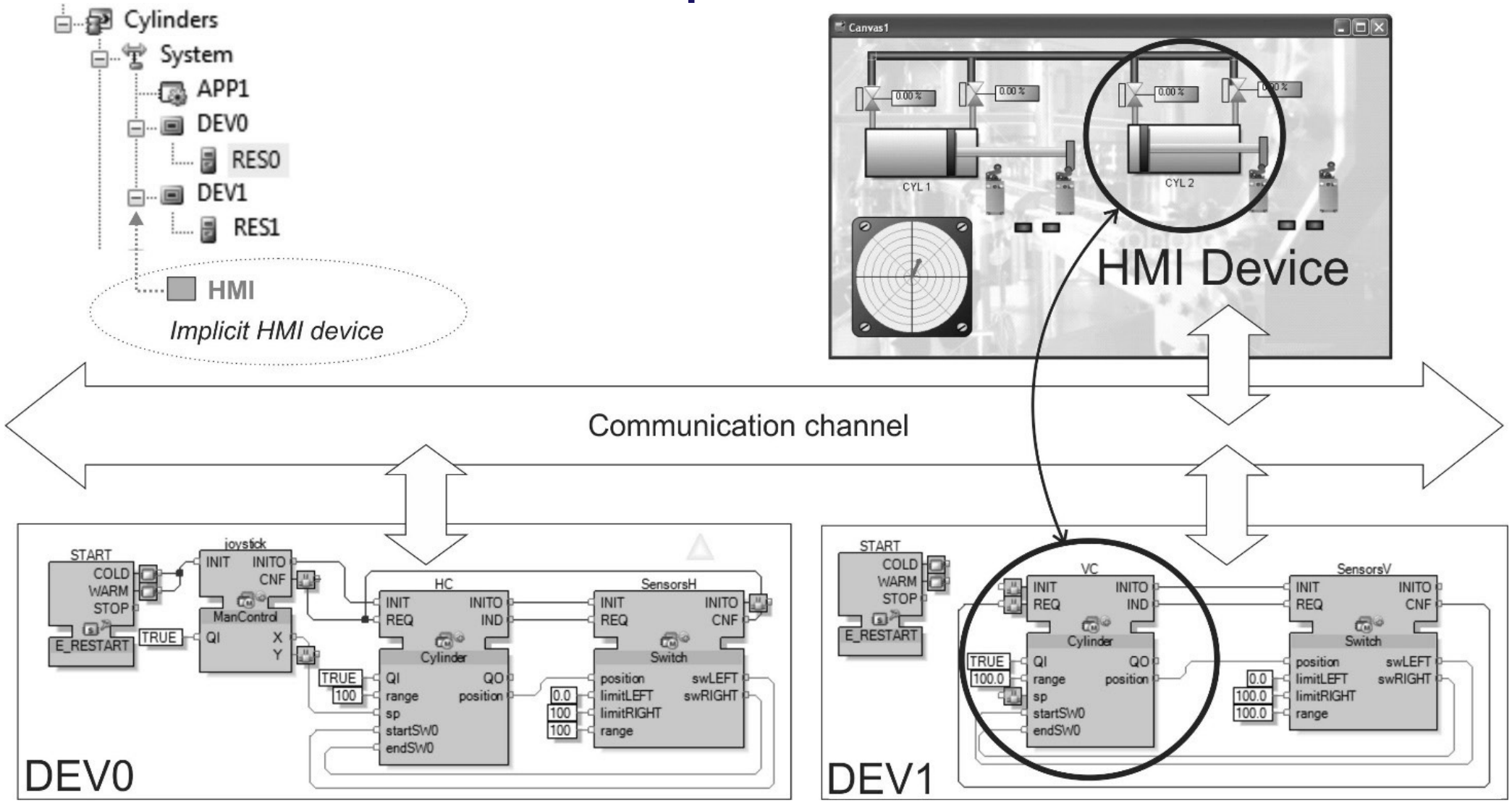
CAT

- Software representation of real instruments/devices, e.g. pump
- Predefined control structure with abstract hardware interface
- Automatic HMI configuration and connection



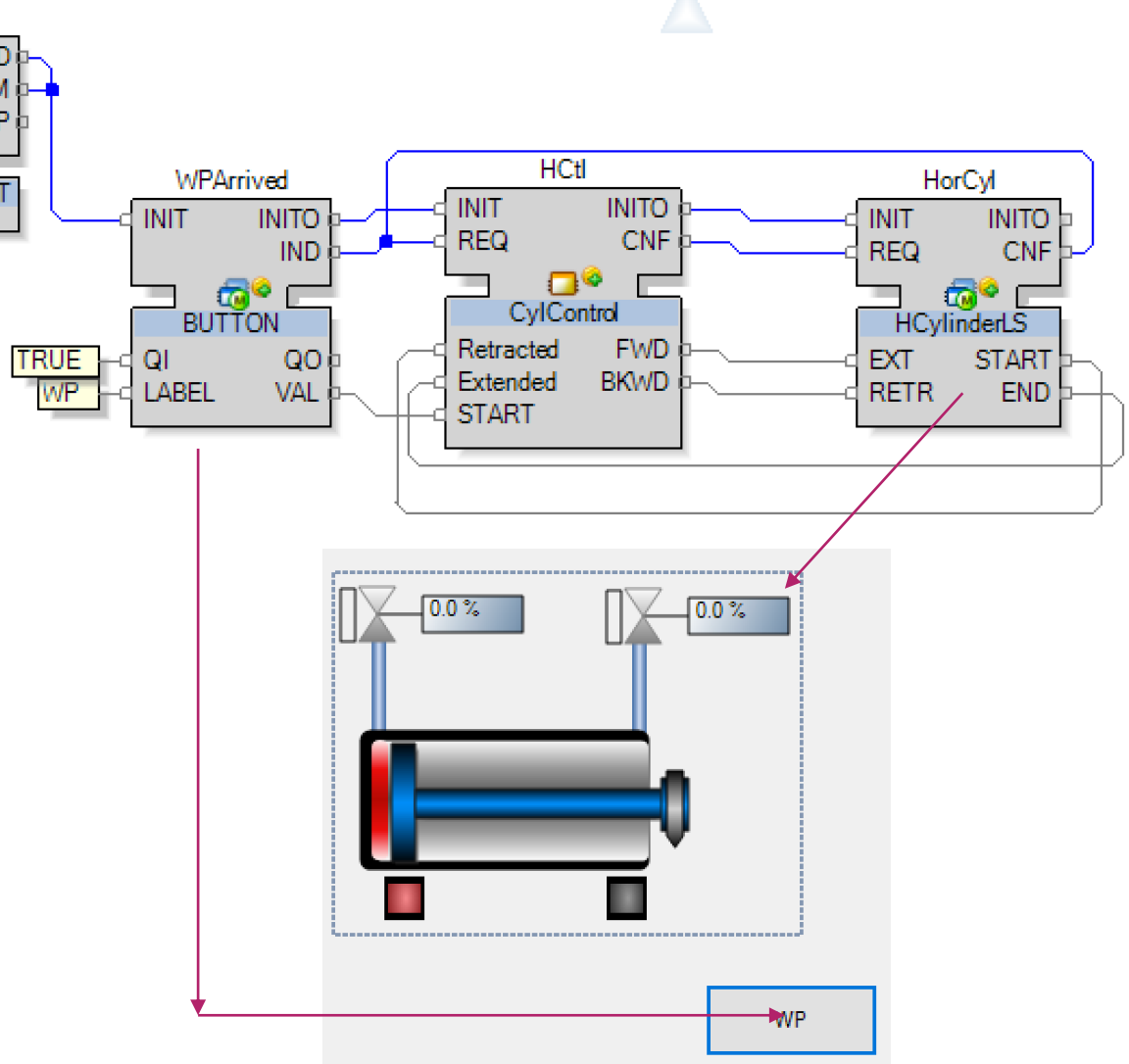
81

System architecture with an implicit HMI device



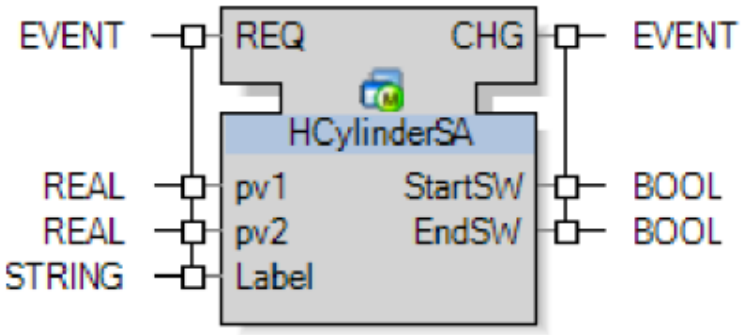
82

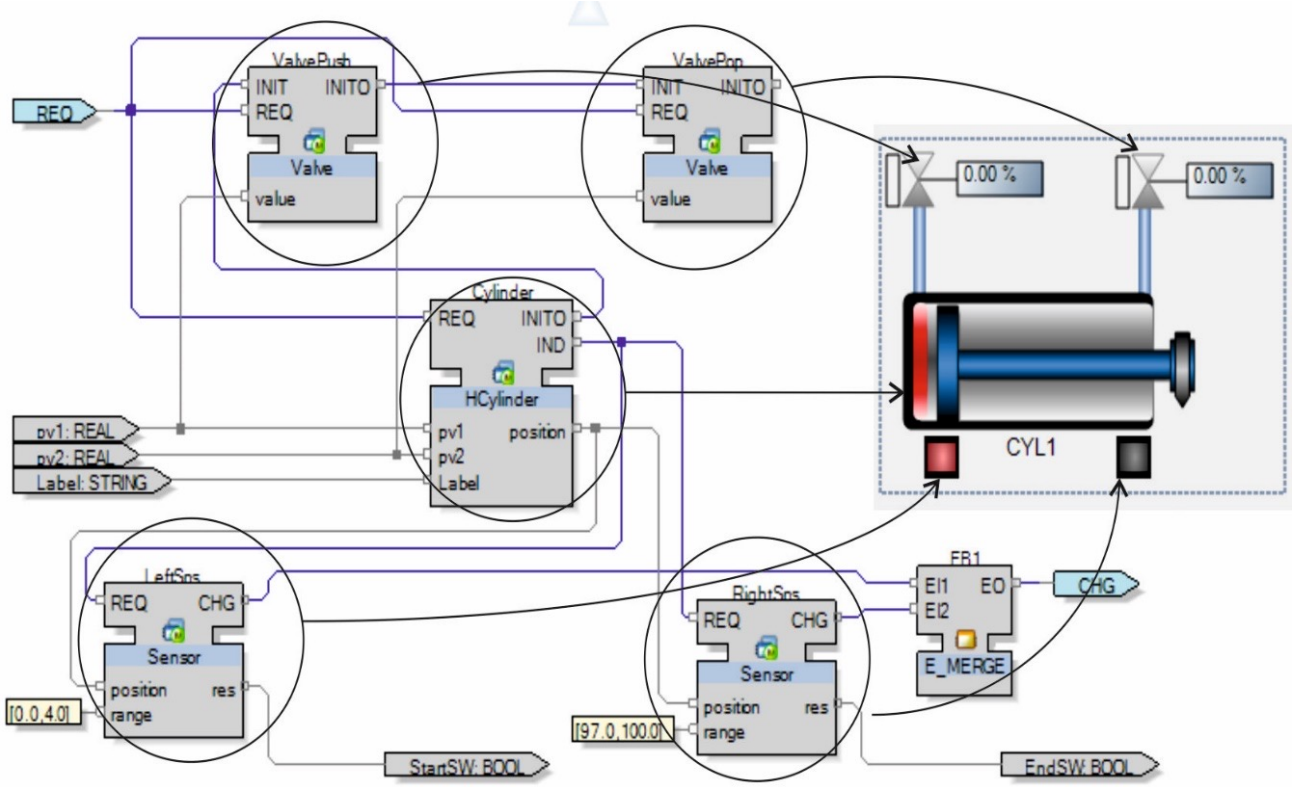
Application with CATs



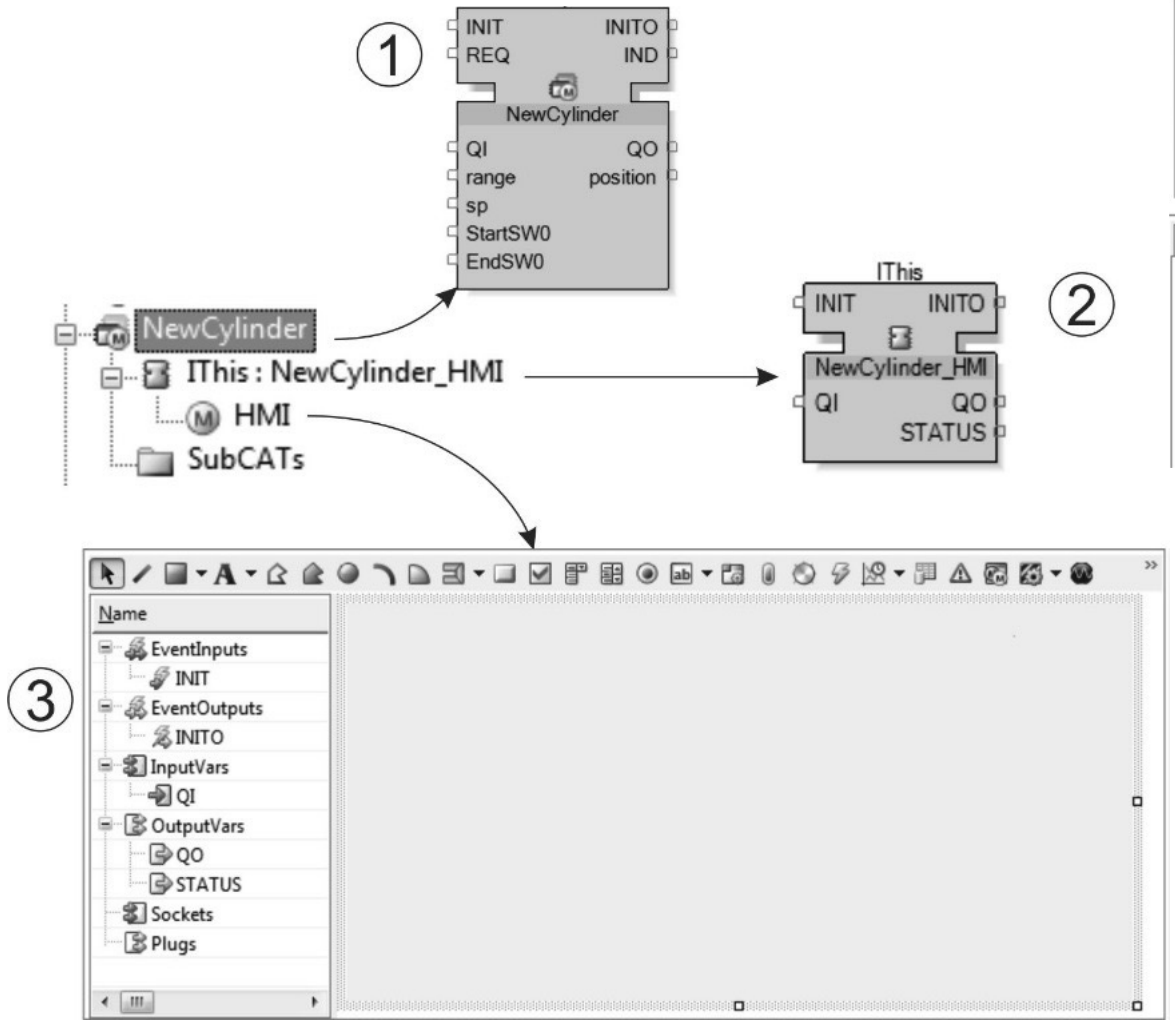
83

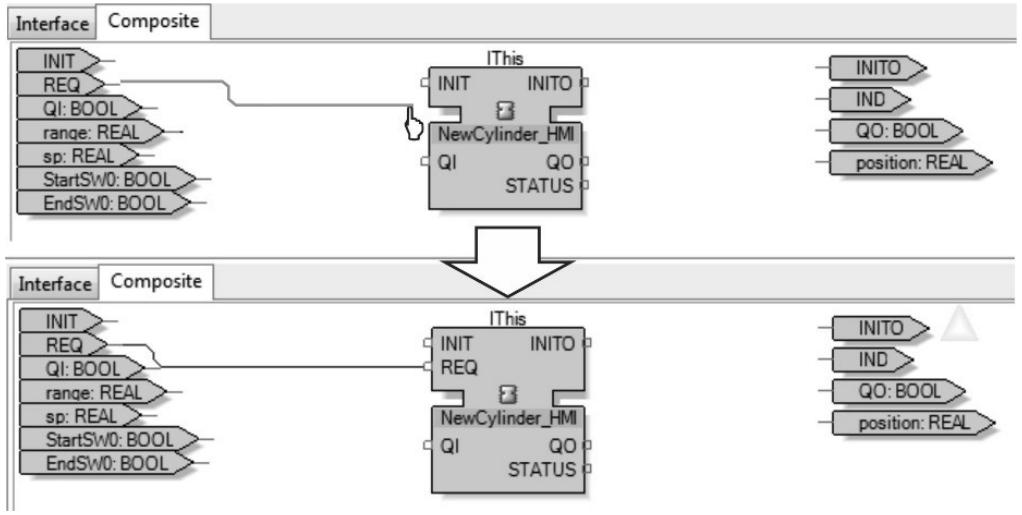
Model of Composite Cylinder





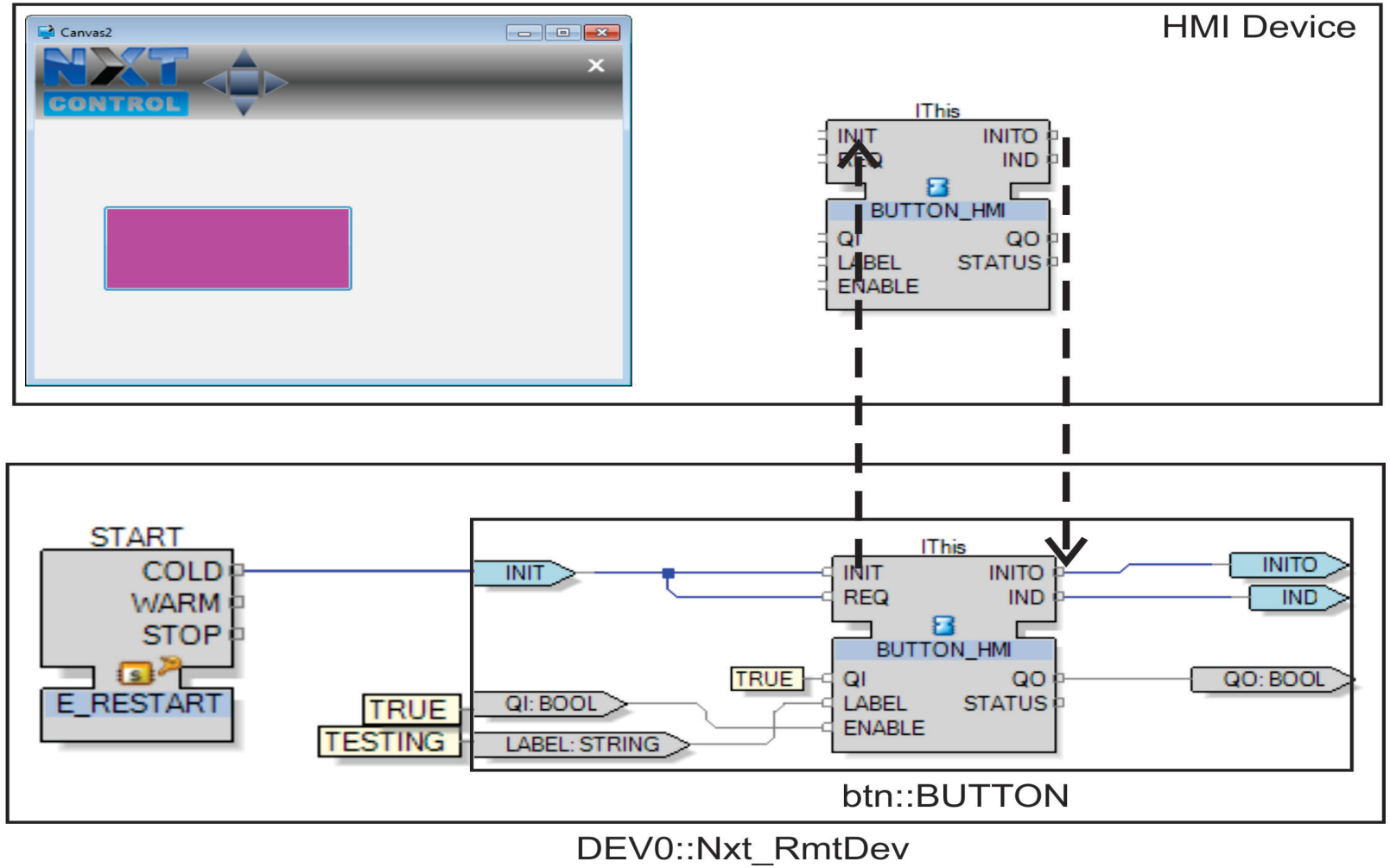
CAT creation





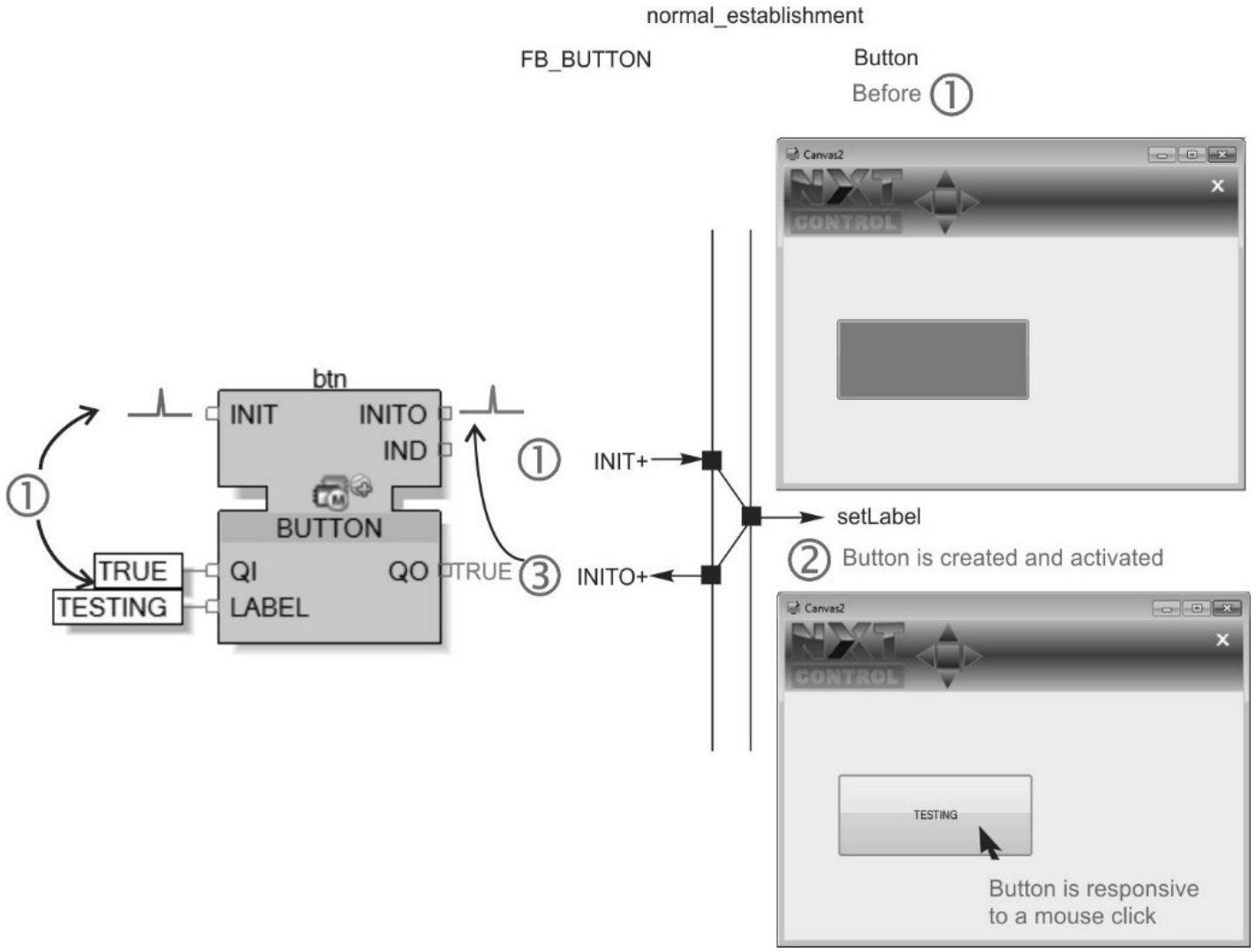
85

How does a CAT work?



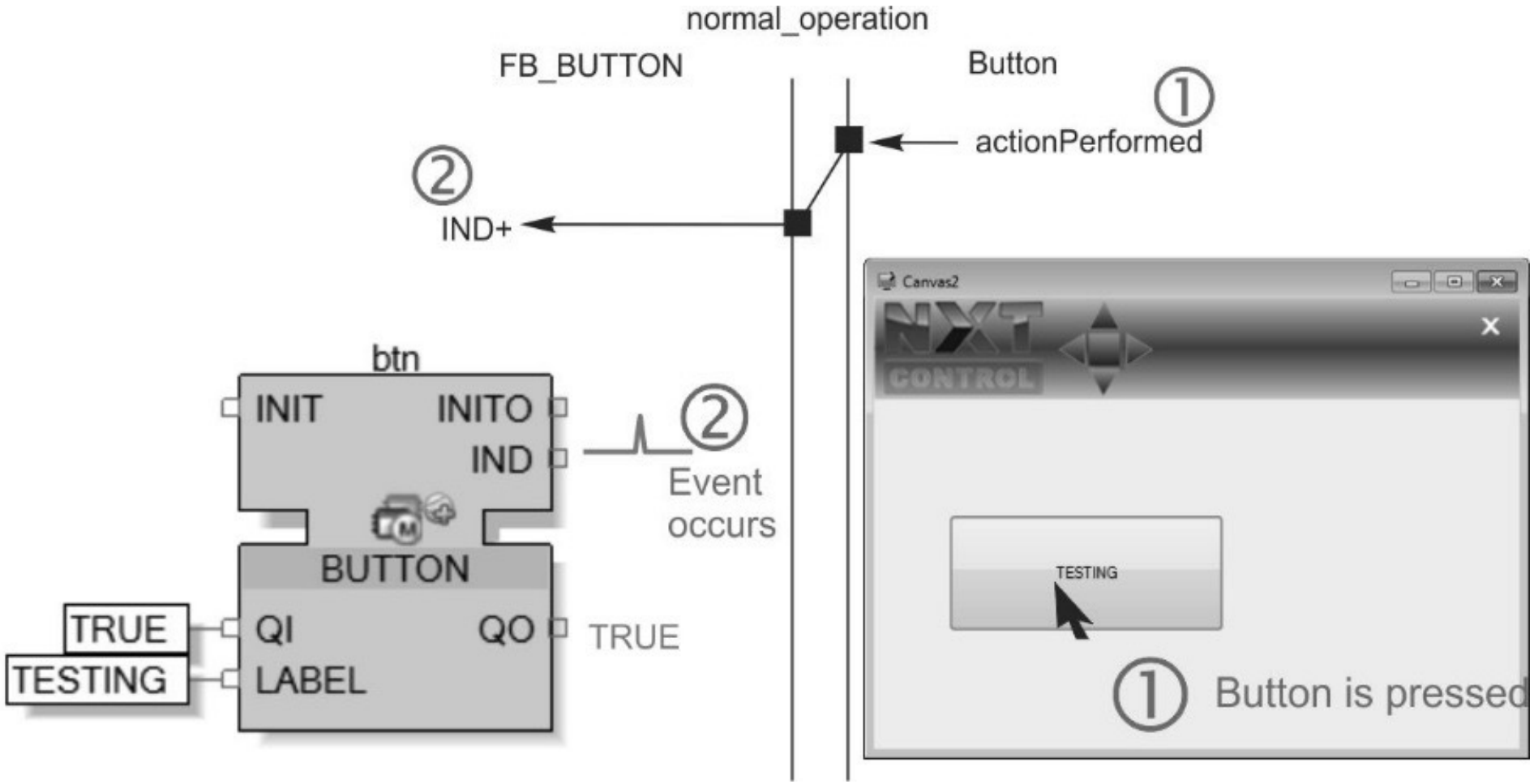
86

Normal establishment



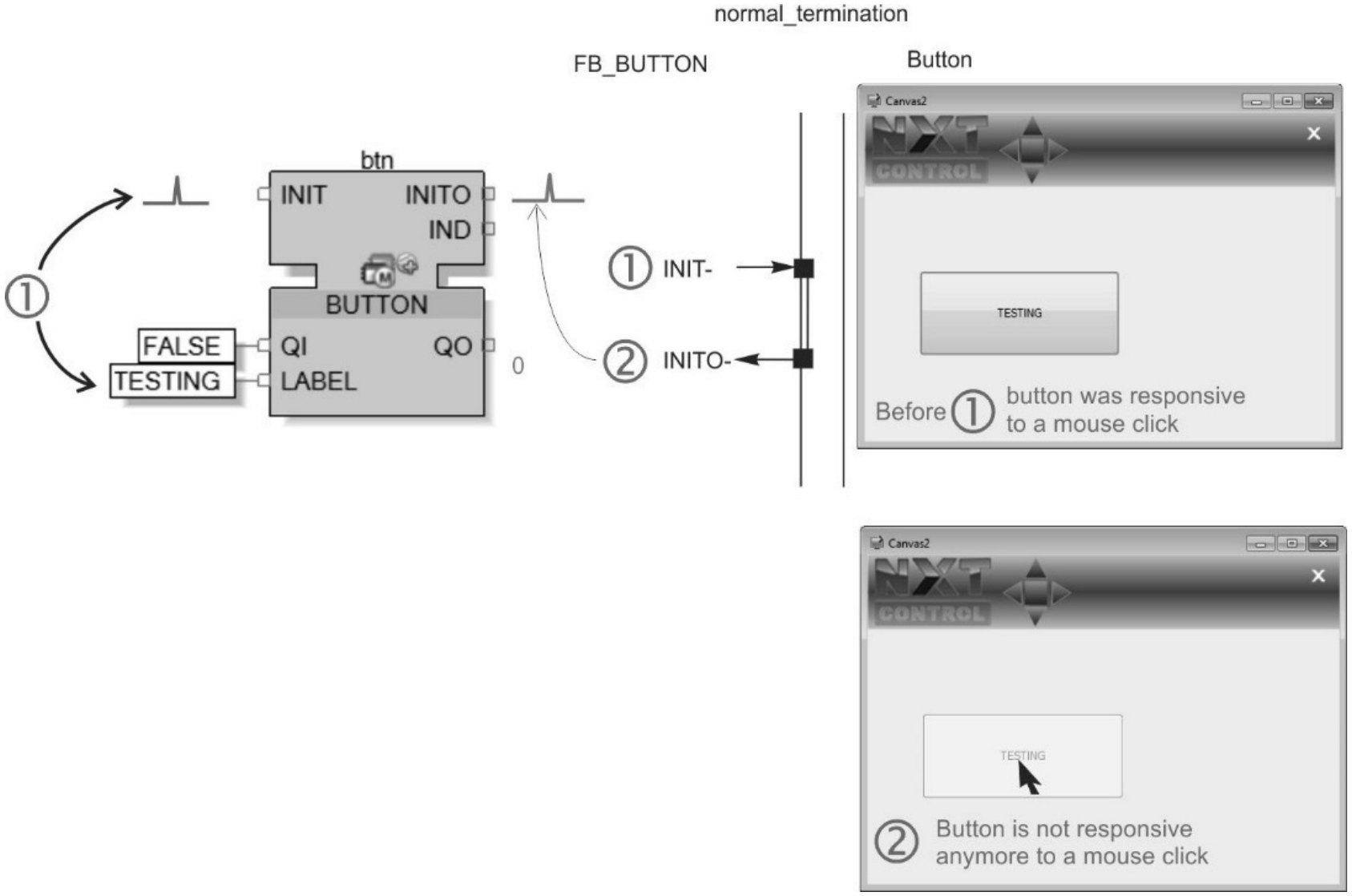
87

Normal operation



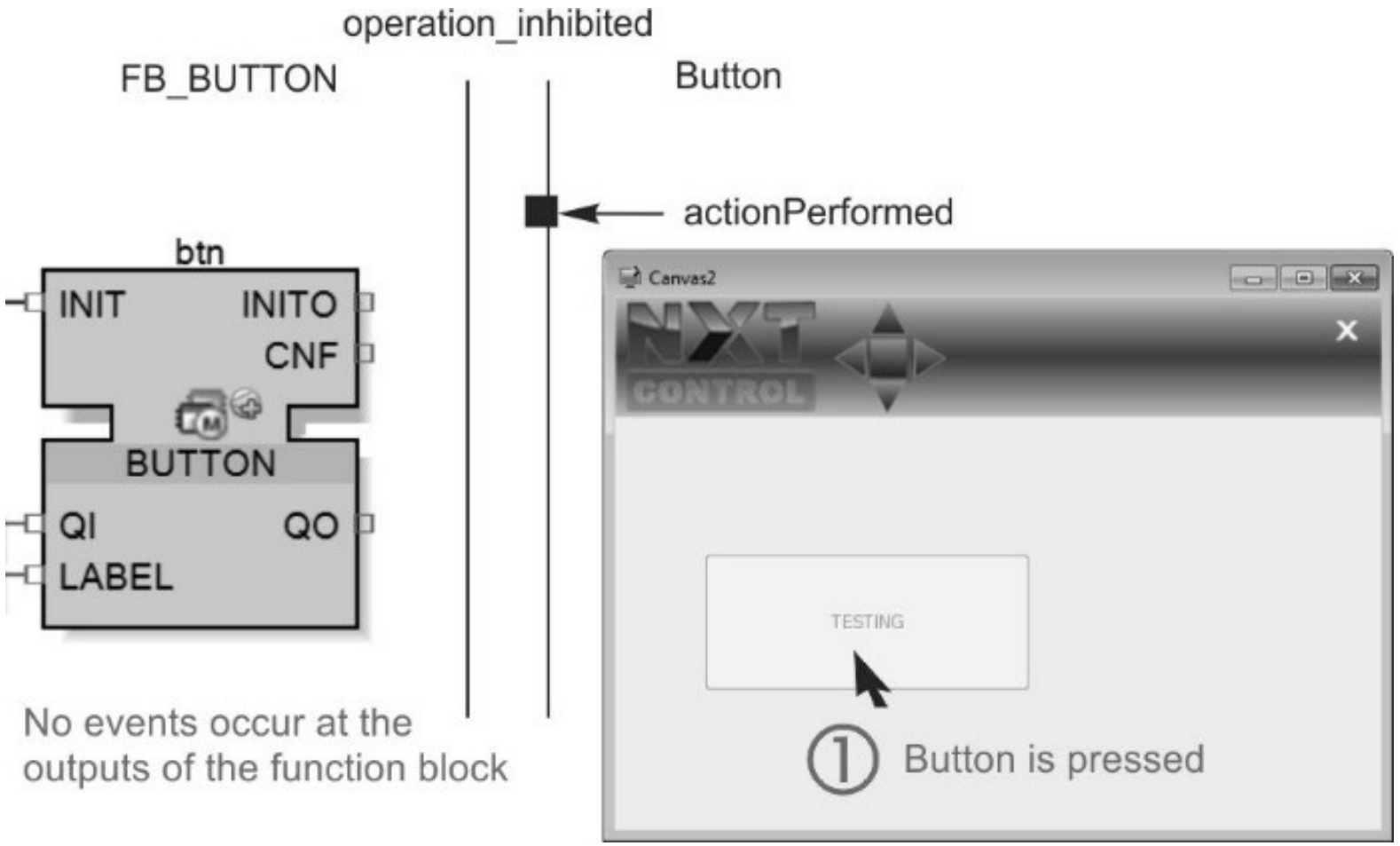
88

Normal termination



89

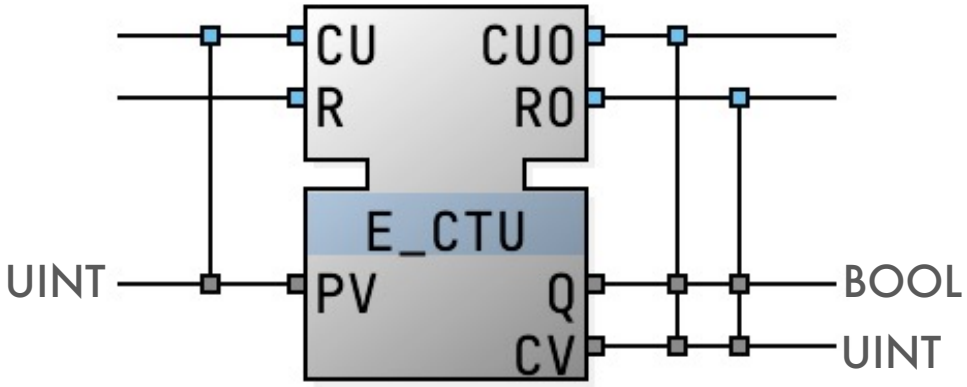
Operation is inhibited

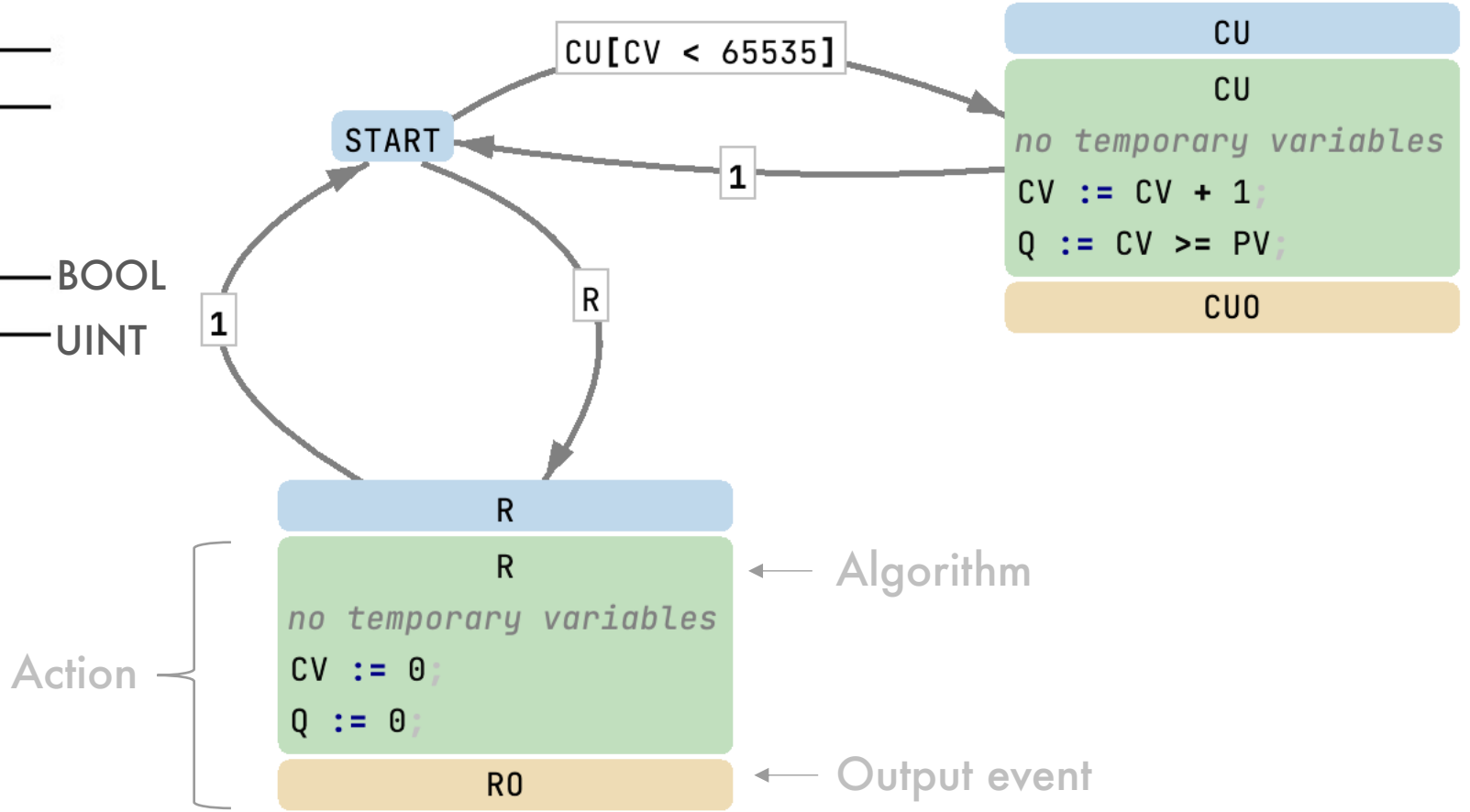


90

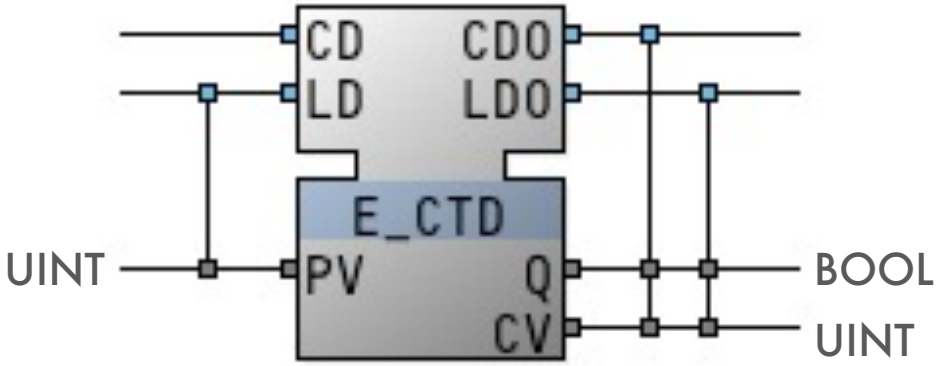
Event-handling Function Blocks

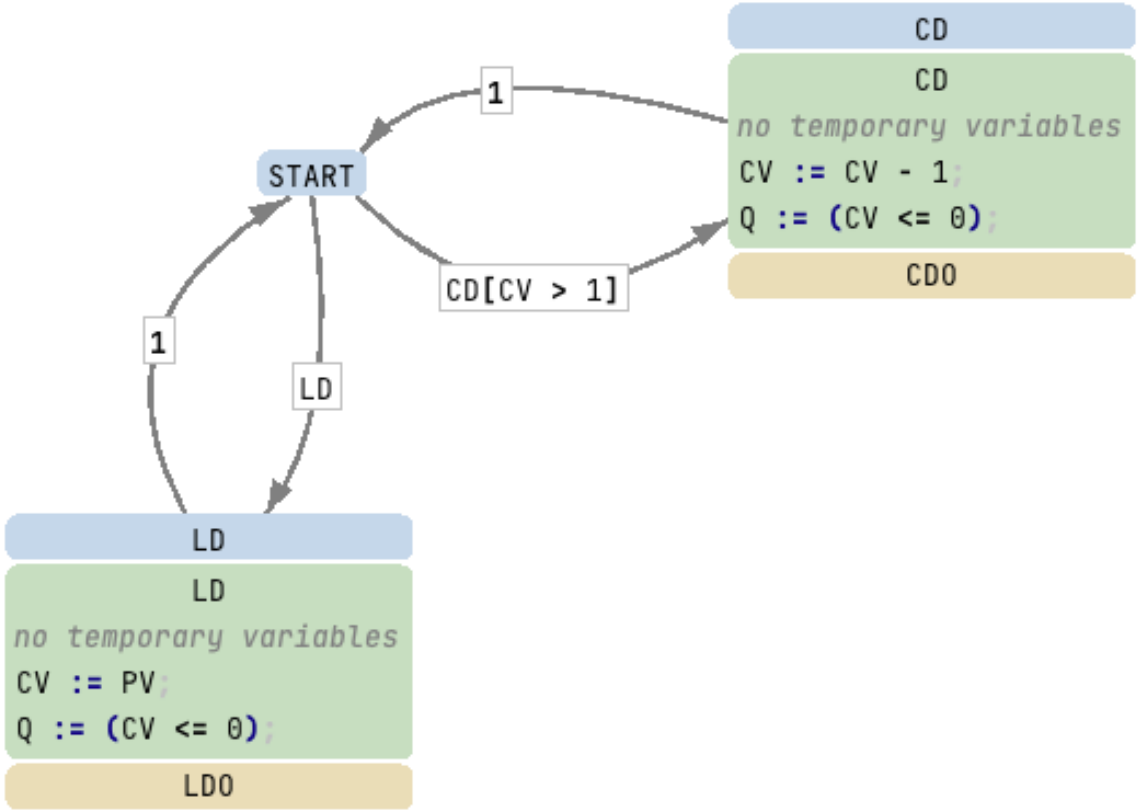
Count UP



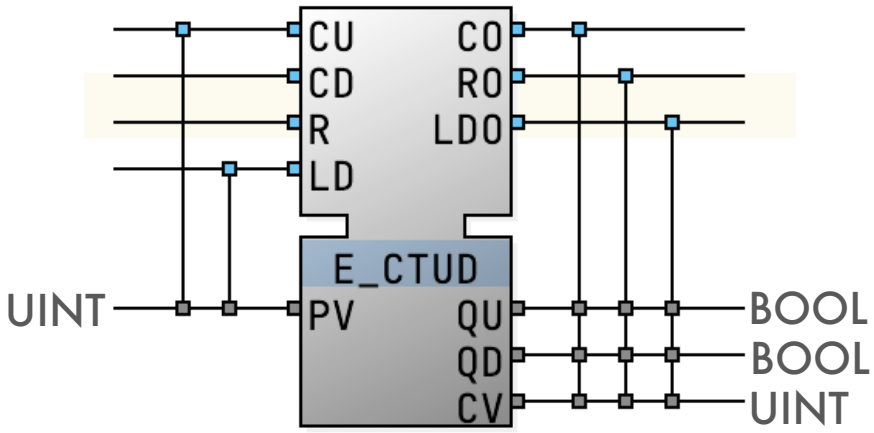


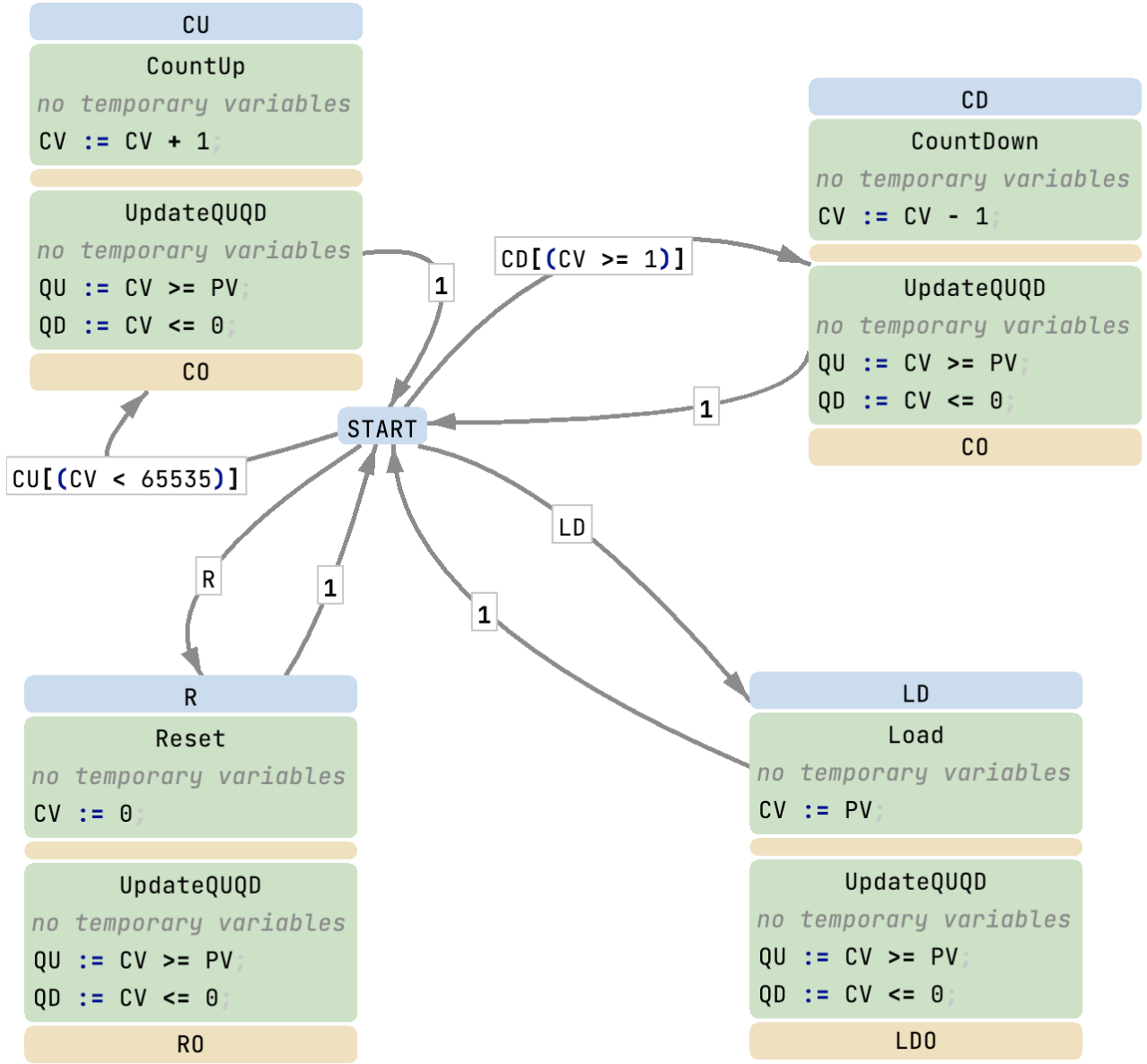
Count down



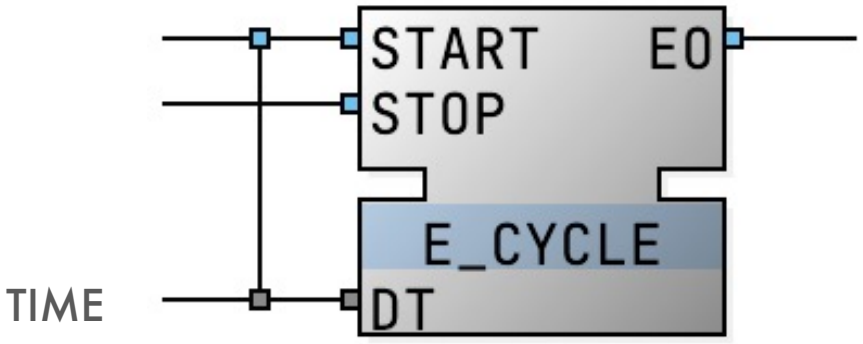


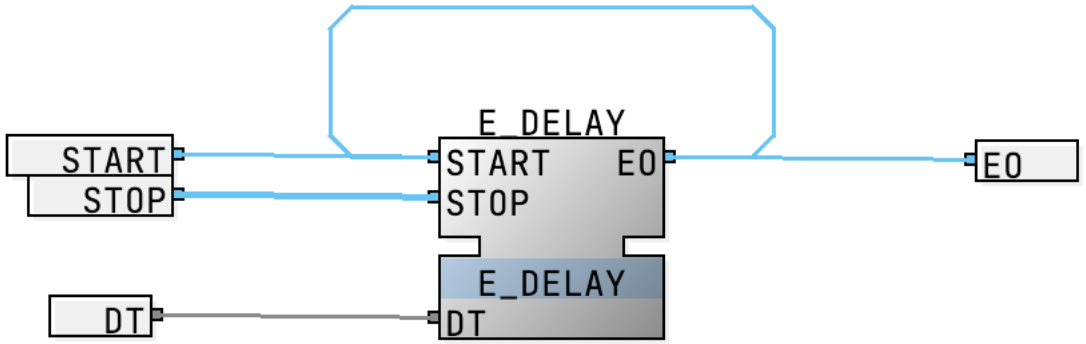
Count UP/Down



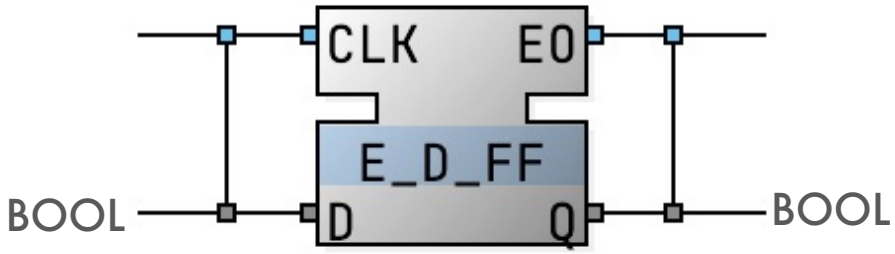


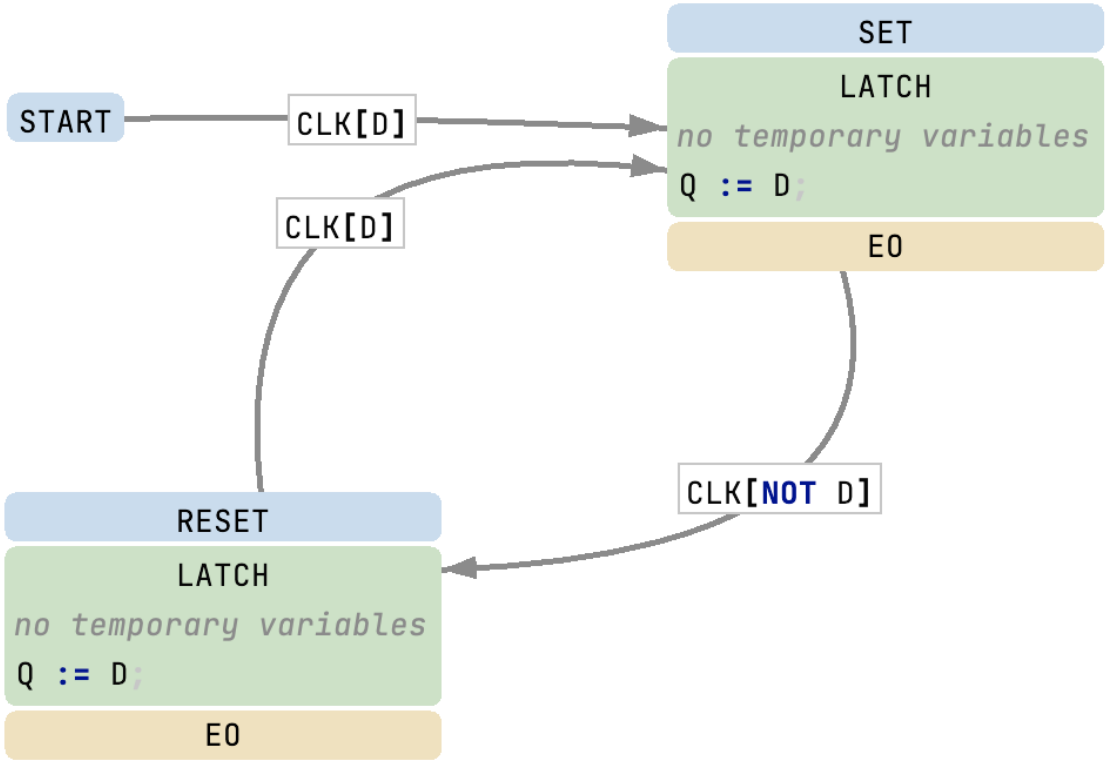
E_CYCLE

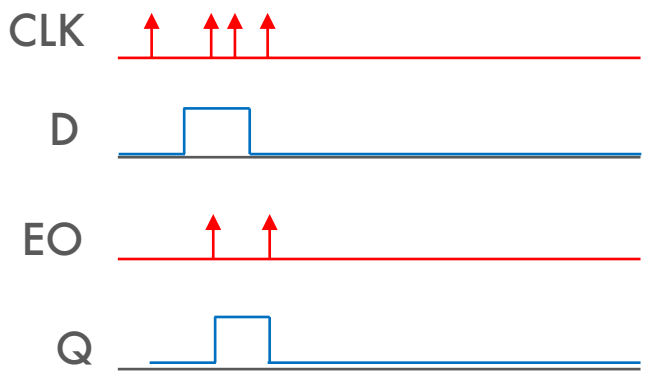




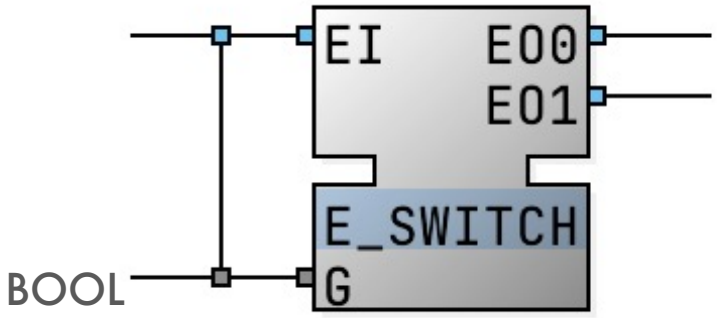
Data latch

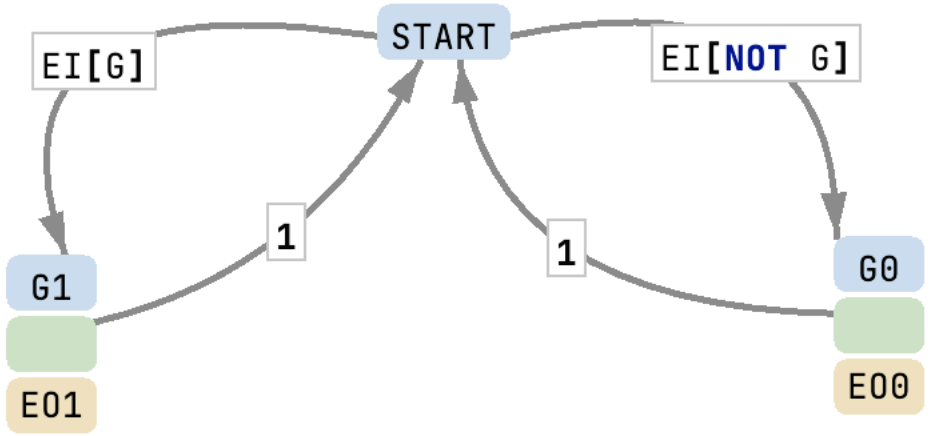


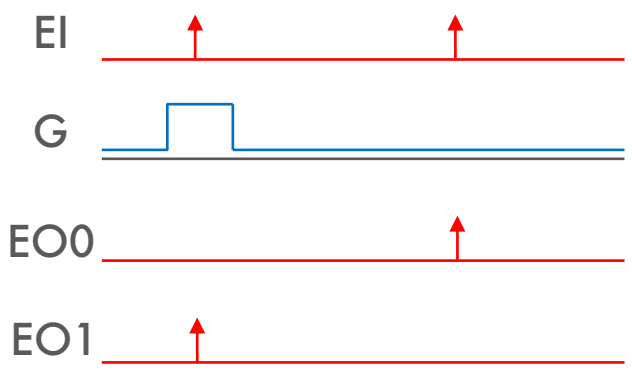




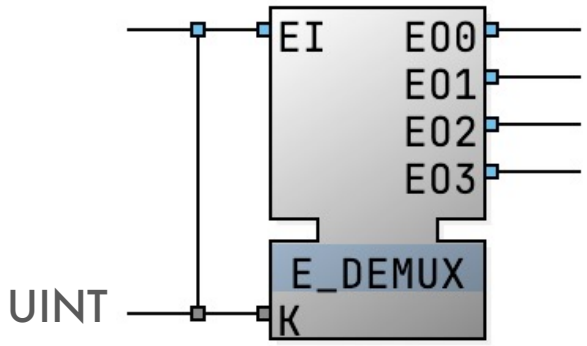
Event switch E_SWITCH

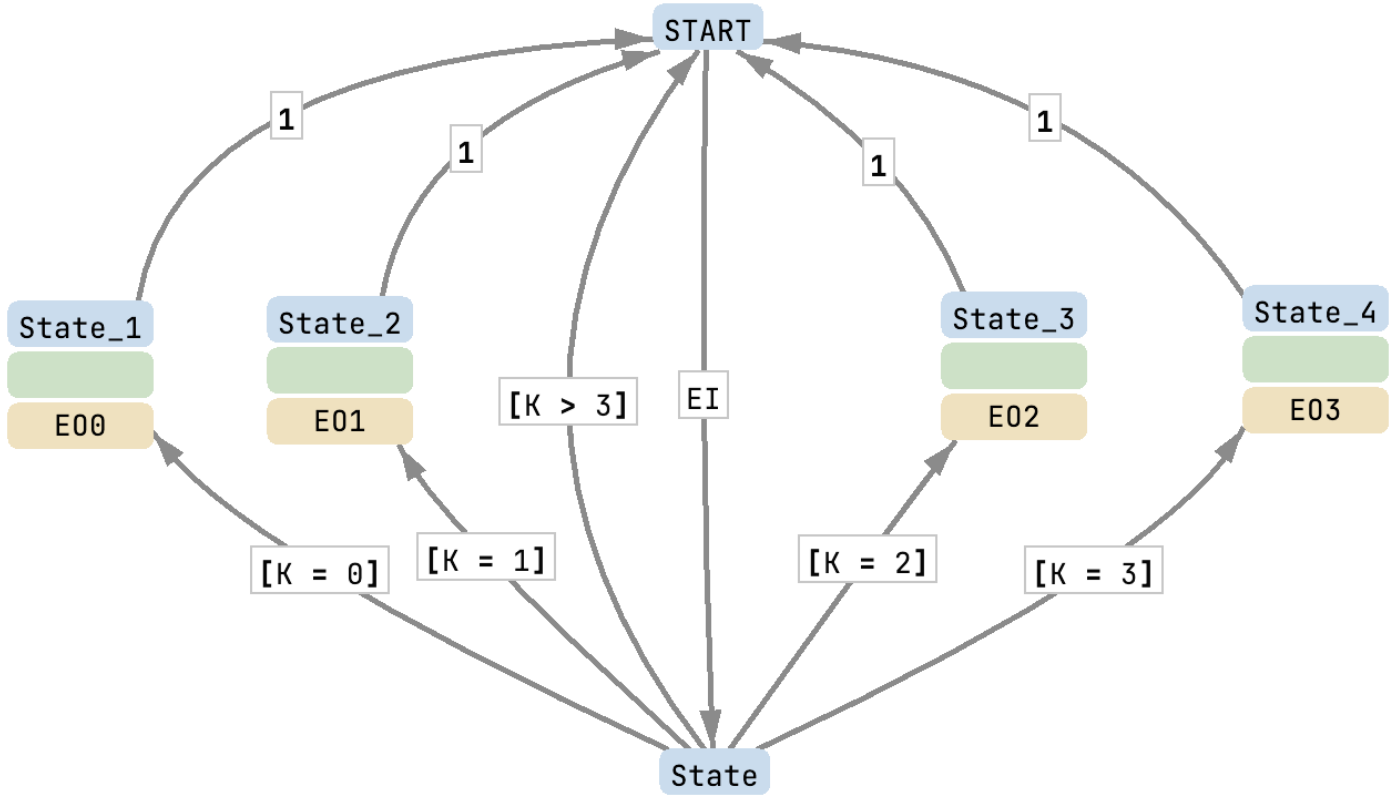




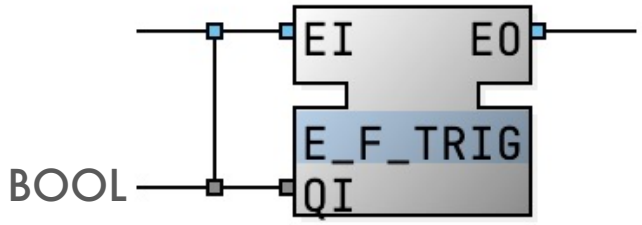


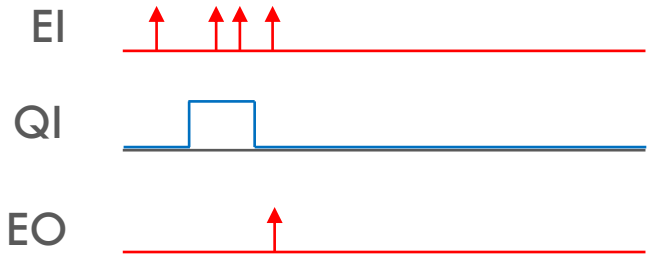
Event demultiplexor



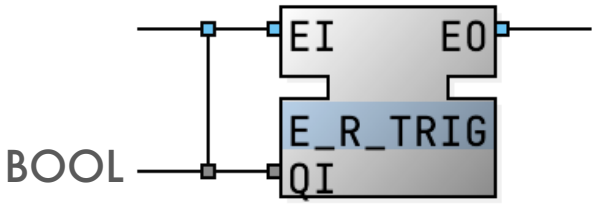


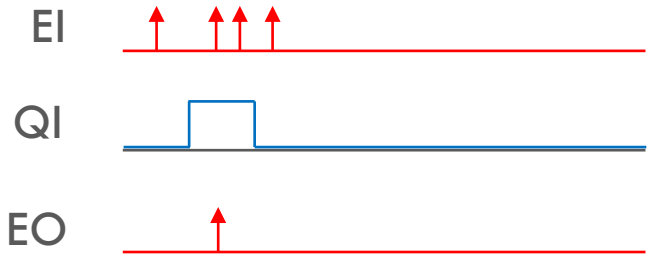
Boolean falling edge detection



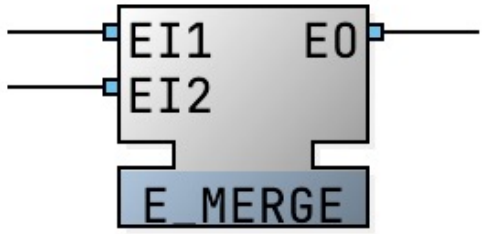


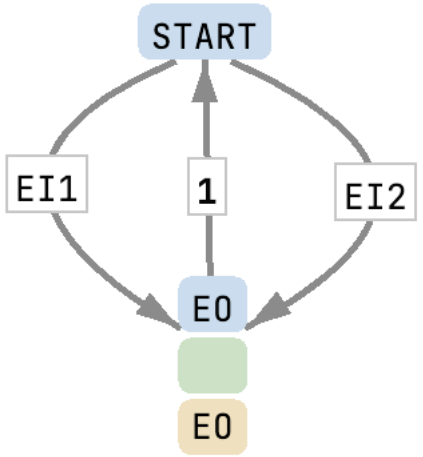
Boolean raising edge detection



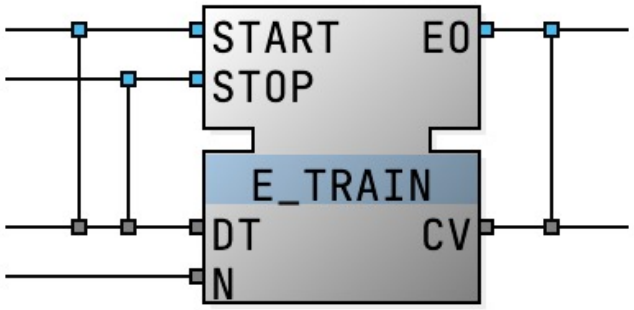


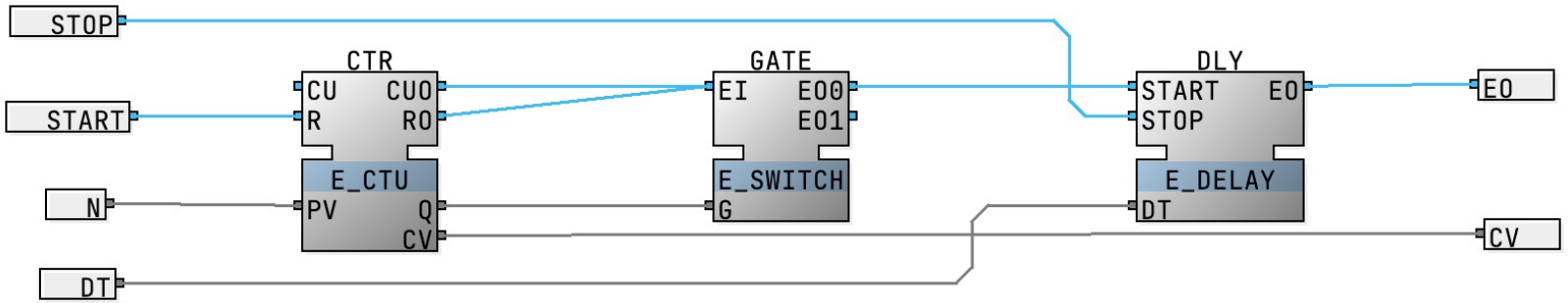
E_MERGE



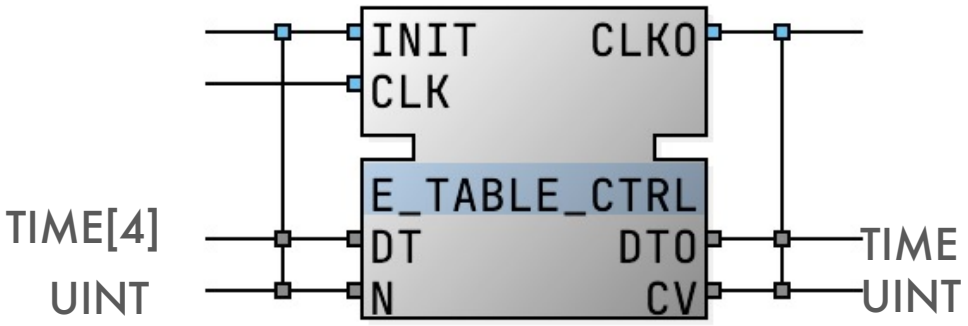


Train of events





E_TABLE_CTRL



In this example N<=4

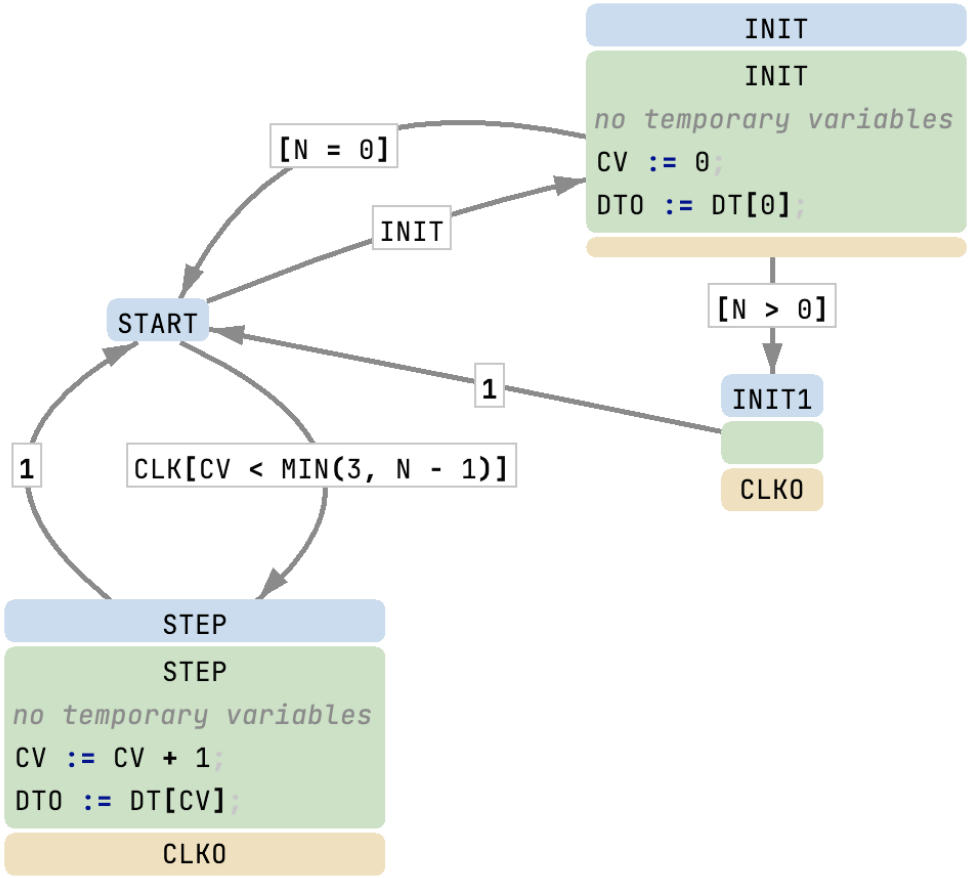
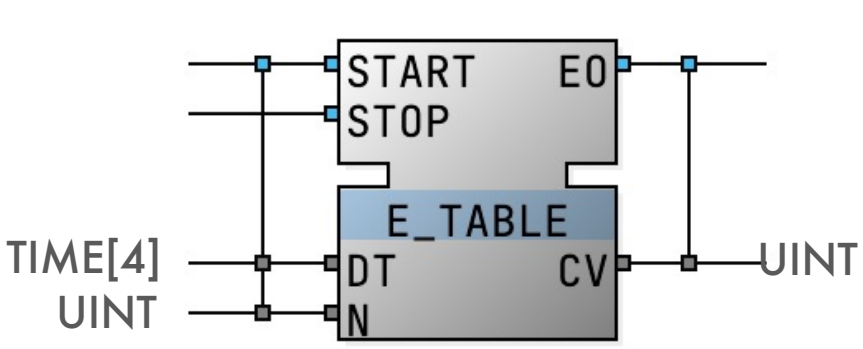
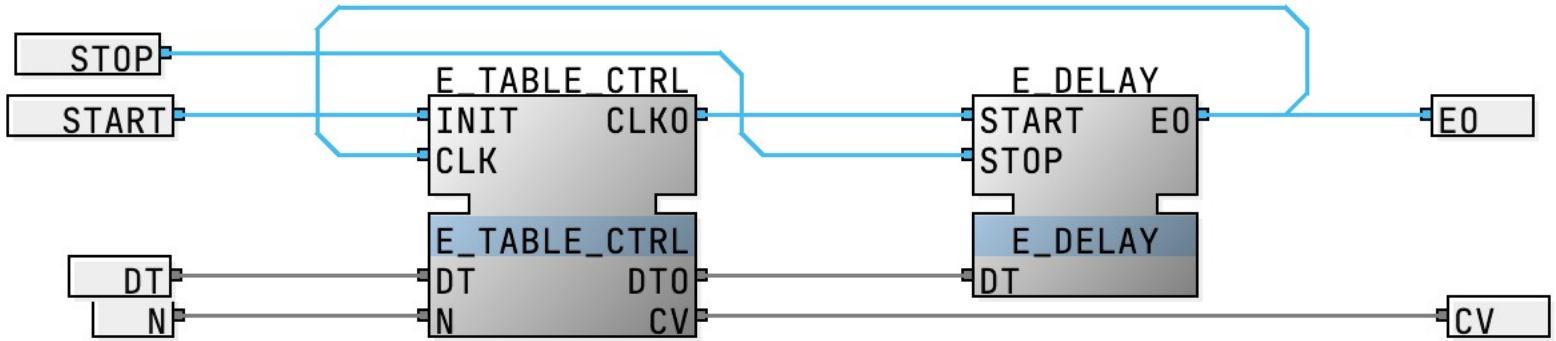
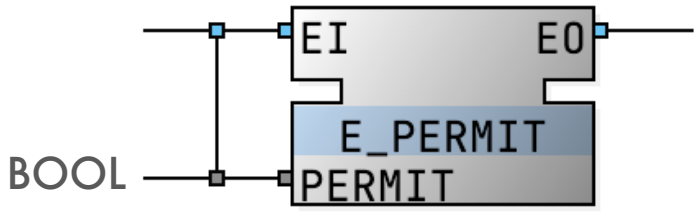


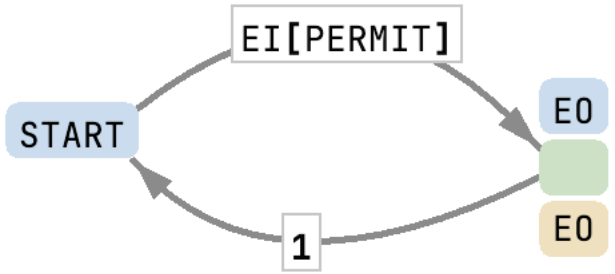
Table-driven sequence of events

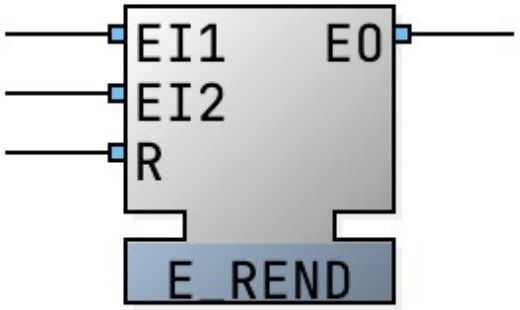


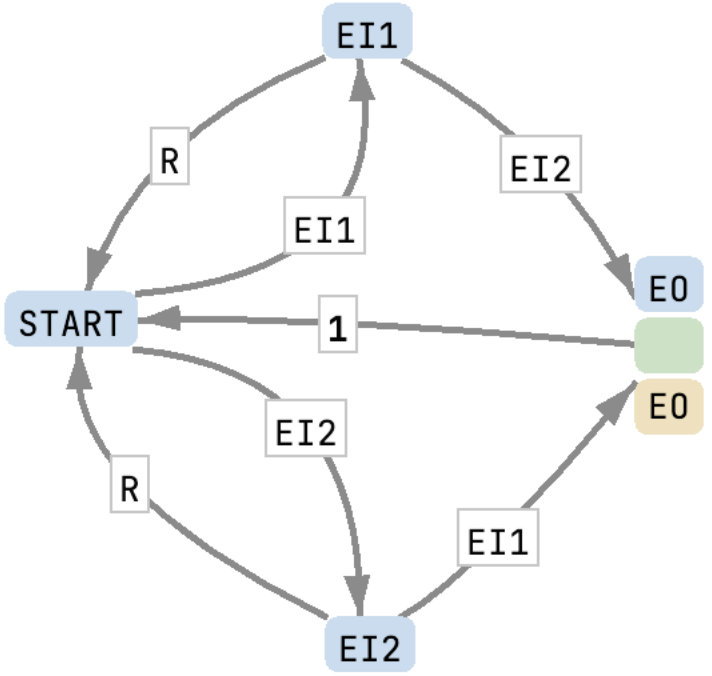


Controlled Event propagation E_PERMIT

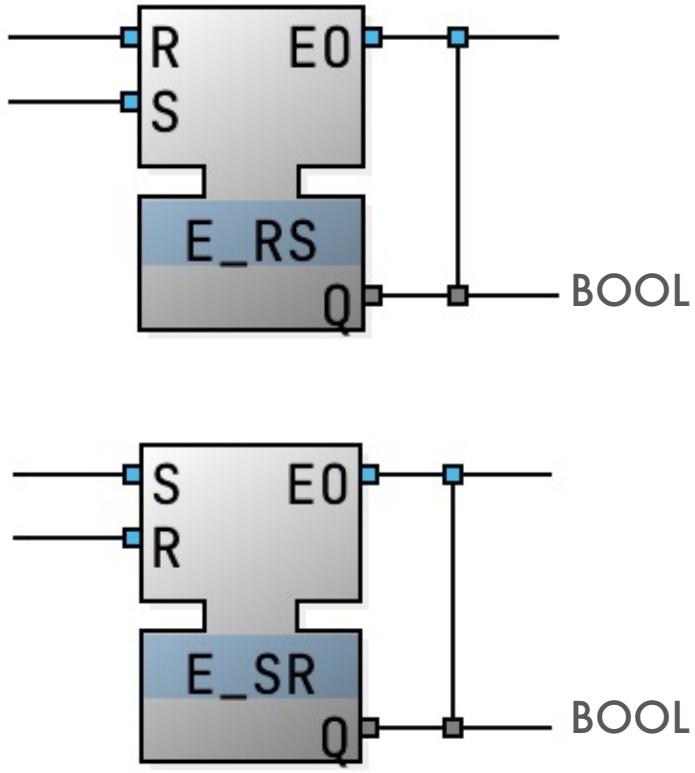


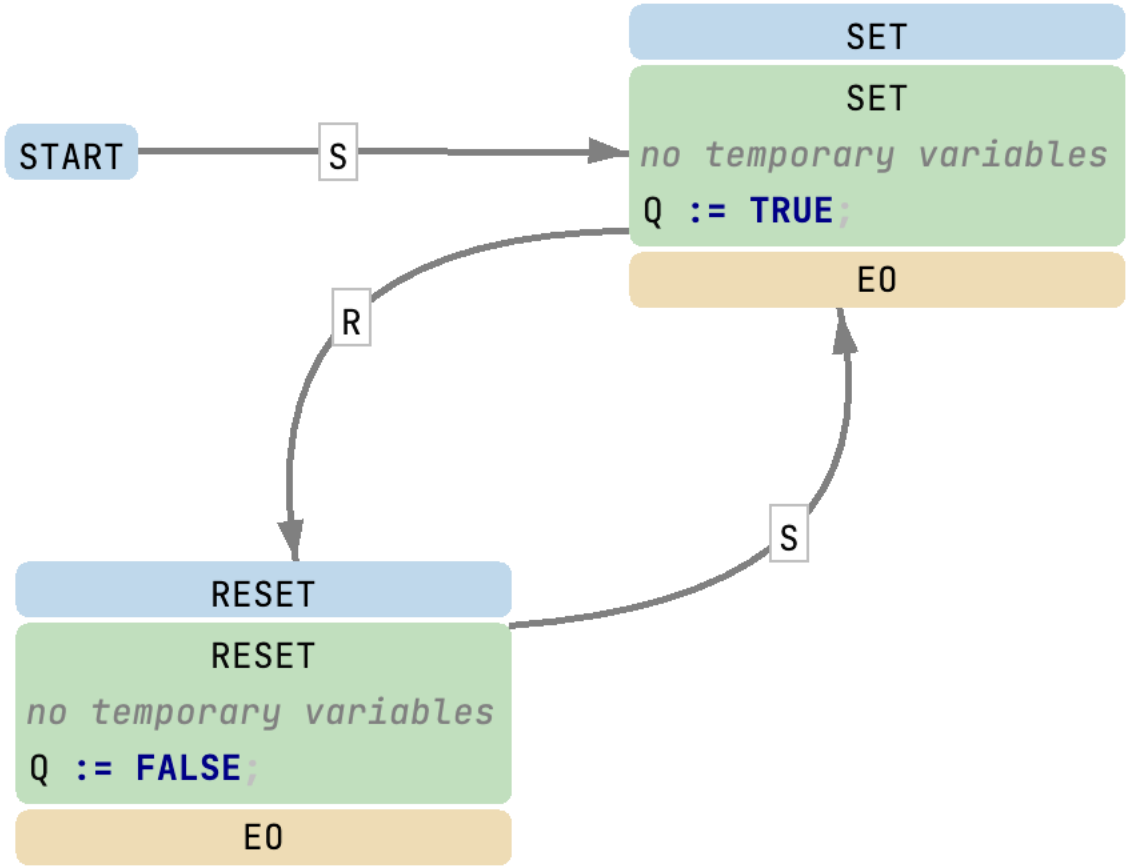




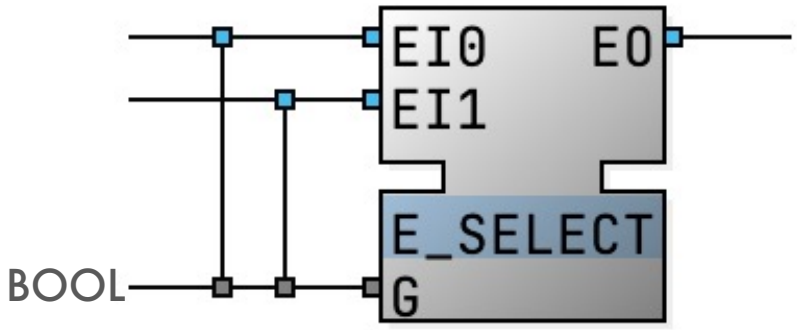


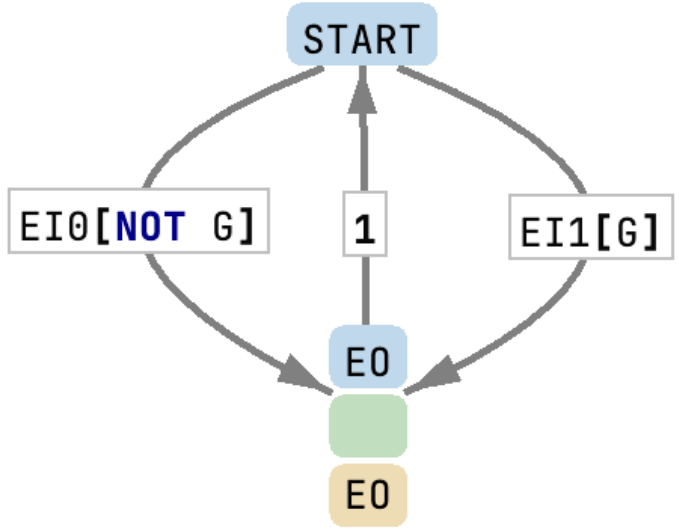
RS, SR triggers



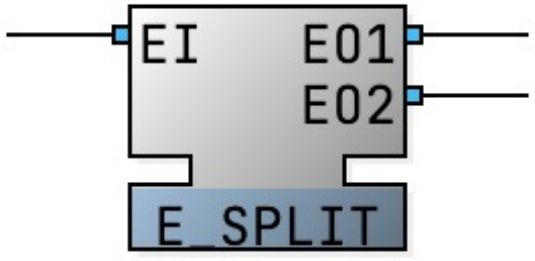


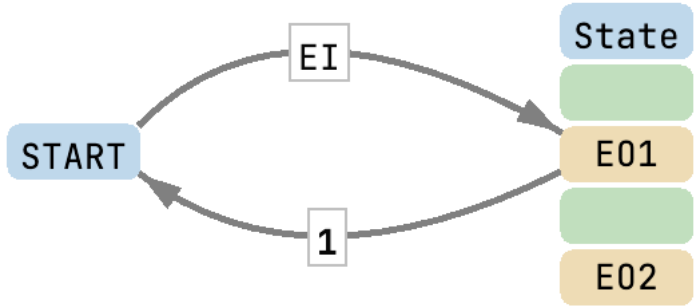
Event selector



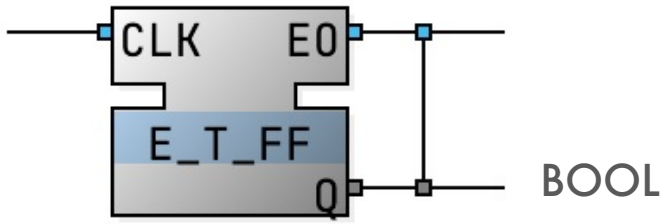


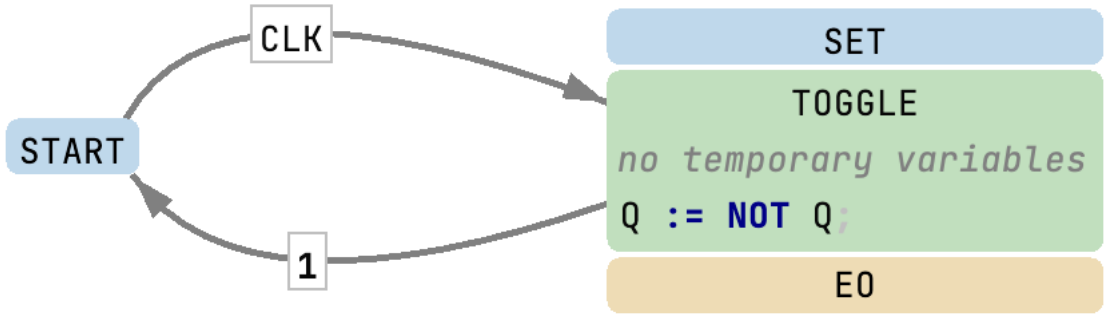
Event splitter





Event driven toggle of a Boolean output





UNIVERSAL

AUTOMATION.ORG