Industrial Automation
MKT4152

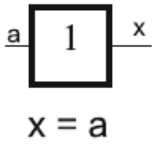
Introduction to
Programmable Logic Controllers
(PLC)

Boolean Arithmetics
Programming Logic

Logic Functions and Boolean Algebra

Identity

$$X = a$$



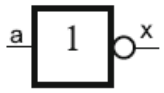
x = a

a	x
0	0
1	1

Negation (Not)

$$X = \bar{a}$$

Both inputs and outputs can be negated!!!!!

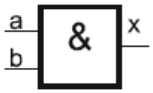


$x = \overline{a}$

a	x
0	1
1	0

Conjunction (AND)

$$X = a \wedge b$$



$x = a \wedge b$

a	b	x
0	0	0
1	0	0
0	1	0
1	1	1

Function chart

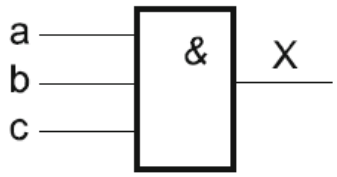
Logic Functions and Boolean Algebra

Multiple input Variables

Number of possible combinations : $K = 2^n$

Function Equation

$$X = a \wedge b \wedge c$$

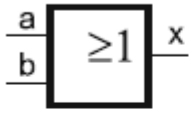


a	b	c	Y
0	0	0	0
1	0	0	0
0	1	0	0
1	1	0	0
0	0	1	0
1	0	1	0
0	1	1	0
1	1	1	1

Disjunction (OR)

$$X = a \vee b$$

Function equation



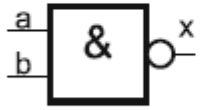
$x = a \vee b$

a	b	x
0	0	0
1	0	1
0	1	1
1	1	1

Function chart for three inputs

a	b	c	Y
0	0	0	0
1	0	0	1
0	1	0	1
1	1	0	1
0	0	1	1
1	0	1	1
0	1	1	1
1	1	1	1

NAND Function (Negated AND)



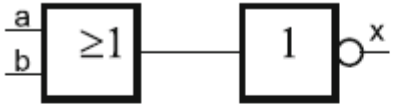
a	b	x
0	0	1
1	0	1
0	1	1
1	1	0

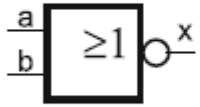
Function equation

$$X = \overline{a \wedge b}$$

$$x = \overline{a \wedge b}$$

NOR Function (Negated OR)





a	b	x
0	0	1
1	0	0
0	1	0
1	1	0

Function equation

$$X = \overline{a \vee b}$$

$$x = \overline{a \vee b}$$

Example: Agitator (Mixer)

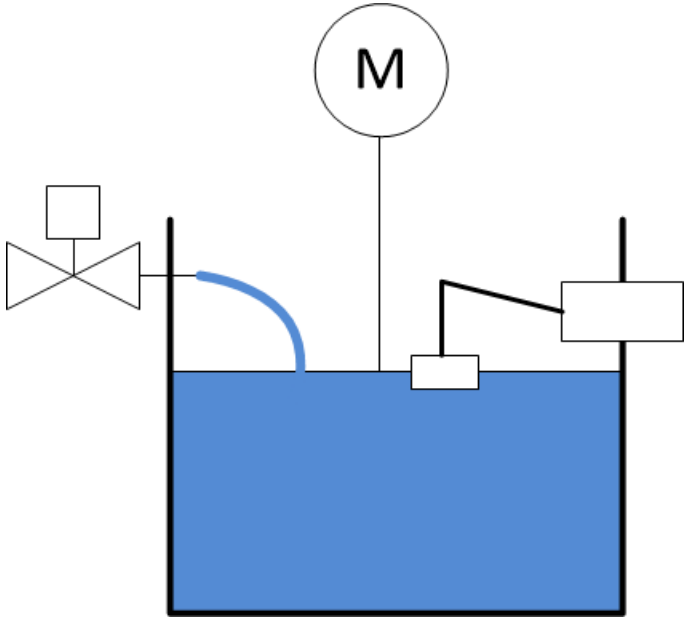
An agitator should only start stirring, when:

- fluid level has been reached „F“ and valve „V“ is closed
- fluid level has not been reached and filling valve is open

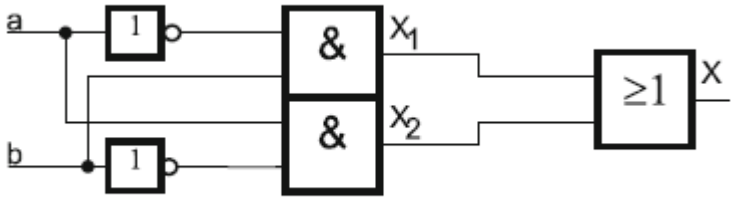
In all other cases, the agitator should remain off

Function chart

V	F	A



Antivalence (Exclusive OR, XOR)



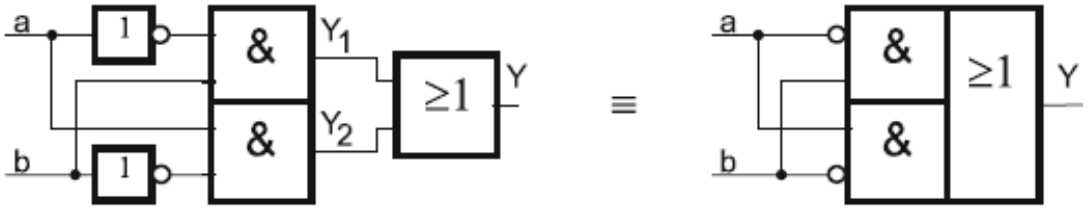
Function equation

$$X = (\overline{a} \wedge b) \vee (a \wedge \overline{b}) = a \underline{\vee} b$$

a	b	x
0	0	0
1	0	1
0	1	1
1	1	0

Procedure

- Draw Function chart
- Seperate minterms (x=1)
- Bundle subfunctions with OR



Logic Functions and Boolean Algebra

Associativity of $\vee$:	$x \vee (y \vee z) = (x \vee y) \vee z$
Associativity of $\wedge$:	$x \wedge (y \wedge z) = (x \wedge y) \wedge z$
Commutativity of $\vee$:	$x \vee y = y \vee x$
Commutativity of $\wedge$:	$x \wedge y = y \wedge x$
Distributivity of $\wedge$ over $\vee$:	$x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$
Identity for $\vee$:	$x \vee 0 = x$
Identity for $\wedge$:	$x \wedge 1 = x$
Annihilator for $\wedge$:	$x \wedge 0 = 0$

Boolean algebra satisfies many of the same laws as ordinary algebra when one matches up “and” with addition and “or” with multiplication.

These laws are common to both kinds of algebra.

Logic Functions and Boolean Algebra

Associativity of $\vee$:	$x \vee (y \vee z) = (x \vee y) \vee z$
Associativity of $\wedge$:	$x \wedge (y \wedge z) = (x \wedge y) \wedge z$
Commutativity of $\vee$:	$x \vee y = y \vee x$
Commutativity of $\wedge$:	$x \wedge y = y \wedge x$
Distributivity of $\wedge$ over $\vee$:	$x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$
Identity for $\vee$:	$x \vee 0 = x$
Identity for $\wedge$:	$x \wedge 1 = x$
Annihilator for $\wedge$:	$x \wedge 0 = 0$

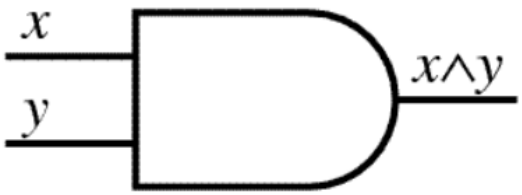
Boolean algebra satisfies many of the same laws as ordinary algebra when one matches up “and” with addition and “or” with multiplication.

These laws are common to both kinds of algebra.

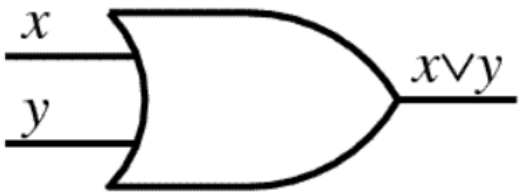
Annihilator for $\vee$:	$x \vee 1 = 1$
Idempotence of $\vee$:	$x \vee x = x$
Idempotence of $\wedge$:	$x \wedge x = x$
Absorption 1:	$x \wedge (x \vee y) = x$
Absorption 2:	$x \vee (x \wedge y) = x$
Distributivity of $\vee$ over $\wedge$:	$x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$

These laws hold in Boolean Algebra, but not in ordinary algebra

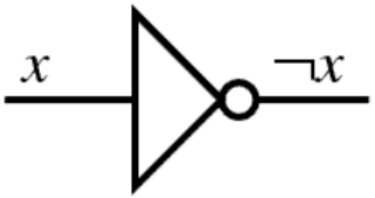
Logic Elements



AND



OR



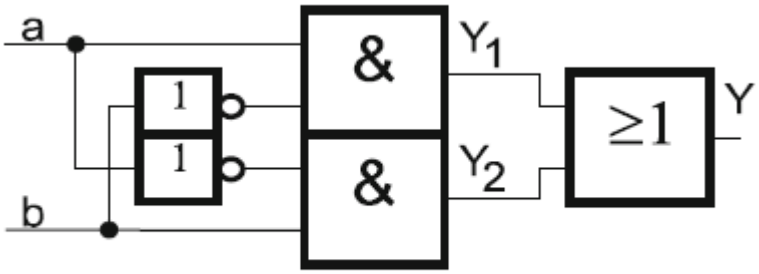
NOT

Ladder Logic

Function Chart

a	b	Y	Y_1	Y_2
0	0			
1	0			
0	1			
1	1			

Logic Plan



Function Chart

a	b	Y	Y_1	Y_2
0	0			
1	0			
0	1			
1	1			

Logic Plan

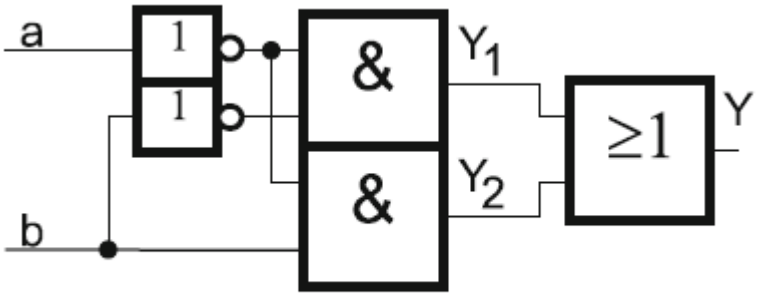
$$Y = Y_1 \vee Y_2 = (\overline{a} \wedge \overline{b}) \vee (a \wedge b)$$

Ladder Logic

Function Chart

a	b	Y	Y_1	Y_2
0	0			
1	0			
0	1			
1	1			

Logic Plan

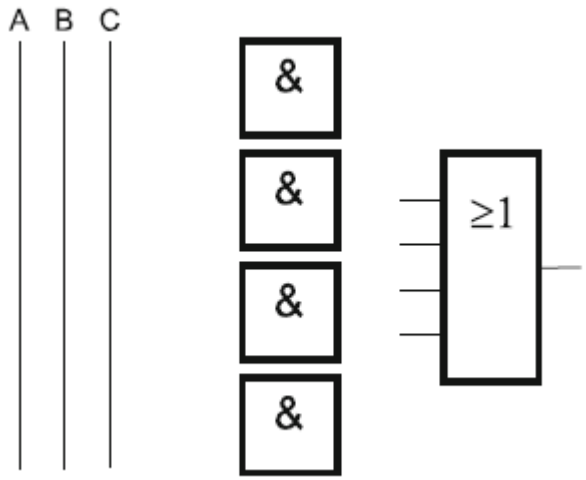


Stairway Lighting – 3 Story building

Design a cross circuit with 3 switches. Switching ONE should change the state of light.

Procedure:
- Draw function chart
- Start with the first line. All zero means, the light should be off
- Taking the first line as basis, find lines with single change of state

A	B	C	Y	Y1	Y2	Y3	Y4
0	0	0	0				
1	0	0					
0	1	0					
1	1	0					
0	0	1					
1	0	1					
0	1	1					
1	1	1					

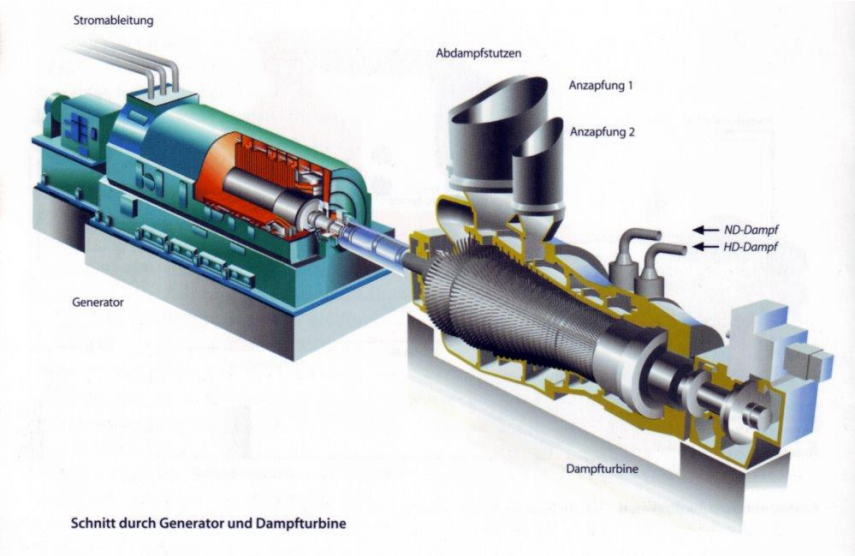


Example: 2/3 circuit – Redundant Process Monitoring

The temperature of a steam turbine system is monitored. Three separate temperature sensors X1 X2 X3 are installed, that trigger when a threshold is exceeded.

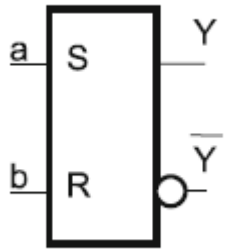
Objective 1: a single sensor triggering should set an alarm „A“. Two or more sensors signaling exceeded thresholds should set the error state „E“.

Objective2:
Once set, the system should remain in its state until reset with button „R“.



Schnitt durch Generator und Dampfturbine

RS-FLIP FLOP

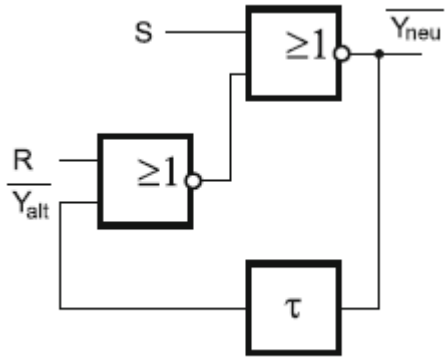


a	b	Y
0	0	Y (no change)
1	0	1 (set)
0	1	0 (reset)
1	1	?? forbidden

It is not allowed to switch both inputs to „1“ at the same instant. In that case, the output can be undetermined!!!!!!

S	R	Y_{alt}	$\overline{R} \wedge Y_{alt}$	Y_{neu}
0	0	0	0	0
1	0	0	0	1
0	1	0	0	0
1	1	0	0	-
0	0	1	1	1
1	0	1	1	1
0	1	1	0	0
1	1	1	0	-

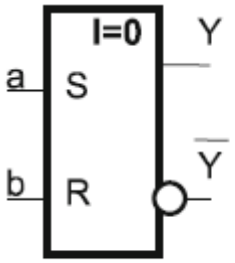
$$Y_{neu} = S \vee (\overline{R} \wedge Y_{alt})$$

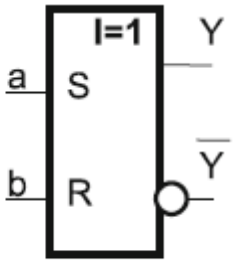


RS FLIP FLOP with delay

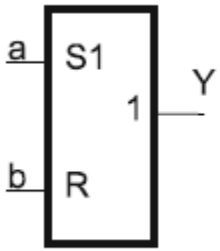
Memory Elements RS FLIP FLOP

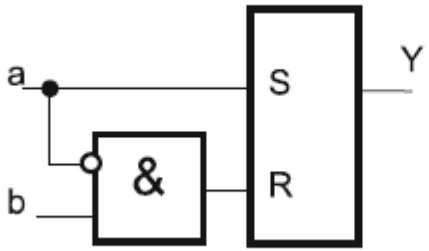
Initial conditions
-> state of outputs upon start of program





Case S=R=1 :
In acutal use, it can‘t be guarantied that both inputs won‘t be set simulataneously.
For that reason, an additional AND is integrated:

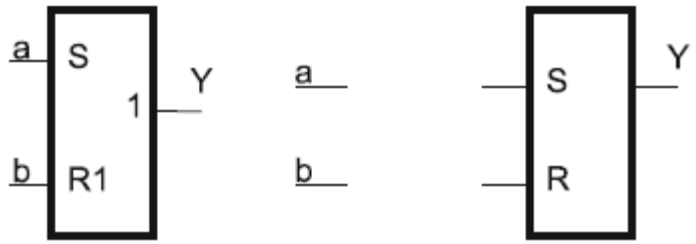




a	b	Y	
0	0	Y	
1	0	1	Set
0	1	0	Reset
1	1	1	Priority

Here, the undefined state becomes „1“. This is called DOMINANT SET
Input S is marked „S1“

The second case is DOMINANT RESET.
Here, the output is defined „0“ for both inputs equal „1“

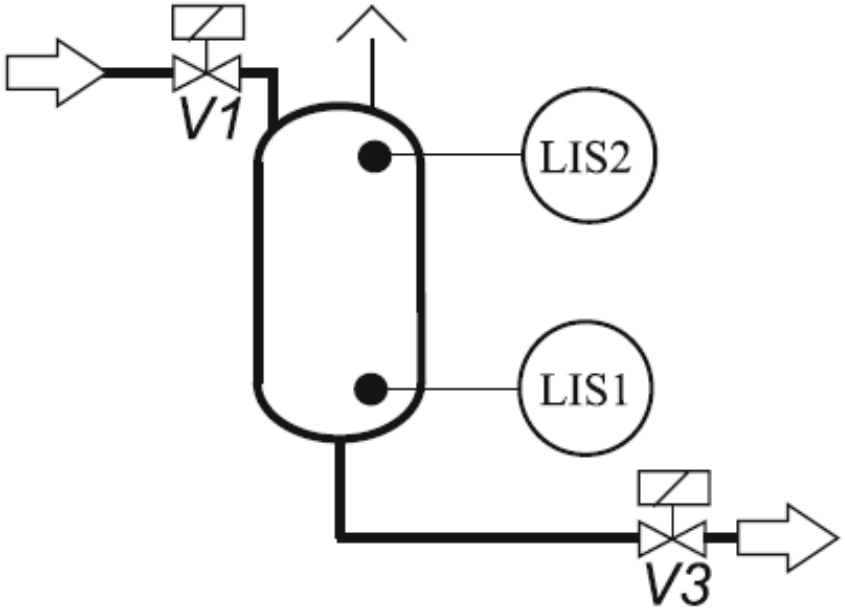


a	b	Y	
0	0	Y	
1	0	1	Set
0	1	0	Reset
1	**1**	**0**	Priority

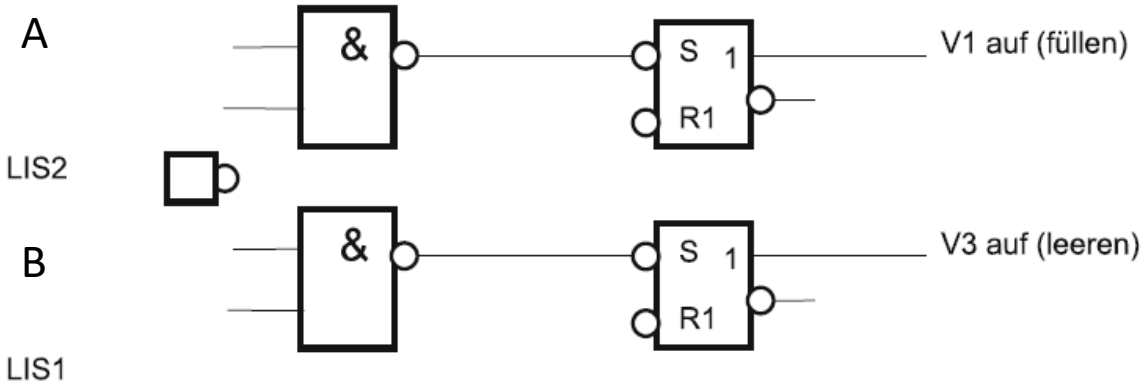
How would you realize this?

Example: Filling of a tank

The tank is filled with valve V1, and emptied with Valve V3.
Two sensors indicate tank full (LIS2) or tank empty (LIS1).
Objective:
Control System with two push-buttons, A for filling, B for emptying.
While emptying, Valve V1 should be closed.
While filling, valve V3 should remain closed.



Example: Filling of a tank

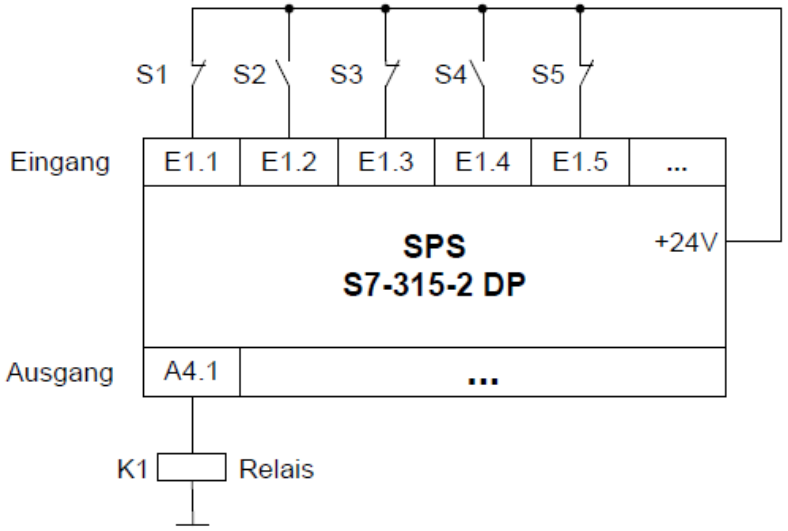


Example: PLC Programming

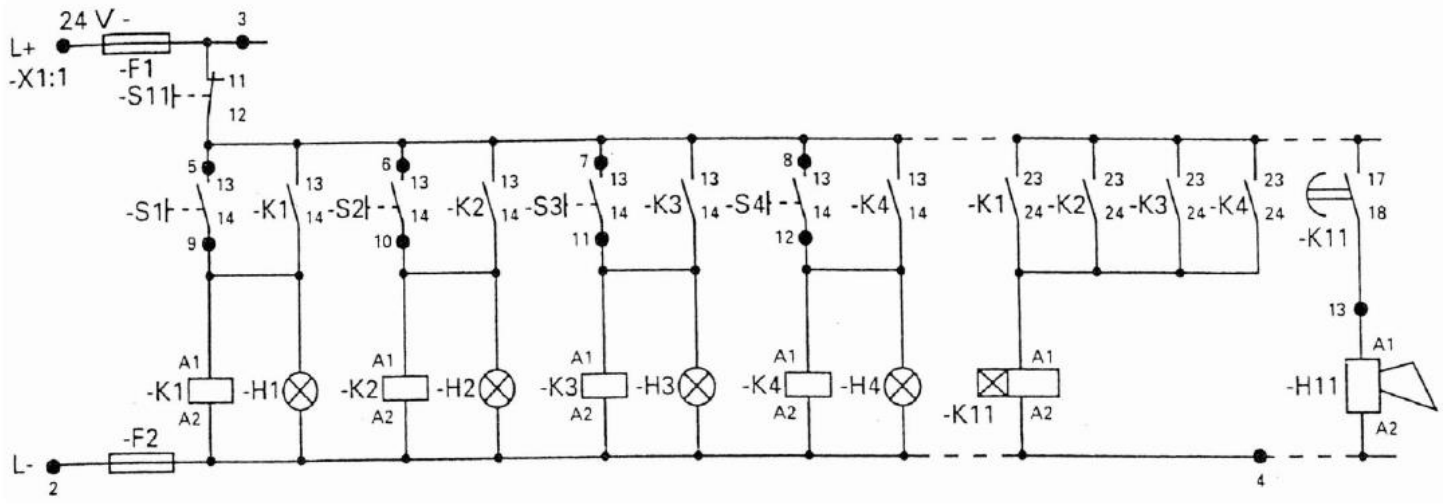
Program a PLC for the following function:

$$K1 = S1 \wedge ((\overline{S2} \wedge \overline{S3} \vee S4) \wedge \overline{S5} \vee S3)$$

- Create function table
- Program in Function block symbols
- Program in Ladder Logic



Example: Call System in Hospital



The call system for hospital nursing staff in above figure is outdated and should be replaced by a modern PLC system.
Each button S1-S4 switches on the corresponding indicator light H1-H4, which stay on after release. They are switched of using the reset button S11.
If a call is ignored for 3 minutes, additionally the acoustic alarm H11 is triggered.

Example: Light switch with trigger (Bistable input)