

School of Electrical Engineering

Department of Electrical Engineering and Automation

ELEC 8201 Control & Automation

Programming in IEC 61499

Valeriy Vyatkin

IEC 61499 International Standard

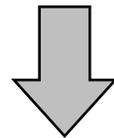
International Electrotechnical Commission IEC TC 65B/ WG7/ MT15

A component-based, open reference architecture for
Distributed Industrial - Process Measurement & Control Systems (IPMCS)
which can meet both current and future requirements for intelligent automation

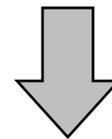
1996 – project started

2005 – first edition

2011 – second edition



Based on and extends
the standards



PLC Function Blocks (IEC 61131-3)

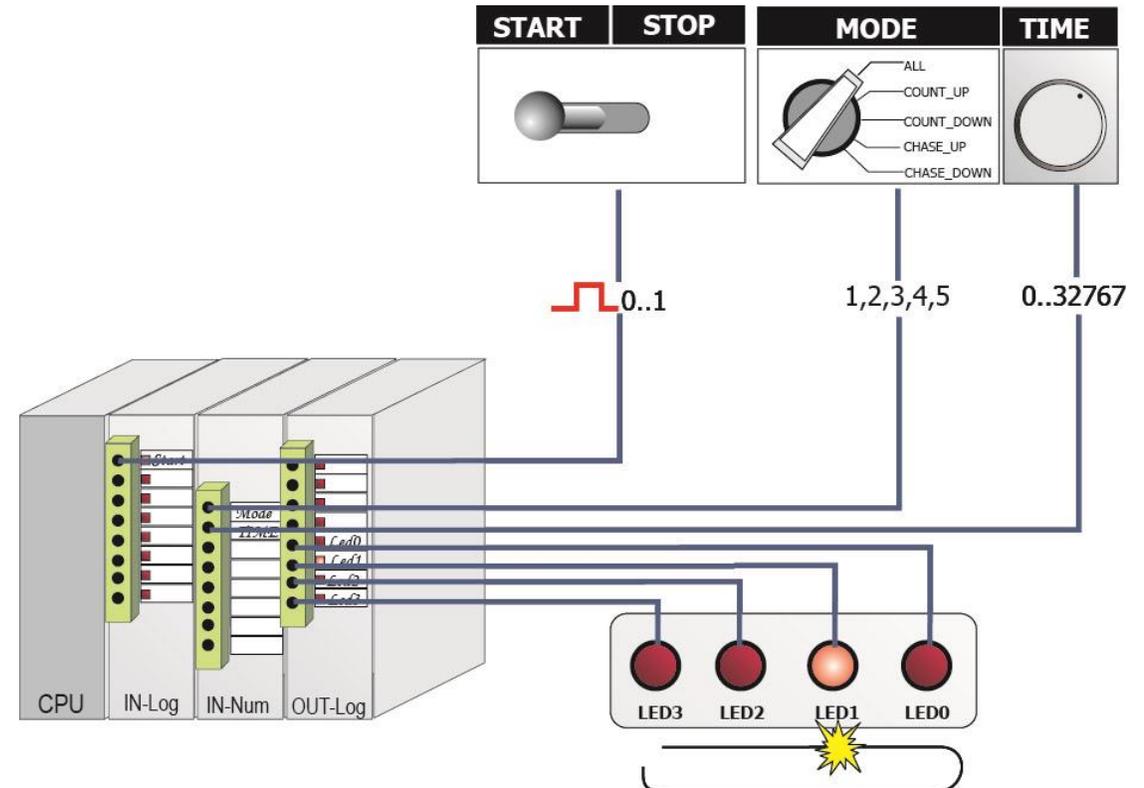
DCS Function Blocks (IEC 61804 project)

Example: FLASHER

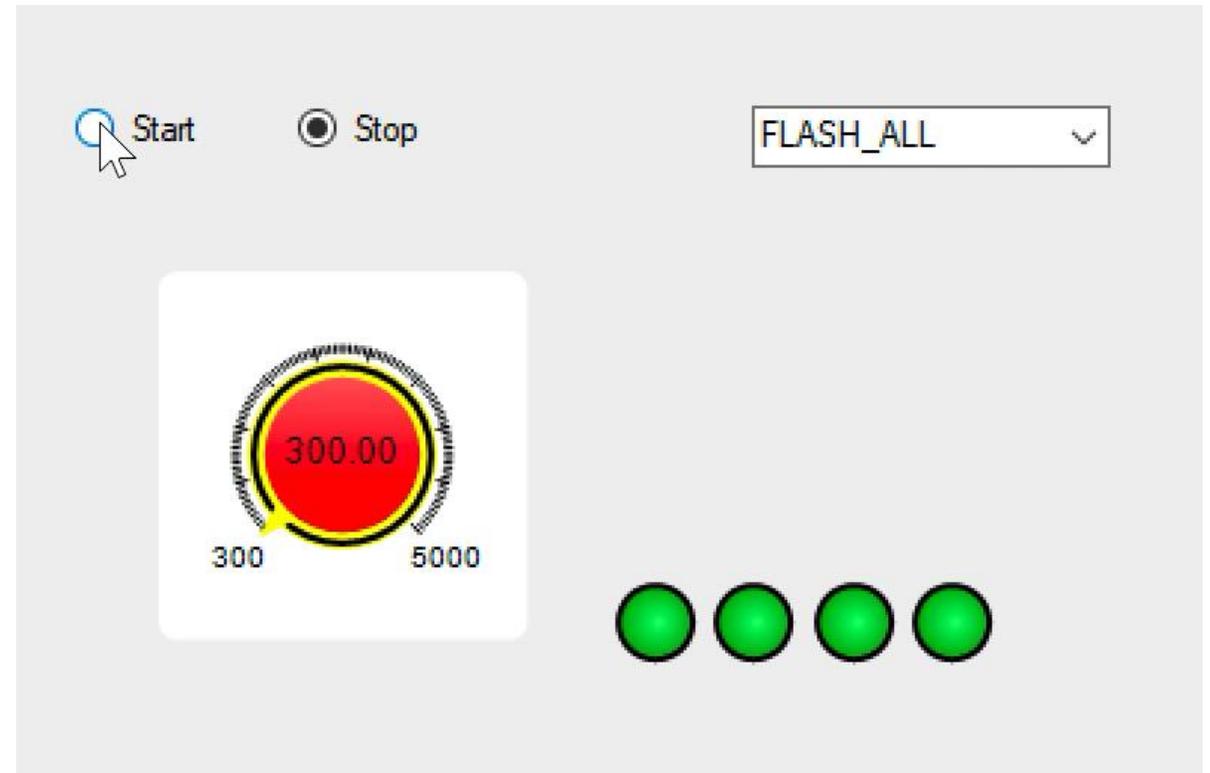
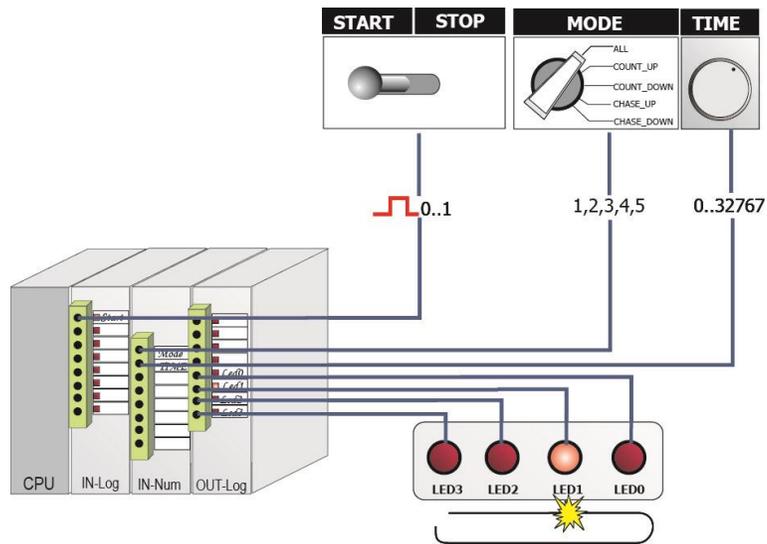
Consider a simple product Flashing Lights

Components of the physical system:

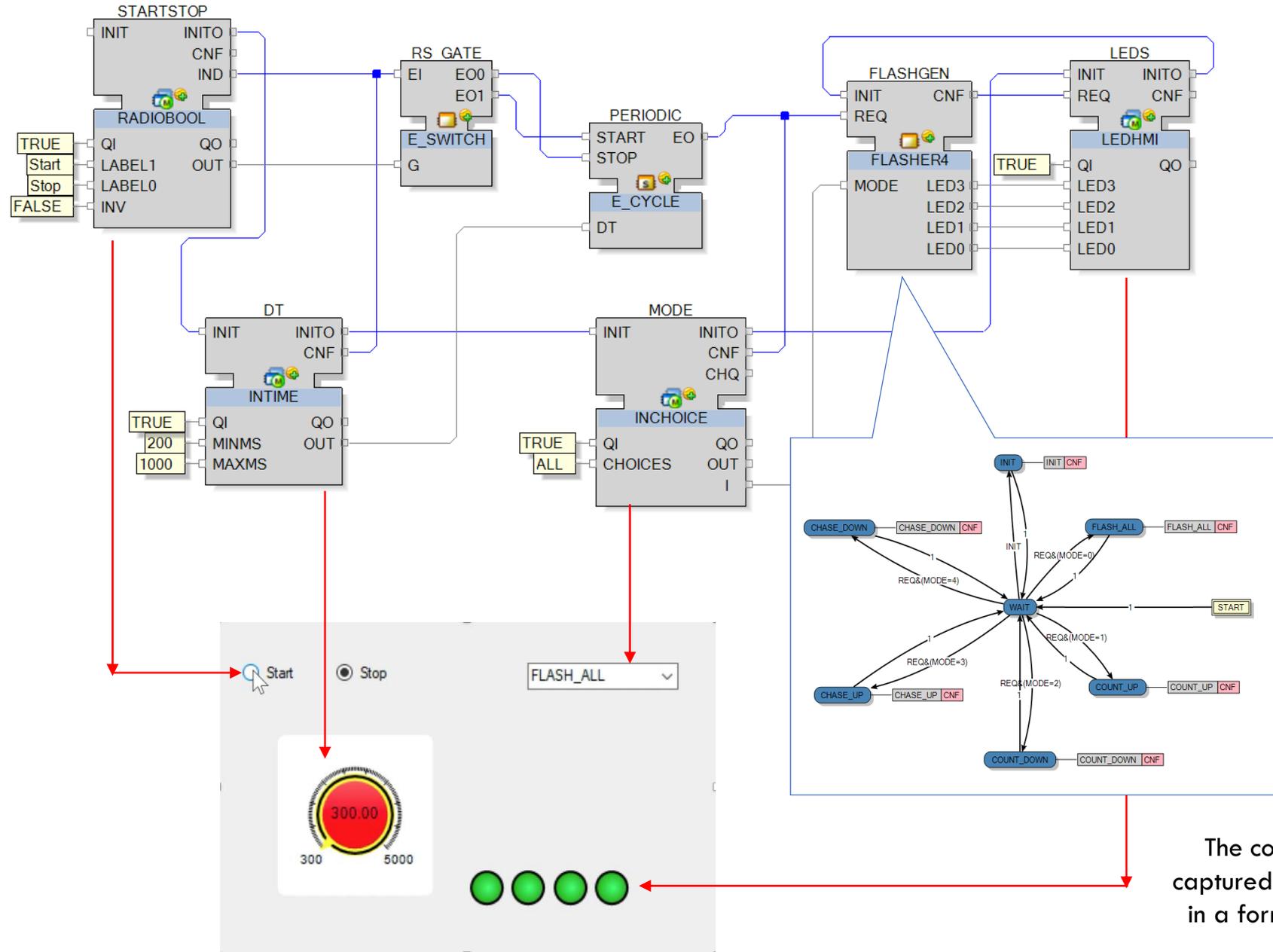
1. PLC (CPU)
2. START/STOP Button
3. Mode Selection Switch
 - All lights on, count up, count down, chase up and chase down
4. Time Delay Nub
 - Delay between 2 consecutive light flashes
5. 4 LEDs



FLASHER modelled in software



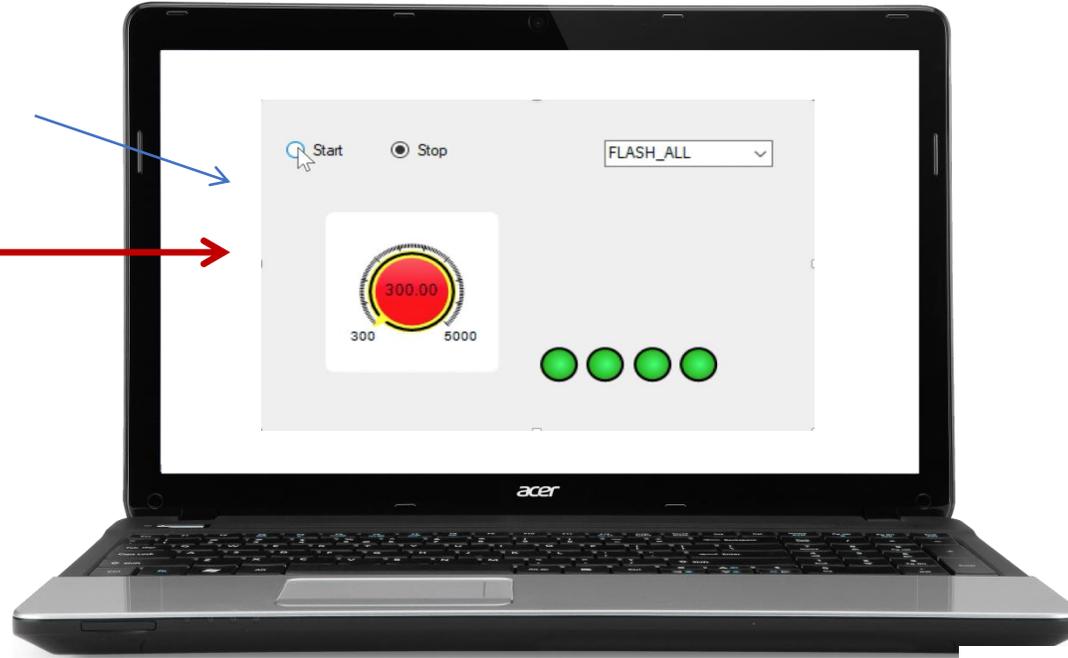
FLASHER application



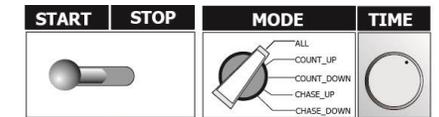
The core functionality is captured in the FLASHER4 FB in a form of state machine

Model and simulate control logic with IEC 61499

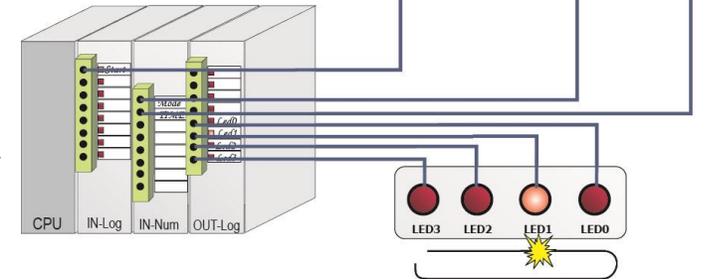
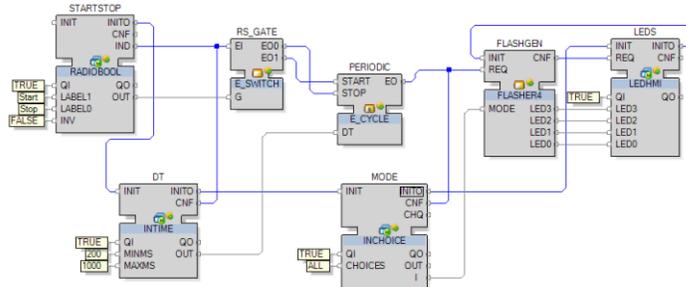
HMI
Runs on a computer



Physical system

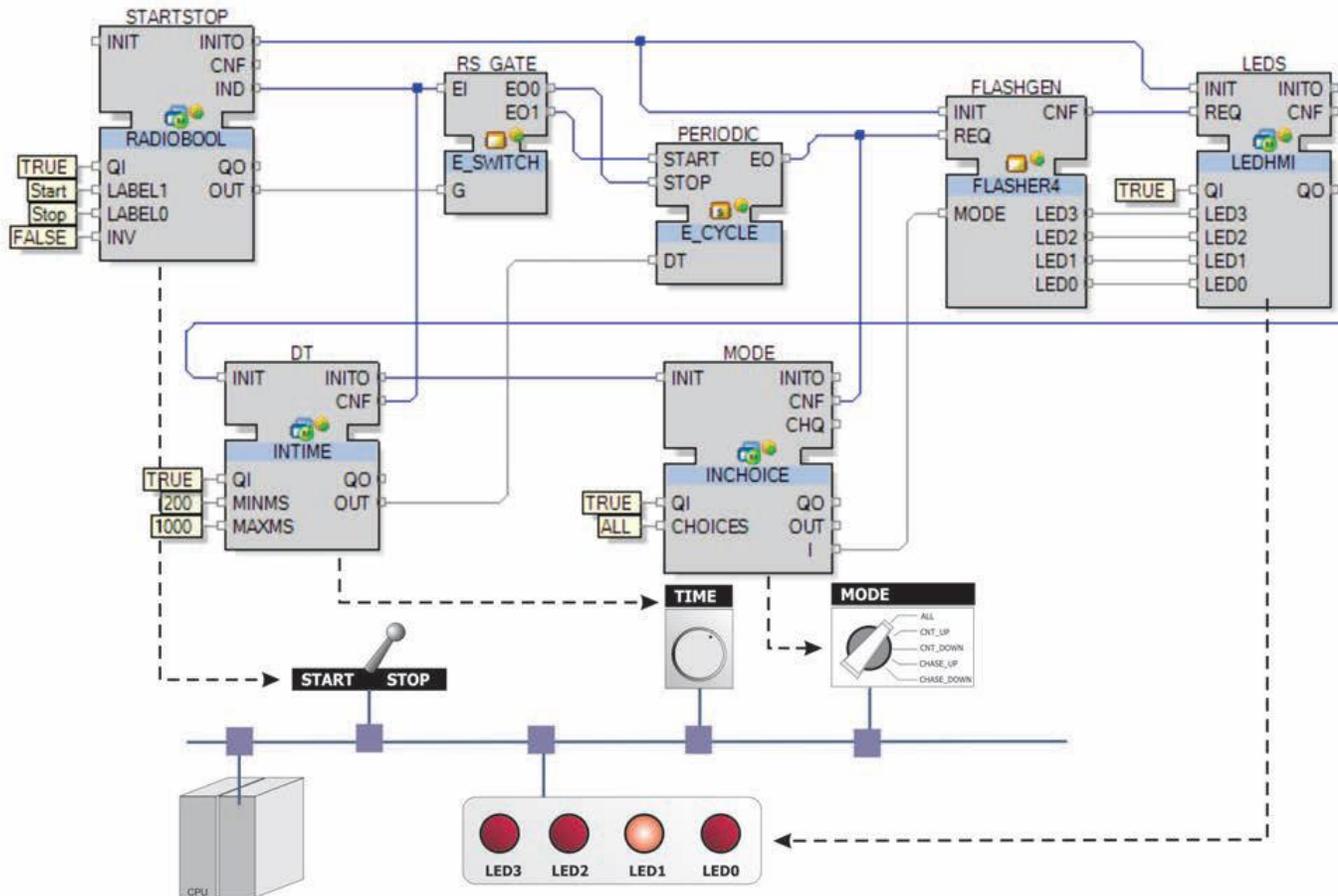


IEC 61499 executable model: Runs on PLC or PC (Soft PLC)

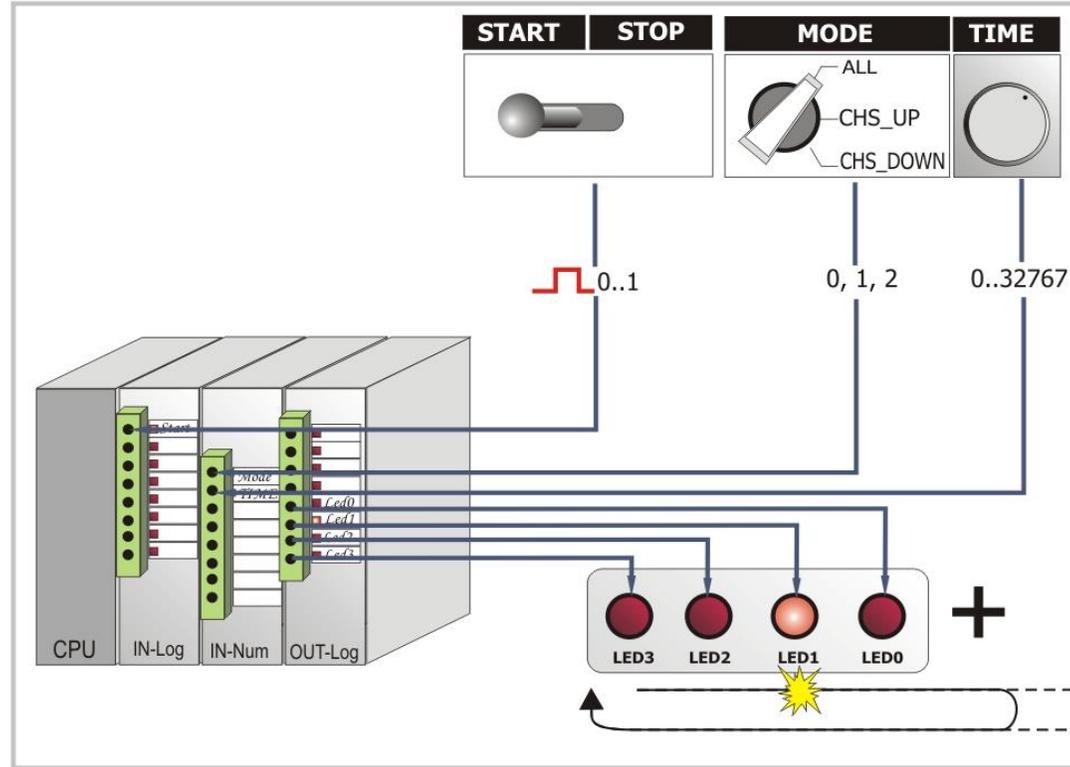


Distributed FLASHER

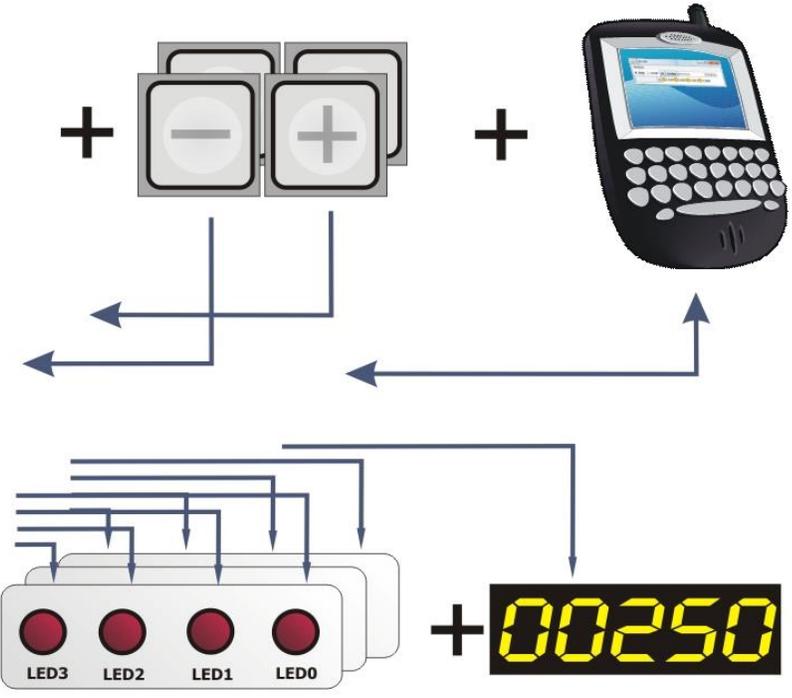
- It is beneficial to model the functionalities in a hardware independent way.
- Same block diagram can be executed on different types of controllers or even on a distributed hardware architecture (running on a network of PLCs).



Motivation: Extensibility

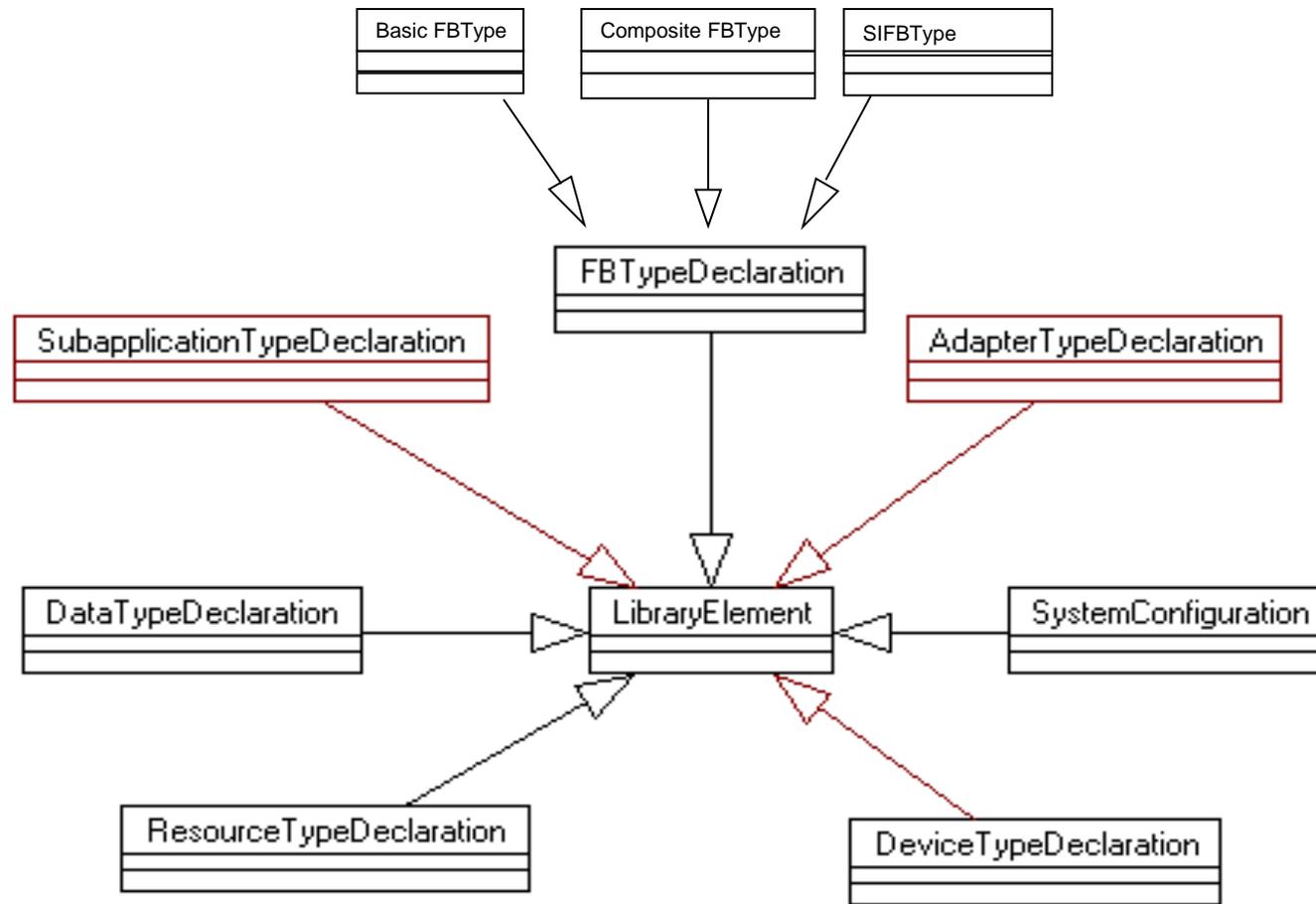


Old system



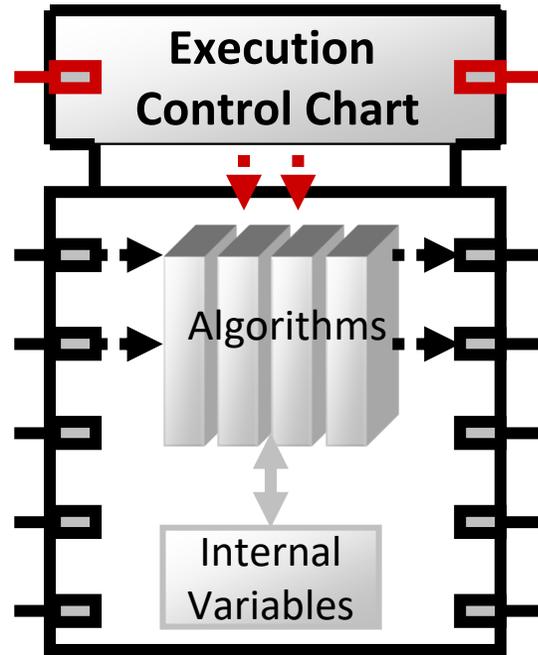
Extensions

Function blocks - Part 1: Architecture

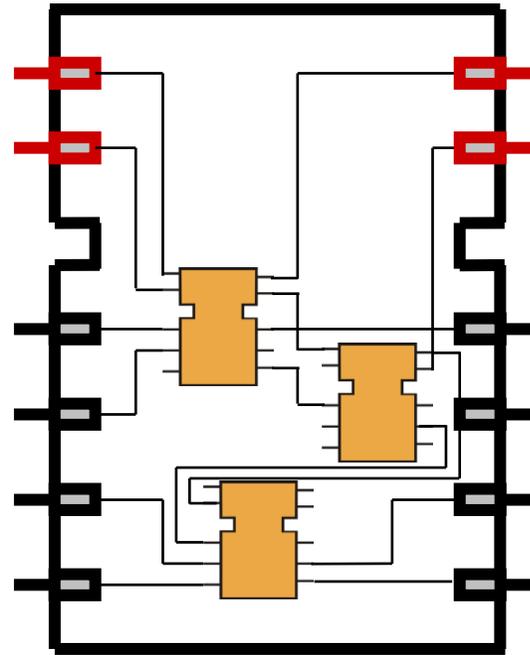


3 kinds of Function Blocks in IEC 61499

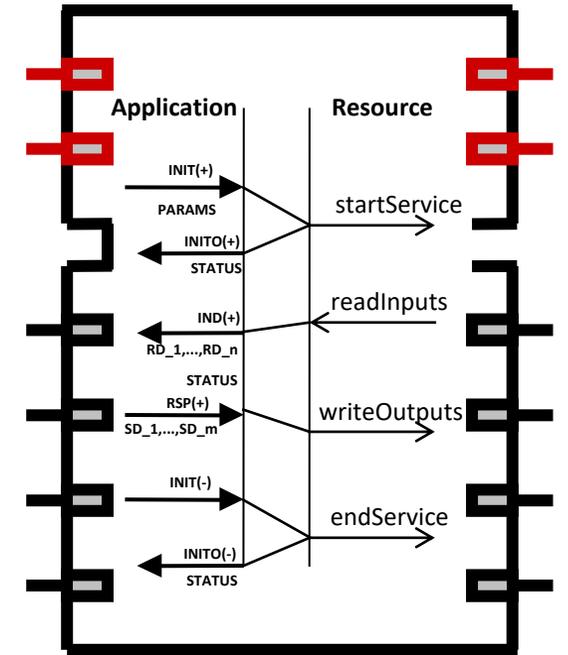
Basic, Composite, and Service Interface Function Blocks



Basic FB

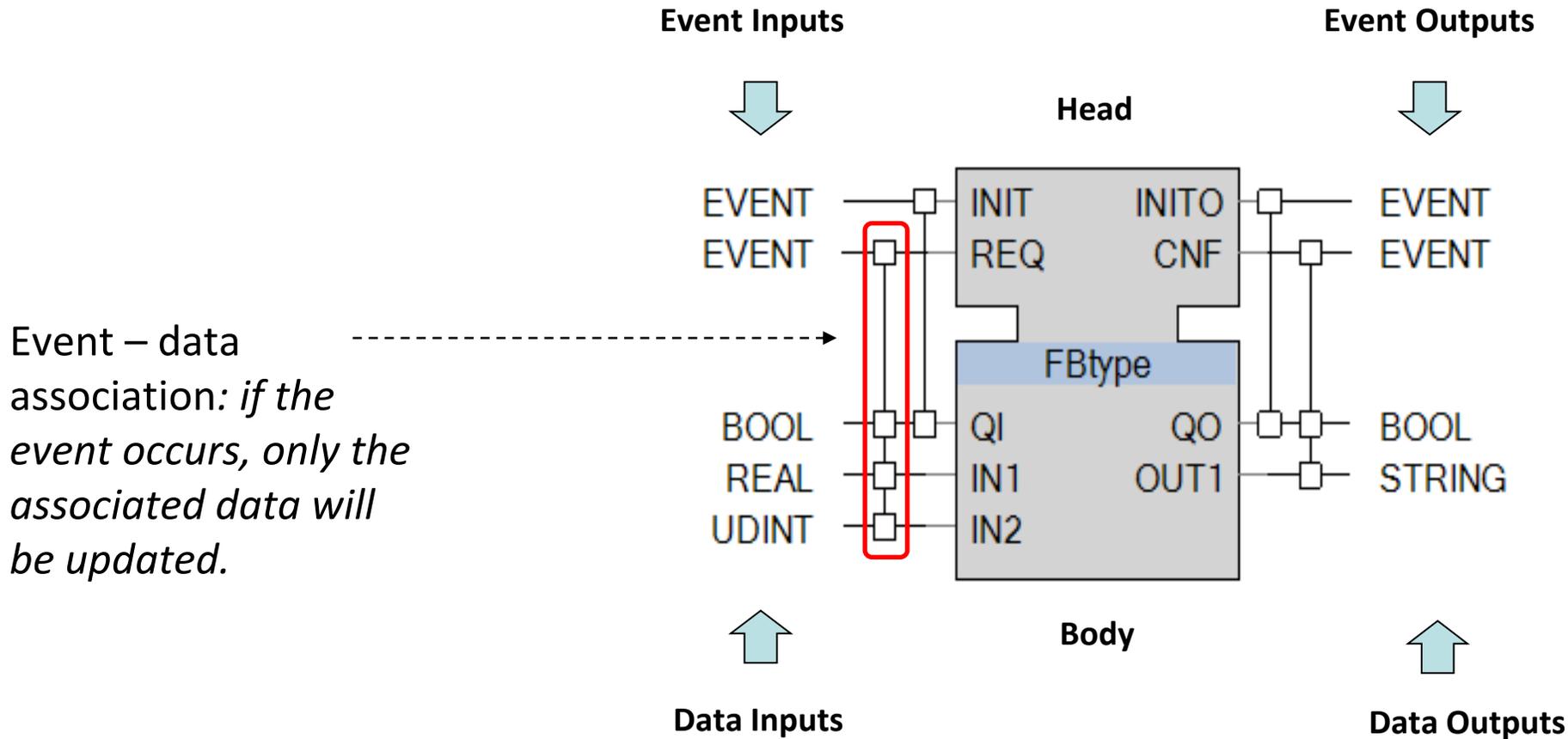


Composite FB



Service Interface FB

Function Block of IEC 61499: Interface

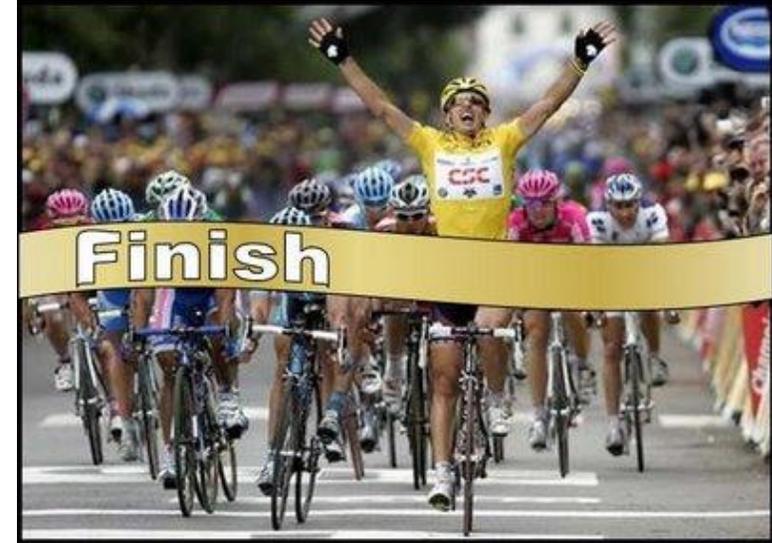


Basic concepts of IEC 61499

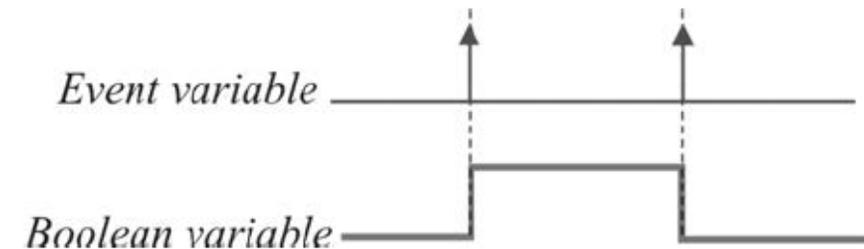
- Event

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

Event of crossing the line



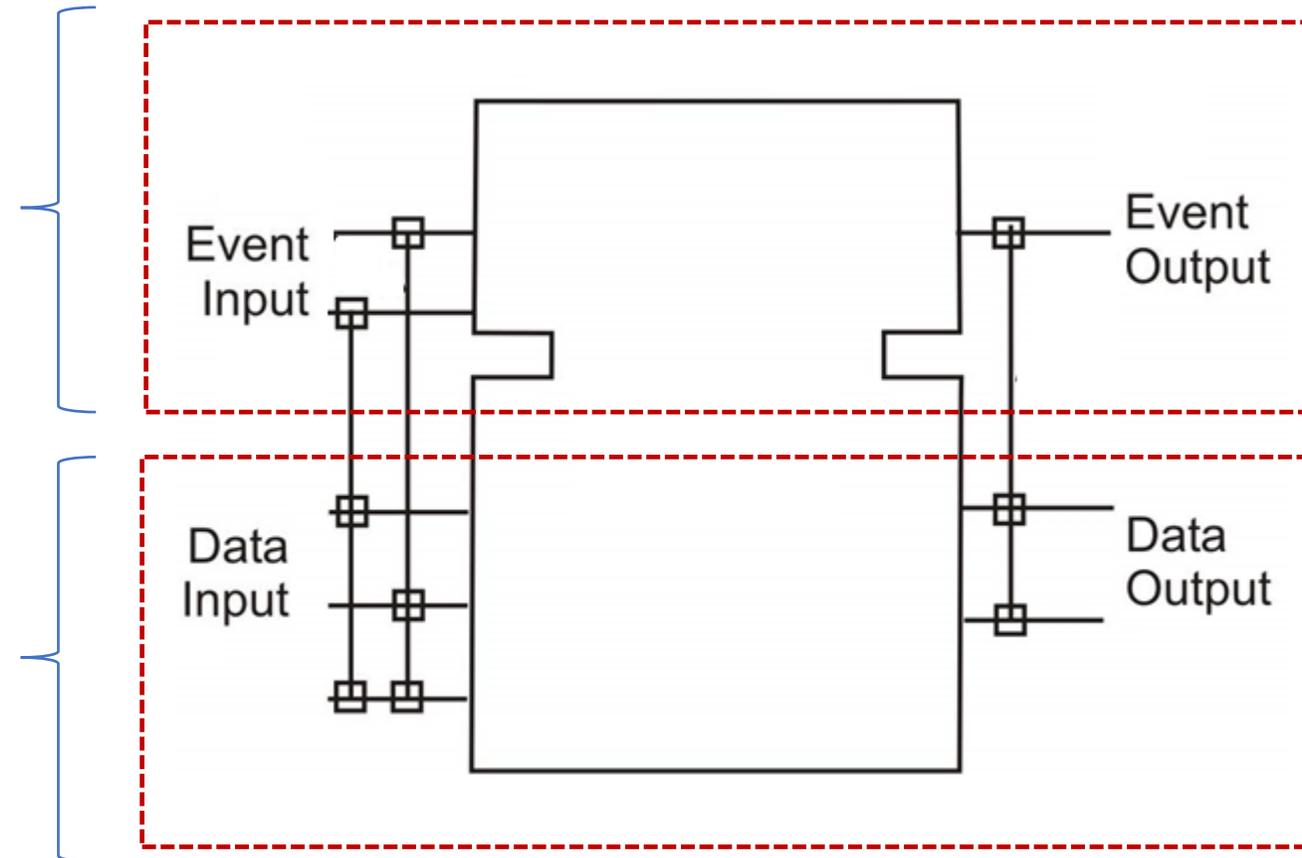
- IEC 61499 function block can only be activated by an event



Function Block: Header and Body

Upper part of FB is called **Header**

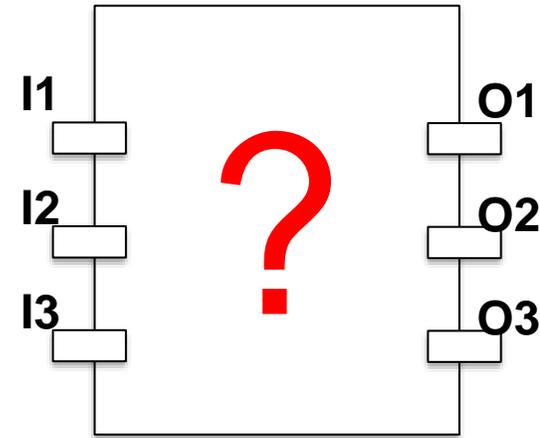
Lower part of FB is called **Body**



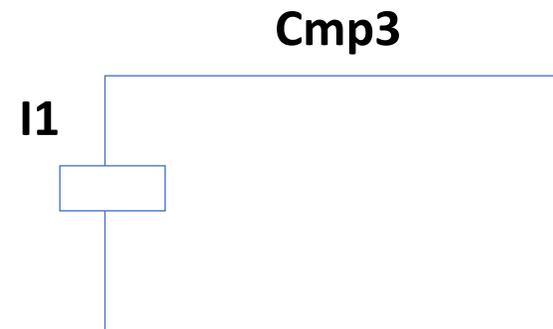
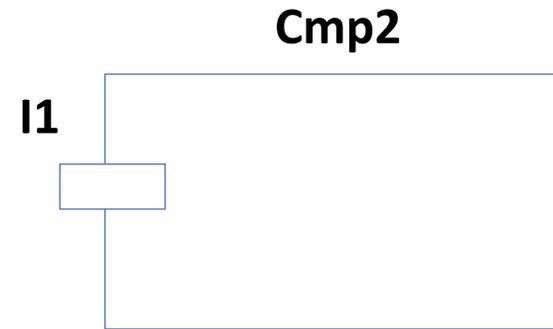
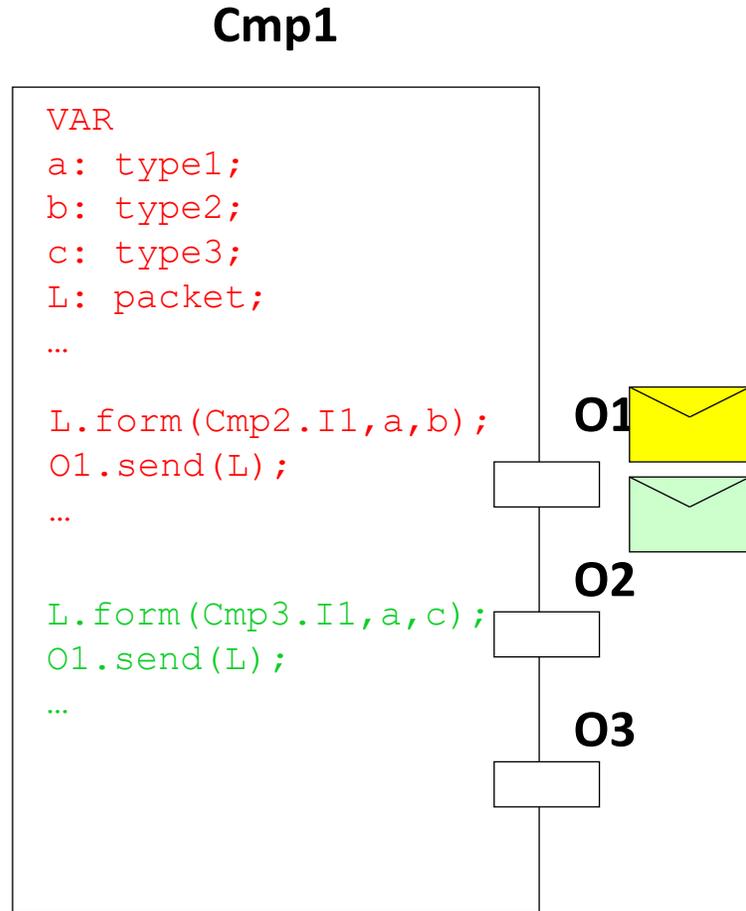
Motivation: Intelligent Automation Component



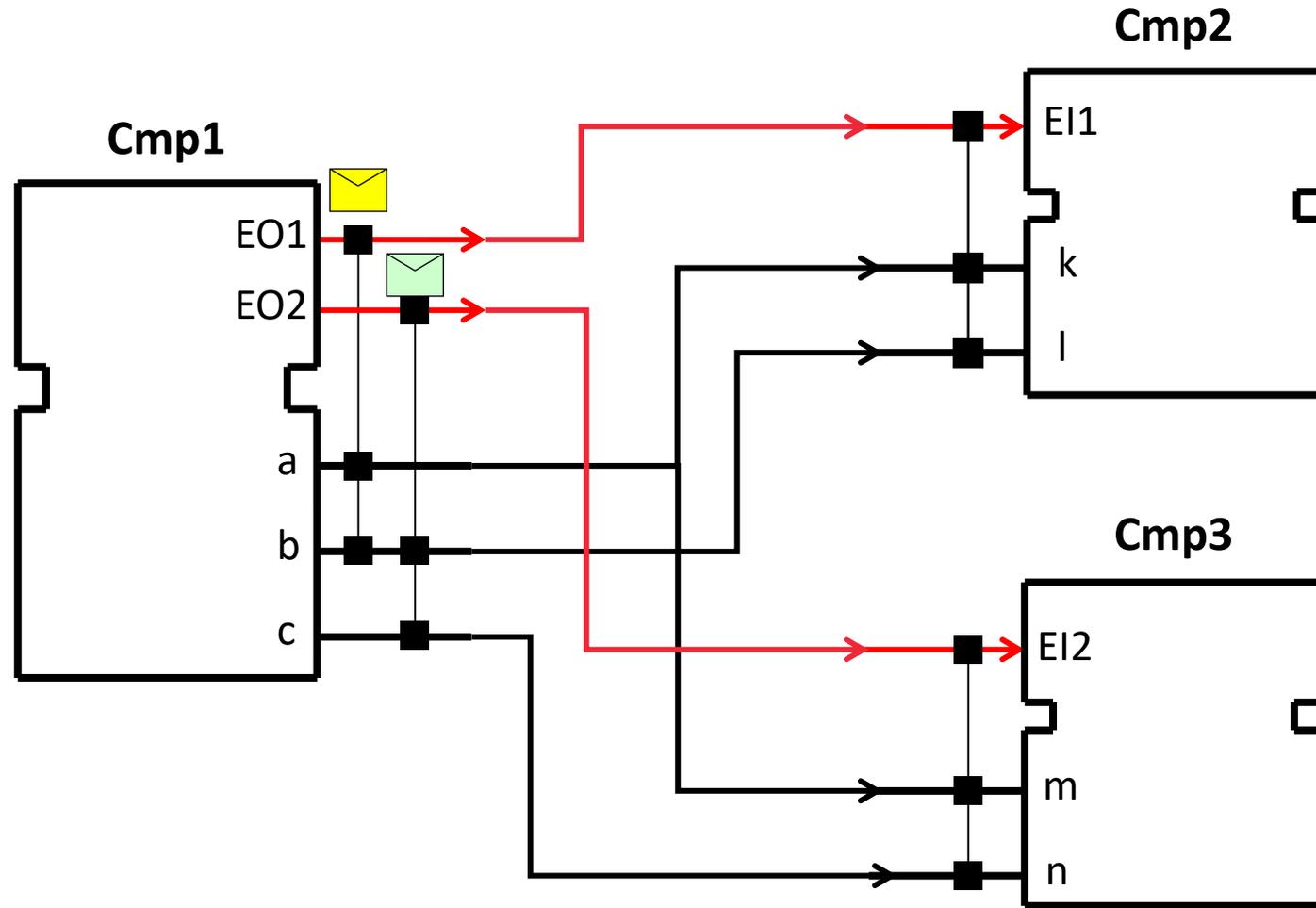
Cylinder Software Component



Communicating components

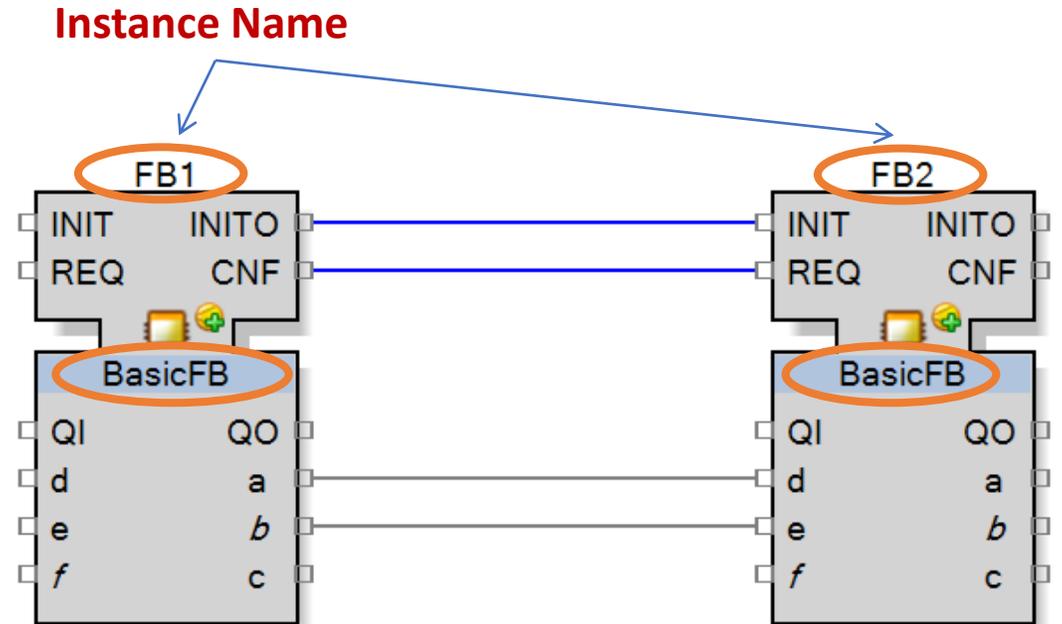
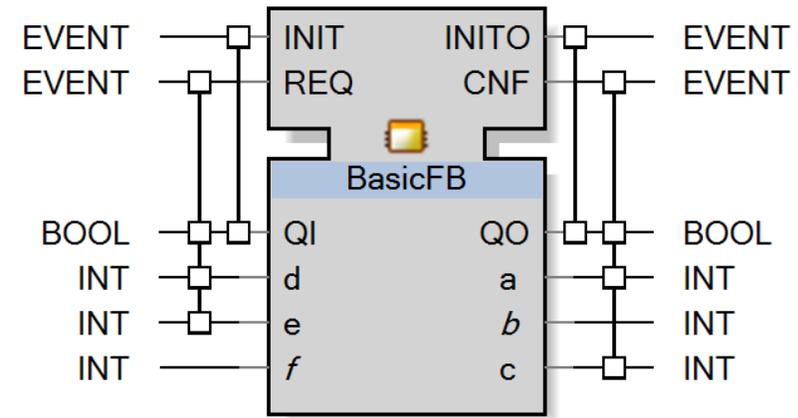


Communicating IEC 61499 function blocks



Types, instances, and connections

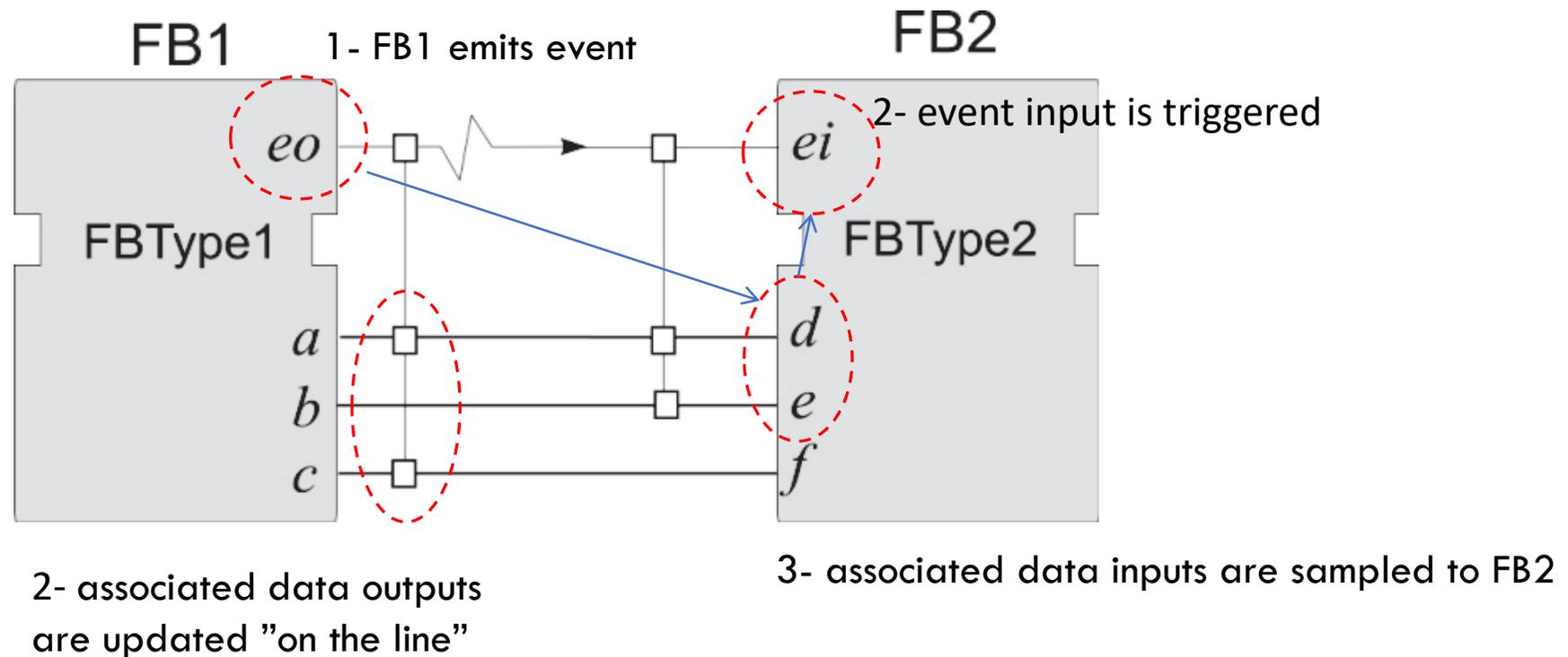
- FB type declaration
 - like objects and classes
 - FB type – class
 - FB instance – object
- Two instances FB1 and FB2
 - Suppose event FB1.CNF is emitted
 - FB1.a is updated and value is sent to FB2.d.
 - FB2.d is sampled
 - FB1.b is not updated (no association). Therefore FB2.e will not receive the updated value.



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”.



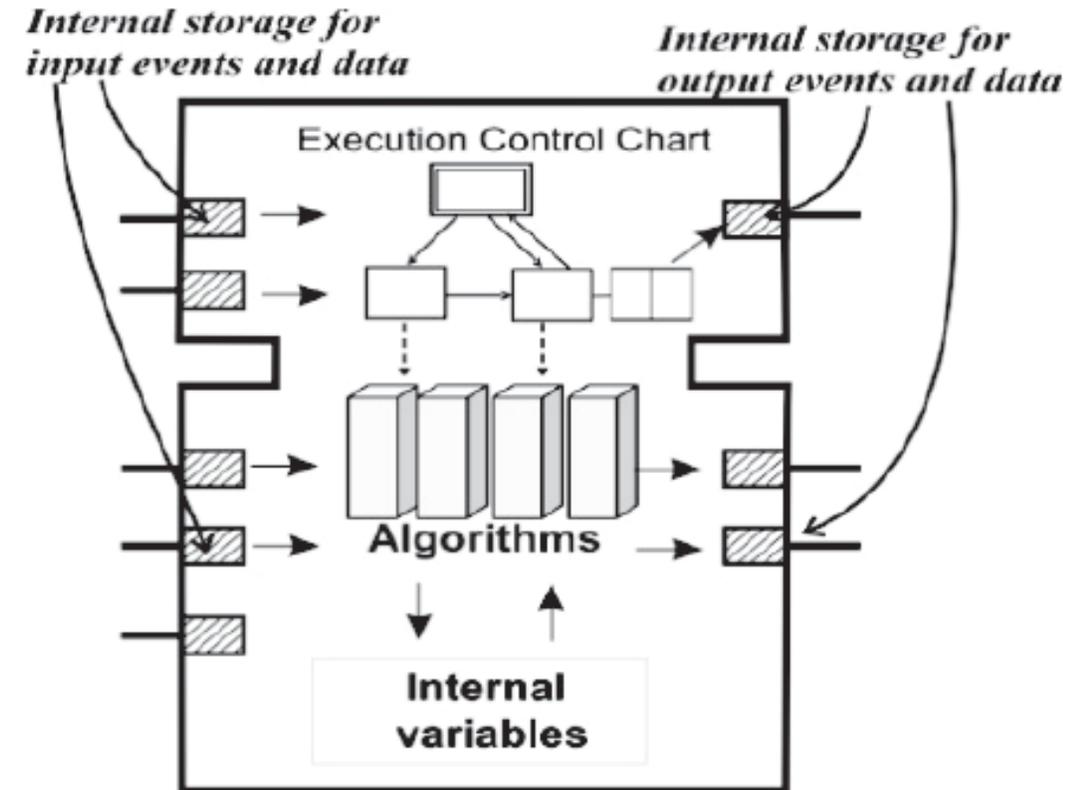
Data Types

- Standard data types (IEC 61131)
- Integer types: Byte, Word, Int...
- Floating types: Real...
- Boolean: BOOL
- String types: String, Wstring
- Date and time
- User defined data types (not supported in nxtStudio)

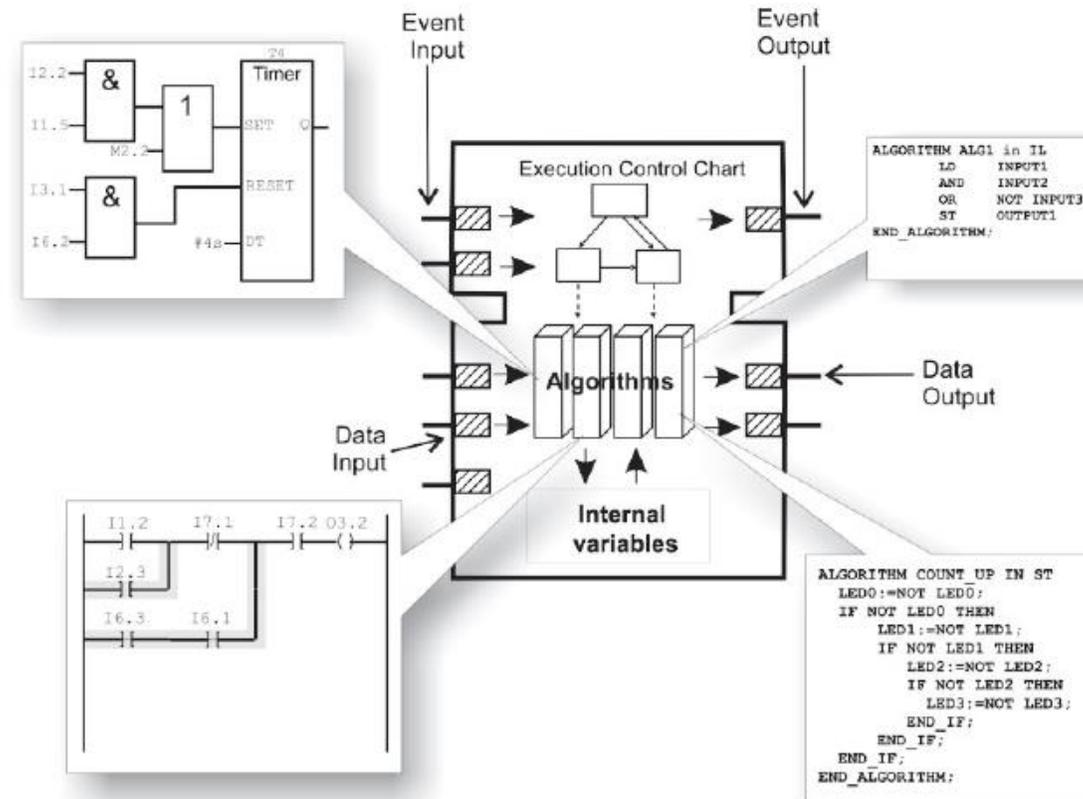
3. Basic FB

Internal structure of a Basic Function Block

1. Internal variables
2. **There are no global variables in IEC 61499!**
3. Algorithms
 - Internal variables of the function block are shared between all algorithms
4. Execution control chart (ECC)
5. Execution Control Function



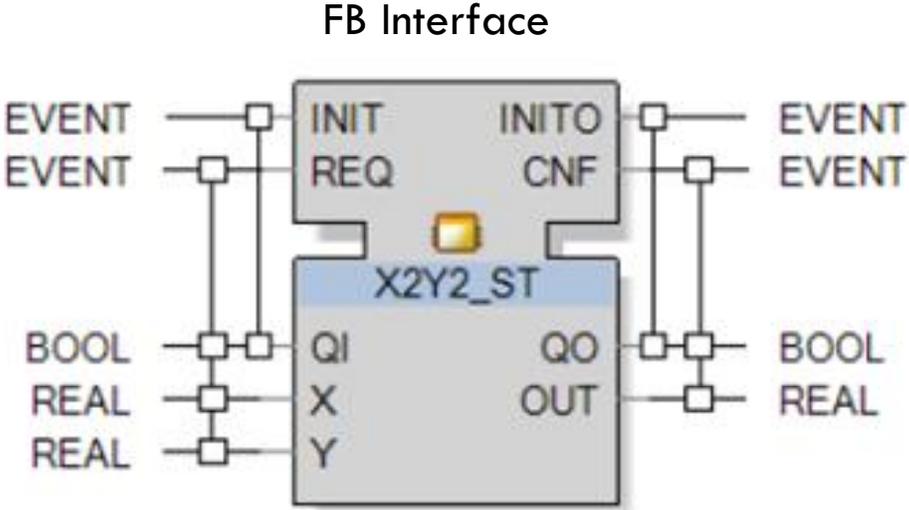
Basic Function Block Algorithms



Basic function block encapsulate algorithms in the languages traditional for industrial automation.

Basic FB – Execution Control Chart (ECC)

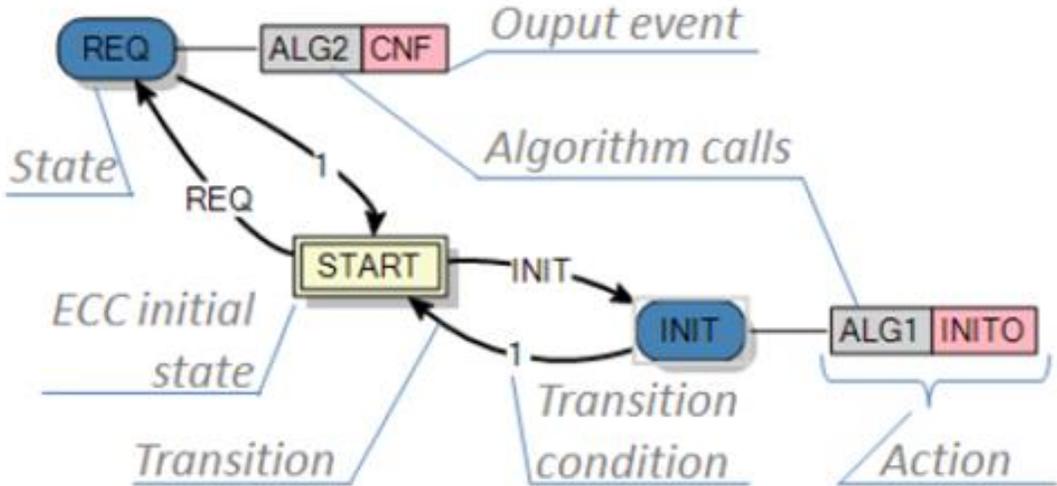
Example: An FB that can calculate $OUT = X^2 - Y^2$



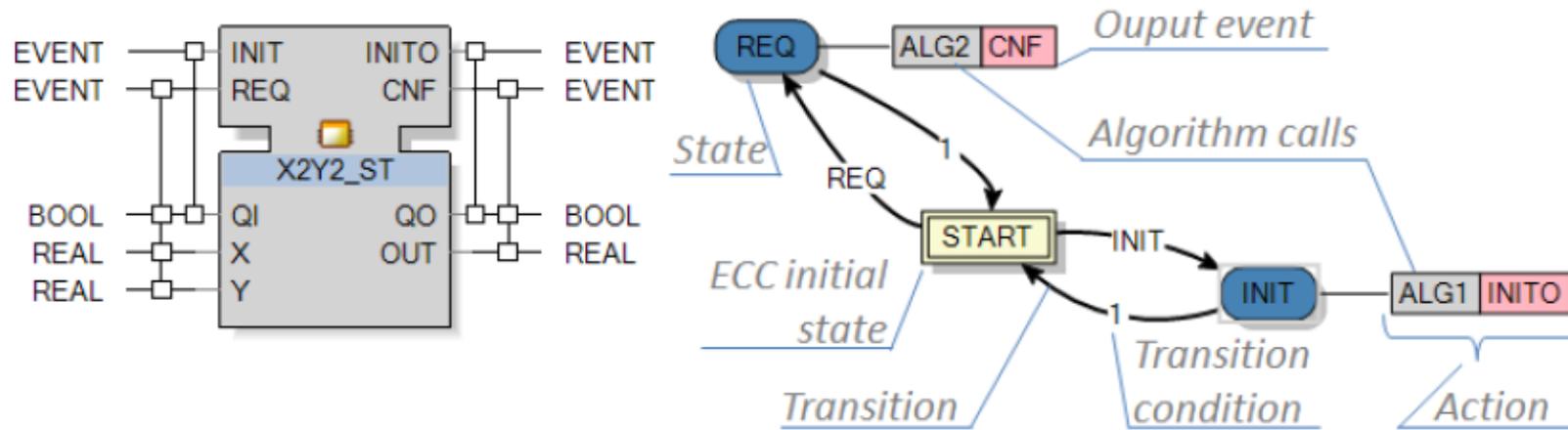
In order to do the calculation, the FB needs values of X,Y and to receive the REQ event



ECC for X2Y2_ST



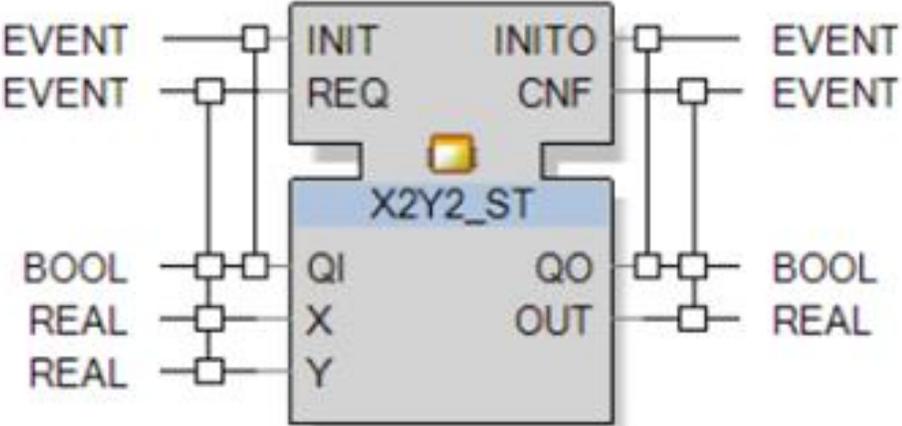
Basic FB – execution control chart (ECC)



How to Interpret the Execution control chart for X2Y2_ST function Block:

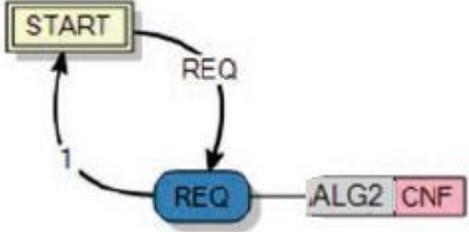
- When FB is in **START** State, if **INIT** event is triggered FB's state will change to **INIT** then
 - 1- Algorithm **ALG1** will be executed
 - 2- **INITO** output event will be triggered
 - 3- State will change back to **START** state
- When FB is in **START** State, if **REQ** event is triggered FB's state will change to **REQ** then
 - 1- Algorithm **ALG2** will be executed
 - 2- **CNF** output event will be triggered
 - 3- State will change back to **START** state

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.

Execution Control Chart



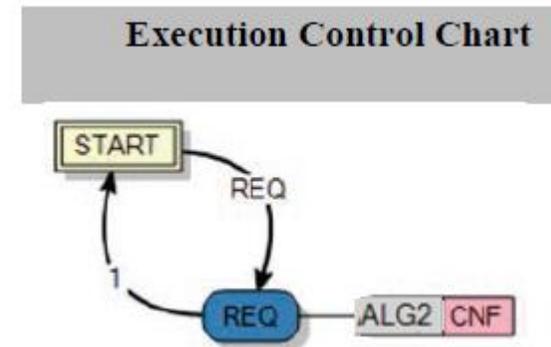
Algorithms

```
ALGORITHM (ALG2) IN ST:  
  OUT := (X-Y)*(X+Y);  
END_ALGORITHM
```

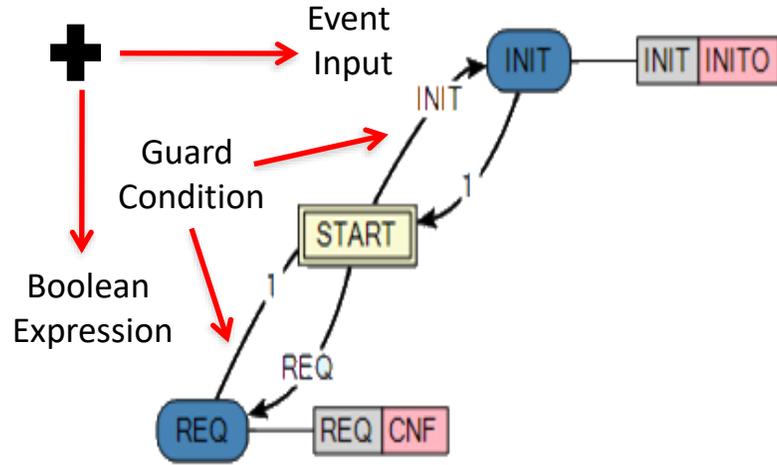
Basic FB – execution control function

The Execution Control function specifies:

- the algorithm that must be invoked after a certain input event
- a certain state of the execution control function. It is specified by means of the Execution Control Chart (ECC).



Transition conditions syntax



A transition condition can be one of the following:

1

Event input

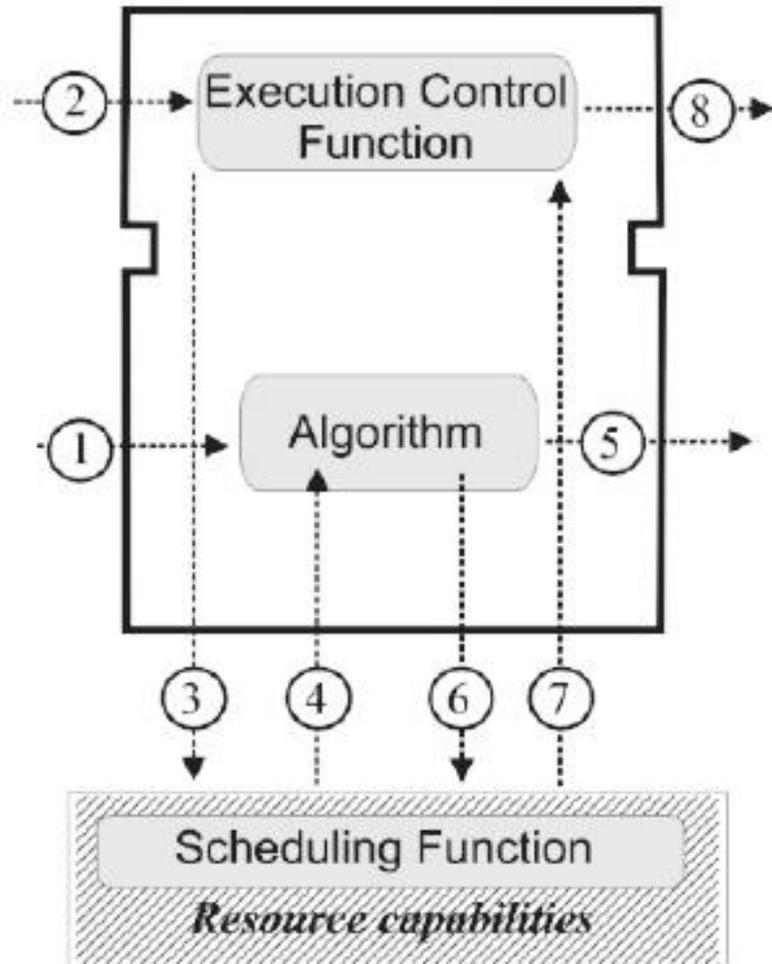
Boolean expression over data

Event input & [Boolean expression over data]

Examples:

1. REQ
2. $((\text{Input_Var}=1) \text{ OR } (\text{Internal_Var}=0)) \text{ AND } (\text{NOT } \text{QO})$
3. REQ AND $(\text{Input_Var}=0)$

Execution control function of Basic Function Block



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.

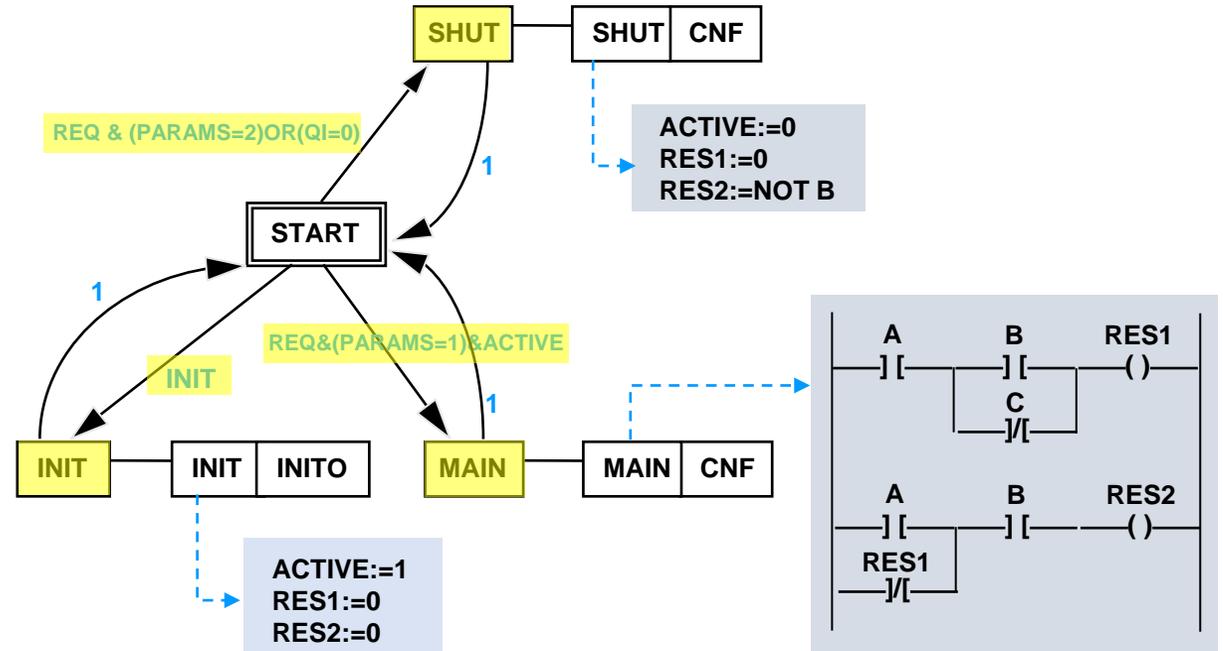
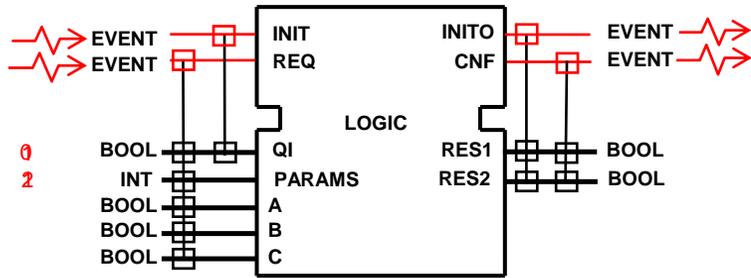
Step 5. The algorithm completes the assignment of values for the output variables associated with the event output.

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

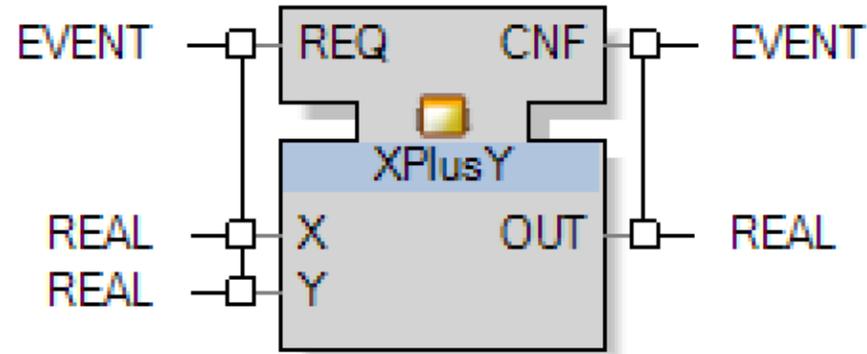
Step 7. The scheduling function invokes the execution control function.

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

Working of a Basic FB



Example: A Basic FB that adds two real numbers



A Basic FB that adds two real numbers

The screenshot displays the nxtSTUDIO software interface. The main window shows the configuration for a function block named 'XPlusY'. The interface includes a menu bar, a toolbar, a Solution Overview tree on the left, a Properties panel on the right, and a central editor area.

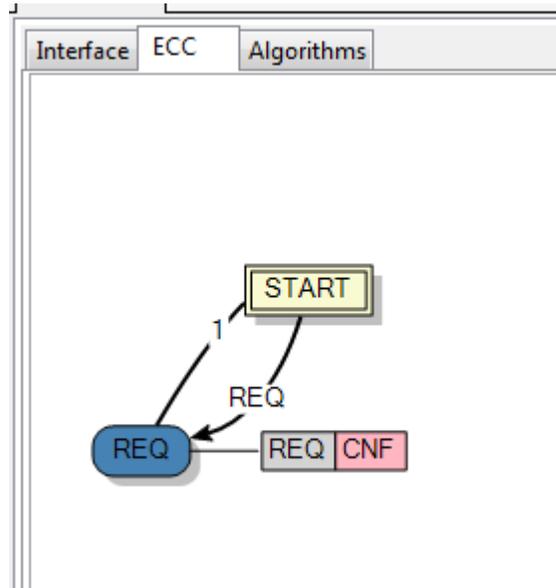
The Properties panel shows the following configuration for the 'XPlusY' block:

Name	Type	Arra...	Initial v...	With	Attr	Cor
EventInputs				X, Y	<none>	No
EventOutputs				OUT	<none>	Exc
InputVars						
X	REAL				<none>	Inp
Y	REAL				<none>	
OutputVars						
OUT	REAL				<none>	Ou
InternalVars						

The central editor area shows a block diagram of the 'XPlusY' function block. It has two input ports labeled 'X' and 'Y', both of type 'REAL'. It has one output port labeled 'OUT', also of type 'REAL'. There are also event ports: 'REQ' (EventInputs) and 'CNF' (EventOutputs), both of type 'EVENT'. A red warning message is overlaid on the diagram: "You are running a demo version. Some functionality will be disabled".

The bottom of the interface shows the 'Output' window, which is currently empty. The status bar at the bottom indicates the current position: 'In 1 col 1 ch 1 INS ...'.

A Basic FB that Adds 2 Real Numbers

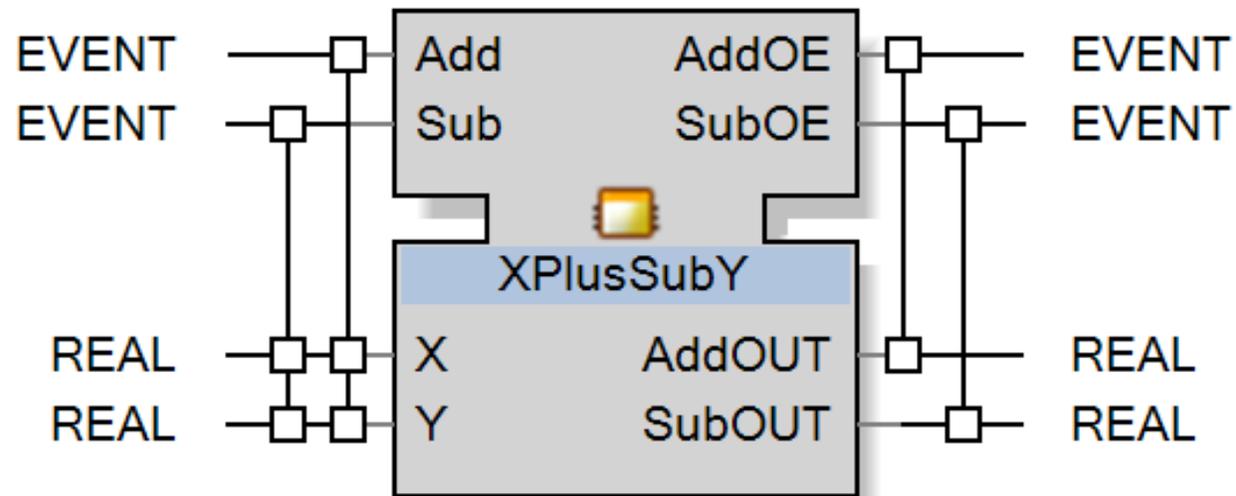


```
XPlusY*
Interface ECC Algorithms
REQ
1  ALGORITHM REQ IN ST:
2  (* Add your comment (as per IEC 61131-3) here
3  Normally executed algorithm
4  *)
5  ;
6  OUT := X + Y;
7  END_ALGORITHM
```

Running the basic FB that Adds 2 Real Numbers

The screenshot displays the nxtSTUDIO software interface. The main window shows a function block diagram (FBD) for the 'XPlusY' function block. The diagram includes a 'START' block connected to a 'REQ' block, which is connected to the 'REQ' input of the 'XPlusY' block. The 'XPlusY' block has two inputs, 'X' and 'Y', and one output, 'OUT'. The 'X' input is connected to a 'REAL' value of 5, and the 'Y' input is connected to a 'REAL' value of 4. The 'OUT' output is connected to a 'REAL' value of 9. A warning message at the top of the diagram area reads: "You are running a demo version. Some functionality will be disabled." The interface also shows a menu bar (File, Edit, View, Build, Device, Debug, Search, Tools, Window, Help), a toolbar, and a Solution Overview pane on the left. The bottom status bar indicates "NXT CONTROL Build finished" and shows coordinates "ln 1 col 1 ch 1 INS".

Example : A Basic FB that Adds and Subtracts two real numbers



A Basic FB that Adds and Subtract 2 Real Numbers

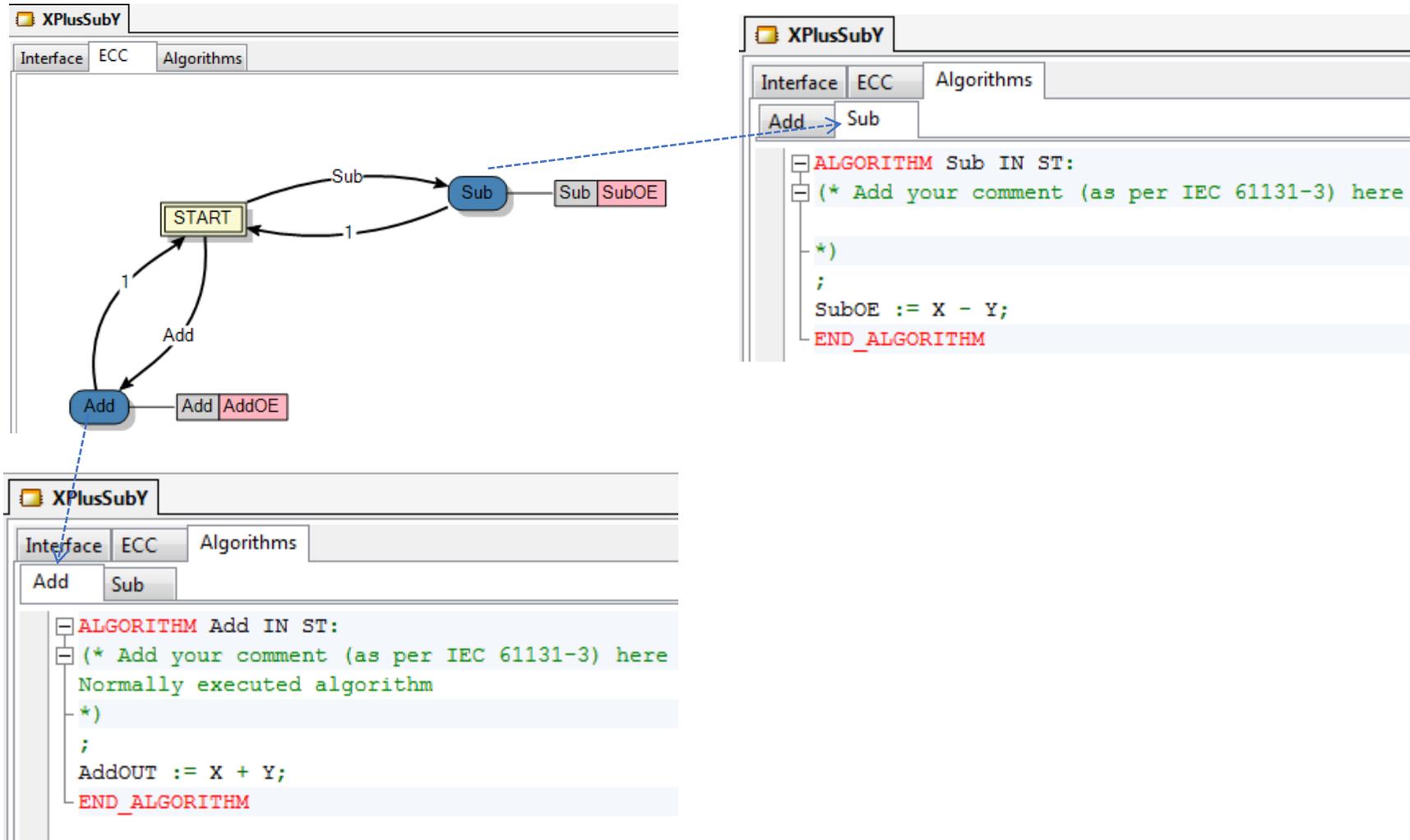
The screenshot displays the NxtStudio IDE interface for a project named 'XPlusSubY'. The main window is divided into several panes:

- Left Pane (Solution Overview):** Shows a tree view of the project structure, including 'Runtime.Base', 'nxtControl.Standard', 'Tutorial', 'System', 'CAT', 'CAT Instances', 'SubApp', 'Composite', 'Basic', 'XPlusSubY', 'XPlusY', 'Service', 'Adapter', 'Canvases', and 'Graphics'. 'XPlusSubY' is selected.
- Top Pane (Interface):** A table defining the function block's interface. It lists EventInputs (Add, Sub), EventOutputs (AddOE, SubOE), InputVars (X, Y), OutputVars (AddOUT, SubOUT), and InternalVars.
- Right Pane (Diagram):** A block diagram showing the 'XPlusSubY' block with its inputs (EVENT, REAL) and outputs (EVENT, REAL). A red warning message is visible: "You are running a demo version. Some functionality will be disabled".
- Bottom Pane (Output):** A log window showing the build process: "Build started", "Compiling XPlusY.fbt", "Build finished", and state transitions from 'START' to 'REQ' and back to 'START'.

Name	Type	Agr...	Initial v...	With	Attr
EventInputs					
Add				X, Y	<none>
Sub				X, Y	<none>
<new interface>					
EventOutputs					
AddOE				AddOUT	<none>
SubOE				SubOUT	<none>
<new interface>					
InputVars					
X	REAL				<none>
Y	REAL				<none>
<new variable>					
OutputVars					
AddOUT	REAL				<none>
SubOUT	REAL				<none>
<new variable>					
InternalVars					
<new variable>					

```
Build started
Compiling XPlusY.fbt
Build finished
Active state changed to 'START'
Active state changed to 'REQ'
Transition from 'START' to 'REQ' [1] cleared
Active state changed to 'START'
Transition from 'REQ' to 'START' [2] cleared
```

A Basic FB that Adds and Subtract 2 Real Numbers



A Basic FB that Adds and Subtract 2 Real Numbers

The screenshot displays the nxtSTUDIO development environment. The main workspace shows a state machine diagram with three states: **START** (green), **Add** (blue), and **Sub** (blue). Transitions are labeled with 'Add' and 'Sub', and a '1' is shown on the transition from Add to START. Each state is associated with a function block: 'Add' leads to 'Add AddOE' and 'Sub' leads to 'Sub SubOE'. A red warning message at the top of the workspace reads: "You are running a demo version. Some functionality will be disabled."

To the right, a detailed view of the **XPlusSubY** function block is shown. It has two input ports labeled **X** and **Y**, and two output ports labeled **AddOUT** and **SubOUT**. The block is connected to a data table with the following values:

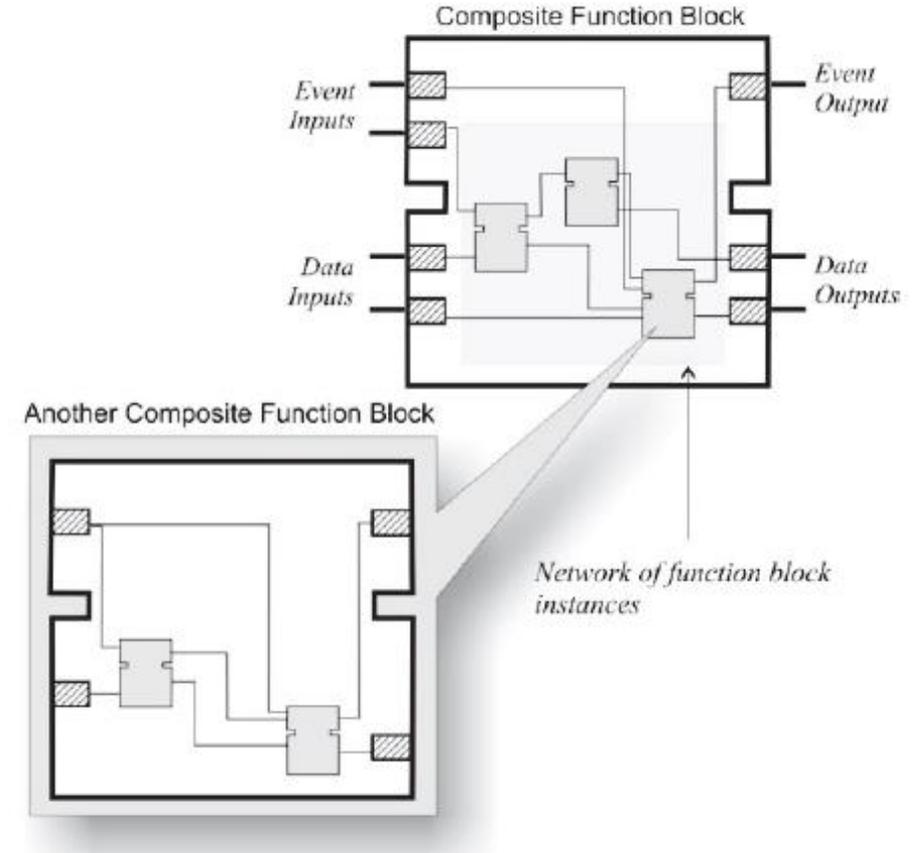
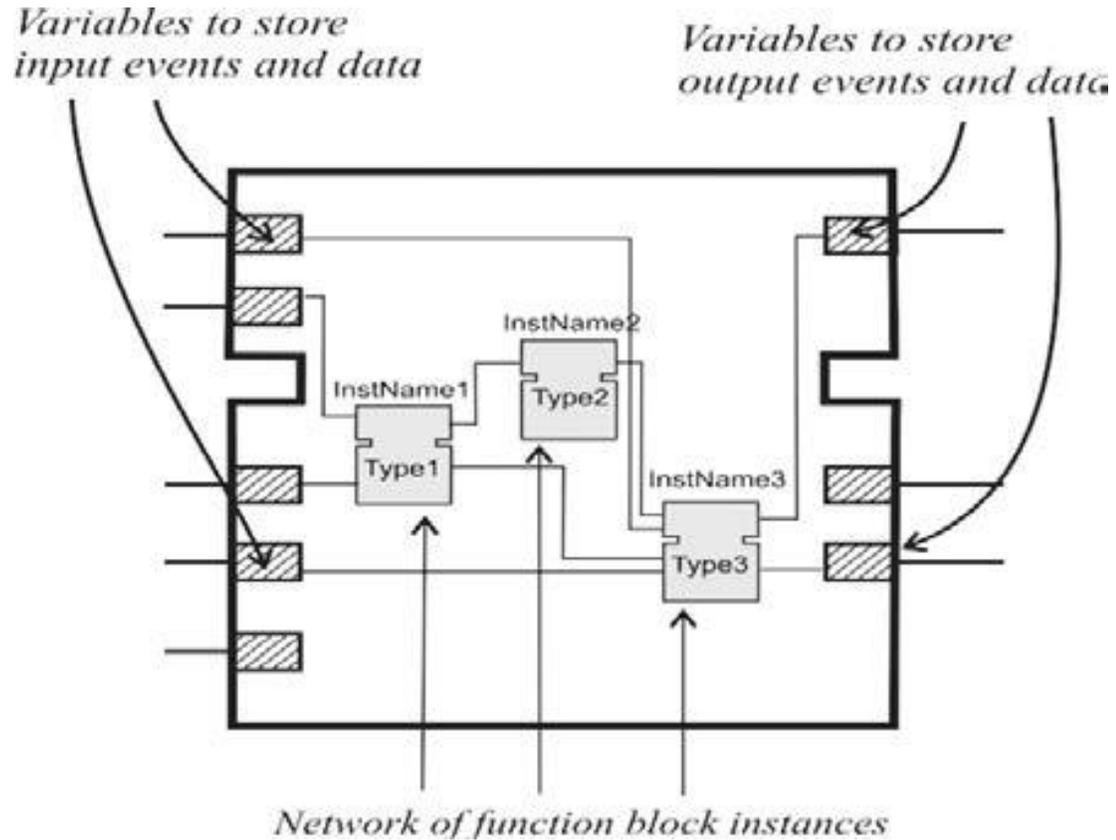
REAL	Value	REAL	Value
X	10	AddOUT	17
Y	7	SubOUT	3

The bottom of the interface shows the **Output** window with the message: "Active state changed to 'START'". The status bar at the bottom indicates "Build finished" and "In 1 col 1 ch 1 INS ...".

4. Composite FB and FB Networks

Composite Function Block

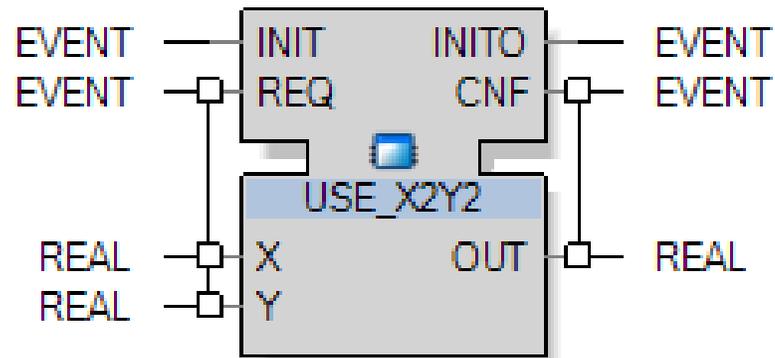
- Network of interconnected FBs
- No internal variables
 - storing the values of input and output events and data
- Hierarchy



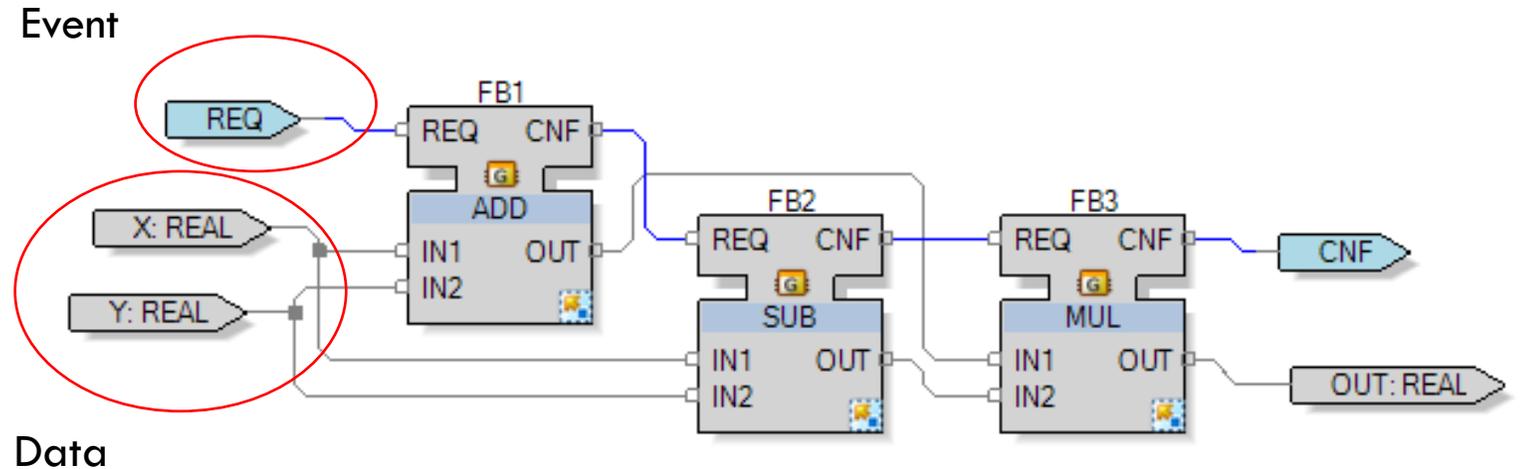
Composite Function Block: Example

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

Interface



Implementation

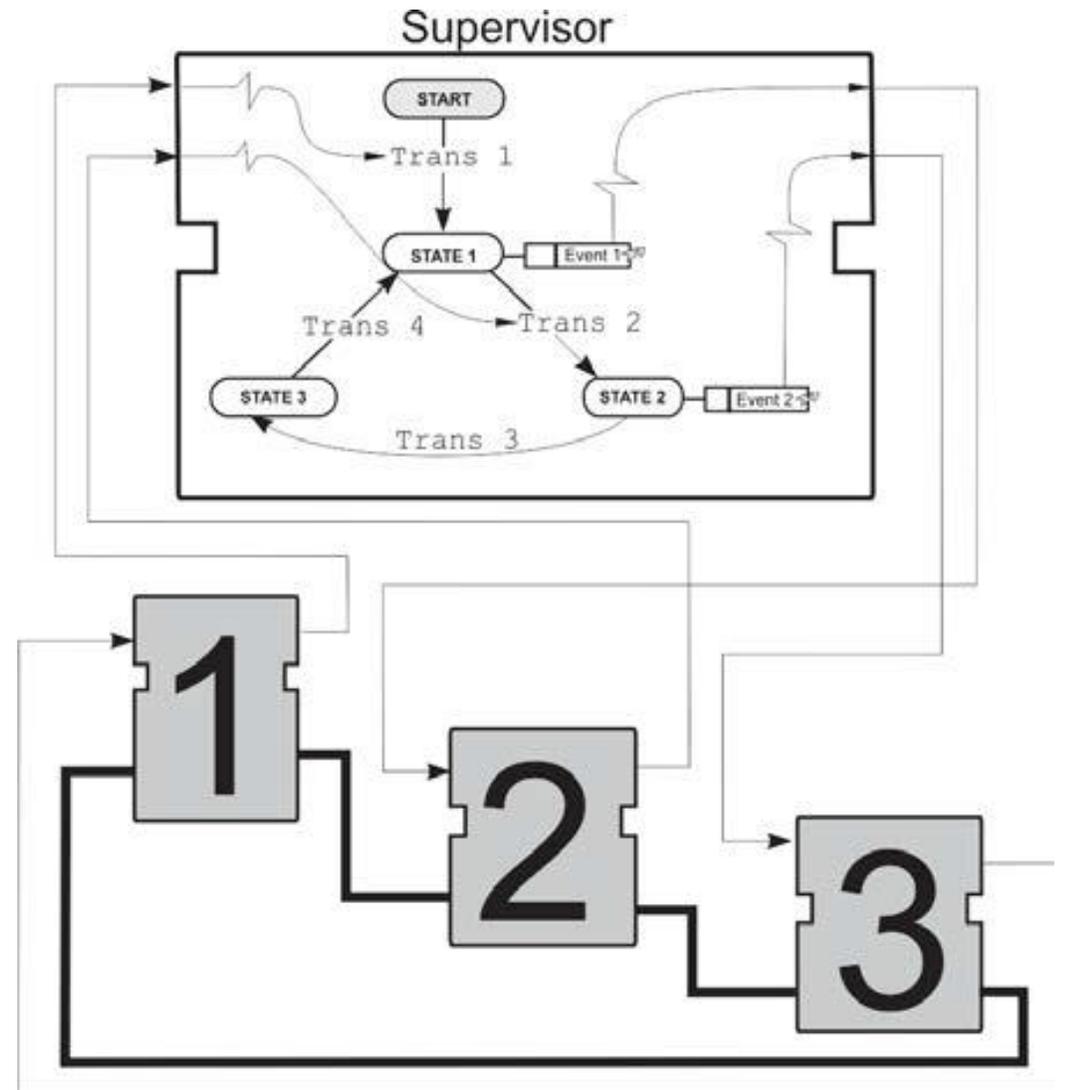


Data

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

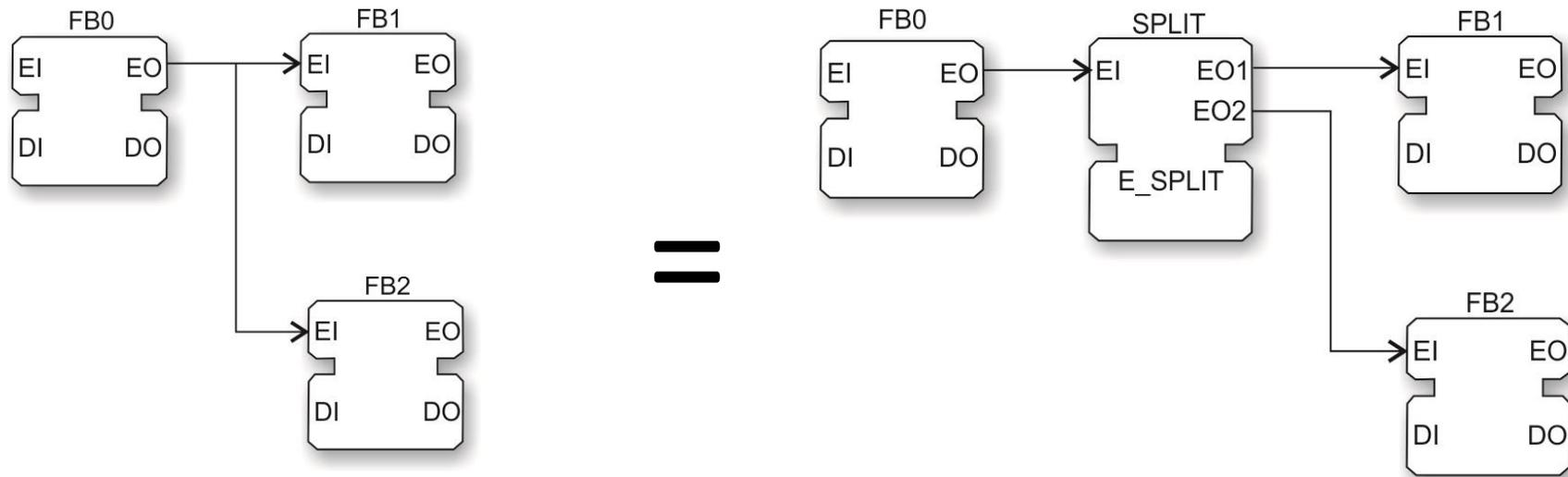
Composite Function Block

- Execution Control:
 - No ECC
 - To ensure a particular order of execution, you can implement a supervisor



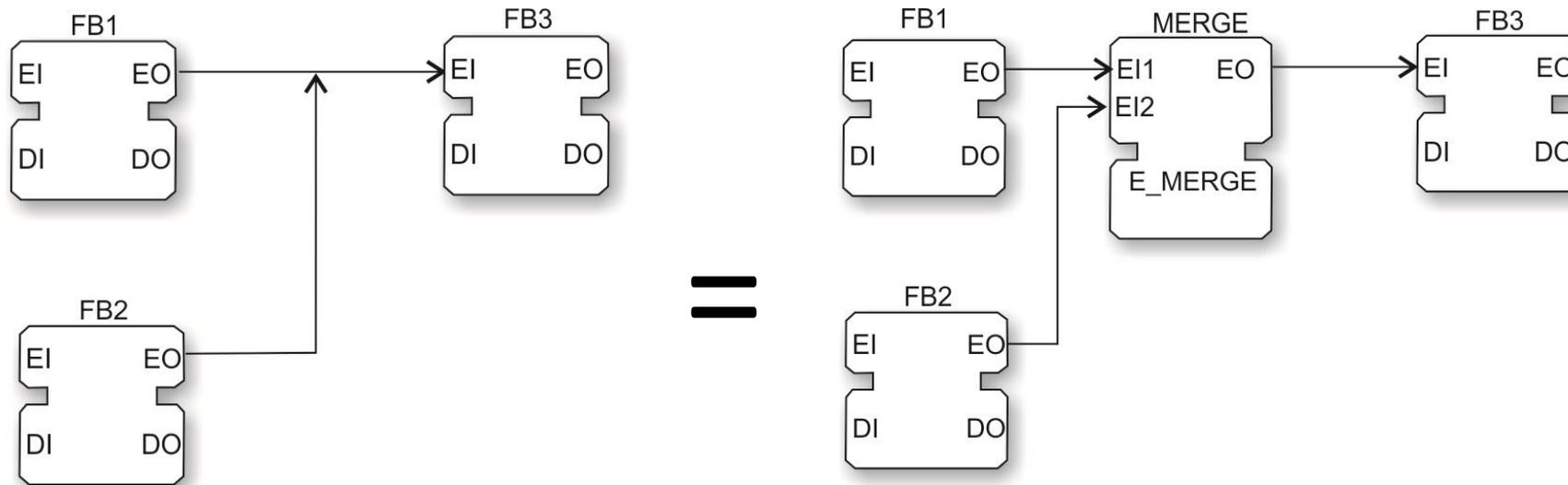
Rules for event connections

Event split is equivalent to using E_SPLIT standard library FB



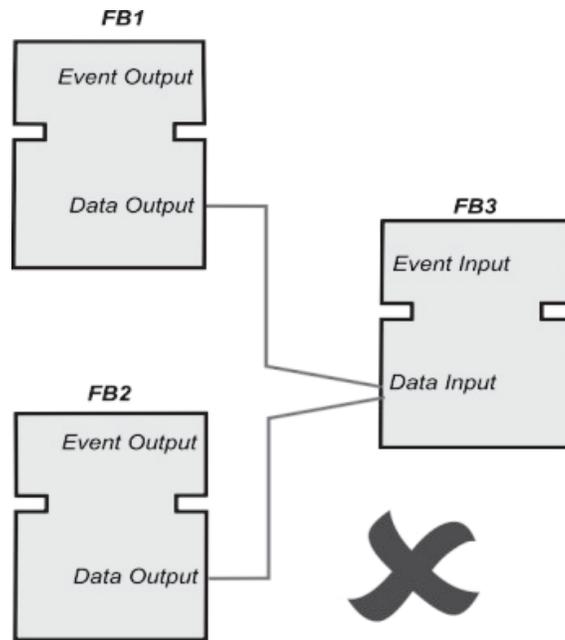
Event merge

Event merge is equivalent to using E_MERGE standard library FB

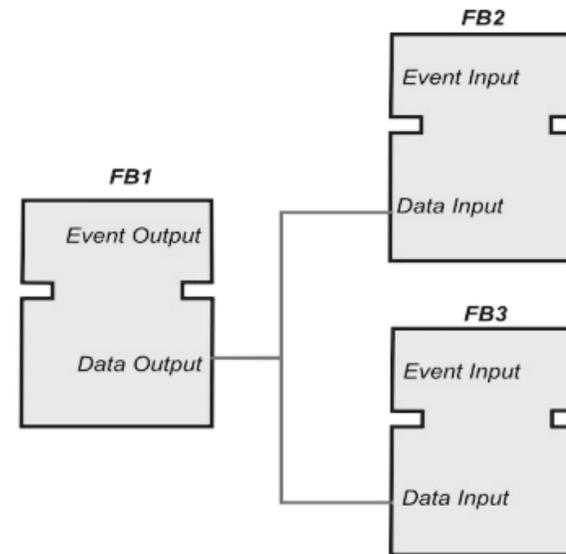


Data connections

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



Incorrect

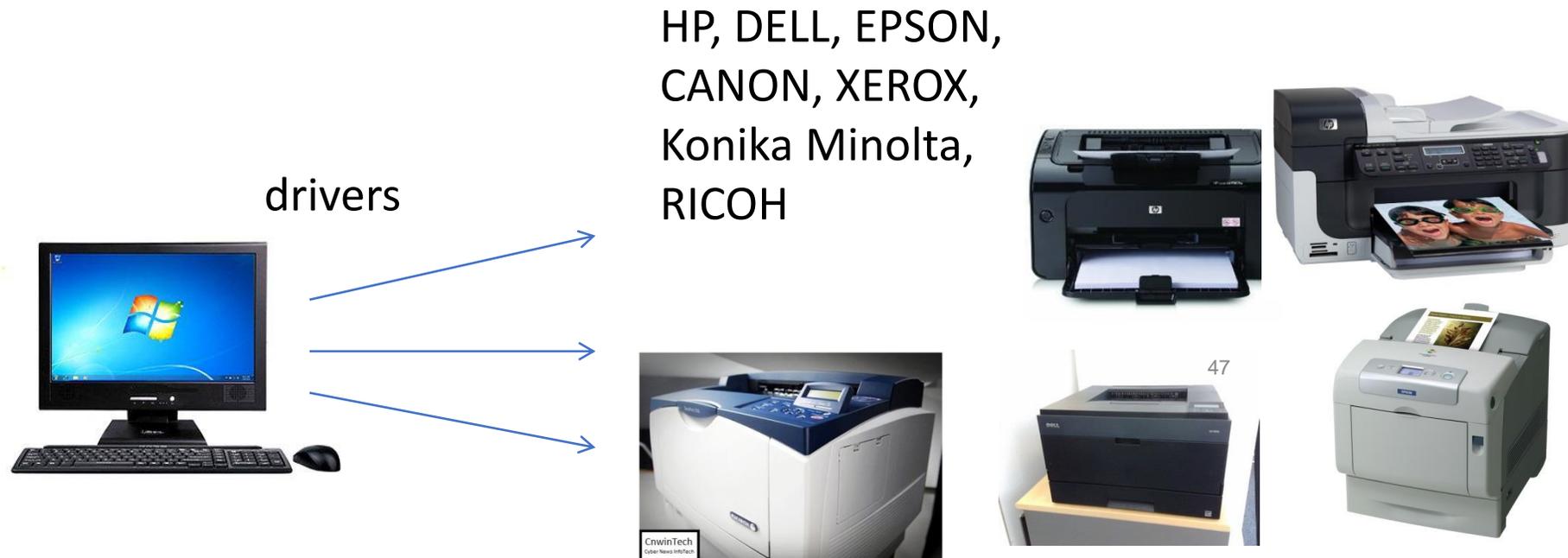


Allowed

5. Service Interface FBs

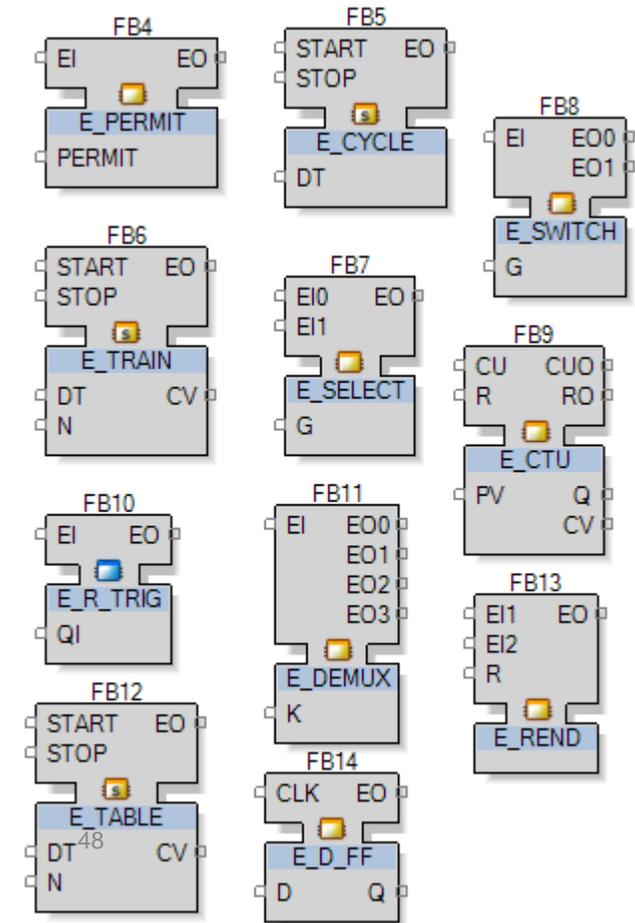
Service interface function blocks (SIFB)

- 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 device vendor
- Used for encapsulation and protection of IP



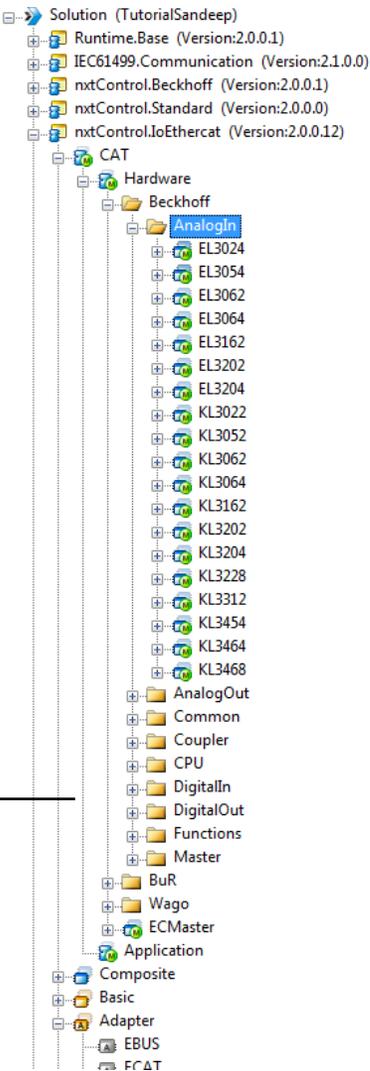
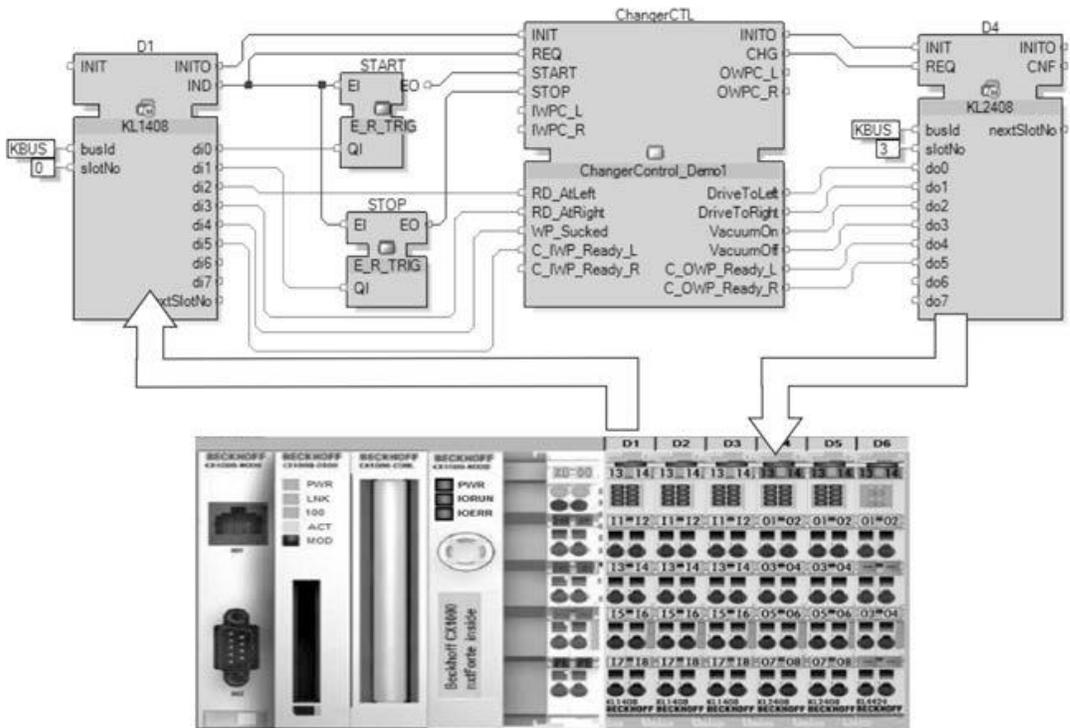
SIFB example: Operations with events

- **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.



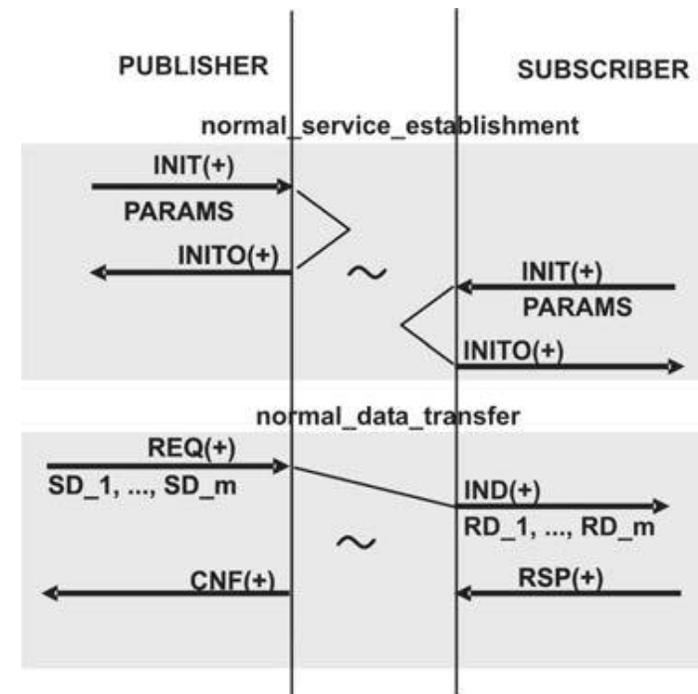
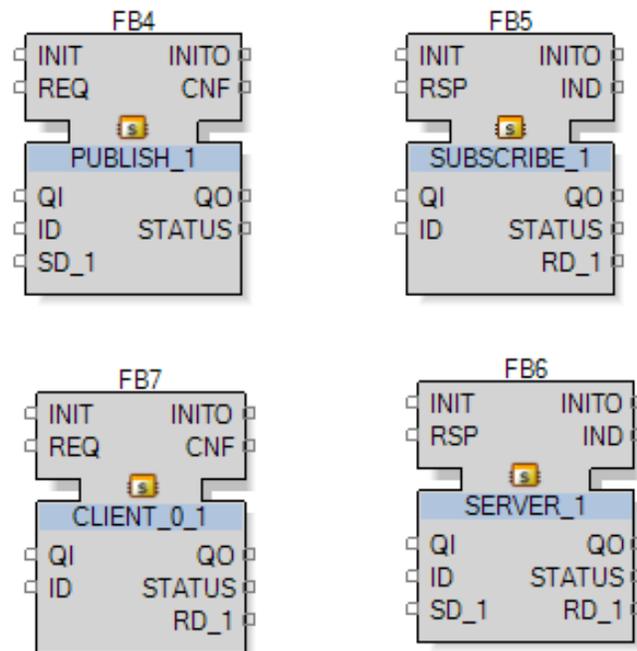
SIFB example: Process interface

Read inputs and write outputs of a PLC



Particular case of SIFB: Communication interface function blocks

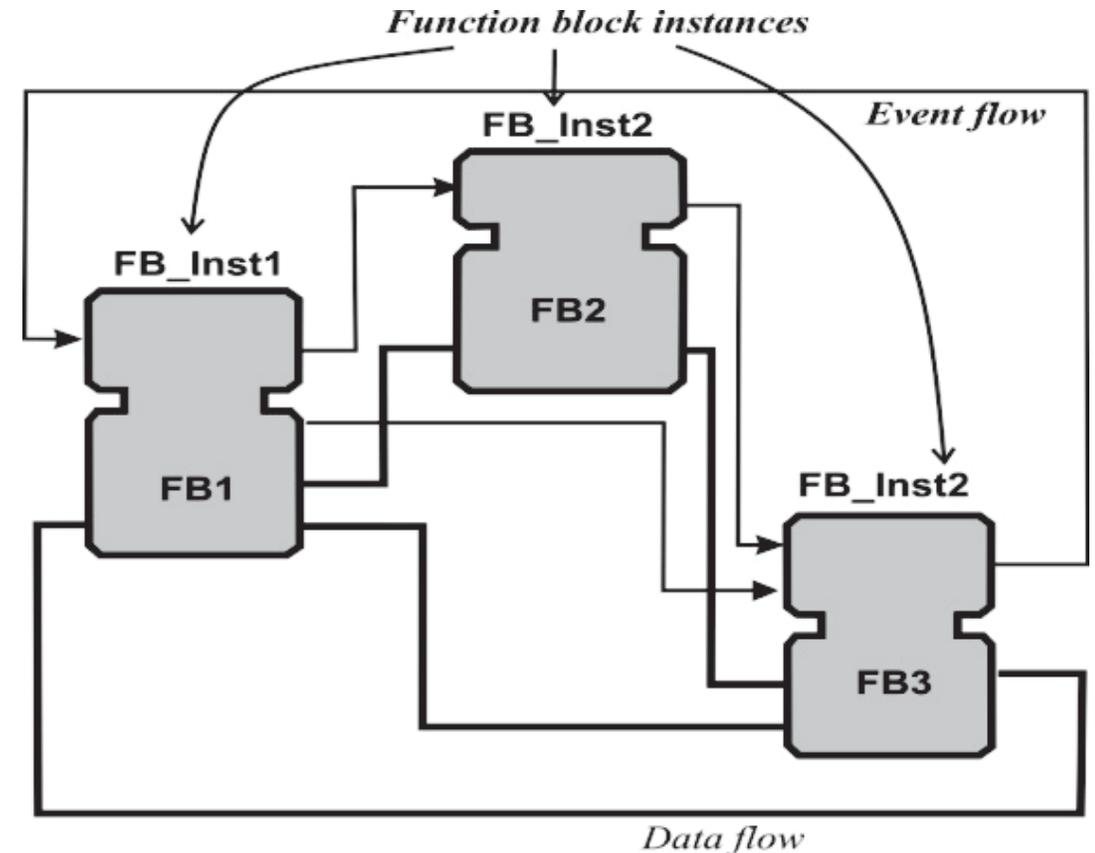
- Publish/subscriber – uni-directional transactions
- Client/server – bi-directional communication



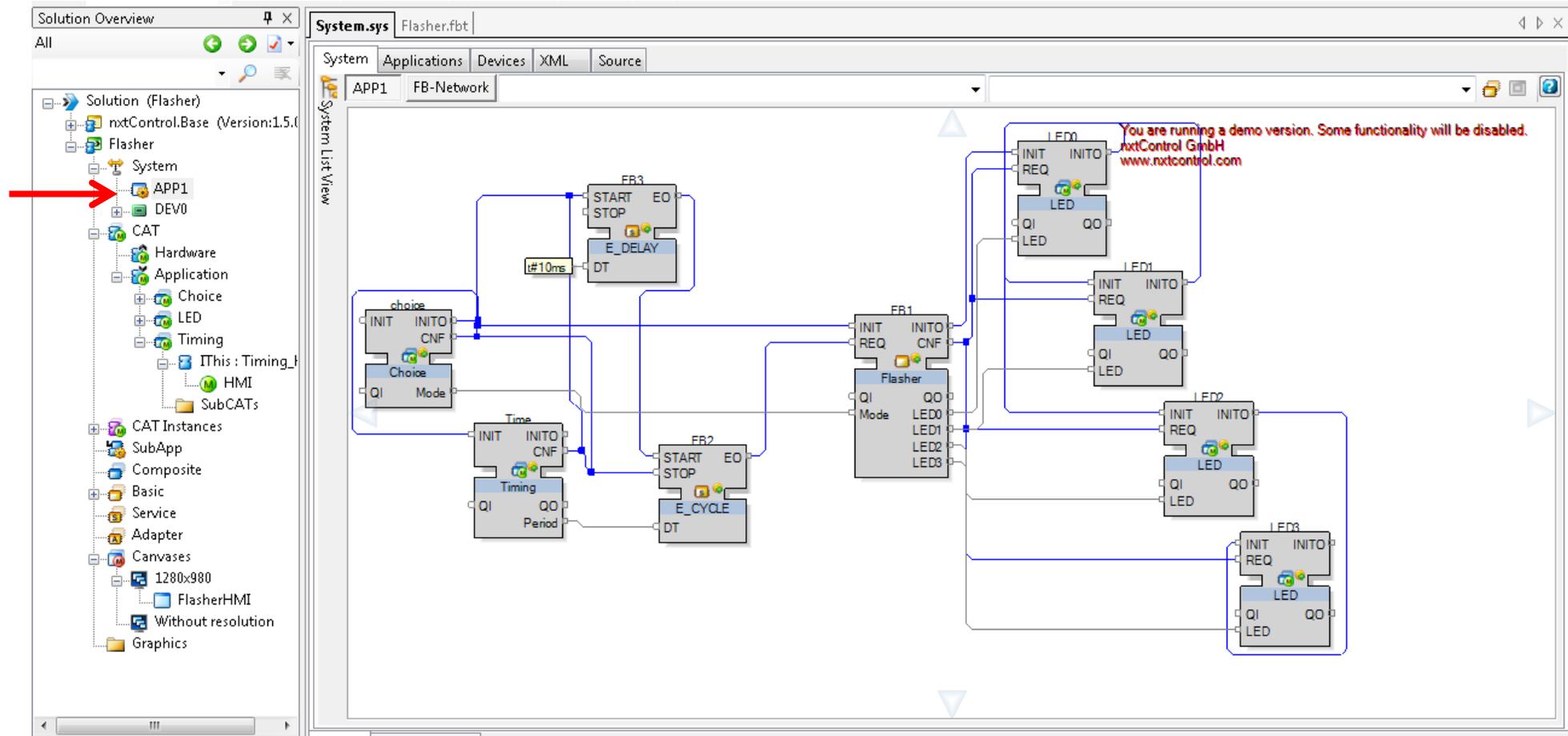
6. Applications

Application and sub-applications

- Application does not have interface
- Network of FB instances
- No internal variables
- Abstract definition of the desired behavior of the system
- Captures functional and structural properties in a platform independent way



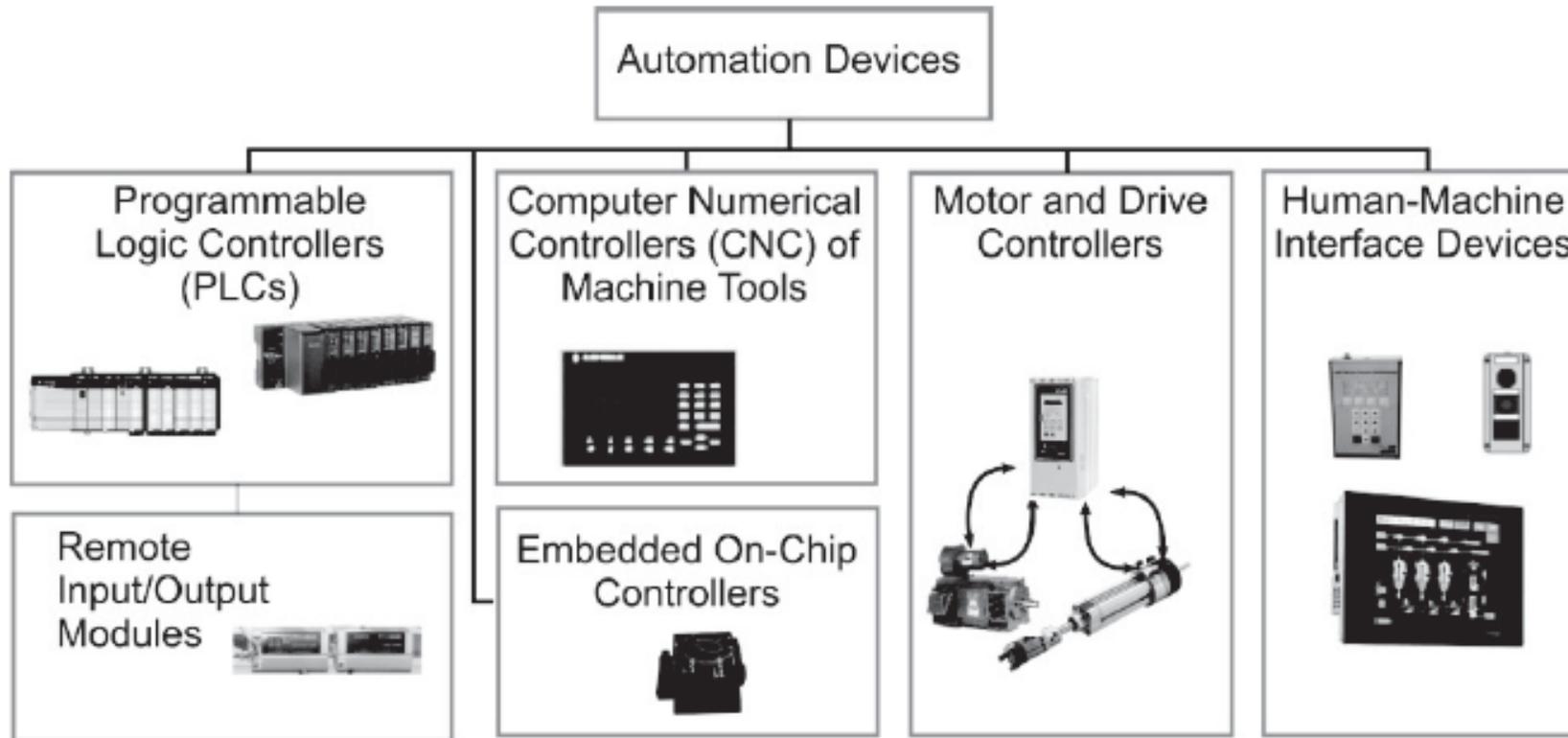
Application: Example



7. Distributed systems

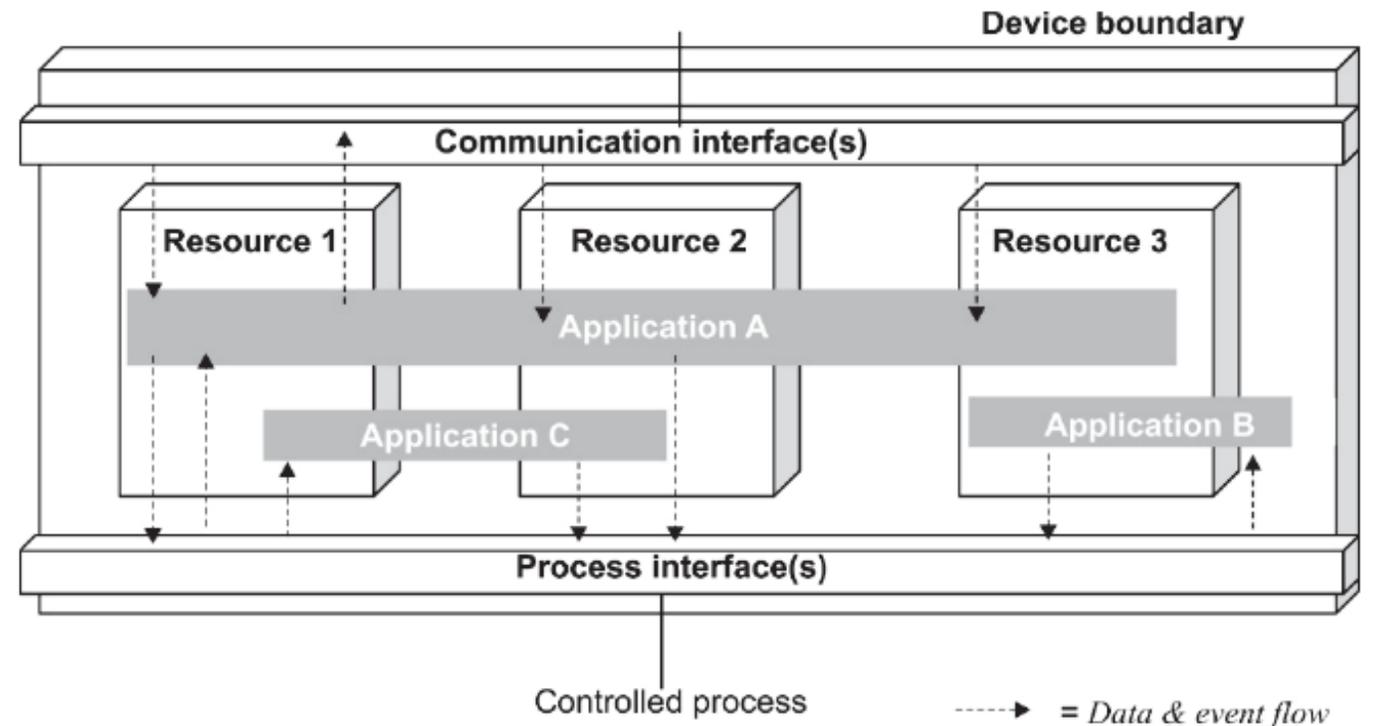
Device

- A model of a functional unit like a computer, microcontroller, or an embedded ship
- Capable to interact with equipment and process information

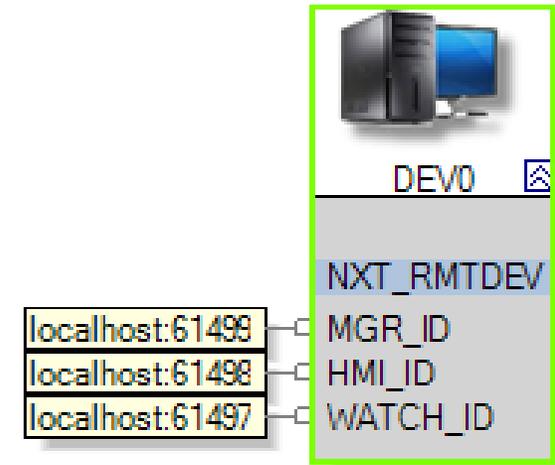
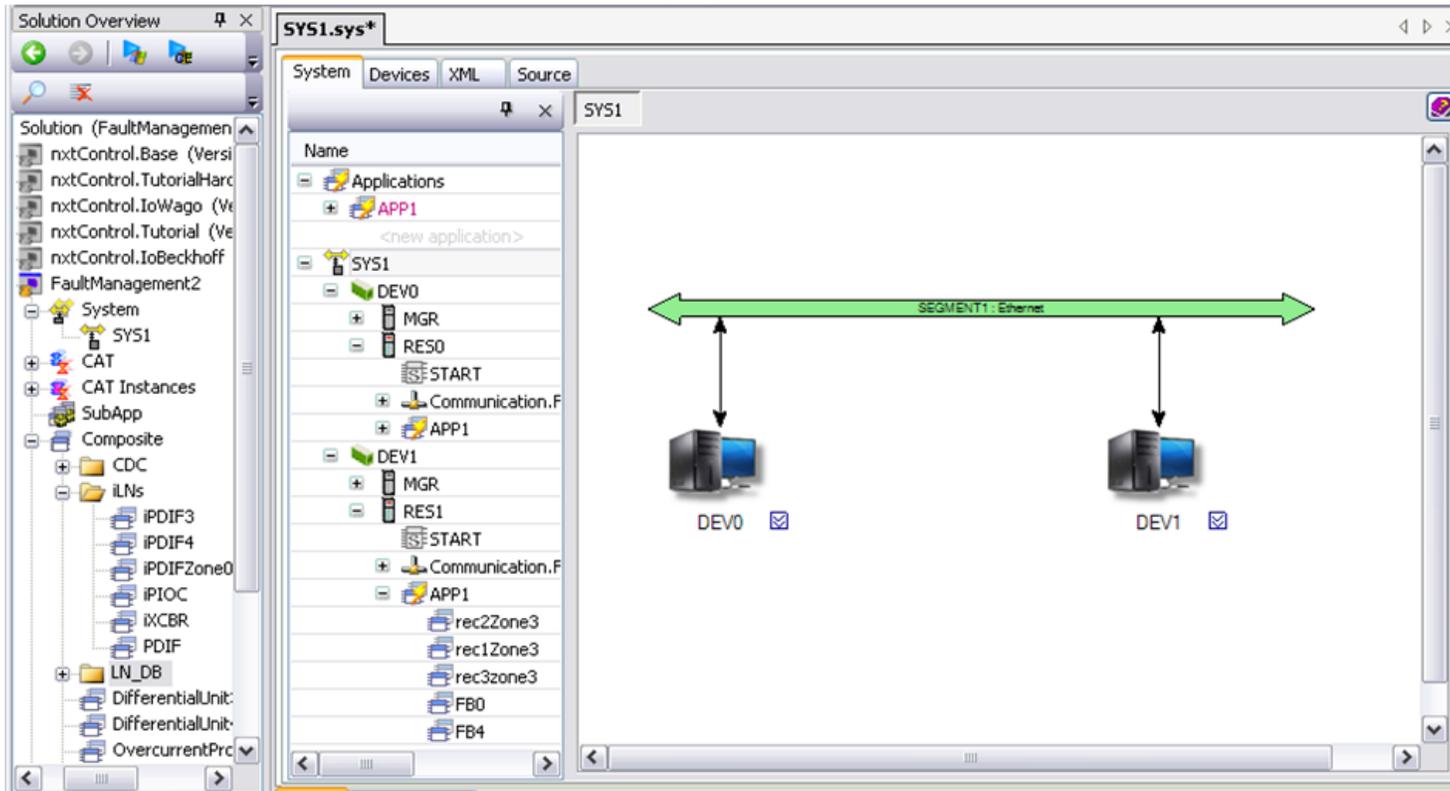


Device

- Abstract model
 - Captures information processing properties of a device
- Process interface
 - Mapping between physical process and resources
 - As events, data or both
- Communication interface
 - Mapping between resources and the information exchanged via network
- Interfaces implemented as SIFBs
 - Specific for a device
- ≥ 0 resources
- Network of FBs



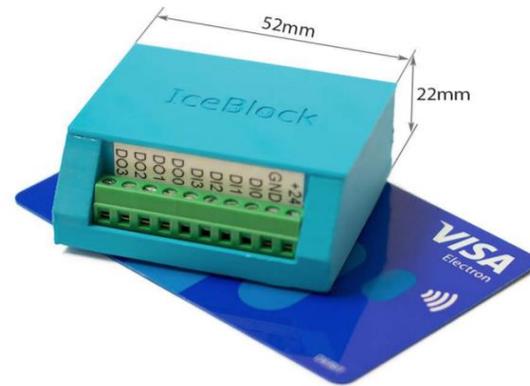
Device: Example



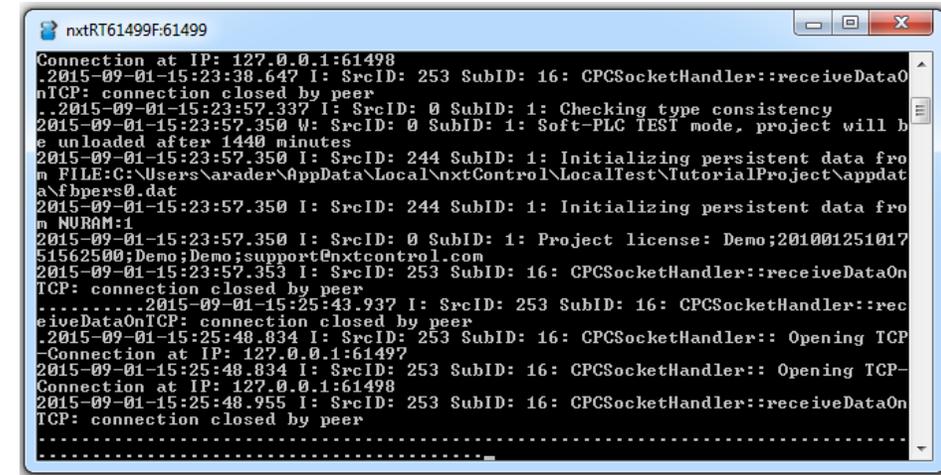
Device: Examples

A Device can represent either a physical PLC, or a soft-PLC.

A soft-PLC is a computer program, which emulates a real controller.



IceBlock controller
compatible with nxtSTUDIO

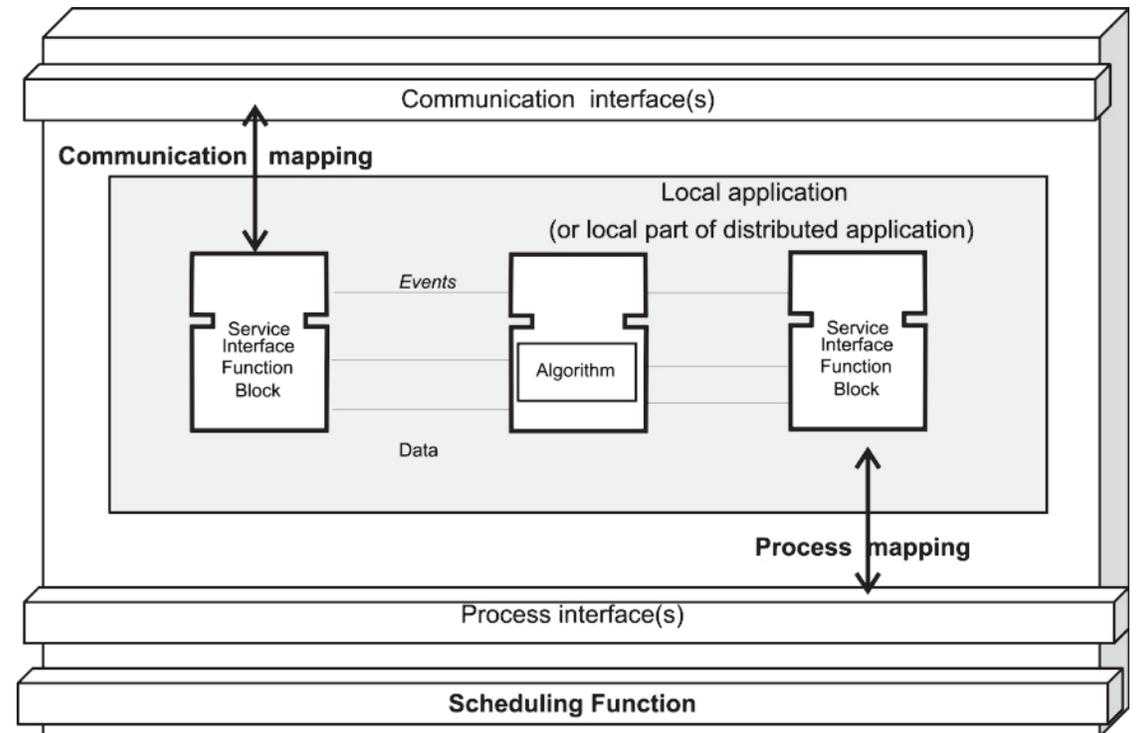
A screenshot of a console window titled "nxtRT61499F:61499" showing a log of network connections and PLC initialization messages. The text includes timestamps, IP addresses, and messages from the CPCSocketHandler and CPCSocketsHandler, such as "Opening TCP-Connection at IP: 127.0.0.1:61498" and "Soft-PLC TEST mode, project will be unloaded after 1440 minutes".

```
nxtRT61499F:61499
Connection at IP: 127.0.0.1:61498
2015-09-01-15:23:38.647 I: SrcID: 253 SubID: 16: CPCSocketHandler::receiveDataOn
TCP: connection closed by peer
2015-09-01-15:23:57.337 I: SrcID: 0 SubID: 1: Checking type consistency
2015-09-01-15:23:57.350 W: SrcID: 0 SubID: 1: Soft-PLC TEST mode, project will be
unloaded after 1440 minutes
2015-09-01-15:23:57.350 I: SrcID: 244 SubID: 1: Initializing persistent data fro
m FILE:C:\Users\arader\AppData\Local\nxtControl\LocalTest\TutorialProject\appdat
a\fbpers0.dat
2015-09-01-15:23:57.350 I: SrcID: 244 SubID: 1: Initializing persistent data fro
m NURAM:1
2015-09-01-15:23:57.350 I: SrcID: 0 SubID: 1: Project license: Demo;201001251017
51562500;Demo;Demo;support@nxtcontrol.com
2015-09-01-15:23:57.353 I: SrcID: 253 SubID: 16: CPCSocketHandler::receiveDataOn
TCP: connection closed by peer
.....2015-09-01-15:25:43.937 I: SrcID: 253 SubID: 16: CPCSocketHandler::rec
eiveDataOnTCP: connection closed by peer
2015-09-01-15:25:48.834 I: SrcID: 253 SubID: 16: CPCSocketHandler:: Opening TCP
-Connection at IP: 127.0.0.1:61497
2015-09-01-15:25:48.834 I: SrcID: 253 SubID: 16: CPCSocketHandler:: Opening TCP-
Connection at IP: 127.0.0.1:61498
2015-09-01-15:25:48.955 I: SrcID: 253 SubID: 16: CPCSocketHandler::receiveDataOn
TCP: connection closed by peer
.....
```

A soft PLC created by nxtSTUDIO

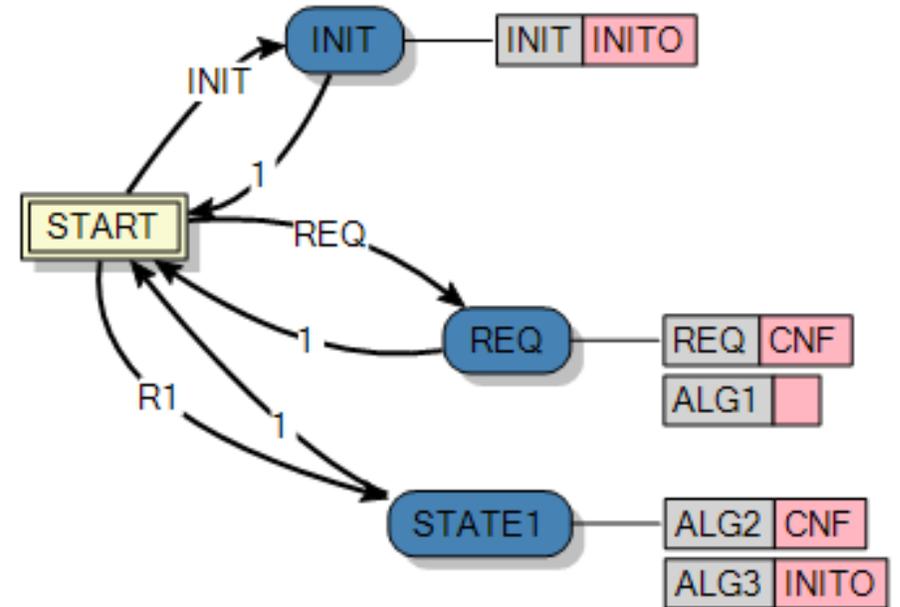
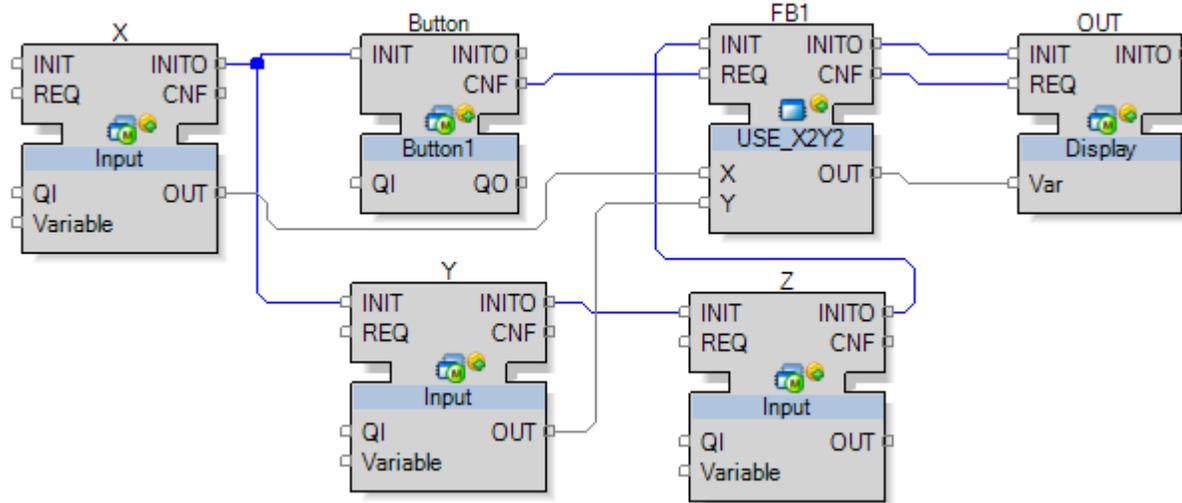
Resource

- An execution container
- Independent control of its operation
- Function of resource
 - Accept data and events from interfaces
 - Process data and events
 - Return data and events to the interfaces
- Physical means to run function blocks
 - Store data and events, algorithms
 - Execution control
- Capabilities:
 - Interfaces
 - Scheduling function



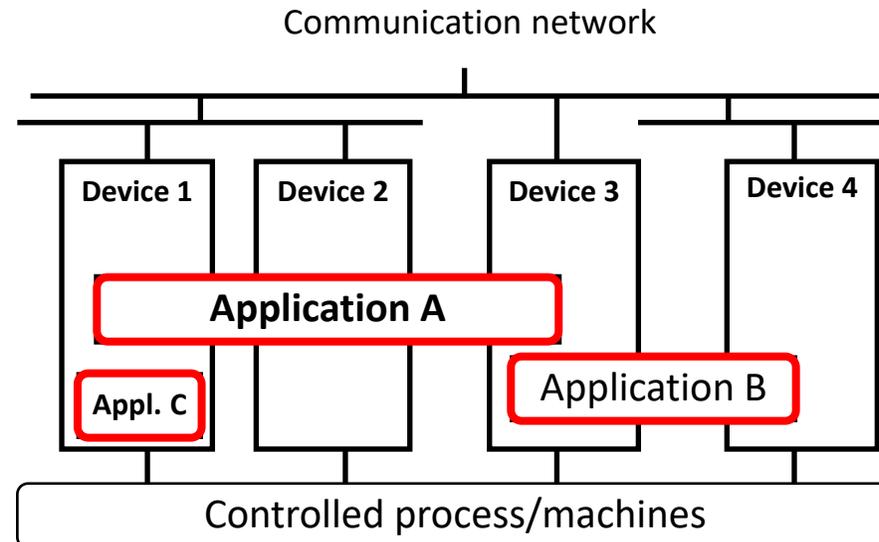
Resource

- Scheduling function of resource schedules algorithms for execution
- Determines sequence
 - Of FBs execution
 - Of algorithms execution



System Configuration

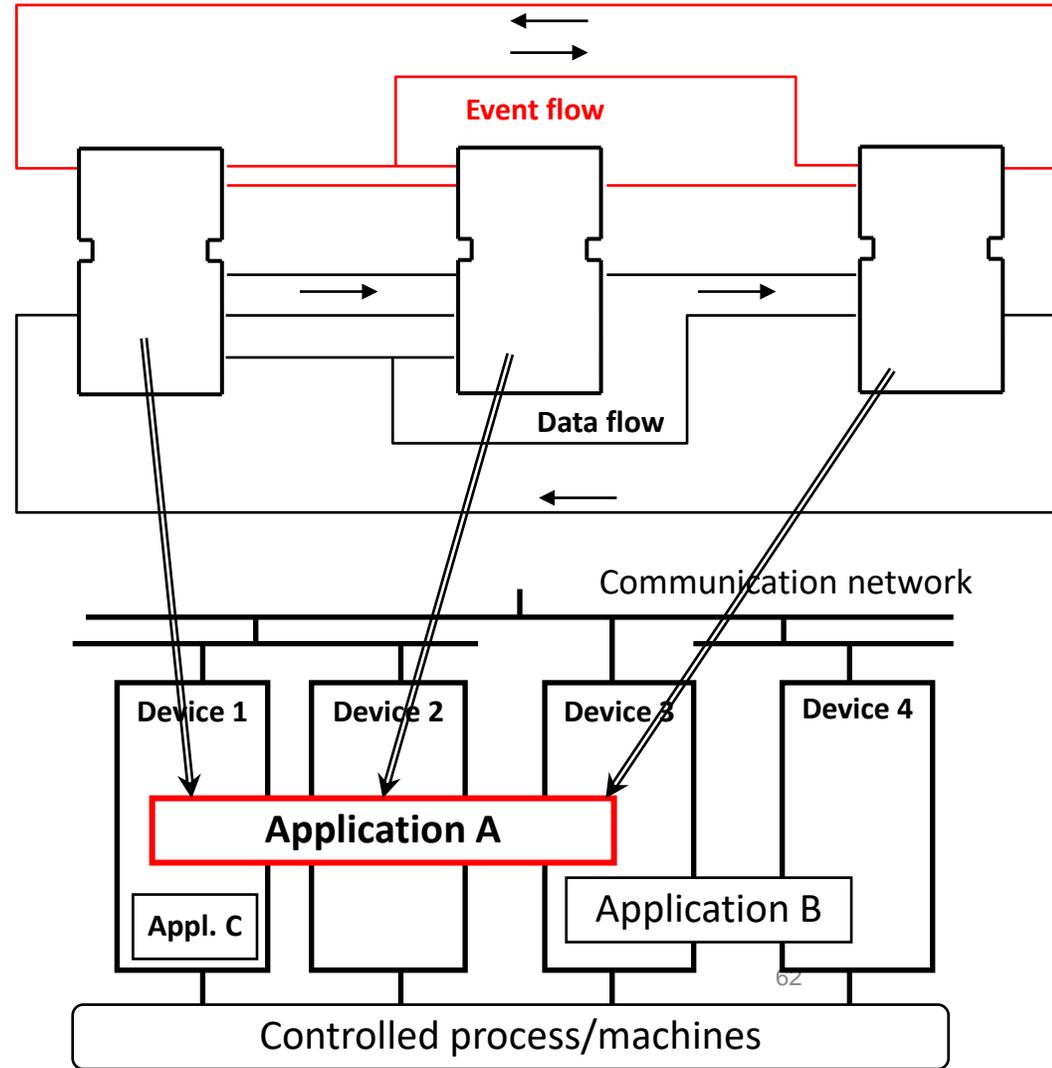
- Represents the physical deployment of the design
- Consists of:
 - Communication network
 - Devices
 - Controlled process and machines



Distribution of Application

An application can be deployed to several devices

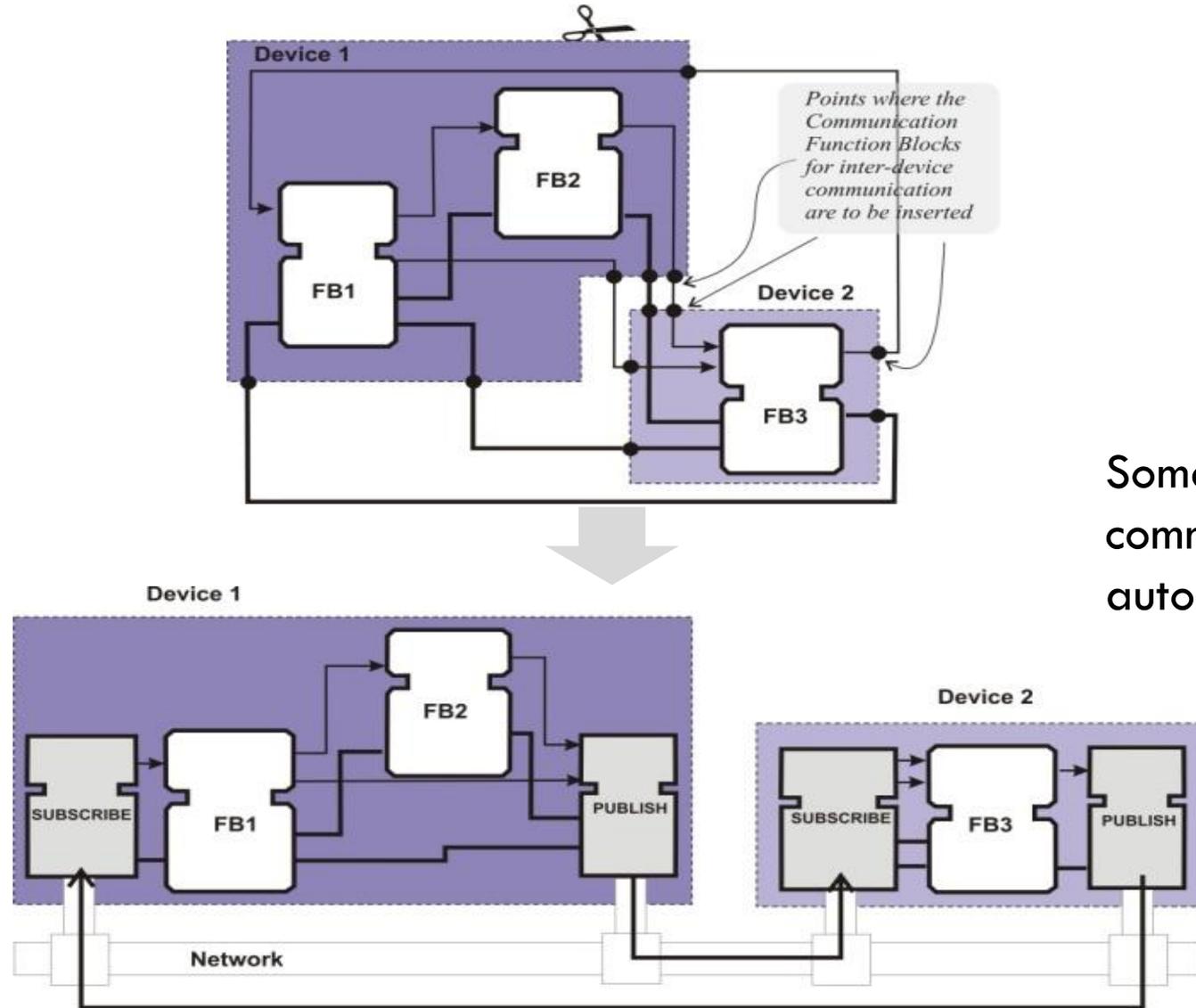
A device in the system can contain and execute parts of different applications



Application
=
Function Block Network

System
=
Communication Network
+
Devices
+
Process/Machines

Distribution of Application: Communication



Some tools insert the communication blocks automatically

Conclusions and remarks

- Although the main power of IEC 61499 is distributed systems design, it can be perfectly used for a central PLC programming
- In this course we will use it this way
- Benefits of IEC 61499 include object-oriented design and openness: easier portability, interoperability and configurability