Project 1

Logic control for a door sluice with the language SIMATIC STEP 7

List of contents

1 Main objective and learning objectives

Main objective

Mode of operation and programming of a Programmable Logic Controller (PLC).

Learning objectives:

- Characteristics of a PLC (synchronous, asynchronous)
- Analysis of the process schematic
- Statement of the interlocking functions and the safety requirements
- Creating of a control system function chart in conformity with DIN 40719, part 6
- Selection of the necessary hardware units
- Programming
- Simulation
- Start-up procedure, testing

2 Introduction

2.1 Structure of an automation system

The automation system SIMATIC S7-300 used in the project is a modern programmable logic controller. It consists of the following units:

- Power supply unit PS 3075A
  provides the internal supply voltage
- Central processing unit CPU 315-20P
  saves and processes the user program created in STEP 7
- Input module
  for connecting 16 digital sensors
- Output module
  for connecting up to 16 actuators, e.g. contactor coils, valve coils or signal lamps.

In contrast to the wired-program controllers, where the operation of the control is determined by direct connections between the single sensors and actuators, the connection of the terminals (input and output) does not depend on the program to realize with a programmable logic controller.
With the same configuration (hardware), it is possible to solve various control tasks without changing the wiring, only by changing the saved command logic (software).
Easy and fast changing of the program provides special advantages the PLC in different situations:

- during the set up
- frequently changing program processing
- machines with changeable assembling

The basic item of each PLC is the central processing unit (CPU) with the processor, the address counter, the program memory and in most cases the power supply unit, too.
The central processor contains the control unit with a clock generator, the arithmetic-logic unit and the memories for retentive flags, output flags and the process input/output image. This is a memory area representing the signal state of in- and outputs, of flags and the states of timers and counters.

The control unit controls the succession of processing the commands of a user program. It decodes these commands, checks the state of the flags and outputs the digital signals for the execution of the commands. The pulses for reading and for the execution of the program commands are produced by the internal clock generator with clock times between 1 and 5µs, depending on the type used.
The arithmetic-logic unit executes the actual logic operation of the operands according to the logic and arithmetic conditions, but also other operations like comparing, converting or shifting.
Figure 1 displays the schematic structure of a PLC.

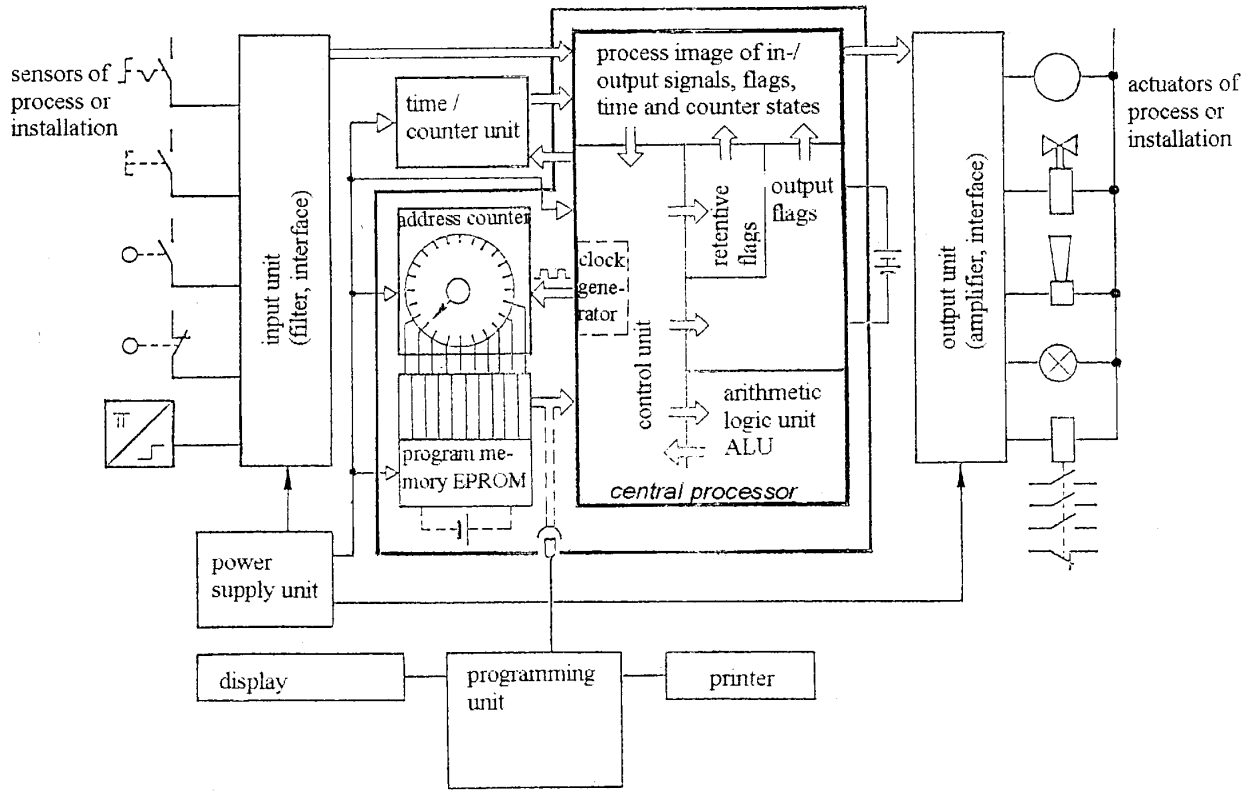


Figure 1: Programmable logic controller

2.2 Fundamental principles of the basic programming software STEP 7

STEP 7 is a software for programming and configuring SIMATIC S7-300/400 – Automation Systems, running under Windows 95/98 or Windows NT.
The basic software gives support in all phases during the programming of automation solutions, as e.g.:

- Set up and management of projects
- Configuration and parameter assignment of hardware
- Programming for S7 destination systems
- Loading of programs into destination systems
- With optional software S7-PLCSIM:
  Execution of the program simulation
- Testing the automation system
- Diagnosis of system disturbances

As the STEP 7 user environment is arranged modernly, it is easy to learn and you can get into it very quickly.

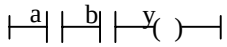
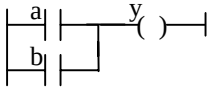
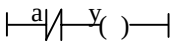
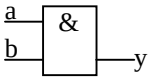
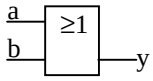
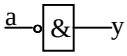
Methods of representation (LAD, STL, CSF)

Following programming languages are integrated in the basic software:

- LAD (ladder diagram) is a graphical programming language similar to a circuit diagram. It allows following the signal flow in the network easily.
- STL (statement list) is a machine-oriented language. The single steps correspond largely with the steps, the CPU processes.
- CSF (control system flowchart) is a graphical language which uses the logical elements of the boolean algebra for representing the operation logic.

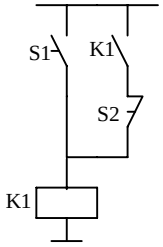
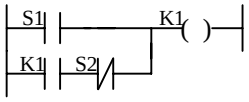
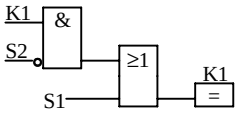
Logic functions

The logic operations work with two states: Number "1" for logical "yes" and Number "0" for logical "no". These numbers are called "binary digits", short "Bits", and form the binary numerative system.
In the programming languages above the fundamental combinations of the binary signal states are represented as follows:

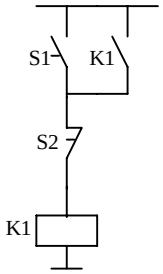
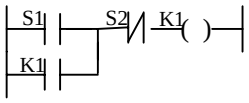
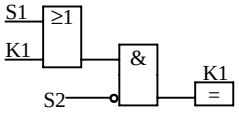
	AND – conjunction	OR – disjunction	NOT – negation
LAD			
STL	U a U b = y	O a O b = y	UN a = y
CSF			
Equation	$y = a \wedge b$	$y = a \vee b$	$y = \overline{a}$
Truth table	a b y 0 0 0 0 1 0 1 0 0 1 1 1	a b y 0 0 0 0 1 1 1 0 1 1 1 1	a b 0 1 1 0

Examples of logic functions

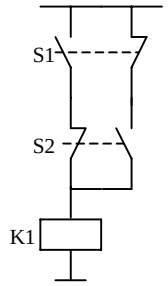
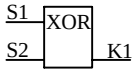
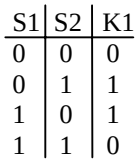
AND-before-OR (symbolic representation)

Circuit diagram	LAD	STL	CSF
		U S1 0 U K1 UN S2 = K1	

OR-before-AND (symbolic representation)

Circuit diagram	LAD	STL	CSF
		U(O S1 O K1) UN S2 = K1	

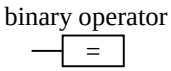
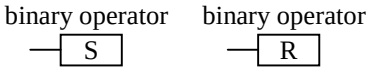
Exclusive OR (symbolic representation)

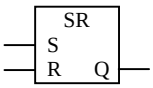
Circuit diagram	LAD	STL	CSF
		X S1 X S2 = K1	 

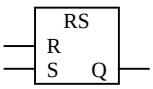
S1	S2	K1
0	0	0
0	1	1
1	0	1
1	1	0

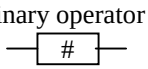
Memory functions

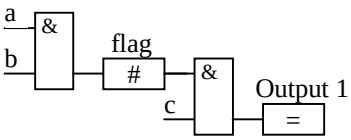
The following memory functions are used in combination with binary operations in order to exercise an influence on binary operands:

- the *Assignment* box for dynamic control: 
- the *Set* and *Reset* box as single memory function: 
- the *Set* and *Reset* box as complete memory function:

  SR box: 

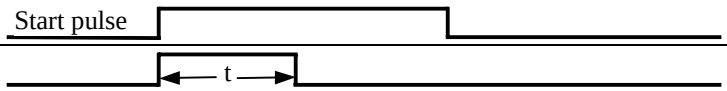
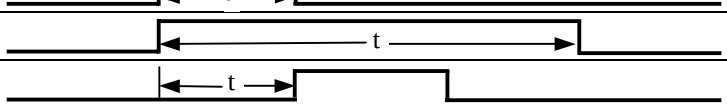
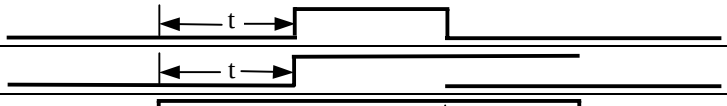
  RS box: 

- the *Connector* box as temporary memory: 

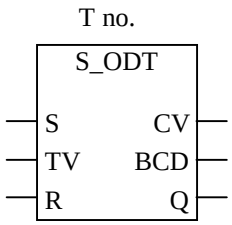
  e.g. 

Time functions

Time functions show the following performance:

Time function as		
Pulse generator	S_IMPULS	
Expanded pulse	S_VIMP	
ON delay	S_EVRZ	
Latching ON delay	S_SEVRZ	
OFF delay	S_AVRZ	

Time functions in box representation (in the example: ON delay)

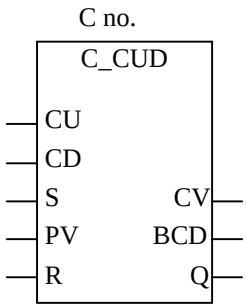


Parameters	Data Type	Description
T no.	Timer	Timer identification number, the range depends on the CPU
S	BOOL	Start input
TV	S5T#	Preset time value, e.g. S5T# 10s
R	BOOL	Reset input
CV	WORD	Time remaining (value in integer format)
BCD	WORD	Time remaining (value in BCD format)
Q	BOOL	Status of the timer

Counter functions

With the counter functions, counter tasks can be executed directly by the CPU. The counter functions can both count up and down: The counter range extends over three decades (000 to 999). A counter function can be programmed completely as a box, but also with a single programm block.

Counter functions in box representation (in the example: ON delay)



Parameters	Data Type	Description
C no.	Counter	Counter identification number. The range depends on the CPU.
CU	BOOL	Input Up Counter
CD	BOOL	Input Down Counter
S	BOOL	Input for presetting counter
PV	WORD	Counter value in the range between 0-999 or entered as C#<value> in BCD format
R	BOOL	Reset input
CV	WORD	Current count value (hexadecimal number)
BCD	WORD	Current count value (BCD format)
Q	BOOL	Status of the counter

Blocks of STEP 7 (OB1, FC, FB)

Organization block OB1

The organization block OB1 forms an interface between the operating system of the CPU and the user program. The operation system of the CPU processes the OB1 cyclically. In the OB1 functions (FC) or function blocks (FB) can be called, but you can also create programs which are executed immediately.

Function FC

A FC function is a code block without a "memory", however it can pass over parameters. This block is especially suitable for programming functions frequently repeating themselves. Example: control with binary logic.

Function block FB
A function block is a block with an "assigned memory". It refers to a data block as memory (instance data block). A FB contains a program which is executed whenever the FB is called by another code block, e.g. OB1. FB makes programming easier when there are frequently returning, complicated functions.

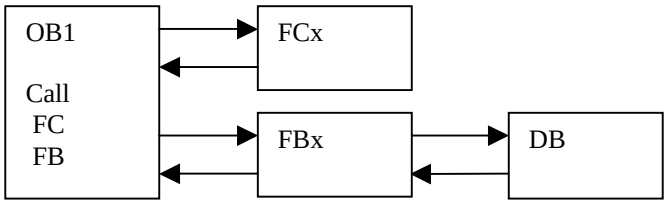


Figure 2: Block structure

3 Hard- and Software

The execution of the logic control requires following hard- and software components:

a) Programmable controller S7-300 with components:
   - mounting channel as a mounting rack
   - Power supply module PS 3075A (6ES7-307-1EA00-0AA0)
   - Central processing unit CPU 315-2DP (6ES7-315-2AF03-0AB0)
   - Digital input module 16xDC 24V (6ES7-321-1BH01-0AA0)
   - Digital output module 16xDC 24V, 0.5A (6ES7-322-1BH01-0AA0)

b) MPI interface cable for the connection CPU / PC
c) PC with a minimum configuration: Pentium processor, Windows 95/98/NT, 32MB RAM, hard disc 3 GB, CD-ROM drive, colour display
d) Software package SIMATIC STEP 7

4 Procedure for setting up and programming a logic control with STEP 7

(= left mouse click)
(= doubleclick)

SIMATIC Manager - Start
Wizard - Cancel
Menu field: File → New →
Name (Project) enter „**OK**“
Menu field: Insert → Station → SIMATIC 300
→ ⊞ - Project-Name
└⊞ SIMATIC 300 → Hardware
↑

<u>Hardware configuration</u>
Toolbar: Catalog
→ ⊞ - SIMATIC 300
→ ⊞ - RACK 300
→ └[] - Rail
→ ⊞ - PS300
└[] - PS3075A (6ES7 307-1EA00-0AA0)
→ ⊞ - CPU 300
→ └[]- CPU 315-2DP
→ └⊞ - 6ES7 315-1AF03-0AB0
→ └[] V1.1
Properties-Profibus interface DP Master (RO/2.1) „**OK**“
Insert (2775:790) **„No“**
Insert (13:4242) **„No“**
Column 3 in the configuration table should be left empty, column 4 clicked
→ ⊞ - SM 300
→ ⊞ - DI 300
└[] - SM 321 DI 16xDC 24V (6ES7 321-1BH01-0AA0)
→ ⊞ - DO 300
└[] - SM322 DO 16xDC24V/0,5A (6ES7 322-1BH01-0AA0)
Configuration save: → Station → Save
Configuration exit: → Station → Exit

In project → ⊞ - SIMATIC 300
→ └⊞ - CPU315-2DP
→ └⊞ - S7 Program
├ Sources
→ └ Block
Menu field: Insert → S7 block → Organization block menu field:
Properties-Function block General-Part 1:
Name: OB1
Created in Language: LAD „**OK**“
Menu field: Insert → S7 Block → Function block
Properties-Function block General-Part 1:
Name: FC1
Created in language: FUP „**OK**“

Programming FC1

In Project:  FC1 Program write, → Toolbar → Save

In project: →  OB1 → Toolbar → Program elements → FC → FC1

→ Toolbar → Save

OB1 and FC1 with Ctrl

Menu field: View → Load (mode On-line)

Toolbar: Simulation

Start the Simulation: S7-PLCSIM

Toolbar: Insert Input, Output, Bit memory, Timer, Counter

→ Input, Bit memory, Timer, put down

CPU 300/400 → RUN

Simulation performing.

For the simulation the time is given in minutes, in the sequence of the chain it can be watched better.

5 Problem description of a door sluice

5.1 Technology schematics

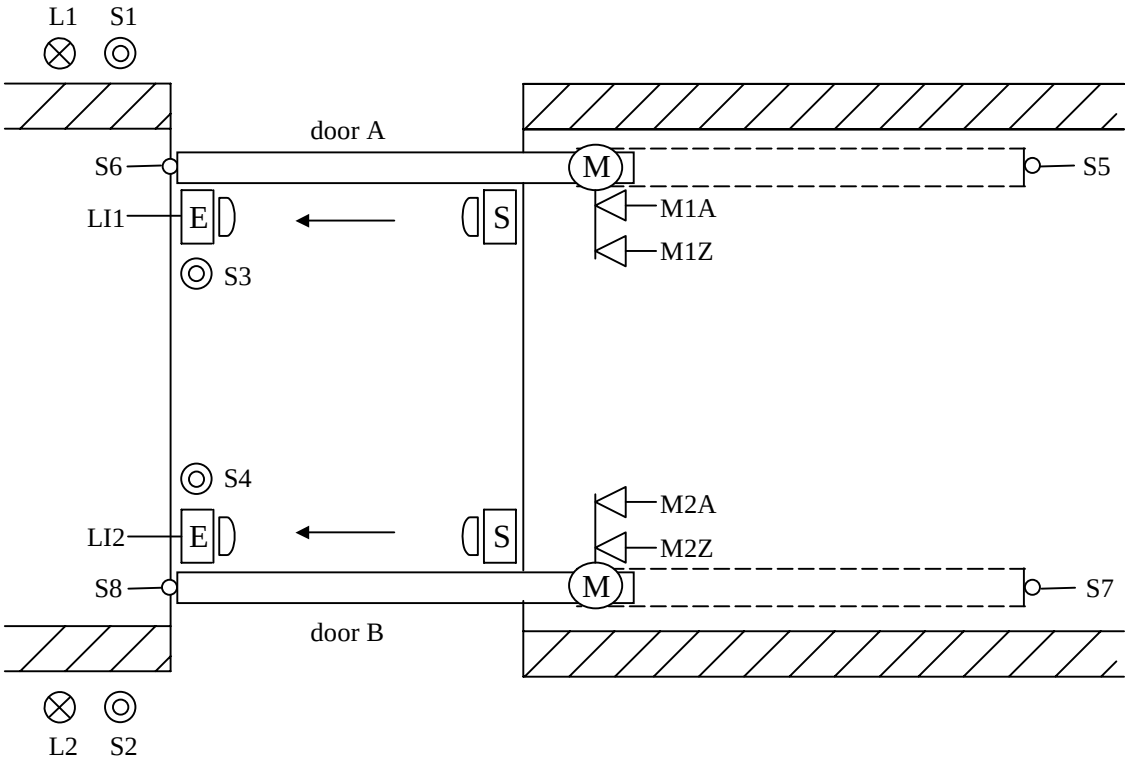


Figure 3: Door sluice

5.2 Problem description

To keep a room free of dust as far as possible, a sluice with two sliding doors „A“ and B“ is installed. The control of the sluice shall be carried out as a logic control with storage properties (control for states – SR flag) in the STEP 7 programming language.

Passing the sluice through door A (or B), the pushbutton S1 (S2) must be actuated. The impulse operates on a memory element (e.g. flag with SR performance), which causes

- a signal lamp L1 (L2) to display that door A (B) is being opened; L1 (L2) goes out when the door is open.
- opening of door A (B) until the door has reached the position switch S5 (S7).

After door A (B) being open for 3 sec., it shall close again reaching the position switch S6 (S8).
Door B (A) only opens automatically after door A (B) has been closed.
Each entry to the sluice is observed by a light barrier LI1 (LI2). As long as the light barrier is interrupted, the opened door may not close. In the sluice two push-buttons S3 (S4) are attached to open the corresponding door in case of emergency.
If during the closing of a door the light barrier is interrupted or the corresponding pushbutton S1, S3 (S2, S4) is actuated, door A (B) opens again immediately. Besides, it must be considered that door A and door B cannot be opened at the same time.

6 Assignment list for a door sluice

Symbol	Inputs	Comment
S1	I 0.1	pushbutton, open door A (outside)
S2	I 0.2	pushbutton, open door B (outside)
S3	I 0.3	pushbutton, open door A (inside)
S4	I 0.4	pushbutton, open door B (inside)
S5	I 0.5	inductive position switch; make contact when door A open
S6	I 0.6	inductive position switch; make contact when door A closed
S7	I 0.7	inductive position switch; make contact when door B open
S8	I 0.8	inductive position switch; make contact when door B closed
LI1	I 1.1	light barrier, door A - break contact
LI2	I 1.2	light barrier, door B - break contact
	Outputs	
L1	Q 4.0	display – door A opening
M1A	Q4.1	motor of door A - open
M1Z	Q 4.2	motor of door A - close
L2	Q 4.3	display – door B opening
M2A	Q 4.4	motor of door B - open
M2Z	Q 4.5	motor of door B - open
	T1	time delay 3 sec. for automatic closing of the doors

7 Tasks during the execution of the project

- Analysis of the technology schematics
- Design of the logic diagram
- Statement of the interlock function and the according safety requirements
- Programming of the control according to the defined procedure
- Executing the simulation on the PC
- Loading the program into the programmable controller, testing mode