Program Organization Units

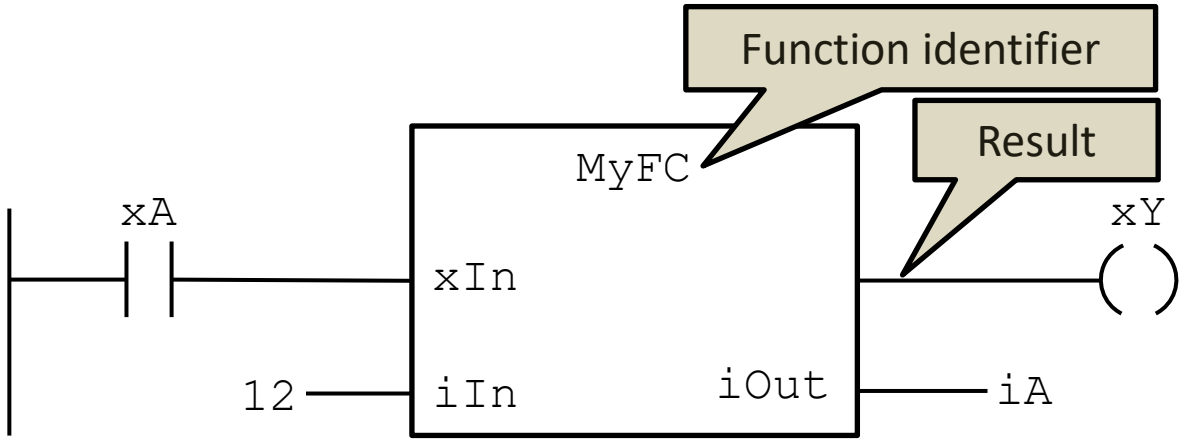
Industrial Control

KOVÁCS Gábor

gkovacs@iit.bme.hu

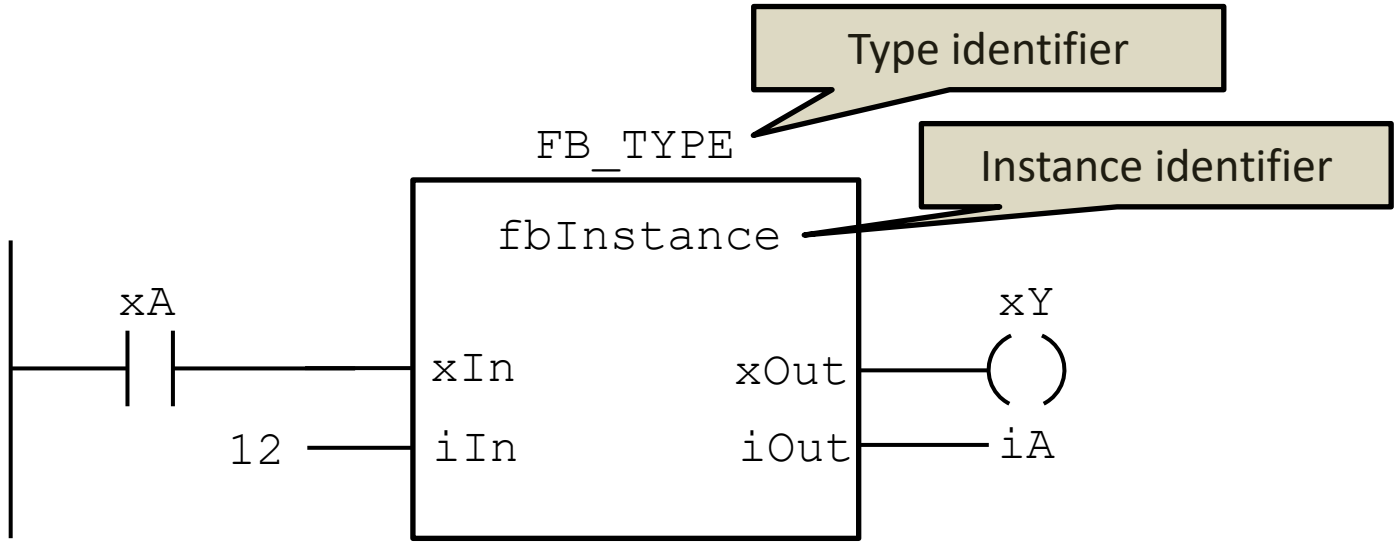
Function call in Ladder Diagram

- The call can be inserted to a rung (network) as a box
- Identifier of the function is displayed at the top, inside the box
- The first input and first output are necessarily Booleans, which allow the box to be inserted to a rung
- The first output is the result (return value) of the function
- In case of non-Boolean return value or no Boolean inputs, the `EN`/`ENO` input/output pair shall be used



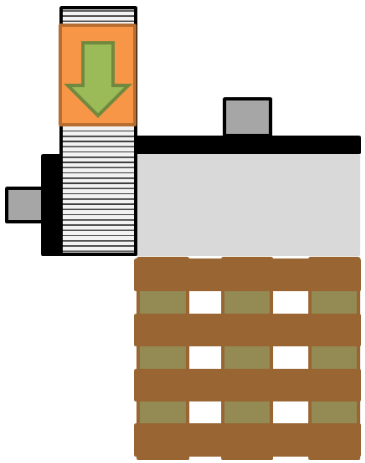
Function block call in Ladder Diagram

- The call can be inserted to a rung (network) as a box
- Type of the instance is displayed above the box, the instance identifier is displayed inside the box
- The first input and first output are necessarily Booleans, which allow the box to be inserted to a rung

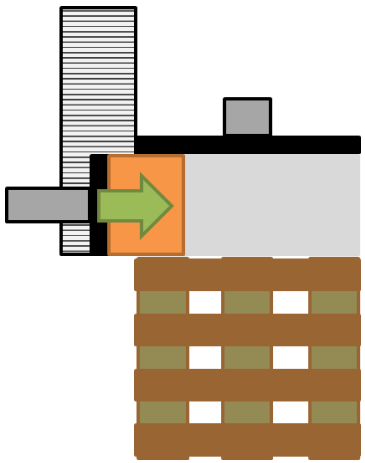


Problem: palletizing system

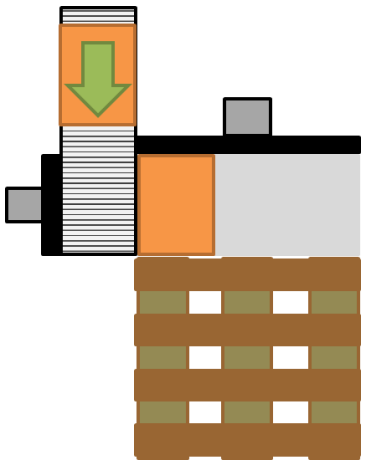
Palletizing system



Boxes to be stacked on a pallet arrive on a roller conveyor.

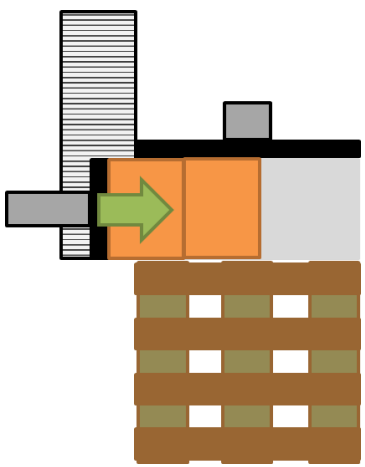


A pneumatic pusher pushes the box at the end of the conveyor to the preparation area.

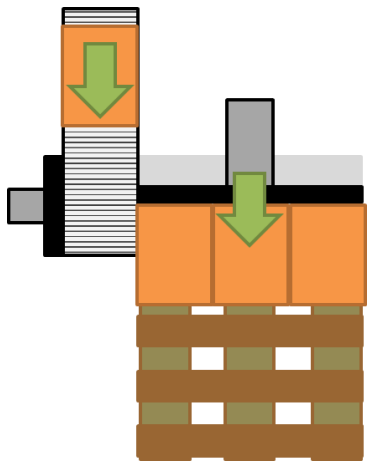


The pusher is retracted before the arrival of the next box.

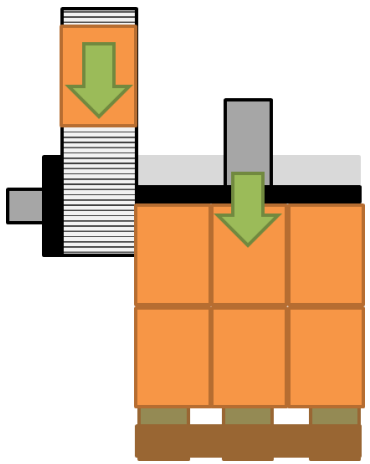
Palletizing system



The preparation area is filled with boxes by the first pusher.

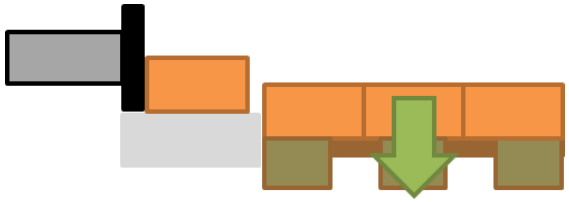


If 3 boxes are transferred to the preparation area, the second pusher forwards the row of boxes onto the pallet.

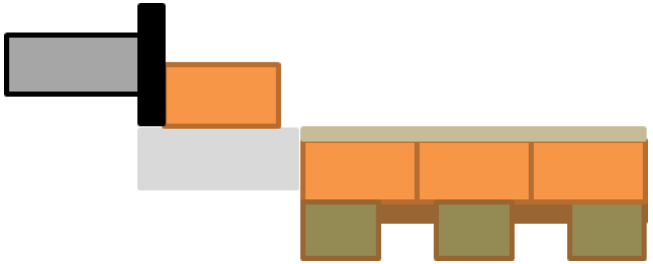


Further boxes arriving on the conveyor are again transferred to the preparation area. If the preparation area is filled with 3 boxes, the second pusher forwards the second row to the pallet.

Palletizing system

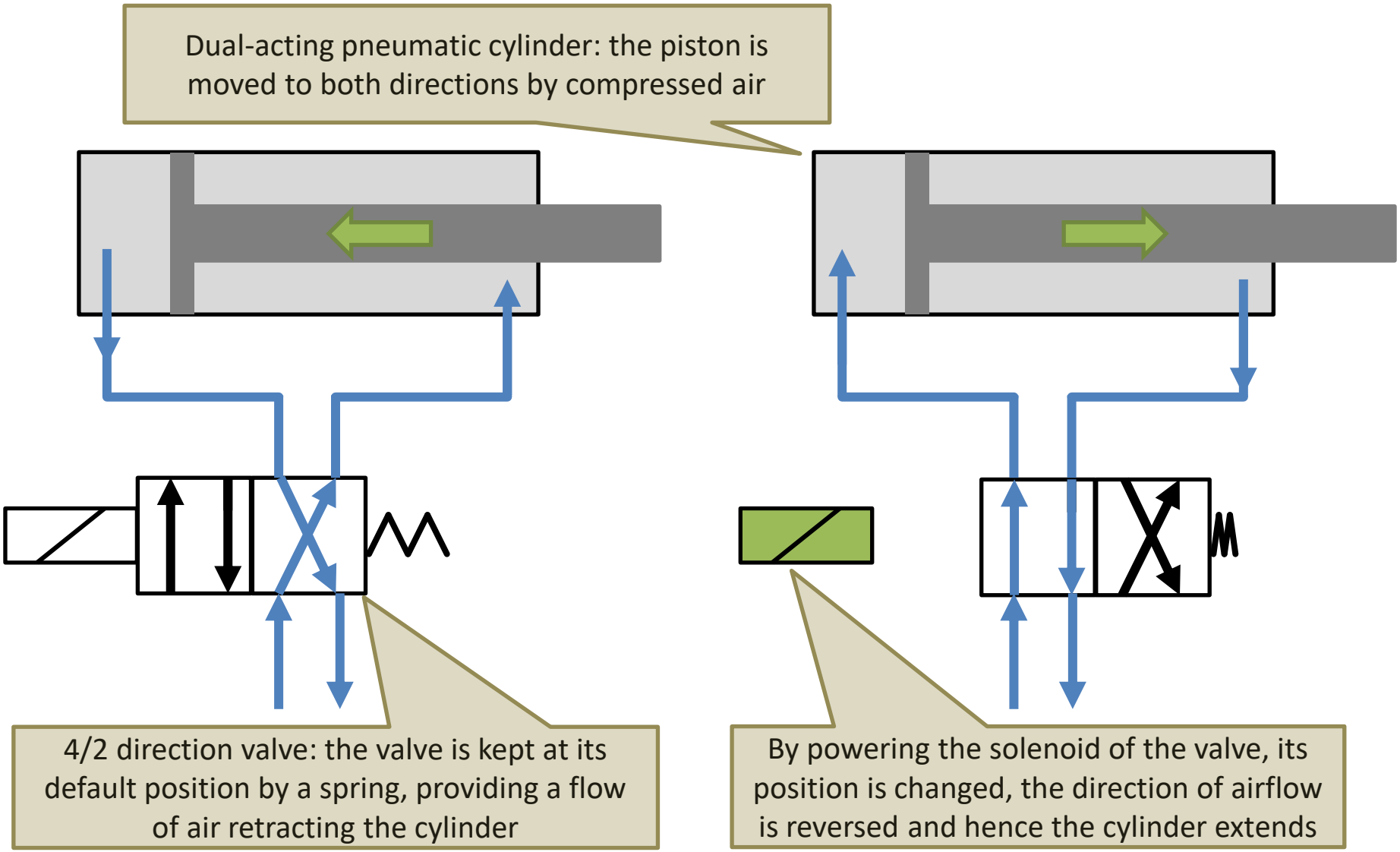


The pallet is fully loaded if two rows of boxes are forwarded. Then a request is sent to another controller to lower the pallet.



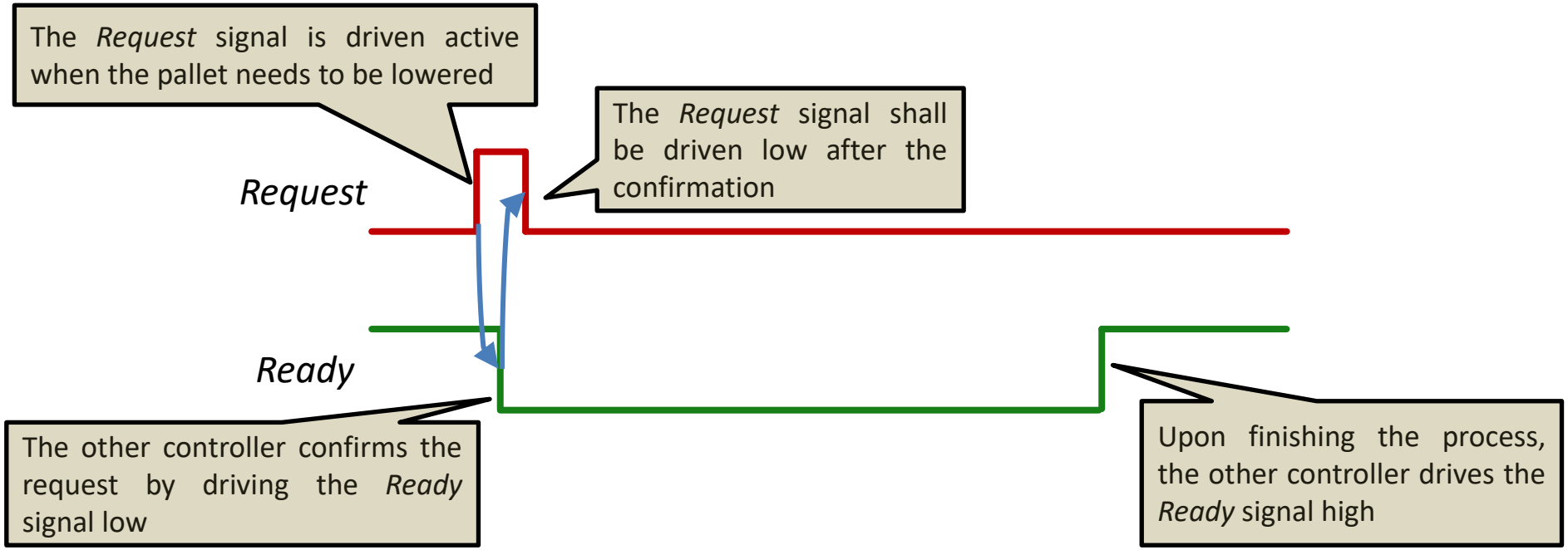
After the pallet is lowered and boxes are covered with a carton sheet, the palletizing system can forward the next layer of boxes onto the pallet.

Operation of pushers

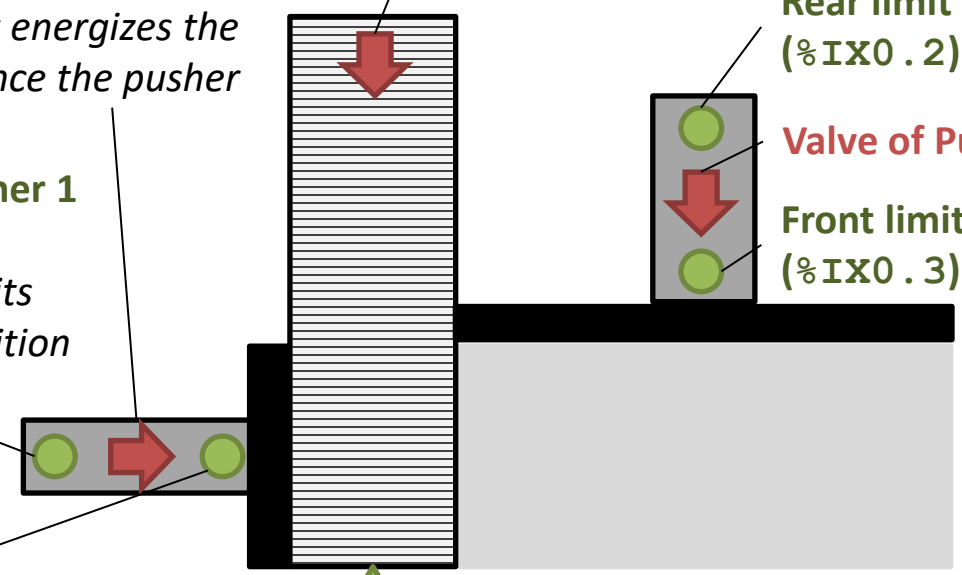


Lowering the pallet

- The pallet is lowered by an other controller, with which we communicate by two digital lines
- Lowering the pallet can be requested by driving the *Request* signal high
- The other controller drives the *Ready* signal high again when the lowering is finished and a new level of boxes can be forwarded



Sensors and actuators



Conveyor on/off (%QX0.2)

Valve of Pusher 1 (%QX0.0)
Active level of the output energizes the solenoid of the valve, hence the pusher extends

Rear limit switch of Pusher 1 (%IX0.0)
Active if the pusher is in its rear (fully retracted) position

Rear limit switch of Pusher 2 (%IX0.2)

Valve of Pusher 2 (%QX0.1)

Front limit switch of Pusher 2 (%IX0.3)

Front limit switch of Pusher 1 (%IX0.1)
Active if the pusher is in its front (fully extended) position

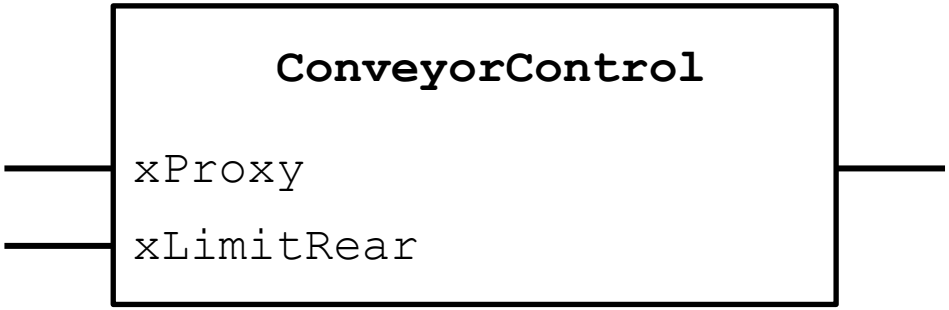
Proximity switch (%IX0.4)
Active if there is a box at the end of the conveyor

Pallet ready (%IX0.5)
Input from an other part of the control system, active level reports that boxes can be forwarded to the pallet

Lowering request (%QX0.3)
Output to an other part of the control system, the pallet is lowered if the signal is active

Conveyor control

- The conveyor can be operated only if
  - there is no box at the end of the conveyor, i.e. the value of the proximity switch is false
  - pusher 1 is in its fully retracted position
- Signals of the sensors are forwarded to the inputs of the function by the main program
- Boolean result (return value) of the function (conveyor on/off) is forwarded to the physical output by the program



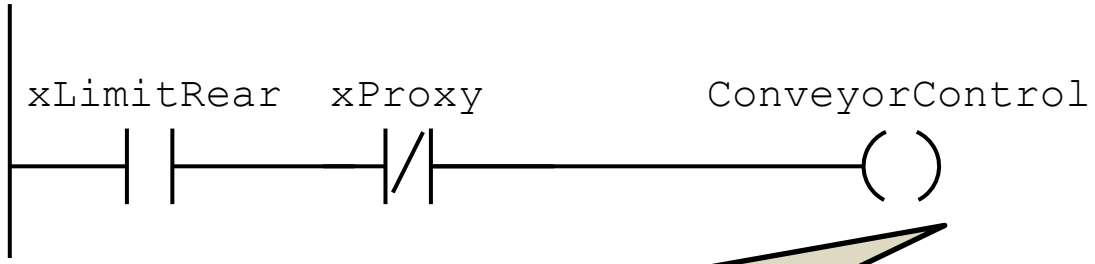
Conveyor control

```
FUNCTION ConveyorControl : BOOL;
VAR_INPUT
   xProxy : BOOL;
   xLimitRear  : BOOL;
END_VAR
```

Data type of the result (return value) is specified during the declaration

Proximity switch at the end of the conveyor

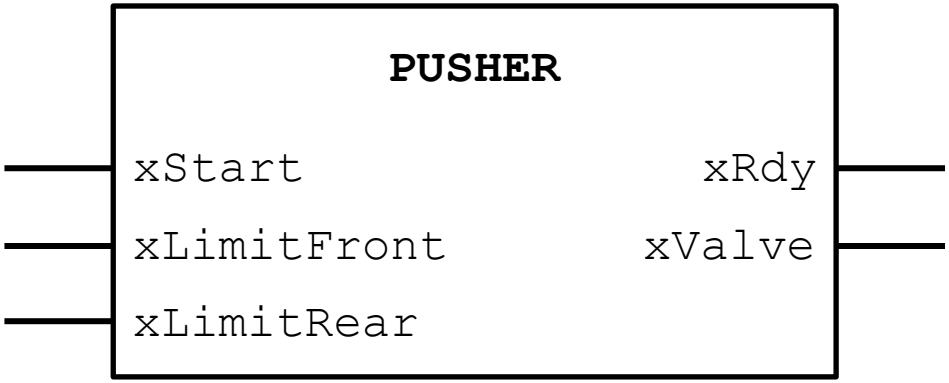
Rear limit switch of pusher 1



The result of the function is set by assigning a value to an implicitly declared variable with the identifier of the function

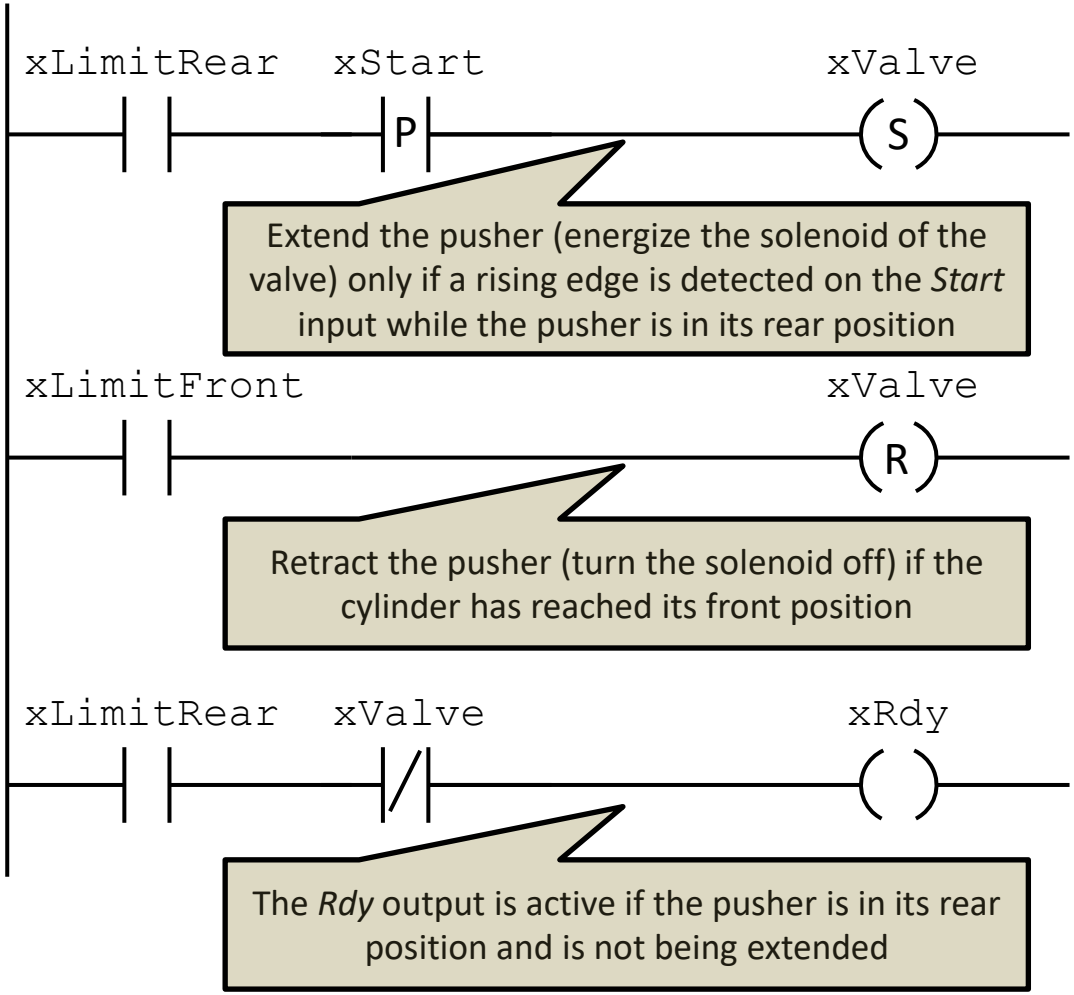
Function Block of a pusher

- If the pusher is in its rear position and there is a rising edge detected on the *Start* input, the pusher shall be extended to its front position and then retracted to its rear position
- Active level of the *Rdy* output reports that an extend-retract cycle has been finished
- In order to define a reusable function block, signals of sensors and actuators are interfaced to the function block as input and output variables – such variables are connected to physical IOs in the main program only



Function Block of a pusher

```
FUNCTION_BLOCK PUSHER
VAR_INPUT
   xStart      : BOOL;
   xLimitFront : BOOL;
   xLimitRear  : BOOL;
END_VAR
VAR_OUTPUT
   xRdy   : BOOL;
   xValve : BOOL;
END_VAR
```



Main program

Physical inputs and outputs are connected to input and output variables of the main program, respectively

```
PROGRAM Palletizer
VAR_INPUT
   xLimitRear1  AT %IX0.0 : BOOL; // Pusher 1 rear limit switch
   xLimitFront1 AT %IX0.1 : BOOL; // Pusher 1 front limit switch
   xLimitRear2  AT %IX0.2 : BOOL; // Pusher 2 rear limit switch
   xLimitFront2 AT %IX0.3 : BOOL; // Pusher 2 front limit switch
   xProxy       AT %IX0.4 : BOOL; // box at the end of the conveyor
   xPalletReady AT %IX0.5 : BOOL; // pallet ready to receive boxes
END_VAR
VAR_OUTPUT
   xValve1  AT %QX0.0 : BOOL; // solenoid of valve driving pusher 1
   xValve2  AT %QX0.1 : BOOL; // solenoid of valve driving pusher 2
   xConv    AT %QX0.2 : BOOL; // input roller conveyor on/off
   xRequest AT %QX0.3 : BOOL; // request to lower the pallet
END_VAR
```

Main program

- Pushers are controlled by function blocks of type `PUSHER`
- Two count-up counters (`CTU`) are used to count the number of boxes at the preparation area and the number of rows forwarded to the pusher
- These function blocks need to be instansiated (formally the same way as declaring local variables)

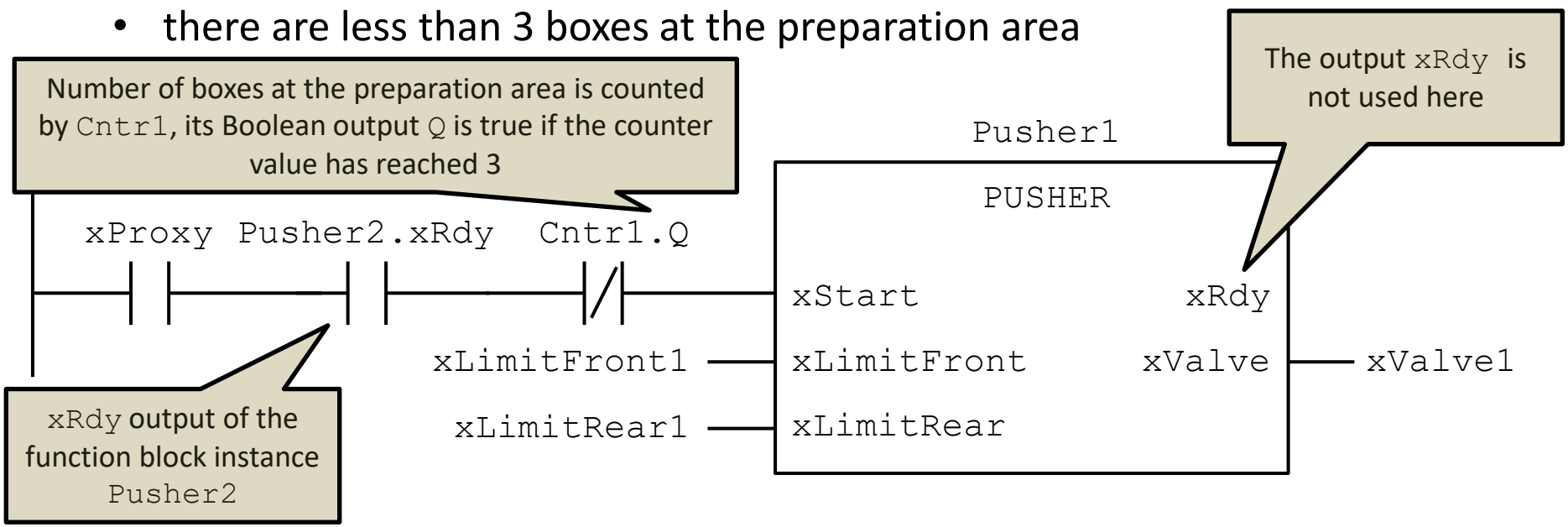
```
VAR
   Pusher1, Pusher2 : PUSHER;
   Cntr1      : CTU := (PV:=3);
   Cntr2      : CTU := (PV:=2);
END_VAR
```

Pusher 2 can be started if 3 boxes are present at the preparation area

The pallet needs to be lowered if 2 rows have been transferred to it

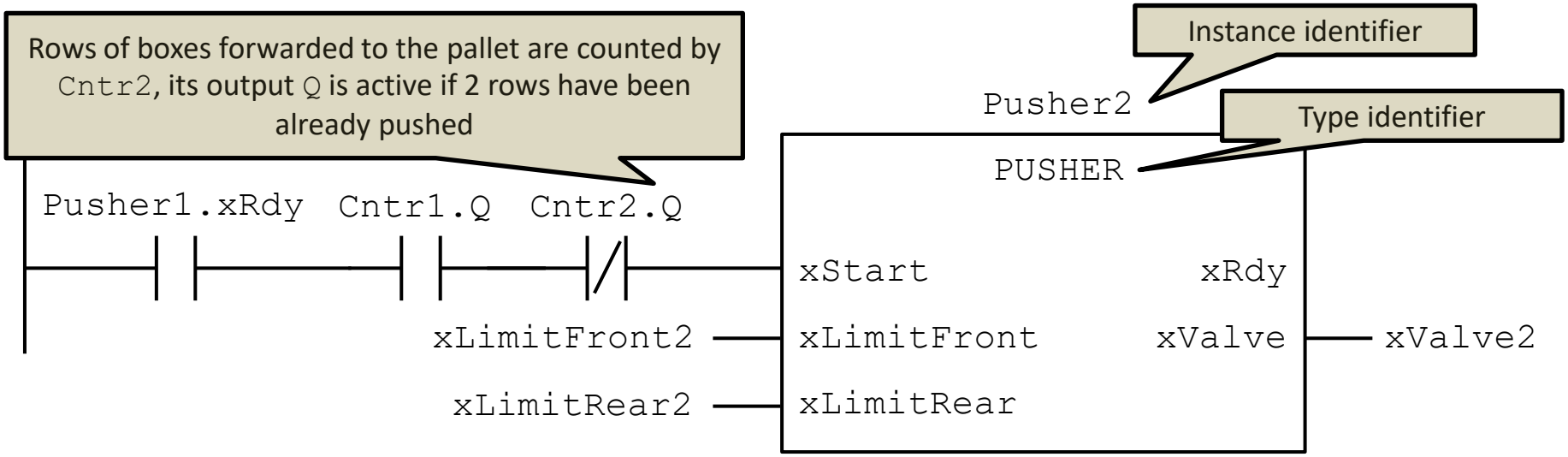
Pusher control

- Input variables of the program, binded to physical inputs, shall be connected to `xLimitFront` and `xLimitRear` inputs of pusher FBs
- Output `xValve` of the function blocks shall be assigned to output variables of the program
- The first pusher shall be started if
  - there is a box at the end of the conveyor
  - the second pusher is ready (collision avoidance)
  - there are less than 3 boxes at the preparation area



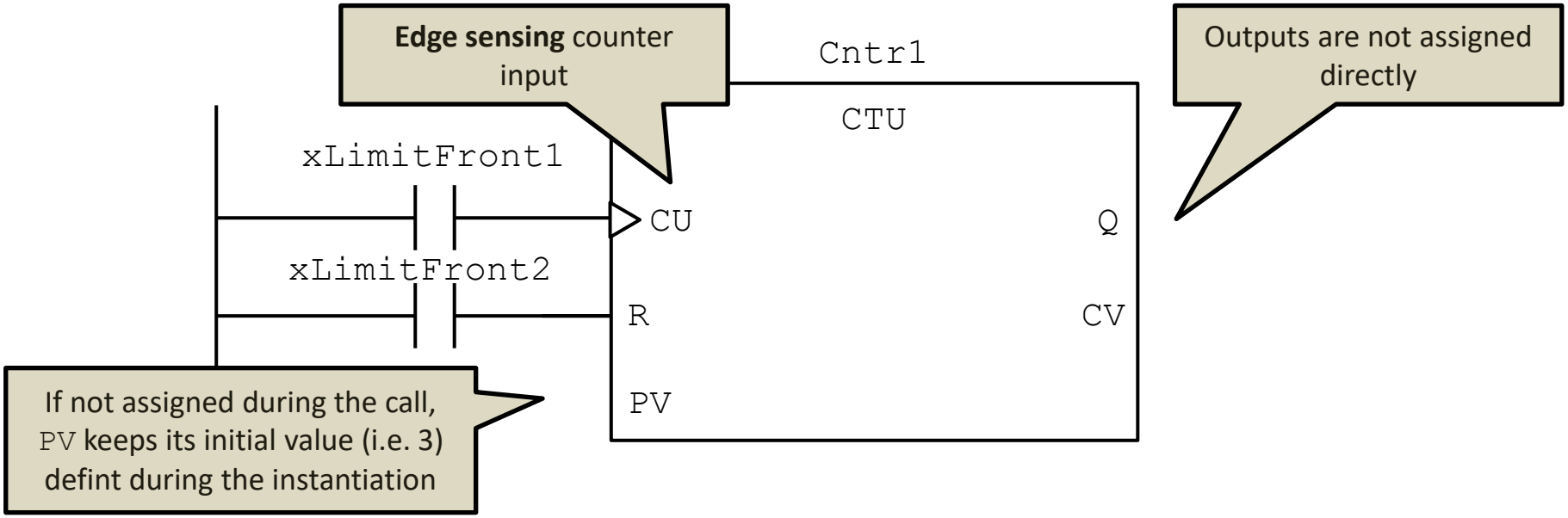
Pusher control

- The second pusher shall be started if
  - the first pusher is ready (collision avoidance)
  - there are 3 boxes at the preparation area
  - the pallet has not been loaded with 2 rows of boxes



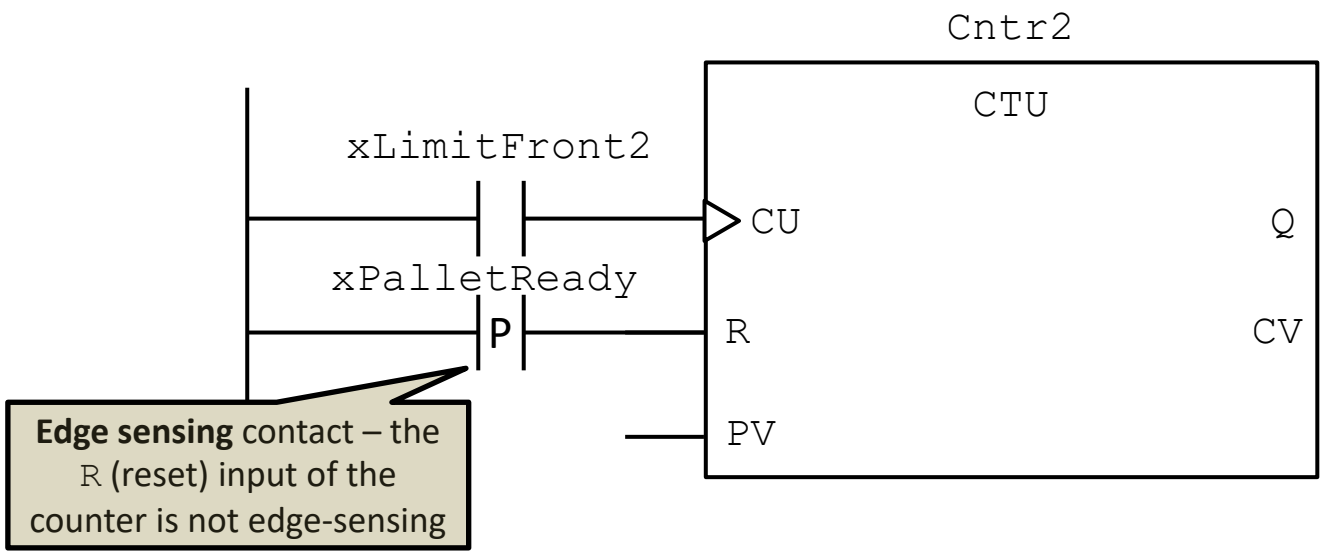
Counters

- Number of boxes forwarded to the preparation area:
  - counter value shall be increased if the first pusher has reached its extended position (box pushed to the preparation area)
  - the counter value shall be reset if the second pusher reaches its extended position (all boxes at the preparation area have been forwarded onto the pallet)



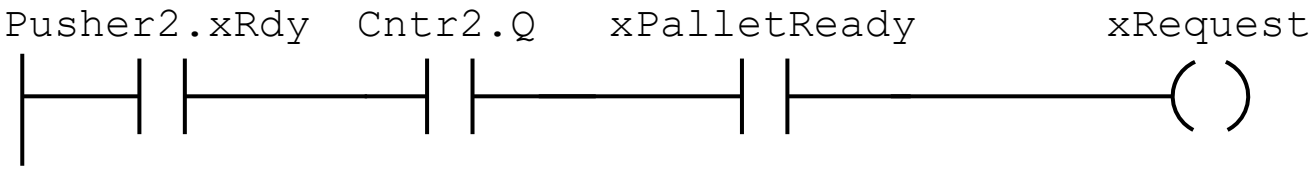
Counters

- Number of rows forwarded onto the pallet:
  - counter shall be increased when the second pusher reaches its front position (a row of boxes has been transferred onto the pallet)
  - the counter value shall be reset if the lowering of the pallet has been finished (rising edge of variable `xPalletReady`)



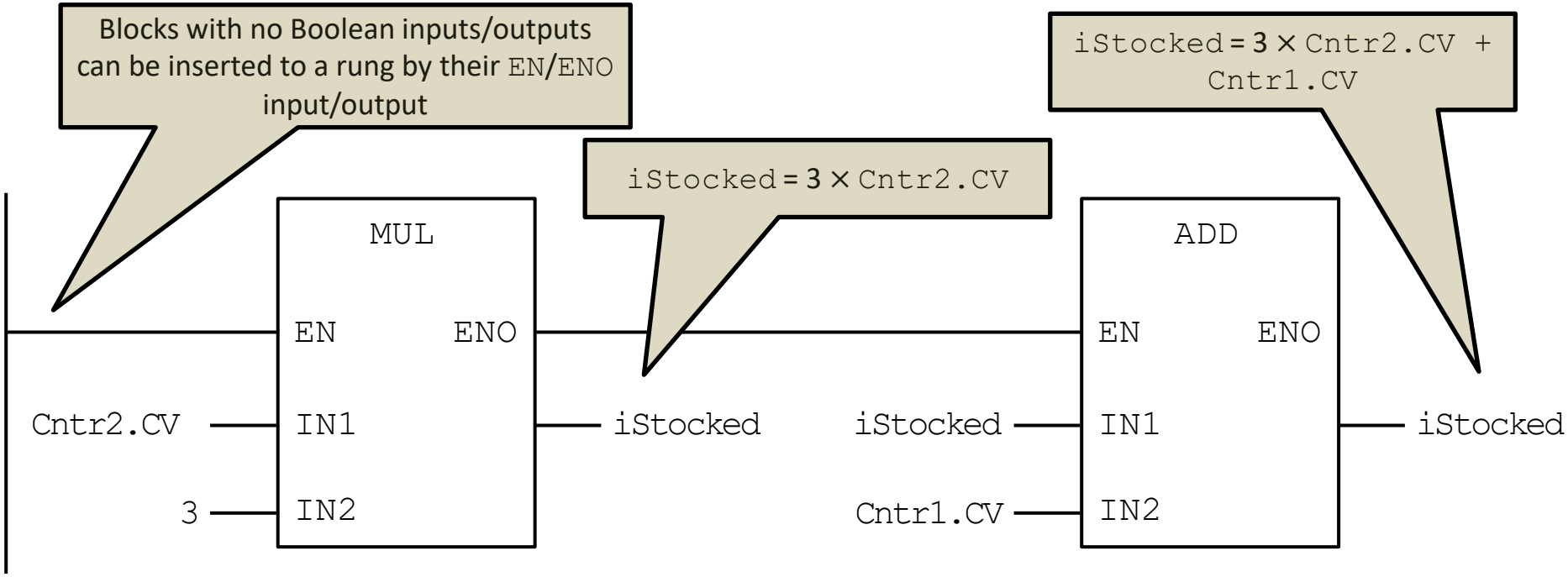
Requesting the lowering of the pallet

- Lowering of the pallet shall be requested if
  - pusher 2 is ready
  - 2 rows have been already transferred to the pallet
  - `xPalletReady` is active (the request shall be turned off when the other controller confirms the request by driving `xPalletReady` low)



Extra

- Calculate the number of boxes forwarded from the conveyor towards the current level and store it in the variable `iStocked`
- Number of boxes forwarded: `Cntr2.CV` × 3 + `Cntr1.Cv`



Implementation in the template project

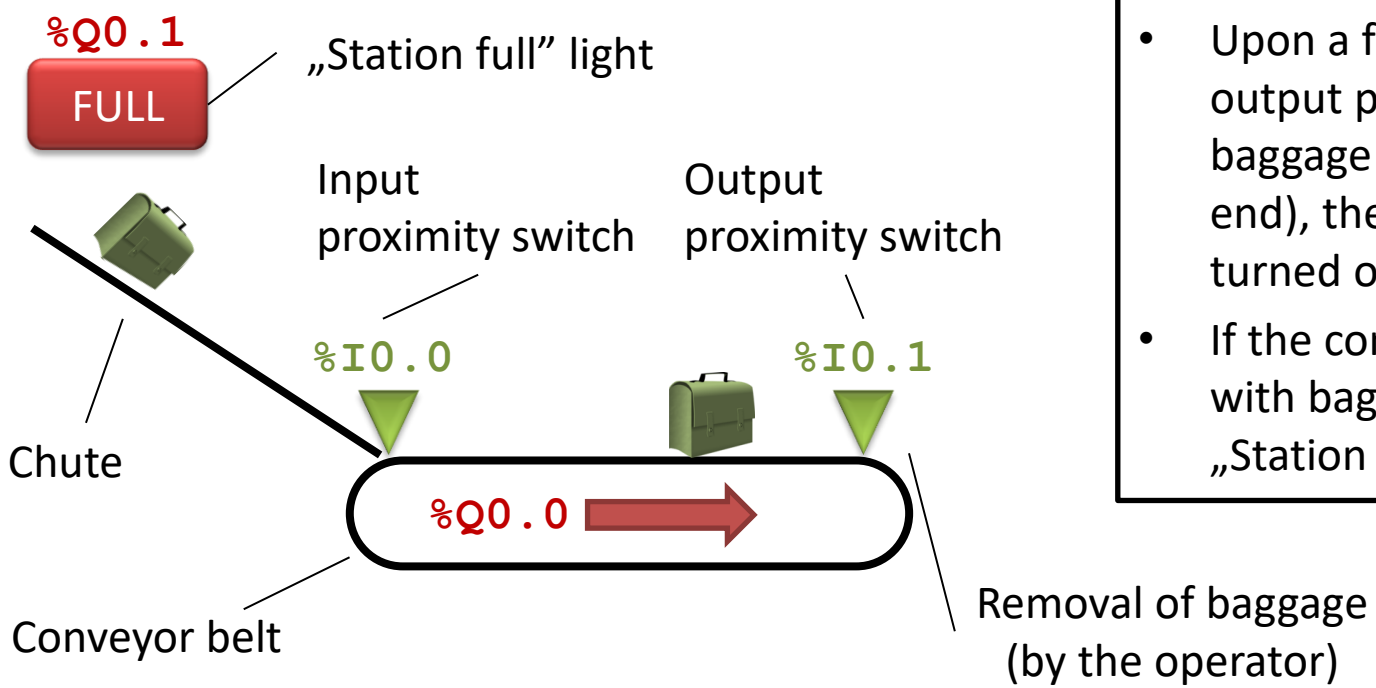
- Unused input and output ports of a call can be deleted by right-clicking the block and selecting the *Remove Unused FB Call Parameters* option
- Don’t forget to add the task executing the program!
  - tasks can be added to the *Task Configuration* element of the project tree
  - the program shall be executed by a *Freewheeling* task
  - priority level of the task (*Priority* parameter) shall be set greater than 0

CODESYS

Problem: Make Up station

Make Up station

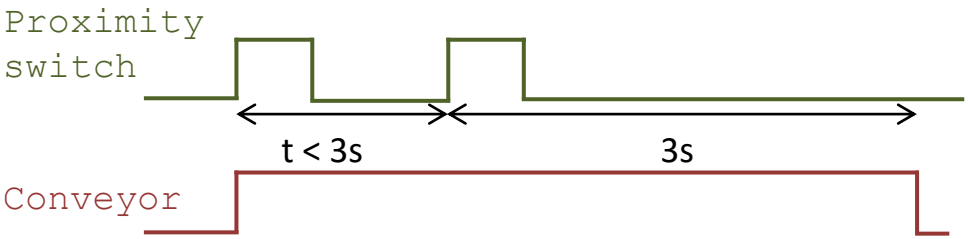
Baggages arrive at the conveyor through a chute. Arriving baggages shall be forwarded to the end of the conveyor, where an operator removes them and place them on a pully. Proximity sensors are located at both ends of the conveyor, reporting if a baggage is present.



Specification

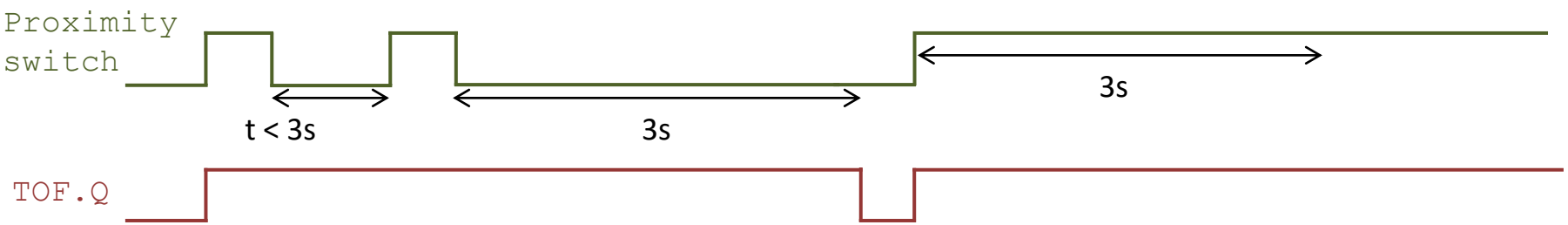
- If a baggage arrives from the chute, the conveyor shall be turned on for 3 seconds
- Upon a falling edge of the output proximity sensor (a baggage is remove from the end), the conveyor shall be turned on for 1 second
- If the conveyor is fully loaded with baggages, turn on the „Station full" light

Timing at the input side



If a new baggage arrives within 3 second after starting the conveyor by the arrival of the first baggage, the conveyor shall operate for additional 3 seconds

Solution: `TOF` timer – falling edge of the input proximity switch is delayed by 4 seconds



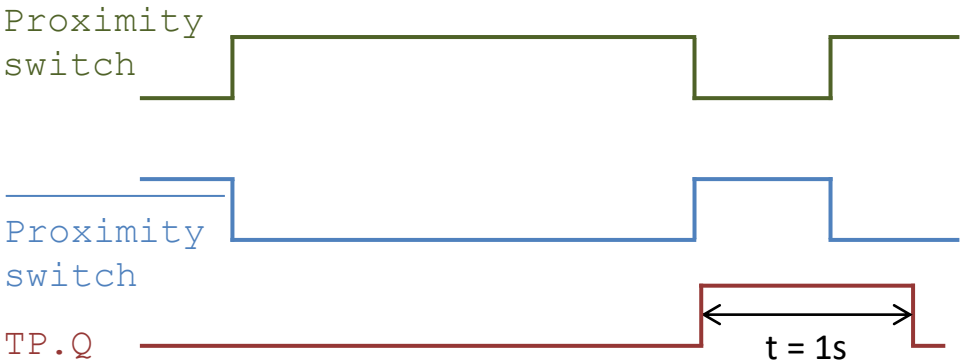
Problems:

- the timer is started when the baggage leaves the input proximity switch (acceptable delay)
- if the conveyor is full and the baggage can not move, the input proximity switch will stay active hence the conveyor will be running – it shall be solved afterwards

Timing at the output side

If a baggage is removed az the output (falling edge of output proximity switch), the conveyor shall be moved for 1 second

Solution: `TP` timer with the negated value of the proximity sensor as its input

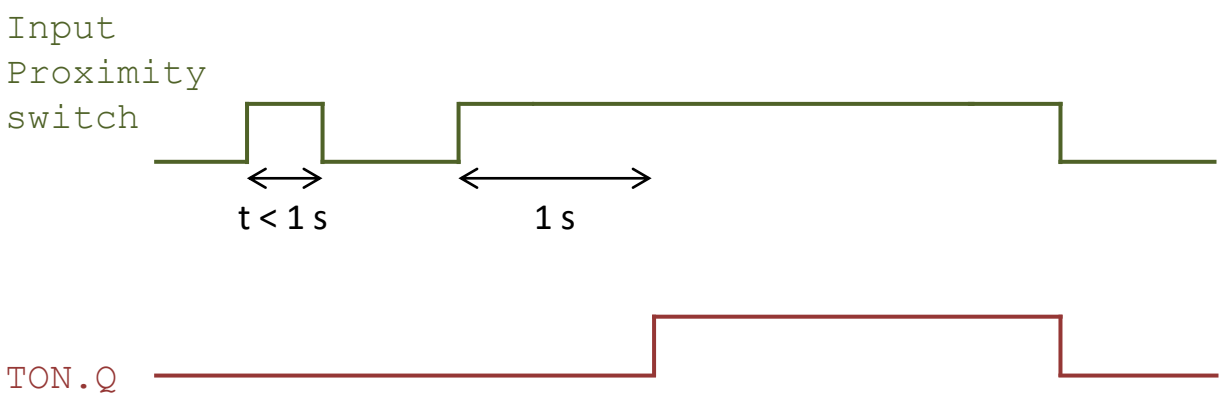


Problem: if the conveyor is empty at startup (i.e. the value of the proximity switch is 0), the conveyor is started for 1 second

Solution: instead of its level, use the falling edge of the proximity switch as input of the `TP` timer

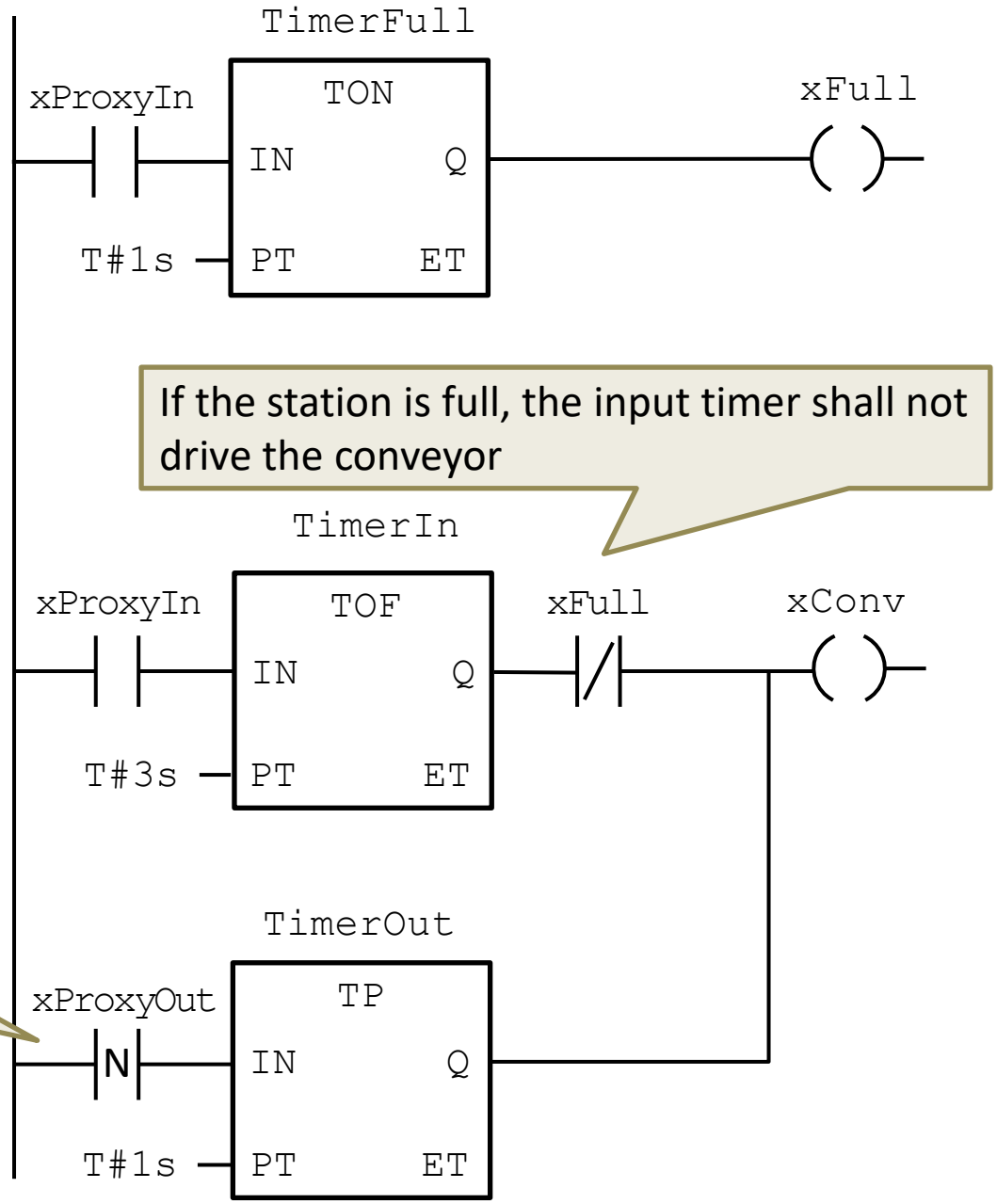
Full conveyor signal

- When shall we consider the conveyor to be full?
- If a baggage arrives and the conveyor is started, but the baggage does not leave the vicinity of the input proximity switch
- If the baggage leaves later, the station is not considered full anymore
- Solution: `TON` timer



```
PROGRAM MakeUp

VAR_INPUT
 xProxyIn  AT %I0.0 : BOOL;
 xProxyOut AT %I0.1 : BOOL;
END_VAR
VAR_OUTPUT
 xConv AT %Q0.0 : BOOL;
 xFull AT %Q0.1 : BOOL;
END_VAR
VAR
 TimerIn   : TOF;
 TimerOut  : TP;
 TimerFull : TON;
END_VAR
```



The edge sensing contact provides starting the conveyor at startup

Implementation in the template project

- Unused input and output ports of a call can be deleted by right-clicking the block and selecting the *Remove Unused FB Call Parameters* option

- Don’t forget to add the task executing the program!
  - tasks can be added to the *Task Configuration* element of the project tree
  - the program shall be executed by a *Freewheeling* task
  - priority level of the task (*Priority* parameter) shall be set greater than 0

CODESYS