MASTER IN ENTREPRENEURSHIP
INNOVATION MANAGEMENT
IN COLLABORATION WITH MIT SLOAN

IN COLLABORATION WITH
MIT MANAGEMENT SLOAN SCHOOL
UNIVERSITÀ DEGLI STUDI DI NAPOLI PARTHENOPE

MASTER MEIM 2021-2022

Programming exercises

Sara Dubbioso

A cura del prof. Lorem Ipsum

Prof. Di Economia e Management all'Università degli Studi di Napoli Parthenope

Lecture overview

- Ladder Diagram (LD) language
  - Boolean operators
  - Logic functions
  - Programming exercises (CODESYS and FACTORY IO)

Software overview

CODESYS

- Integrated **development environment** for programming controller applications according to the standard IEC 61131-3
- Download: https://store.codesys.com/en/

FACTORY IO

- 3D **factory simulation** for learning automation technologies
- offers scenes inspired by typical industrial applications
- Download (30 days trial): https://factoryio.com/start-trial

Boolean algebra

Boolean algebra is the branch of algebra in which the **values of the variables are** the truth values **true** and **false**

- usually are represented with the bits (or binary digits), namely **1** and **0**
- **logic sentences** have an equivalent expression in Boolean algebra

From **George Boole**, an English mathematician of the 1800s

- Introduced in his first book "*The Mathematical Analysis of Logic*", in 1847

Boolean operators

The **truth of logic sentences** can be systematically proven by **logic equation**

The basic operations of Boolean algebra are **conjunction**, **disjunction**, and **negation**

- expressed with the corresponding **binary operators AND**, and **OR** and the **unary operator NOT**

Logical operation	Operator	Notations		
Conjunction	**AND**	a **AND** b	a ∧ b	a · b
Disjunction	**OR**	a **OR** b	a ∨ b	a + b
Negation	**NOT**	**NOT** a	¬ a	- a

Boolean operators

Negation, NOT, ¬

Takes a single Boolean value, either true or false, and **negates it**

- **Flips** true to false, and false to true

Boolean operators

Negation, NOT, ¬

Takes a single Boolean value, either true or false, and **negates it**

- **Flips** true to false, and false to true

Logic Table

a	NOT a
0	1
1	0

Boolean operators

Negation, NOT, ¬

Takes a single Boolean value, either true or false, and **negates it**

- **Flips** true to false, and false to true

Ladder Diagram

Logic Table

a	NOT a
0	1
1	0

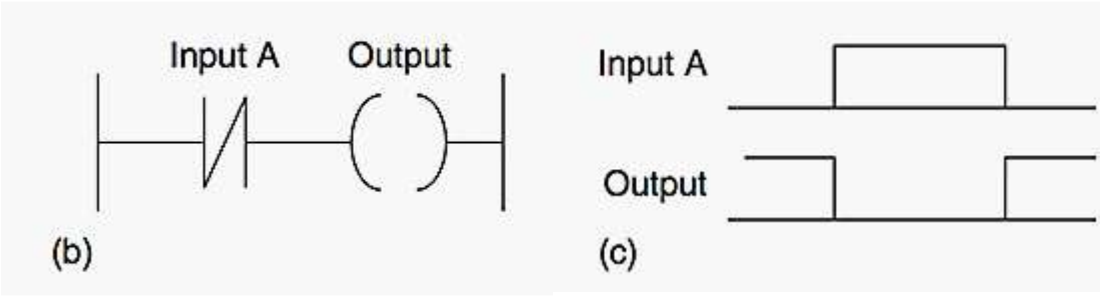
Boolean operators

Negation, NOT, ¬

Takes a single Boolean value, either true or false, and **negates it**

- **Flips** true to false, and false to true

Ladder Diagram



Logic Table

a	NOT a
0	1
1	0

Boolean operators

Conjunction, AND, ∧ , ·

Takes two inputs but still has a single output

- **the output is true only if both inputs are true**

In a logical sentence, you are telling the truth only when you **are never lying**

Boolean operators

Conjunction, AND, ∧ , ·

Takes two inputs but still has a single output

- **the output is true only if both inputs are true**

In a logical sentence, you are telling the truth only when you **are never lying**

<u>My name is Sara</u> AND <u>I'm wearing pants</u>

<u>My name is Sara</u> AND <u>I'm wearing a dress</u>

Boolean operators

Conjunction, AND, ∧ , ·

Takes two inputs but still has a single output

- **the output is true only if both inputs are true**

In a logical sentence, you are telling the truth only when you **are never lying**

My name is Sara ✓ **AND** I'm wearing pants ✓ → ✓

My name is Sara **AND** I'm wearing a dress ✗ → ✗

Boolean operators

Conjunction, AND, ∧ , ·

Takes two inputs but still has a single output

- **the output is true only if both inputs are true**

Boolean operators

Conjunction, AND, ∧ , ·

Takes two inputs but still has a single output

- **the output is true only if both inputs are true**

Logic Table

a	b	a AND b
0	0	0
1	0	0
0	1	0
1	1	1

Boolean operators

Conjunction, AND, ∧ , ·

Takes two inputs but still has a single output

- **the output is true only if both inputs are true**

Ladder Diagram

Logic Table

a	b	a AND b
0	0	0
1	0	0
0	1	0
1	1	1

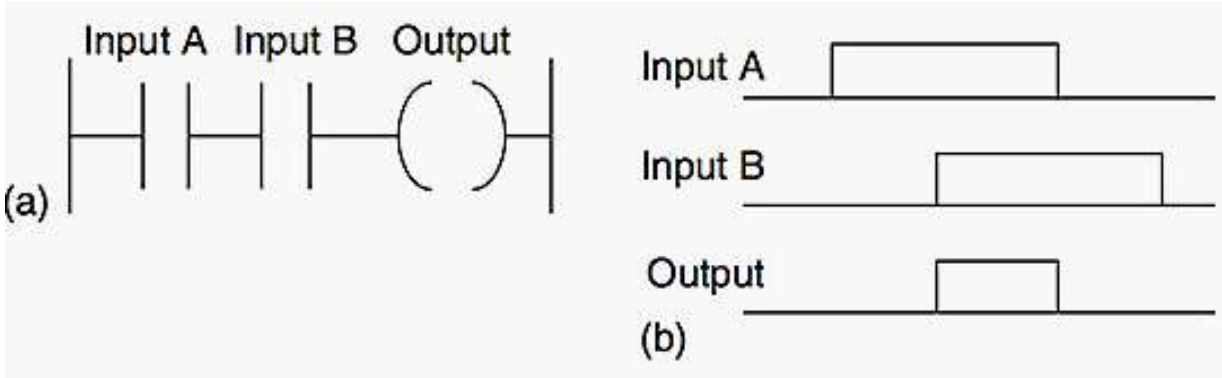
Boolean operators

Conjunction, AND, ∧ , ·

Takes two inputs but still has a single output

- **the output is true only if both inputs are true**

Ladder Diagram: coils series



Logic Table

a	b	a AND b
0	0	0
1	0	0
0	1	0
1	1	1

Boolean operators

Disjunction, OR, ∨ , +

Takes two inputs but still has a single output

- **the output is false only if both inputs are false**

Boolean operators

Disjunction, OR, ∨ , +

Takes two inputs but still has a single output

- **the output is false only if both inputs are false**

Just one sentence needs to be true for the whole sentence to be true

<u>My name is Dua Lipa</u> ✗ **OR** <u>I'm wearing pants</u> ✓ ✓

Boolean operators

Disjunction, OR, ∨ , +

Takes two inputs but still has a single output

- **the output is false only if both inputs are false**

Boolean operators

Disjunction, OR, ∨ , +

Takes two inputs but still has a single output

- **the output is false only if both inputs are false**

Logic Table

a	b	a OR b
0	0	0
1	0	1
0	1	1
1	1	1

Boolean operators

Disjunction, OR, ∨ , +

Takes two inputs but still has a single output

- **the output is false only if both inputs are false**

Ladder Diagram

Logic Table

a	b	a OR b
0	0	0
1	0	1
0	1	1
1	1	1

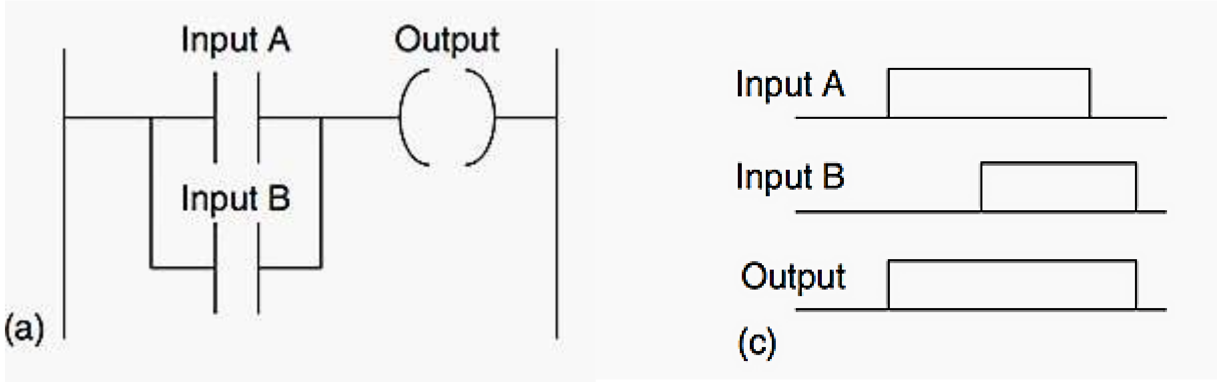
Boolean operators

Disjunction, OR, ∨ , +

Takes two inputs but still has a single output

- **the output is false only if both inputs are false**

Ladder Diagram (coils parallel)



Logic Table

a	b	a OR b
0	0	0
1	0	1
0	1	1
1	1	1

Logic function

NAND

An **AND** operator **followed by a NOT** operator

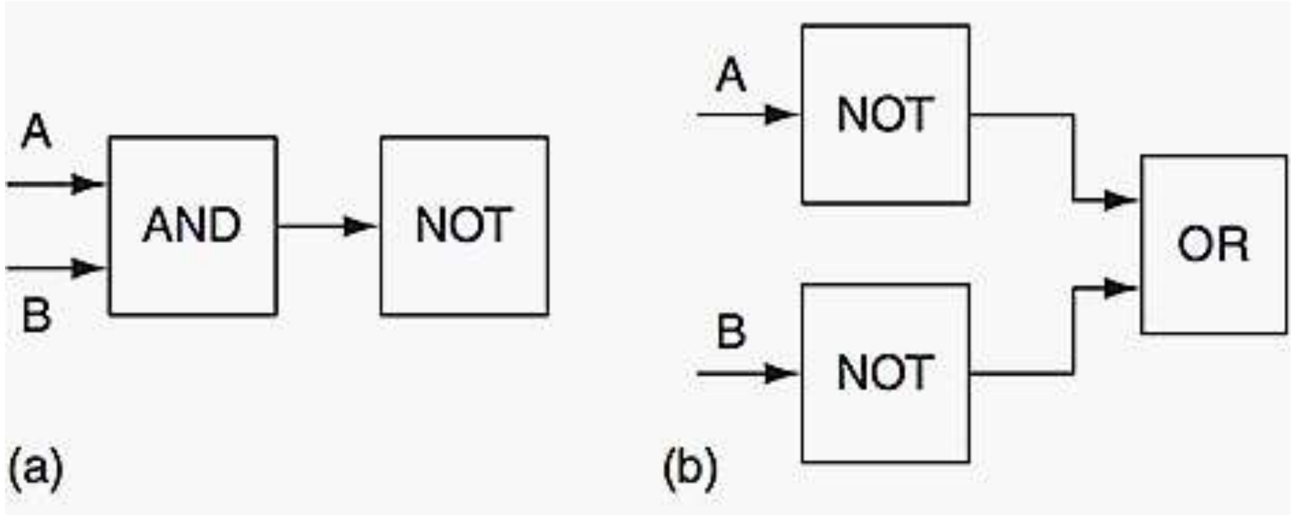
- The NOT operator inverts the output of the AND
- **NAND → NOT** (A **AND** B)

Logic function

NAND

An **AND** operator **followed by a NOT** operator

- The NOT operator inverts the output of the AND
- An alternative is to put a **NOT on each input,** then **follow by an OR**



NOT (a **AND** b)

=

(**NOT** a) **OR** (**NOT** b)

Logic function

NAND

An **AND** operator **followed by a NOT** operator

- The NOT operator inverts the output of the AND

a	b	a AND b	NOT (a AND b)
0	0	0	1
1	0	0	1
0	1	0	1
1	1	1	0

Logic function

NAND

An **AND** operator **followed by a NOT** operator

- An alternative is to put a **NOT on each input,** then **follow by an OR**

Ladder Diagram

a	b	a NAND b
0	0	1
1	0	1
0	1	1
1	1	0

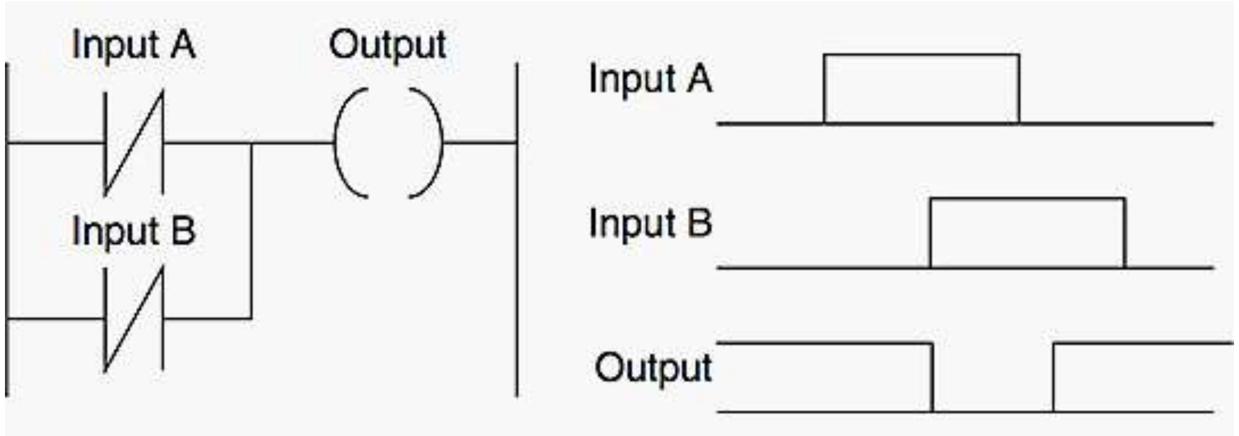
Logic function

NAND

An **AND** operator **followed by a NOT** operator

- An alternative is to put a **NOT on each input,** then **follow by an OR**

Ladder Diagram



a	b	a NAND b
0	0	1
1	0	1
0	1	1
1	1	0

Logic function

NOR

Combination of a **OR** and an **NOT** operator

- The output is 1 when neither inputs is 1

Logic function

NOR

Combination of a **OR** and an **NOT** operator

- The output is 1 when neither inputs is 1

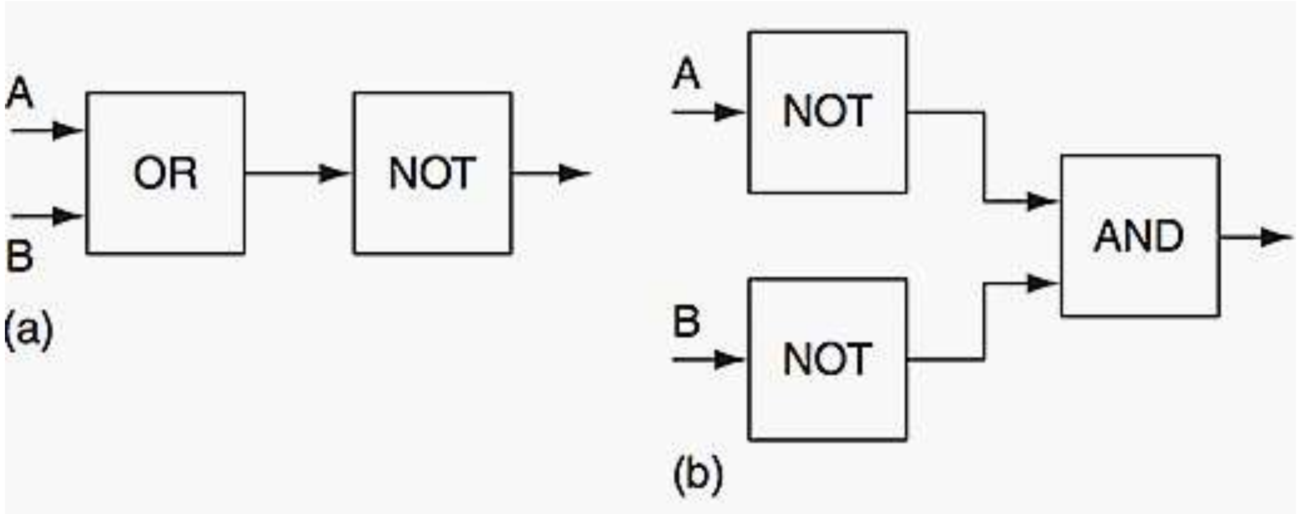
a	b	a OR b	NOT (a OR b)
0	0	0	1
1	0	1	0
0	1	1	0
1	1	1	0

Logic function

NOR

Combination of a **OR** and an **NOT** operator

- The output is 1 when neither inputs is 1
- An alternative is to put a **NOT on each input,** then **follow by an AND**



NOT (a **OR** b)

=

(**NOT** a) **AND** (**NOT** b)

Logic function

NOR

Combination of a **OR** and an **NOT** operator

- An alternative is to put a **NOT on each input,** then **follow by an AND**

Ladder Diagram

a	b	a NOR b
0	0	1
1	0	0
0	1	0
1	1	0

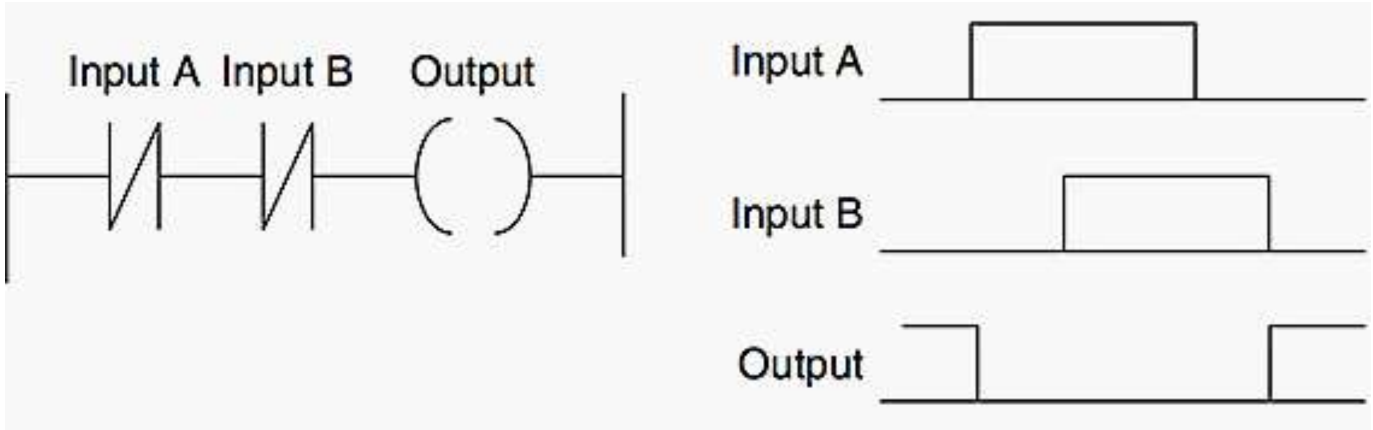
Logic function

NOR

Combination of a **OR** and an **NOT** operator

- An alternative is to put a **NOT on each input,** then **follow by an AND**

Ladder Diagram



a	b	a NOR b
0	0	1
1	0	0
0	1	0
1	1	0

Logic function

XOR, exclusive OR

The output is 1 when either of the inputs is 1 but not when both are 1

- ((**NOT** A) **AND** B) **OR** (A **AND** (**NOT** B))

Logic function

XOR, exclusive OR

The output is 1 when either of the inputs is 1 but not when both are 1

- ((**NOT** A) **AND** B) **OR** (A **AND** (**NOT** B))

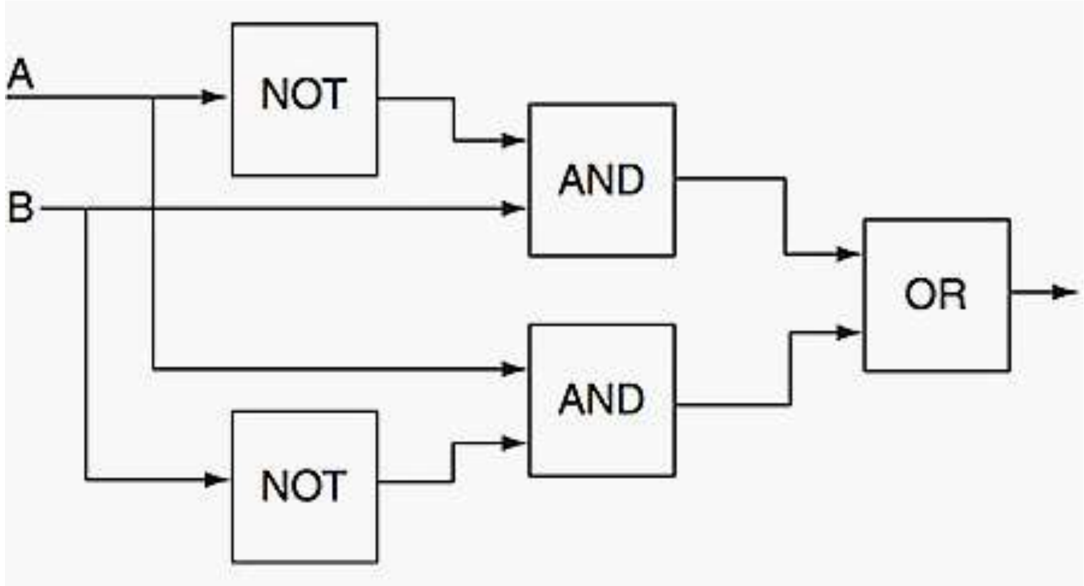
a	b	NOT a	NOT b	(NOT A) AND B	A AND (NOT B)	a XOR b
0	0	1	1	0	0	0
1	0	0	1	0	1	1
0	1	1	0	1	0	1
1	1	0	0	0	0	0

Logic function

XOR, exclusive OR

The output is 1 when either of the inputs is 1 but not when both are 1

- ((**NOT** A) **AND** B) **OR** (A **AND** (**NOT** B))



((**NOT** A) **AND** B) **OR** (A **AND** (**NOT** B))

=

(a **OR** (**NOT** b)) **AND** ((**NOT** a) **OR** B)

Logic function

XOR, exclusive OR

The output is 1 when either of the inputs is 1 but not when both are 1

- ((**NOT** A) **AND** B) **OR** (A **AND** (**NOT** B))

Ladder Diagram

a	b	a XOR b
0	0	0
1	0	1
0	1	1
1	1	0

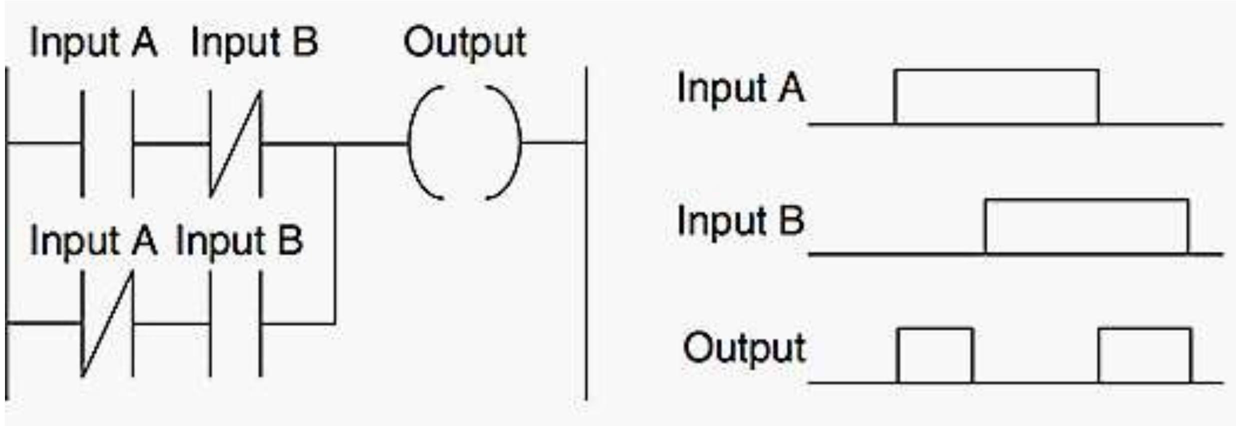
Logic function

XOR, exclusive OR

The output is 1 when either of the inputs is 1 but not when both are 1

- ((**NOT** A) **AND** B) **OR** (A **AND** (**NOT** B))

Ladder Diagram

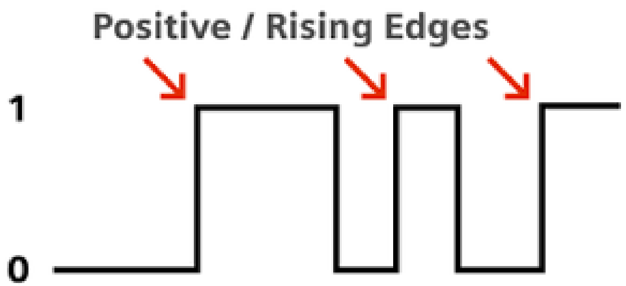


a	b	a XOR b
0	0	0
1	0	1
0	1	1
1	1	0

Logic function

Rising edge detection

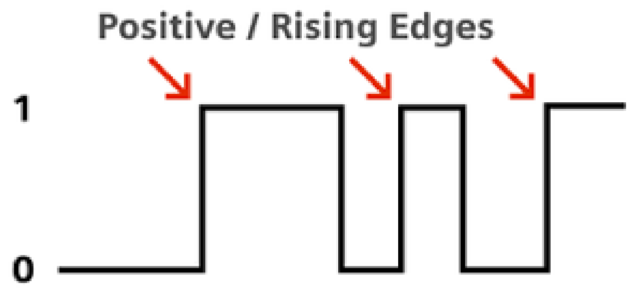
- **Input** → the signal **a**
- **Output** → should be 1 when a has a rising edge (goes from 0 to 1)



Logic function

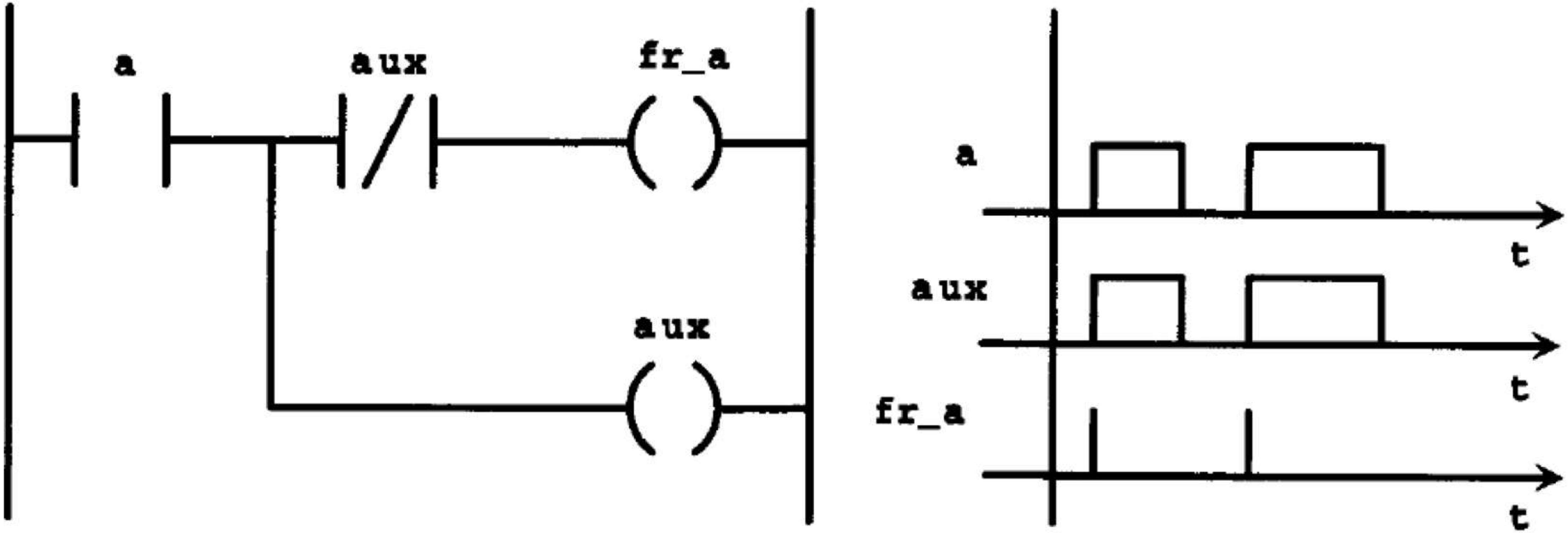
Rising edge detection

```
FUNCTION rising_edge : BOOL
    INPUT_VAR
        a : BOOL;
    END_VAR
    VAR
        aux : BOOL;
    END_VAR
    rising_edge := NOT (aux) AND a;
    aux := a;
END_FUNCTION
```



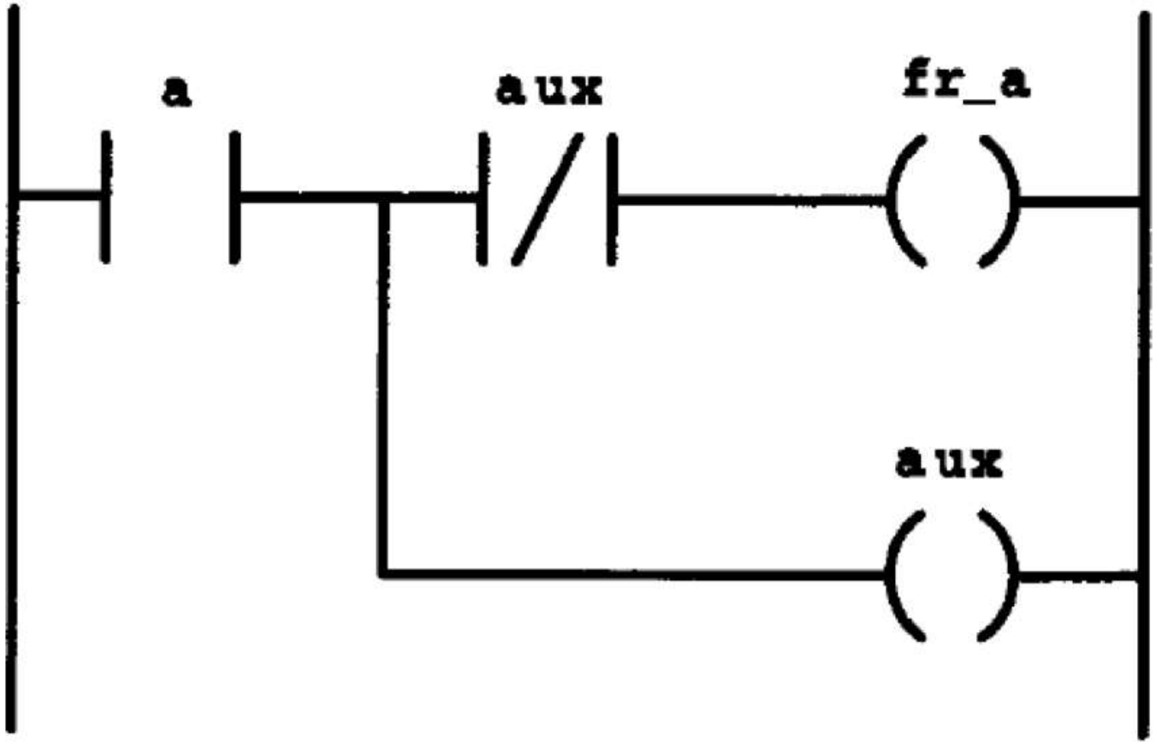
Logic function

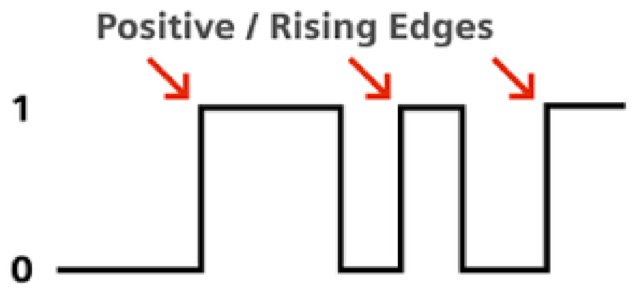
Rising edge detection



Logic function

Rising edge detection





Equivalent to use a positive transition sensing contact

Ladder logic exercises

Exercise 1)

A start and a stop button is used for starting and stopping a motor. But make sure that the buttons can only start and stop the motor on a positive or rising edge.

Ladder logic exercises

Exercise 1)

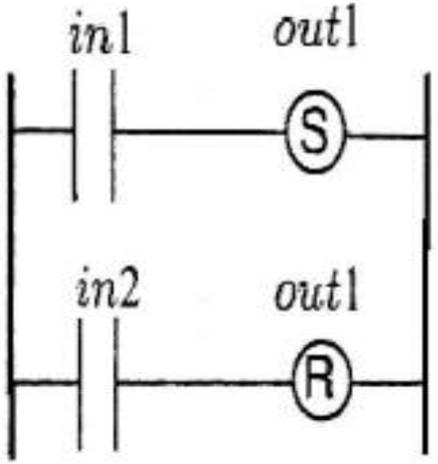
A start and a stop button is used for starting and stopping a motor. But make sure that the buttons can only start and stop the motor on a positive or rising edge.

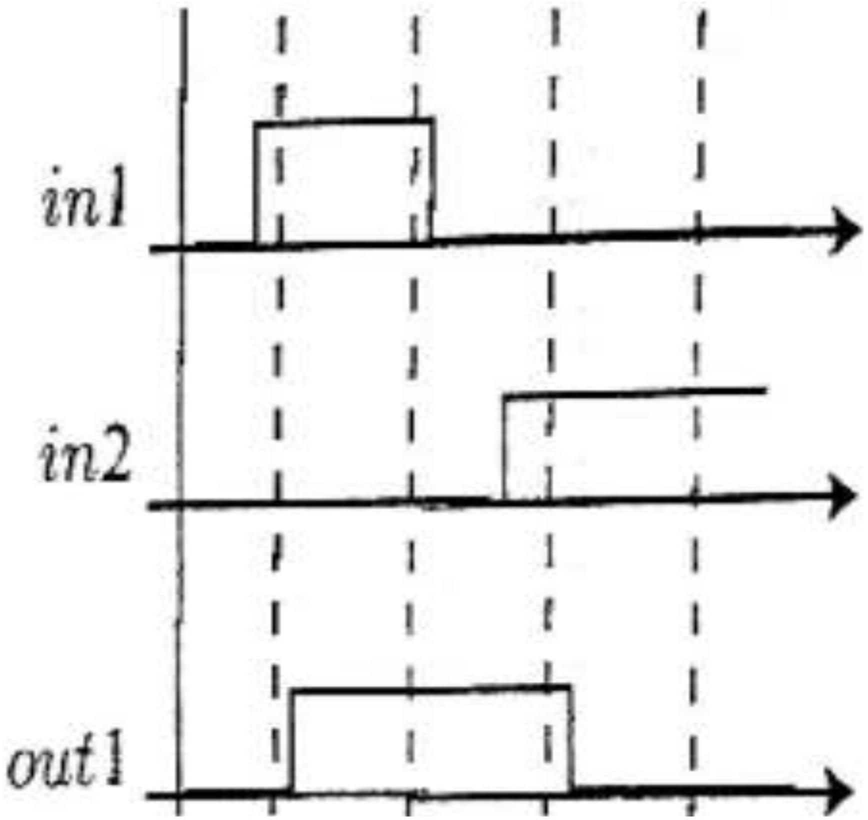
You may want to use

- (S) **SET coil**
  - if fed the associated bit is set to 1 and retains the value 1
- (R) **RESET coil**
  - if fed the associated bit is set to 0 and retains the value 0

LD coils

SET and RESET coils





After a SET coil, there must be a RESET coil associated with the same variable

Ladder logic exercises

Exercise 2)

A start and two stop buttons turn on and off a heating element and a fan. When the heating element turns off, a second fan has to start. The second fan will turn off as soon as the heating element and the first fan turn on.

Ladder logic exercises

Exercise 2)

A start and two stop buttons turn on and off a heating element and a fan. When the heating element turns off, a second fan has to start. The second fan will turn off as soon as the heating element and the first fan turn on.

You may want to use

- (/) **negated coil**
  – if fed the associated bit is set to 0, otherwise is 1

Ladder logic exercises

Exercise 3)

Start / stop of 3 motors, but only 2 motors can run simultaneously. For example, if motor 2 and motor 3 is running, you cannot start motor 1.

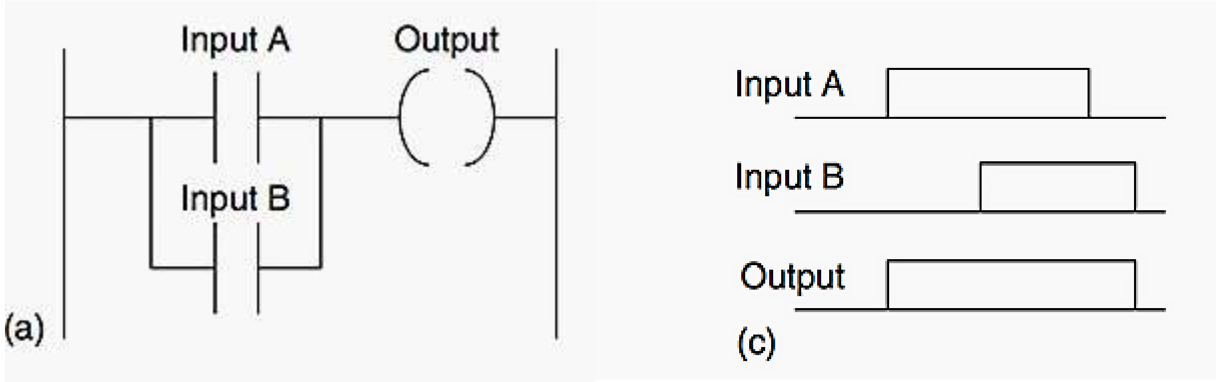
Ladder logic exercises

Exercise 3)

Start / stop of 3 motors, but only 2 motors can run simultaneously. For example, if motor 2 and motor 3 is running, you cannot start motor 1.

You may want to use

- A parallel of coils (**OR**)



a	b	a OR b
0	0	0
1	0	1
0	1	1
1	1	1

Ladder logic exercises

Operators precedence:
NOT
AND
OR
XOR
NOR

Exercise 4)

Implement the following logic for a valve and a motor output:

VALVE

START_V **AND NOT** SENSOR1 **OR** VALVE **AND NOT** STOP_V **AND NOT** MOTOR

MOTOR

START1_M **OR** MOTOR **AND** START2_M **OR NOT** VALVE **AND NOT** STOP_M

Ladder logic exercises

Operators precedence:
NOT
AND
OR
XOR
NOR

Exercise 4)

Implement the following logic for a valve and a motor output:

VALVE

(START_V **AND** (**NOT** SENSOR1)) **OR** (VALVE **AND** (**NOT** STOP_V) **AND** (**NOT** MOTOR))

MOTOR

START1_M **OR** (MOTOR **AND** START2_M) **OR** ((**NOT** VALVE) **AND** (**NOT** STOP_M))

FACTORY IO

Exercise 5)

From A to B

- Transport the box until it reaches the sensor

FACTORY IO

Exercise 5)

From A to B

- Transport the box until it reaches the sensor

Retroreflective Sensor and Reflector

When an object is intercepted, the light is interrupted: the sensor goes from $1 \rightarrow 0$

Tag	I/O	Type	Description
Sensor	Input	BOOL	Light beam interrupted

FACTORY IO

Exercise 5)

From A to B

- Transport the box until it reaches the sensor
- Add a panel with START and STOP buttons

FACTORY IO

Exercise 5)

From A to B

- Transport the box until it reaches the sensor
- Add a panel with START and STOP buttons

NB: The stop button is normally closed

Tag	I/O	Type	Description
Start/Stop Button	Input (Sensor)	BOOL	Pressed
Start/Stop Button Light	Output (Actuator)	BOOL	Led on/off

FACTORY IO

Exercise 6)

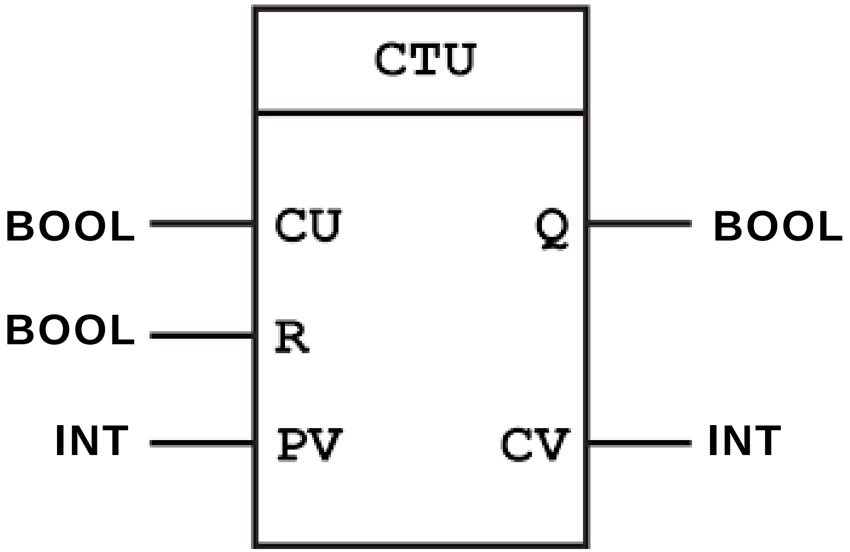
Queue of Items

- Load and unload the boxes
- Count the number of unloaded boxes

Functions block instances

CTU, Counter UP

```
TYPE CTU :
    STRUCT
        (* inputs *)
        CU  : BOOL;     (* count up *)
        R   : BOOL;     (* reset *)
        PV  : INT;      (* preset value *)
        (* outputs *)
        Q   : BOOL;     (* output up *)
        CV  : INT;      (* current value *)
    END_STRUCT;
END_TYPE
```



FACTORY IO

Exercise 6)

Queue of Items

- Load and unload the boxes
- Count the number of unloaded boxes

Tag	I/O	Type	Description
Reset Button	Input (Sensor)	BOOL	Pressed
Reset Button Light	Output (Actuator)	BOOL	Led on/off
Digital Display	Output	INT	Display numerical values

MASTER MEIM 2021-2022

Thank you