

UNIVERSAL
AUTOMATION.ORG

IEC 61499: primer course

Module 3: IEC 61499 standard

Valeriy Vyatkin

Luleå University of Technology and Aalto University



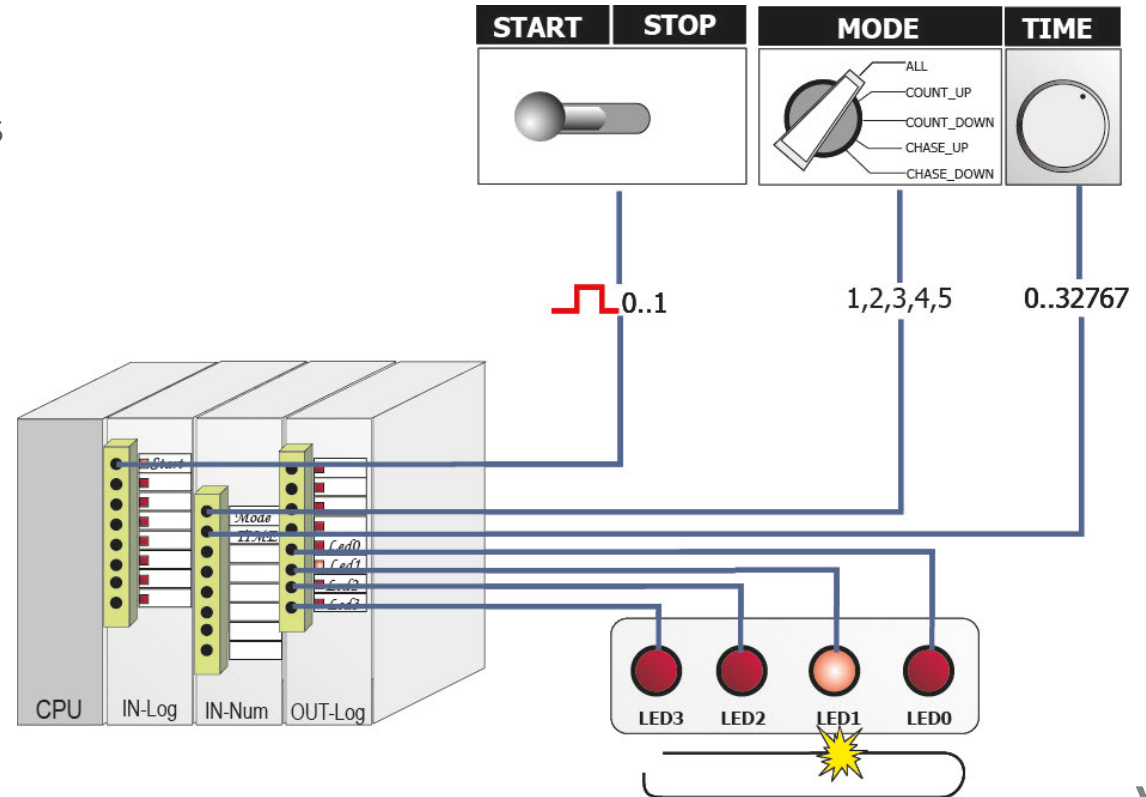
Example of a simple system **FLASHER**

Example: FLASHER

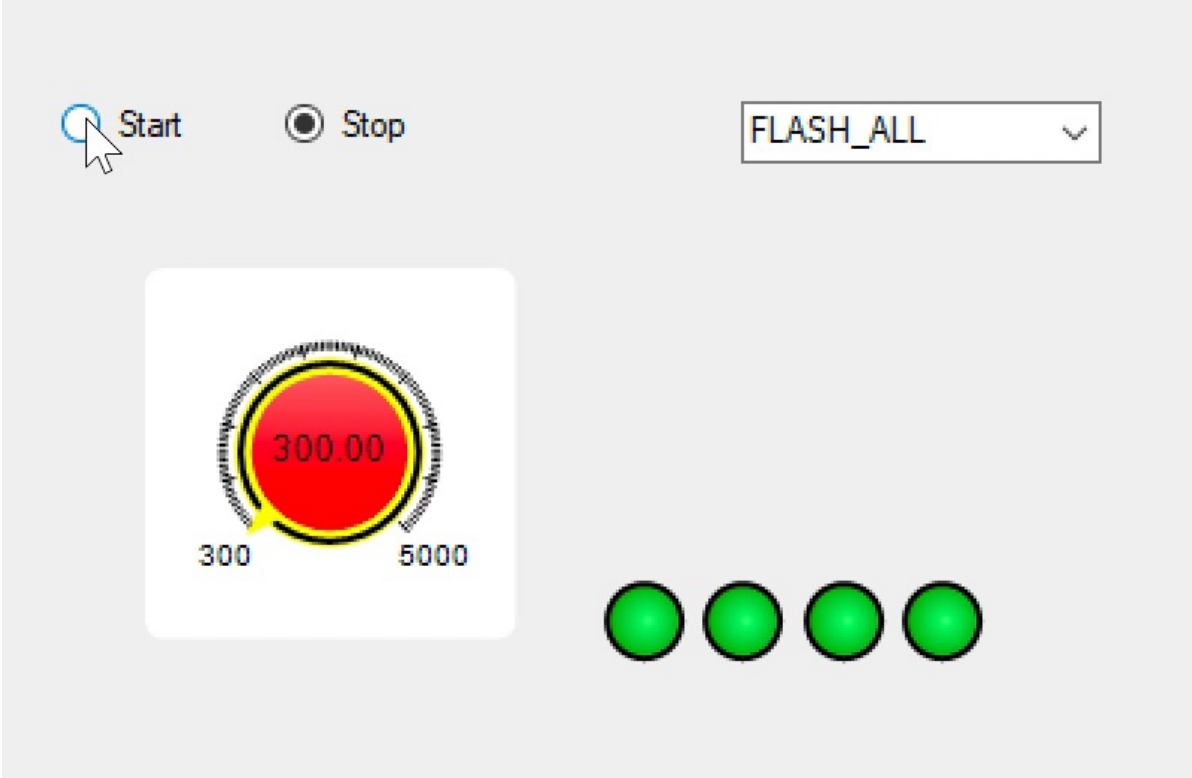
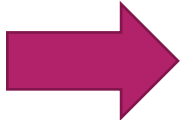
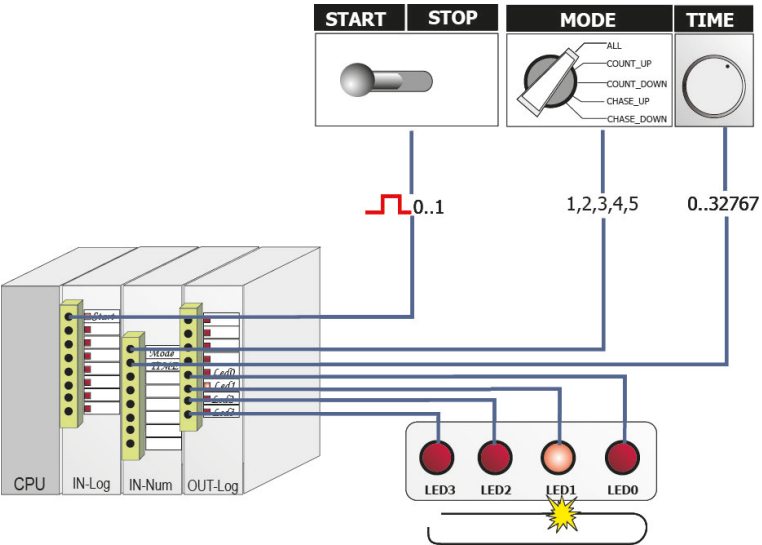
Consider a simple product Flashing Lights

Components of the physical system:

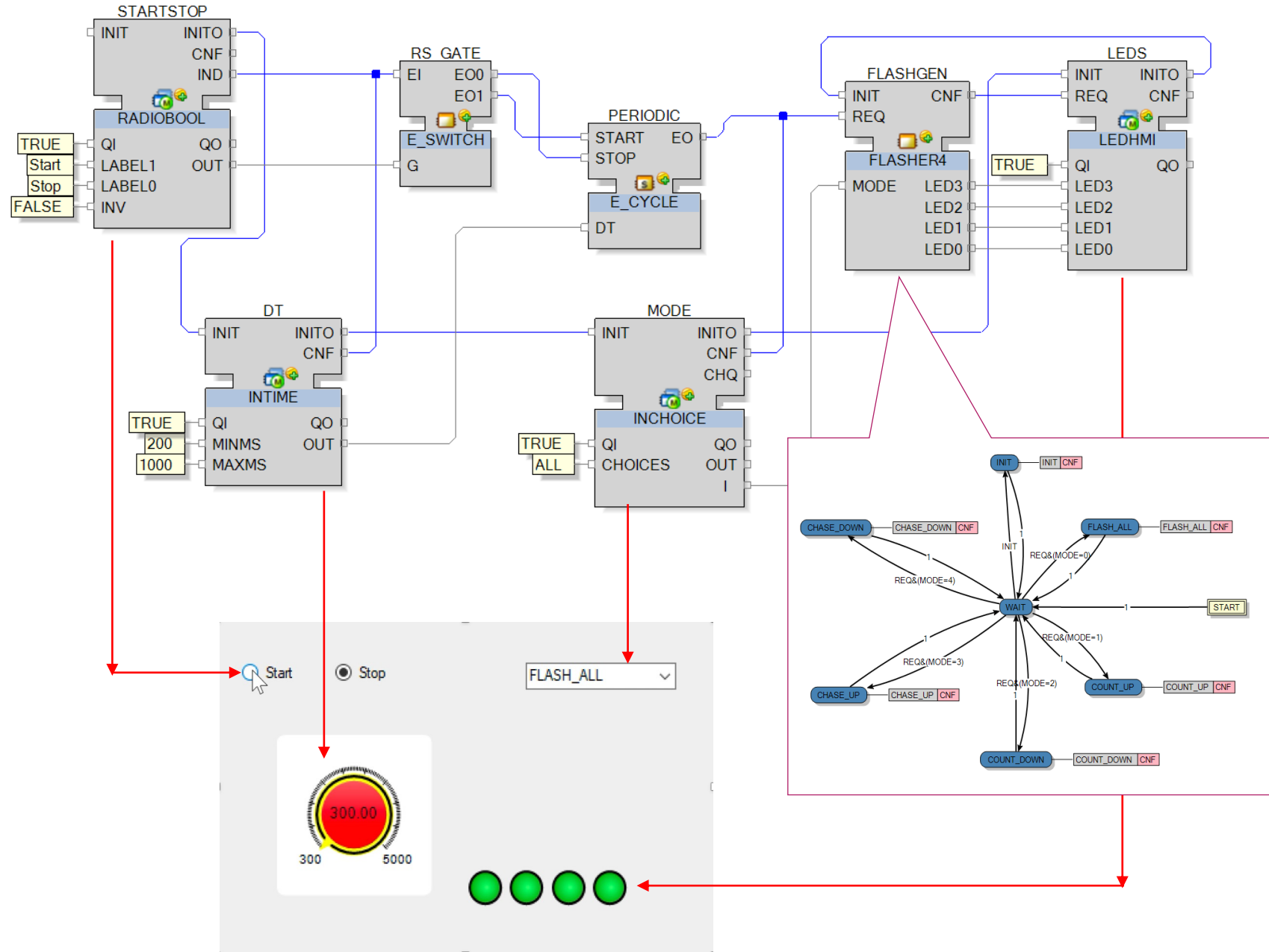
1. PLC (CPU)
2. START/STOP Button
3. Mode Selection Switch
 - All lights on, count up, count down, chase up and chase down
4. Time Delay Nub
 - Delay between 2 consecutive light flashes
5. 4 LEDs



FLASHER modelled in software



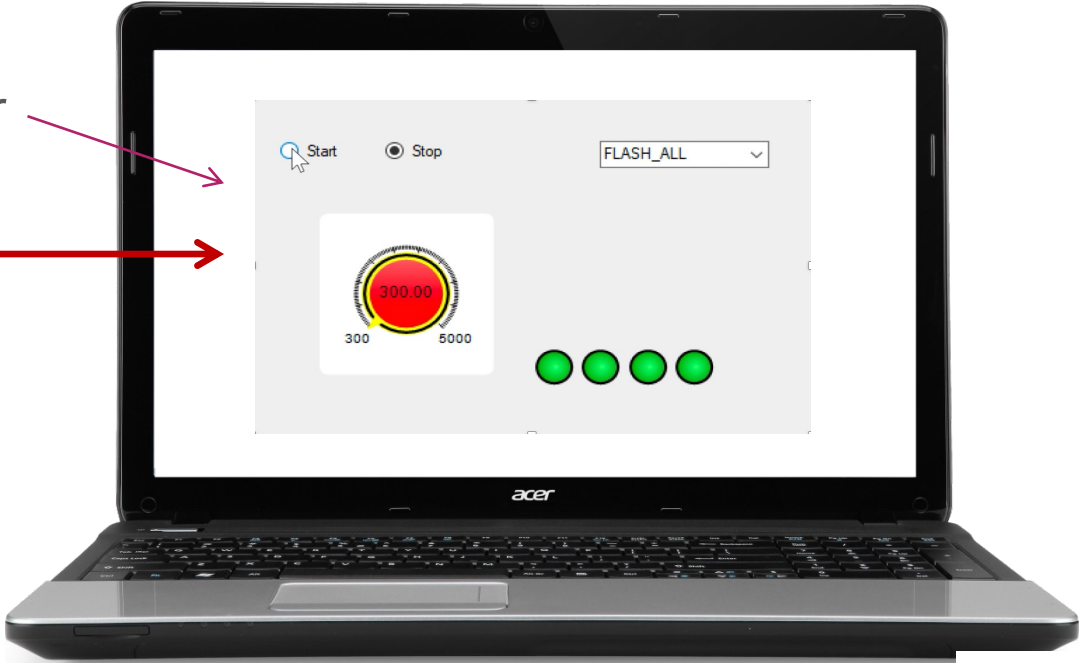
FLASHER application



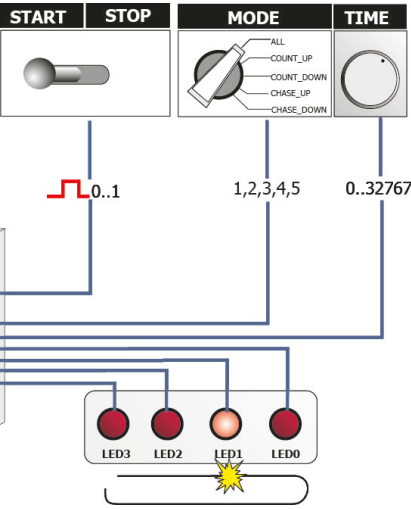
The core functionality is captured in the FLASHER4 FB in a form of state machine

Model and simulate control logic with IEC 61499

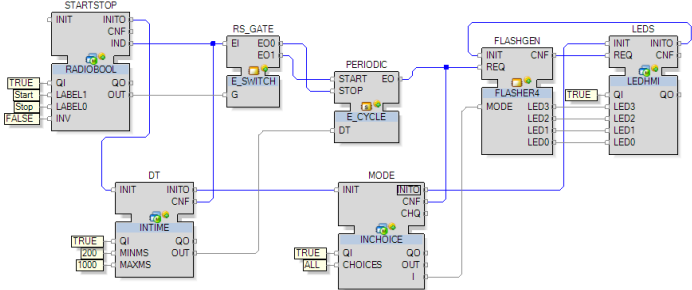
HMI
Runs on a computer



Physical system

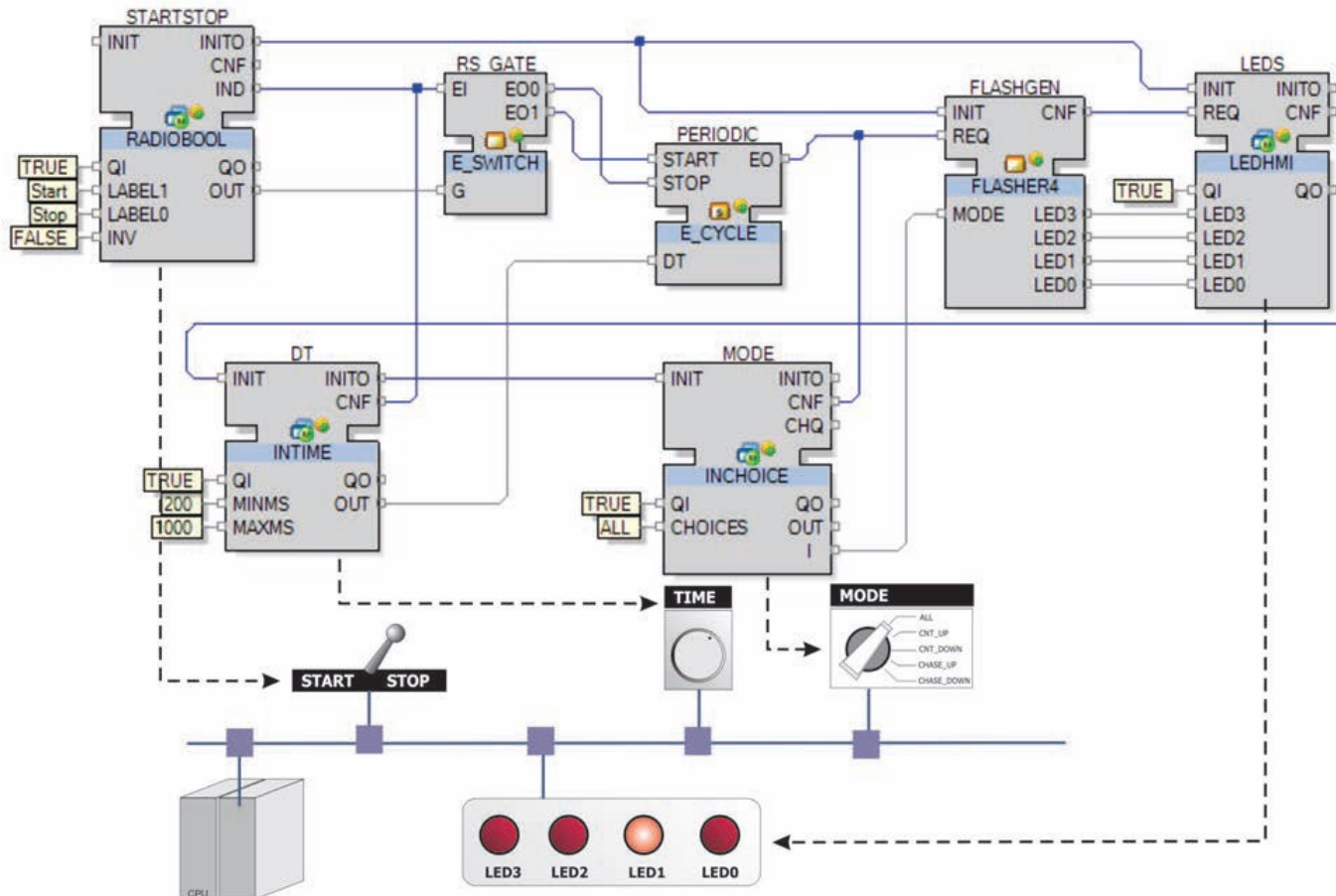


IEC 61499 executable model: Runs on PLC or PC (Soft PLC)

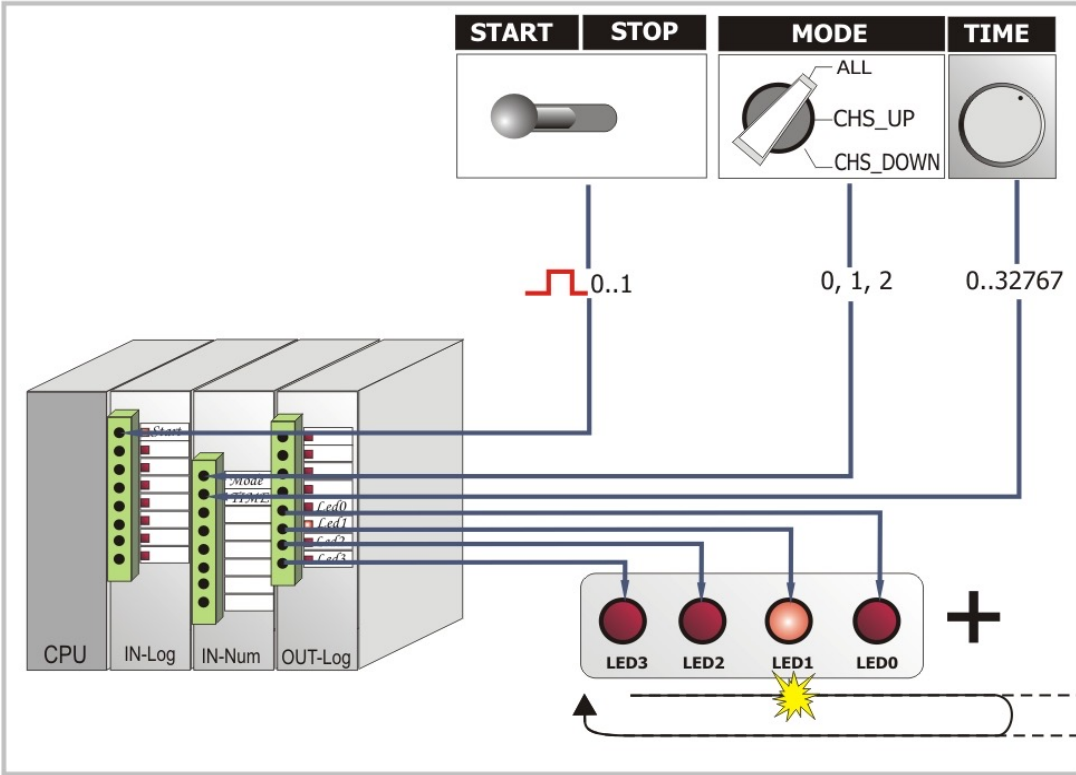


Distributed FLASHER

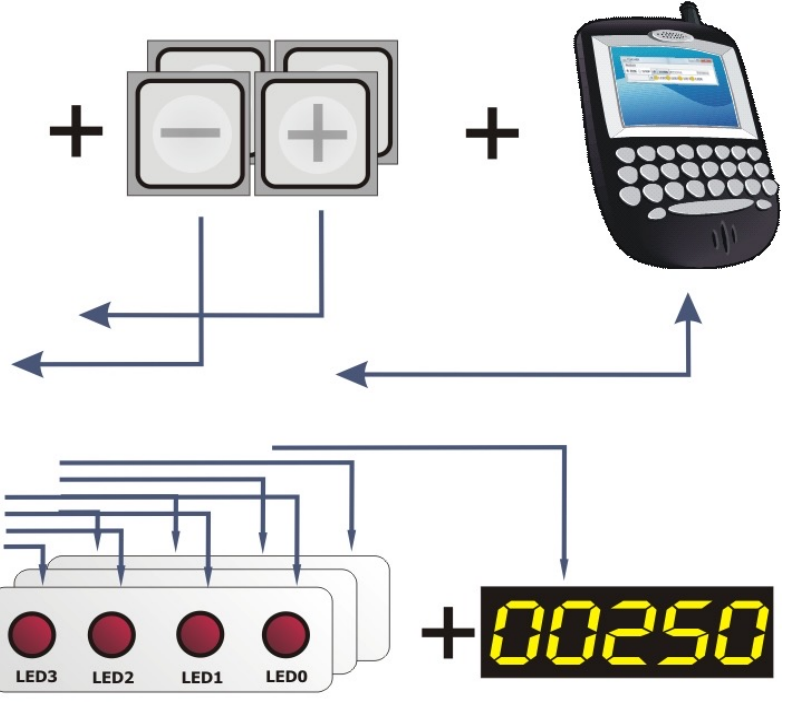
- It is beneficial to model the functionalities in a hardware independent way.
- Same block diagram can be executed on different types of controllers or even on a distributed hardware architecture (running on a network of PLCs).



Motivation: Extensibility

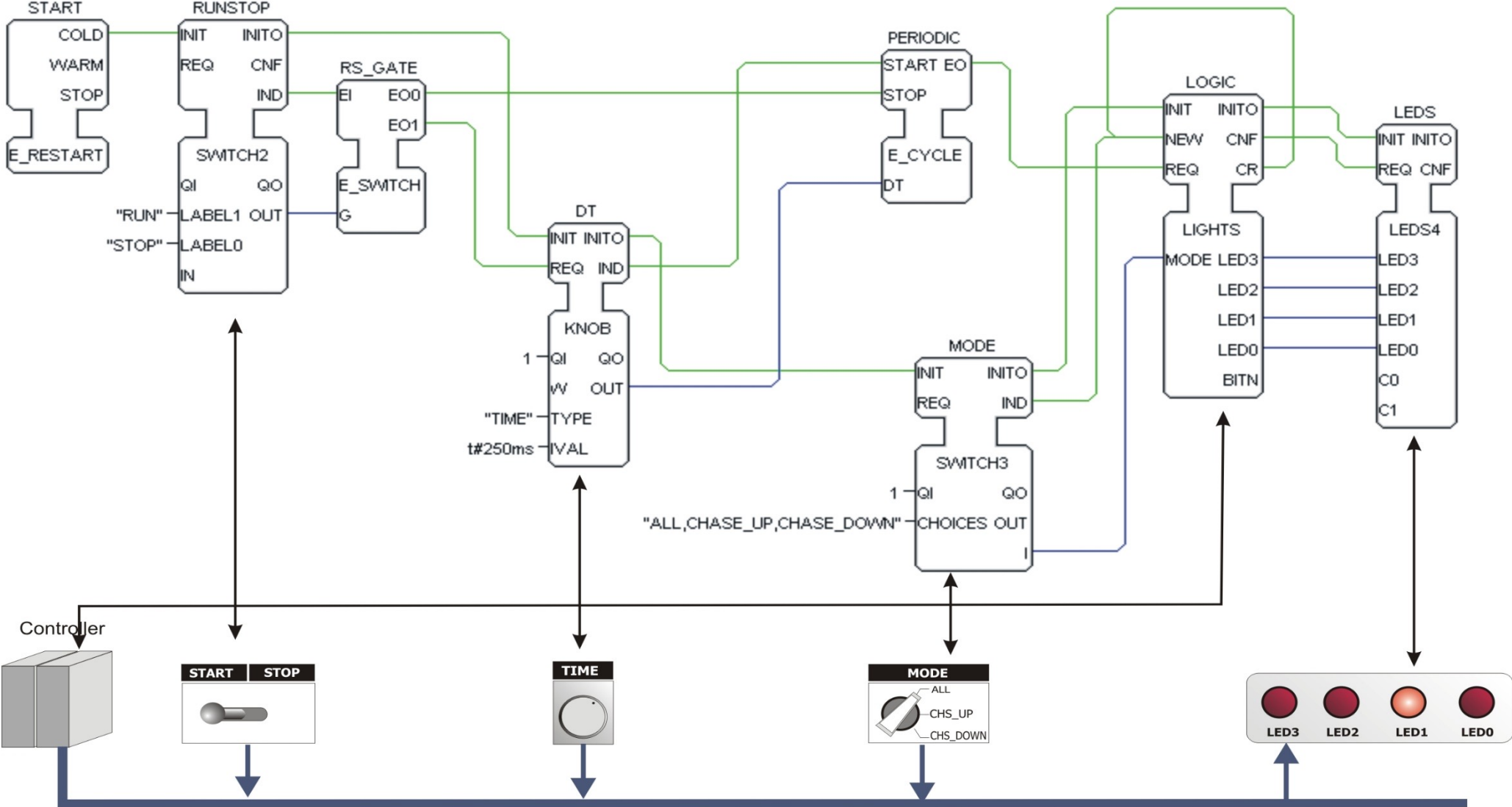


Old system

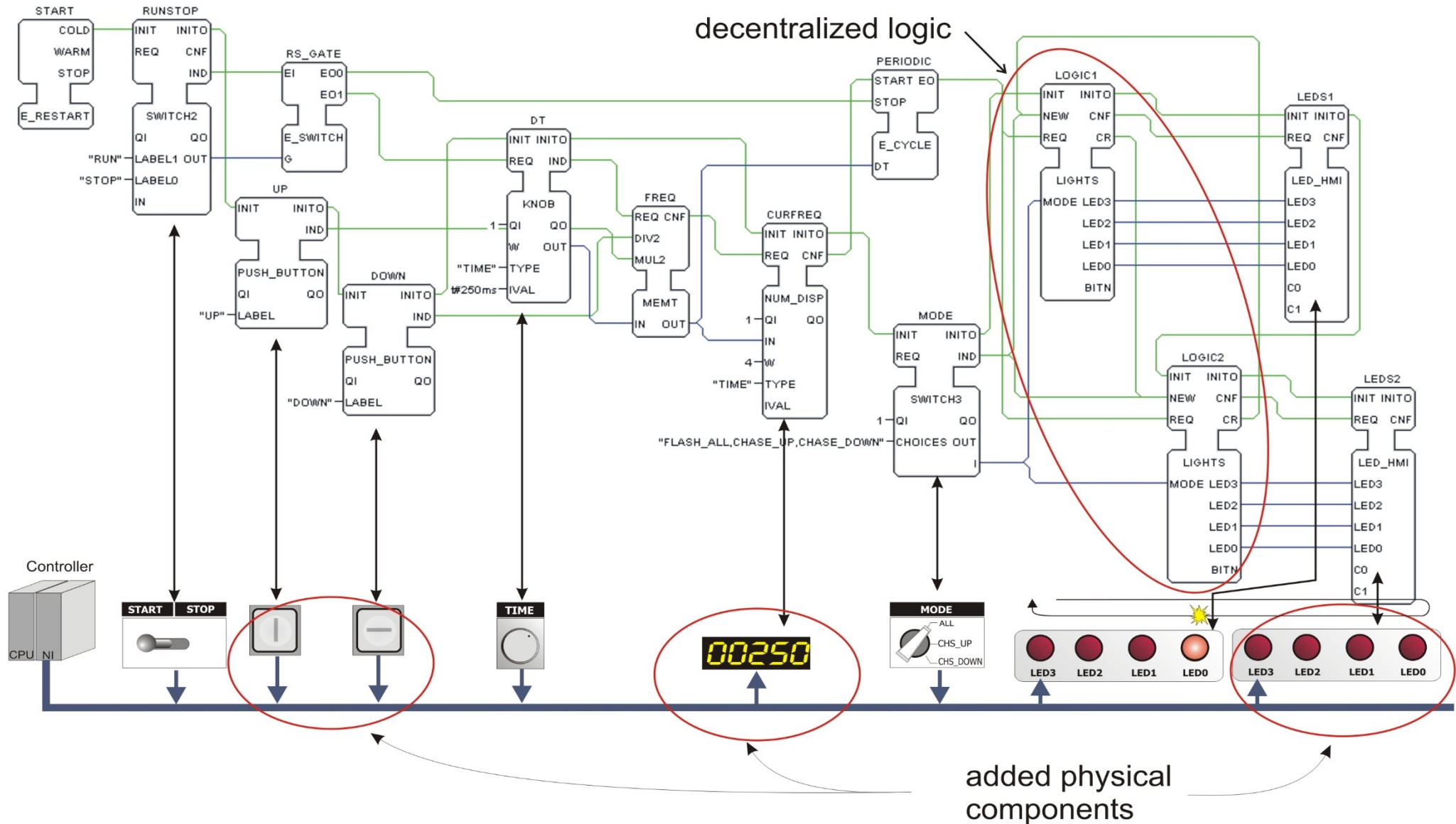


Extensions

Distributed deployment



Extended system

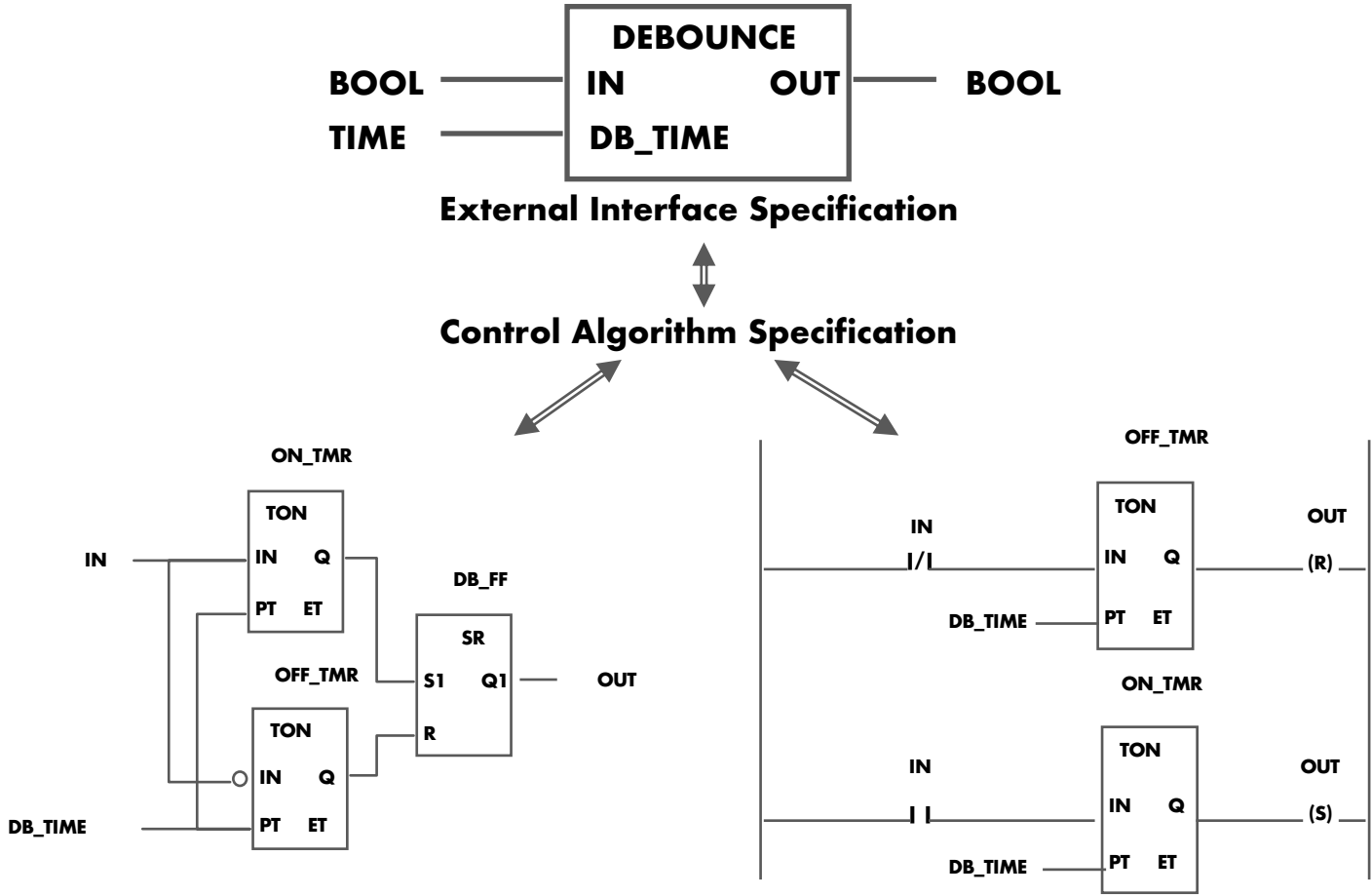


Common elements

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

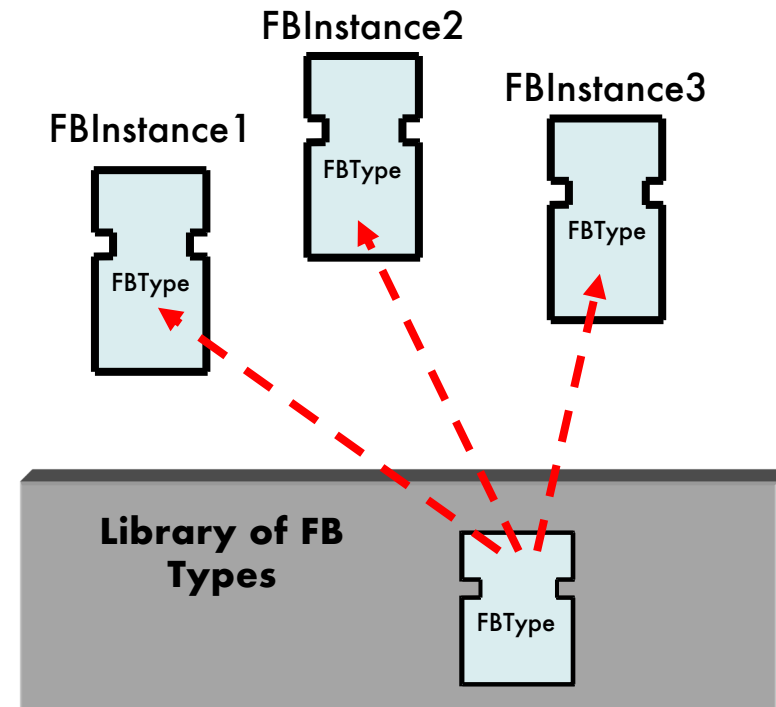


IEC 61131-3 and 61499 Data Types

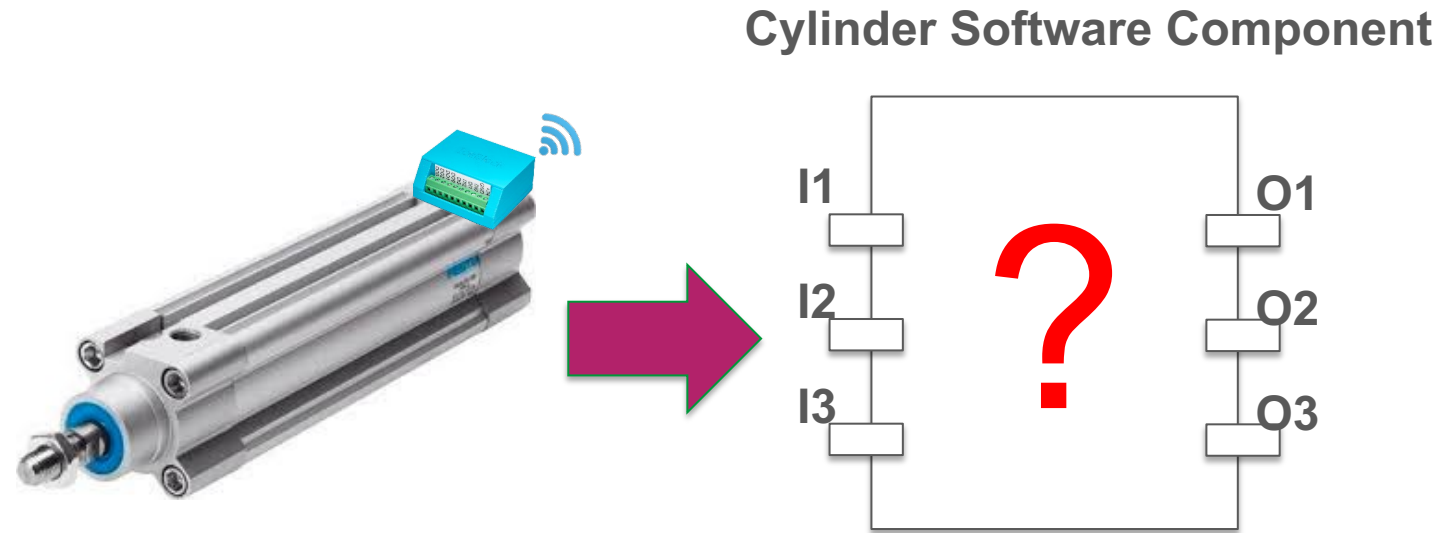
- ❑ Signed Integers: SINT(8), INT(16), DINT(32), LINT(64)
- ❑ Unsigned Integers: USINT(8), UINT(16), UDINT(32), ULINT(64)
- ❑ Floating Point: REAL(32), LREAL(64)
- ❑ Bit Strings: BOOL(1), BYTE(8), WORD(16), DWORD(32), LWORD(64) (TRUE, FALSE, 1, 0, 255, 16#FF, etc.)
- ❑ Character Strings: STRING(8)
- ❑ 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

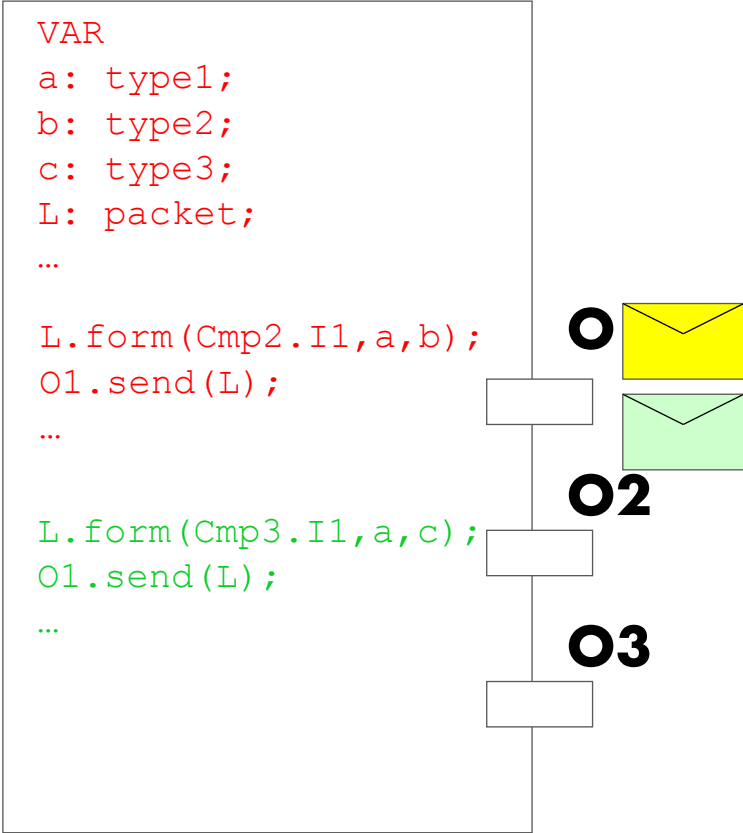


Motivation: Intelligent Automation Component



Communicating components

Cmp1



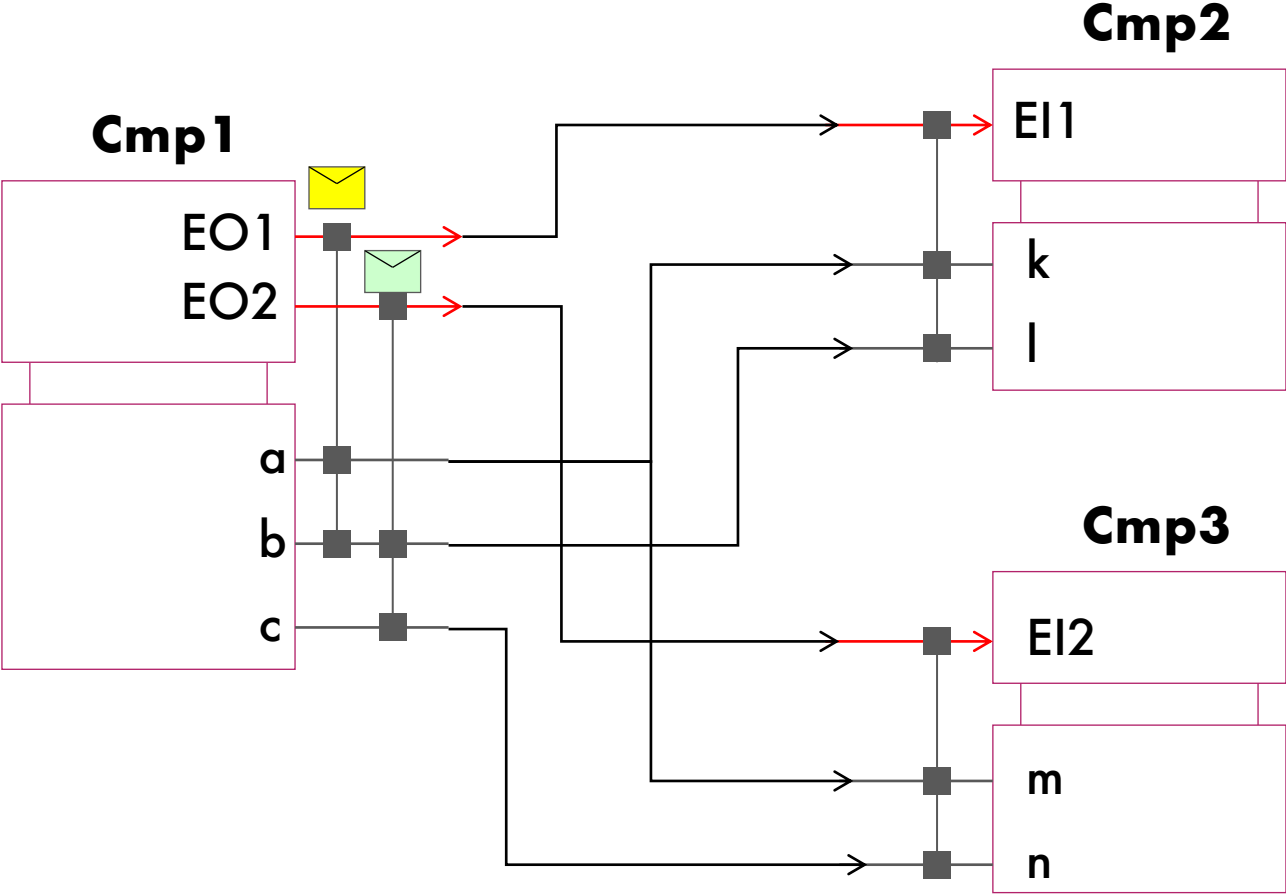
Cmp2



Cmp3



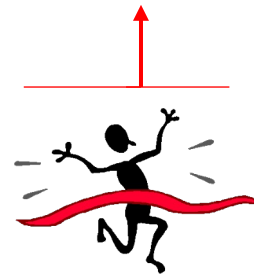
Message=Event + associated Data



Events

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

Event of crossing the line



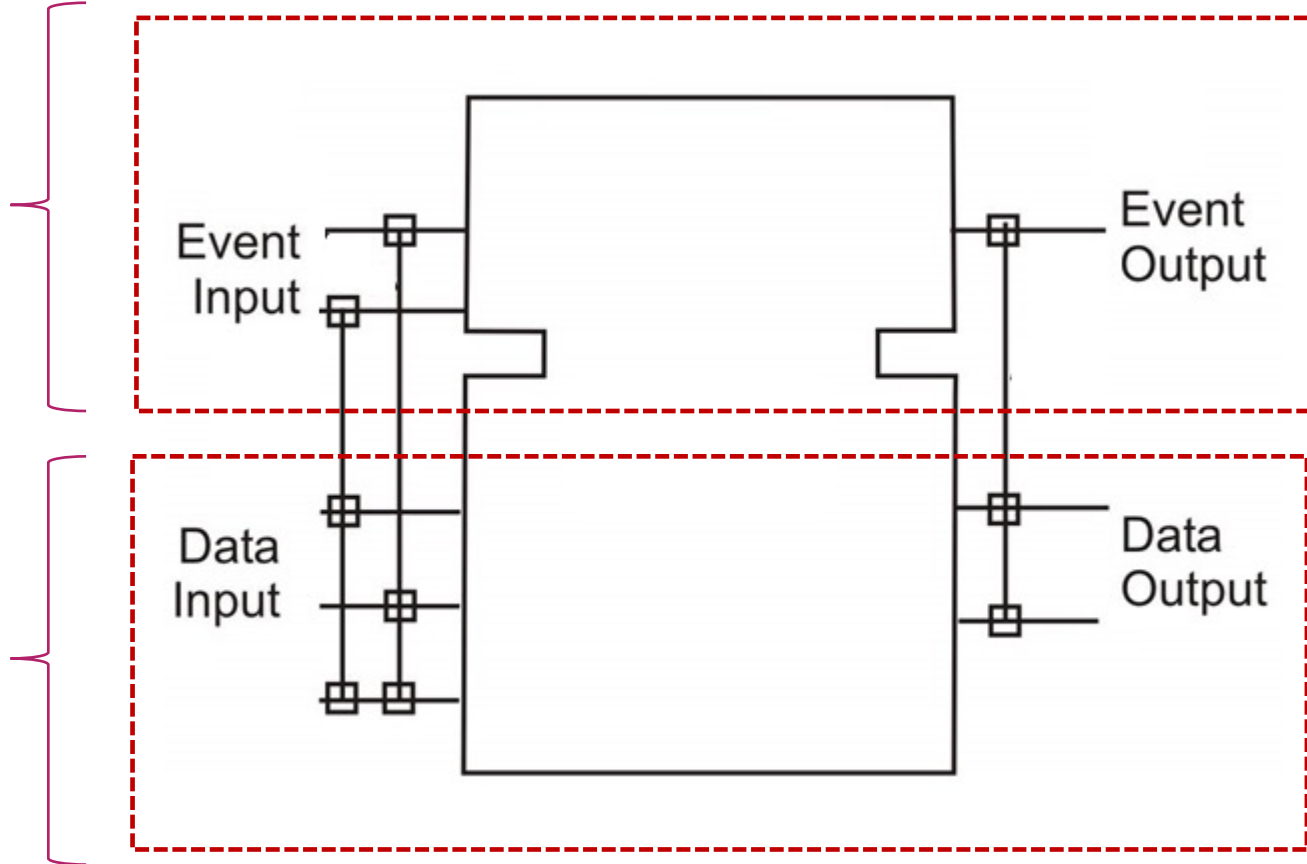
- IEC 61499 function block can only be activated by an event



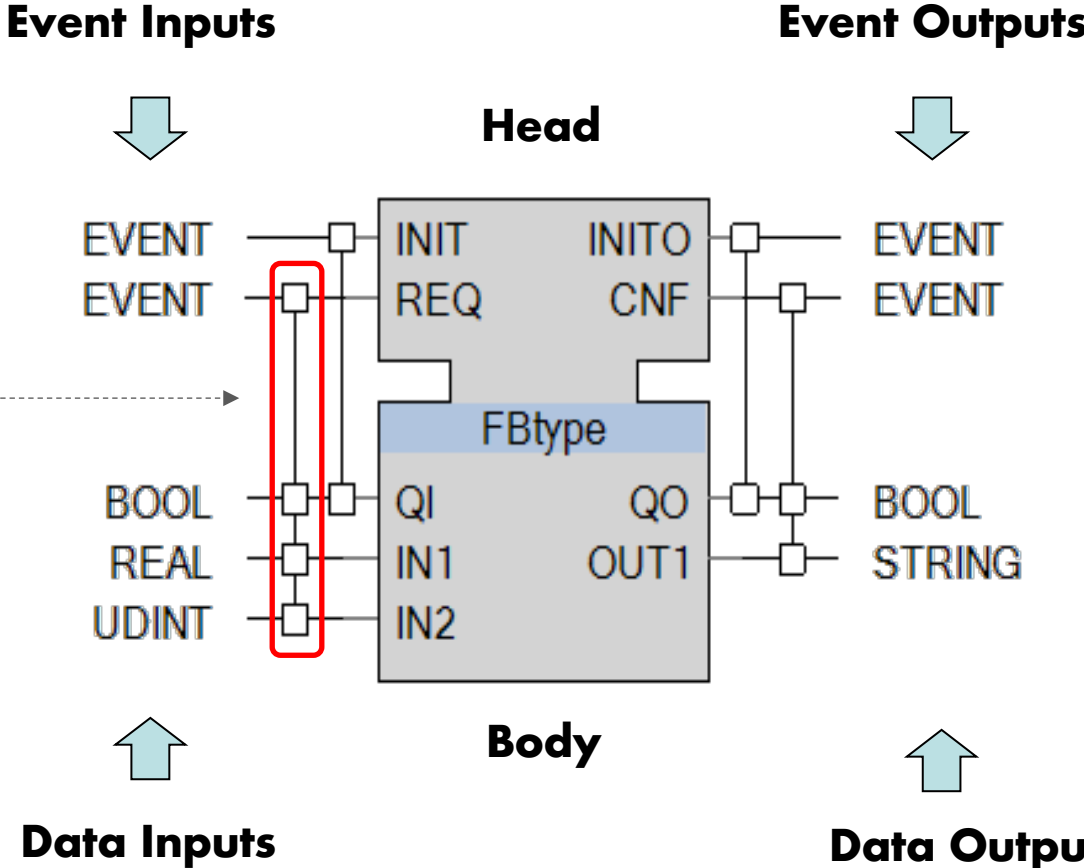
Function Block: Header and Body

Upper part of FB is called **Header**

Lower part of FB is called **Body**



Function Block of IEC 61499: Interface

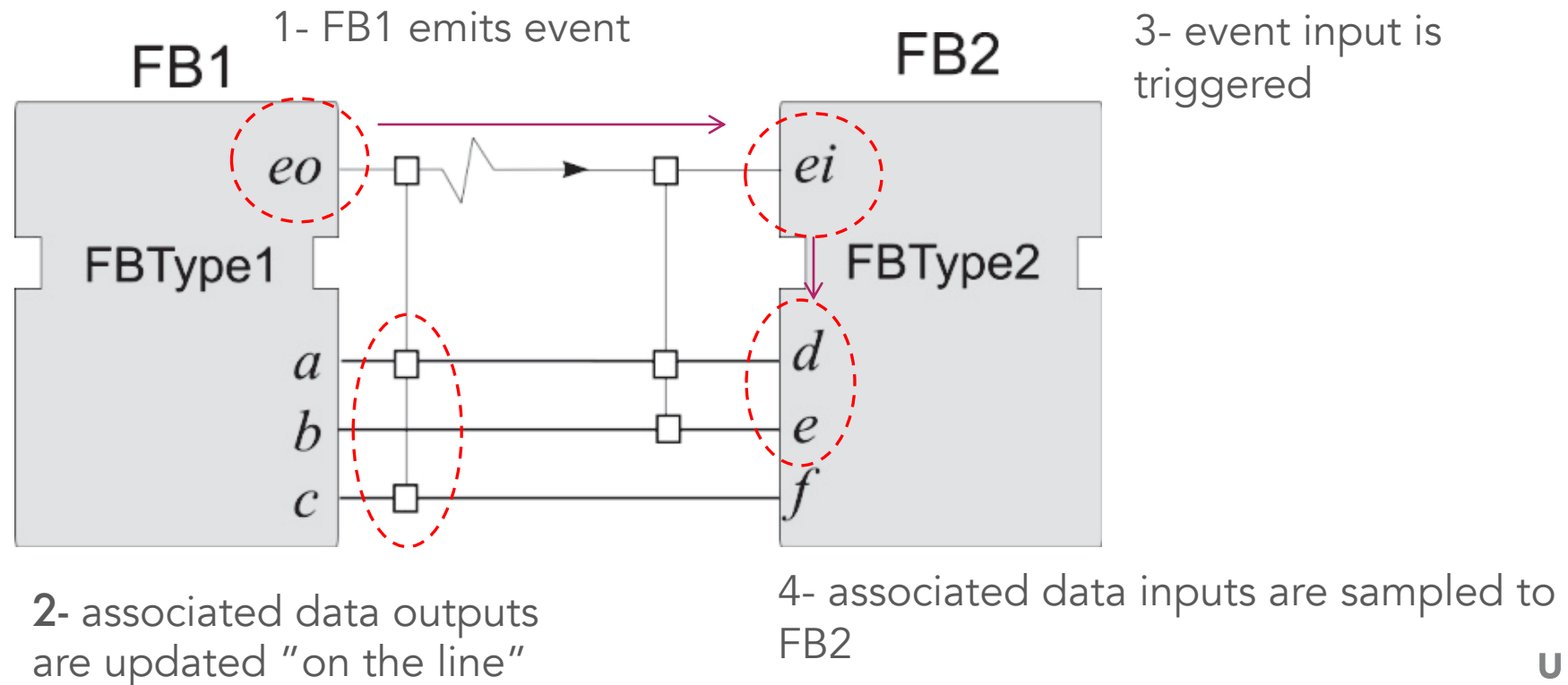


Event - data association: if the event occurs, only the associated data will be updated.

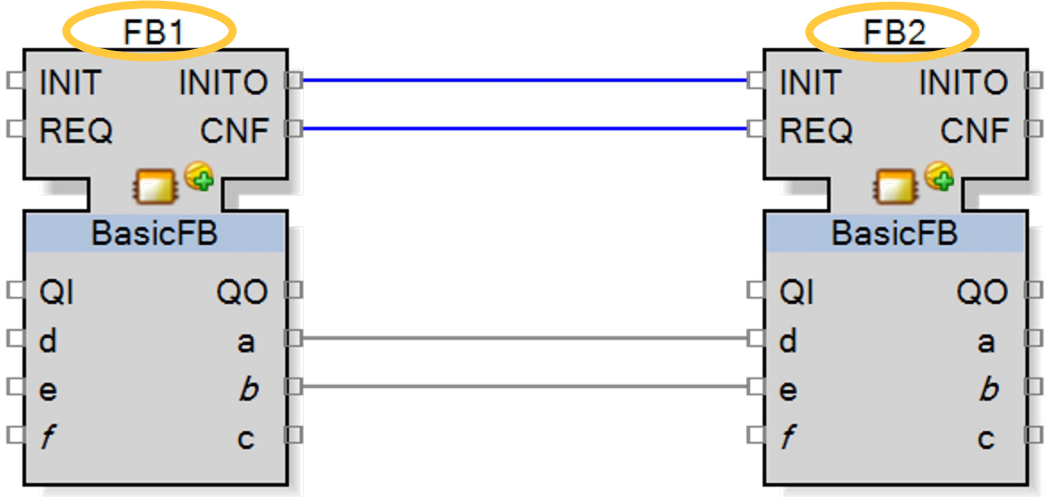
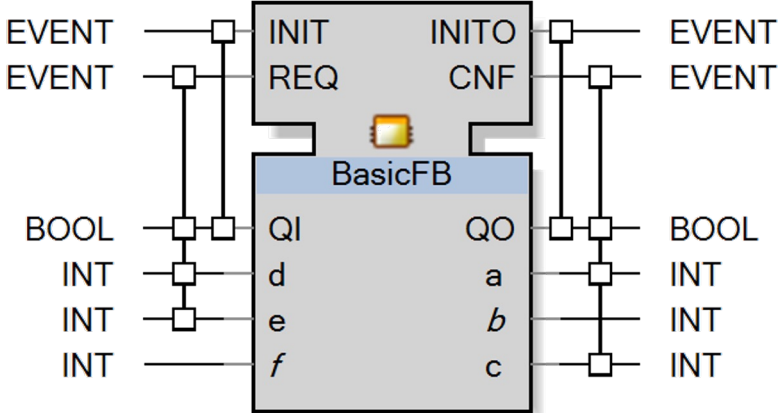
Function block: Event-Data Association

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".

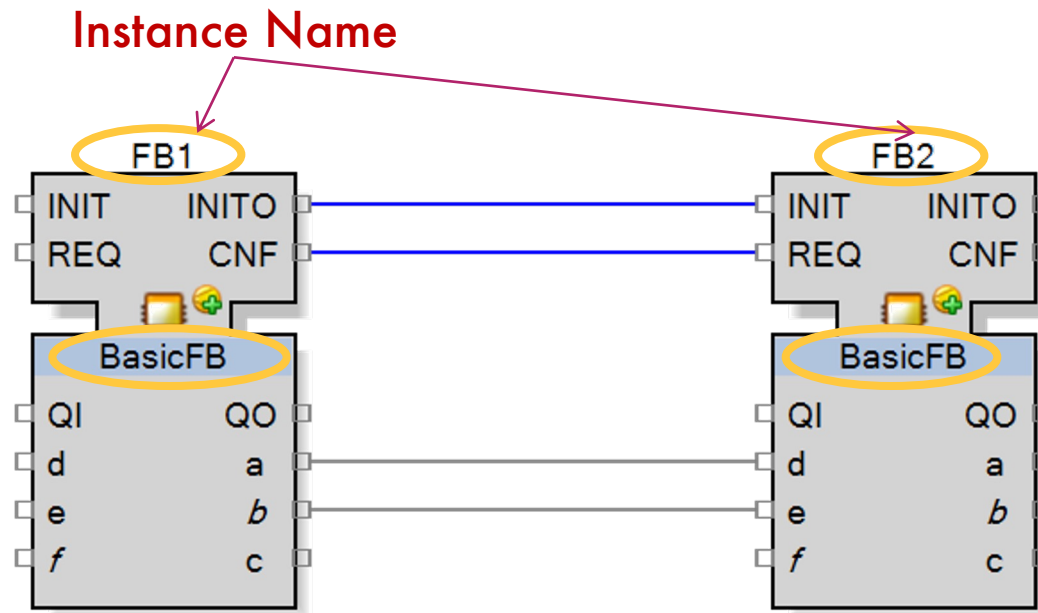
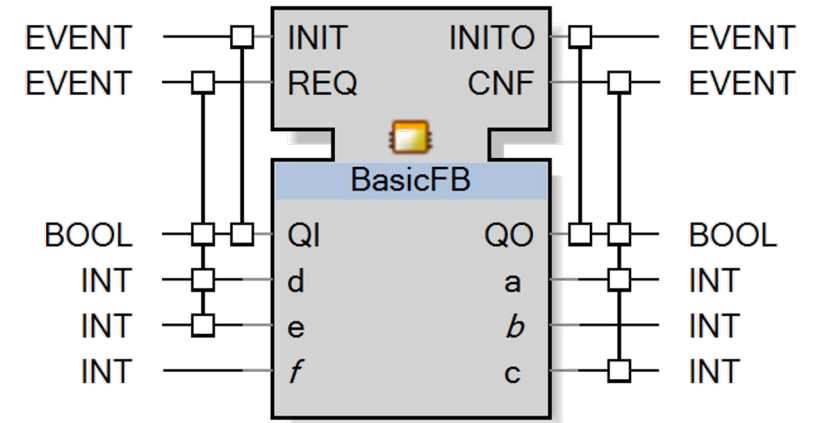


Event-data association



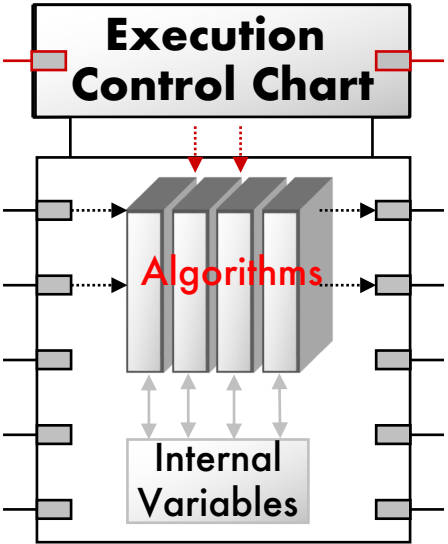
Event-data association

- 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.

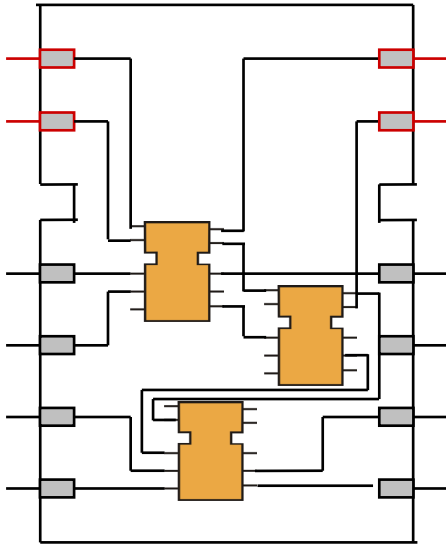


IEC 61499 Function Block Kinds

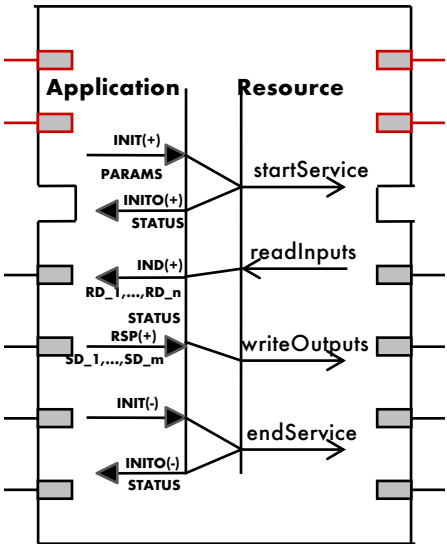
- Basic, Composite, and Service Interface Function Blocks



BFB

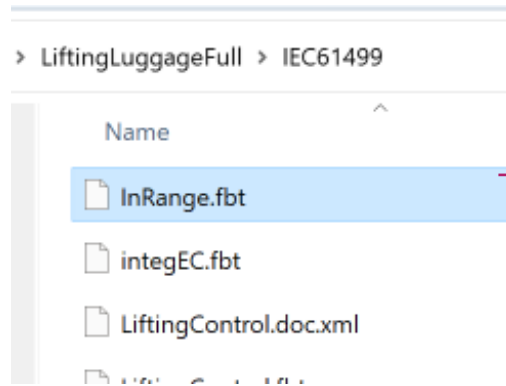


CFB



SIFB

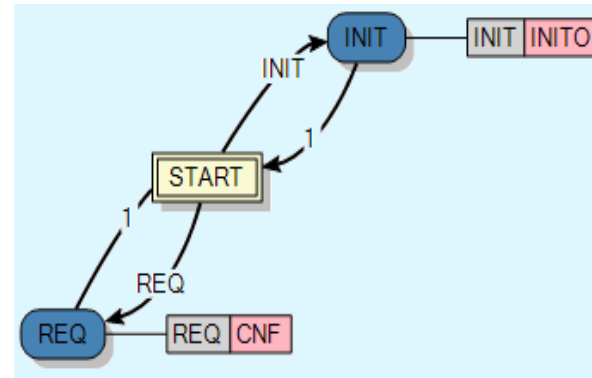
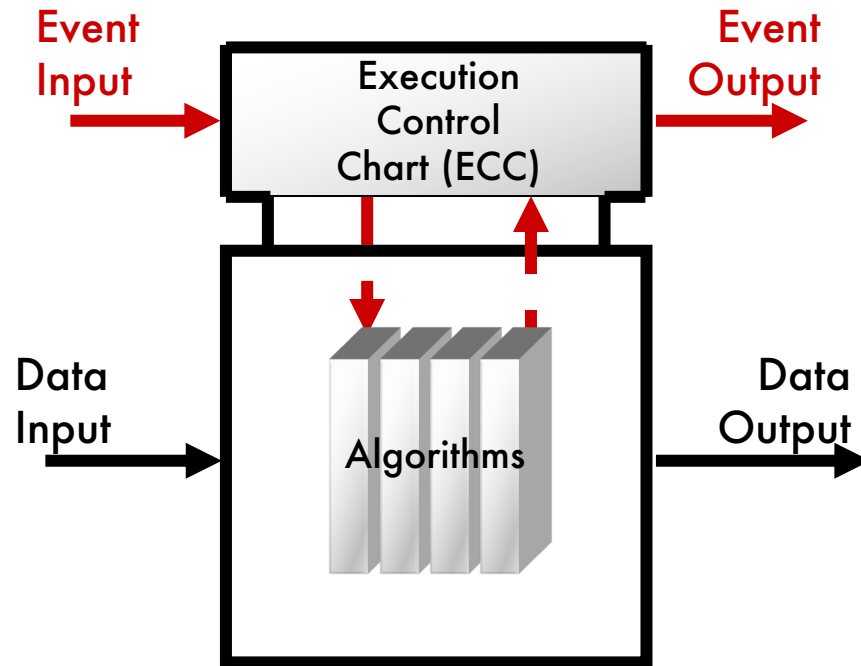
XML Exchange Format



```
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" />
    </InputVars>
  </InterfaceList>
</FBType>
```

Basic Function Block

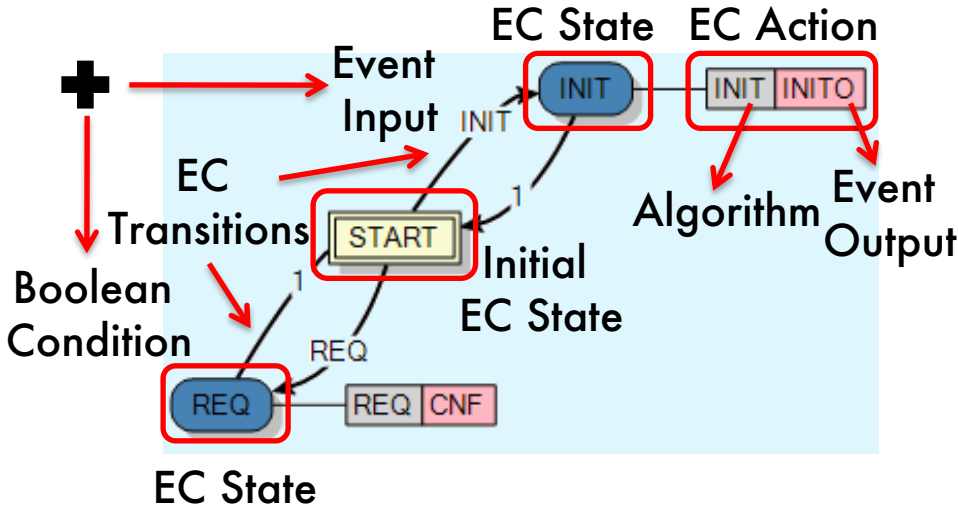
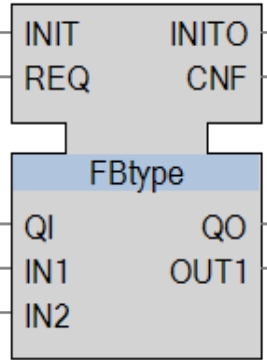
Basic Function Blocks



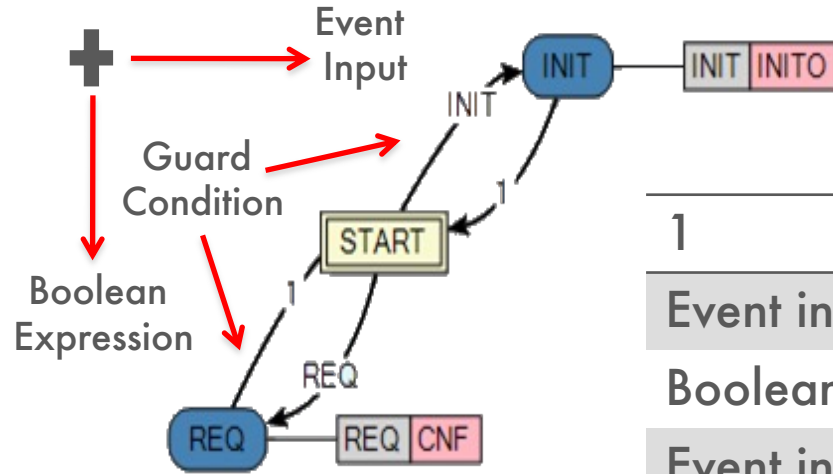
Algorithms can be programmed in PLC languages

- 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

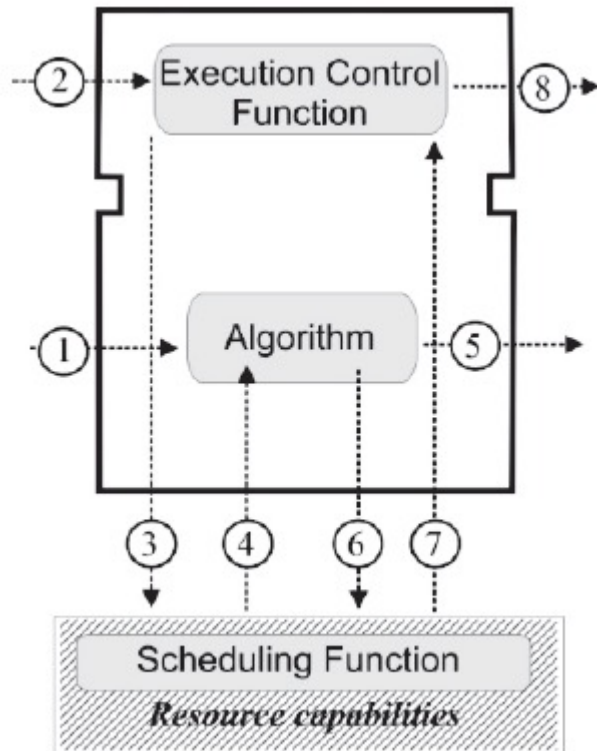
Event input {&}[Boolean expression over data]

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.

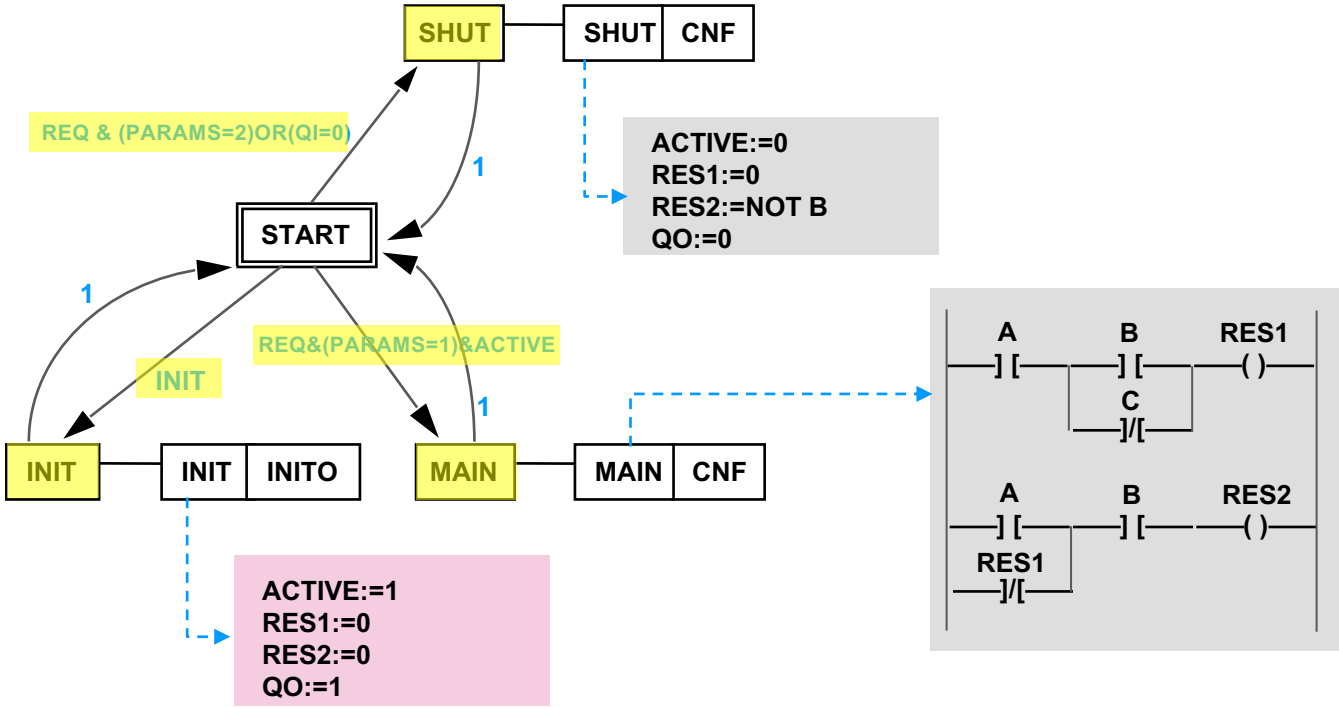
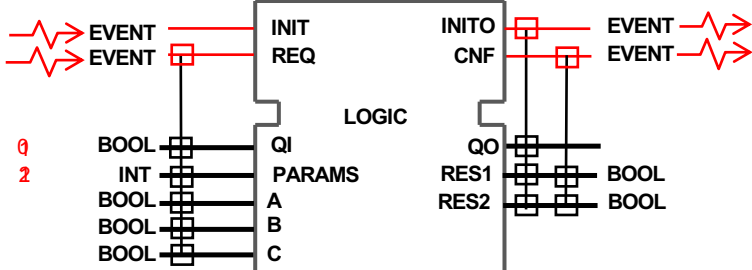
Step 5. The algorithm completes the assignment of values for the output variables associated with the event output.

Step 6. The resource scheduling function is notified that algorithm execution has ended.

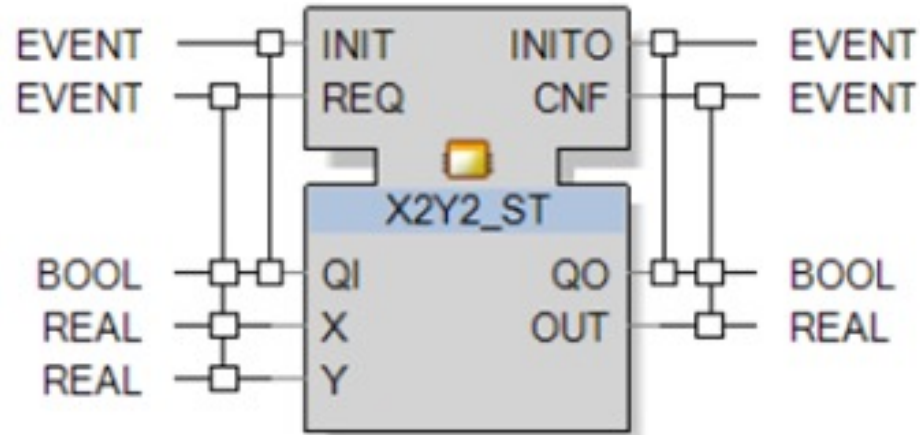
Step 7. The scheduling function invokes the execution control function.

Step 8. The execution control function signals event at the event output.

How does Basic Function Block work?

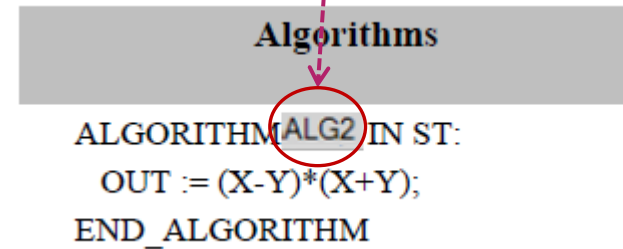
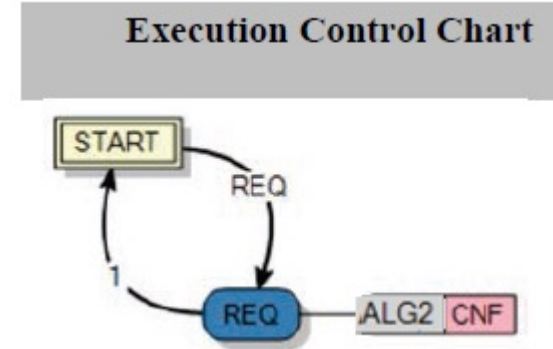


Basic FB – execution control chart (ECC)



Each state of ECC can have one or more actions.

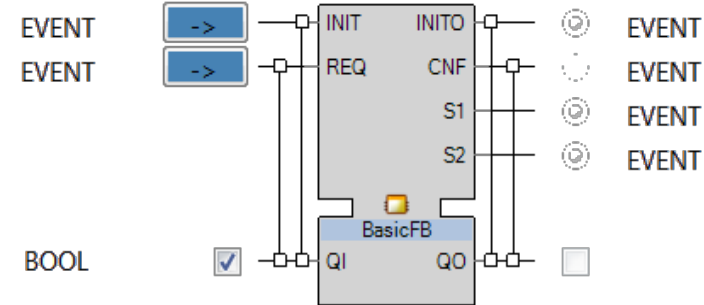
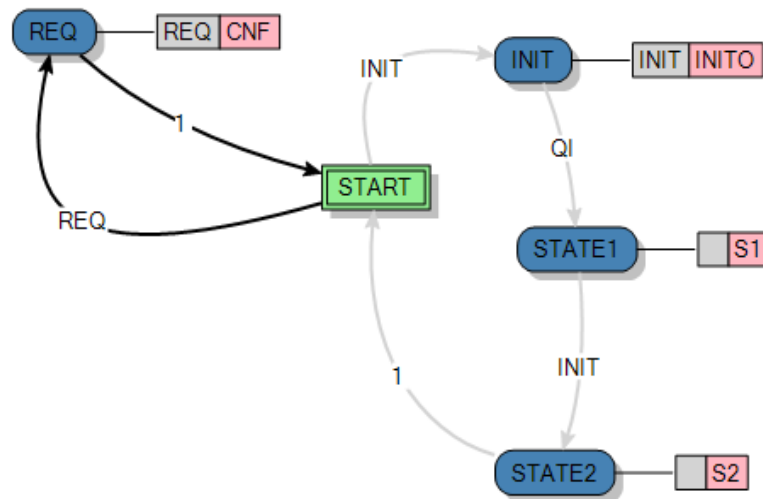
An action may have an algorithm call and an output event emission, or both.



Lifetime of Event Input

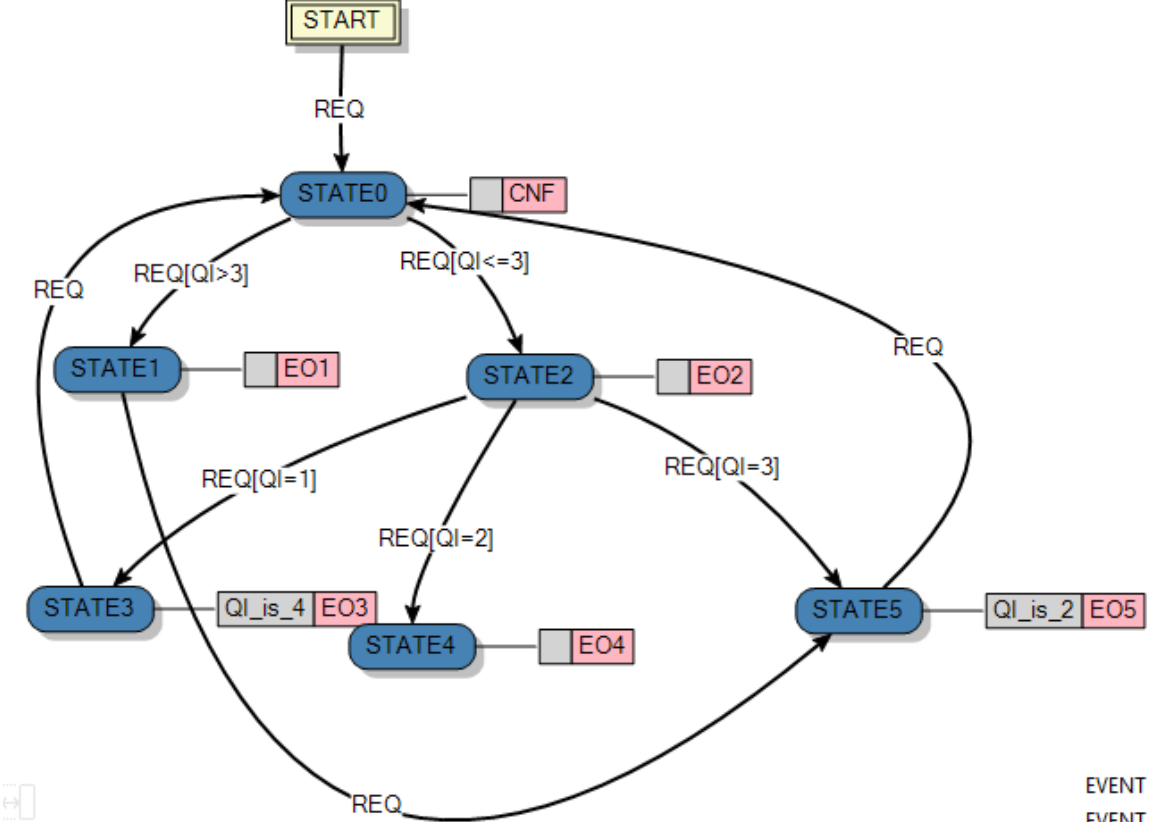
EAE stops ECC execution when:

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

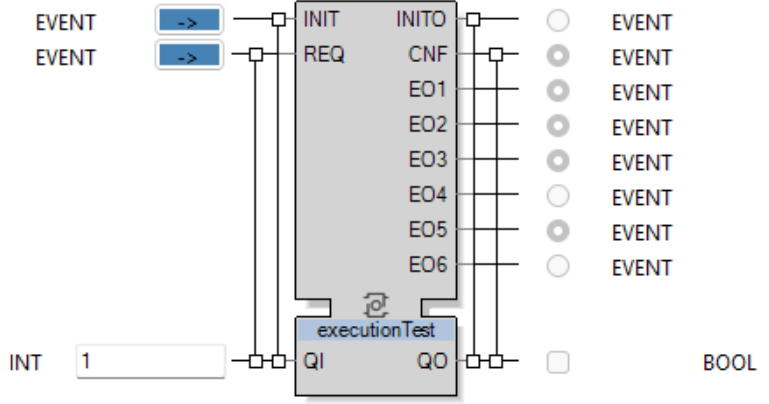


Trick: assign an algorithm, ALG1, to STATE1, within which set INIT:=False. This workaround is not portable.

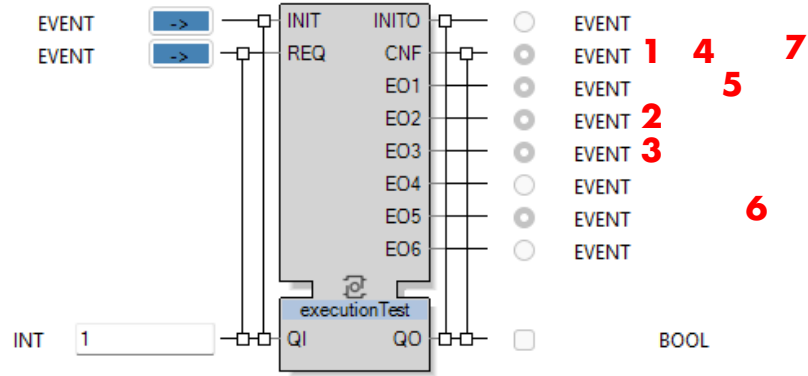
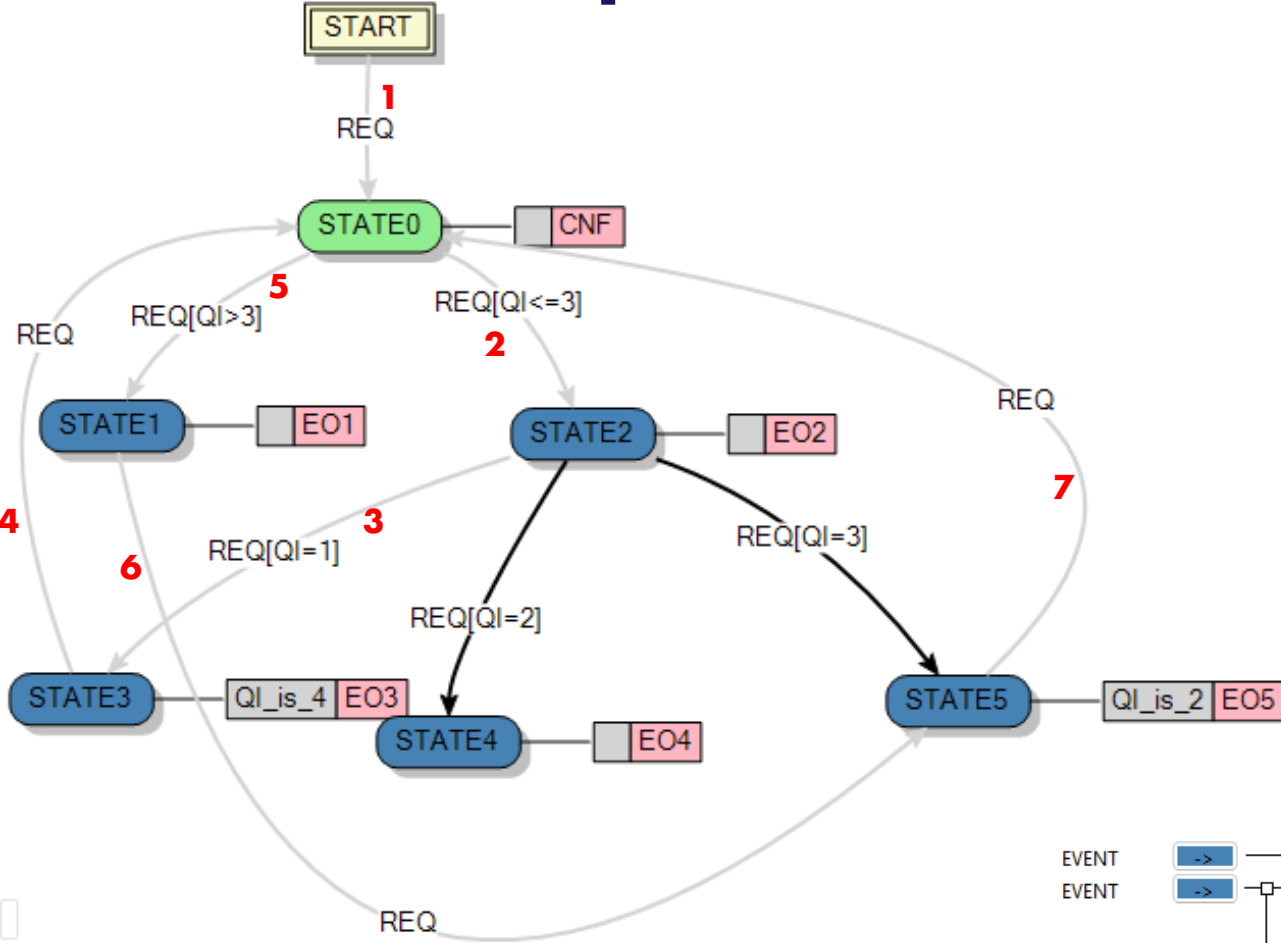
Lifetime of Event Input



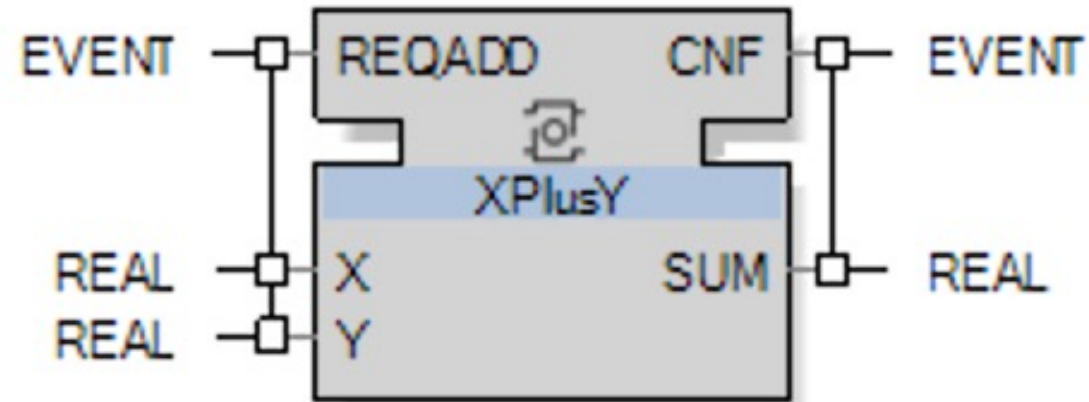
- The first transition with a true condition is executed and the corresponding ECC status of the block is occupied.
- The recently used transition is marked so that it cannot be used a second time.
- Once there are no transitions to other ECC states left to be fired, because they are already marked and 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

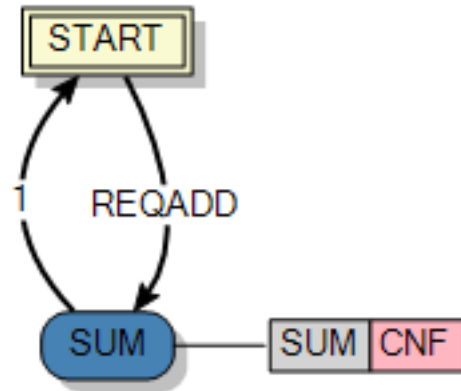
The screenshot displays the EcoStruxure Automation Expert interface. The top menu bar includes File, Home, View, Help, and Debug. The Debug menu is active, showing Run, Stop, Step into, Step over, and Step out options. The Solution Explorer on the left shows a project structure with folders for Libraries (external), Course, References, System, CAT, CAT Instances, SubApp, Composite, Basic, XPlusMinusY, XPlusY, Adapter, Canvases, Graphics, and Data Type. The XPlusY folder is selected, and its contents are shown in the main workspace. The XPlusY folder contains the following items:

Name	Type
EventInputs	
REQADD	
<new interface>	
EventOutputs	
CNF	
<new interface>	
InputVars	
X	REAL
Y	REAL
<new variable>	
OutputVars	
SUM	REAL
<new variable>	
InternalVars	
<new variable>	
Sockets	
<new adapter>	

The Message Log at the bottom shows the status: Build successfully finished. The status bar indicates the current position: ln 5 col 4 ch 4.

On the right side of the screenshot, a diagram shows the XPlusY function block. It has two input ports labeled X and Y, both of type REAL. It has two output ports labeled SUM and CNF, both of type REAL. The block is connected to a REQADD interface block, which is also connected to an EVENT output port.

A Basic FB to Add Two Real Numbers

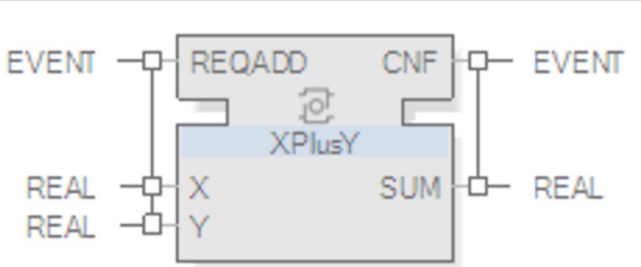


```
SUM
1  = ALGORITHM SUM IN ST:
2  = (* Add your comment (as per IEC 61131-3) here
3  = Normally executed algorithm
4  = *)
5  = SUM:=X+Y;
6  = END_ALGORITHM
```

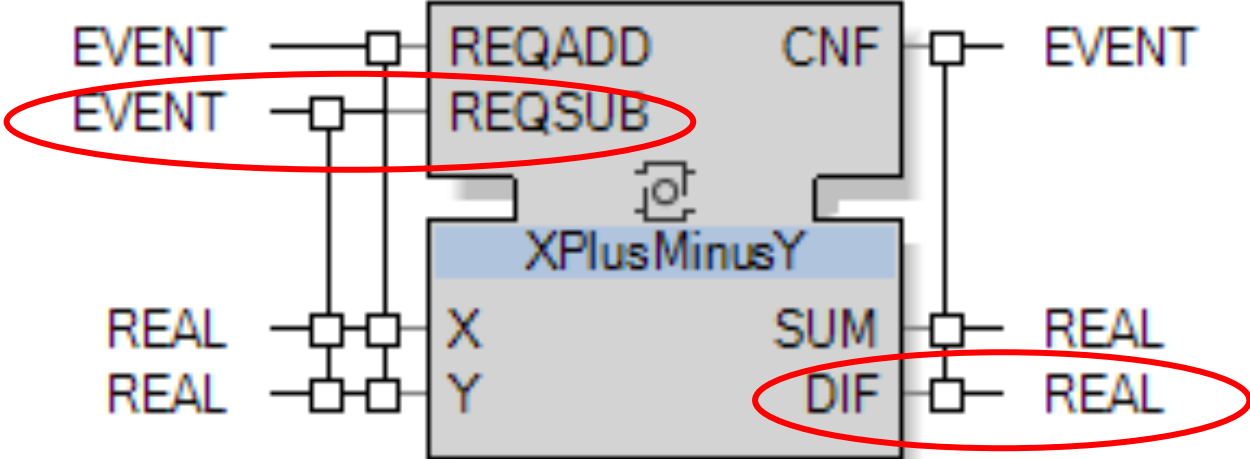
Running the basic FB to Add Two Real Numbers

The screenshot displays the EcoStruxure Automation Expert interface. The top menu bar includes File, Home, View, Help, and Debug. The Debug menu is active, showing options like Continue, Stop, Step into, Step out, and Step over. The Solution Explorer on the left shows a project structure with folders for Libraries (external), Course, References, System, CAT, CAT Instances, SubApp, Composite, Basic, XPlusMinusY, Adapter, Canvases, Graphics, and Data Type. The main workspace shows a ladder logic diagram with a START button connected to a SUM function block via a REQADD contact. The SUM function block is connected to a SUM output contact. The Defaults window on the right shows the configuration for the XPlusY function block, with inputs X (4) and Y (3), and output SUM (7). The Message Log at the bottom shows 0 Errors, 0 Warnings, and 0 Messages. The Output window shows the build status as 'Build successfully finished'.

A Basic FB to Add and Subtract Two Real Numbers



extend

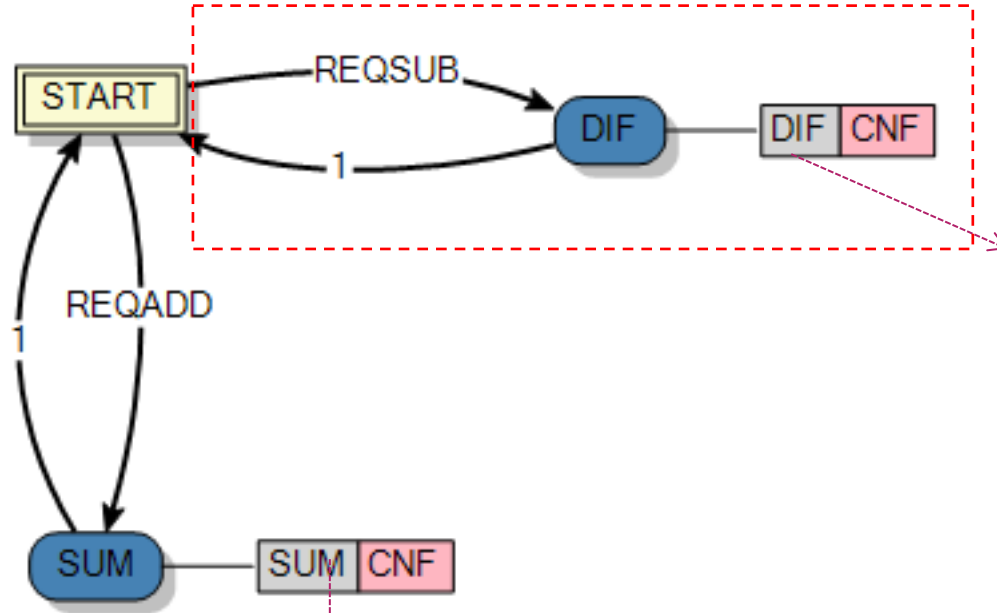


A Basic FB to Add and Subtract Two Real Numbers

The screenshot displays the EcoStruxure Automation Expert software interface. The main window is titled "Test Container" and "Course.sln - EcoStruxure Automation Expert". The interface is divided into several panes:

- Debug Panel:** Located at the top, it includes a menu (File, Home, View, Help, Debug) and a toolbar with buttons for "Continue", "Stop", "Step into", "Step out", and "Step over". The "Debug tools" section shows "NFBTCompiler 3.08s" and "Settings".
- Solution Explorer:** On the left, it shows a project tree with folders like "CAT Instances", "SubApp", "Composite", "Basic", "XPlusMinusY", "XPlusY", "Adapter", "Canvases", "Graphics", and "Data Type". The "XPlusMinusY" folder is selected.
- Object Browser:** In the center, it displays a table of objects for the selected "XPlusMinusY" object. The table has columns for "Name", "Type", "Agra...", and "Initia".
- Diagram:** On the right, a block diagram shows the "XPlusMinusY" function block. It has two input ports labeled "X" and "Y" (both of type "REAL") and two output ports labeled "SUM" and "DIF" (both of type "REAL"). The block also has event inputs and outputs for "REQADD", "REQSUB", and "CNF".
- Message Log:** Below the object browser, it shows "0 Errors", "0 Warnings", and "0 Messages".
- Output Panel:** At the bottom, it shows "Build successfully finished" and "In 45 col 10 ch 10".

A Basic FB to Add and Subtract Two Real Numbers



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

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

A Basic FB that Adds and Subtract 2 Real Numbers

The screenshot displays the EcoStruxure Automation Expert interface. The main workspace shows a function block diagram with a **START** block (green) and a **SUM** block (blue). The **START** block has two outgoing request lines: **REQADD** (labeled '1') to the **SUM** block, and **REQSUB** to a **DIF** block (blue). The **DIF** block has an outgoing **CNF** line to a **DIF** block (pink), which in turn has a **CNF** line back to the **START** block. The **SUM** block has a **CNF** line to a **SUM** block (pink).

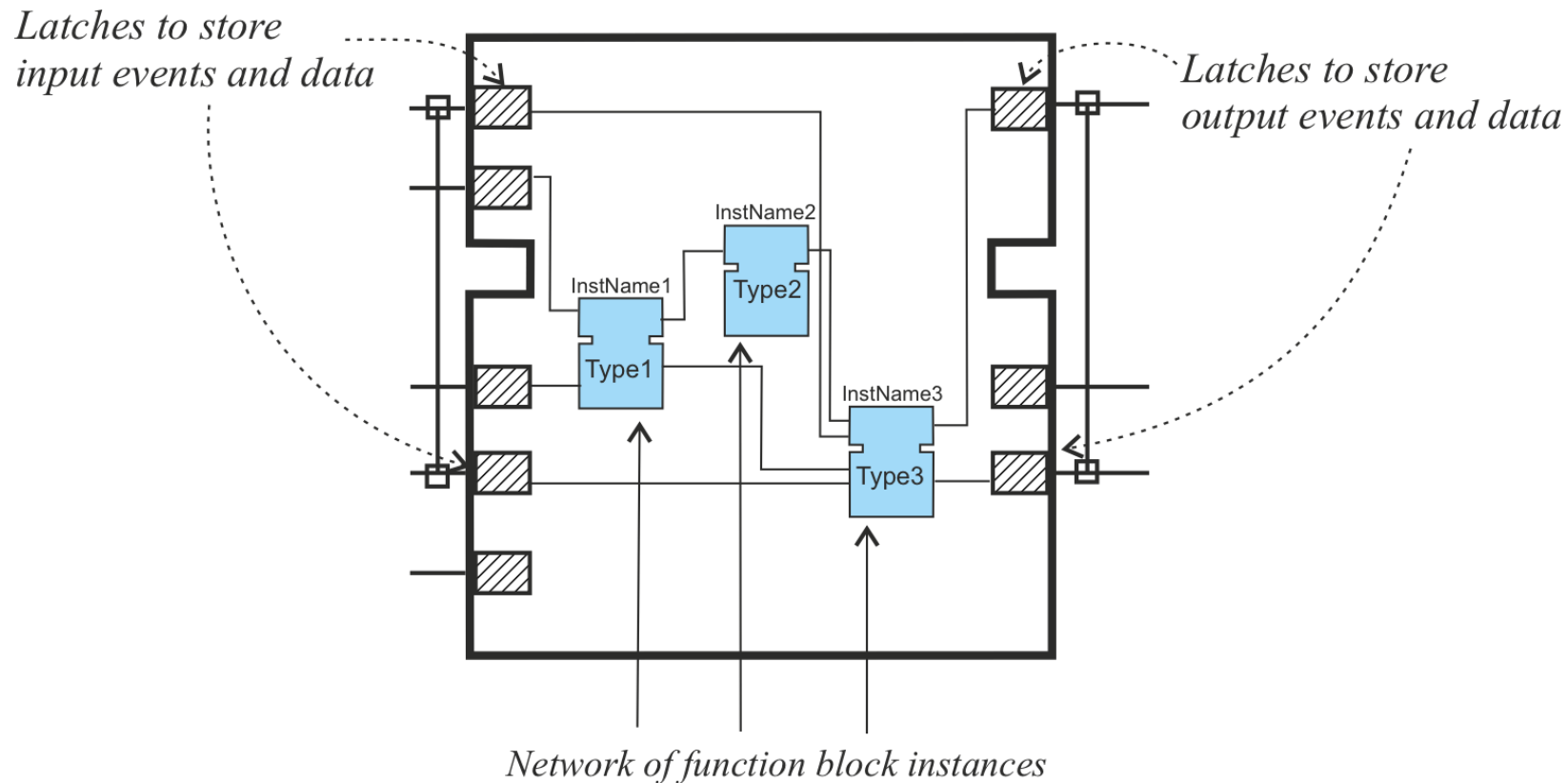
On the right, the **Defaults** panel shows the configuration for the **XPlusMinusY** function block. It features two **EVENT** inputs (blue arrows) for **REQADD** and **REQSUB**. The **REAL** inputs are **X** (value 5) and **Y** (value 3). The **REAL** outputs are **SUM** (value 8) and **DIF** (value 2). There are also **CNF** inputs and outputs.

The **Message Log** at the bottom shows 0 Errors, 0 Warnings, and 0 Messages. The **Output** window shows the message: "Build successfully finished".

Composite Function Block

Composite Function Block

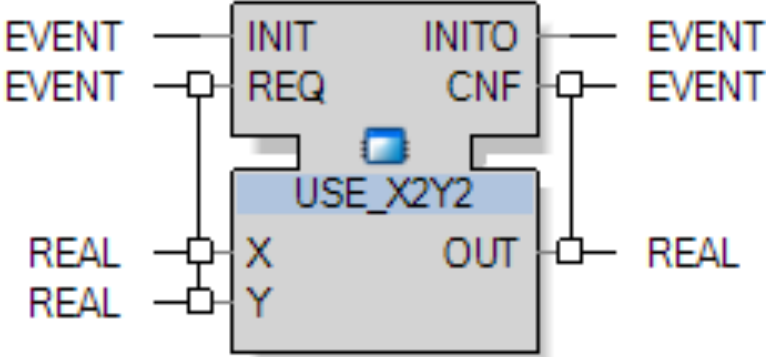
- Network of interconnected FBs
- No internal variables
 - Latches storing the values of input and output events and data
- Nested composite-in-composite



Composite Function Block: Example

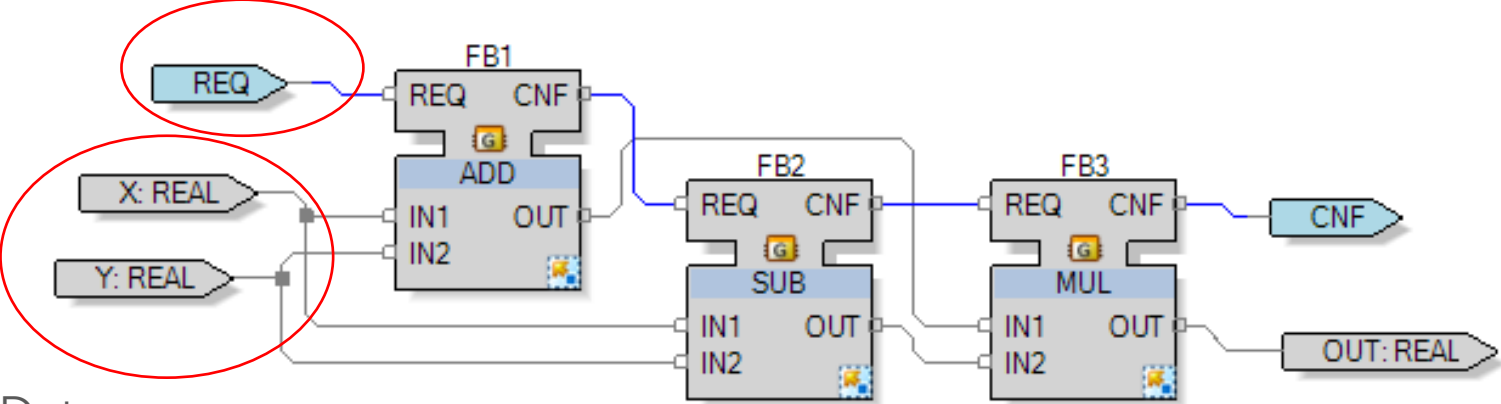
Computing $OUT = X^2 - Y^2$

Interface



Implementation

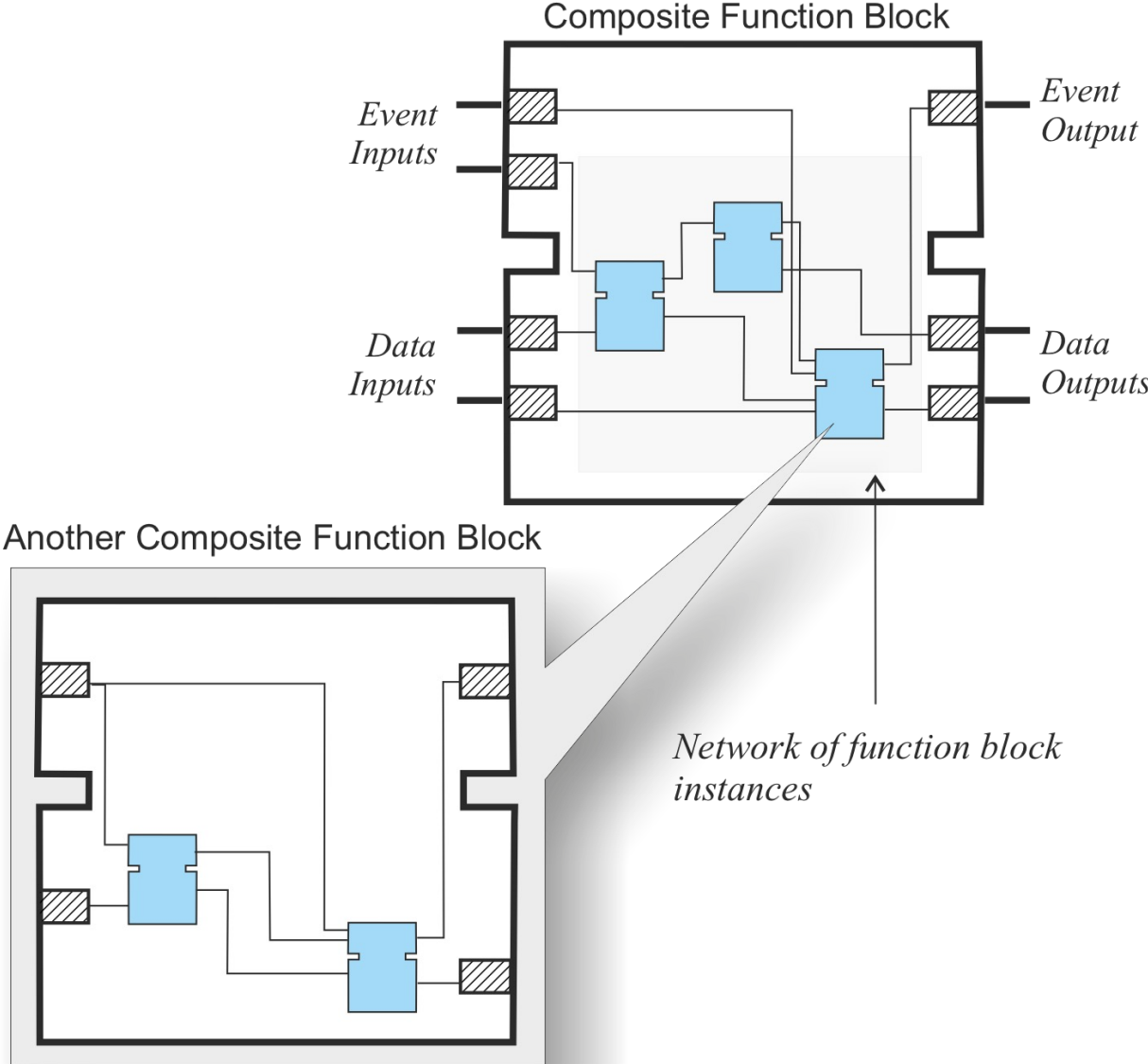
Event



Data

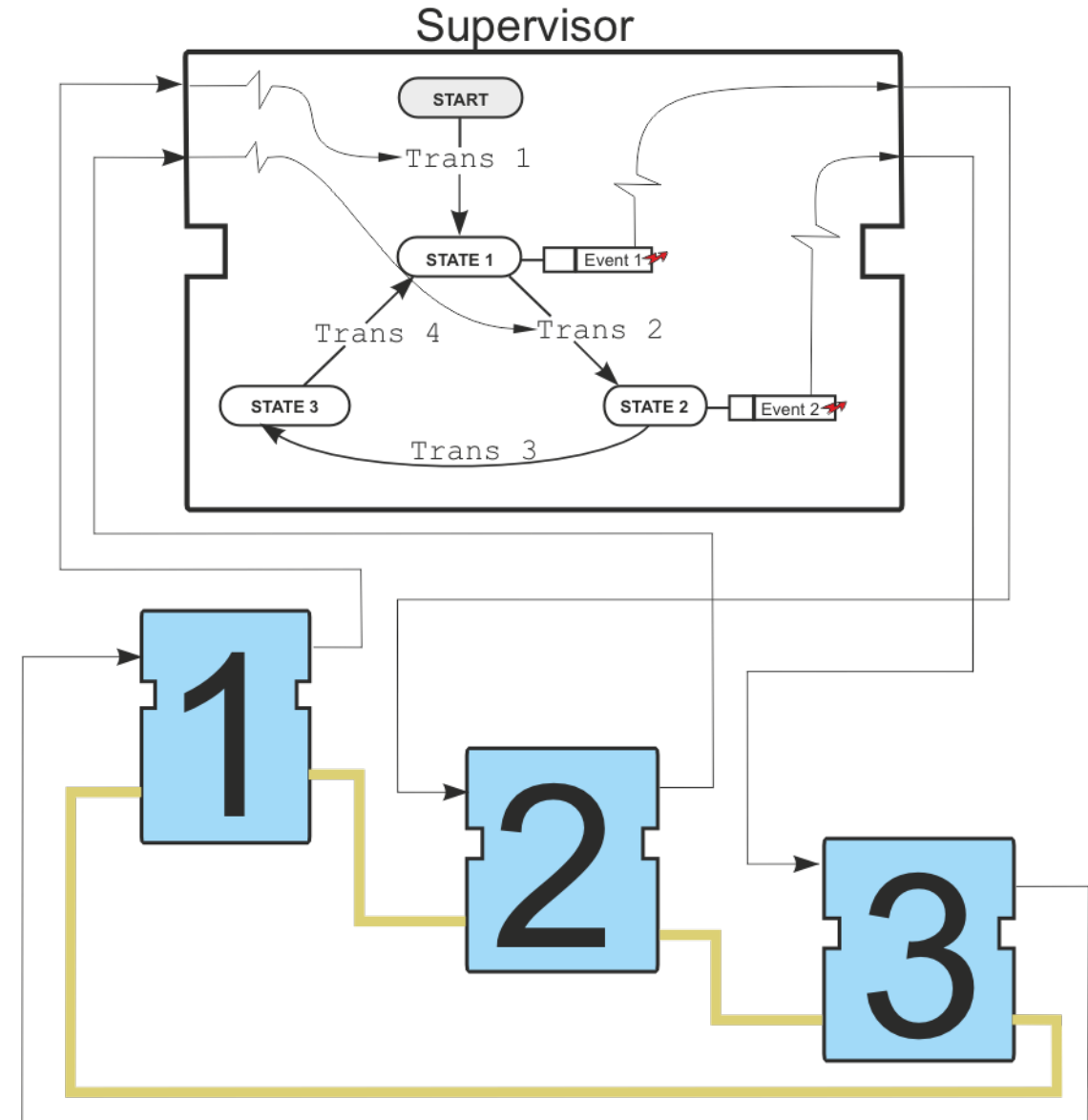
$OUT := (X - Y) * (X + Y);$

Hierarchical composition



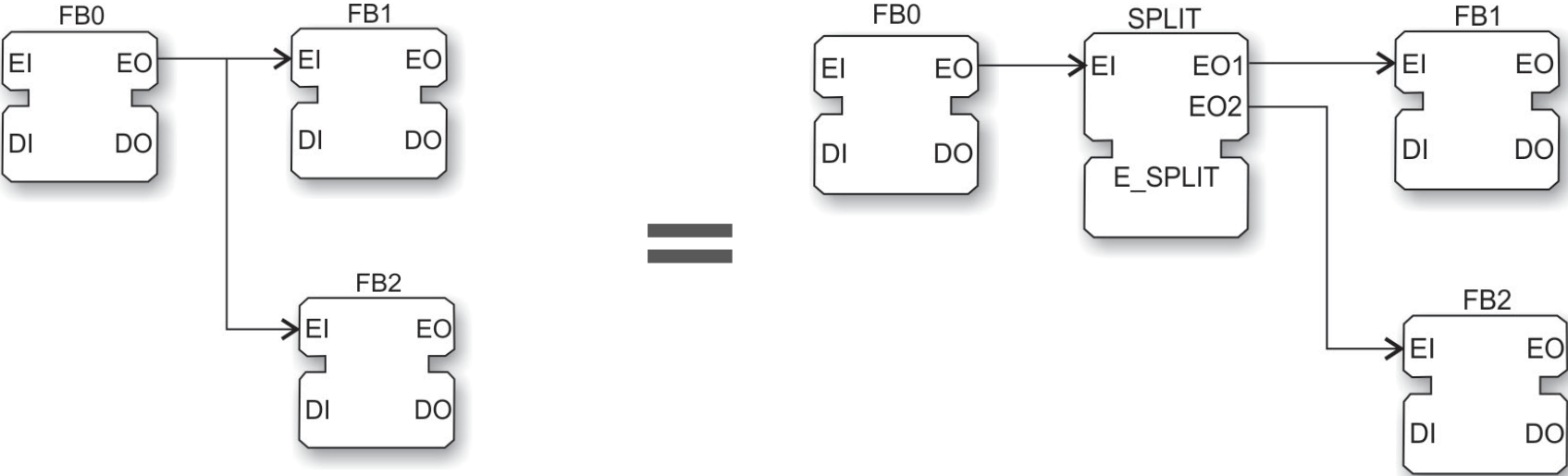
Composite Function Block

- Execution Control:
 - No ECC
 - No internal variables
- To ensure a particular order of execution, user can implement a supervisor basic function block



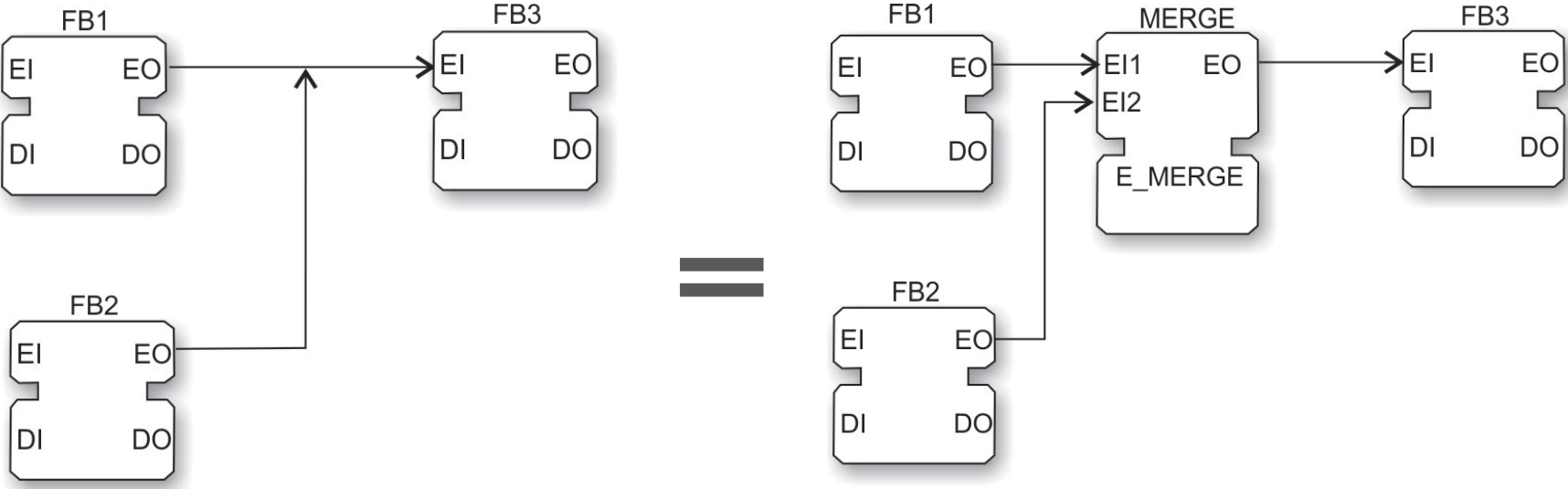
Rules for event connections

Event split is equivalent to using E_SPLIT standard library FB



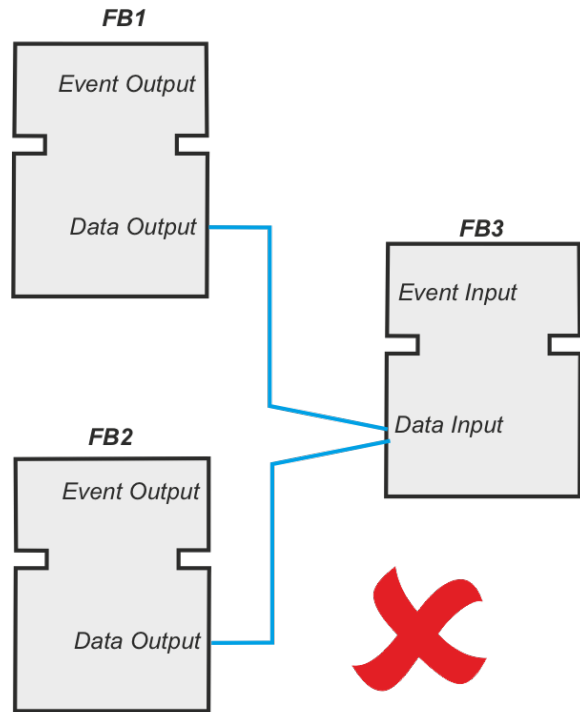
Event merge

Event merge is equivalent to using E_MERGE standard library FB

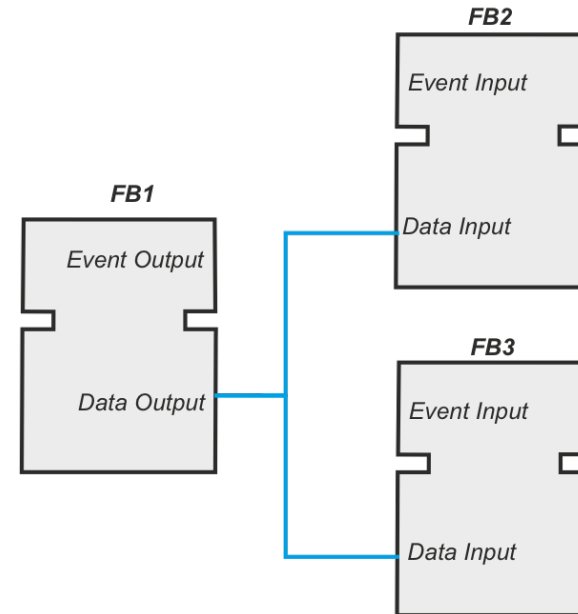


Data connections

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



Incorrect



Allowed

Service Interface Function Block

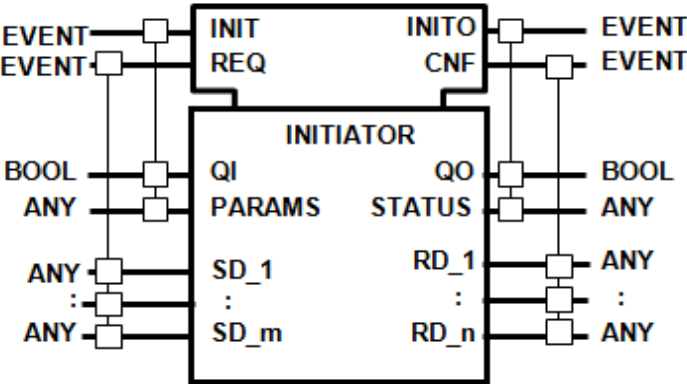
Service interface function blocks (SIFB)

- 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

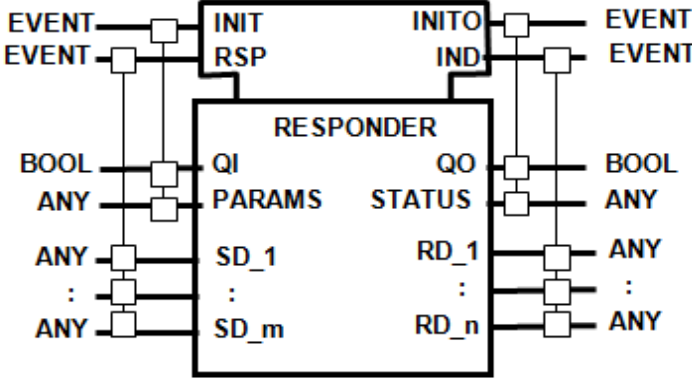
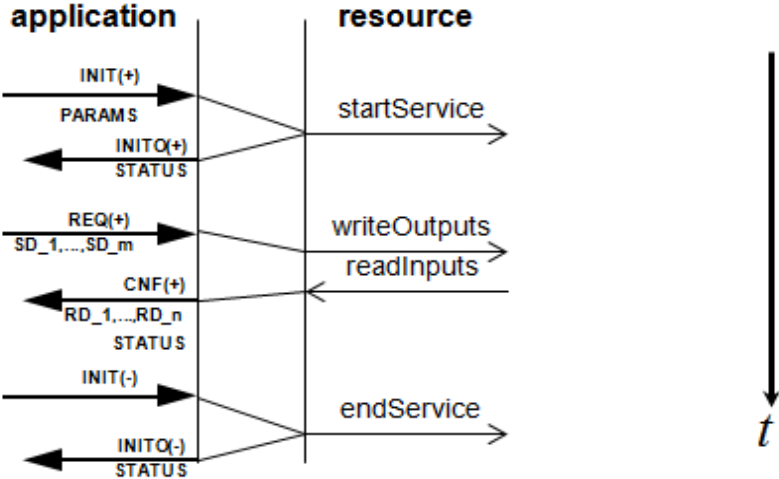


Service Interface Function Blocks (SIFB)

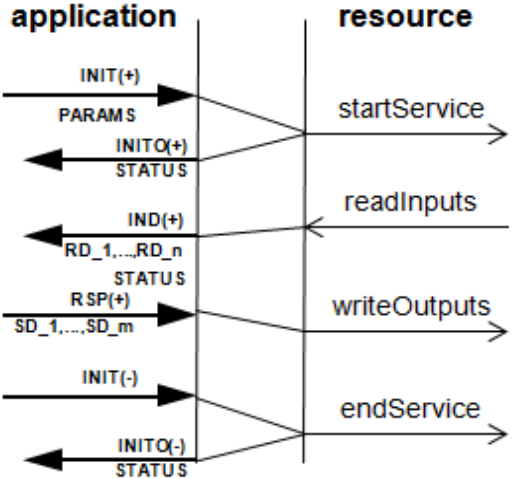
- Modelled as sequences of service primitives per ISO TR 8509



(application-initiated transactions)

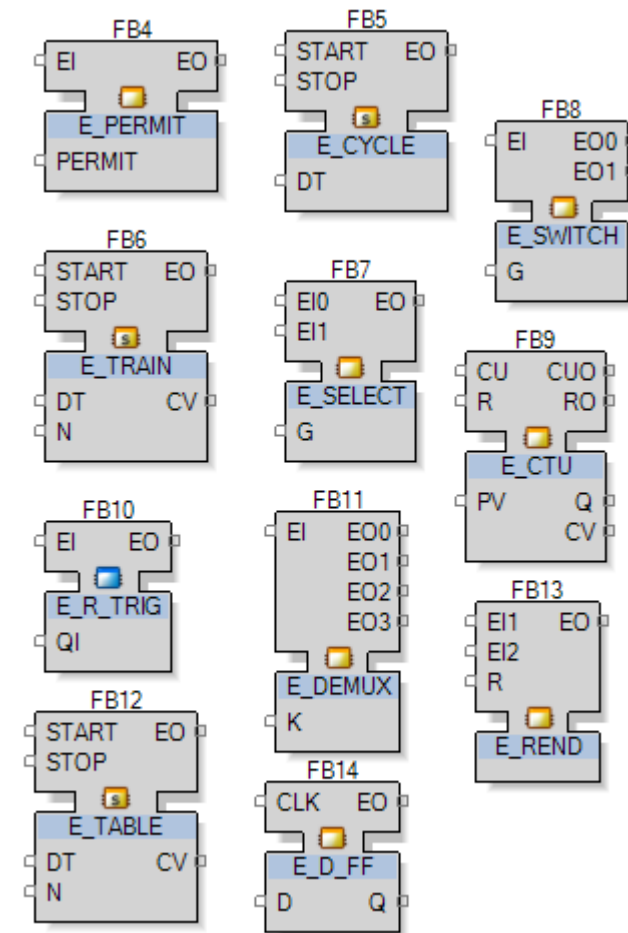


(resource-initiated transactions)

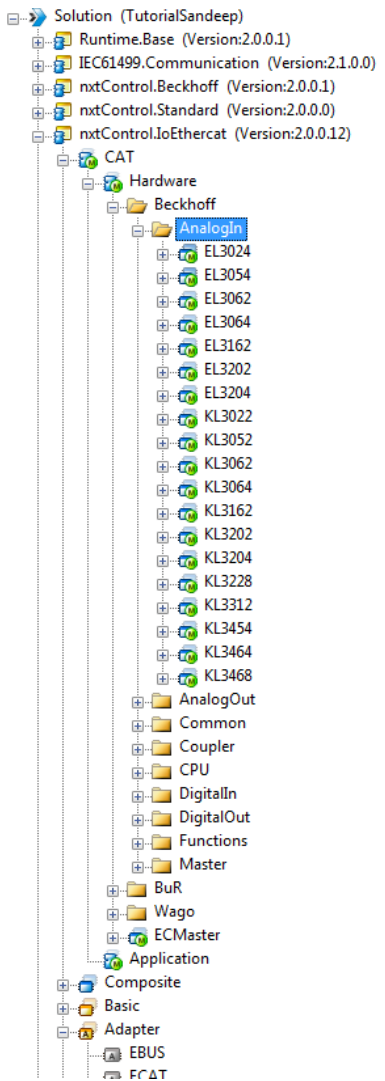
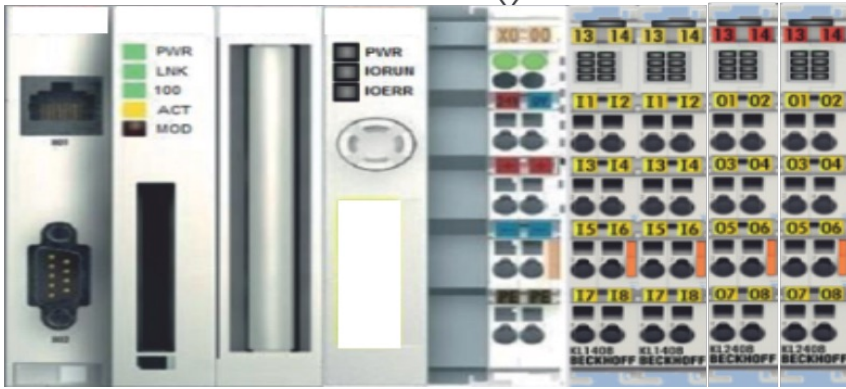
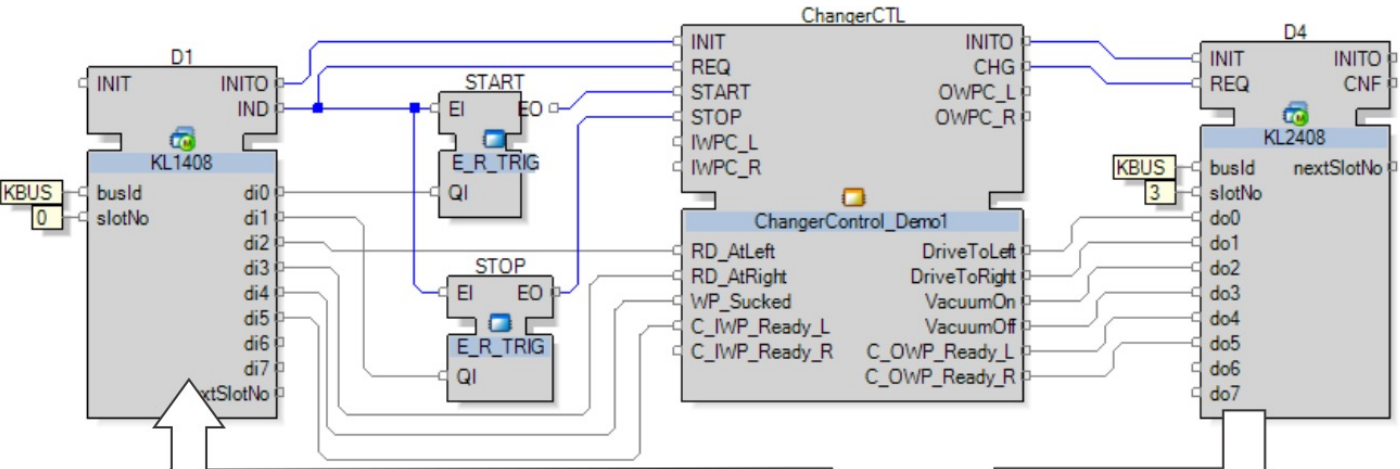


Operations with events

- **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.



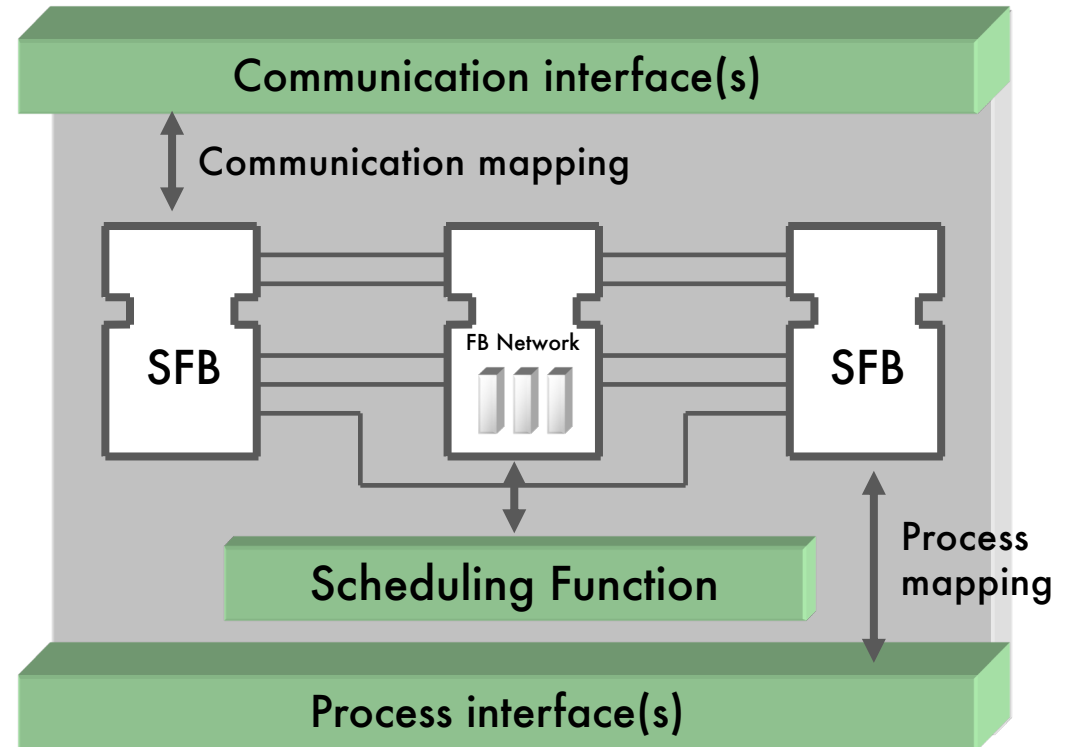
Process interface: read inputs and write outputs



Elements of Distributed systems

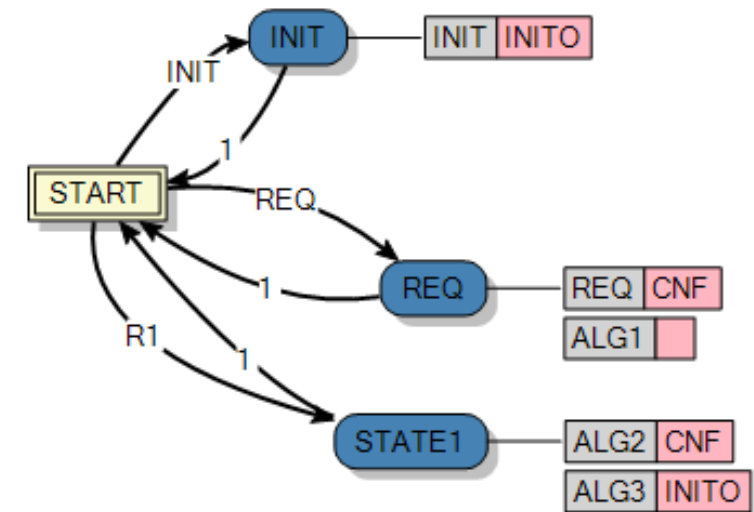
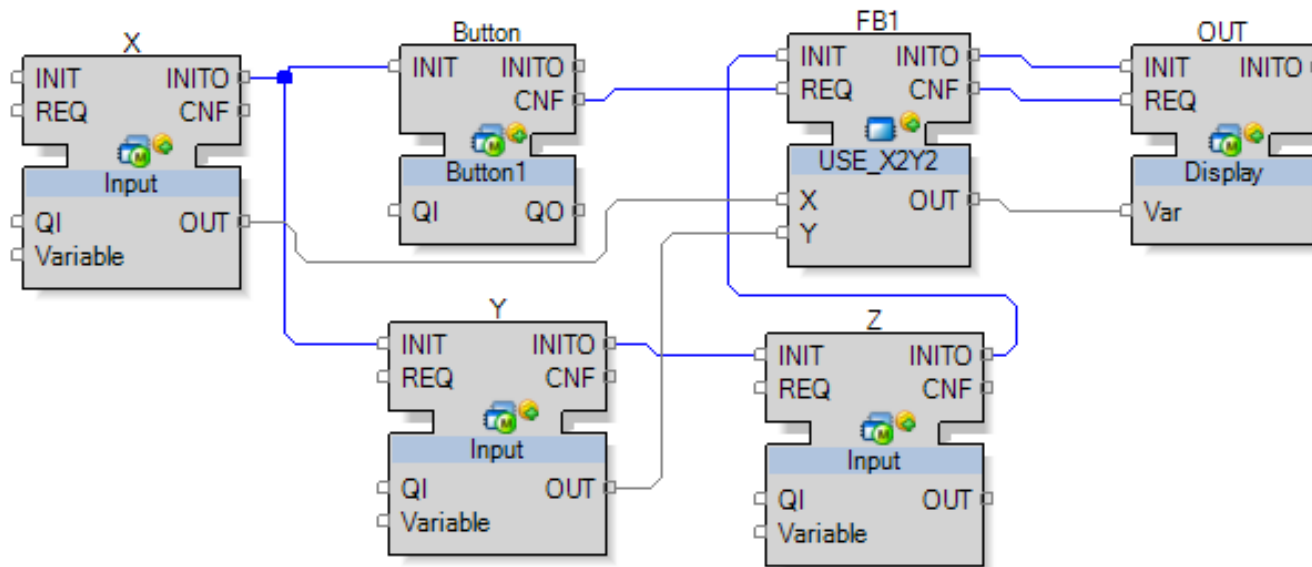
Resource Model

- 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



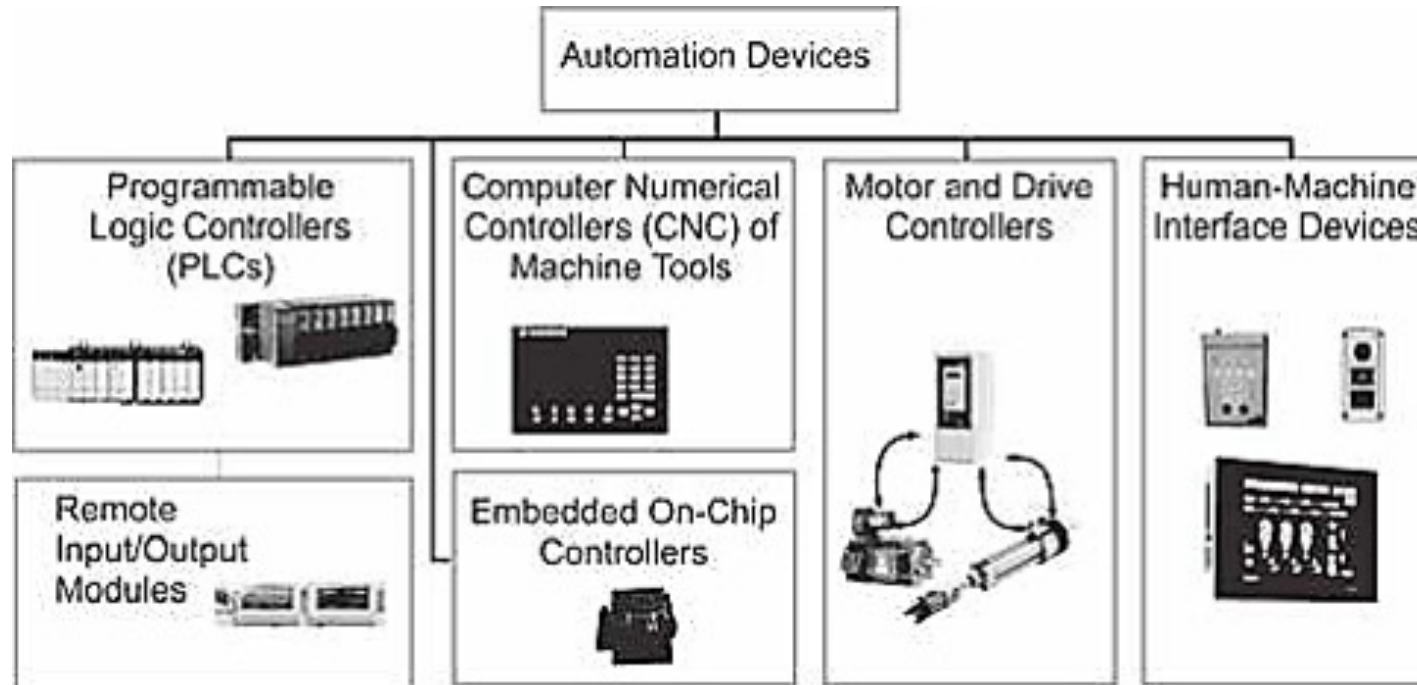
Resources – scheduling functions

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



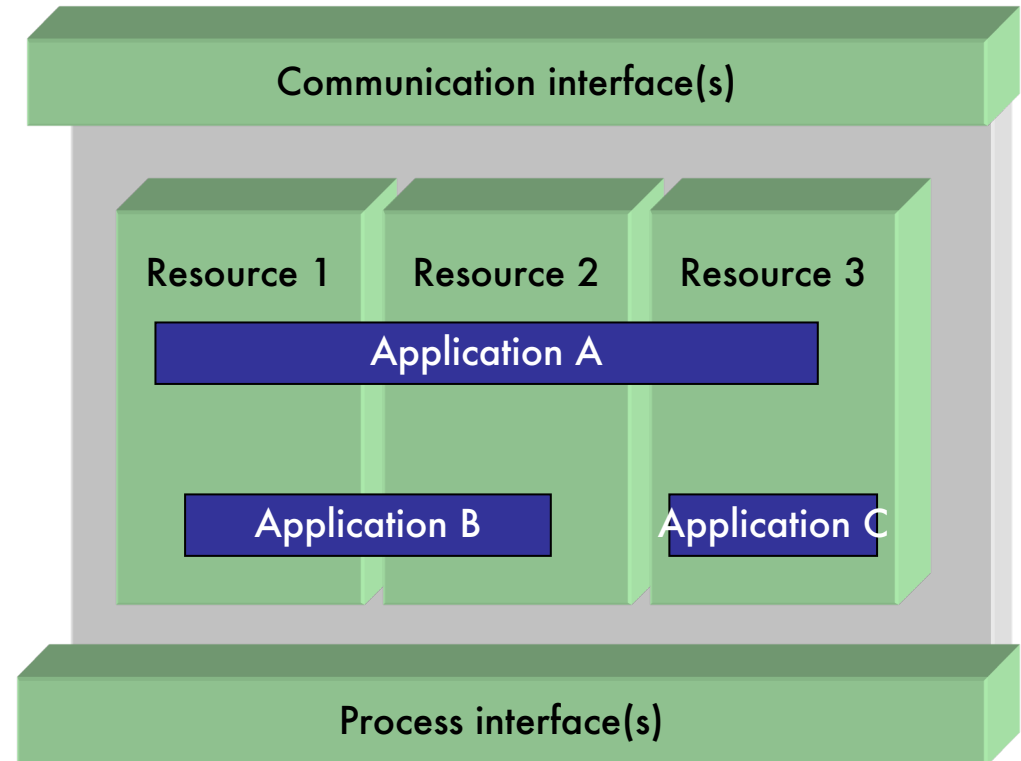
Device Model

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



Device Model

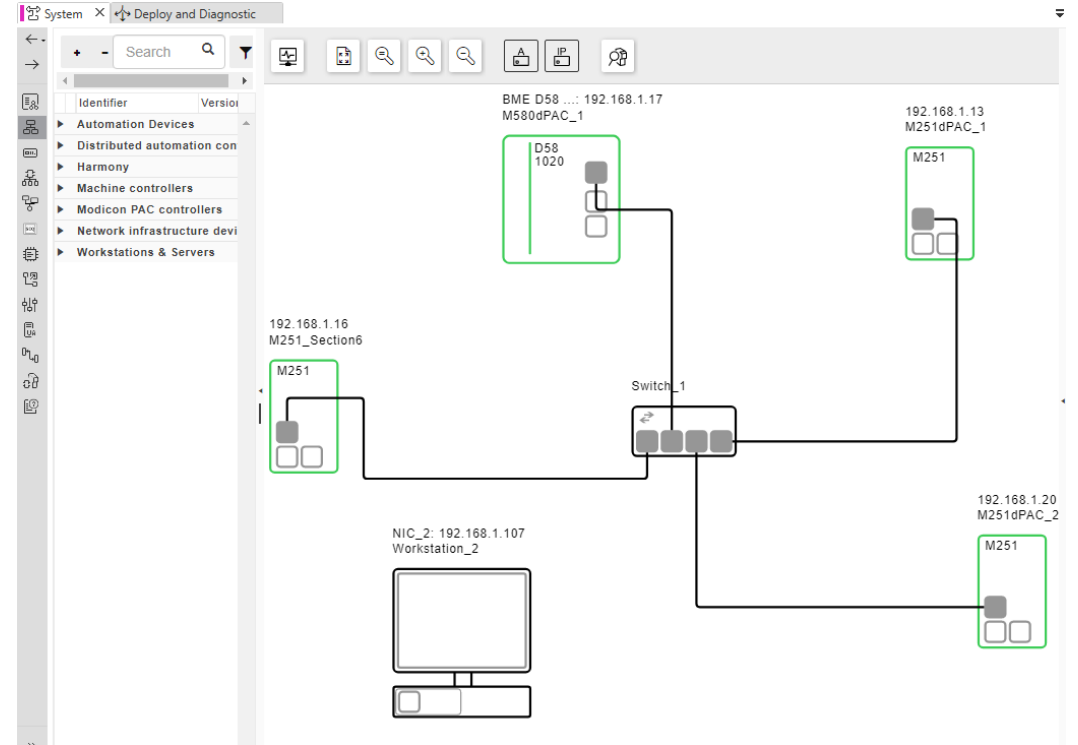
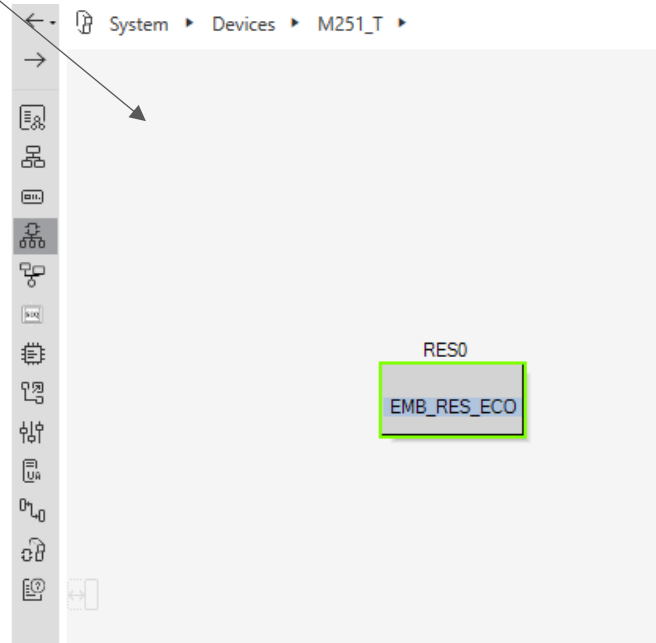
- A Device is specified by its process interfaces and communication interfaces
 - Process interface is the mapping between the physical entities (sensors, actuators) and the Resource
- Contains multiple Resources
- All Resources use the same Communication and Process interfaces of the Device



Device model: example of use

Network Profile: (Default)

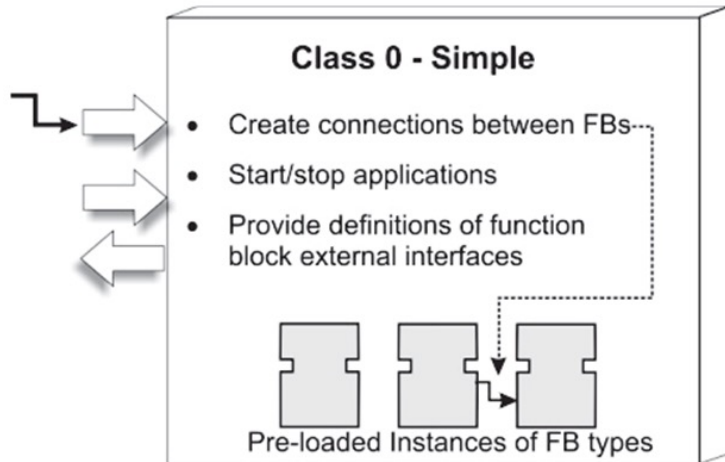
Logical Device Name	Device Type	Info	Physical Device
M251_Section6	SE.DPAC.M251_dPAC		M251_Section6
IB_Section1	nxtControl.Standard.ECORT		
IB_Section2	nxtControl.Standard.ECORT		
IB_Section4	nxtControl.Standard.ECORT		
IB_Section5	nxtControl.Standard.ECORT		
M580	SE.DPAC.M580_dPAC		M580dPAC_1\BME D58 1020 #0
M251_T	SE.DPAC.M251_dPAC		M251dPAC_1
M251_Sec4	SE.DPAC.M251_dPAC		M251dPAC_2
DEV1	SE.DPAC.Soft_dPAC		(None)



Device classes

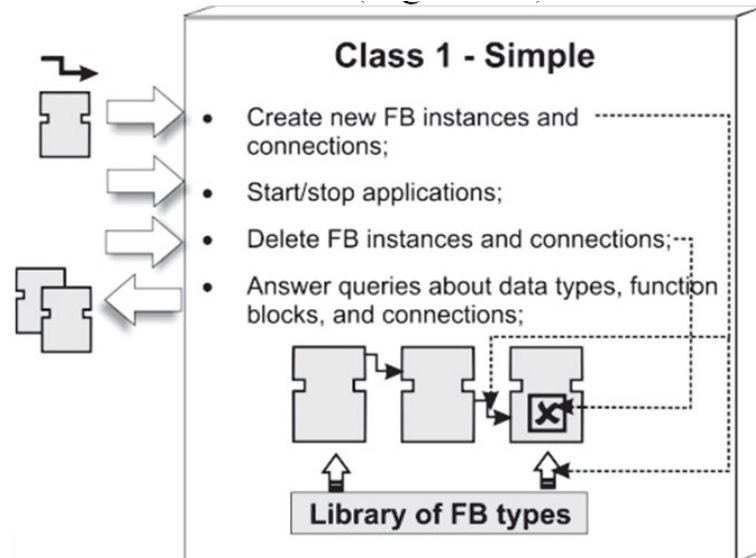
Class 0:

- Preprogrammed FB instances
- FBs connections can be changed



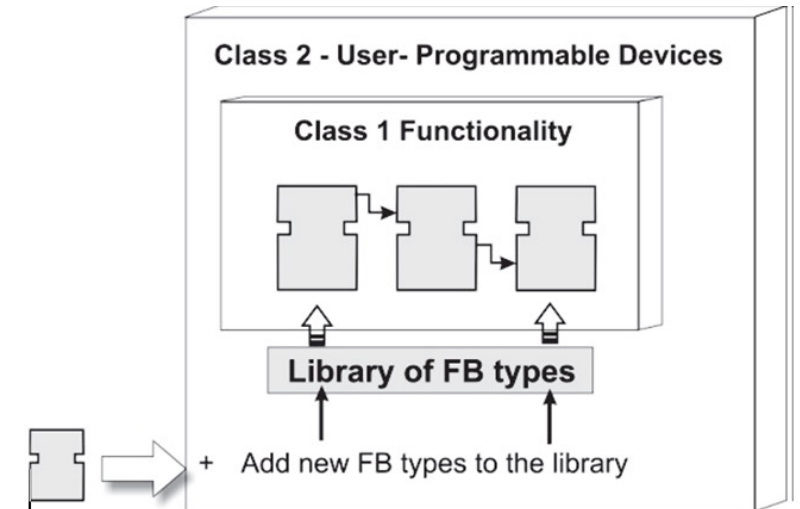
Class 1

- 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

New Device ✕

Name

Count

Type SE.DPAC:Archive_Database

SE.DPAC:Archive_Database

SE.DPAC:Archive_Link

SE.DPAC:ATV_dPAC

SE.DPAC:M251_dPAC

SE.DPAC:M262_dPAC

SE.DPAC:M580_dPAC

SE.DPAC:Soft_dPAC

SE.DPAC:Soft_dPAC_RDNT

SE.Standard:HMI_NET

←
→
Add Device
Add Device CAT
Add Folder
Delete

Network Profile:
(Default) ▼

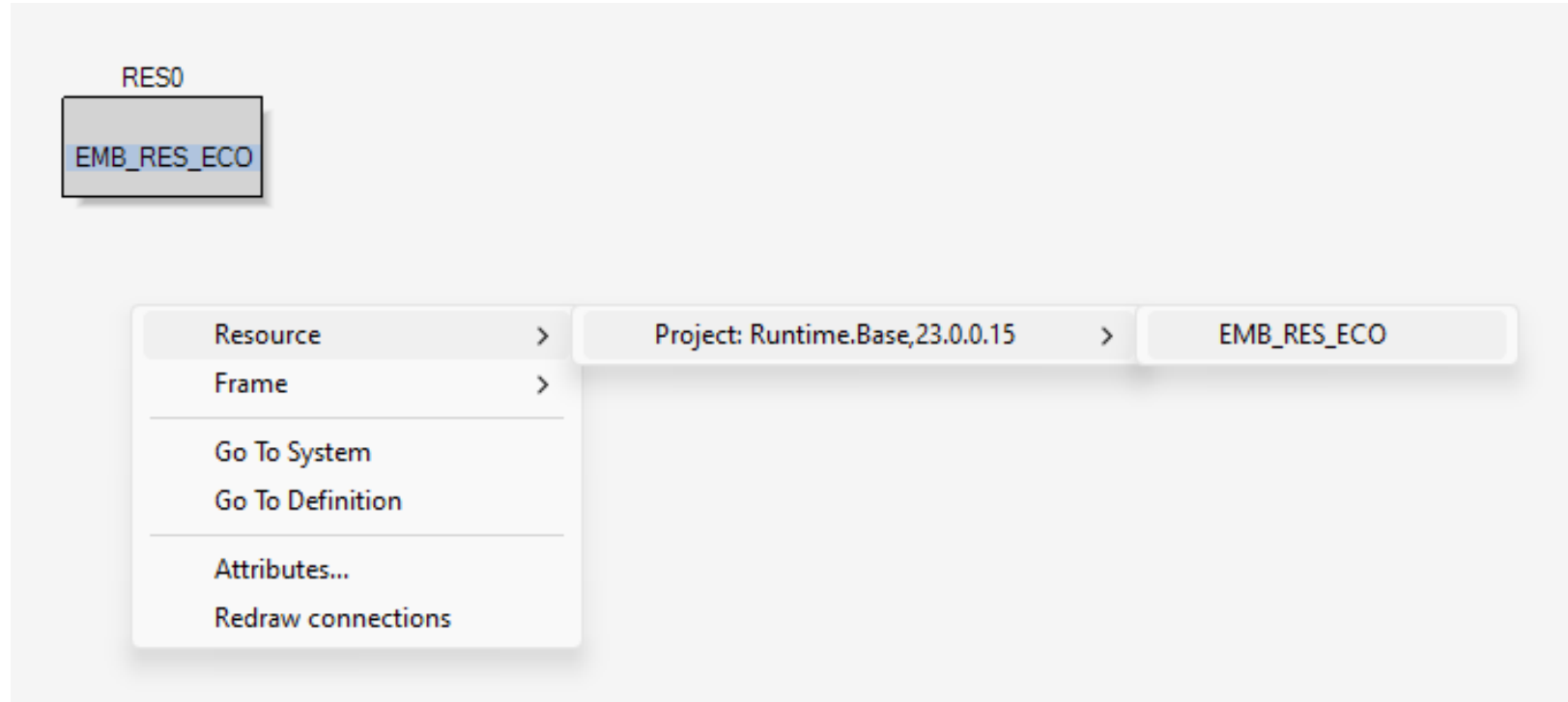
Logical Device Name	Device Type	Info	Physical Device
M251_Section6	SE.DPAC.M251_dPAC		M251_Section6
IB_Section1	nxtControl.Standard.ECORT		
IB_Section2	nxtControl.Standard.ECORT		
IB_Section4	nxtControl.Standard.ECORT		
IB_Section5	nxtControl.Standard.ECORT		
M580	SE.DPAC.M580_dPAC		M580dPAC_1\BME D58 1020 #0
M251_T	SE.DPAC.M251_dPAC		M251dPAC_1

Service Name	Interface	IP Address	Logical Port
Deployment	Any	All local IPs	51443
HMI			
Watch			
Archive Service	Any	All local IPs	51496

M251_Sec4	SE.DPAC.M251_dPAC		M251dPAC_2
DEV1	SE.DPAC.Soft_dPAC		(None)

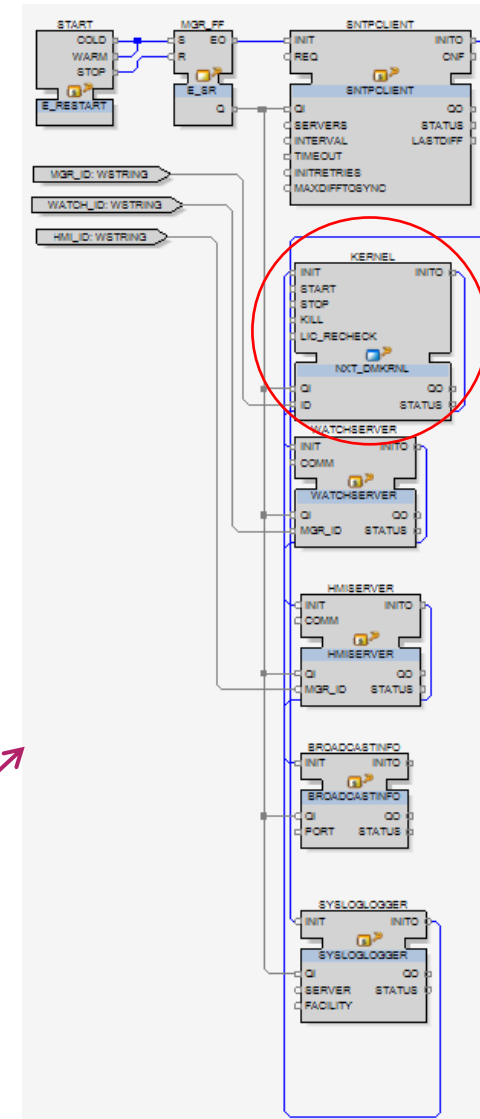
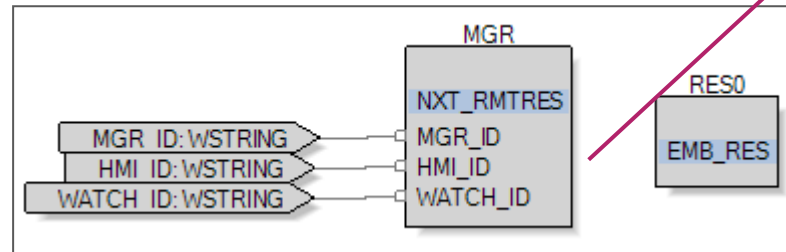
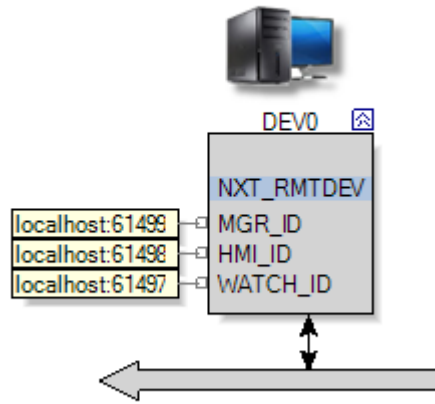
Resource type definition

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



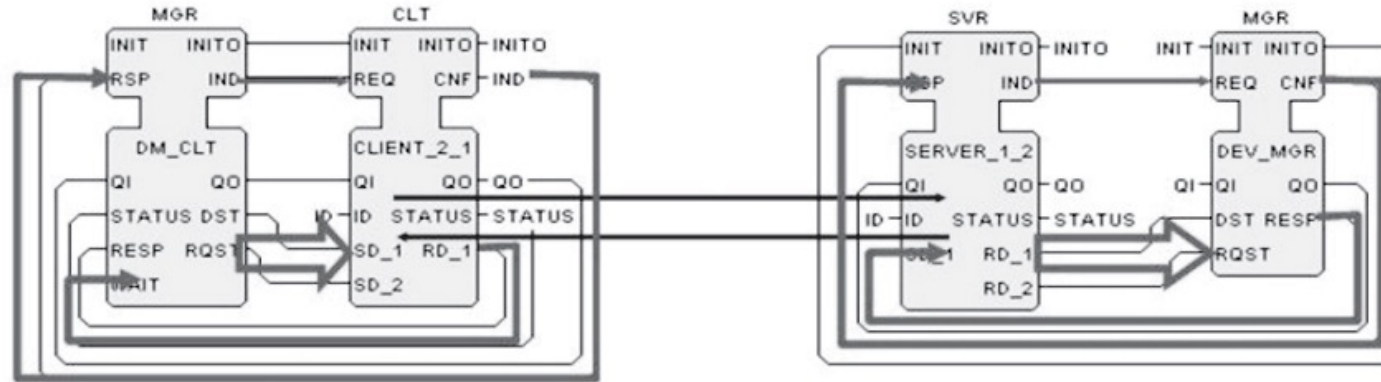
Device management

- 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 Proxy
(in Software Toolset)**

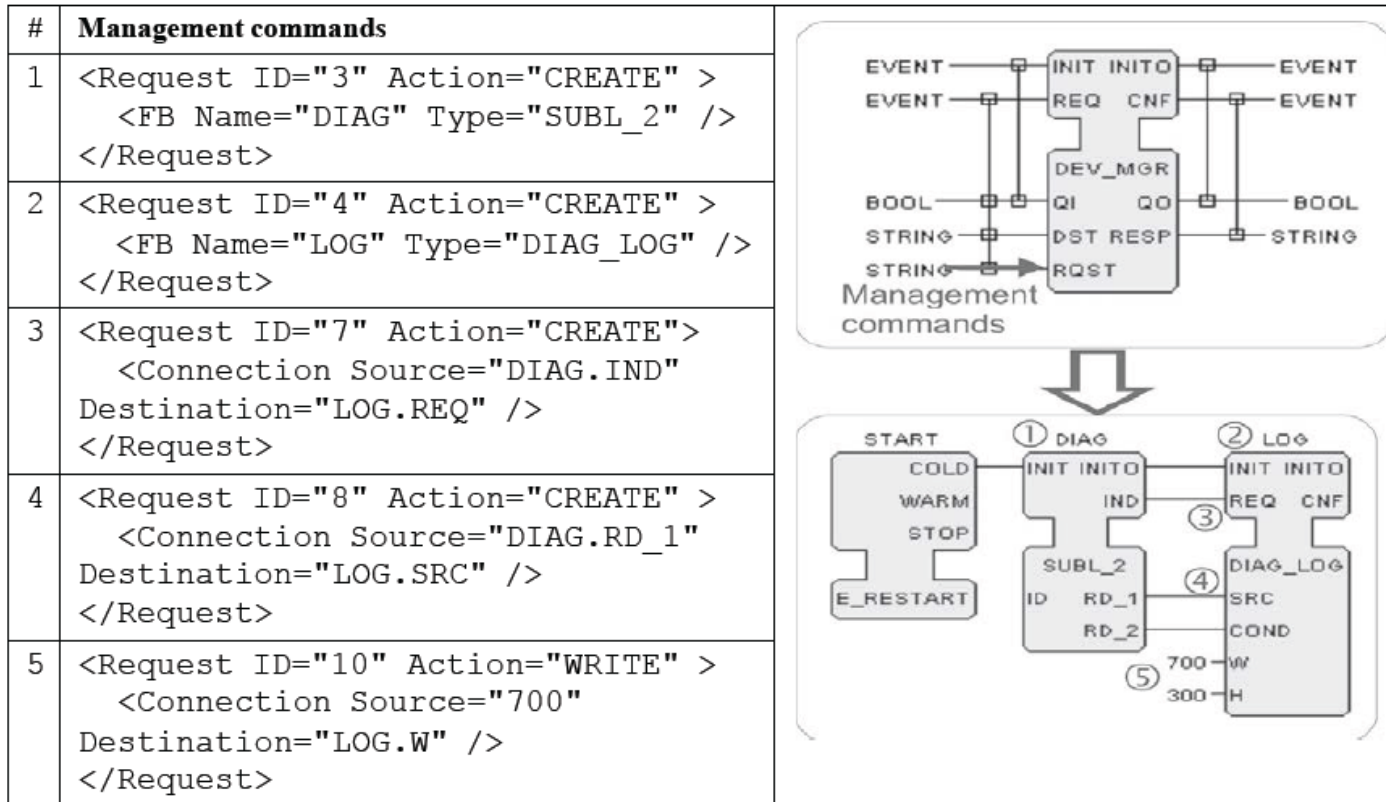
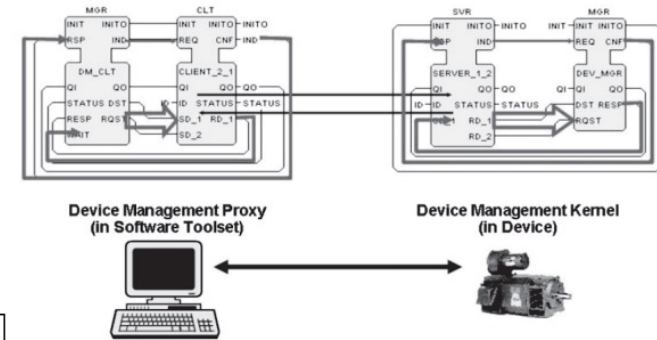


**Device Management Kernel
(in Device)**



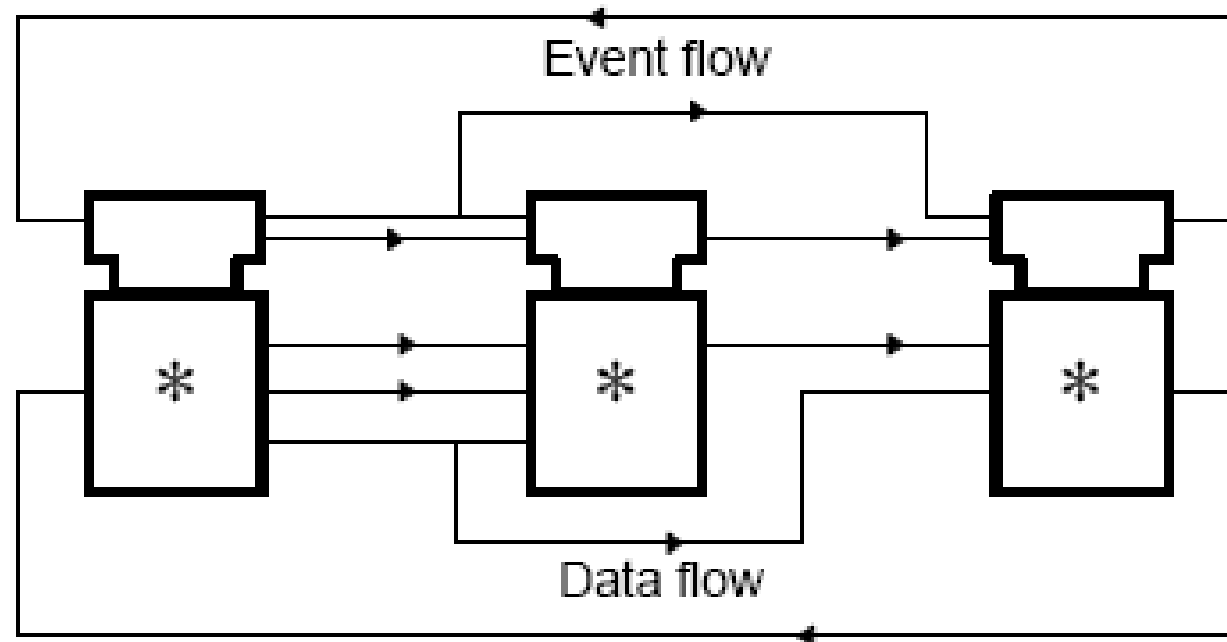
Device management

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

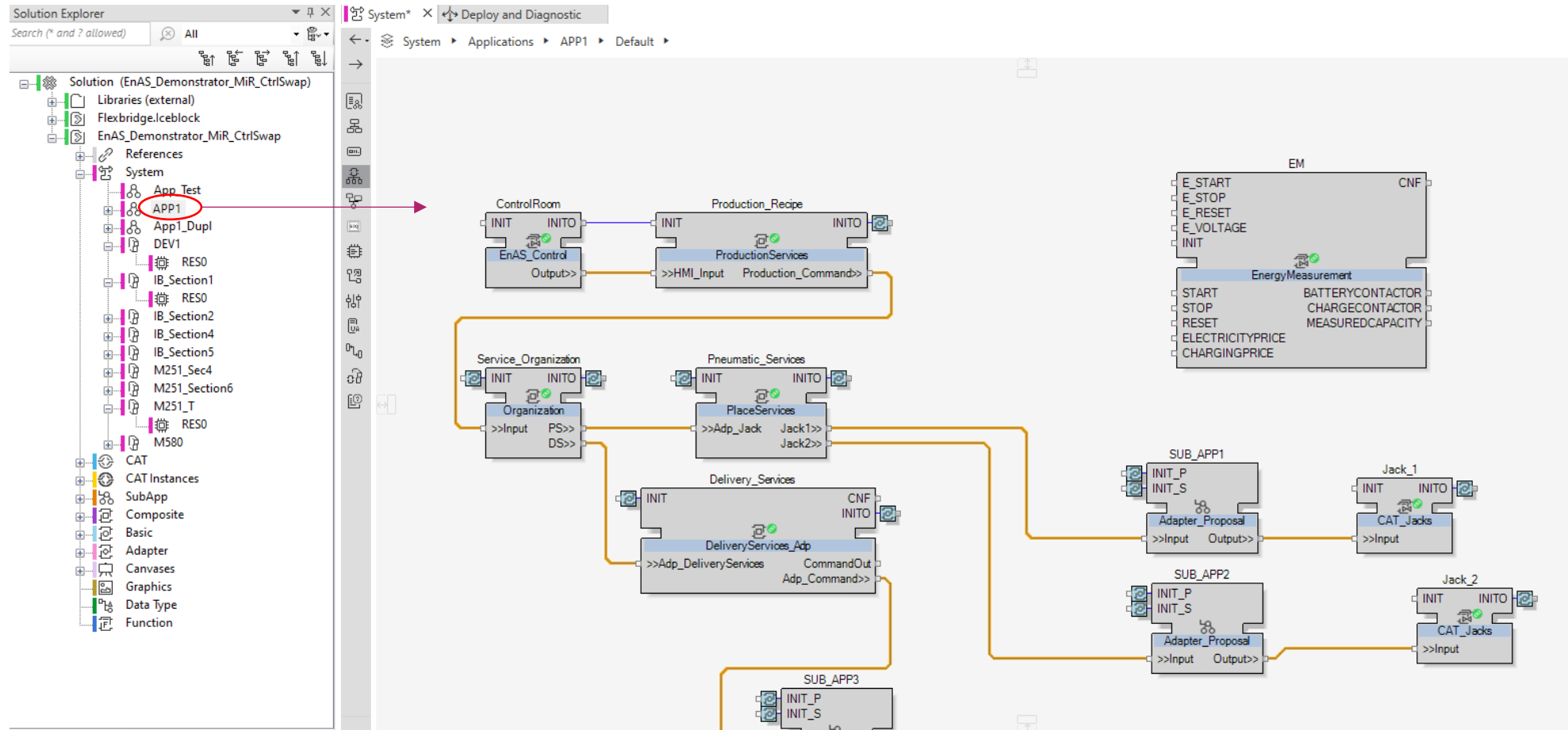


Application

- 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

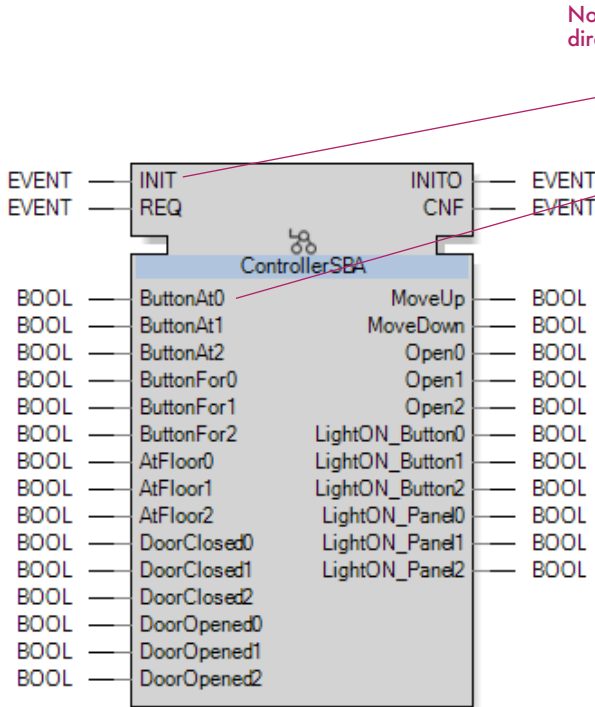


Application: Example

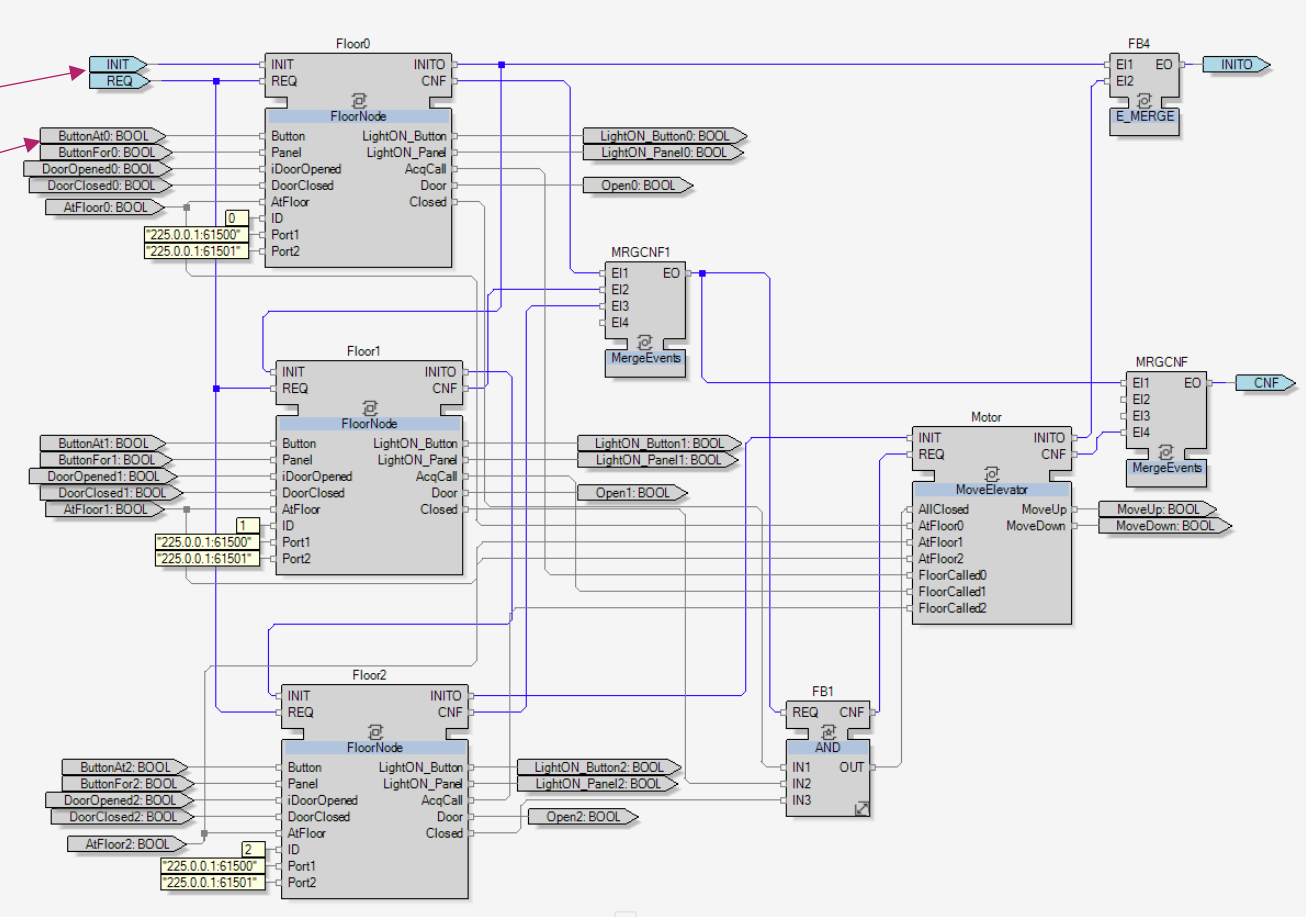


Subapplication

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

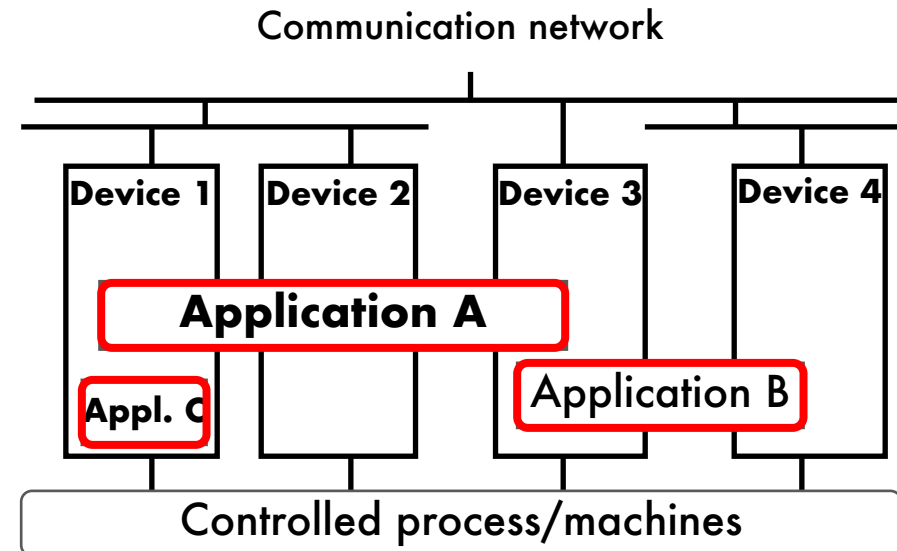


No latching of events and data, direct propagation



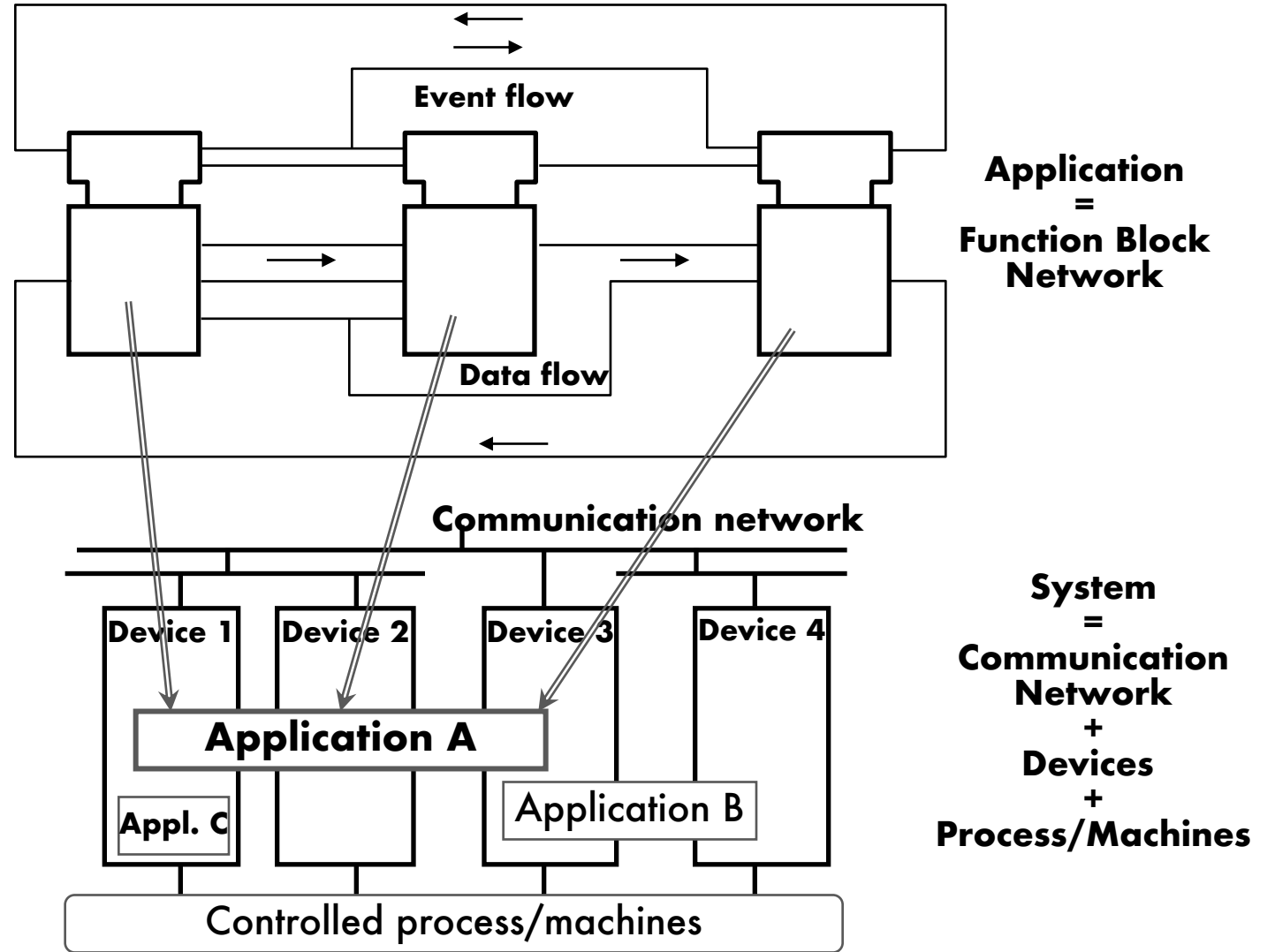
System Configuration

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

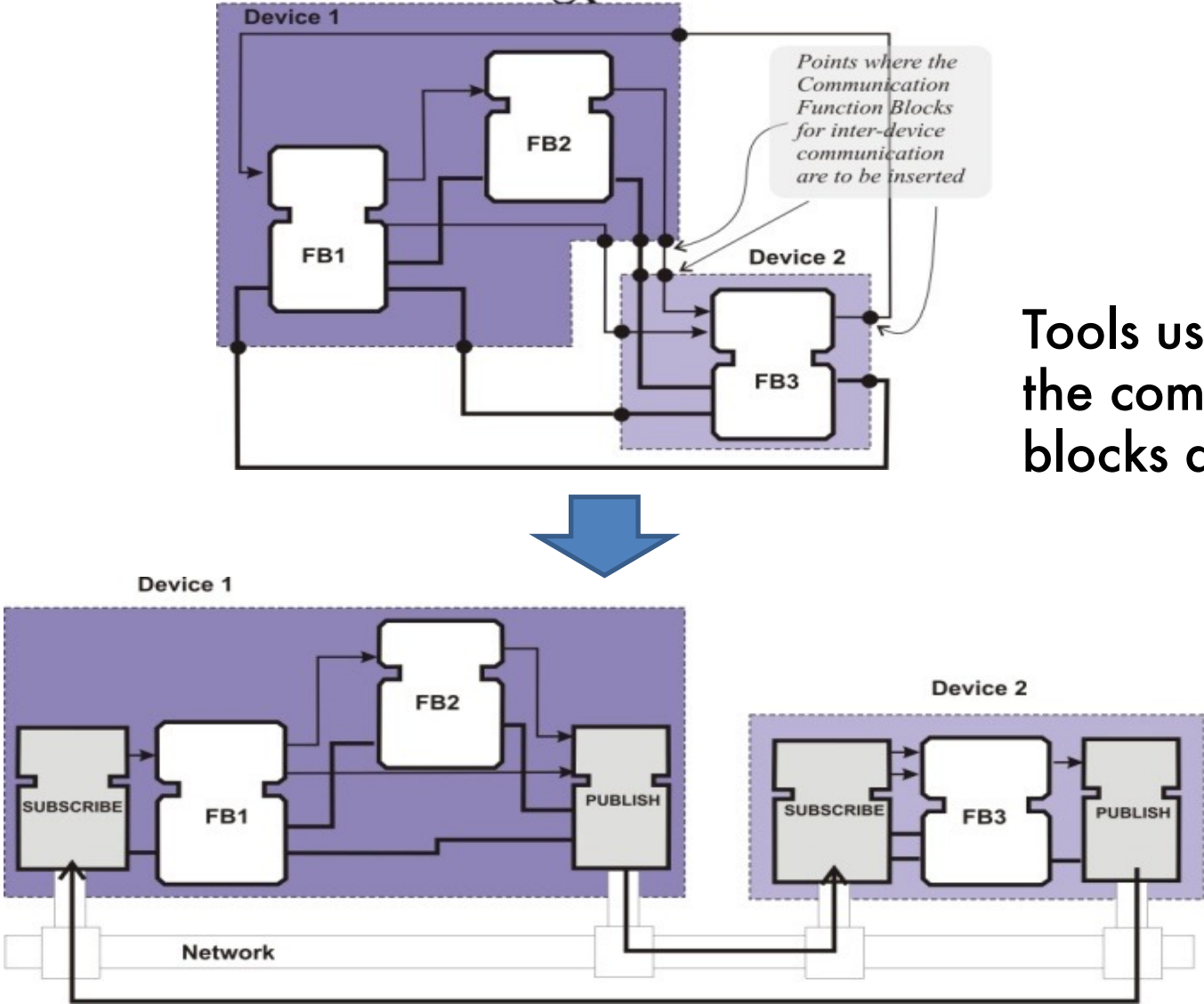


Distribution of Application

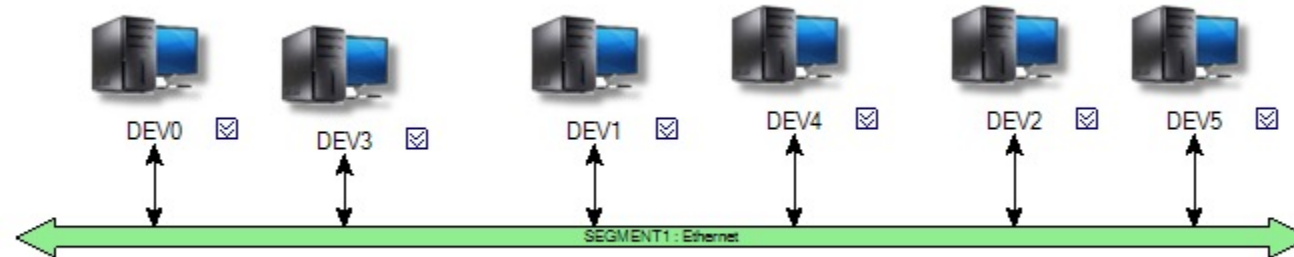
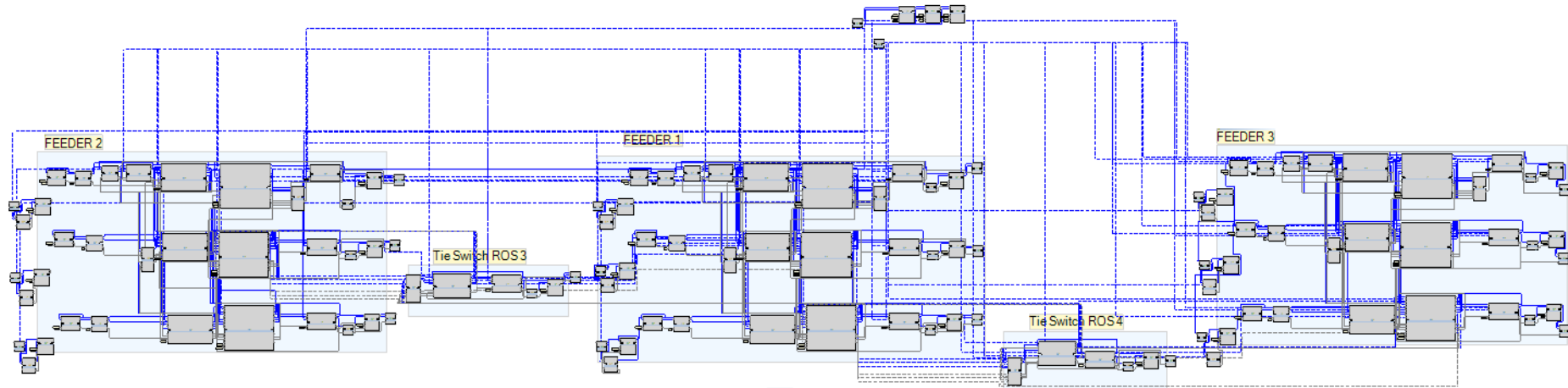
- Application can be deployed to several devices
- A device in the system can contain and execute parts of different applications



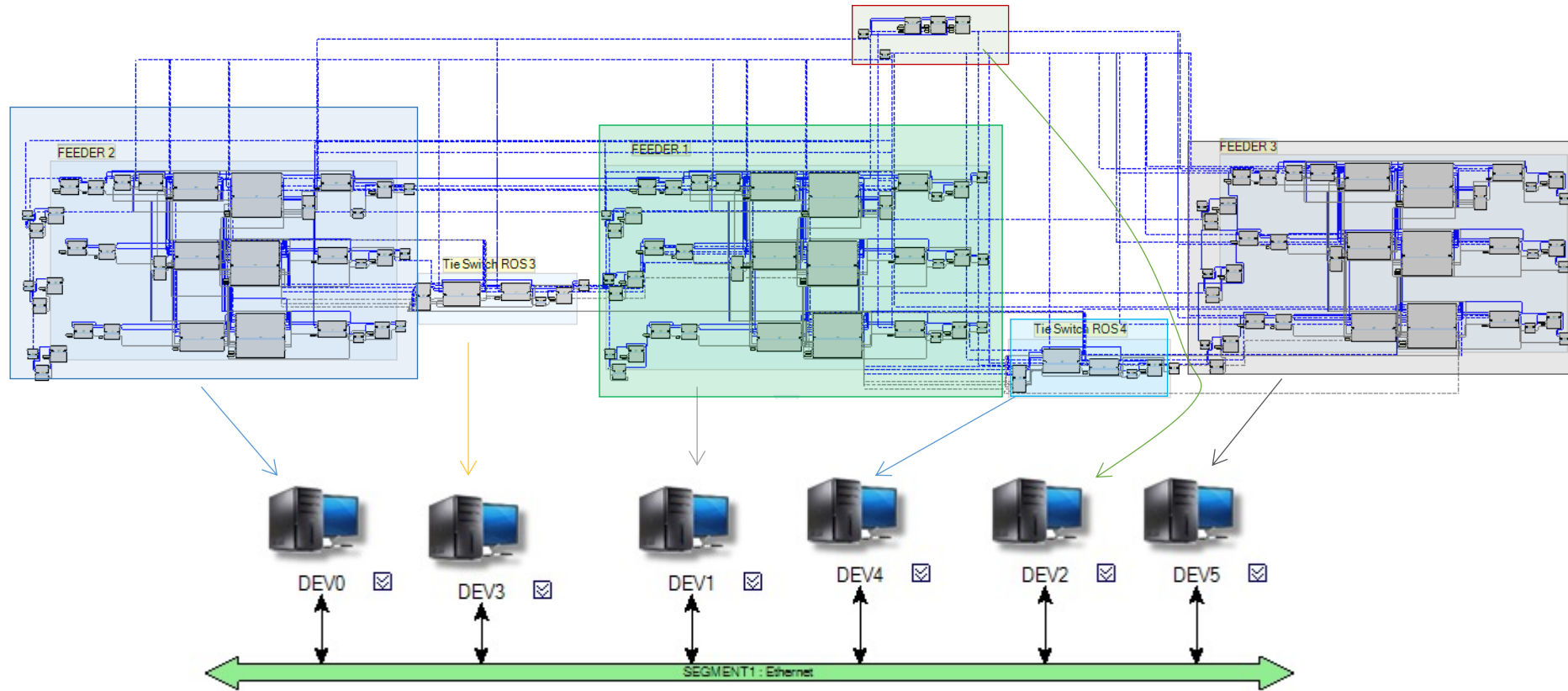
Distribution of Application



Example: Distributed systems configuration



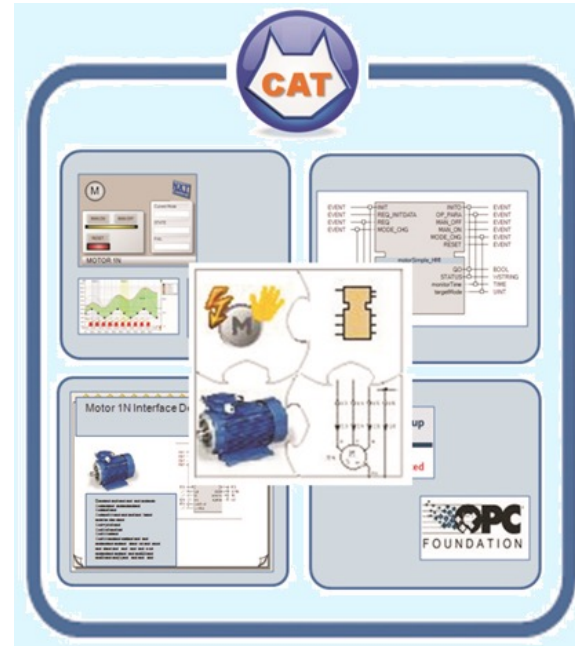
Example: Distributed systems configuration



Composite Automation Type (CAT)

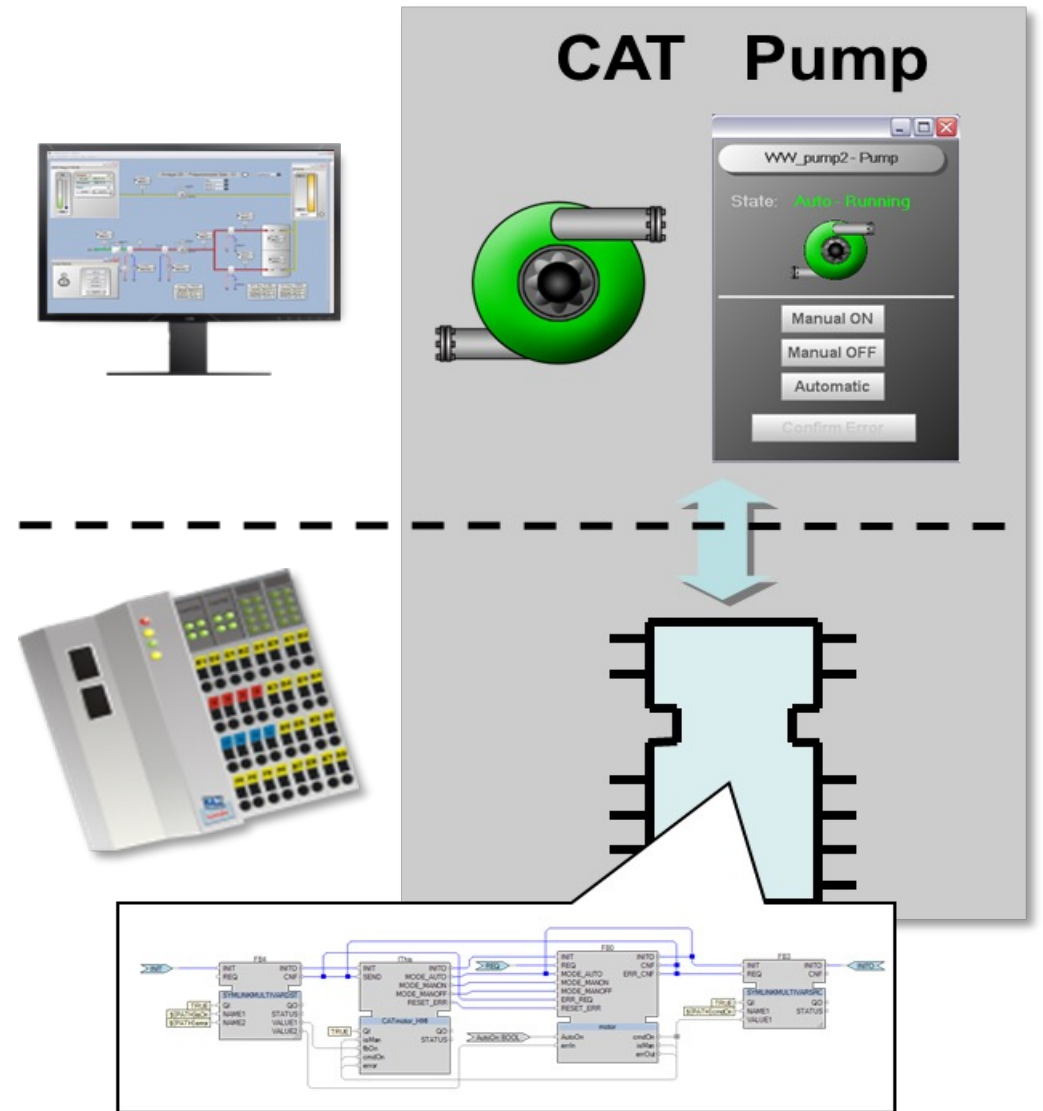
Composite Automation Type (CAT)

- ❑ 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

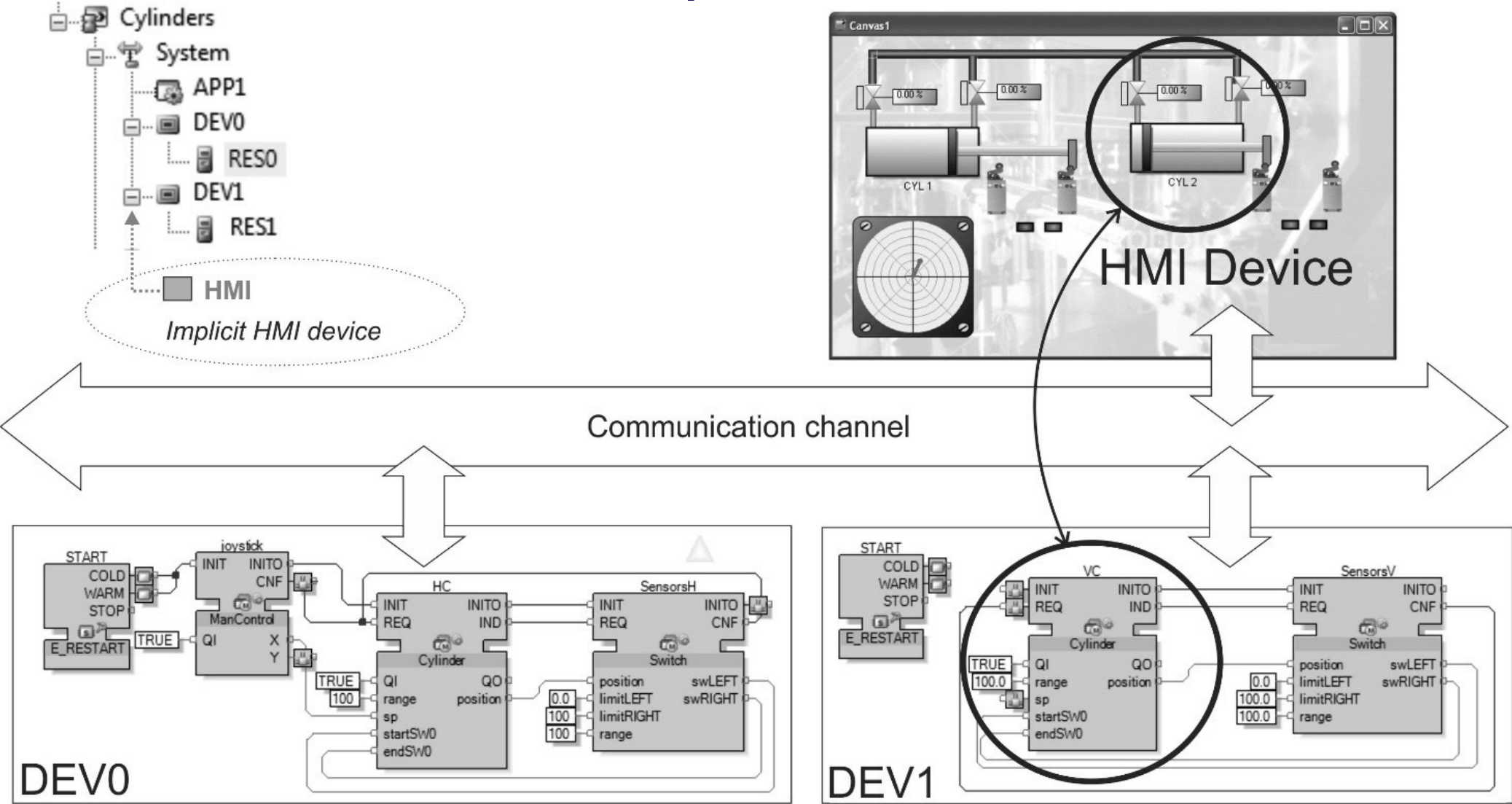


CAT

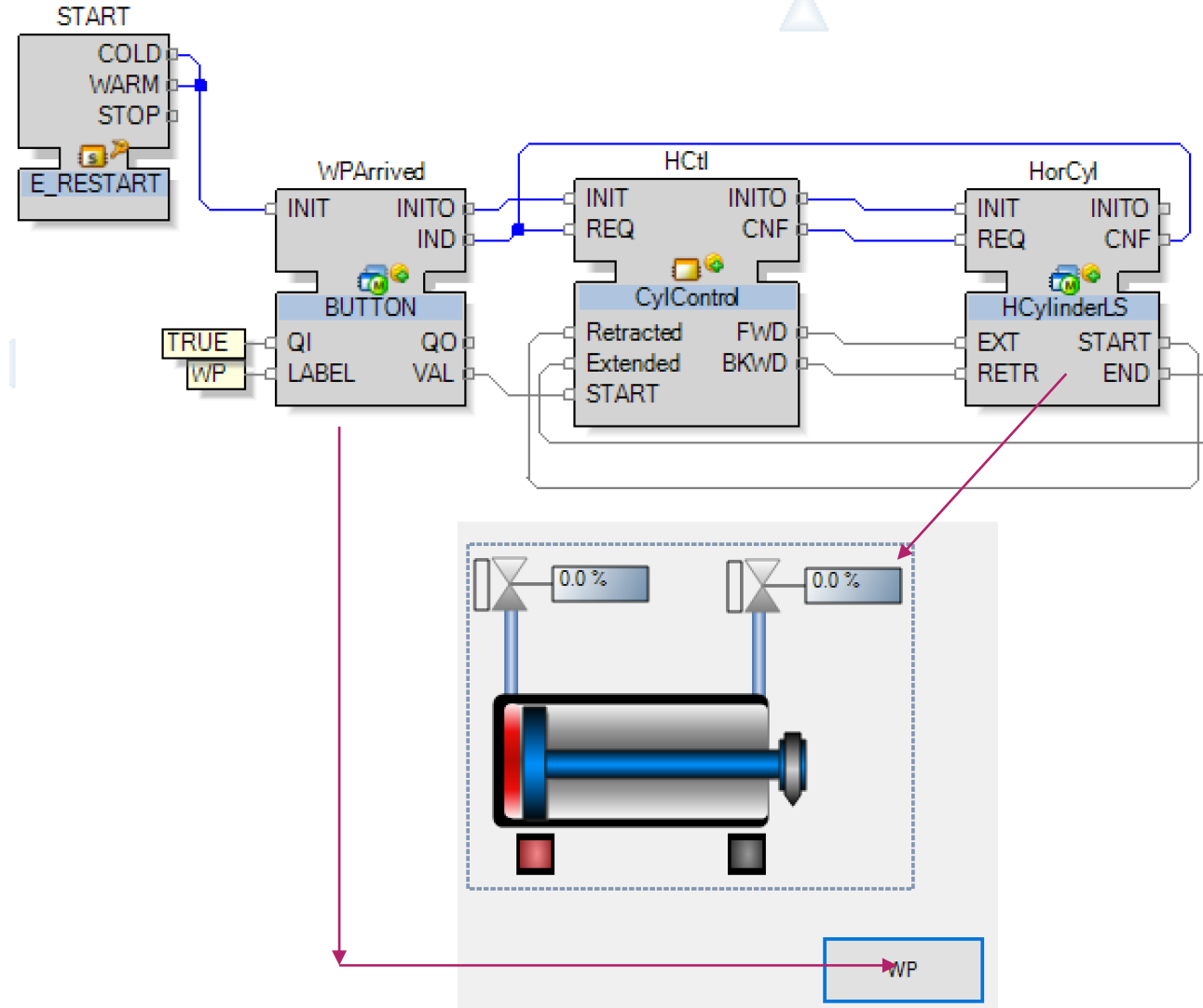
- Software representation of real instruments/devices, e.g. pump
- Predefined control structure with abstract hardware interface
- Automatic HMI configuration and connection



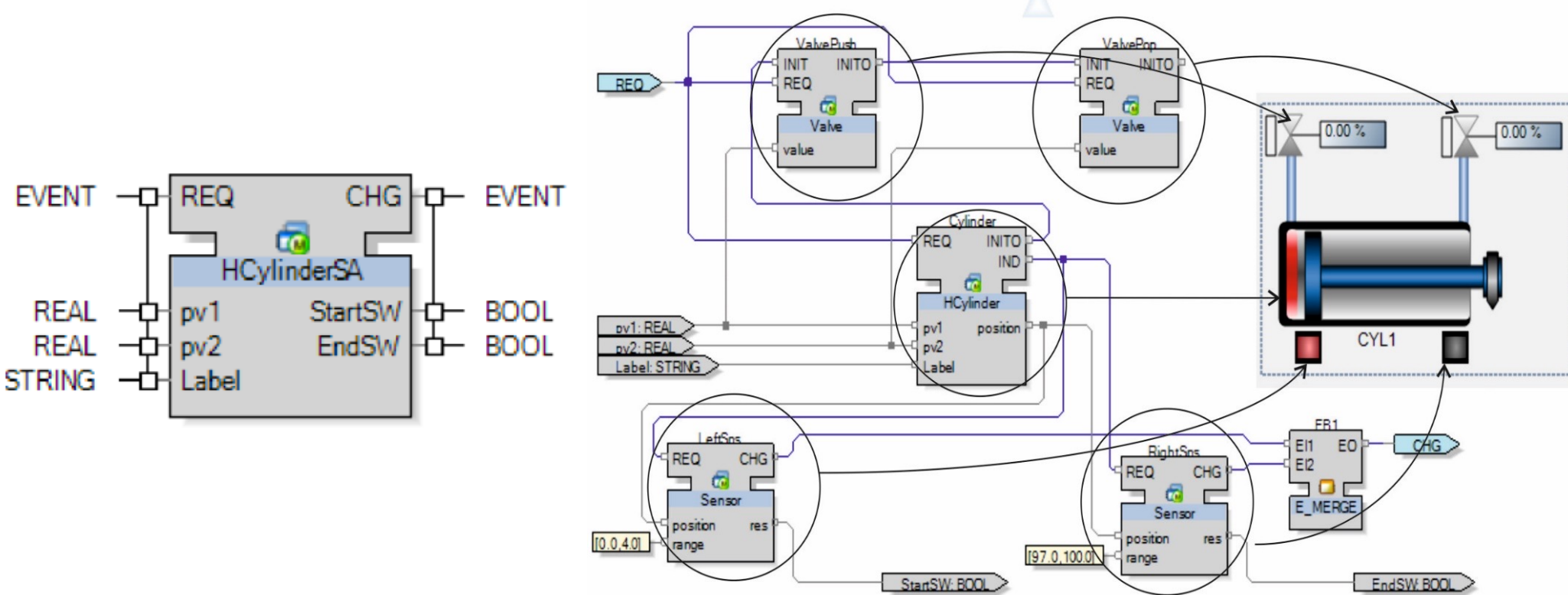
System architecture with an implicit HMI device



Application with CATs



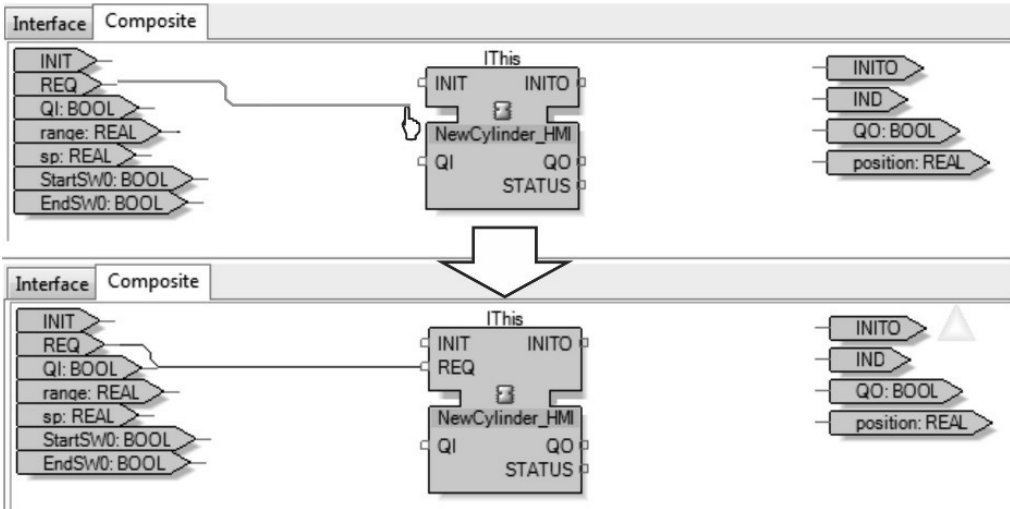
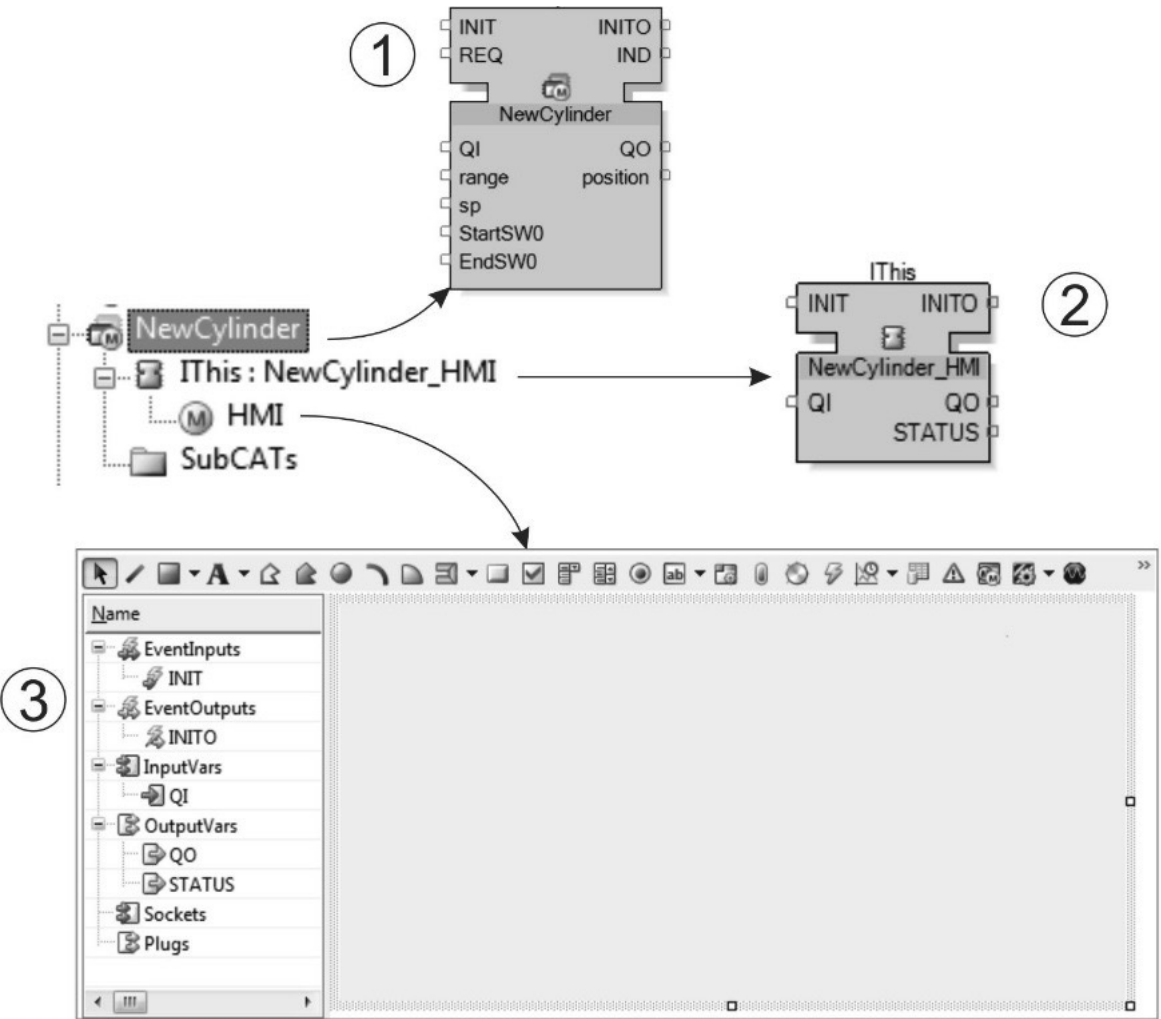
Model of Composite Cylinder



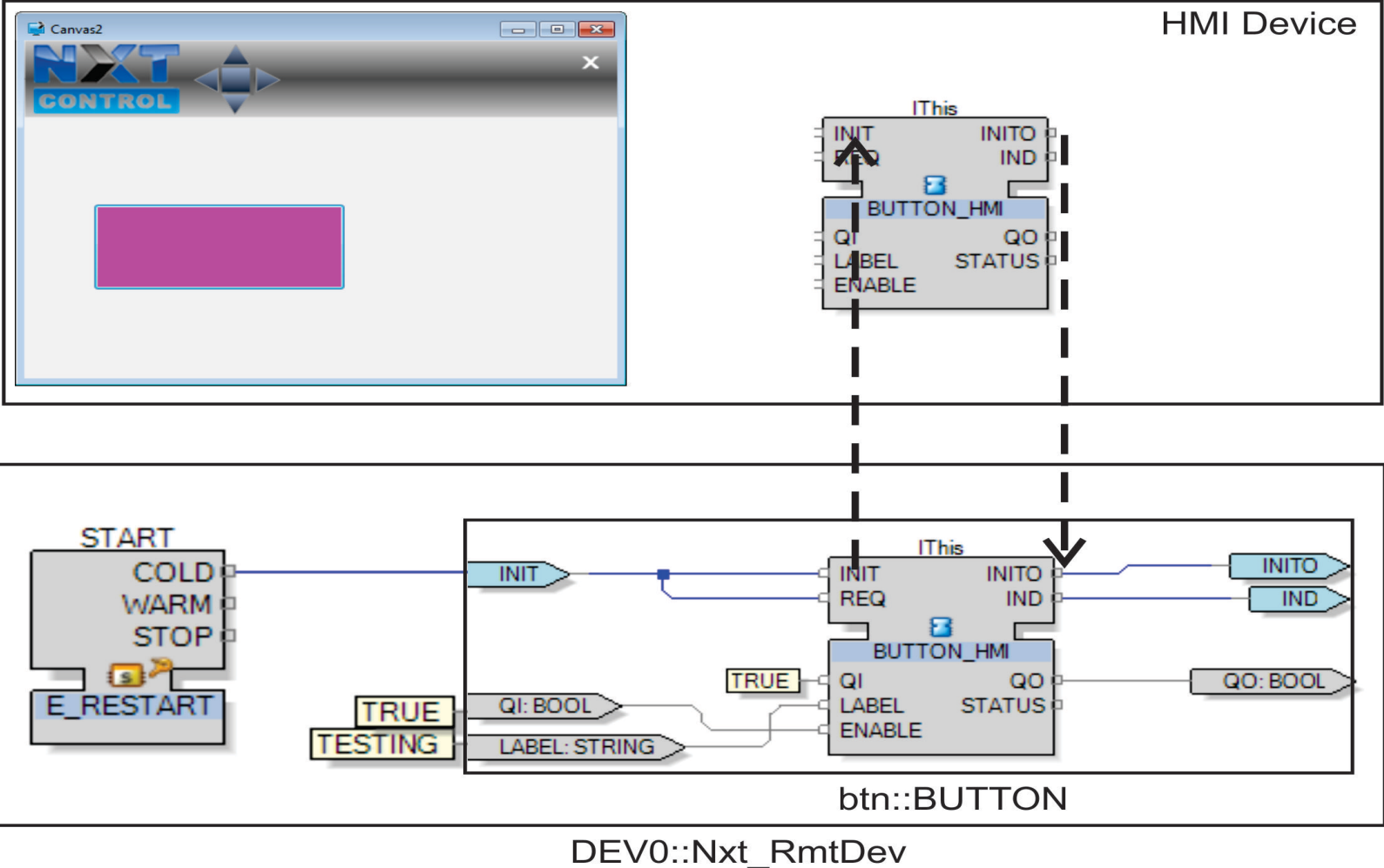
21/05/2024

84

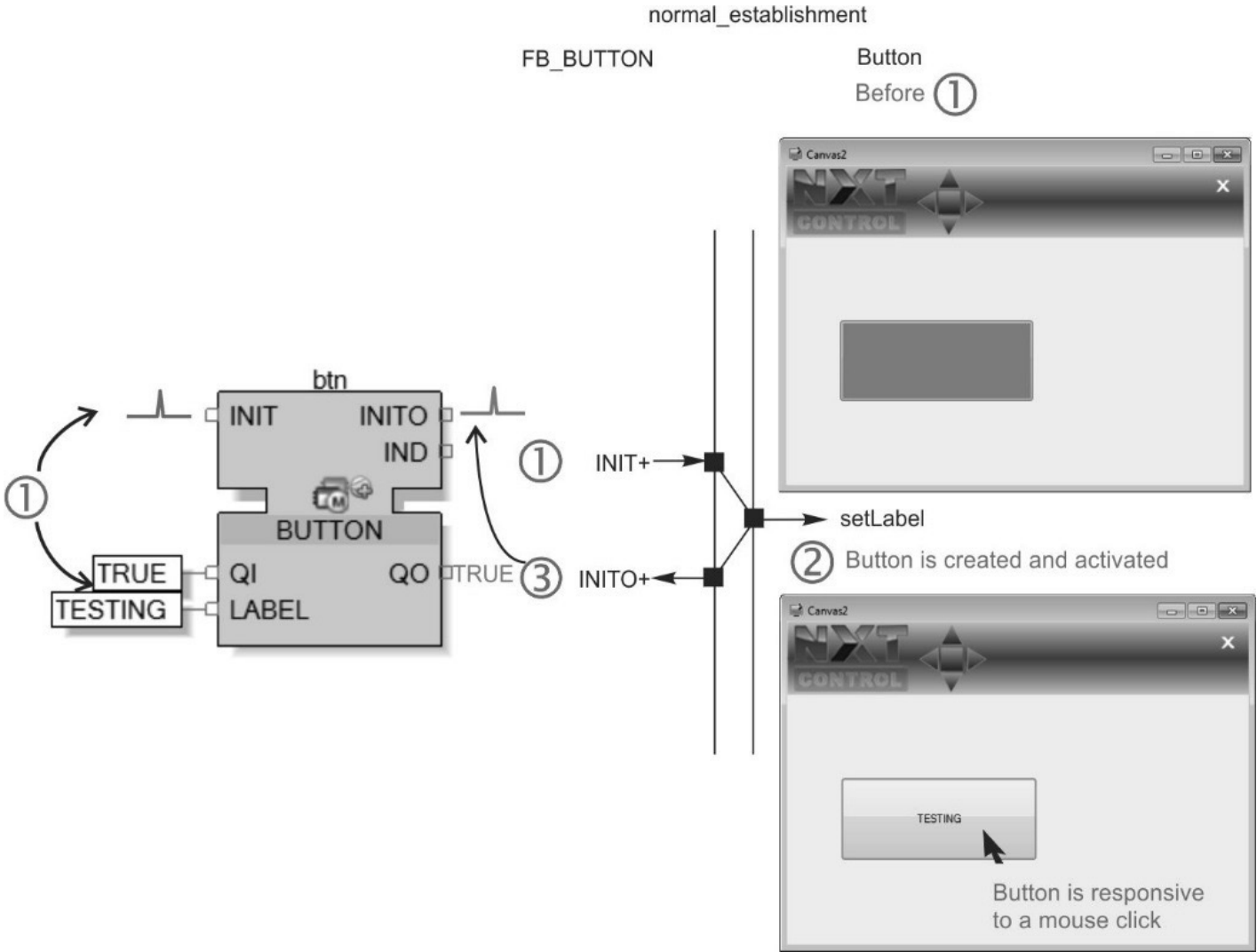
CAT creation



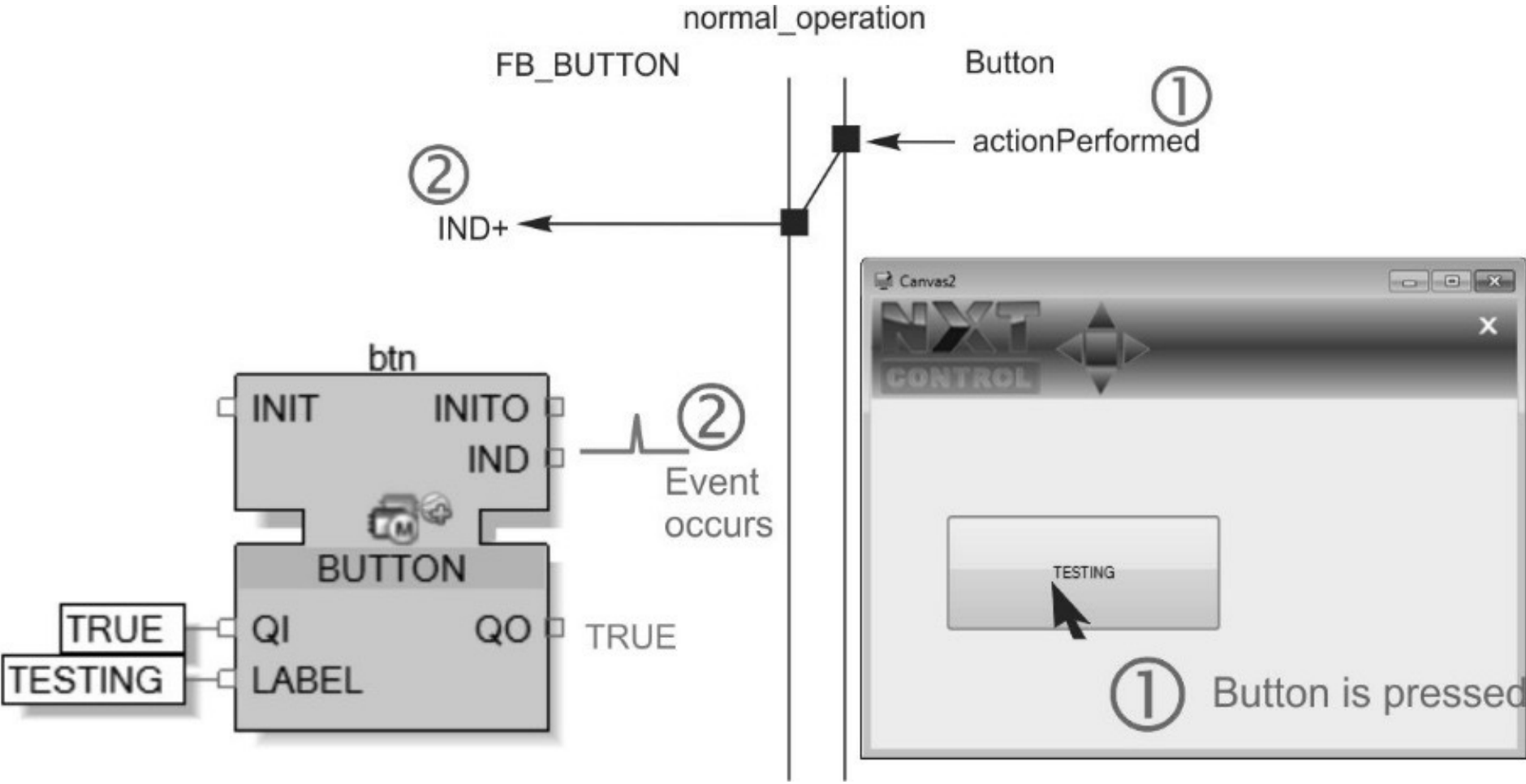
How does a CAT work?



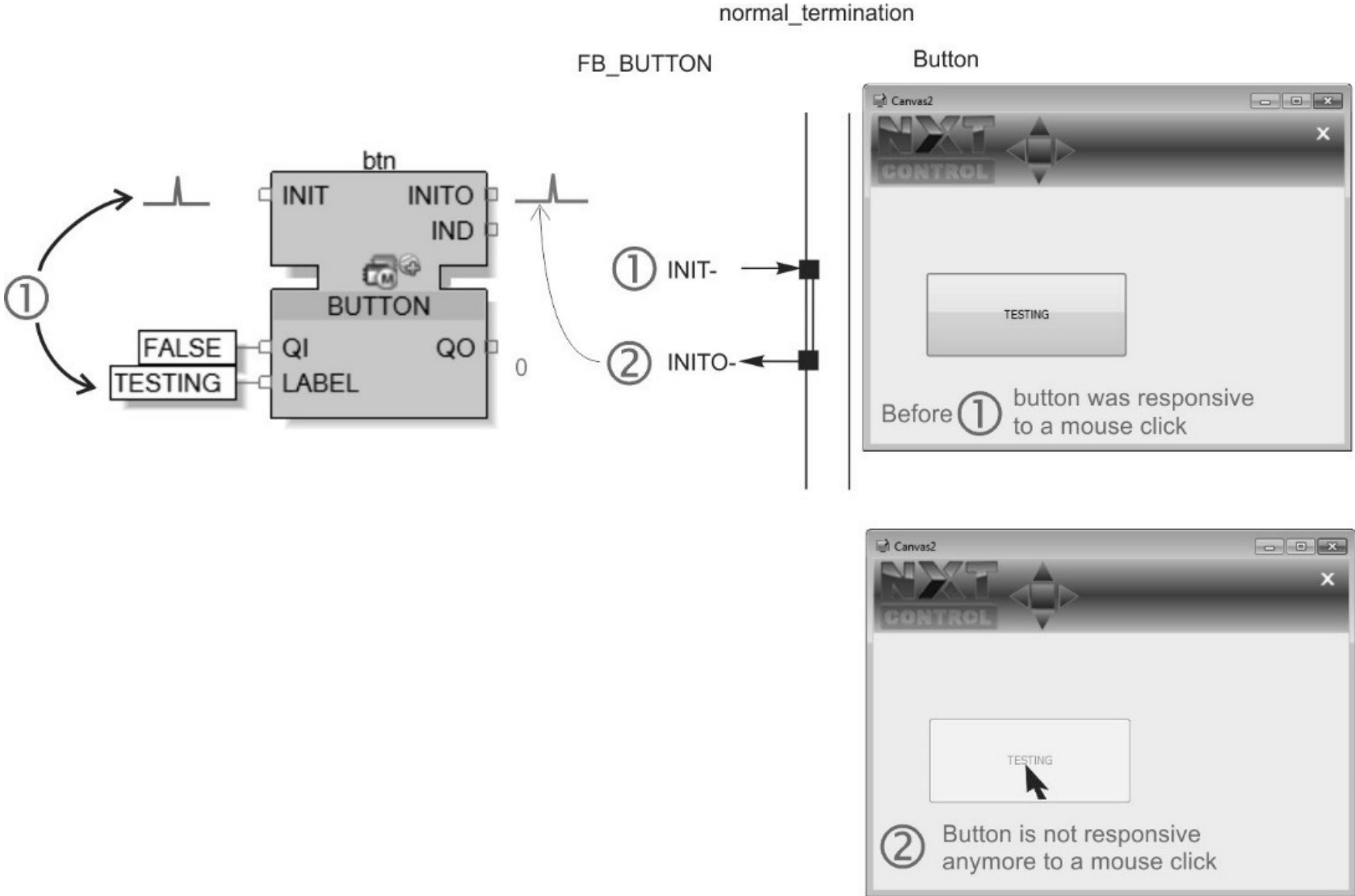
Normal establishment



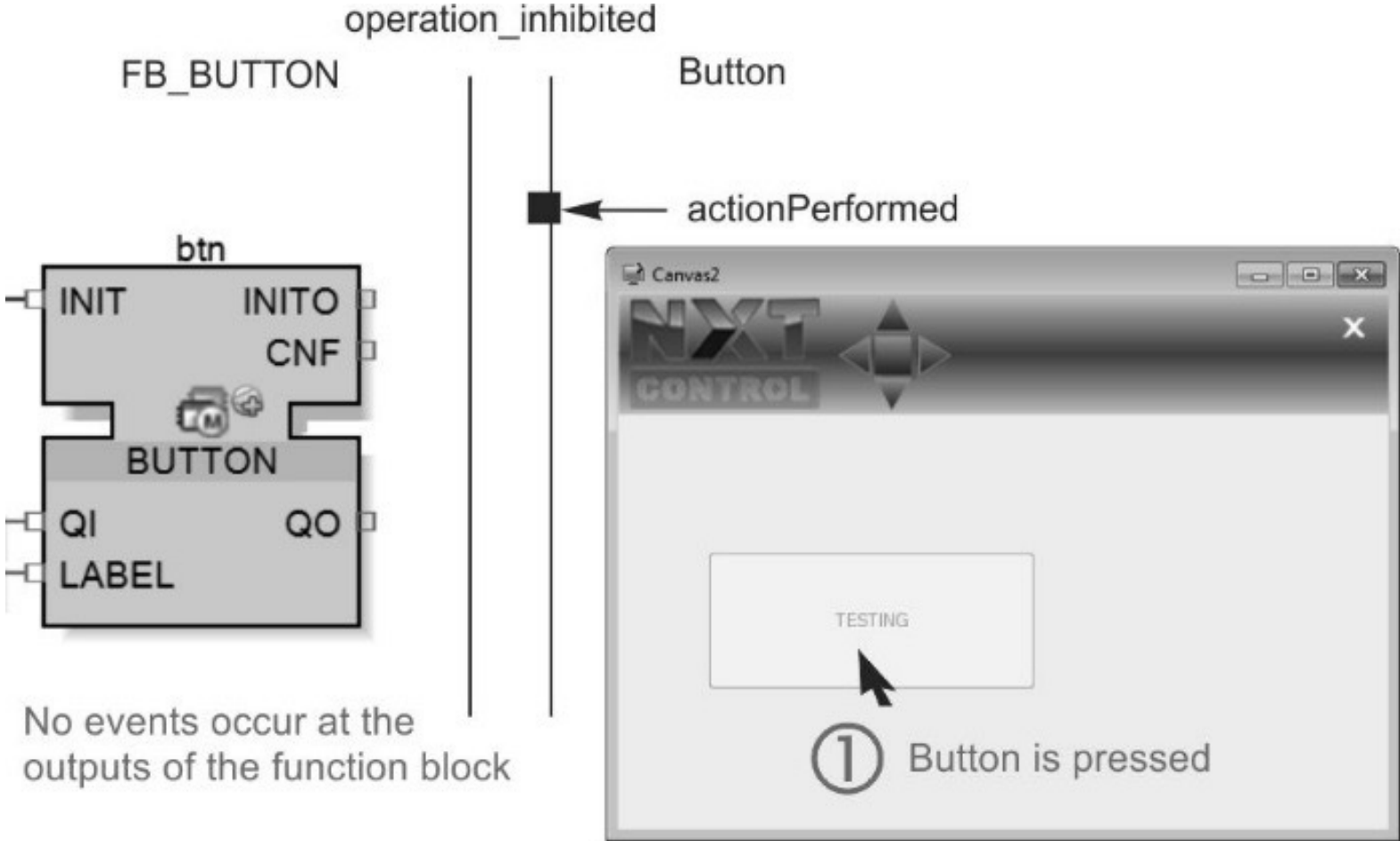
Normal operation



Normal termination

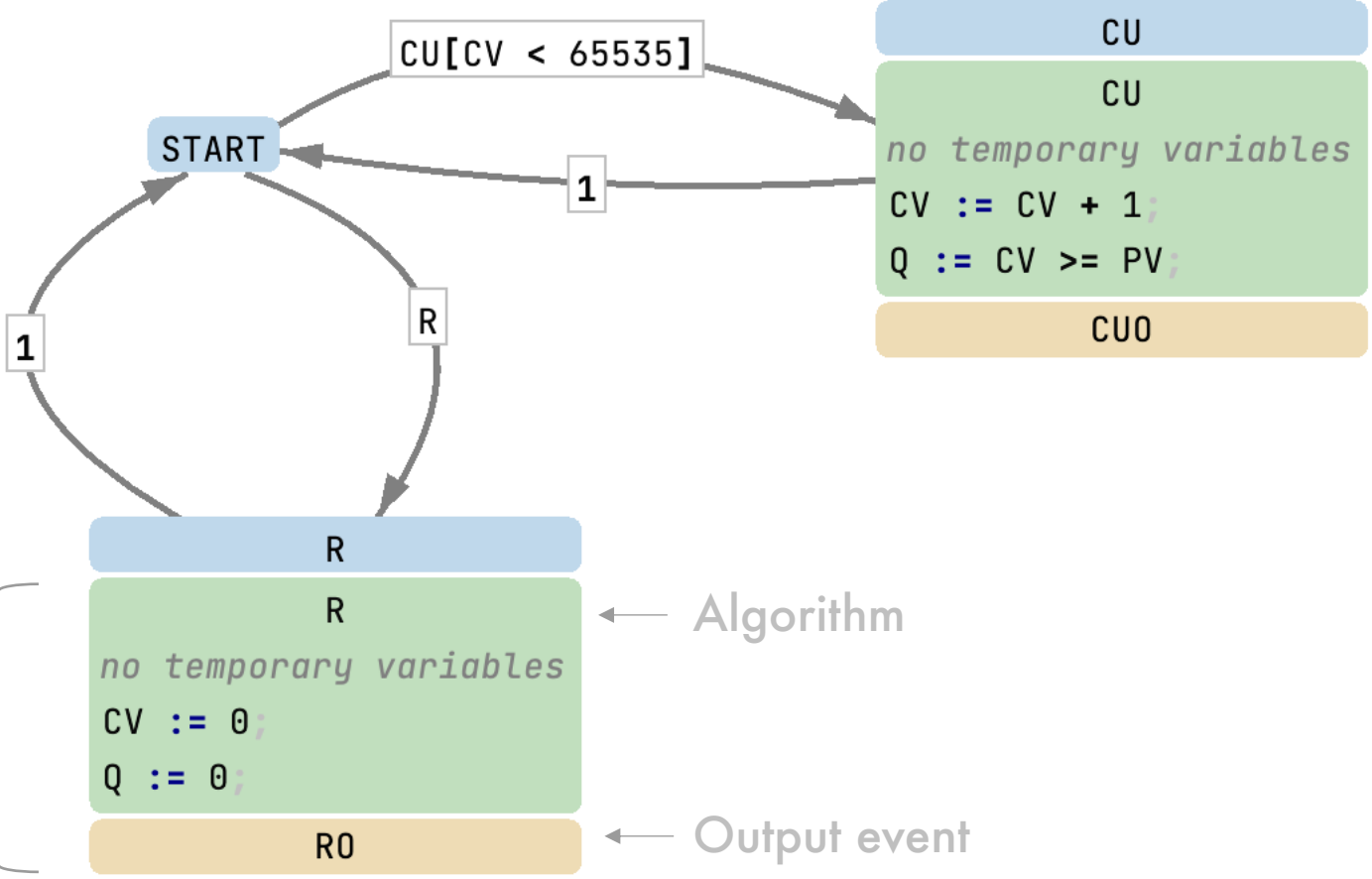
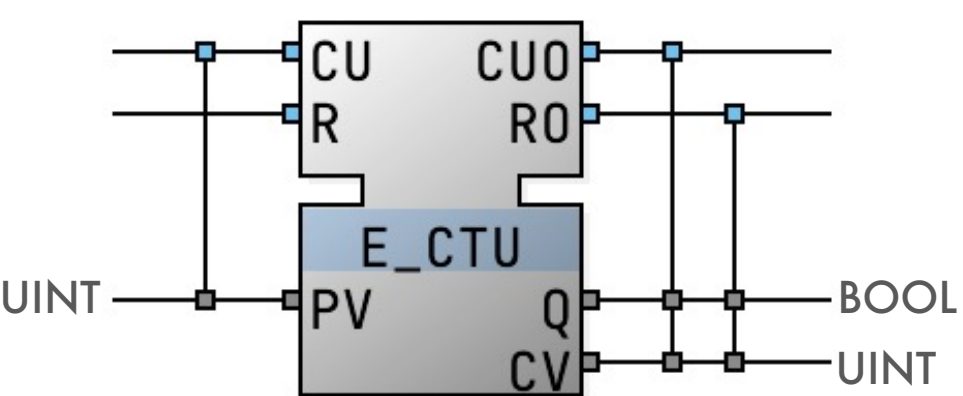


Operation is inhibited



Event-handling Function Blocks

Count UP

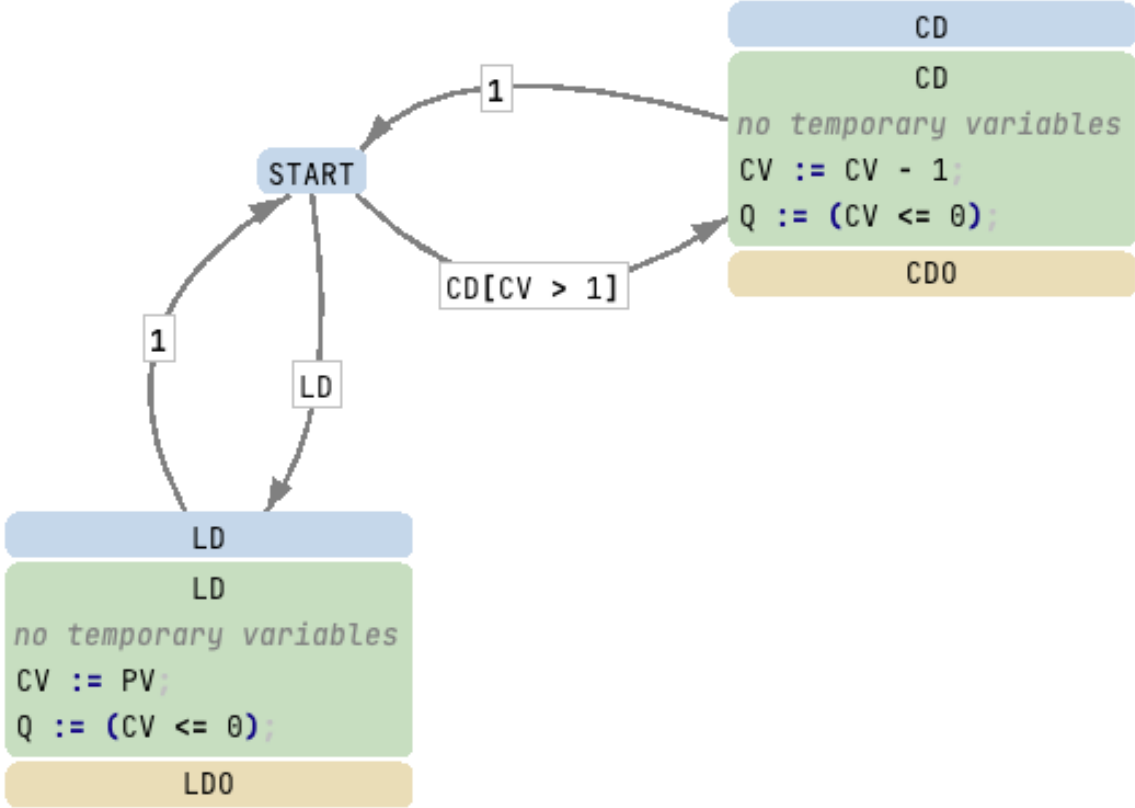
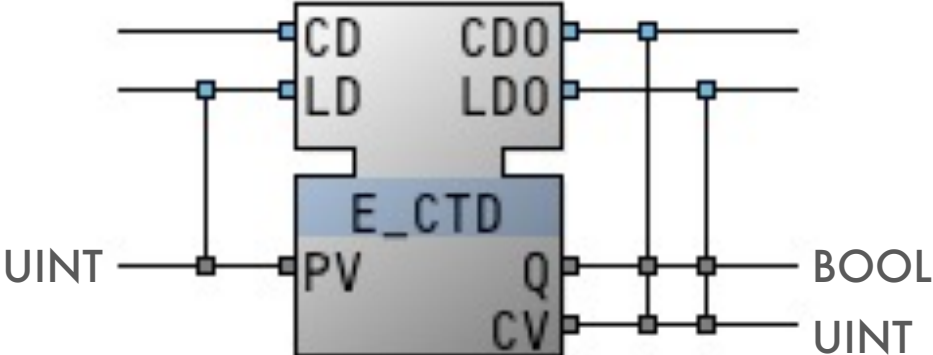


Action

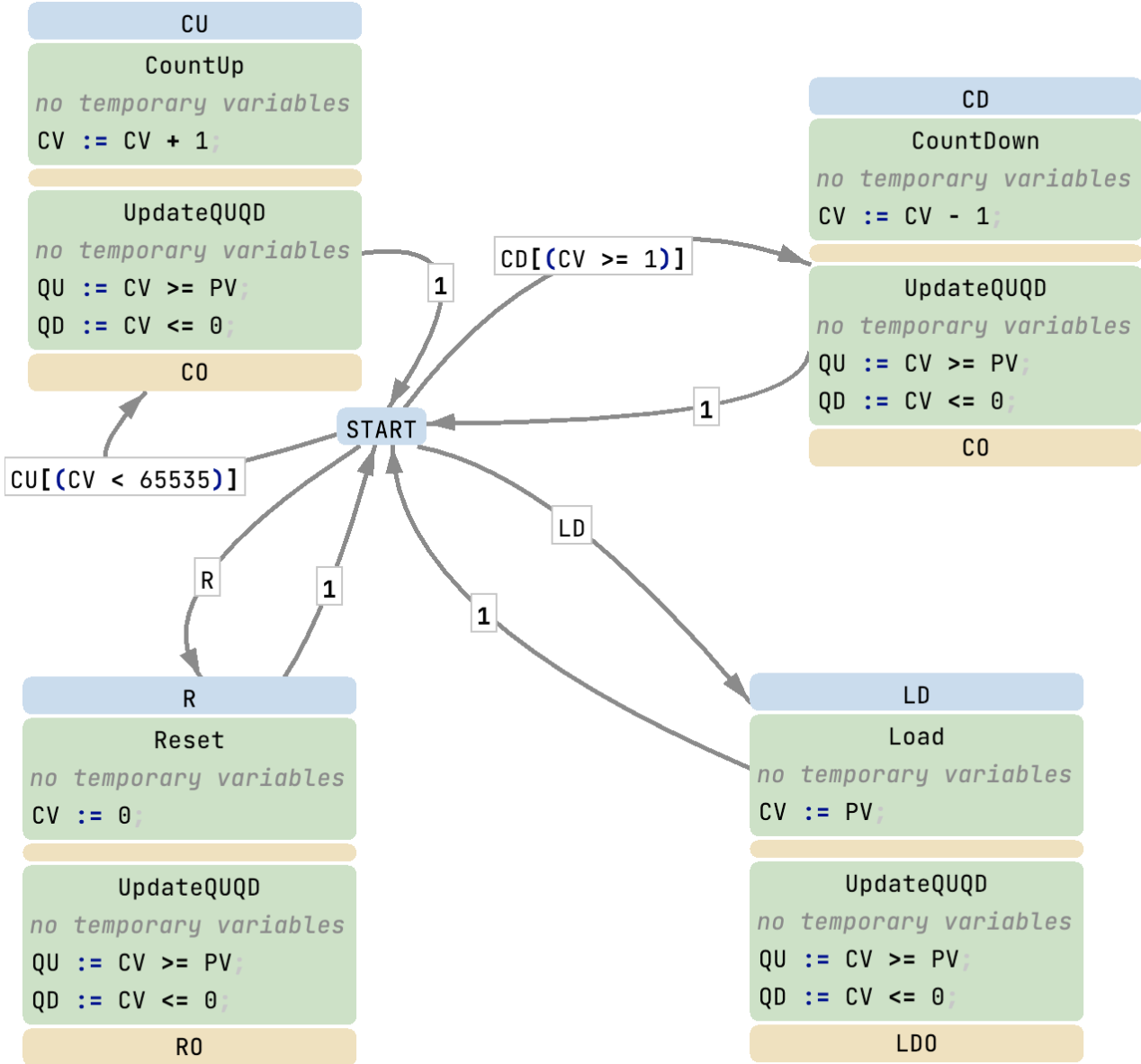
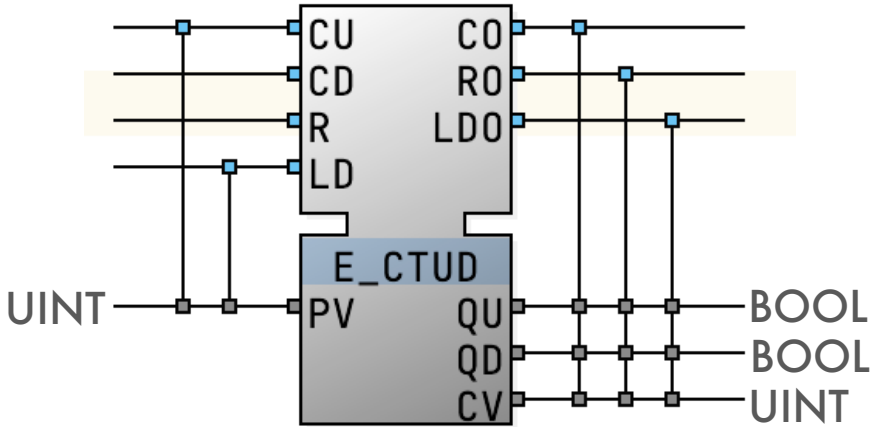
Algorithm

Output event

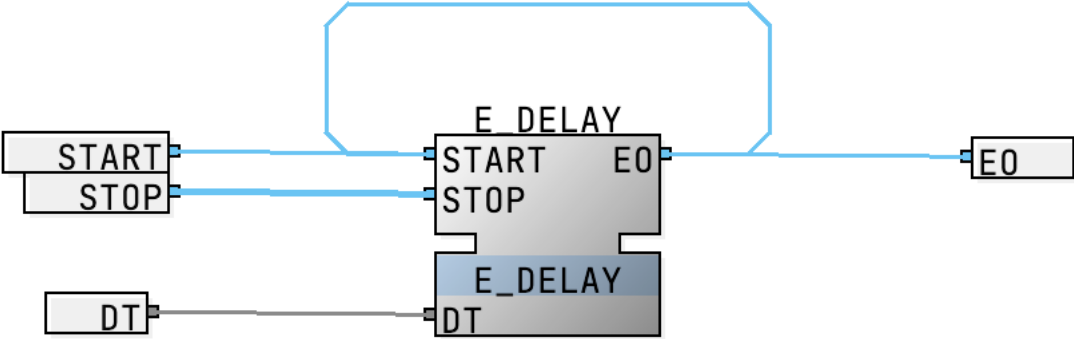
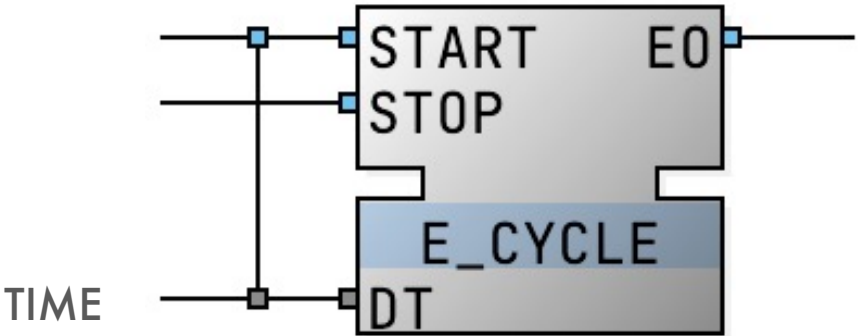
Count down



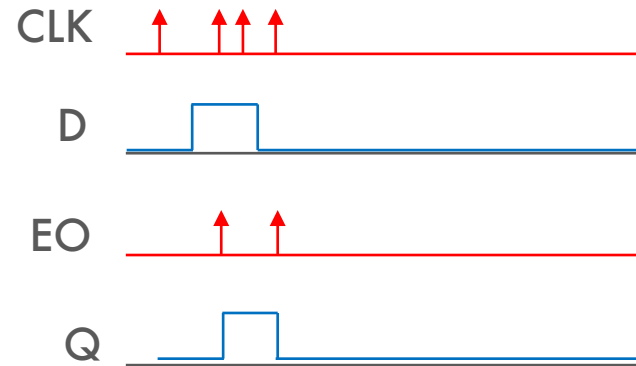
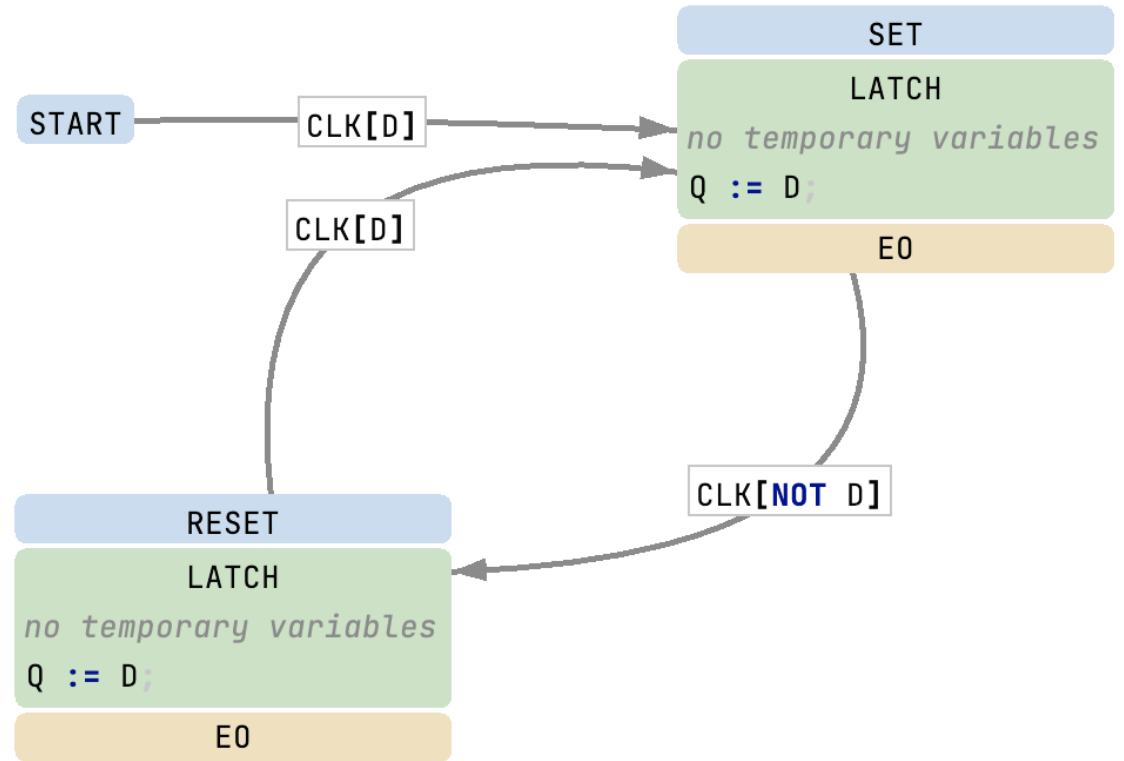
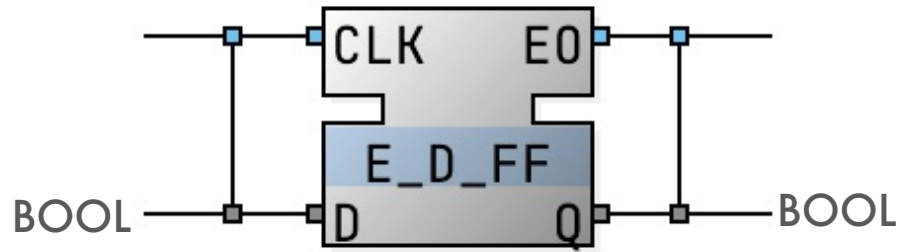
Count UP/Down



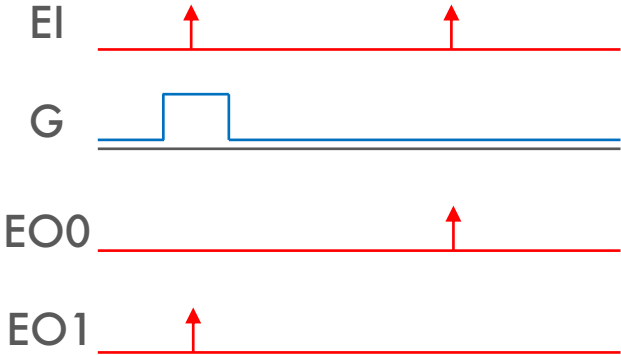
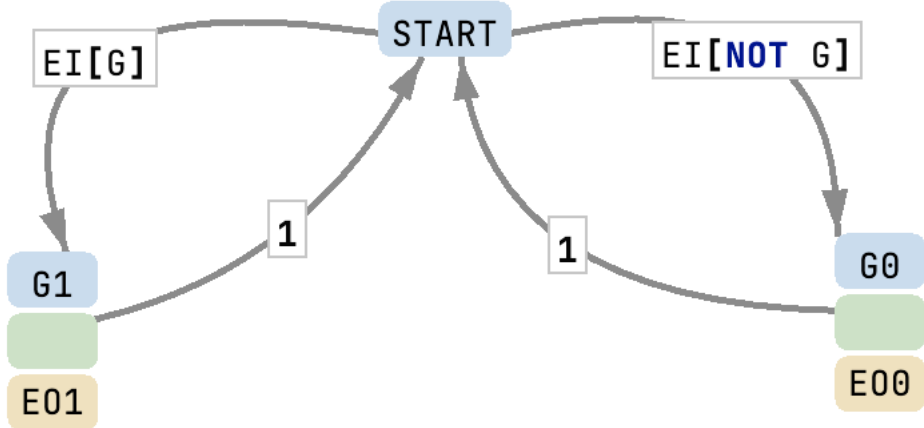
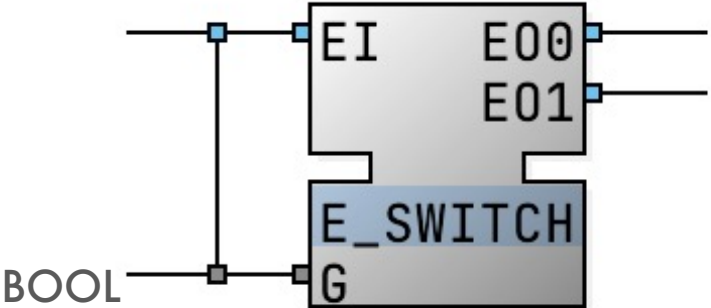
E_CYCLE



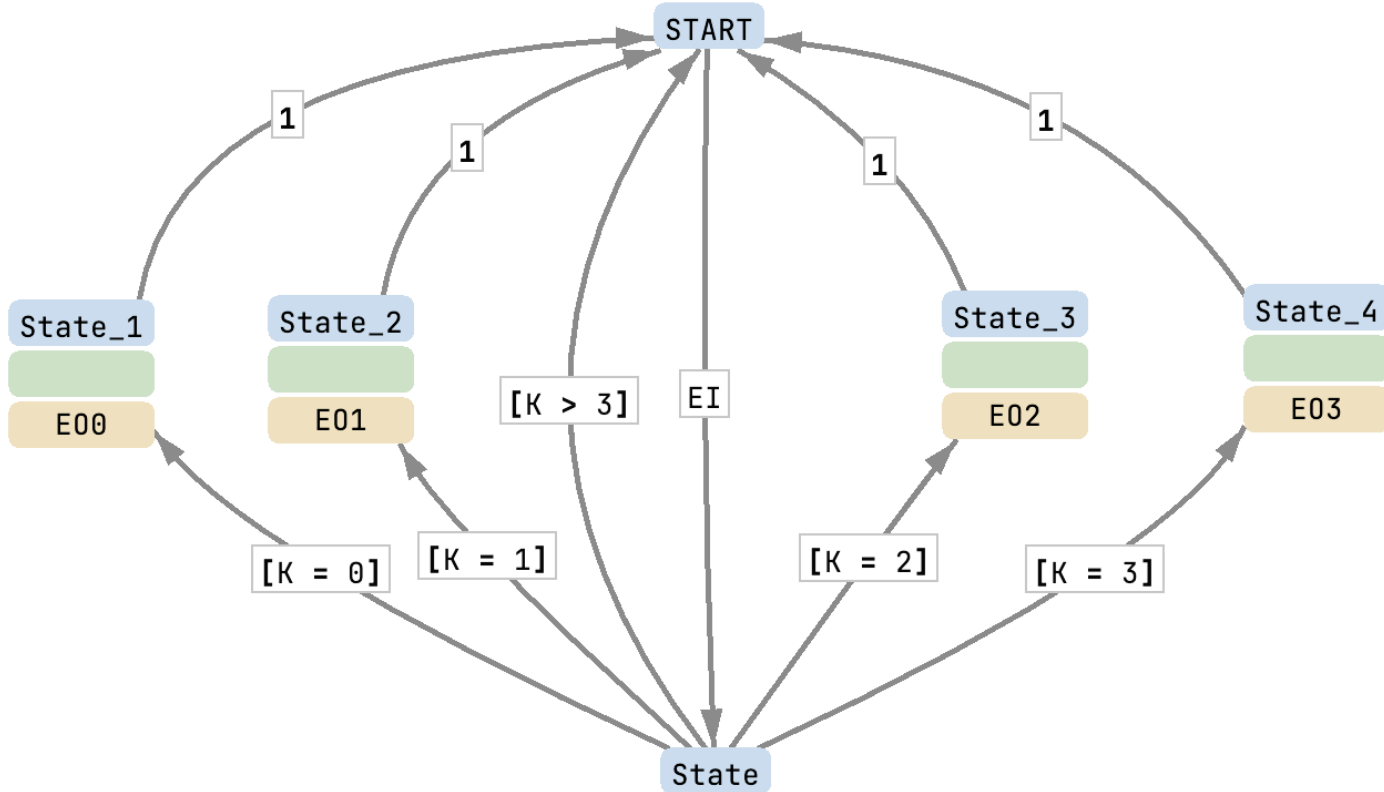
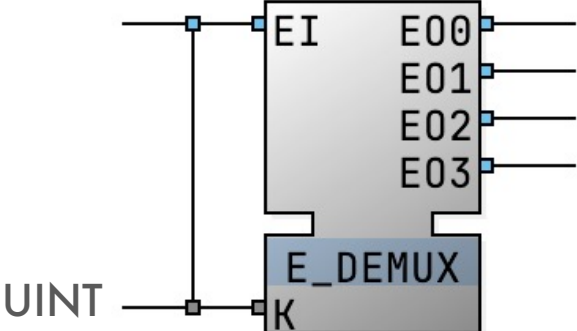
Data latch



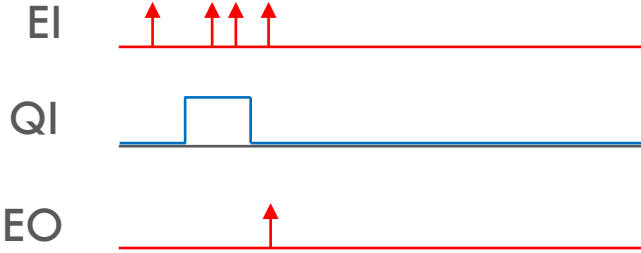
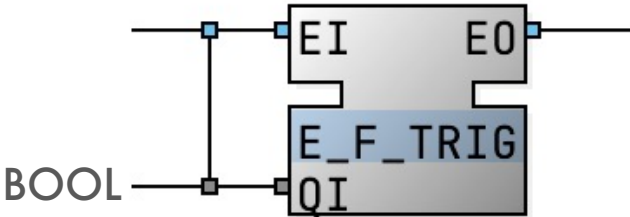
Event switch E_SWITCH



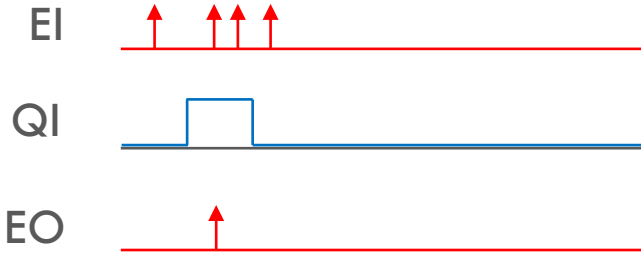
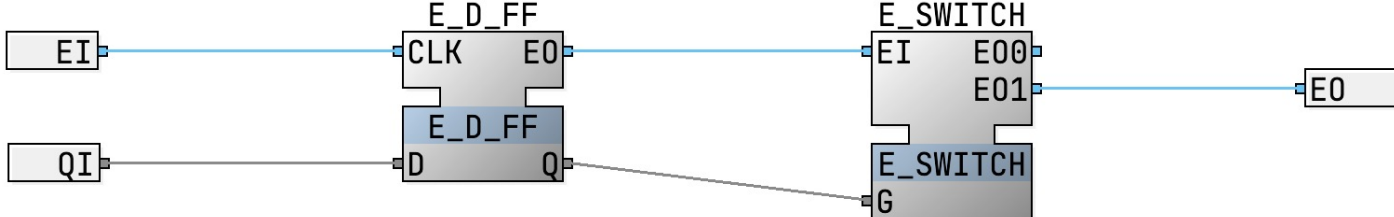
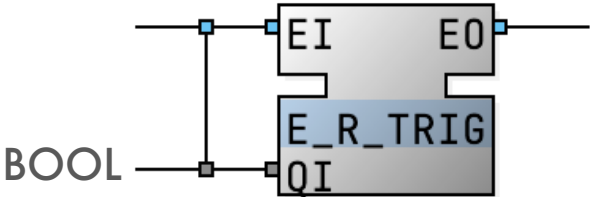
Event demultiplexor



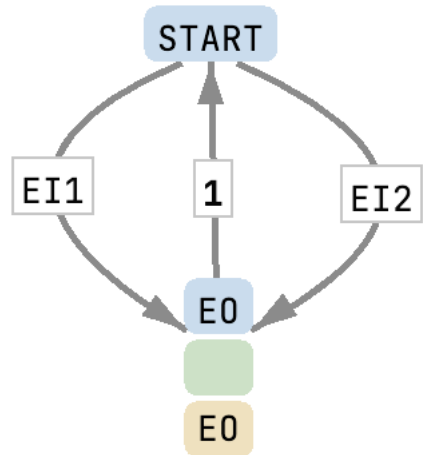
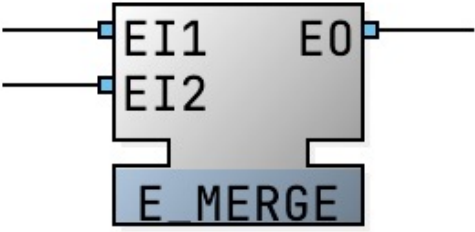
Boolean falling edge detection



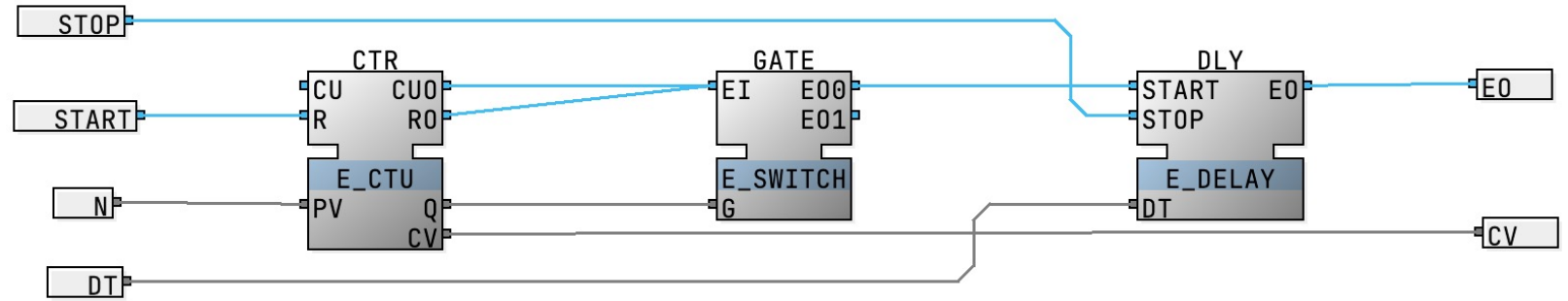
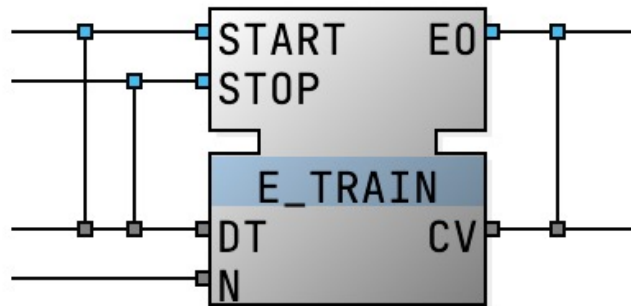
Boolean raising edge detection



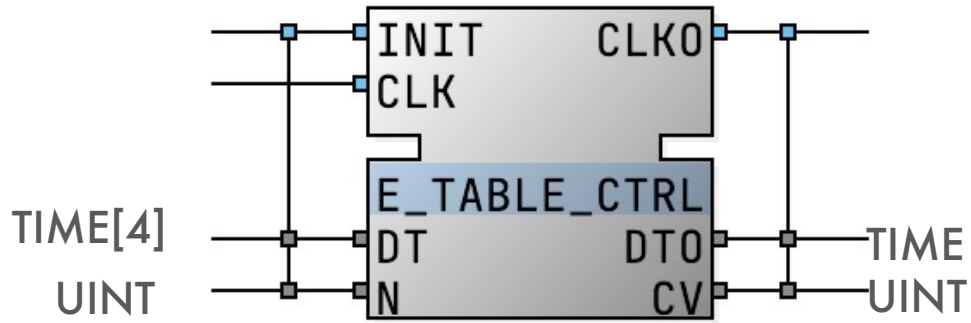
E_MERGE



Train of events



E_TABLE_CTRL



In this example $N \leq 4$

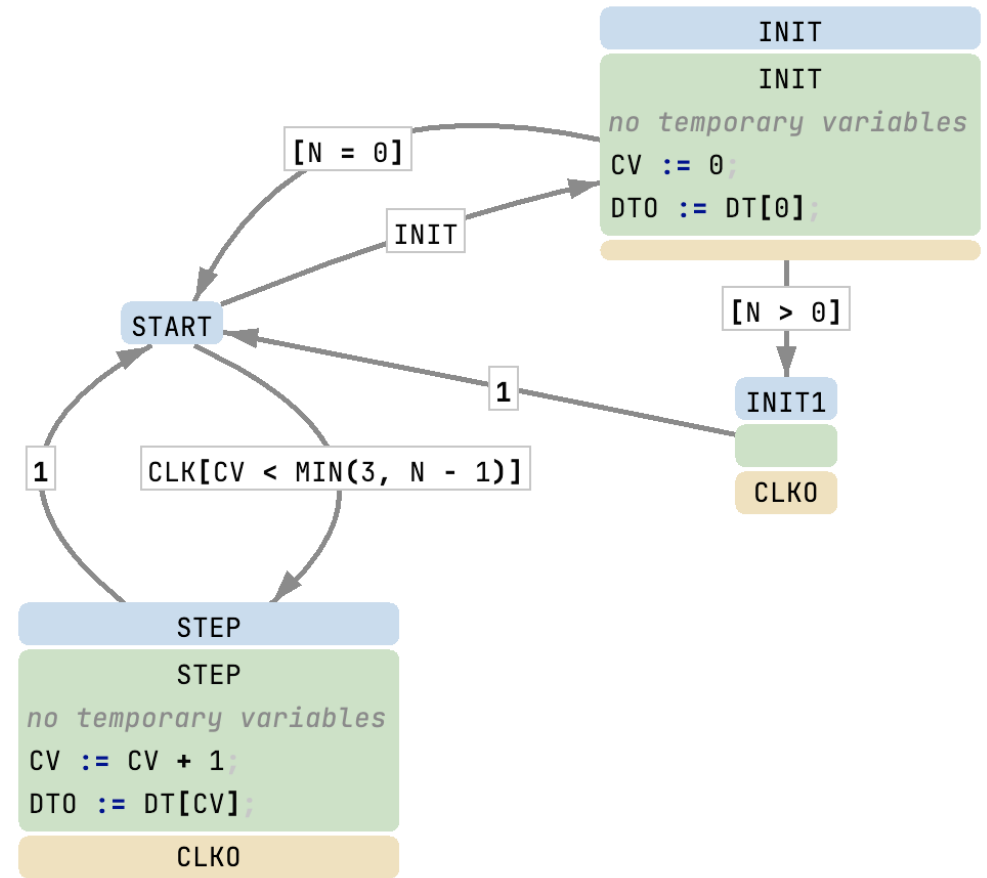
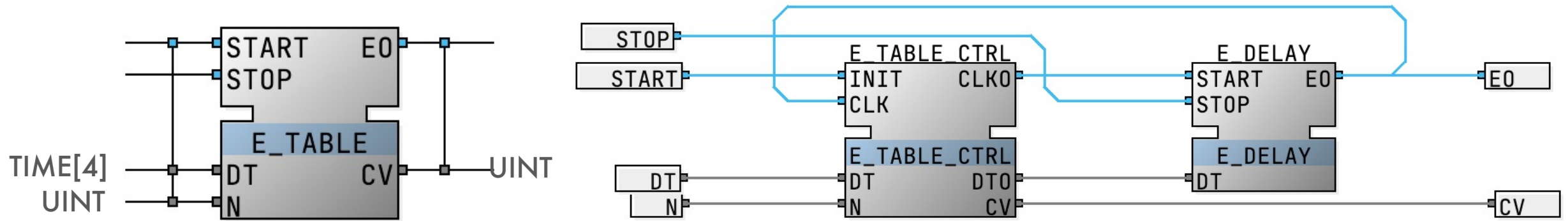
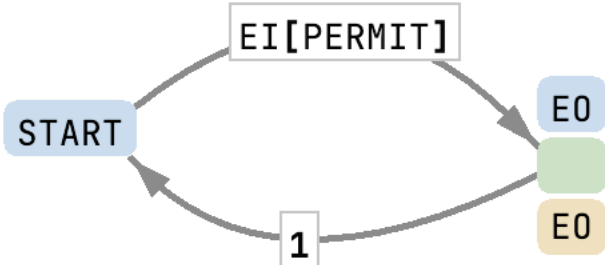
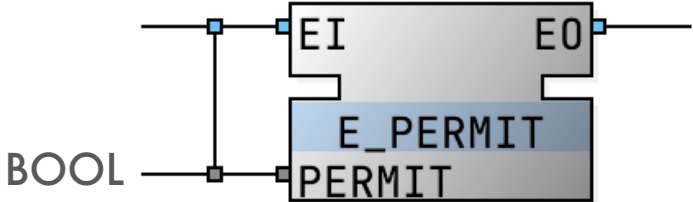
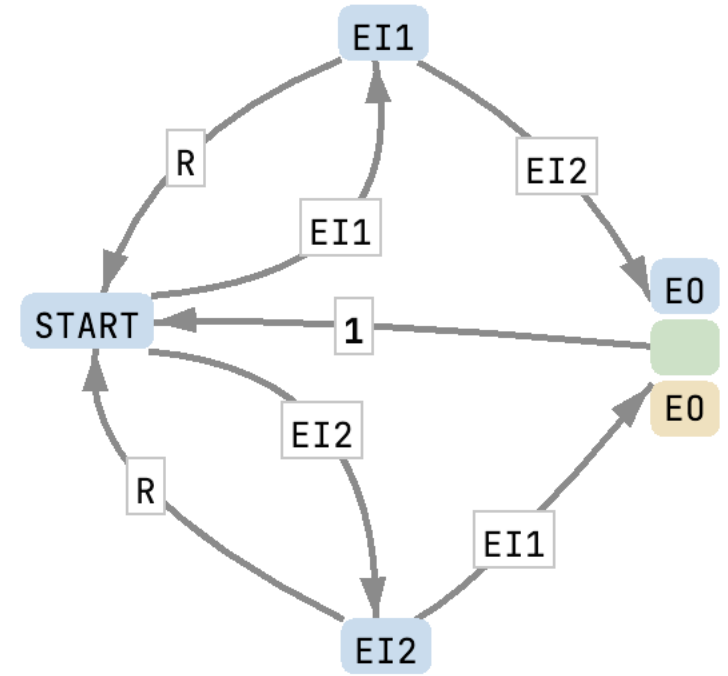
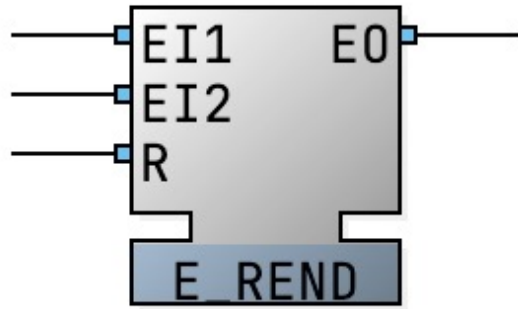


Table-driven sequence of events

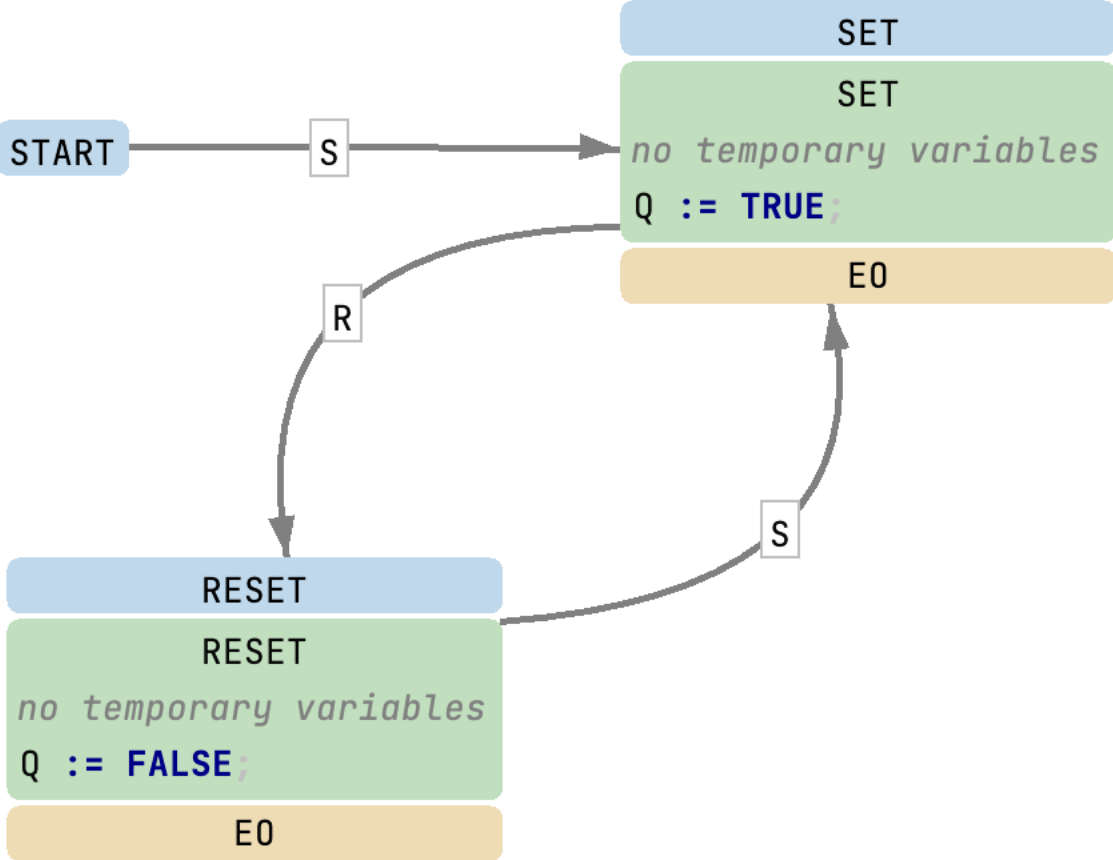
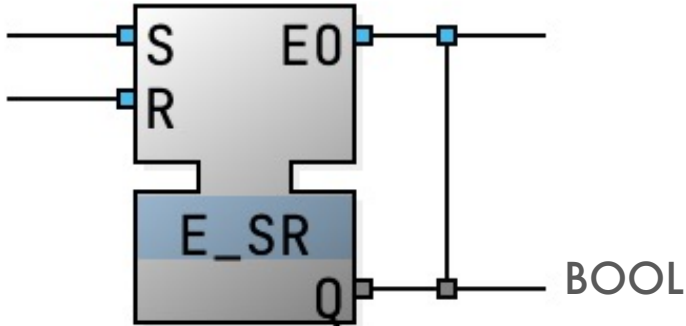
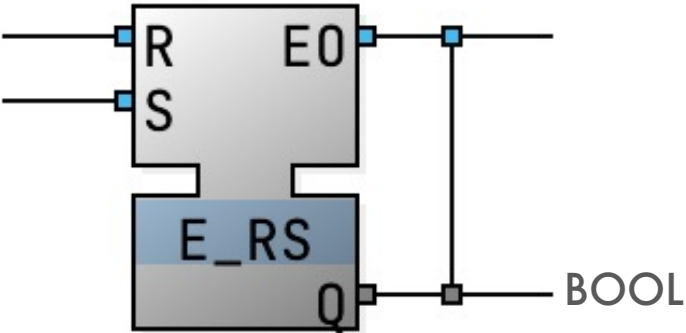


Controlled Event propagation E_PERMIT

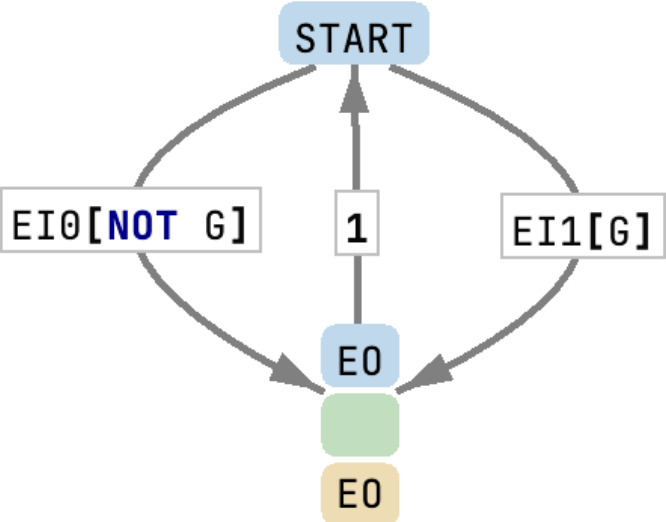
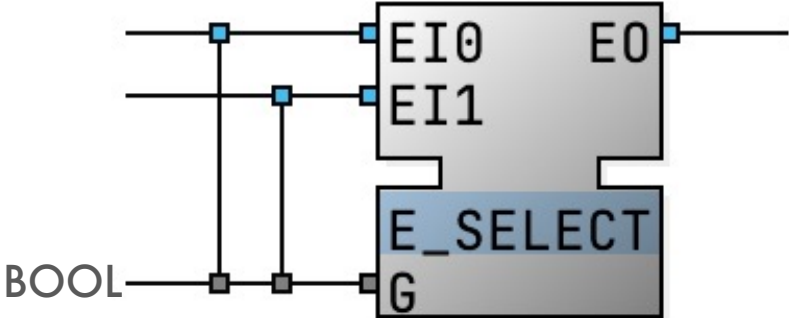




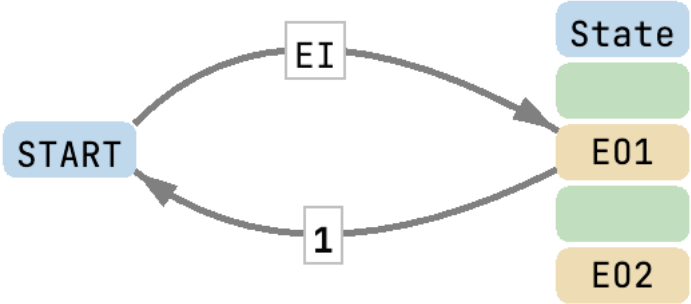
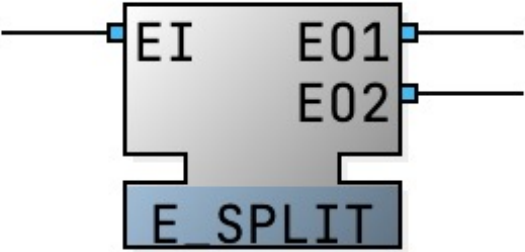
RS, SR triggers



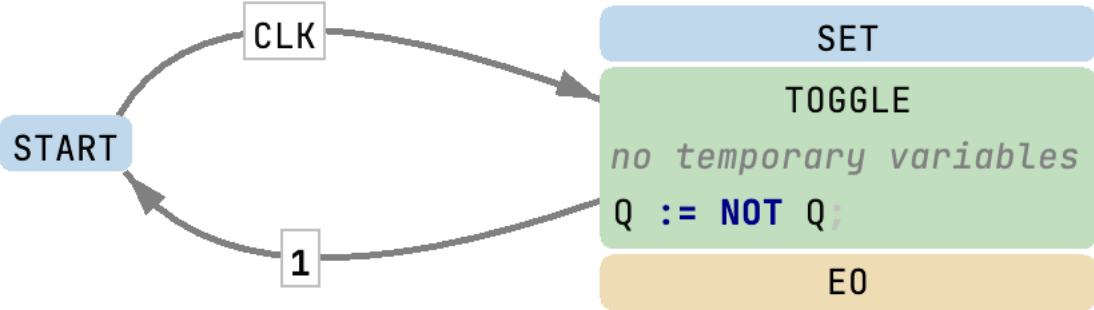
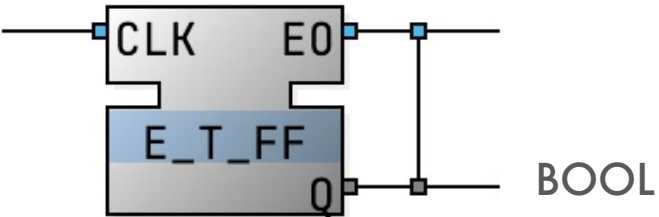
Event selector



Event splitter



Event driven toggle of a Boolean output



UNIVERSAL
AUTOMATION.ORG