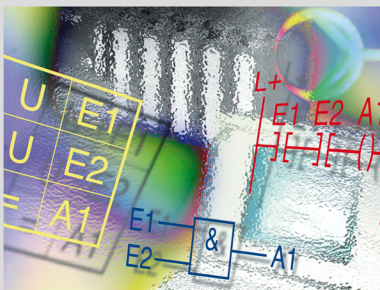


Vogel Fachbuch

Jürgen Kaftan

# PLC-Basic Course with SIMATIC S7



Jürgen Kaftan

# **PLC Basic Course with SIMATIC S7**

Jürgen Kaftan

# PLC Basic Course with SIMATIC S7

Structure and Function of  
Programmable Logic Controllers,  
Programming with the SIMATIC S7

First Edition

## JÜRGEN KAFTAN

- 1967-1971 Apprenticeship as an electrician
- 1971-1973 Skilled tradesman in the trade of electrician
- 1973-1975 Training as a state-certified electrical engineer
- 1975-1977 Employed as a technician
- 1977-1978 School for master craftsmen, graduated as master electrician
- 1979-1992 Nuremberg vocational training centre for hearing and speech impaired persons, Master Trainer
- 1985-1992 Course Leader for PLCs in vocational training (BFH) for hearing and speech impaired persons
- 1985-1992 Course Leader for PLCs at the Chamber of Crafts
- 1992-1995 IKH Elektrogerätebau – System Training Courses (Managing Director)
- Since 1995 Manager of IHK Systemschulungen for Hardware and Software (Chamber of Industry and Commerce hardware and software system training courses) in Weißenburg and Roth (Central Franconia)

Jürgen Kaftan is the author of the following reference books published by Vogel:

PLC Basic Course 1

PLC Basic Course 2

PLC Basic Course with SIMATIC S7

The book includes a Siemens AG demonstration CD-ROM »STEP 7 Professional, Edition 2004 SR4, Trial License« including: STEP 7 V5.3 SP3, S7-GRAPH V5.3 SP2, S7-SCL V5.3 SP1, S7-PLCSIM V5.3 SP1 and can be used for test purposes for 14 days.

The software only runs under Windows 2000 Professional SP4 and above or Windows XP Professional SP1 and above.

For further information on the Internet, visit [www.siemens.de/sce/promotoren](http://www.siemens.de/sce/promotoren).

---

## Further information:

[www.vogel-buchverlag.de](http://www.vogel-buchverlag.de)

---

ISBN 978-3-8343-3201-1

First Edition 2011

All rights reserved including translation. No part of this work may be reproduced, processed using any type of electronic system, duplicated or transmitted in any way or in any form (printing, photocopying, microfilm or using any other process) without the express written consent of the publisher. The exceptions stated expressly in Para. 53, 54 of the German Copyright Act are not affected.

Printed in Germany

Copyright 1998 by Vogel Industrie Medien GmbH & Co. KG,  
Würzburg

Book jacket illustration: Michael M. Kappenstein, Frankfurt

# Foreword

Programmable logic controllers, known for short as PLCs, are an integral part of automation at the lower, middle and upper performance levels. Due to the ever increasing size and complexity of the hardware and software required in automation projects, the industry has had to develop faster, more capable and effective automation systems and to simplify handling for users. All the exercises in this book have been developed and tested using Siemens' SIMATIC S7-300 PLC. The command set of this controller ranges from binary processing to 32-bit floating point arithmetic. The controller is programmed under the Windows XP operating system and it is assumed that users have an appropriate basic knowledge. All the procedures for programming the SIMATIC S7-300 are demonstrated exactly and are easy to understand for users. The topic builds up from "easy to hard" and is ideally suited for use in vocational and technical training colleges, etc. as well as for use on a teach-yourself basis.

I would like to thank the Wükro-Lehrsysteme company in Würzburg as well as everybody who has helped in the completion of this book. I am always grateful to receive feedback from readers and users.

Weißenburg/Heuberg (Central Franconia)

Jürgen Kaftan



# Contents

Foreword.....	5
<b>1 Introduction .....</b>	<b>13</b>
1.1 Number system .....	13
1.2.2 Byte.....	14
1.2.3 Word.....	14
1.2 Terms from computer science.....	14
1.2.1 Bit .....	14
1.2.4 Bit address.....	15
1.2.5 Byte address .....	15
1.2.6 Word address .....	15
<b>2 Arrangement of a PLC.....</b>	<b>17</b>
2.1 Structure of a PLC.....	18
2.2 Structure of an automation unit .....	18
2.3 Hardware requirements.....	18
2.3.1 Hardware structure .....	20
2.4 Software requirements.....	20
2.4.1 STEP 7 programming language .....	20
2.4.2 Objects.....	20
2.4.3 Projects .....	21
2.4.4 Configuring an S7-300 .....	22
2.4.5 Parameterization .....	22
<b>3 Way of Functioning of a PLC .....</b>	<b>23</b>
3.1 Modules of the PLC .....	23
3.1.1 Power supply unit .....	23
3.1.2 Program memory .....	24
3.1.3 Central processing unit (CPU) .....	25
3.1.4 Bus system.....	27
3.1.5 Input and output modules .....	27
<b>4 Program Processing and Programming .....</b>	<b>29</b>
4.1 Linear program processing.....	29
4.2 Structured programming.....	30
4.3 Control instruction.....	31
4.3.1 Operation part .....	32

4.3.2	Examples for digital operations.....	32
4.3.3	Examples of binary operations.....	32
4.3.4	Examples of organizational operations.....	33
4.3.5	Operand part .....	33
4.4	Addressing .....	34
4.4.1	Symbolic Addressing .....	34
4.4.2	Absolute addressing .....	34
4.4.3	Immediate addressing.....	34
4.4.3.1	Direct addressing.....	34
4.4.3.2	Memory-indirect addressing.....	35
4.5	Program representation .....	35
4.5.1	Ladder diagram (LAD) .....	36
4.5.2	Function block diagram FBD (STEP 7 V3.x and above) .....	36
4.5.3	Statement list STL .....	36
4.6	Flags.....	37
4.6.1	Retentive flags.....	37
4.6.2	Non-retentive flags.....	38
<b>5</b>	<b>Logic operations.....</b>	<b>39</b>
5.1	Basic logic operations.....	39
5.1.1	Clearing the CPU .....	40
5.1.2	Creating projects .....	42
5.1.3	Inserting the SIMATIC 300 station.....	43
5.1.4	Configuring and parameterizing.....	43
5.1.5	Arrangement of the power supply .....	45
5.1.6	Arrangement of the CPU 314 .....	46
5.1.7	Arrangement of the input module .....	46
5.1.8	Arrangement of the output module .....	47
5.1.9	Parameterizing the CPU 314 .....	47
5.1.10	Saving the global configuration.....	49
5.1.11	Transferring the configuration to the CPU .....	49
5.2	Logical AND operation user program .....	50
5.2.1	Entering FCs (FC 1) .....	50
5.2.2	S7 block function .....	52
5.2.3	Entering OB 1 .....	55
5.2.4	Downloading .....	58
5.2.5	Testing.....	59
5.2.6	Specifying trigger conditions .....	60
5.2.7	Deactivating the FBD program status.....	63
5.2.8	Testing with STL .....	63
5.2.9	Testing with LAD .....	65
5.2.10	Extending from two to three inputs (3rd. input I0.2).....	67
5.2.10.1	Extending with STL .....	67



5.2.10.2	Extending with LAD .....	69
5.2.10.3	Extending with FBD .....	71
5.2.11	Reducing from 3 inputs to 2 (delete I0.2) .....	73
5.2.11.1	Reducing with STL.....	73
5.2.11.2	Reducing with LAD .....	74
5.2.11.3	Reducing with FBD .....	74
5.3	Logical OR operation user program.....	75
5.3.1	Entering the program using the PC (FBD) .....	75
5.3.2	Create the project.....	76
5.3.3	Copy SIMATIC station 1 into another project.....	77
5.3.4	Change OB 1.....	80
5.3.5	Downloading .....	82
5.3.6	Testing.....	82
<b>6</b>	<b>Program Input.....</b>	<b>85</b>
6.1	AND before OR.....	85
6.2	OR before AND.....	88
6.3	Poll for signal state 0.....	91
6.4	Exclusive OR operation .....	94
6.5	Polling outputs .....	96
6.6	Inserting networks.....	98
6.7	Latch circuit with the PC.....	101
6.8	Practical examples of control with the PC .....	105
6.8.1	Temperature difference.....	105
6.8.2	Drinks machine .....	106
6.8.3	Intercom.....	108
6.8.4	Generator.....	110
6.8.5	Boiler control .....	112
6.8.6	Smelting furnaces .....	113
6.9	Flip-flop .....	116
6.9.1	R-S flip-flop.....	116
6.9.2	Entering the program .....	118
6.9.3	Pump controller .....	122
<b>7</b>	<b>Creating Momentary Impulses (Edge Instructions).....</b>	<b>127</b>
7.1	Momentary impulse with a rising edge (FP) .....	127
7.2	Momentary impulse with a falling edge (FN) .....	128
7.3	Program input.....	128
7.4	Acknowledgement circuit .....	132
<b>8</b>	<b>Timing Functions .....</b>	<b>135</b>
8.1	Timing value specification .....	135
8.2	Release a time (FR) .....	136
8.3	Current value .....	136

8.4	Reset time .....	137
8.5	Selection of times (five different ones) .....	137
8.5.1	Pulse timer (SP) .....	137
8.5.2	Extended pulse timer (SE) .....	139
8.5.3	Switch-on delay timer (SD).....	140
8.5.4	Retentive switch-on delay timer (SS) .....	142
8.5.5	Switch-off delay timer (SF).....	143
8.6	PC program input of timing functions.....	144
8.6.1	Garage lighting.....	147
8.6.2	Filling system .....	149
8.6.3	Compressor system .....	150
<b>9</b>	<b>Clock Generators .....</b>	<b>153</b>
9.1	PC program input with clock generator .....	154
9.1.1	Channel switch.....	156
9.1.2	Paging system.....	157
9.1.3	Air supply .....	159
<b>10</b>	<b>Counters .....</b>	<b>163</b>
10.1	Load and transfer functions .....	163
10.2	Counter functions .....	164
10.2.1	Release a counter (FR) .....	164
10.2.2	Counting forwards .....	164
10.2.3	Counting backwards .....	164
10.2.4	Set counter .....	165
10.2.5	Count value definition.....	165
10.2.6	Reset counter (R) .....	165
10.2.7	Poll count value (L/LC) .....	165
10.2.8	Poll signal state of counter (binary) .....	166
10.3	PC program input cleaning bath.....	167
<b>11</b>	<b>Comparators .....</b>	<b>169</b>
11.1	Comparison functions.....	169
11.1.1	Equal to = .....	169
11.1.2	Not equal <> .....	170
11.1.3	Greater than equal to >= .....	170
11.1.4	Greater than > .....	170
11.1.5	Less than equal to <= .....	170
11.1.6	Less than < .....	171
11.2	PC program input runway light.....	171
11.3	Program input sequence function .....	174
<b>12</b>	<b>Practical Examples with Simulators.....</b>	<b>177</b>
12.1	Seven-segment display .....	177
12.2	Star-delta starting.....	179

12.3	Traffic light controller .....	181
12.4	Conveyor belt controller .....	183
12.5	Reaction vessel .....	186
12.6	Container filling system.....	188
12.7	Automatic tablet filler .....	190
12.8	Door access control system .....	193
12.9	Pump controller .....	195
<b>13</b>	<b>Sequence Control Systems .....</b>	<b>199</b>
13.1	Introduction .....	199
13.2	Components of a sequence control system .....	200
13.3	Type of representation.....	201
13.4	Linear sequence cascade .....	202
13.5	Sheet metal bending device.....	202
<b>14</b>	<b>Safety Regulations .....</b>	<b>207</b>
14.1	Rules .....	207
14.2	Emergency STOP control release .....	208
14.3	Example of a control release .....	209
<b>Appendix</b> .....	<b>211</b>	
Solutions according to the examples.....		211
Example of a solution according to chapter 6.8.1 .....		212
Example of a solution according to chapter 6.8.2 .....		214
Example of a solution according to chapter 6.8.3 .....		216
Example of a solution according to chapter 6.8.4 .....		218
Example of a solution according to chapter 6.8.5 .....		220
Example of a solution according to chapter 6.8.6 .....		222
Example of a solution according to chapter 6.9.3 .....		226
Example of a solution according to chapter 7.4 .....		230
Example of a solution according to chapter 8.6.1 .....		233
Example of a solution according to chapter 8.6.2 .....		234
Example of a solution according to chapter 8.6.3 .....		240
Example of a solution according to chapter 9.1.1 .....		246
Example of a solution according to chapter 9.1.2 .....		247
Example of a solution according to chapter 9.1.3 .....		250
Example of a solution according to chapter 10.3 .....		252
Example of a solution according to chapter 11.2 .....		258
Example of a solution according to chapter 11.3 .....		265
Example of a solution according to chapter 12.1 .....		268
Example of a solution according to chapter 12.2 .....		284
Example of a solution according to chapter 12.3 .....		290
Example of a solution according to chapter 12.4 .....		298
Example of a solution according to chapter 12.5 .....		307

Example of a solution according to chapter 12.6 ..... 312

Example of a solution according to chapter 12.7 ..... 319

Example of a solution according to chapter 12.8 ..... 337

Example of a solution according to chapter 12.9 ..... 351

Example of a solution according to chapter 13.5 ..... 367

# 1 Introduction

The programmable logic controller (PLC) has the job of carrying out open-loop or closed-loop control of a machine or plant's individual operations in accordance with a specified function cycle depending on encoder signals.

## 1.1 Number system

A programmable logic controller does not use the decimal system for processing the addresses of storage locations, inputs, outputs, times, bit memories, etc.; rather, the **dual number system** is used.

The dual number system only includes the digits 0 and 1 that can easily be represented and evaluated in data processing. **It is a binary number system.**

The significances of a dual number are, as shown below, assigned to the powers of 2.

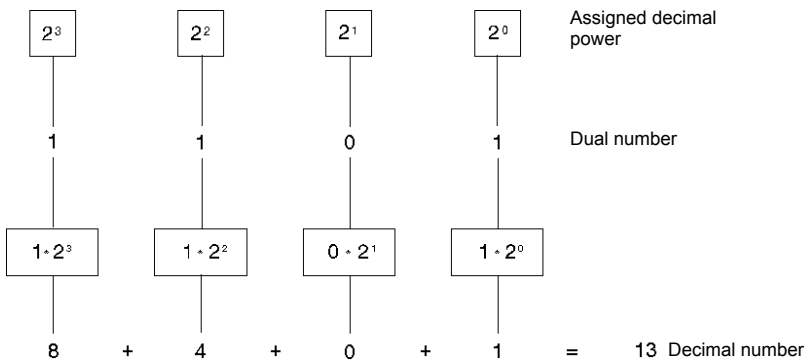


Figure 1.1 Each digit within a dual number is assigned to a power of two

## 1.2 Terms from computer science

In connection with programmable logic controllers, people often use terms from data or information processing like bit, byte, word and double word.

### 1.2.1 Bit

A bit (an abbreviation of binary digit) is the smallest binary (dual-value) unit of information.

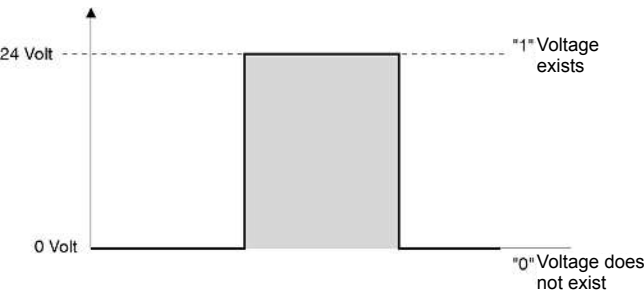


Figure 1.2  
A bit can take  
on a signal state  
of "1" or "0"

### 1.2.2 Byte

A byte is a term for a unit of eight bits.

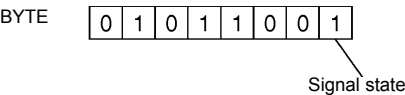


Figure 1.3  
A byte has a size  
of 8 bits

### 1.2.3 Word

A word consists of two bytes or 16 bits. Using "words", you can, for example, represent:

- ☐ Dual numbers,
- ☐ Letters,
- ☐ Control instructions.

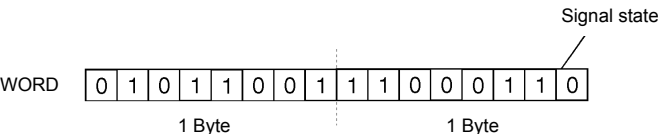
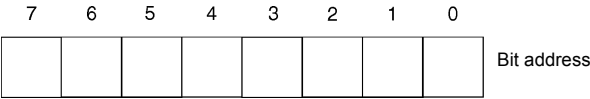


Figure 1.4  
A word has a size  
of 2 bytes or 16 bits

1.2.4 Bit address

For the system to detect and address bits, each individual bit in a byte is assigned a digit, what is known as a bit address.

Figure 1.5  
In each byte, the right-most bit is given bit address 0 and the left-most one gets the bit address 7

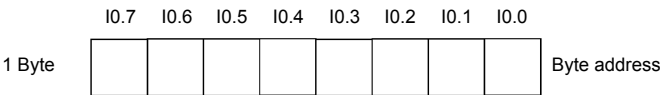


1.2.5 Byte address

The individual bytes are also given numbers, i.e. the byte addresses. Additionally, the operand is labelled. This means, for example, that IB 2 stands for input byte 2 or QB 4 represents output byte 4. Individual bits are uniquely addressed by the combination of the bit and byte addresses.

The bit address is separated from the byte address by a full stop. To the right of the full stop, there is the bit address, with the byte address being to the left of it.

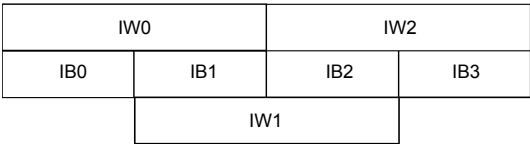
Figure 1.6  
Example:  
Byte address I0.0 or  
Q4.5



1.2.6 Word address

The numbering of words yields the word address. When using words, e.g. IW (input word), QW (output word), MW (memory word), DW (data word), the word address is always the lower byte address of the two associated bytes.

Figure 1.7  
Word address







## 2 Arrangement of a PLC

PLCs are mass-produced. Initially, they do not yet have a task. Manufacturers integrate all the components necessary for the control engineering such as the logic elements, latching/unlatching functions, times, counters, etc. and these components are linked to form a functioning controller by means of programming. There are a large number of different control units that differ from one another by virtue of the following functional units:

- ☐ Inputs and outputs,
- ☐ Storage locations,
- ☐ Counters,
- ☐ Times,
- ☐ Bit memory functions,
- ☐ Special functions,
- ☐ Operating speed,
- ☐ Type of program processing.

Relatively large control units are assembled from individual components on a modular basis. Using a modular system like this, it is possible to start from a basic configuration and build up PLC systems that you can customize for specific applications. For relatively small control tasks, manufacturers offer compact control units. These are self-contained devices that have a fixed number of inputs and outputs.

## 2.1 Structure of a PLC

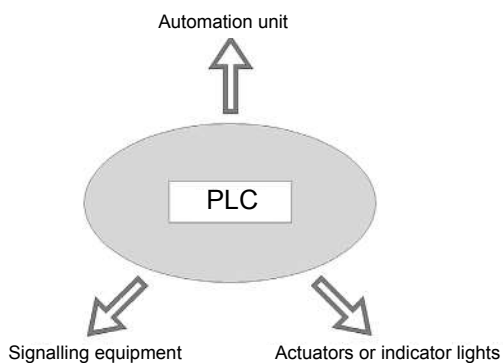


Figure 2.1  
Basic structure of a  
programmable logic  
controller

## 2.2 Structure of an automation unit

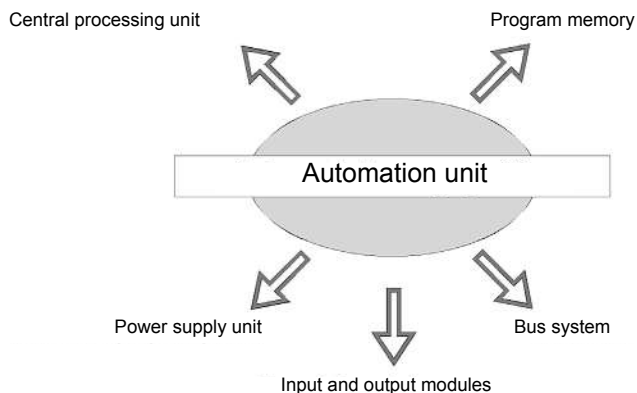


Figure 2.2  
Basic structure of an  
automation unit

- ☐ Signalling equipment supplies digital and analog signals for further processing to the PLC,
- ☐ Actuators or indicator lights receive digital and analog signals from the PLC.

## 2.3 Hardware requirements

To be able to work with the examples below, you need the following hardware components.

### DIN rail

The individual modules are mounted on the DIN rail.

### Power supply unit (PS)

The power supply unit converts the 120/230 V AC mains supply to 24 V DC to supply the S7-300 modules.

### Central processing unit (CPU)

The central processing unit is for executing the user programs. It communicates with other modules via the MPI interface.

### Input and output module

The system inputs signals from the encoder via the input module into the S7-300 or outputs them to the output module. An LED display shows the signal state at the modules.

### MPI cable

The MPI cable connects the computer to the central processing unit (CPU). Without this cable it is not possible to communicate with the PC.

### Personal computer (PC)

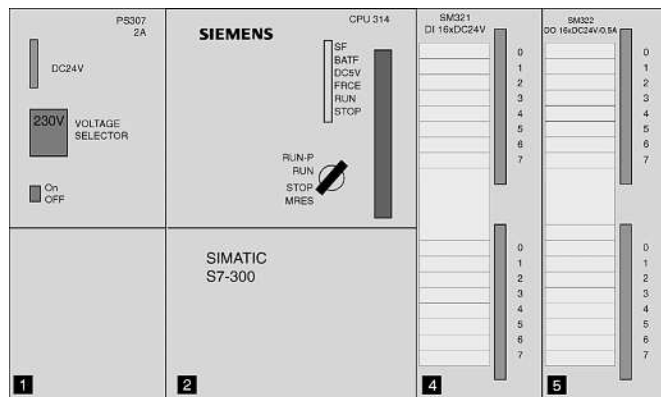
The PC is for configuring, parameterizing and programming the S7-300.

### Programming unit (PU)

The PU is also for configuring, parameterizing and programming the S7-300.

For this course, we will be using as the setup an S7-300 (CPU 314) with one digital input module (16 inputs) and one digital output module (16 outputs). In this connection, the input module contains addresses I 0.0...I 1.7 (IW0) and the output module contains addresses Q 4.0...Q 5.7 (QW4).

Figure 2.3  
Structure of the  
training unit



### 2.3.1 Hardware structure

When setting up an S7-300, you must observe slot rules. You plug in the modules from left to right:

- ☐ You must always plug in the power supply unit (PS) as the first module on the DIN rail,
- ☐ You must plug in the central processing unit (CPU) as the second module,
- ☐ You plug in the input and output modules after the CPU,
- ☐ Next to the central processing unit (CPU), you may only plug in at a maximum 8 signal modules,
- ☐ You can plug in the modules horizontally or vertically.

## 2.4 Software requirements

For programming the S7-300, we use the Windows XP operating system as well as **Version 3.xx** of the **STEP 7 software package**. This version integrates the following three methods of representation: Statement List (STL), Ladder Diagram (LAD) and Function block diagram (FBD). Previous versions, e.g. 2.xx did not contain the Function block diagram (FBD) representation.

For programming the exercises in the book, we assume that Siemens' STEP 7 software package is already installed. To be able to use the STEP 7 programming language, you also need a knowledge of the Windows XP operating system (file management). You must carry out installation on the Windows XP user interface.

### 2.4.1 STEP 7 programming language

Following the replacement of SIMATIC S5 by SIMATIC S7 in the Autumn of 1995, a new programming language (STEP 7) was developed based on the IEC 1131 standard. This programming language provides the entire functionality for parameterizing, configuring and programming the S7-300 automation unit. STEP 7 runs under Windows XP and works on an object-oriented basis. The symbols for the objects are mapped on the graphical user interface. STEP 7 has an integrated online help system that provides valuable hints and tips.

#### 2.4.2 Objects

On the graphical user interface, the system represents objects with symbols. A symbol is assigned to a specific object. STEP 7 objects offer access to the following processing functions:

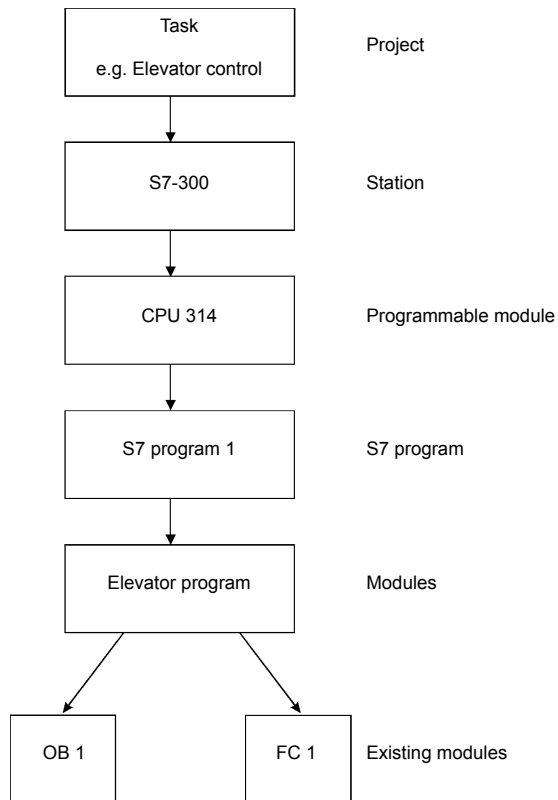
- ☐ Create and open objects,
- ☐ Edit and save objects,
- ☐ Rename objects,
- ☐ Delete objects,
- ☐ Copy and paste objects.

An object can, for example, contain as a symbol a project, a SIMATIC station, an S7 program, etc.

### 2.4.3 Projects

STEP 7 allows you to divide a plant into projects. A project contains all the data for an automation solution. You can consider a project to be the most important object.

Figure 2.4  
Project structure with  
SIMATIC S7



The data for an automation solution is managed in a project. In our example, the project is a lift controller. This project contains the entire automation solution. The project includes, for example, the S7-300 station as well as the programmable controller (PC) with the CPU 314. You enter the S7 program in this PC. The program is then broken down into the corresponding blocks, such as, OB 1 and FC 1, for example.

#### **2.4.4 Configuring an S7-300**

When configuring, the S7 modules are arranged in a configuration table.

You can choose the modules from an electronic catalogue and enter them in the configuration table to match the slot. The slot in the configuration table corresponds to the slot on the subrack. An address is then automatically assigned to each module.

#### **2.4.5 Parameterization**

You can set the properties of the individual modules in various ways. One of the parameters is, for example, cycle time monitoring with the CPU S7-300. The parameter can be changed.

## 3 Way of Functioning of a PLC

### 3.1 Modules of the PLC

The various modules of a PLC are explained in Figure 3.1.

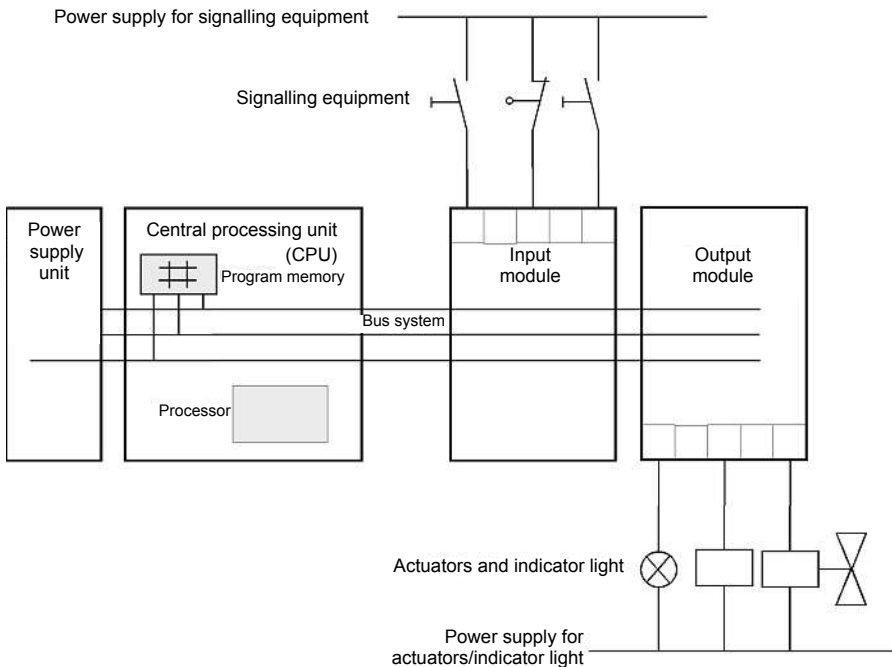


Figure 3.1 Interaction and arrangement of the modules of a PLC

#### 3.1.1 Power supply unit

The power supply unit generates the voltage for the programmable controller's electronic modules from the mains voltage. This voltage is 24 V DC. The voltages for signalling equipment, actuators and indicator lights, which are above 24 V (24 to 220 V), are supplied by mains units or control-power transformers that are additionally provided for this purpose.

### 3.1.2 Program memory

Storage elements are components in which information can be stored in the form of binary signals. Semi-conductor components are mainly used as program memory. A memory comprises:

- ❑ 512,
- ❑ 1024,
- ❑ 2048 etc. storage locations.

It is normal to state the capacity of program memory (i.e. the number of storage locations) in **multiples of 1 K** (here, 1 K stands for 1024). In each storage location, you can use a programming unit to write (program) a control instruction. In this connection, each of the binary elements of a storage location can take on a signal state of "1" or "0".

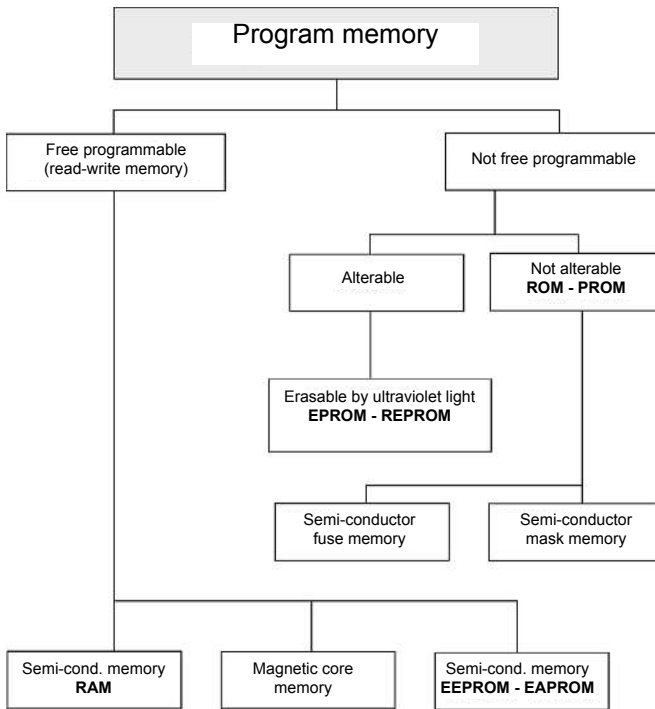


Figure 3.2  
Schematic of a  
program memory

**RAM** (random-access memory)

RAM is semi-conductor Read-Write memory. The individual storage locations are labelled with addresses that are used to freely access the storage locations.



It is possible to write information to the storage locations as often as you like. The information is read out without its content being lost.

However, RAM is volatile memory, i.e., the information is lost when the power supply fails or is switched off. RAM is deleted electrically.

## **ROM**

Read-only memory contains information that cannot be deleted or changed.

ROM stands for Read-Only memory. Manufacturers enter the information and they are the only ones who can change it.

## **EPROM**

EPROM stands for erasable programmable read-only memory. You can erase the entire content of the EPROM using UV light; after this, you can reprogram it.

This makes it very suitable for transportation without data loss.

## **REPROGRAM**

REPROGRAM stands for reprogrammable erasable read-only memory. You can also only erase its contents using UV light.

## **EEPROM**

EEPROM stands for electrically erasable programmable read-only memory. In an EEPROM, it is possible to delete and write to each storage location electrically.

## **EAPROM**

EAPROM stands for electrically alterable programmable read-only memory.

### **3.1.3 Central processing unit (CPU)**

The voltage coming from the signals is switched to the terminal strip of the input module. In the CPU (central processing unit, see Figure 3.3), the processor processes the program in memory and when doing this polls whether the individual inputs of the unit are carrying a voltage or not. Depending on this status at the inputs and the program in memory, the processor instructs the output module to switch a voltage to the corresponding connections of the terminal strip. Again, depending on the voltage status at the connections of the output module, the system switches on or off the connected actuators or indicator lights.

The address counter polls successively (on a serial basis) the program memory instruction by instruction and brings about program-dependent transfer of information from the program memory to the statement register. All the memory in a processor is normally referred to as registers.

The processor gets its instructions from the statement register. While the processor is processing the current statement, the address counter shifts the next statement into the statement register.

The status transfer of the inputs to the process input image (PII) is followed by linking, application of the timers, counters, and accumulators and transfer of the results of logic operations (RLO) to the process output image (PIQ). If the system detects a block end (BE) after processing of the user program, the respective status is transferred from the PIQ to the outputs.

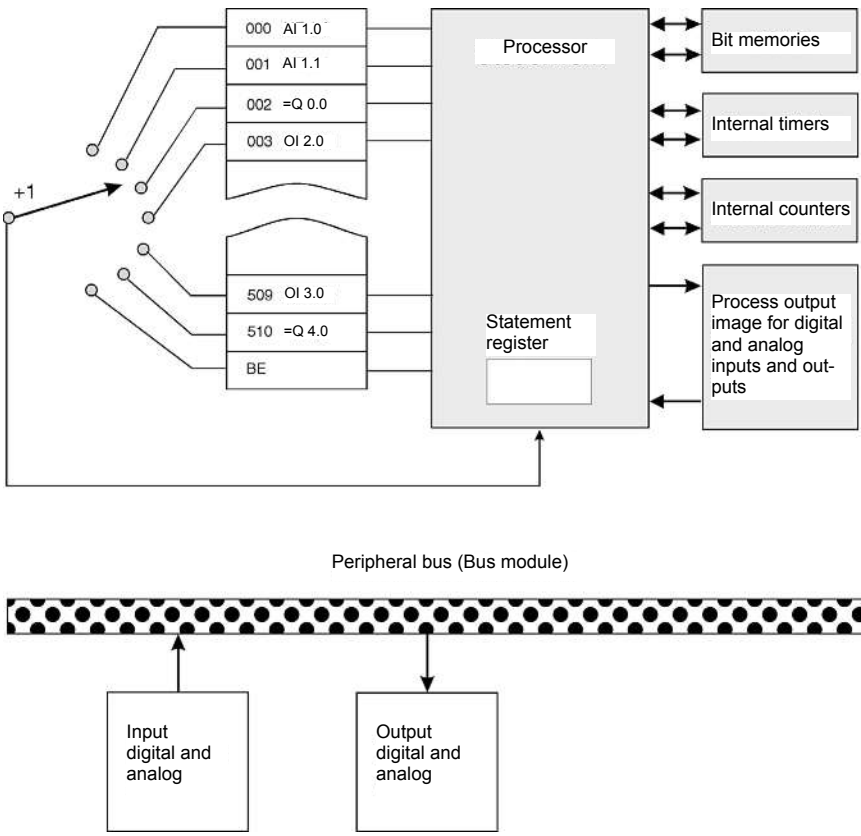


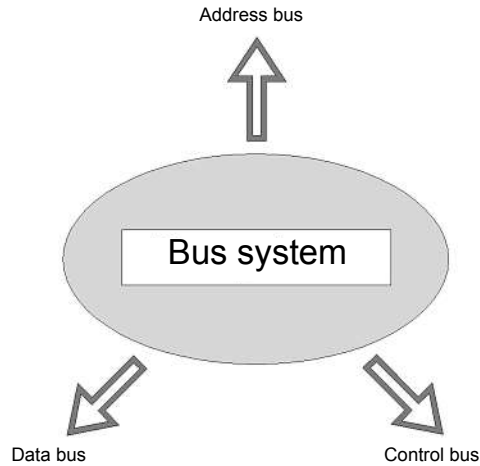
Bild 3.3 Central processing unit (CPU)

The peripheral bus handles data exchange between the central processing unit and the peripherals. Peripherals include the digital input and output modules, the analog input and output modules as well as the timer, counter and limit value modules.

### 3.1.4 Bus system

The bus system (see Figure 3.4) is a group line for transferring signals. The system exchanges signals in the programmable controller (PC) between the processor and the input and output modules across what is known as a process bus system. The process bus consists of three parallel signal lines:

Figure 3.4  
The bus system



- ☐ Using the address bus, the system addresses the addresses on the individual modules,
- ☐ Using the data bus, the system transfers data, e.g. from the input to the output modules,
- ☐ On the control bus, the system transfers the signals for controlling and monitoring the function cycle within the programmable controller.

### 3.1.5 Input and output modules

Using the input and output modules, the system exchanges data with the process to be controlled.



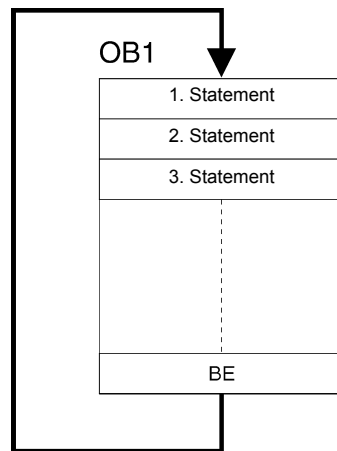
## 4 Program Processing and Programming

You have the option of linear and structured program processing.

### 4.1 Linear program processing

Linear program processing (see Figure 4.1) is mostly used for simple, not too comprehensive controllers and can be implemented in a single organization block (OB). In this connection, the control unit processes the statements in the order in which they are stored in the program memory. When the end of program (BE) is reached, program processing starts from the beginning again. The time that a control unit needs to carry out processing of all the statements once is called the cycle time. This is why this type of processing is also referred to as cyclical processing.

Figure 4.1  
Linear program  
processing



## 4.2 Structured programming

With comprehensive control jobs, you divide the program into small manageable program blocks that are arranged by function. This has the advantage of allowing you to test program sections individually and, if they function, to combine them into an overall function. STEP 7 provides the following user blocks for this:

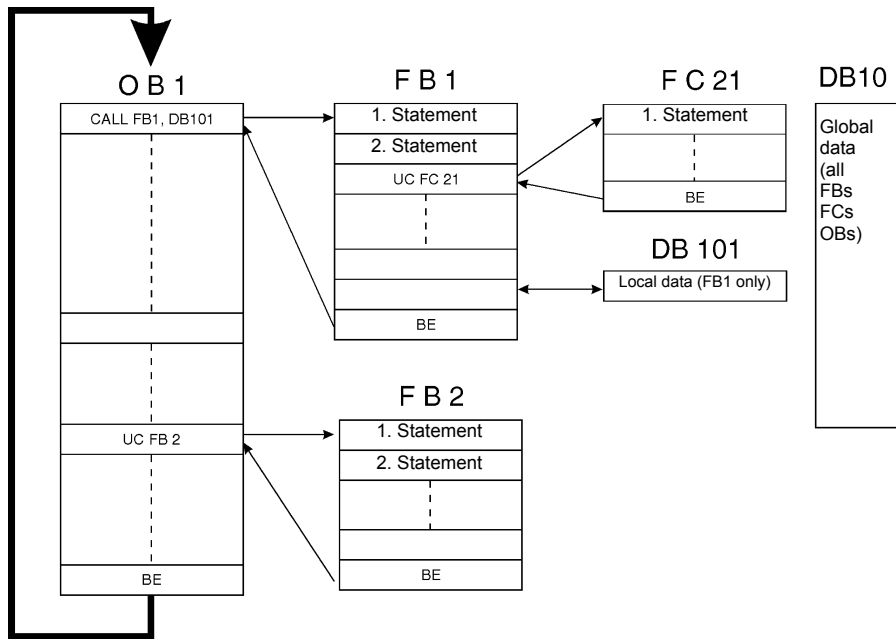


Figure 4.2 Structured programming using user blocks with: OB organization block, FB function block, FC function, DB data block

### Organization block (OB)

The operating system cyclically calls an OB which functions as the interface between the user program and the operating system. In the OB, the system informs the processor of the programmable controller (PLC) by means of block call commands about the program blocks that it has to process.

### Function block (FB)

FBs have an assigned memory area. When an FB is called, it is possible to assign a data block (DB) to it. It is possible to access the data in this instance DB by means of calls from the FB. An FB can be assigned to different DBs. It is possible to call further FBs and FCs via block call commands in a function block.

**Function (FC)**

A FC has no assigned memory area. The local data of a function is lost after processing of the function.

**Data block (DB)**

DBs are used to make available memory for data variables. There are two types of data blocks: global DBs where all the OBs, FBs and FCs read stored data or can write data themselves to the DB and instance DBs that are assigned to a specific FB.

**System blocks for standard and system functions**

System blocks are ready-to-use functions that are stored in the CPU. Users can call these blocks and use them in the program.

**System blocks of STEP 7****System function block (SFB)**

A function block that is stored in the CPU's operating system and that users can call.

**System function (SFC)**

A function that is stored in the CPU's operating system and that users can call.

**System data block (SDB)**

A memory area in the program that various STEP 7 tools (e.g. S7-Configuration, Communication Configuration, etc.) create to save data for the automation system.

The system blocks that are available differ from CPU to CPU and can be seen in the STEP 7 reference manuals.

## **4.3 Control instruction**

For processing by a programmable logic controller, the control job is broken up into individual control instructions (see Figure 4.3). A control instruction is the independent unit of a controller program. It represents a directive. The designations, identifiers and symbols for control instructions are specified in DIN 19 239.

Control instruction		
Operation part	Operand part	
	Identifier	Parameter
A	I	0.0

Figure 4.3  
Structure of a control instruction

#### 4.3.1 Operation part

The operation describes the function to be executed. Figure 4.4 shows the differentiation between operations according to DIN 19 239.



Figure 4.4

#### 4.3.2 Examples for digital operations









L	Load
T	Transfer
>I	Compare to "greater than integers"
==R	Compare to "same as real numbers"
-R	Subtraction with real numbers

#### 4.3.3 Examples of binary operations

Figure 4.5 shows an extract from DIN 19 239.



Figure 4.5  
Extract from DIN 19 239

FBD	LAD	STL	
		A	AND
		N	NOT
		O	OR
		=	Assignment

4.3.4      Examples of organizational operations

- CC      Block call conditioned
- UC      Block call unconditioned
- OPN    Open data block
- JU      Jump, unconditional
- JC      Jump, conditional
- BEU    Block end, unconditional
- BEC    Block end, conditional

4.3.5      Operand part

The operand part contains all the information that is necessary for executing the operation. It states what the processor is to execute an operation with. The operand identifier contains the type of operand:

- I      for inputs
- Q      for outputs
- M      for bit memories
- T      for times
- C      for counters
- OB    for organization blocks
- FB    for function blocks
- FC    for functions
- DB    for data blocks
- SFB   for system function blocks
- SFC   for system functions
- ..     for 32-bit constants
- etc.

The operand parameter indicates the address of the operand.

## 4.4 Addressing

### 4.4.1 Symbolic Addressing

Symbolic addressing often makes things easier to understand. It makes it possible to assign a symbolic name to a specific absolute address. This means that you can, for example, assign the name END\_STOP and the data type BOOL to input I0.0. Each symbolic name may appear only once. You carry out assignment using the Symbol Editor tool.

### 4.4.2 Absolute addressing

In STEP 7 there are the following absolute addressing options:

- ☐ Immediate addressing,
- ☐ Direct addressing,
- ☐ Memory-indirect addressing.

### 4.4.3 Immediate addressing

In the case of immediate addressing, the operand is contained in the operation, i.e. there either follows directly the value with which this operation is to work, or it implies the operand.

#### Examples:

SET      set RLO to 1  
L+27    load the integer 27 into ACCU 1  
L L#+5   load the 32-bit constant 5 into ACCU 1

#### 4.4.3.1 Direct addressing

In the case of direct addressing, the operand address is contained in the operation, i.e. the operand specifies the address of the value that the operation is to process.

The operand consists of an operand identifier and a parameter and points directly to the address of the value.

#### Examples:

AI 0.0   execute logical ANDing of input bit I0.0  
L IB0    load input byte IB0 into ACCU 1  
=Q4.0    assign output bit Q4.0 to the RLO

#### 4.4.3.2 Memory-indirect addressing

In the case of memory-indirect addressing, the operand address is specified indirectly by means of another operand that contains the address of the first one, i.e. the operand indicates the address of the value that the operation is to process.

The operand indirectly specifies the address of the value or the number by means of a pointer. The word or double word can be located in a bit memory (M), data block (DB), instance data block (DI) or in local data (L).

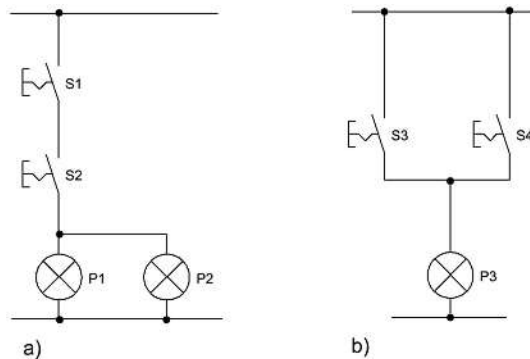
#### Examples:

- AI [MD3]      Execute logical ANDing of input bit. The exact address is located in memory double word MD3.
- LIB [DID 4]    Load the input byte into ACCU 1. The exact address is located in an instance data double word DID 4.
- OPN DB [LW 2] Open the data block. The data block number is located in a local data word LW 2.

### 4.5 Program representation

Any program representation is based on the terms of reference which describe the function that is to be implemented in a program. The terms of reference are usually in the form of a schematic diagram (see Figure 4.6).

Figure 4.6  
Schematic diagram:  
a) Logical AND operation,  
b) Logical OR operation

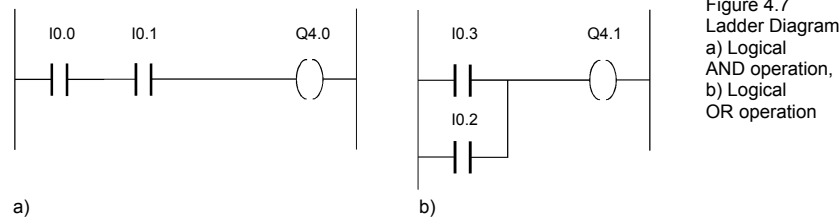


In STEP 7, you can carry out representation and programming of a program in three different ways. The following programming types are possible:

- ☐ Ladder Diagram LAD,
- ☐ Function Block Diagram FBD,
- ☐ Statement List STL.

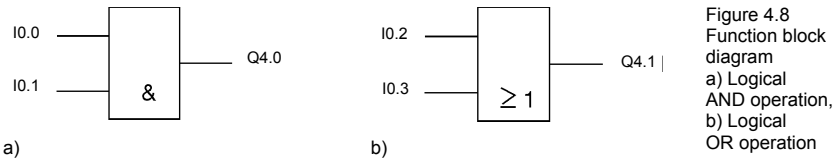
### 4.5.1 Ladder diagram (LAD)

A ladder diagram (see Figure 4.7) is a representation in diagram form of the control job using symbols according to DIN 19 239 that are also commonly used in the USA. It is very similar to a conventional wiring diagram, however, it takes into account viewing on a computer screen and shows the individual lines horizontally rather than vertically.



### 4.5.2 Function block diagram FBD (STEP 7 V3.x and above)

A Function block diagram (see Figure 4.8) is a representation in diagram form of the control job using symbols according to DIN 40 700 and DIN 19 239. The individual functions are represented by a symbol with a function identifier. The inputs are on the left-hand side of the symbol with the outputs on the right-hand side. It uses Boolean algebra logic for representation with its familiar logic boxes.



### 4.5.3 Statement list STL

In a statement list (see Figure 4.9), you describe the control job using individual control instructions. The control instruction (operation and operand) represents the task using mnemonic abbreviations (DIN 19 239).

Figure 4.9  
The statement list

Address	Operation part	Operand part		
		Identifier	Parameter	
000	A	I	0.0	Logical AND operation
002	A	I	0.1	
004	=	Q	4.0	Logical OR operation
006	O	I	0.2	
008	O	I	0.3	
00A	=	Q	4.1	
00C	BE			

Each type of representation has its own special characteristics. If you keep to specific rules when programming, there is no problem in compiling the program for all three types of representation.

Controller programs that have been programmed in LAD or FBD can always be compiled to Statement List (STL). Programs are always saved in the control units' program memory in STL (in machine language, however).

## 4.6 Bit memories

Bit memories are used for logic operations in the controller with which there is no need for passing a signal outside the controller. These bit memories are electronic storage elements (RS flipflops) that can be used to save the signal status conditions "0" and "1".

Every PLC has available a large number of bit memories. They are programmed like outputs. If the operating voltage fails, the saved contents of the bit memories are lost.

### 4.6.1 Retentive bit memories

However, some of these bit memories are retentive (non-resetting on supply failure). A buffer battery in the PLC bridges the power failure. This means that the logical status conditions are retained.

Retentive bit memories

- ☐ retain the last status when the supply voltage is switched off,
- ☐ retain their status when changing from operating mode **RUN > STOP**,
- ☐ can be reset by the user program with menu:  
    **PLC > Diagnostic/Setting > Clear/Reset.**

By using retentive bit memories, you can save the last plant or machine status before leaving the operating status. In the case of a restart, the plant or machine can continue working at the point at which it came to a standstill. You specify the retentive areas by parameterizing in the S7 Configuration tool.

#### **4.6.2 Non-retentive bit memories**

- ☐ Are reset when changing from operating mode RUN > STOP and in the case of POWER ON.

# 5      Logic operations

## 5.1      Basic logic operations

Entering the logical AND operation program using the PC according to FBD (see Figure 5.1).

Table 5.1

Assignment list		
Symbol	Operand	Comment
S1	I0.0	«NO contact» switch
S2	I0.1	«NO contact» switch
P1	Q4.0	Indicator light

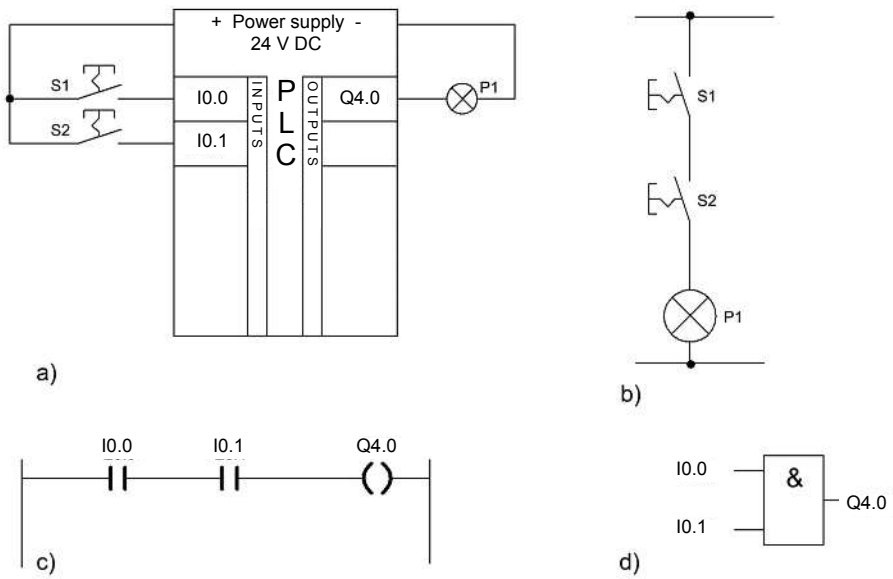


Figure 5.1    Entering the logical AND operation: a) Connection to the PLC, b) Schematic diagram, c) LAD, d) FBD

Before you enter the logical AND operation program in the SIMATIC S7, you must create a project. Only then can you configure and parameterize the CPU. To be on the safe side, you should first clear the CPU to ensure that there are no "old" blocks in it. You program the S7-300, according to the following procedure:

Clear the CPU

Create the project

Configure and parameterize the CPU

Save the configuration table

Load the configuration into the CPU

5.1.1 Clearing the CPU

Proceed as follows:

⇒ Click on the SIMATIC Manager



⇒ In the SIMATIC Manager click on menu item:  
**PLC > Display Accessible Nodes**



⇒ The following window is displayed:



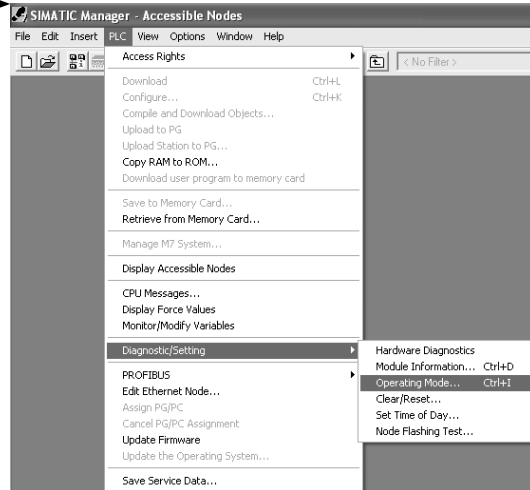


⇒ The following window is displayed:



Click on the target system using menu item:  
**PLC > Diagnostic/Setting > Operating Mode...**  
... display the current operating status of the CPU.  
You switch the CPU to the STOP status and confirm by clicking on **OK** and then leave the dialog box by clicking on **Close**.  
The CPU enters the STOP status.

You can only carry out a memory reset of the PLC in the STOP status.

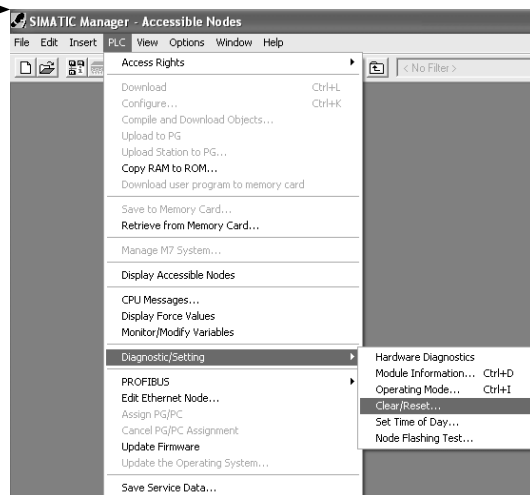


You can now clear the CPU.

Click on menu:  
**PLC > Diagnostic/Setting > Clear/Reset...**

Confirm the action.

**Important:** A key-operated switch is attached to the CPU.  
This switch must be in the RUN-P or STOP position.



The CPU is now reset, i.e. the entire user program is deleted, the system parameters and module parameters are reset to the default settings. With this, the CPU severs all the existing connections.

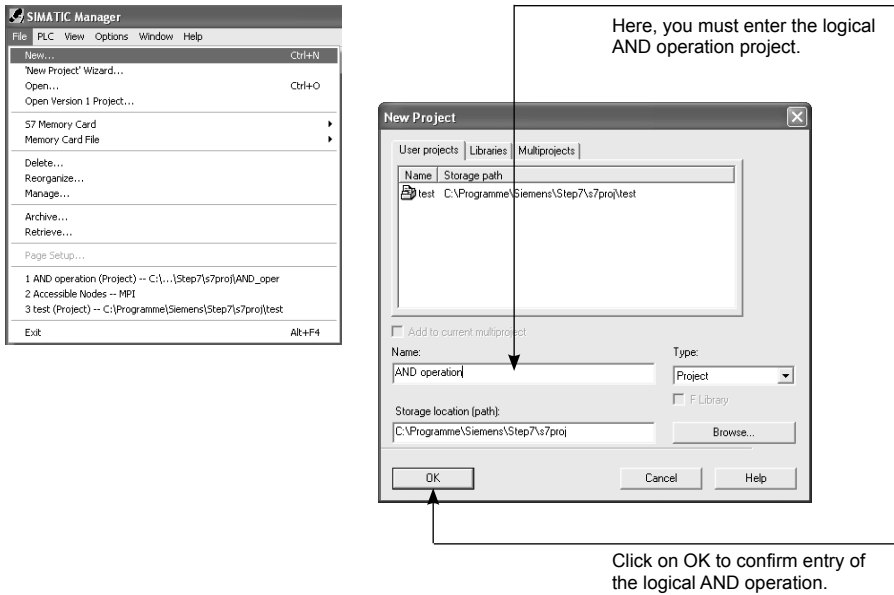
⇒ Close the window and return to SIMATIC Manager.

### 5.1.2 Creating projects

Now, you create the logical AND operation project for the user program.

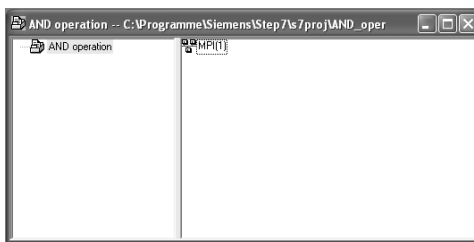
⇒ In the SIMATIC Manager click on menu item: **File > New...**

⇒ The following window is displayed:



⇒ The project has now been created.

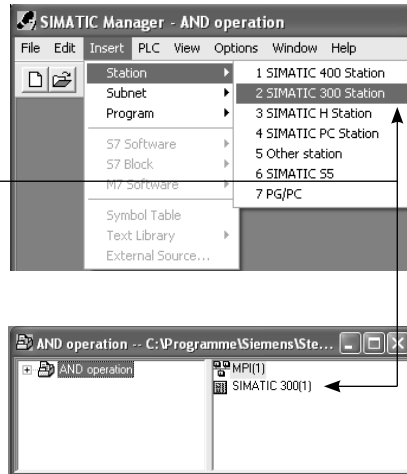
⇒ The following window is displayed:



### 5.1.3 Inserting the SIMATIC 300 station

Go to menu item: **Insert > Station > SIMATIC 300 Station**

The system inserts a SIMATIC 300 station in the current project (**logical AND operation**). The station and its name are displayed in the project window.

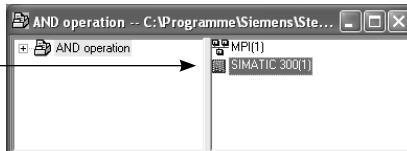


A hardware station has now been inserted in the logical AND operation project.

### 5.1.4 Configuring and parameterizing

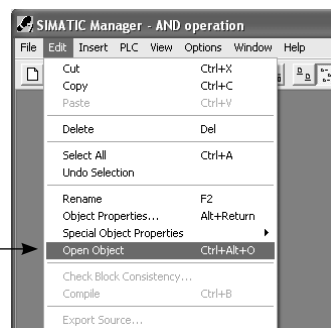
To be able to enter the logical AND operation user program, you must first configure and parameterize the SIMATIC station.

Select the SIMATIC station in the project.

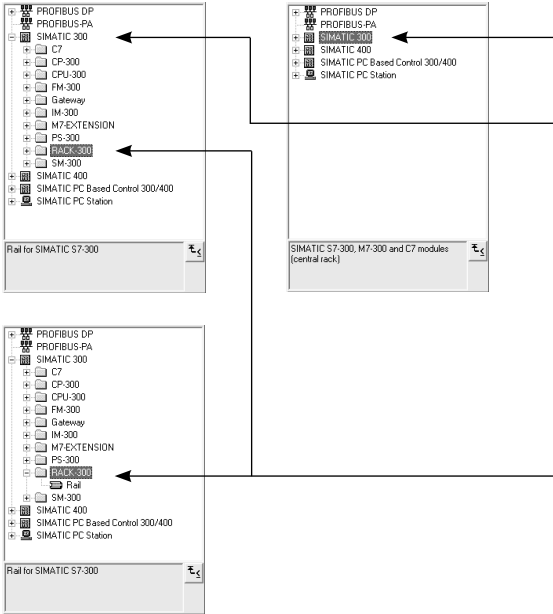


⇒ Call the configuration table using menu item: **Edit > Open Object**

Click here > then insert:  
Hardware components



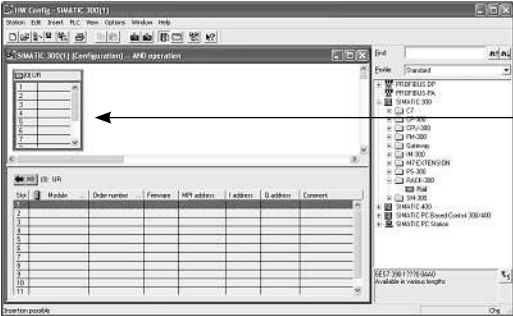
The system displays the hardware catalog that contains all the hardware components.



Select the SIMATIC 300 station. Click on the «+» sign on the left-hand side of the SIMATIC 300 station.

The «+» sign turns into a «-» sign (subdirectory). This subdirectory contains all the hardware components of the S7-300 station.

Choose the **Rail** in Rack 300. The DIN rail is the subrack that all the hardware components are plugged into.



Drag the DIN rail to the workspace and drop it there.

(Click on the DIN rail, hold down the mouse button and drag it to the desired location).

The system displays a table containing the DIN rail and the slots.

⇒ The detail view of the configuration table is shown in the window.

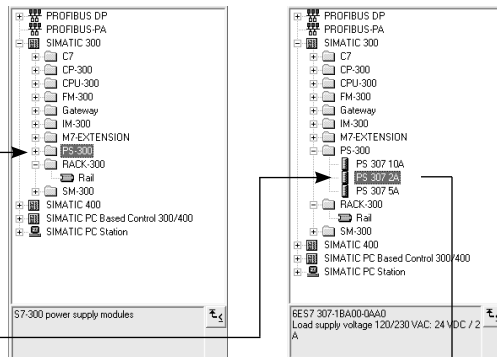
The order numbers and addresses of the modules are displayed in the detail view of the configuration table

Slot	Module	Order number	Firmware	MPI address	I address	Q address	Comment
1							
2							
3							
4							
5							
6							
7							
8							
9							
10							
11							

You now put the power supply onto the DIN rail. For all subsequent examples, we will be using the 2 A PS 307 power supply.

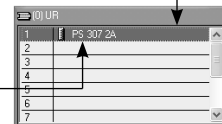
### 5.1.5 Arrangement of the power supply

Click on the «+» sign in the hardware list next to the power supply **PS 300**.



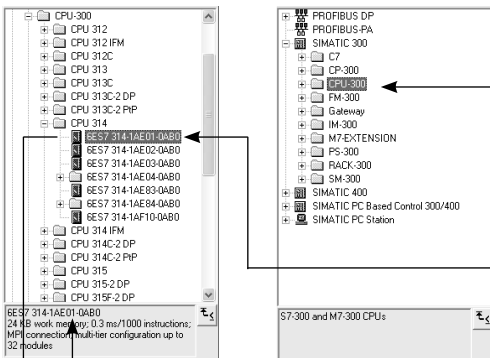
The system opens the subdirectory containing the different power supplies.

Drag the power supply **PS307 2A** into the configuration table and drop it onto slot 1.



The power supply is now installed.

### 5.1.6 Arrangement of the CPU 314



Click on the CPU 300 in the hardware list and use the «+» sign to open the subdirectory containing the different CPU types.

Choose the **CPU314** with order number 314-1AE01-0AB0. We will be using this CPU to test all the exercises.

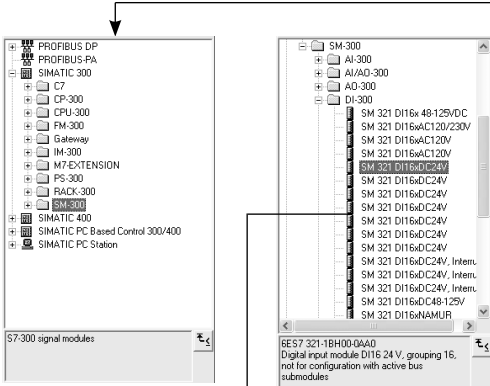
Order number

Slot	Module
1	PS 307 2A
2	CPU 314
3	
4	
5	
6	
7	

Drag the CPU into the configuration table and drop it onto slot 2.

The CPU 314 is now installed.

### 5.1.7 Arrangement of the input module



Click on the hardware list next to the SM modules **DI300**, then click on the «+» sign.

The system opens the subdirectory containing the different DI (digital input modules). Choose module 16 DI with order number 321-1BH00-0AA0. This module has 16 digital inputs.

Slot	Module
1	PS 307 2A
2	CPU 314
3	
4	DI16xDC24V
5	
6	
7	

Drag the input module into the configuration table and drop it onto slot 4.

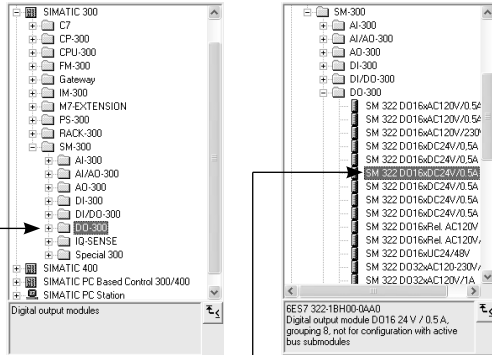
**Important:**  
Slot 3 must be empty. It is reserved for the IM module.

The digital input module (DI) with 16 inputs is now installed.

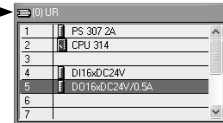
### 5.1.8 Arrangement of the output module

Click on the hardware list next to the SM modules **DO300**, then click on the «+» sign.

The system opens the subdirectory containing the different DO (digital output modules). Choose module 16 DO with order number 322-1BH00-0AA0. It has 16 digital outputs.



Drag the output module into the configuration table and drop it onto slot 5.

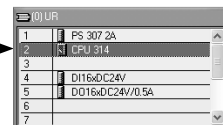


The digital output module is now installed. You have now completed the configuration table.

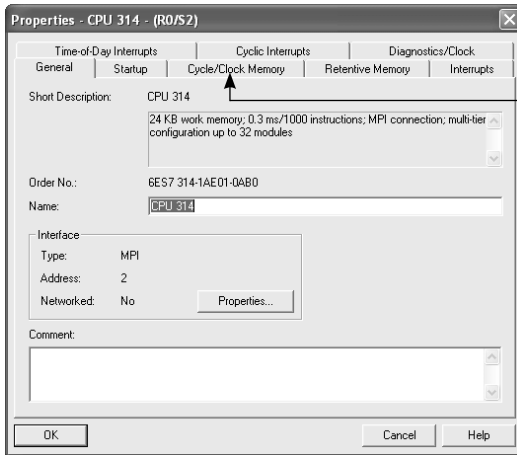
### 5.1.9 Parameterizing the CPU 314

To be able to use the CPU 314 for exercises, you must parameterize it appropriately.

In the configuration table,  
double click on slot 2 (CPU314).



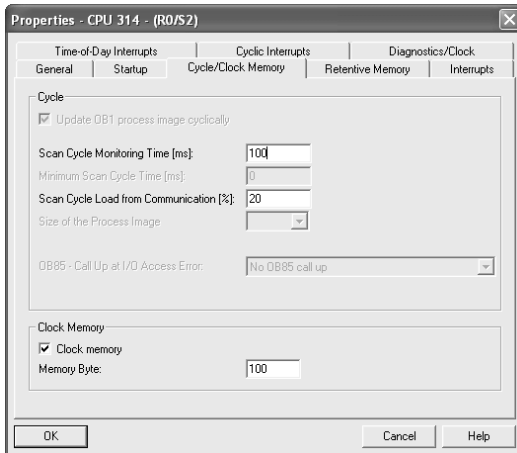
⇒ The following window is displayed:



Click on **Cycle/Clock Memory**

In this dialog box, you store the information on the CPU and its settable parameters.

For other exercises, we will be using a clock memory (built-in clock generator). To be able to use the clock generator, you must parameterize the CPU.



Bit memory 100 (MB 100) is used. The individual bits of MB 100 can then be called using different frequencies.

Change the **Scan Cycle Monitoring Time** to 100 ms.

Click on **OK** to leave the dialog box.

The CPU is now parameterized for exercises.

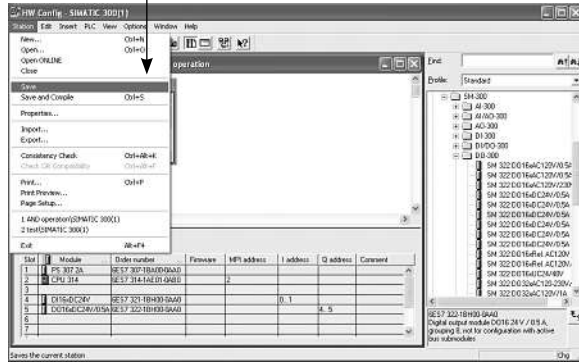


### 5.1.10 Saving the global configuration

To retain the global configuration, you must save it in the logical AND operation project.

Choose menu item:  
**Station > Save**

The entire configuration is saved  
in the logical AND operation  
project on the PC's hard drive.



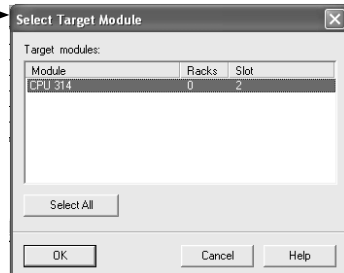
The global configuration is now saved in the logical AND operation project.

### 5.1.11 Transferring the configuration to the CPU

For the configuration to take effect, you must load it in the S7-300.

Go to menu item:  
**PLC > Download...**

Select the modules using menu  
item:  
**Select All**  
Confirm with **OK**.



⇒ The following window is displayed:

Select Node Address

Over which station address is the programming device connected to the module CPU 315-2 DP?

Rack: 0  
Slot: 2

Target Station: ☒ Local  
☐ Can be reached by means of gateway

MPI address	Module type	Station name	CPU name	Plant designation
2	CPU 315-2 DP			

Accessible Nodes				
2	CPU 315-2 DP			

Update

OK Cancel Help

Confirm with **OK**.

The system downloads the generated global configuration into the CPU with node address 2. The changed parameters such as the clock memory and scan cycle monitoring time do not become effective until you carry out a restart.

Stop Target Modules

The following modules will be stopped for loading of the system data.

Module	Racks	Slot
CPU 315-2 DP	0	2

OK Cancel Help

The displayed modules will be stopped for loading of the system data.

Confirm with **OK**.  
The download will be finished.

Restart your system.

## 5.2 Logical AND operation user program

You can create a user program in STL (Statement List), LAD (Ladder Diagram) or FBD (Function Block Diagram). First of all, you program the user program in the FBD type of representation.

For the **logical AND operation** example we have chosen, you need two blocks, namely an organization block and a function block.

### Organization block (OB 1)

Organization block OB 1 is the interface between the operating system of the CPU and the user program. In this block, you specify the sequence of processing.

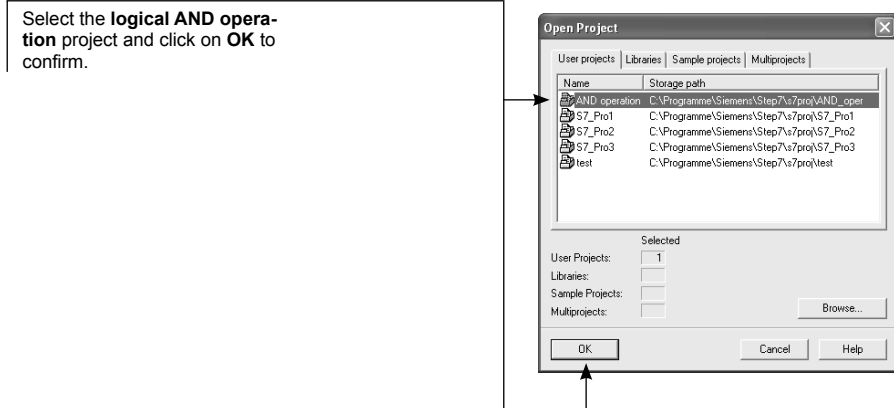
### A Function (FC)

A function is a code block without a memory. It can, however, transfer parameters, which makes it particularly good for recurring functions.

### 5.2.1 Entering FCs (FC 1)

In the SIMATIC Manager, open the logical AND operation project using menu item: **File > Open...**

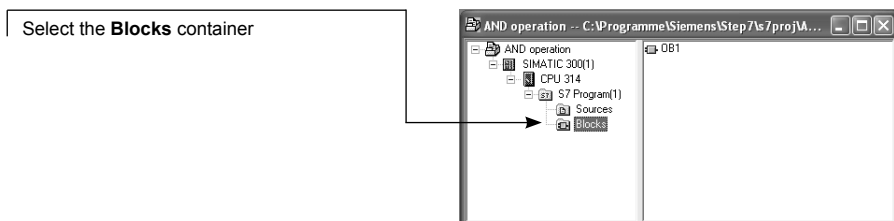
⇒ The following window is displayed:



⇒ The following window is displayed:



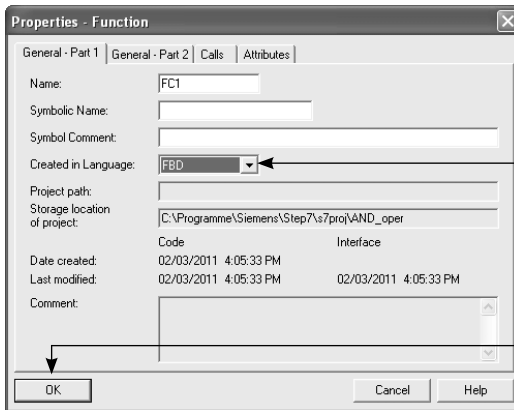
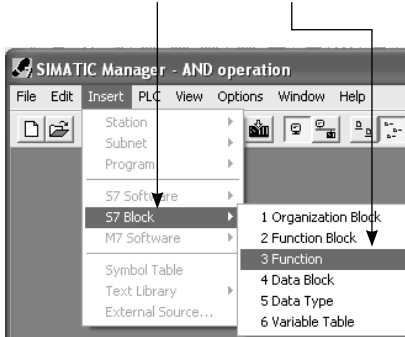
Keep clicking on the «+» signs to get to the bottom level of the subdirectory where the blocks container is installed.



### 5.2.2 S7 block function

Choose menu item:

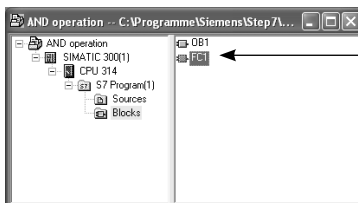
Insert > S7 block > Function



In this dialog box, you choose the type of representation (STL, FBD or LAD): in this case, **FBD**.

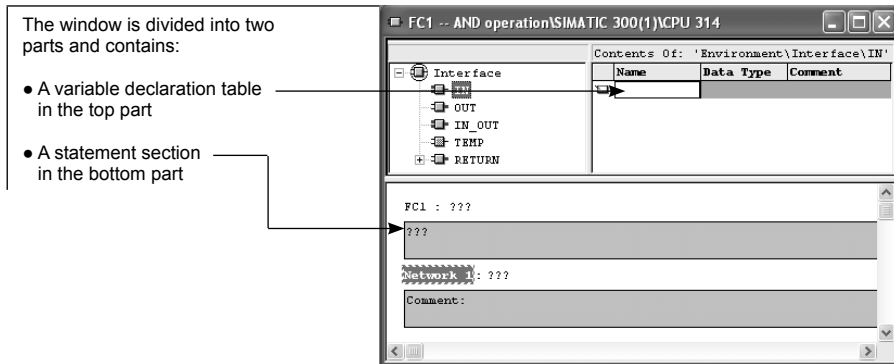
The system displays **FC1** and **OB 1 offline** in the project window, i.e. there is no connection to the CPU yet. Click on **OK** to confirm.

⇒ The following window is displayed:



Double click

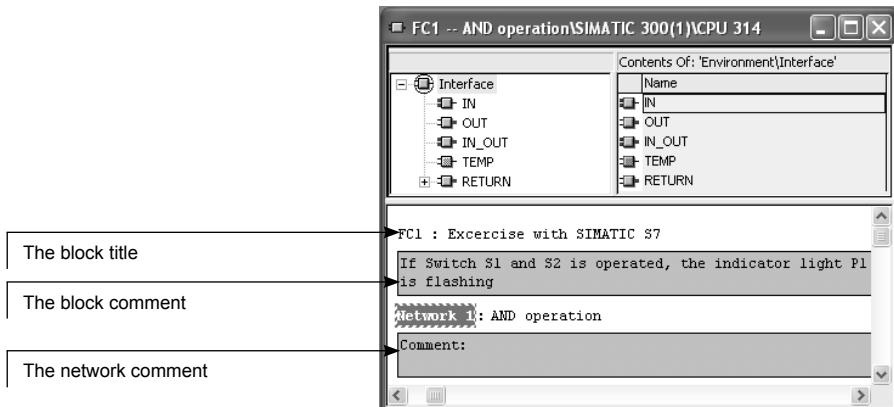
⇒ The following window is displayed:



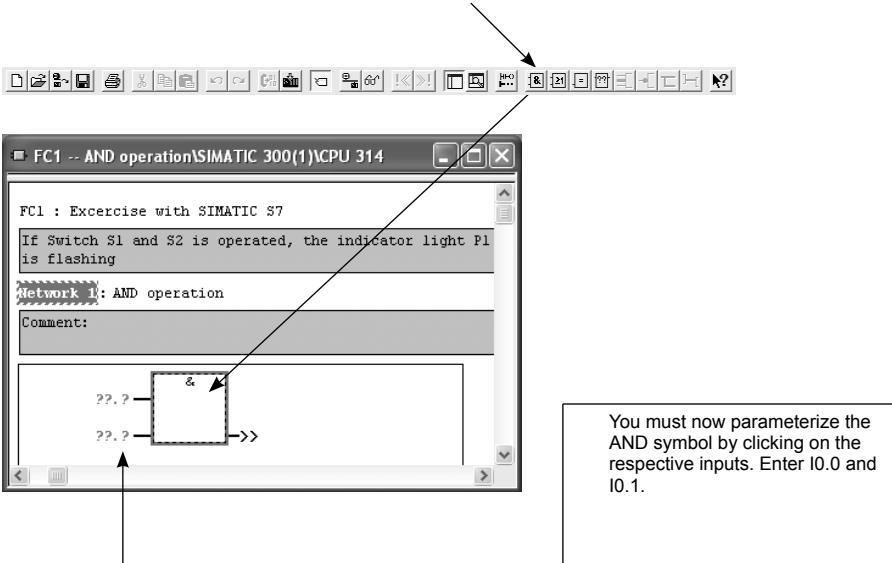
The variable declaration table is not needed, since the user program FC 1 does not contain any local variables (use the mouse to push the variable declaration table upwards out of the way). In the statement section, you enter the program with FBD elements, for example. It is, however, also possible to enter the program in STL or in LAD.

An incremental STL/ LAD/ FBD Editor is integrated in the statement section that carries out a syntax check each time you enter a statement. Possible input errors are indicated in red. You must always correct errors of this type, otherwise there would be no function.

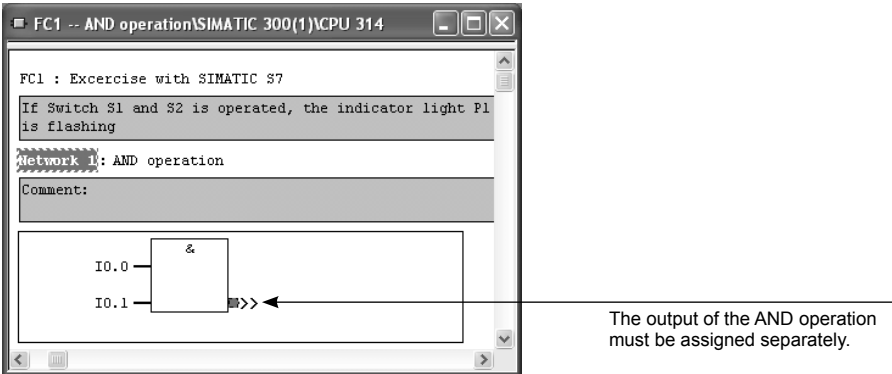
The following texts must be entered in the statement section:



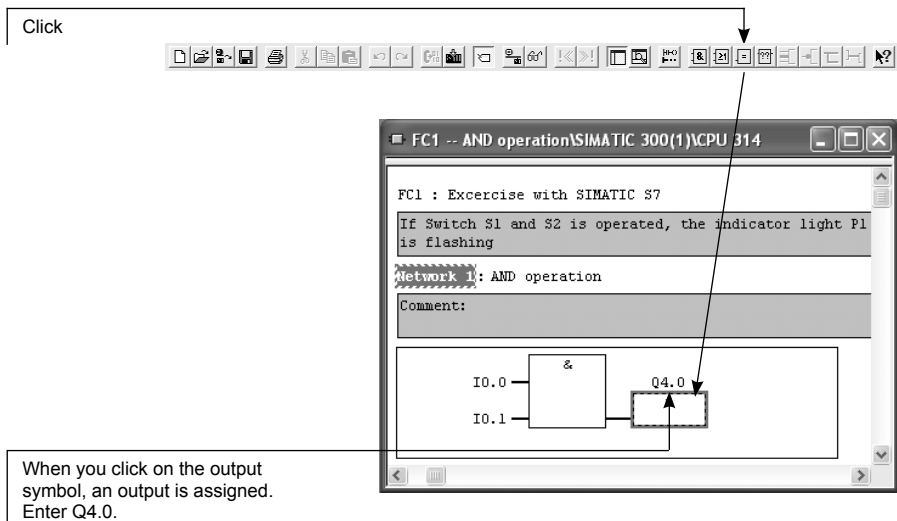
You add the FBD element logical AND operation by clicking on the AND symbol:



You must now parameterize the AND symbol by clicking on the respective inputs. Enter I0.0 and I0.1.



The output of the AND operation must be assigned separately.

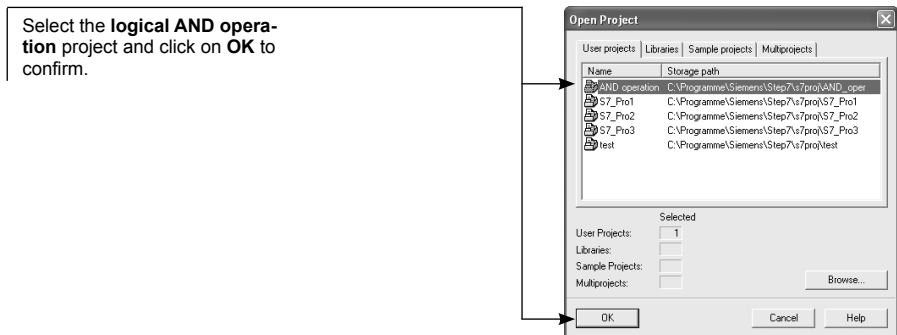


You save the program using menu item: **File > Save**.

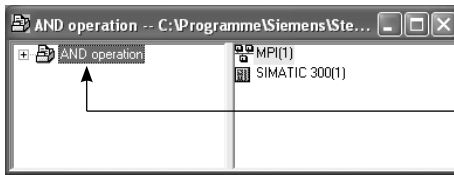
### 5.2.3 Entering OB 1

In the SIMATIC Manager, open the **logical AND operation** project using menu item: **File > Open...**

⇒ The following window is displayed:

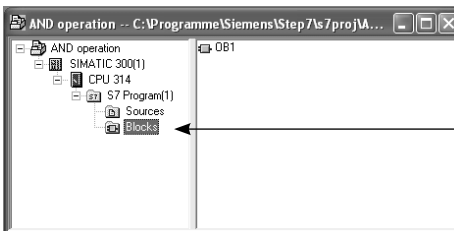


⇒ The following window is displayed:

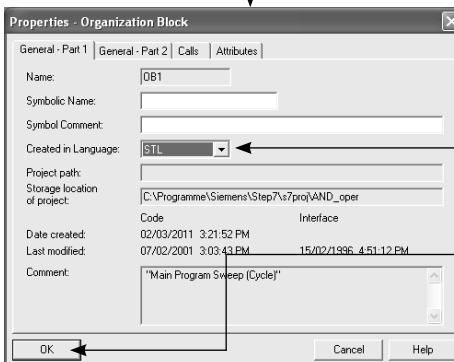
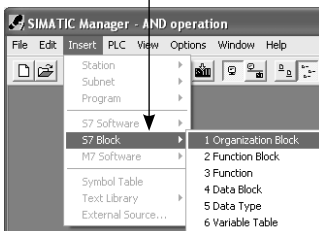


Open the **logical AND operation** project by double clicking on the project.

Keep clicking on the «+» signs to get to the bottom level of the subdirectory where the **Blocks** container is installed.



Select menu item **S7 block** and use menu item **Insert > S7 block > Organization Block** to insert an OB 1.

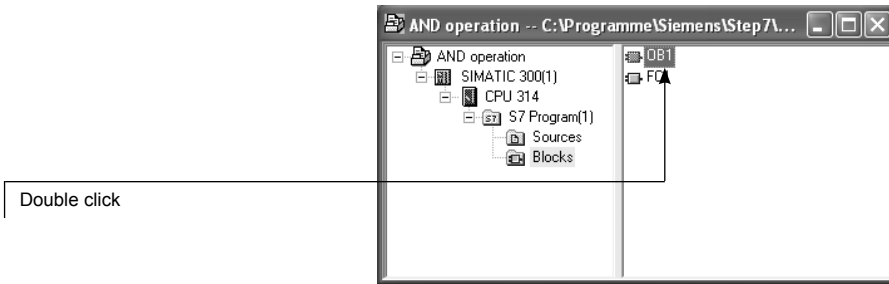


In this dialog box, you choose the type of representation (STL, FBD or LAD): in this case, **STL**. Jump instructions can only be programmed in STL.

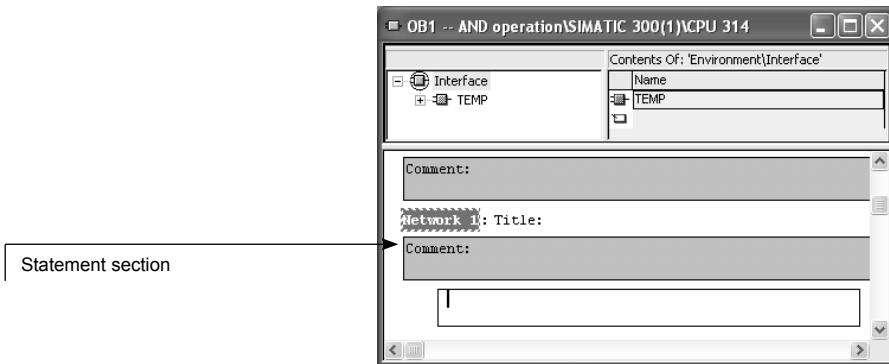
The system displays OB 1 **offline** in the project window, i.e. there is no connection to the CPU yet. Click on **OK** to confirm.



⇒ The following window is displayed:

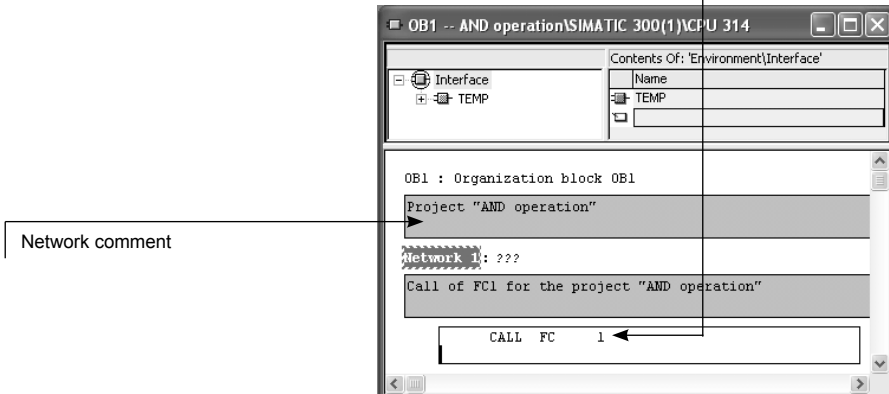


⇒ The following window is displayed:



Now, you write the following absolute jump instruction in network 1:

CALL FC 1



You save OB 1 using menu item: **File > Save**.

### 5.2.4 Downloading

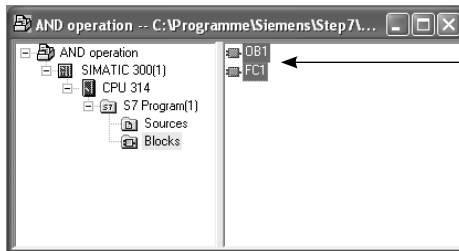
To test the logical AND operation program, it must first be downloaded into the CPU of the S7-300 programmable controller. It is possible to download individual blocks as well as complete user programs into the S7-300. To be able to download the user program into the S7-300 programmable controller, the following requirements must be fulfilled:

There must be an online connection between the PC and the programmable controller.

The program that you want to load must be error-free.

The CPU S7-300 must be in the STOP status.

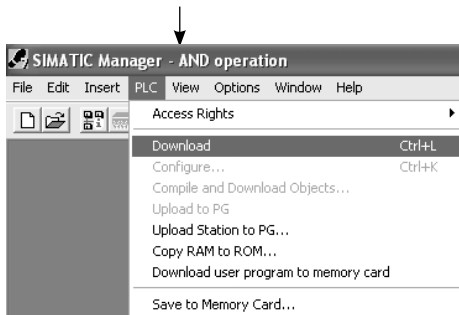
In the SIMATIC Manager (**offline** view), you now call the **blocks** subdirectory.



With the **Shift** key pressed down, click on **FC1** and **OB1**. Both blocks are highlighted.

Then choose menu item:

**PLC > Download...**



Both blocks FC 1 and OB 1 are downloaded into the programmable controller.

This means that the logical AND operation project is loaded in the programmable controller.

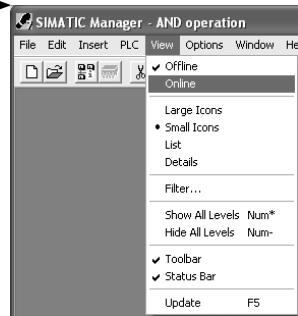
### 5.2.5 Testing

To check the functionality of a user program, you must test it. To do this, you must go through the following sequence:

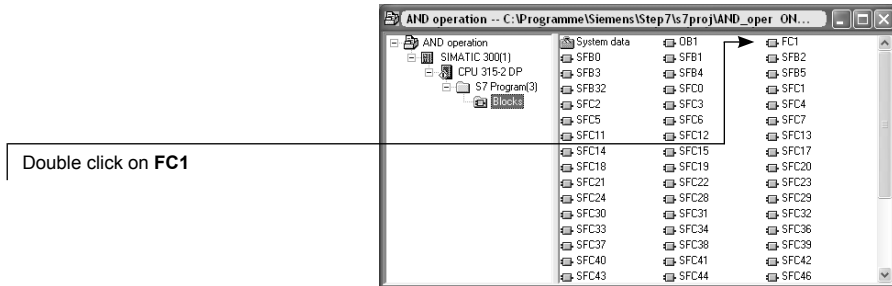
- Open the block online
- Specify the settings for the test display
- Specify the trigger conditions
- Choose the test environment
- Start or stop the test

In the SIMATIC Manager, choose menu item:  
**View > Online**

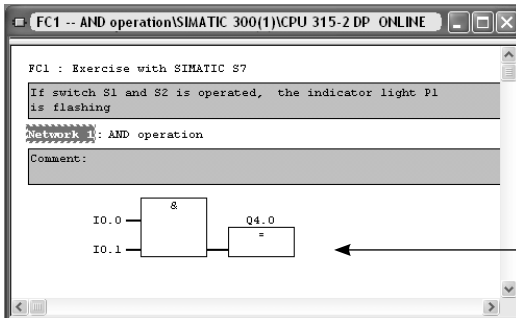
**Important:**  
This is conditional upon the CPU  
is switched to the RUN position  
(**online** connection).



⇒ The following window is displayed:



⇒ The following window is displayed:

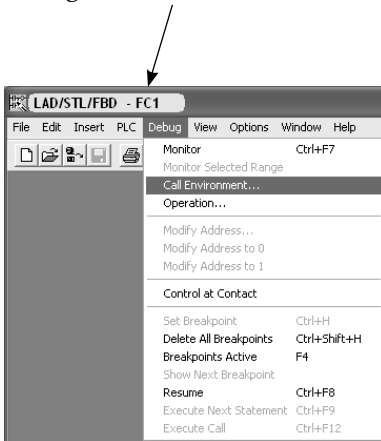


The system now represents the logical AND operation **online**, i.e. there is now a connection from the PC to the programmable controller.

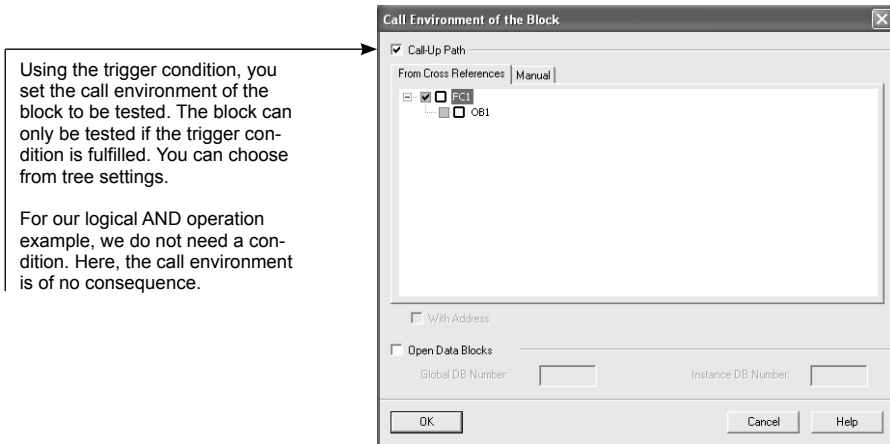
### 5.2.6 Specifying trigger conditions

It is of no consequence whether you carry out testing in STL, LAD or FBD; when doing this, you can always specify the trigger conditions, the test environment or the setting of the test display.

You call the trigger condition using menu item:  
**Debug > Call Environment...**



⇒ The following window is displayed:



You can choose from two test situations:

### 1. Process

Here, the system only determines on the first pass the status of statements looping through.

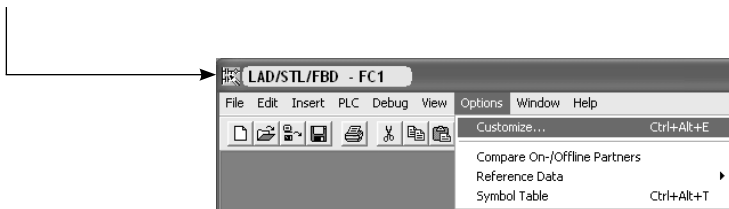
### 2. Laboratory

Here, the system determines on every pass the status of statements looping through.

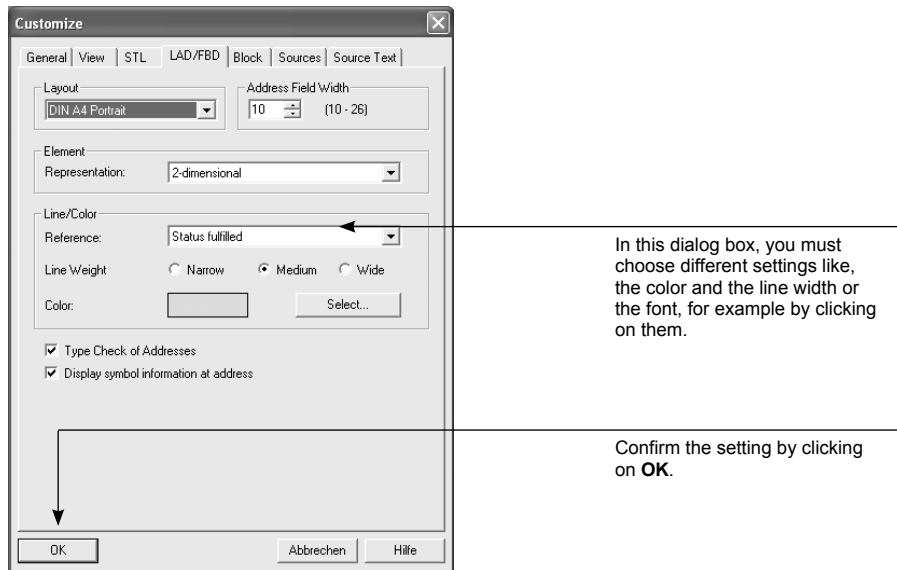
In the present logical AND operation example, we chose the process test situation.

From the menu item, choose:

**Options > Customize...**



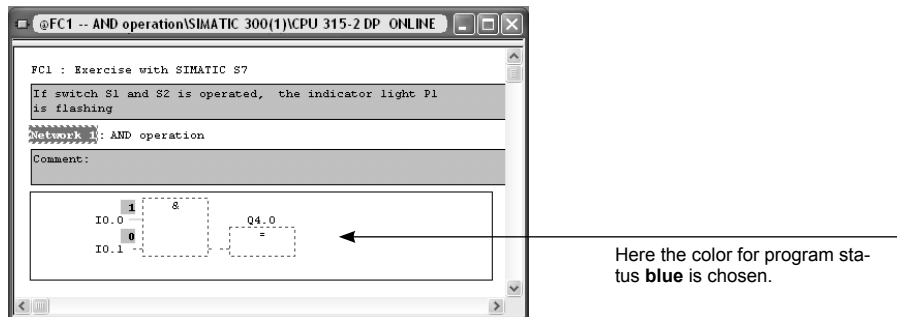
⇒ The following window is displayed:



Choose using menu item:

**Debug > Monitor**

⇒ The following window is displayed:



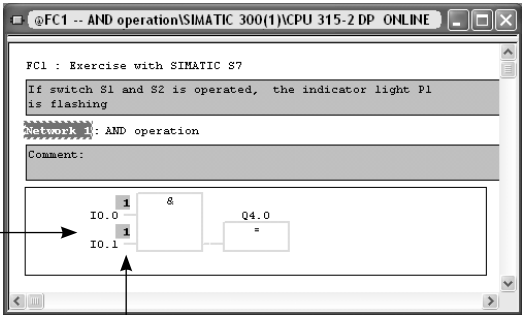
Now, you connect at inputs I0.0 and I0.1 alternating 1 or 0 signals and observe the changing program status.

In the following screenshot, all the inputs I0.0 and I0.1 are carrying a 1 signal. This means that output 4.0 is carrying a 1 signal. The AND condition is fulfilled.

The system only displays the program status for the area that is visible in the editor.

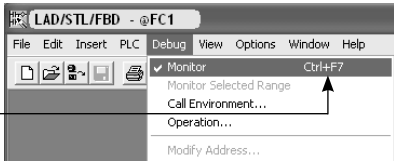
Table 5.2

Function table		
I0.0	I0.1	Q4.0
0	0	0
0	1	0
1	0	0
1	1	1



### 5.2.7 Deactivating the FBD program status

From the menu item, choose:  
**Debug > Monitor** and click on **Monitor**.



You deselect the program status by clicking on **Monitor** or pressing **Ctrl + F7**.

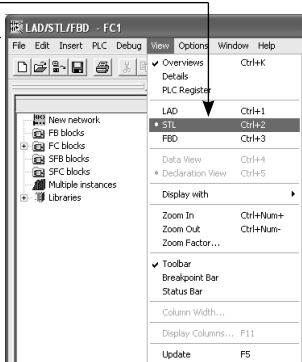
### 5.2.8 Testing with STL

You can also use the STL type of representation to test the logical AND operation program that you entered in FBD.

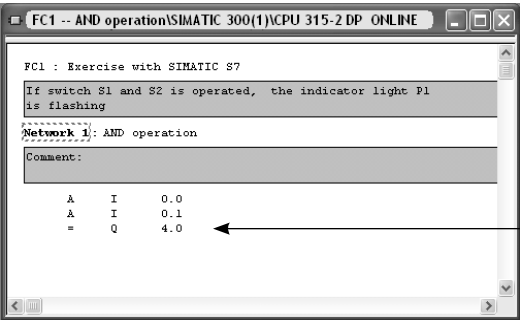
From the menu item, choose:  
**View > STL** the Statement List type of representation

The requirements for displaying the STL in status operation are the same as for FBD:

- the block must be error-free,
- the CPU must be in RUN mode,
- the block must be opened in online mode.



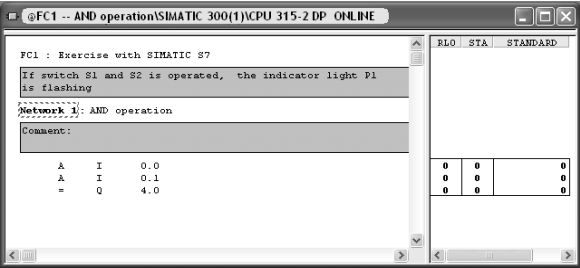
⇒ The following window is displayed:



The logical AND operation program is displayed in STL.

Choose menu item:  
**Debug > Monitor**

⇒ The following window is displayed:



The system displays the STL status in table form. This table shows the status bit, the RLO and the default status.

Now, you connect at inputs I0.0 and I0.1 alternating 1 or 0 signals and observe the changing program status.

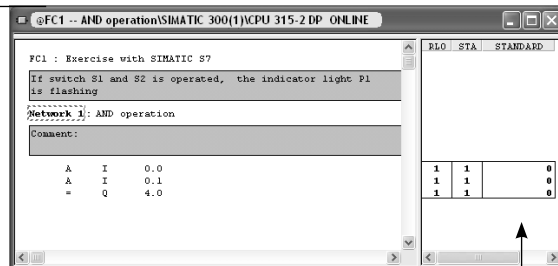
In the following screenshot, all the inputs I0.0 and I0.1 are carrying a 1 signal. This means that output Q4.0 carries a 1 signal. The AND condition is fulfilled.



The system only displays the program status for the area that is visible in the editor.

Table 5.3

Function table		
I0.0	I0.1	Q4.0
0	0	0
0	1	0
1	0	0
1	1	1



You quit the program status using menu item:  
**Debug > Monitor** and click on **Monitor**.

## 5.2.9 Testing with LAD

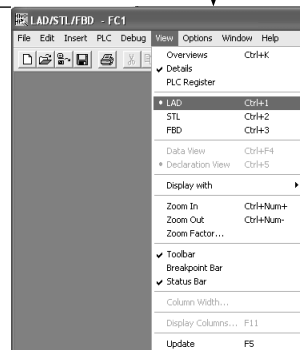
You can also use the LAD type of representation to test the logical AND operation program that you entered in FBD.

From the menu item, choose:

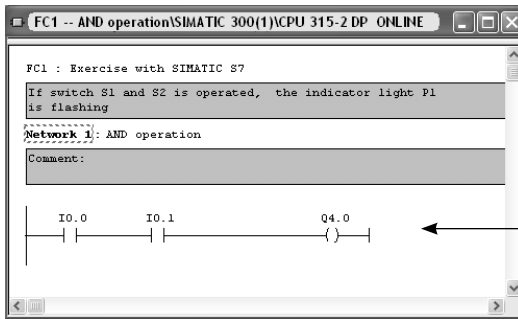
**View > LAD** the Ladder Diagram type of representation

The requirements for displaying the ladder diagram in status operation are the same as for FBD

- the block must be error-free,
- the CPU must be in RUN mode,
- the block must be opened in online mode.



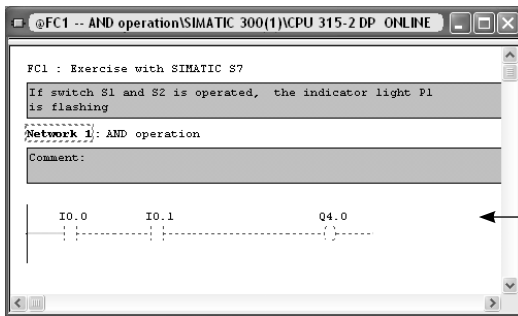
⇒ The following window is displayed:



The logical AND operation program is displayed in LAD.

From the menu item, choose:  
**Debug > Monitor**

⇒ The following window is displayed:



The system displays the program status as LAD for the area that is visible in the editor.

Now, you connect at inputs I0.0 and I0.1 alternating 1 or 0 signals and observe the changing program status.

In the following screenshot, all the inputs I0.0 and I0.1 are carrying a 1 signal. This means that output 4.0 is carrying a 1 signal. The AND condition is fulfilled.

The system only displays the program status for the area that is visible in the editor.

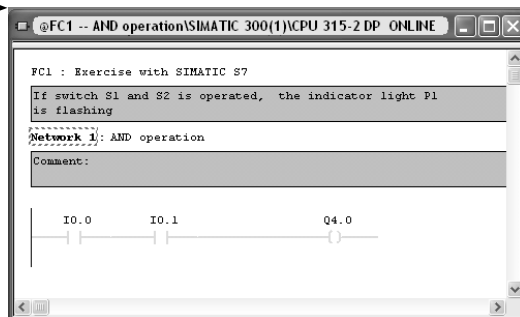


Table 5.4

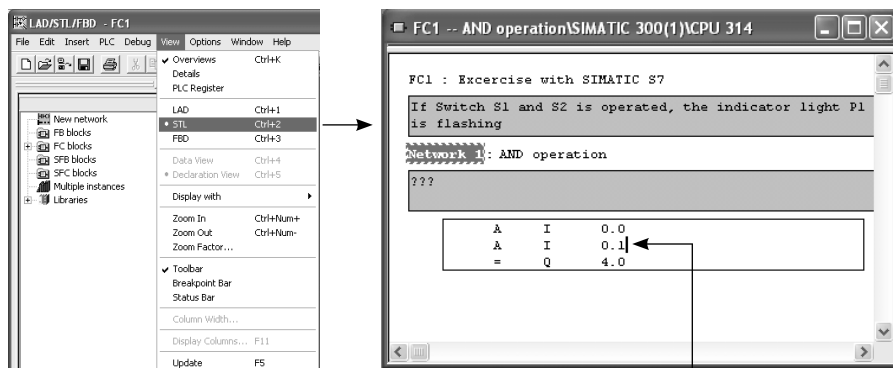
Function table		
E0.0	E0.1	A4.0
0	0	0
0	1	0
1	0	0
1	1	1

You quit the program status using menu item:  
Debug > Monitor and click on Monitor.

## 5.2.10 Extending from two to three inputs (3rd. input IO.2)

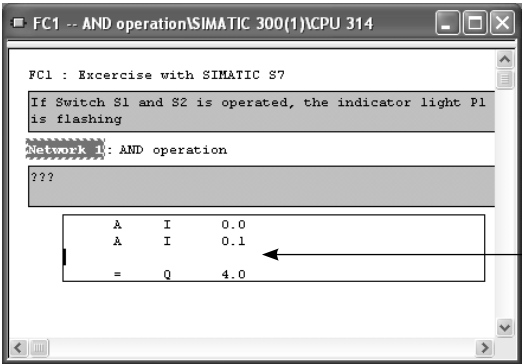
### 5.2.10.1 Extending with STL

Choose the following window:



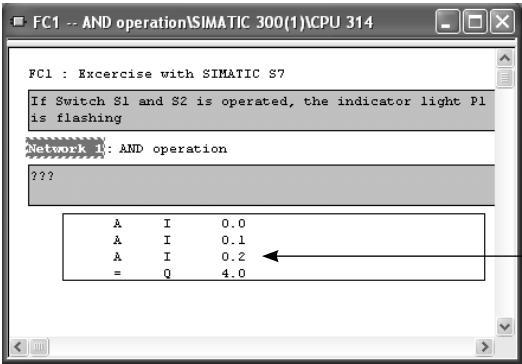
With the cursor in the following position, press the **Return** key.

⇒ The following window is displayed:



In the space you make by pressing the **Return** key, insert statement: A I 0.2.

The window after inserting statement «I0.2»:



The AND operation has now been extended from 2 inputs to 3.  
Switch the CPU to the **STOP status**.

Choose menu item:  
**PLC > Download...**  
Download AND operation with 3 inputs into the CPU.

Switch the CPU to the **RUN status** and test the program.

Window in test mode (STL):

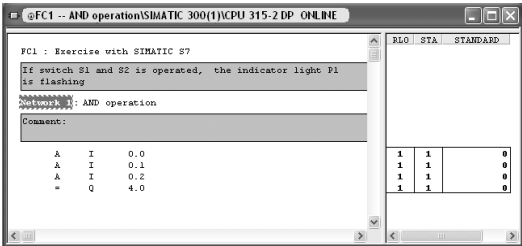


Table 5.5

Function table			
I0.0	I0.1	I0.2	Q4.0
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

You quit the program status using menu item:  
**Debug > Monitor** and click on **Monitor**.

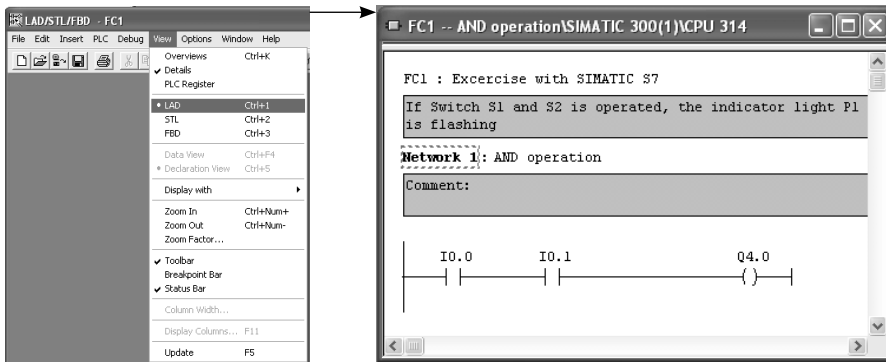
### 5.2.10.2 Extending with LAD

Choose the Ladder Diagram type of representation

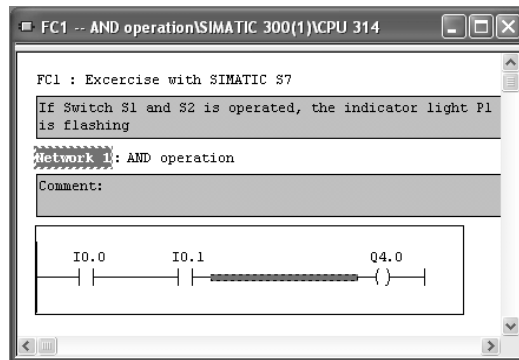
Menu item:

**View > LAD**

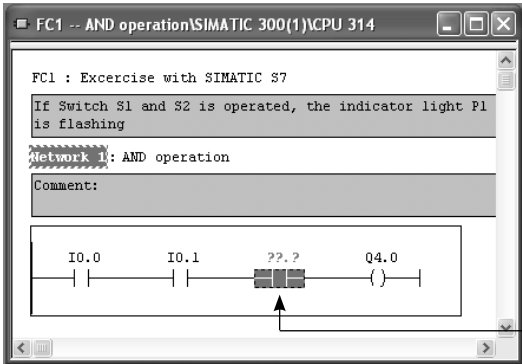
⇒ The following window is displayed:



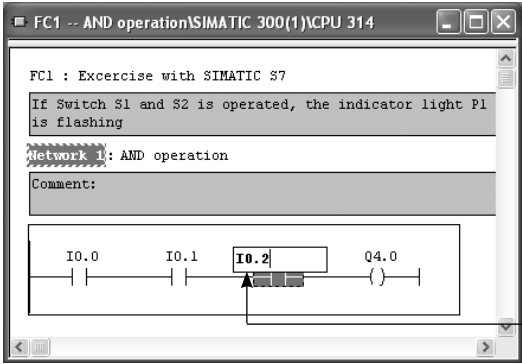
Click on the following grid:



Here, you click on the  
**NO contact** symbol.



The system inserts a **NO** contact into the AND operation.



The added **NO** contact is parameterized with I0.2.

You then switch the CPU to the **STOP** status and load and test (**RUN** status) the program.

⇒ Window in test mode (LAD):

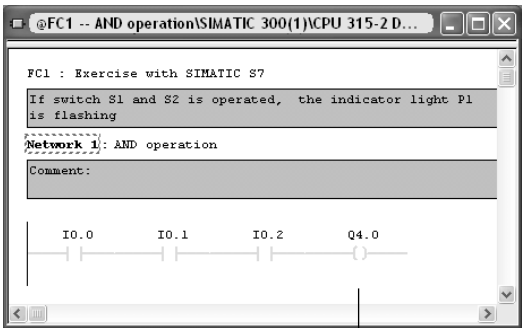


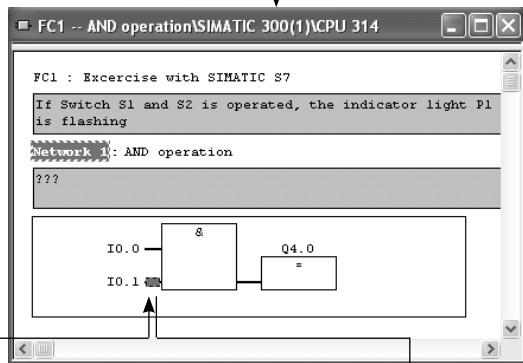
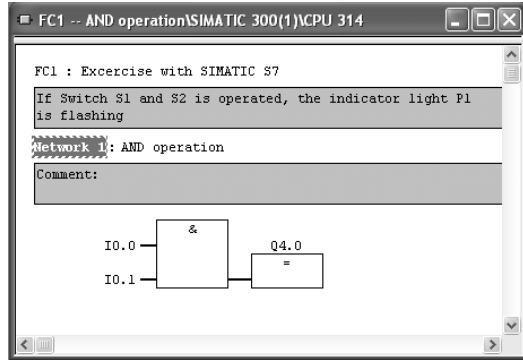
Table 5.6

Function table			
I0.0	I0.1	I0.2	Q4.0
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

You quit the program status using menu item:  
**Debug > Monitor** and click on **Monitor**.

### 5.2.10.3 Extending with FBD

⇒ Choose the following window:

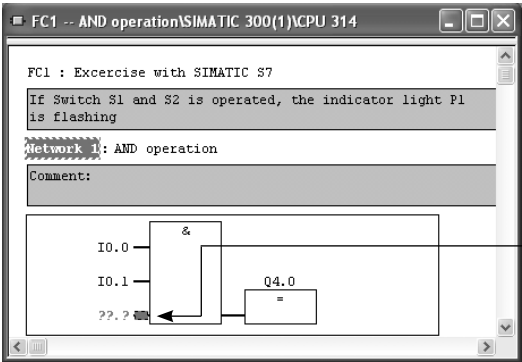


Select input I0.1 with the mouse.

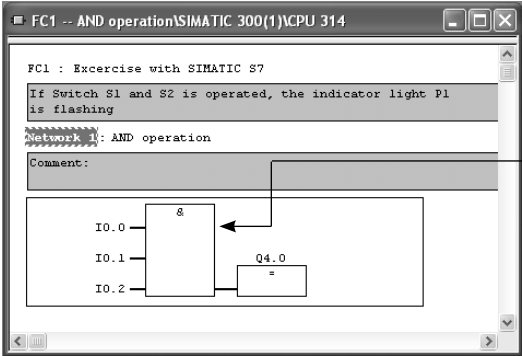


Click on it with the mouse

⇒ The following window is displayed:



The added input I0.2 is parameterized.



The AND operation has now been extended from 2 inputs to 3.

Switch the CPU to the **STOP status**.

Choose menu item: **PLC > Download...**

Download the program into the CPU, test the program.

Window in test mode (FBD):

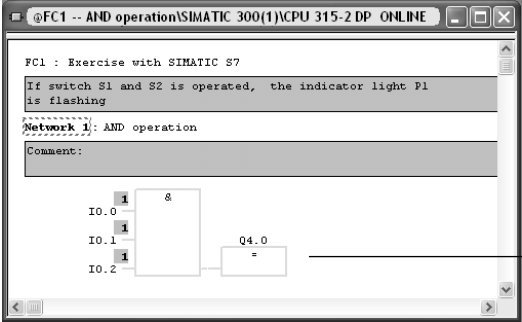


Table 5.7

Function table			
I0.0	I0.1	I0.2	Q4.0
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1



You quit the program status using menu item **Debug > Monitor** and click on **Monitor**.

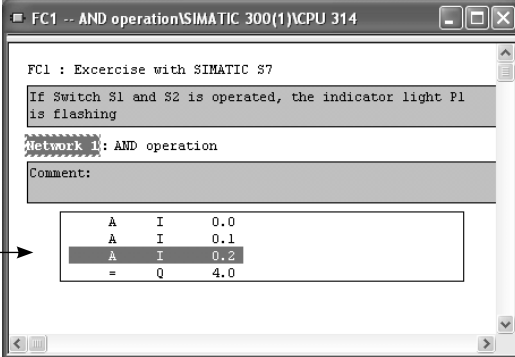
### 5.2.11 Reducing from 3 inputs to 2 (delete I0.2)

#### 5.2.11.1 Reducing with STL

This restores the initial status.

⇒ The initial status is the following window:

Use the mouse to highlight the row to be deleted, press the «Del» key, the row is removed.



A	I	0.0
A	I	0.1
A	I	0.2

= Q 4.0

The AND operation with 2 inputs is restored.

Switch the CPU to the **STOP** status and choose menu item:  
**PLC > Download...**

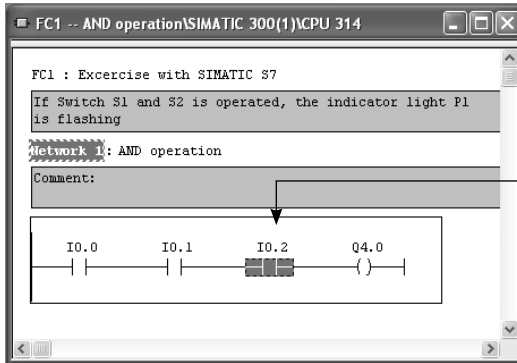
After this, download the AND operation with 2 inputs into the CPU and switch the CPU to the RUN status.

Then test the program.

### 5.2.11.2 Reducing with LAD

This restores the initial status.

⇒ The initial status is the following window:



Use the mouse to highlight the AND symbol to be deleted, press the «Del» key, the NO contact is removed.

The AND operation with 2 inputs is restored.

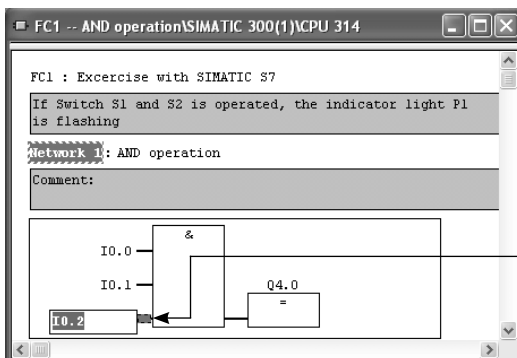
Switch the CPU to the **STOP** status and choose menu item:  
**PLC > Download...**

After this, download the AND operation with 2 inputs into the CPU and switch the CPU to the RUN status.

Then test the program.

### 5.2.11.3 Reducing with FBD

⇒ The initial status is the following window:



Use the mouse to highlight input I0.2, press the «Del» key, input I0.2 is removed.

The AND operation with 2 inputs is restored.

Switch the CPU to the **STOP** status and choose menu item:

**PLC > Download...**

Then, download the AND operation with 2 inputs into the CPU and switch the CPU to the RUN status.

Finally, test the program.

### 5.3 Logical OR operation user program

#### 5.3.1 Entering the program using the PC (FBD)

Table 5.8

Assignment list		
Symbol	Operand	Comment
S3	I0.2	«NO contact» switch
S4	I0.3	«NO contact» switch
P2	Q4.1	Indicator light

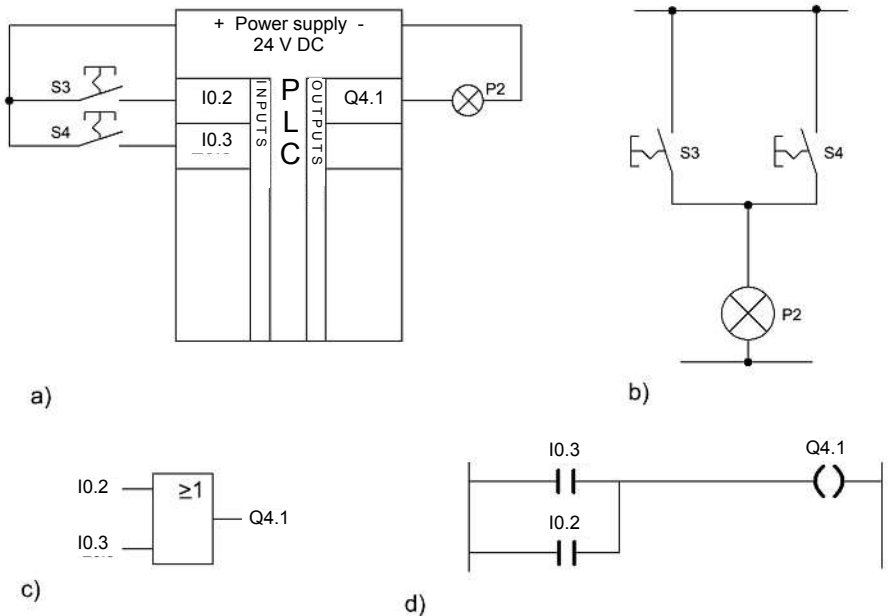


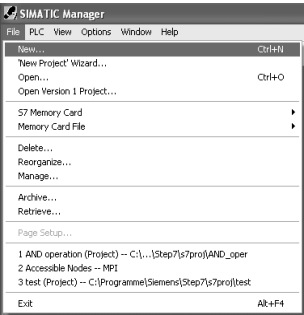
Figure 5.2 Entering the logical OR operation: a) Connection to the PLC, b) Schematic diagram, c) FBD, d) LAD

Before you enter the «OR operation» program into the SIMATIC S7-300, you must create a **project** (see also Figure 5.2). You do this in the same way as with the AND operation.

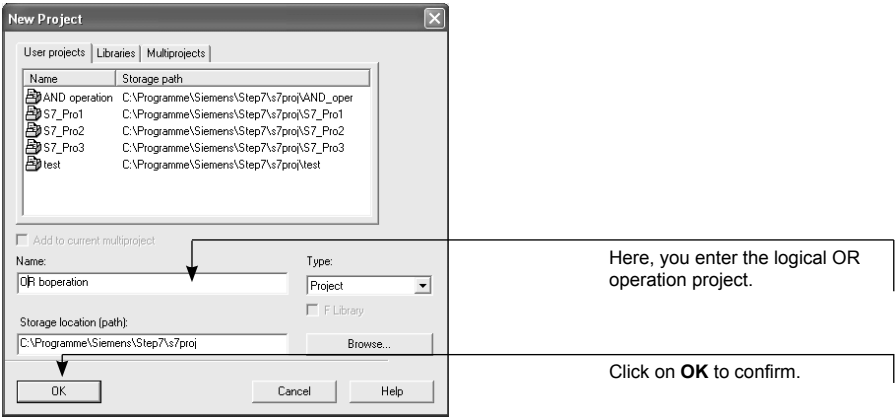
Do not carry out a memory reset on the CPU; otherwise, the logical «AND operation» program will be lost. For **all the following exercises**, we will be using the SIMATIC S7-300 station, just like in the first exercise (logical AND operation program). In the case of a memory reset, our station 1 would also be lost. The following steps explain programming of the logical «OR operation» program.

### 5.3.2 Create a project

In the SIMATIC Manager click on menu item:  
**File > New...**



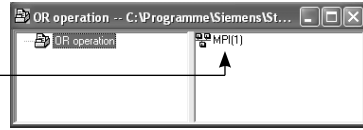
⇒ The following window is displayed:



The logical OR operation project has now been created.

⇒ The following window is displayed.

You must now insert the SIMATIC station into this window. Since we are working with the same SIMATIC station as in the previous project, you do not need to create a new station. You copy the SIMATIC station from the **logical AND operation** project to the **logical OR operation** project.



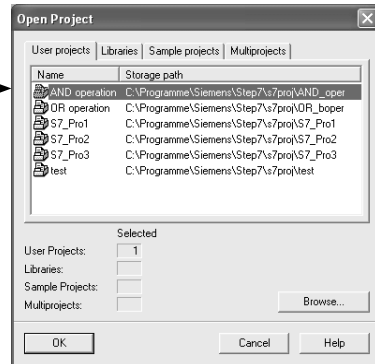
### 5.3.3 Copy SIMATIC station 1 into another project

Use menu item:

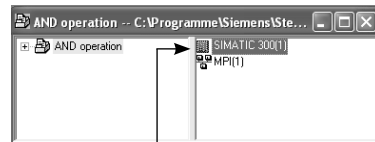
**File > Open...** to open the «logical AND operation» project

⇒ The following window is displayed:

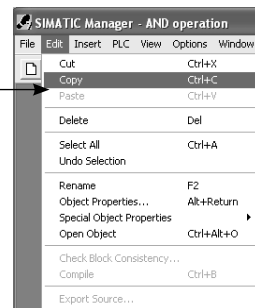
Doubleclick on the **AND operation**



⇒ The following window is displayed:



Use the mouse to select the SIMATIC station (1), click on menu item: **Edit > Copy**, the station (1) is copied to the clipboard.

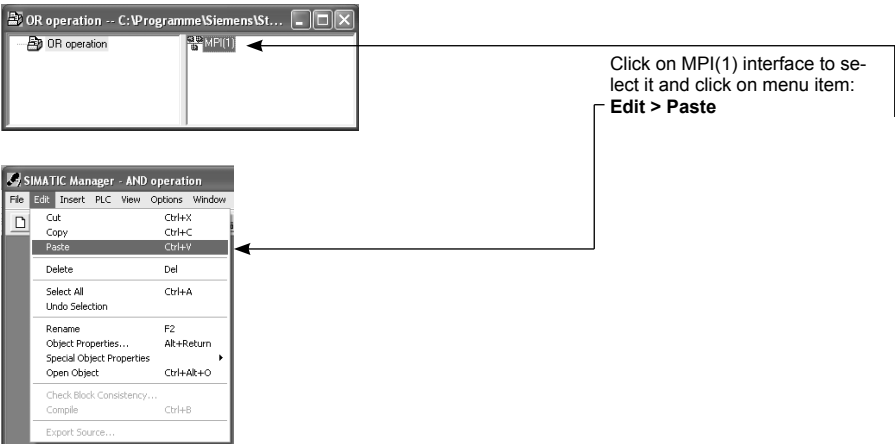


After this, click on menu items:  
**File > Close**

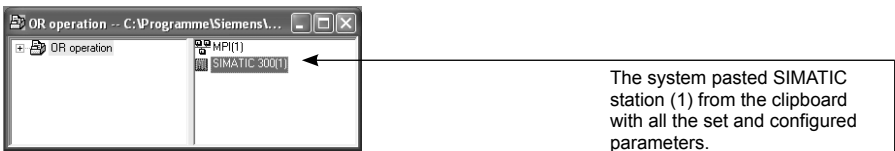
⇒ The following window is displayed:



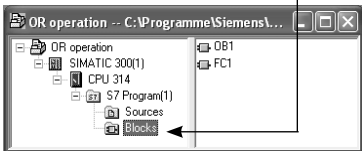
⇒ The following window is displayed:



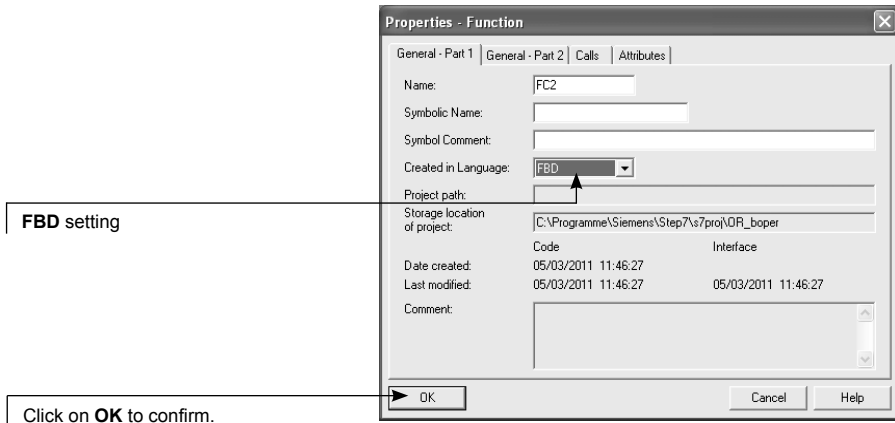
⇒ The following window is displayed:



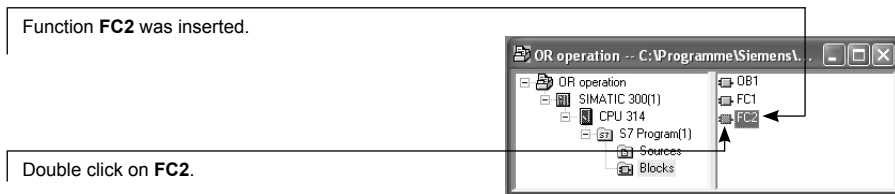
Now click on the «+» signs down to the lowest level of the directory.  
At the lowest level, the **Blocks** container is installed.



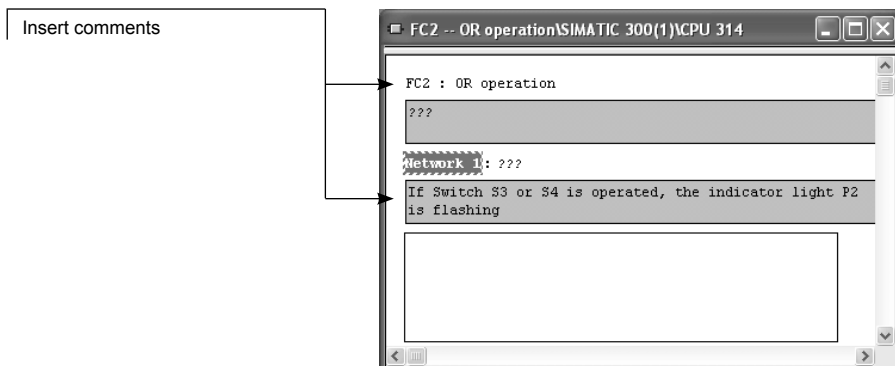
Select the **Block** container and use menu item:  
**Insert > S7 block > Function** to insert a Function FC 2.



⇒ The following window is displayed:



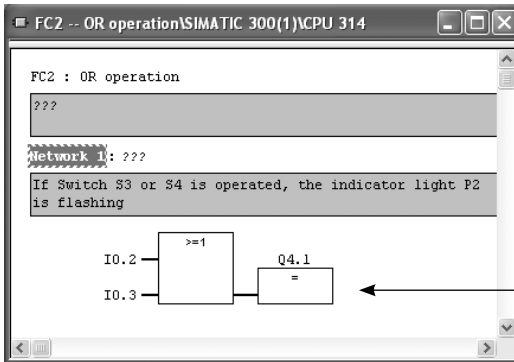
⇒ The following window is displayed:





Clicking on the **OR** box inserts the OR symbol.

⇒ The following window is displayed:

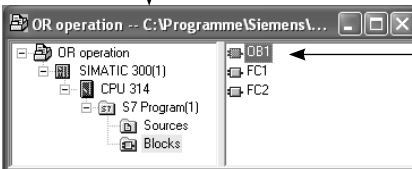


Parameterize the inputs of the OR operation with I0.2, I0.3 and Q4.1.

You save the program using menu item: **File > Save**

### 5.3.4 Change OB 1

Use the SIMATIC Manager to open the following window:



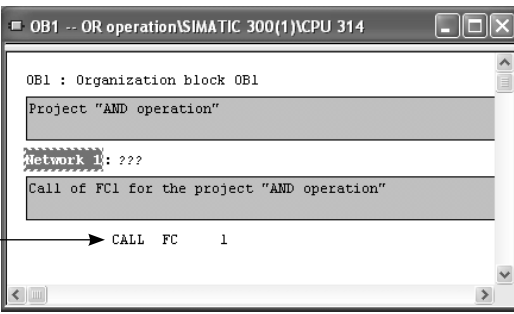
Doubleclick on the **OB1** symbol



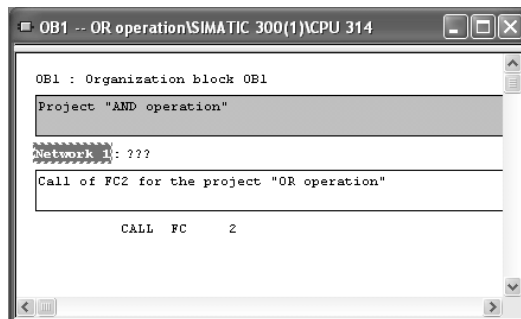
⇒ The following window is displayed:

Change the texts for the OR operation appropriately.

Change the command line:  
CALL FC 1 to **CALL FC 2** and  
change the command texts to an  
appropriate one and confirm by  
pressing **Return**.



⇒ The following window is displayed:



Now save the changed OB 1 under menu item: **File > Save**

OB 1 is now available for the logical OR operation project.

### 5.3.5 Downloading

To test the logical OR operation program, it must first be downloaded into the CPU of the S7-300 programmable controller.

It is possible to download individual blocks as well as complete user programs into the S7-300.

To be able to download the user program into the S7-300 programmable controller, the following requirements must be fulfilled:

- ☐ There must be an online connection between the PC and the programmable controller.
- ☐ The CPU S7-300 must be in the STOP status.

In the SIMATIC Manager (**offline view**), you then call the **Blocks** subdirectory.



Choose menu item:

**PLC > Download...**

The logical OR operation project is now present in the programmable controller.

### 5.3.6 Testing

To check the functionality of a user program, you must test it.

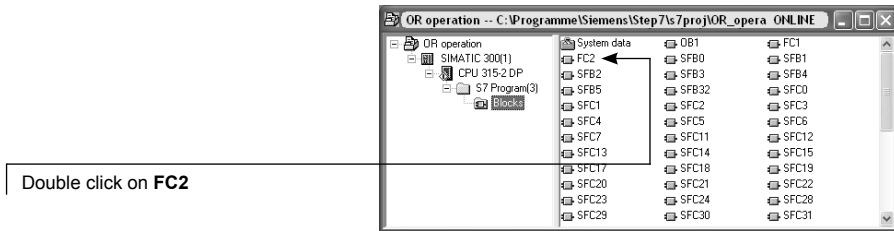
You must keep to the following sequence:

- ☐ Open the block online
- ☐ Specify the settings for the test display
- ☐ Specify the trigger conditions
- ☐ Choose the test environment
- ☐ Start or stop the test

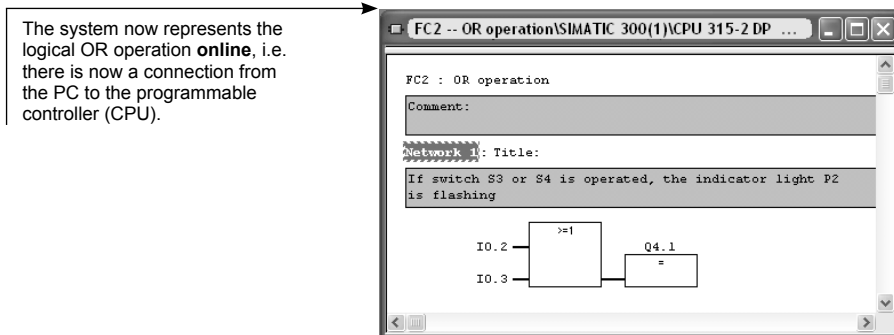
In the SIMATIC Manager, choose menu item:

**View > Online**

⇒ The following window is displayed:



⇒ The following window is displayed:



Now, use menu item:

**Debug > Call Environment...** to call the trigger condition, and choose:  
No condition

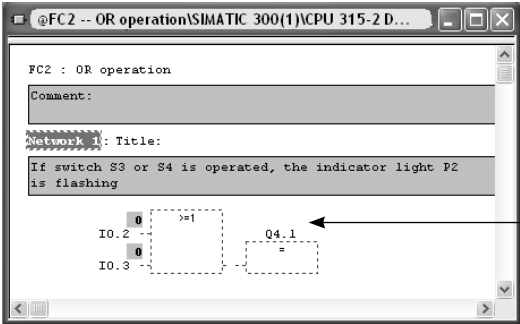
and the desired settings from menu item:

**Options > Customize...**

then from menu item:

**Debug > Monitor** or function keys **Ctrl + F7**

⇒ The following window is displayed:



Now, you connect at inputs I0.2 and I0.3 alternating 1 or 0 signals and observe the changing program status.  
Window:  
**FBD** type of representation

Now you can test the program in LAD and STL

⇒ Window: LAD

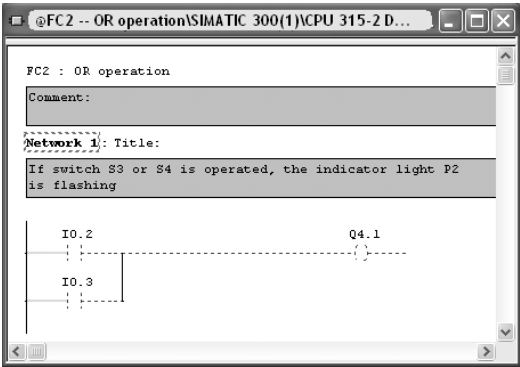
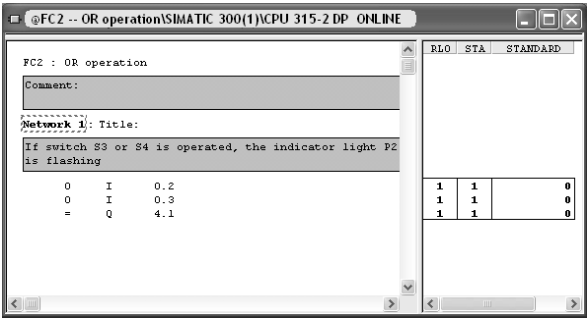


Table 5.9

Function table		
I0.2	I0.3	Q4.1
0	0	0
0	1	1
1	0	1
1	1	1

⇒ Window: STL



You must check the results of logic operations in accordance with the function table.

## 6 Program Input

### 6.1 AND before OR

The logical AND before OR operation corresponds to parallel switching of several series-switched contacts in the wiring diagram. With these subbranches that are composed of series and parallel connections, output Q4.2 carries a signal state of 1 if in at least one subbranch all the series-connected contacts are closed (have a signal state of 1).

AND before OR operations are programmed in the STL method of representation without brackets; however, the parallel-switched subbranches must be separated from one another by entering an O (OR function). In this connection, the AND functions are processed first and the system forms from these results the result of the OR function. The first AND function (I0.0, I0.1) is separated from the second AND function (I0.2, I0.3) by the individual O (OR function).

**AND operations have priority and are therefore always processed before OR operations.**

Table 6.1

Assignment list		
Symbol	Operand	Comment
S1	I0.0	«NO contact» switch
S2	I0.1	«NO contact» switch
S3	I0.2	«NO contact» switch
S4	I0.3	«NO contact» switch
P1	Q4.2	Indicator light

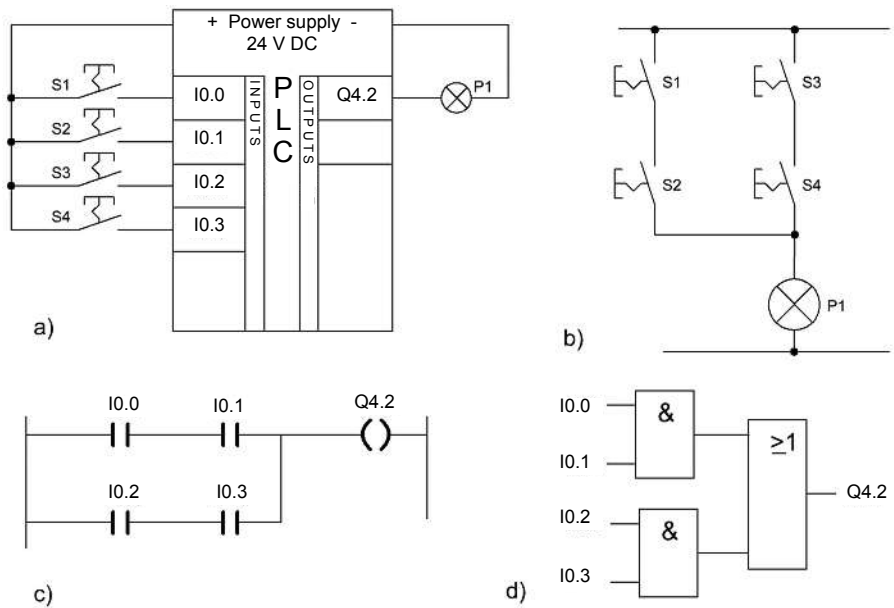


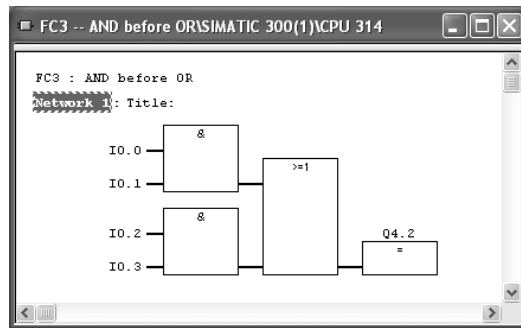
Figure 6.1 Entering the logical AND operation: a) Connection to the PLC, b) Schematic diagram, c) LAD, d) FBD

Before you enter the AND before OR program into the SIMATIC S7-300, you must create a project (see Figure 6.1); in this case project **AND before OR**. In this connection, you must not carry out a memory reset of the CPU, otherwise all the programs would be lost. For **all the following exercises**, we will be using the SIMATIC S7-300 station, just like in the first exercise (logical AND operation program). In the case of a memory reset, our station 1 would also be lost.

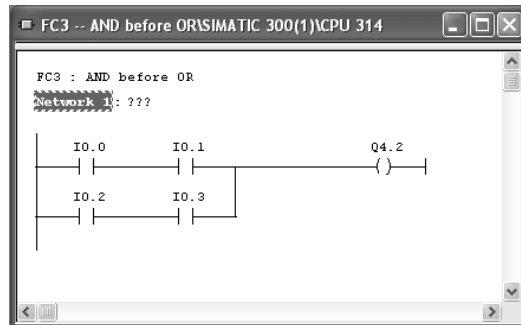
You must keep to the following steps when programming the AND before OR program:

- ☐ Create project AND before OR
- ☐ Create block FC 3
- ☐ Program AND before OR in FBD
- ☐ Change OB 1
- ☐ Download FC 3 and OB 1 into the CPU
- ☐ Test the program

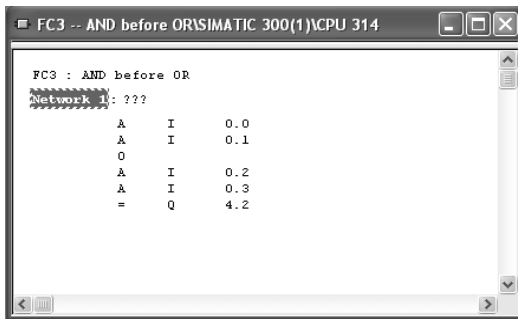
The following window shows the **AND before OR** operation in FBD:



Method of representation: LAD



Method of representation: STL



Test the program.

## 6.2 OR before AND

The logical OR before AND operation corresponds to series switching of several parallel-switched contacts in the wiring diagram. In the case of this connection composed of series and parallel connections, output Q4.3 only has a signal state of 1 if in both parallel subbranches at least one of the contacts has a signal state of 1.

**For OR operations to have priority over AND operations, they must be joined by brackets.**

Before we enter the OR before AND program into the SIMATIC S7-300, we must create a project (see Figure 6.2).



Table 6.2

Assignment list		
Symbol	Operand	Comment
S1	I1.0	«NO contact» switch
S2	I1.1	«NO contact» switch
S3	I1.2	«NO contact» switch
S4	I1.3	«NO contact» switch
P3	Q4.3	Indicator light

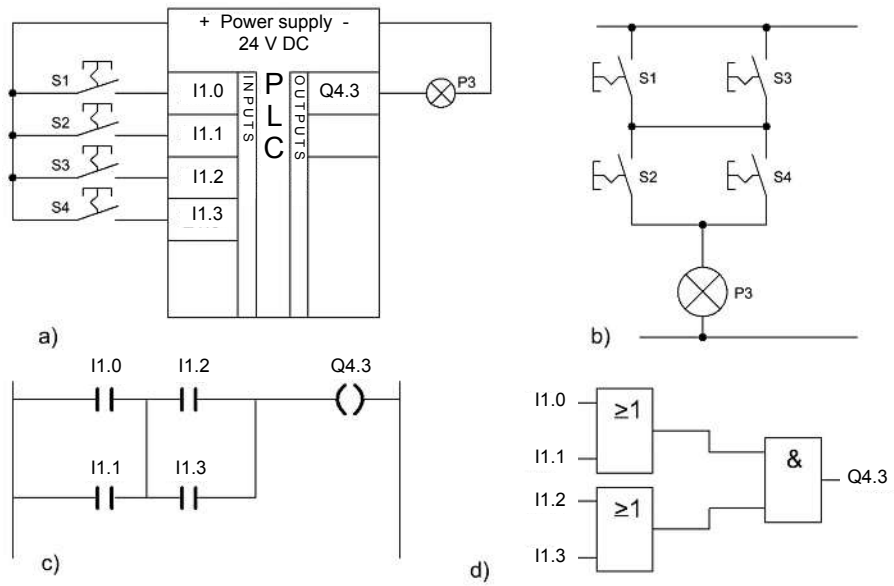


Figure 6.2 Creating the logical AND operation: a) Connection to the PLC, b) Schematic diagram, c) LAD, d) FBD

You create the **OR before AND** project. In this connection, you must not carry out a memory reset of the CPU, otherwise all the programs would be lost.

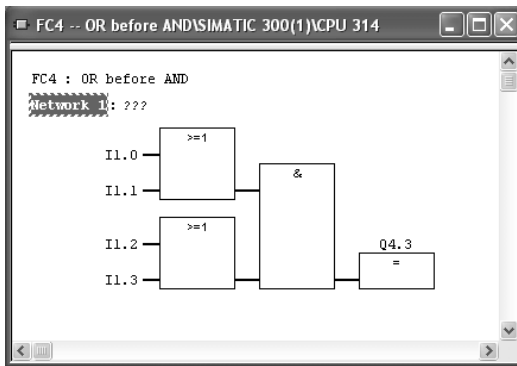
For **all the following exercises**, we will be using the SIMATIC S7-300 station, just like in the first exercise (logical AND operation program). In the case of a memory reset, our station 1 would be lost.

You must keep to the following steps when programming the **OR before AND** program:

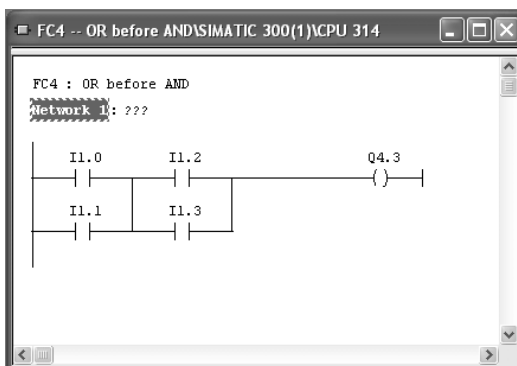
- ☐ Create project OR before AND
- ☐ Create block FC 4
- ☐ Program AND before OR in FBD
- ☐ Change OB 1
- ☐ Download FC 4 and OB 1 into the CPU
- ☐ Test the program

In the following window, the **OR before AND** logic operation is represented in FBD.

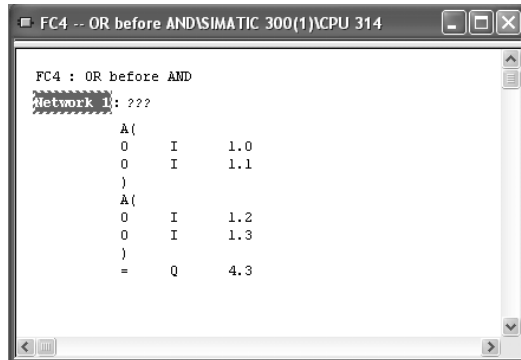
Method of representation: FBD



Method of representation: LAD



Method of representation: STL



Test the program.

### 6.3 Poll for signal state 0

In a circuit with contacts, polling for signal state 0 corresponds to an NC contact.

Before you enter the Poll for signal state 0 program into the SIMATIC S7-300, you must create a **Poll for signal state 0** project (see Figure 6.3).

Again, for **all the following exercises** we will be using the SIMATIC S7-300 station, just like in the first exercise (logical AND operation program). Here too, you must observe the following:

In the case of a memory reset, our station 1 would be lost.

Proceed as follows:

- ☐ Create Poll for signal state 0 project
- ☐ Create block FC 5
- ☐ Program poll for 0 in FBD
- ☐ Change OB 1
- ☐ Download FC 5 and OB 1 into the CPU
- ☐ Test the program

Table 6.3

Assignment list		
Symbol	Operand	Comment
S2	I0.2	Switch «NO contact»
S3	I0.3	Switch «NC contact»
P1	Q4.1	Indicator light

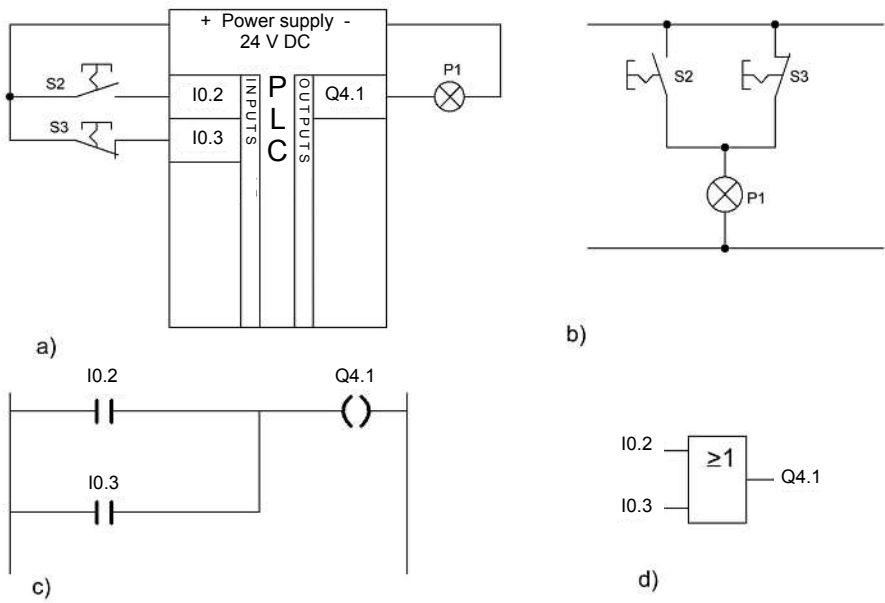
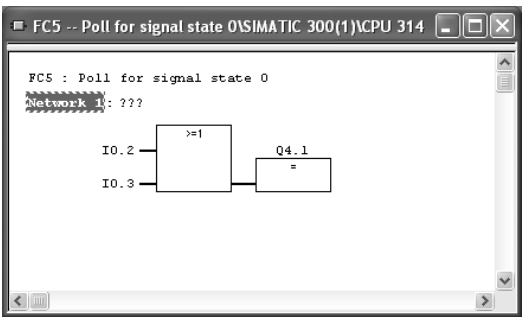


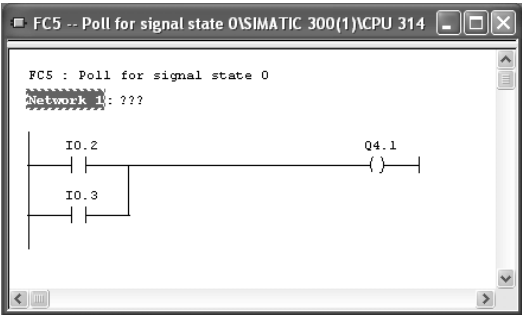
Figure 6.3 Creating the project: a) Connection to the PLC, b) Schematic diagram, c) LAD, d) FBD

The following window shows the Poll for 0 program in FBD.

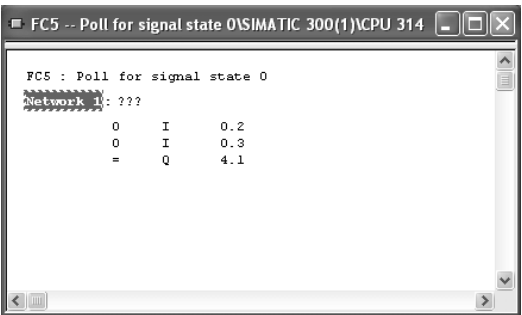
Method of representation: FBD



Method of representation: LAD



Method of representation: LAD



Test the program.

## 6.4 Exclusive OR operation

The circuit shows an Exclusive OR operation (X) in which output Q4.0 is only activated (signal state 1) if one single input has a signal state of 1. In a circuit with contacts, this can only be implemented using NC and NO contacts.

Before you can enter the Exclusive OR operation program into the SIMATIC S7-300, you must create the **Exclusive OR** project (see Figure 6.4). For **all the following exercises** we will be using the SIMATIC S7-300 station, just like in the first exercise (logical AND operation program).

Table 6.4

Assignment list		
Symbol	Operand	Comment
S0	I1.0	Switch «NO contact»
S1	I1.1	Switch «NO contact»
P1	Q4.0	Indicator light

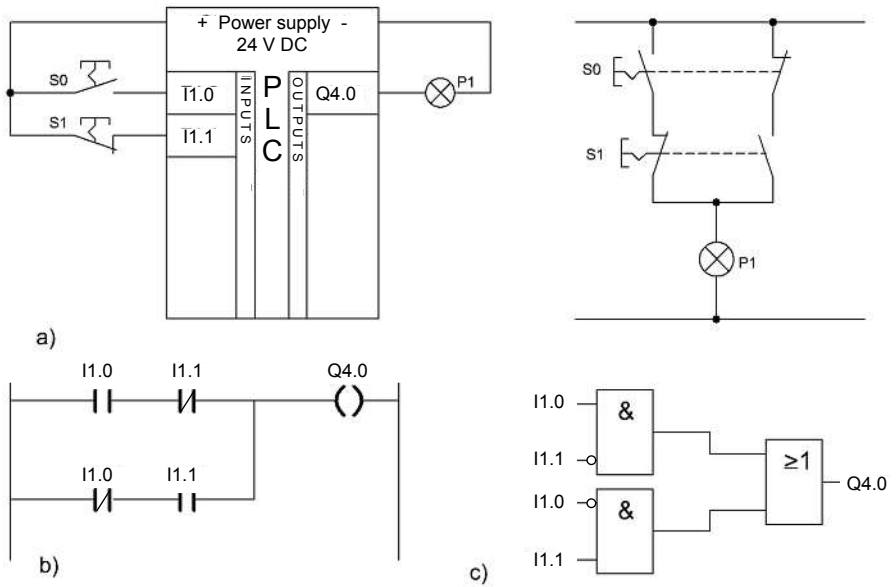


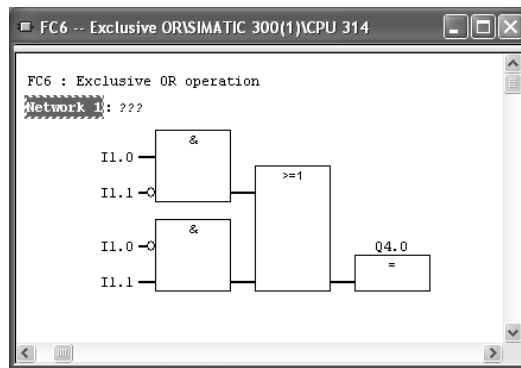
Figure 6.4 Creating the project: a) Connection to the PLC, b) Schematic diagram, c) LAD, d) FBD

Proceed as follows:

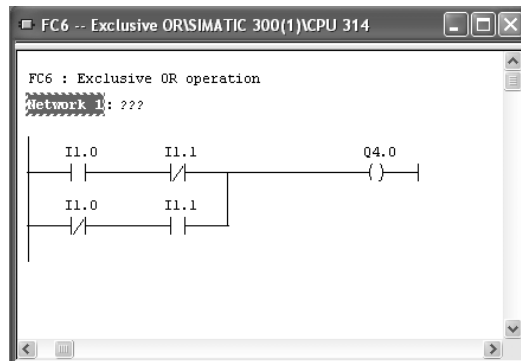
- ☐ Create the Exclusive OR project
- ☐ Create block FC 6
- ☐ Program Exclusive OR in FBD
- ☐ Change OB 1
- ☐ Download FC 6 and OB 1 into the CPU
- ☐ Test the program

The following windows show the Exclusive OR program.

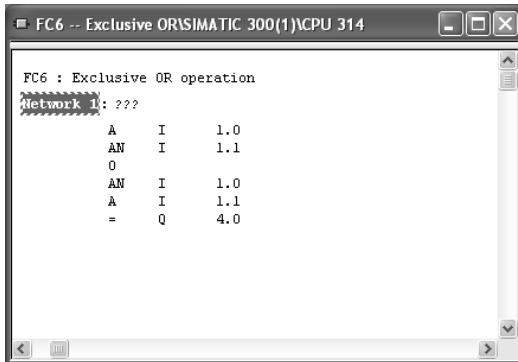
Method of representation: **FBD**



Method of representation: **LAD**



Method of representation: STL



Test the program.

## 6.5 Polling outputs

There are different conditions for activating outputs Q4.0 and Q4.1. In these cases, you must provide each output with its own current path or a separate logic symbol. Since the programmable controller can poll the signal state of inputs as well as of outputs, bit memories, etc., the system polls in the AND operation for output Q4.0 and output Q4.1.

Before you can enter the **Polling outputs** program into the SIMATIC S7-300, you must create the **Polling outputs** project.



Table 6.5

Assignment list		
Symbol	Operand	Comment
S1	I1.0	Switch «NO contact»
S2	I1.1	Switch «NO contact»
S3	I1.2	Switch «NO contact»
P1	Q4.0	Indicator light
P2	Q4.1	Indicator light

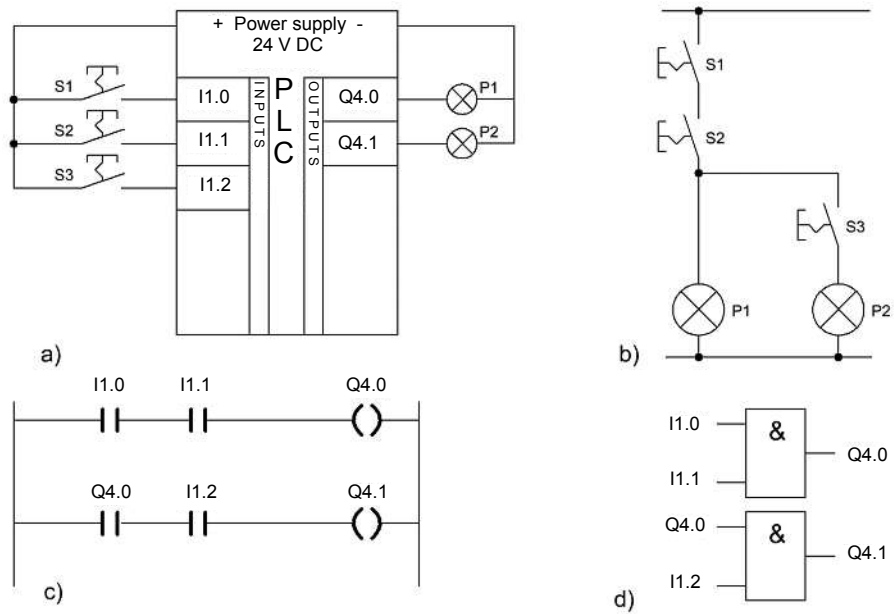


Figure 6.5 Creating the project: a) Connection to the PLC, b) Schematic diagram, c) LAD, d) FBD

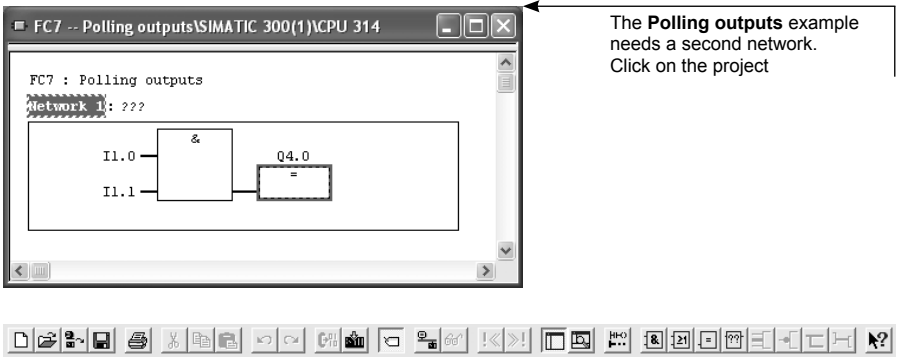
For all the following exercises we will be using the SIMATIC S7-300 station, just like in the first exercise (logical AND operation program). Proceed as follows when programming the **Polling outputs** program:

- ☐ Create the Polling outputs project
- ☐ Create block FC 7
- ☐ Program Polling outputs in FBD
- ☐ Change OB 1
- ☐ Download FC 7 and OB 1 into the CPU
- ☐ Test the program

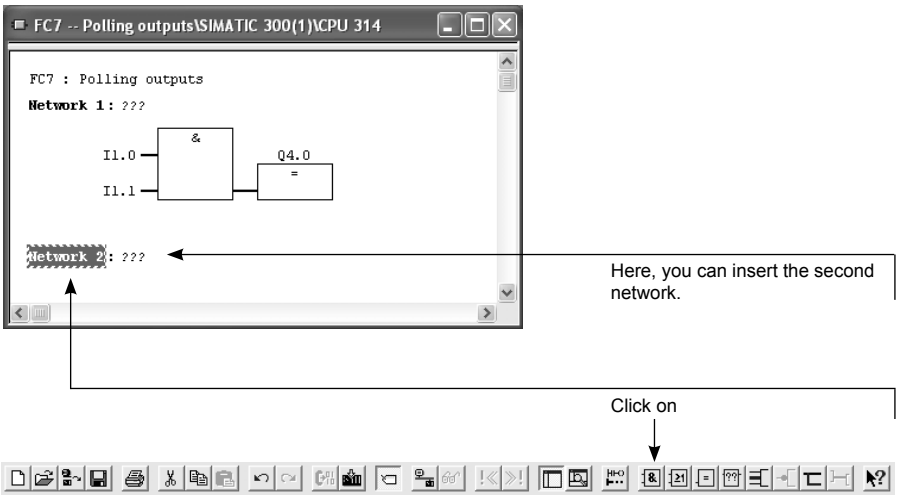
The first two procedures will not be described here (see the AND operation procedure). In this example, you must, however, insert a second network.

# 6.6 Inserting networks

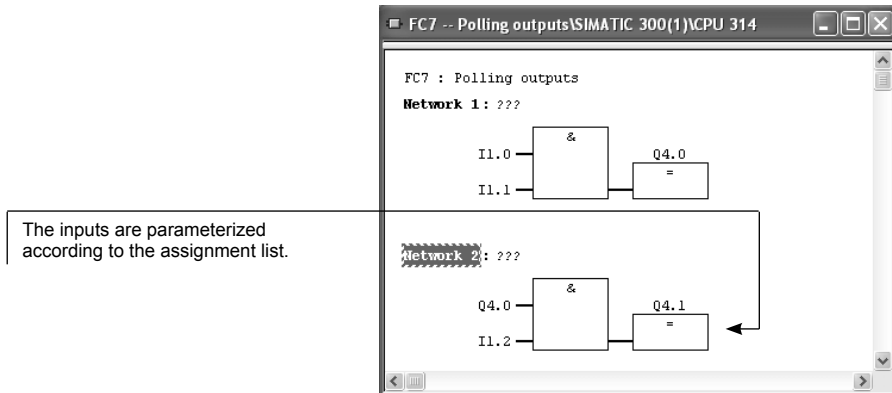
⇒ The following window is open:



⇒ The following window is displayed:



⇒ The following window is displayed:



The program is further executed after the learned steps.

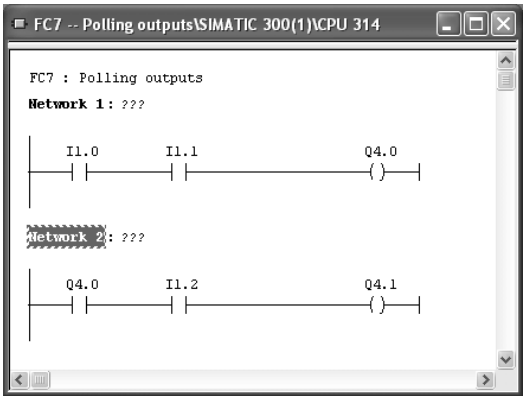
You delete networks in the following order:

- ☐ Click on network 2 (heading),
- ☐ Choose menu item: **Edit > Delete** or press the **DEL** key,
- ☐ The network is deleted.

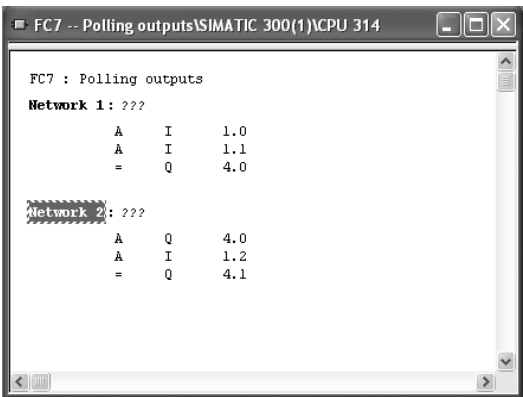
You undo the command using menu item:  
**Edit > Undo** or by pressing **Ctrl + Z**.

The following windows show the **Polling outputs** program.

Method of representation: **LAD**



Method of representation: **STL**



Test the program.

## 6.7 Latch circuit with the PC

The circuit that is commonly used in protective controls for an L/U function is a latch circuit.

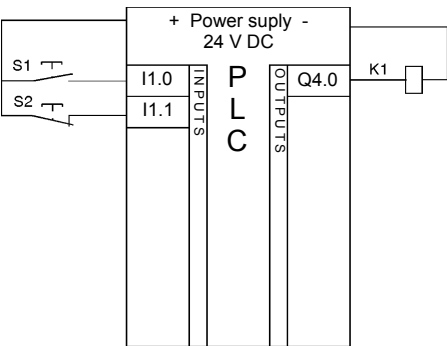
To deactivate the contactor, two variants are possible depending on whether activation or deactivation has priority.

When programming, you must note that even when using a PLC the pushbutton that activates towards OFF should have NC contacts for safety reasons and must be polled for a 1 signal.

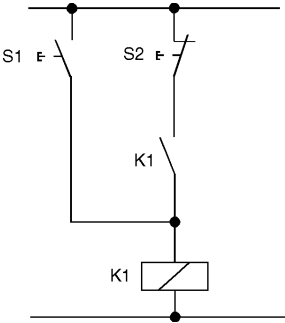
### Switched-on dominant

Table 6.6

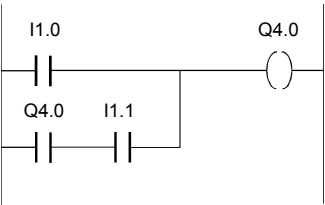
Assignment list		
Symbol	Operand	Comment
S1	I1.0	Button «NO contact»
S2	I1.1	Button «NC contact»
K1	Q4.0	Auxiliary contact



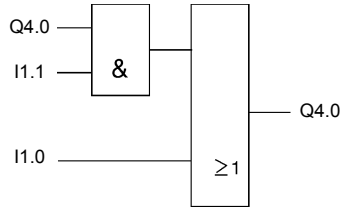
a)



b)



c)



d)

Figure 6.6 Creating the Switched-on dominant project: a) Connection to the PLC, b) Schematic diagram, c) LAD, d) FBD

### Switched-off dominant

Table 6.7

Assignment list		
Symbol	Operand	Comment
S3	I0.0	Button «NO contact»
S4	I0.1	Button «NC contact»
K2	Q4.1	Auxiliary contact

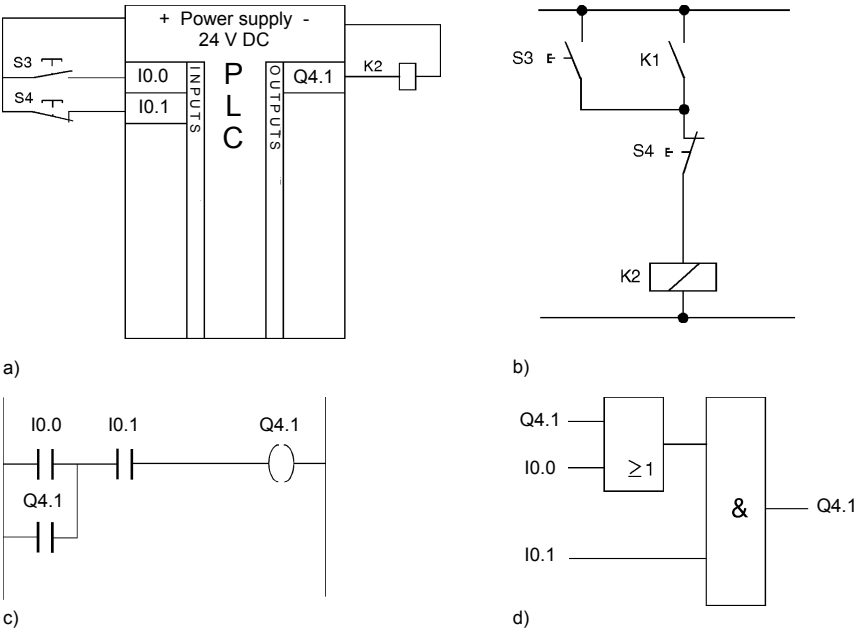


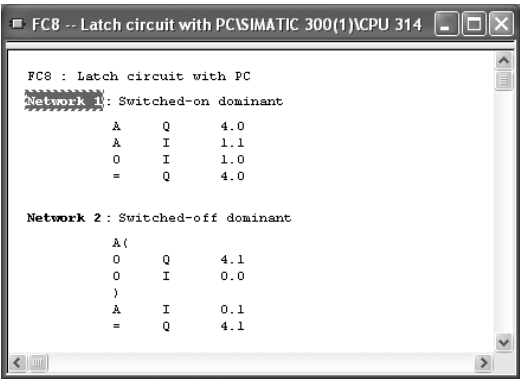
Figure 6.7 Creating the Switched-off dominant project: a) Connection to the PLC, b) Schematic diagram, c) LAD, d) FBD

Proceed as follows:

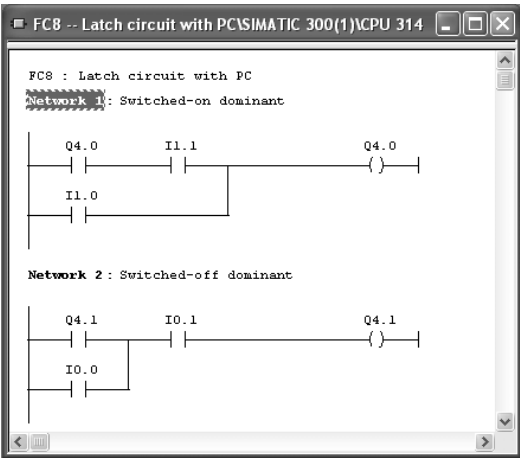
- ☐ Create the Latch circuit project
- ☐ Create block FC 8
- ☐ Program Latch circuit project in FBD
- ☐ Change OB 1
- ☐ Download FC 8 and OB 1 into the CPU

The subsequent windows show the program in the STL, LAD and FBD methods of representation as a solution.

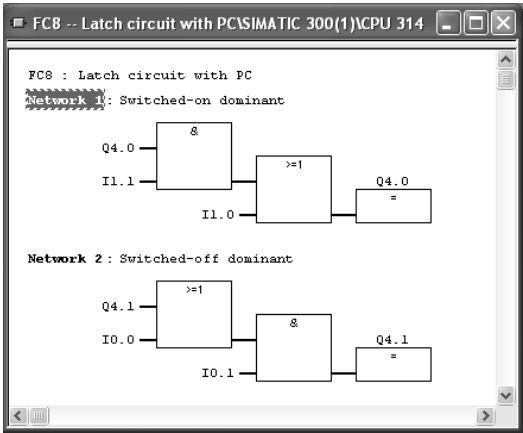
Method of representation: STL



Method of representation: LAD



Method of representation: FBD



Test the program.



# 6.8 Practical examples of control with the PC

## 6.8.1 Temperature difference

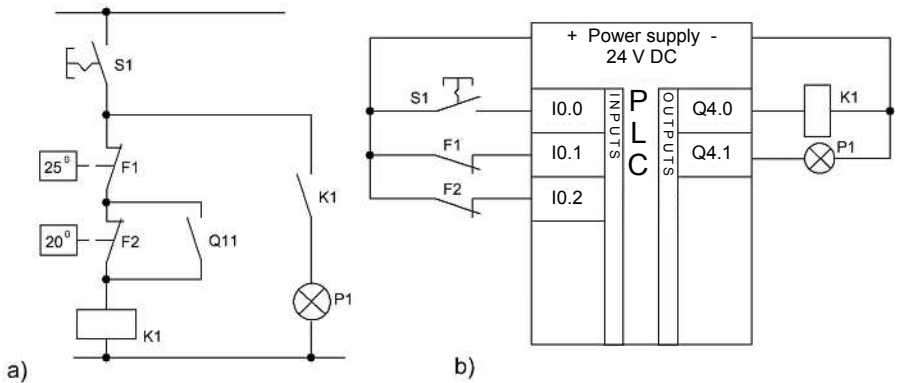


Figure 6.8 Example of control – Temperature difference circuit: a) Schematic diagram  
b) Connection to a PLC

### Description of function (Figure 6.8)

If power controller S1 is activated, the system is ready for operation. Thermostat F1 switches at 25 °C, and thermostat F2 at 20 °C. If the temperature drops below 20 °C, then the contact elements of both thermostats are closed and contactor K1 picks up. If the temperature rises above 20 °C, thermostat F2 opens; however, contactor K1 stays picked-up via the NO contact of K1 that is parallel-switched to F2 until the temperature rises above 25 °C and thermostat F1 opens. Indicator light P1 lights up with the contactor energized. The system can be deactivated again via power controller S1.

Table 6.8

Assignment list		
Symbol	Operand	Comment
S1	I0.0	Plant temperature difference «ON» NO contact
F1	I0.1	Thermostat 25 °C «NC contact»
F2	I0.2	Thermostat 20 °C «NC contact»
K1	Q4.0	Auxiliary contact for switching duty
P1	Q4.1	Indicator light

Proceed as follows:

- ☐ Create a Temperature difference project in FBD
- ☐ Choose a function (we suggest FC 9)
- ☐ Organize OB 1
- ☐ Download the program into the CPU
- ☐ Test the program (**Switch on the CPU to RUN-P**: In this switch position, it is possible to change programs in online mode without needing to put the switch into the STOP position).
- ☐ Change the methods of representation: > FBD > LAD > STL

End of the exercise.

### 6.8.2 Drinks machine

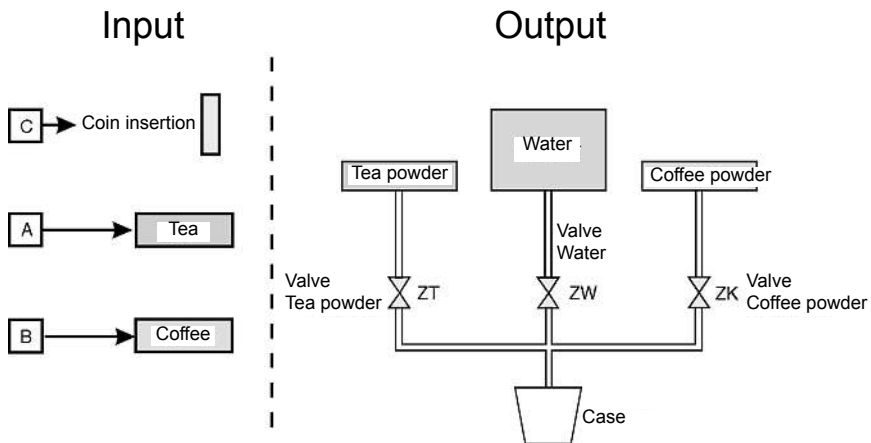


Figure 6.9 Example of control Drinks machine: Technology scheme

#### Description of Function

After inserting an appropriate coin in the slot, users can choose the desired drink by pressing the **Tea** or **Coffee** button. For methodical reasons, the problem is simplified to three input variables. The beverage should always be dispensed if users press the tea or coffee button after inserting a coin.

The following operation is carried out: The solenoid valve **ZW** releases the hot water. Hot water is added to the tea or coffee powder. (To make things simpler, no milk or sugar is added).

ZK = Coffee powder runs out  
 ZT = Tea powder runs out  
 ZW = Hot water runs out

If none of the pushbuttons are pressed, all the input signals are «0».

Table 6.9

Function table					
A (Tea)	B (Coffee)	C (Coin)	ZT (Tea powder)	ZK (Coffee powder)	ZW (Water)
0	0	0	0	0	0
0	0	1	0	0	0
0	1	1	0	1	1
0	1	0	0	0	0
1	0	1	1	0	1
1	0	0	0	0	0
1	1	1	0	0	0
1	1	0	0	0	0

**Task:**  
 A drinks machine is to be programmed in accordance with the specified Function Block Diagram (Figure 6.10).

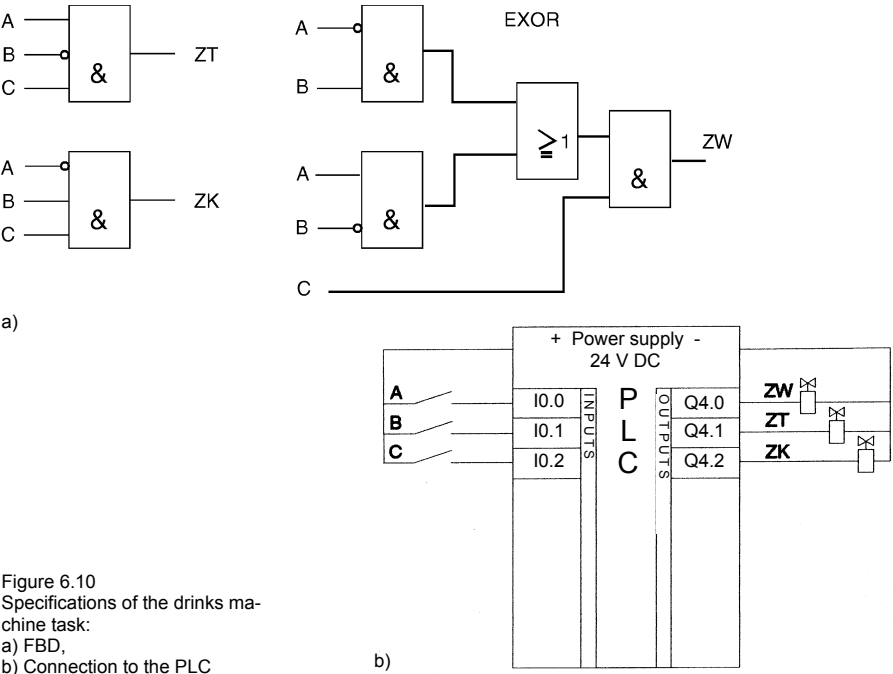


Figure 6.10  
 Specifications of the drinks machine task:  
 a) FBD,  
 b) Connection to the PLC

Table 6.10

Assignment list		
Symbol	Operand	Comment
A	I0.0	Input "Tea preselection"
B	I0.1	Input "Coffee preselection"
C	I0.2	Coin insertion
ZW	Q4.0	Solenoid valve «Water»
ZT	Q4.1	Solenoid valve «Tea»
ZK	Q4.2	Solenoid valve «Coffee»

Proceed as follows:

- ☐ Program the drinks machine in FBD
- ☐ Choose a function (we suggest FC 10)
- ☐ Organize OB 1
- ☐ Download into the CPU
- ☐ Test the program (**Switch on the CPU to RUN-P**: In this switch position, it is possible to change programs in online mode without needing to put the switch into the STOP position).
- ☐ Change the methods of representation: > FBD > LAD > STL

6.8.3 Intercom

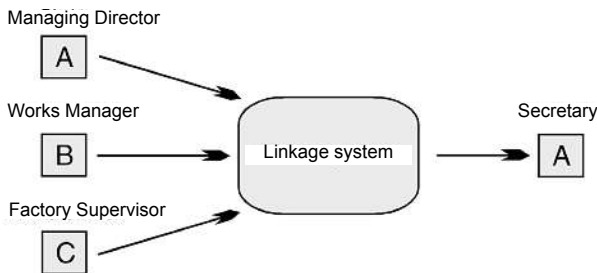
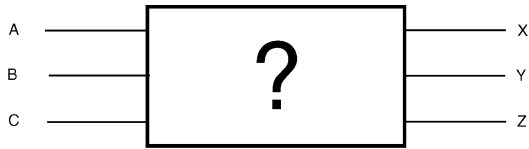


Figure 6.11  
Intercom:  
Technology scheme

Description of Function

In a company, three senior staff members are intended to be able to speak to the secretary (see Figure 6.11). These people are the Managing Director A, the Works Manager B and the Factory Supervisor C. In accordance with the positions of these people in the company, their conversations are given different priorities, i.e. the Managing Director has the highest priority, the Factory Supervisor is only put through if neither of the other users is speaking (see Figure 6.12).

Figure 6.12  
Definition of the switching  
variables



X = 1, A is speaking  
Y = 1, B is speaking, i.e., A = 0  
Z = 1, C is speaking, i.e., A = 0, B = 0

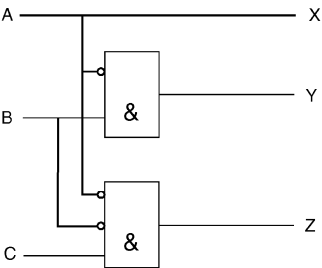
You should also ensure that when somebody wants to speak, the higher priority is prioritized (see Figure 6.13) and a call with a lower priority is interrupted.

Table 6.11

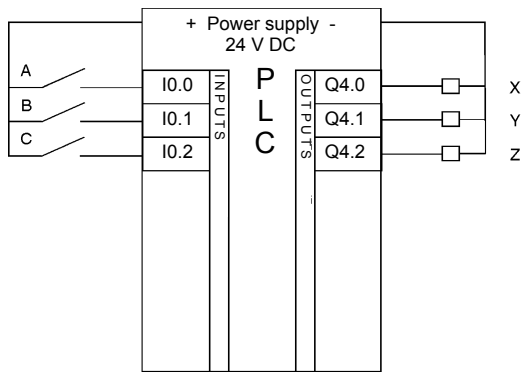
Function table		
Input variable		
A	B	C
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Table 6.12

Function table		
Output variable		
X	Y	Z
0	0	0
0	0	1
0	1	0
1	0	0
1	0	0
1	0	0
1	0	0
1	0	0



a)



b)

Figure 6.13 Create project: a) FBD, b) Connection to the PLC

Table 6.13

Assignment list		
Symbol	Operand	Comment
A	I0.0	Managing Director
B	I0.1	Works Manager
C	I0.2	Factory Supervisor
X	Q4.0	Relay – Managing Director is speaking
Y	Q4.1	Relay – Works Manager is speaking
Z	Q4.2	Relay – Factory Supervisor is speaking

Proceed as follows:

- ☐ Program the Intercom in FBD
- ☐ Choose a function (we suggest FC 11)
- ☐ Organize OB 1
- ☐ Download the program into the CPU
- ☐ Test the program (**Switch on the CPU to RUN-P:** In this switch position, it is possible to change programs in online mode without needing to put the switch into the STOP position).
- ☐ Change the methods of representation: > FDB > LAD > STL

6.8.4 Generator

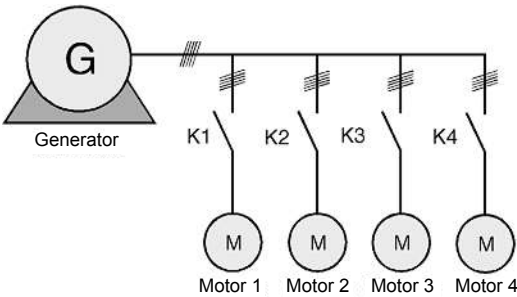


Figure 6.14  
Generator:  
Technology scheme

Description of Function

A generator (see Figure 6.14) supplies a maximum of 7 kW. It is possible to connect four motors to the generator in series. These motors consume the following:

- $P_A = 5 \text{ kW}$
- $P_B = 1 \text{ kW}$
- $P_C = 3 \text{ kW}$
- $P_D = 2 \text{ kW}$

At output A there should be a 1 signal if more than 7 kW is switched-on (see Figure 6.15). The signal renders the last starting command ineffective.

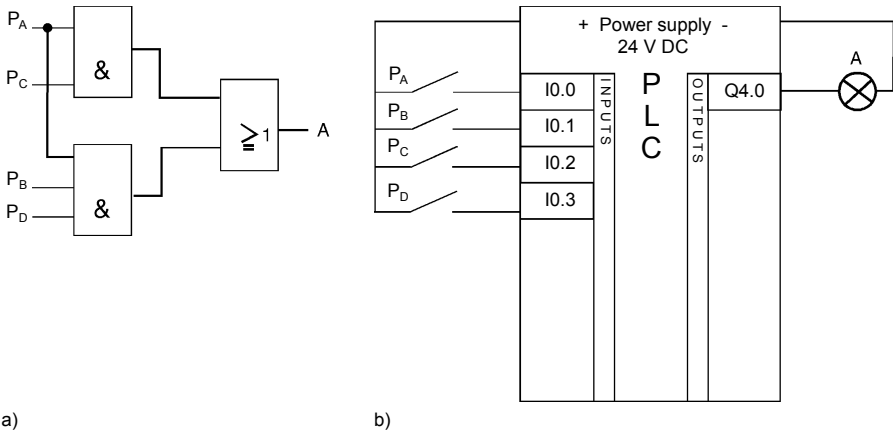


Figure 6.15 Create project: a) FBD, b) Connection to the PLC

Table 6.14

Assignment list		
Symbol	Operand	Comment
P <sub>A</sub>	I0.0	Motor 5 kW
P <sub>B</sub>	I0.1	Motor 1 kW
P <sub>C</sub>	I0.2	Motor 3 kW
P <sub>D</sub>	I0.3	Motor 2 kW
A	Q4.0	Indicator light

Proceed as follows:

- ☐ Program the generator in FBD
- ☐ Choose a function (we suggest FC 12)
- ☐ Organize OB 1
- ☐ Download the program into the CPU
- ☐ Test the program (**Switch on the CPU to RUN-P**: In this switch position, it is possible to change programs in online mode without needing to put the switch into the STOP position).
- ☐ Change the method of representation: > FDB > LAD > STL

End of the exercise.

6.8.5 Boiler control

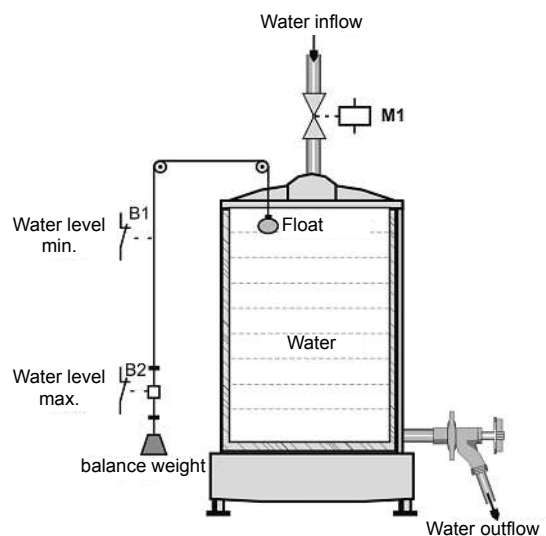


Figure 6.16  
Boiler control:  
Technology scheme

Description of Function

It is intended to fill a boiler (Figure 6.16) with water. If the boiler is empty (the **min.** probe is not activated), water inflow valve M1 activates and the container fills up automatically. On reaching the **max.** probe, the water valve is closed again; when reaching the bottom water level, the process starts from the beginning again. The plant is switched-on with switch S0. The water inflow valve is displayed by indicator light P0 (see Figure 6.17).

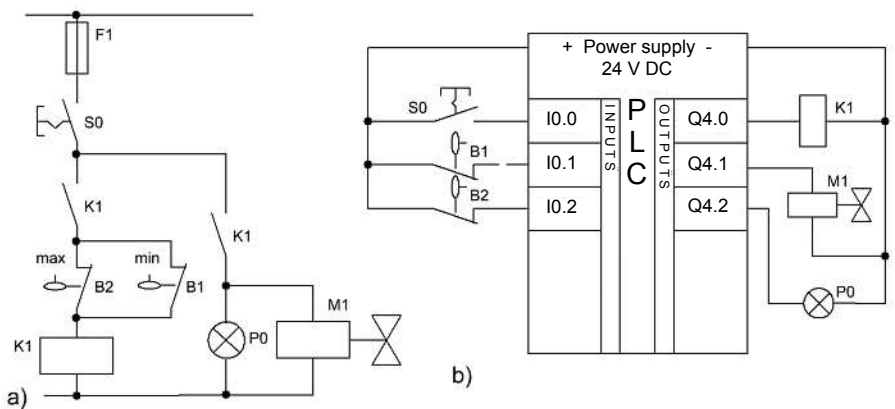


Figure 6.17 Boiler control: a) Schematic diagram, b) Connection to the PLC



Table 6.15

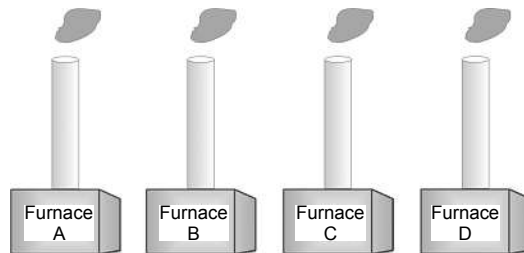
Assignment list		
Symbol	Operand	Comment
S0	I0.0	Plant boiler control «On» – NO contact
S1	I0.1	Min. water probe – NC contact
S2	I0.2	Max. water probe – NC contact
K1	Q4.0	Auxiliary contact
M1	Q4.1	Water inflow valve
P0	Q4.2	Indicator light for water inflow

Proceed as follows:

- ☐ Program the boiler control in FBD
- ☐ Choose a function (we suggest FC 13)
- ☐ Organize OB 1
- ☐ Download the program into the CPU
- ☐ Test the program (**Switch on the CPU to RUN-P**: In this switch position, it is possible to change programs in online mode without needing to put the switch into the STOP position).
- ☐ Change the method of representation: > FDB > LAD > STL

### 6.8.6 Smelting furnaces

Figure 6.18  
Smelting furnaces:  
Technology scheme



#### Description of Function

A company operates four smelting furnaces (A, B, C and D). To supply the electrical energy that they need, daily peak times have been agreed. If this maximum value is exceeded, the company must pay a high price.

The four smelting furnaces have the following individual connection values (i.e. relative to the maximum value on a percentage basis):

- A = 65 %
- B = 45 %
- C = 25 %
- D = 25 %

An interlocking circuit is intended to make it impossible to switch in a furnace if this leads to the permissible maximum current consumption being exceeded. If due to a fault, more furnaces are switched-in, it is intended for an alarm signal to be issued.

It is assumed that two furnaces are never switched-on at the same time (see Figure 6.19).

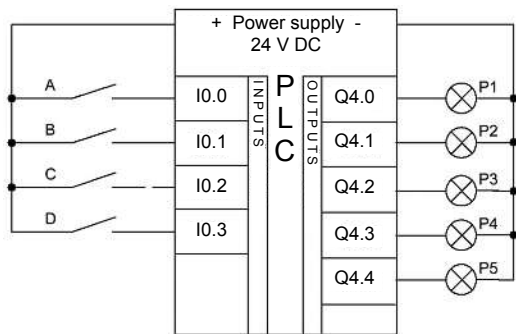


Figure 6.19  
Connection to the PLC

Table 6.16

Assignment list		
Symbol	Operand	Comment
A	I0.0	Furnace 1
B	I0.1	Furnace 2
C	I0.2	Furnace 3
D	I0.3	Furnace 4
Va	Q4.0	Indicator light P1
Vb	Q4.1	Indicator light P2
Vc	Q4.2	Indicator light P3
Vd	Q4.3	Indicator light P4
Vp	Q4.4	Indicator light P5

Table 6.17

Function table								
A	B	C	D	Va	Vb	Vc	Vd	Vp
0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0
0	0	1	0	0	0	0	0	0
0	0	1	1	1	0	0	0	0
0	1	0	0	1	0	0	0	0
0	1	0	1	1	0	0	0	0
0	1	1	0	1	0	0	0	0
0	1	1	1	1	0	0	0	0
1	0	0	0	0	1	0	0	0
1	0	0	1	0	1	1	0	0
1	0	1	0	0	1	0	1	0
1	0	1	1	0	0	0	0	1
1	1	0	0	0	0	0	0	1
1	1	0	1	0	0	0	0	1
1	1	1	0	0	0	0	0	1
1	1	1	1	0	0	0	0	1

Furnace «A» is operating

Furnace «B» is operating

Furnace «C» is operating

Furnace «D» is operating

Va > switchgear equipment for furnace «A» locked.

Vb > switchgear equipment for furnace «B» locked.

Vc > switchgear equipment for furnace «C» locked.

Vd > switchgear equipment for furnace «D» locked.

Vp > alarm has triggered.

The circuit in FBD is shown in Figure 6.20.

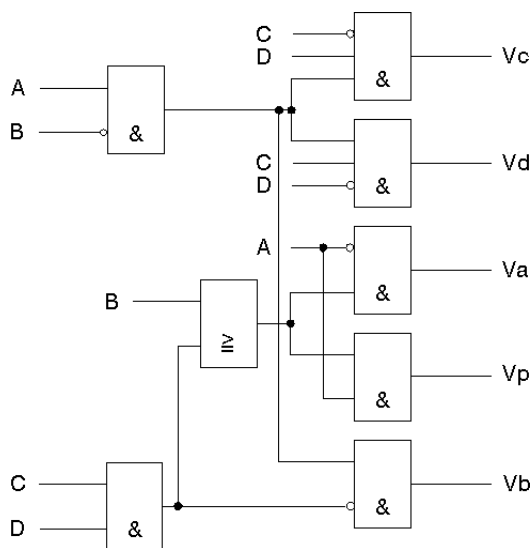


Figure 6.20  
FBD

Proceed as follows:

- ☐ Program the smelting furnaces in FBD
- ☐ Choose a function (we suggest FC 14)
- ☐ Organize OB 1
- ☐ Download the program into the CPU
- ☐ Test the program (**Switch on the CPU to RUN-P:** In this switch position, it is possible to change programs in online mode without needing to put the switch into the STOP position).
- ☐ Change the methods of representation: > FDB > LAD > STL
- ☐ Compare the results with the function table

End of the exercise.

## 6.9 Flip-flop

### 6.9.1 R-S flip-flop

Every circuit that has two stable status conditions and that can be switched from one status to another by means of corresponding input signals is considered to be a signal latch (latch circuit with a control relay). In digital technology, they are also referred to as flip-flops.

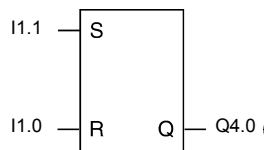
According to DIN 40 900, an R-S function is represented as a rectangle with a set input **S** and a reset input **R**. A brief signal state of 1 at set input **S** sets up the set/reset function. A brief signal state of 1 at reset input **R** leads to the set/reset function being reset (switched off). A signal state of 0 at inputs **R** and **S** does not change the previously set status.

If there is a 1 signal at both inputs at the same time, set or reset dominant is carried out. You must take into account this **Set** or **Reset** dominant at programming.

**Set = switch** on or **reset = switch off**: with a 1 signal only; no effect with a 0 signal (signal state is retained).

### Reset dominant

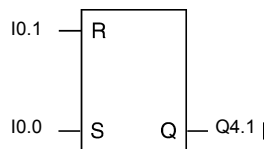
Figure 6.21



The controller processes the statements that you programmed last with priority. In Figure 6.21, the set operation is executed first and then the reset operation, i.e. if both inputs (S and R) have 1 signals in this case, the system resets dominant the flip-flop (memory). Output Q4.0 carries a 0 signal. In practice, this is the memory that is used most.

### Set dominant

Figure 6.22



The controller handles the statements that you programmed last with priority. In Figure 6.22, the system sets the memory if both inputs are carrying 1 signals.

6.9.2 Entering the program

Task (Figure 6.23):

- ☐ Program **R-S flip-flop** in FBD
- ☐ Choose function FC 10
- ☐ Organize OB 1
- ☐ Download the program into the CPU
- ☐ Test the program
- ☐ Change the methods of representation: > FDB > LAD > STL

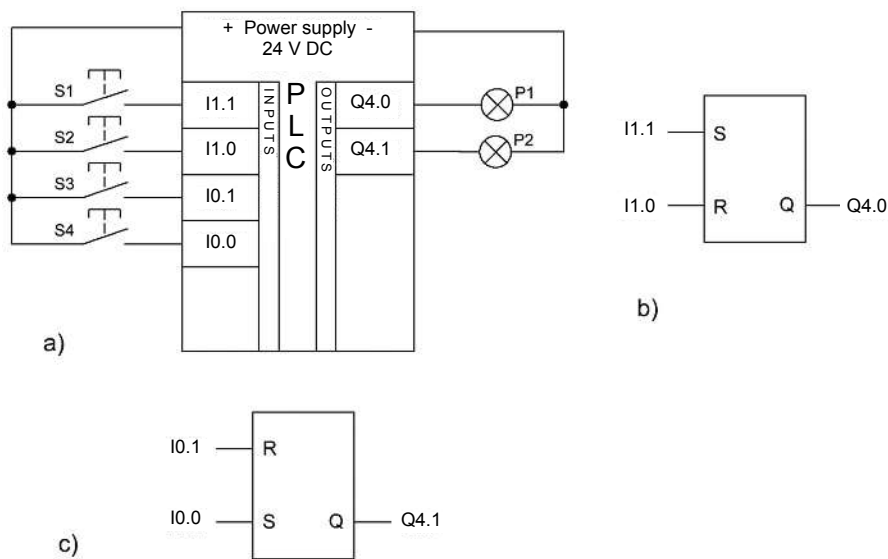


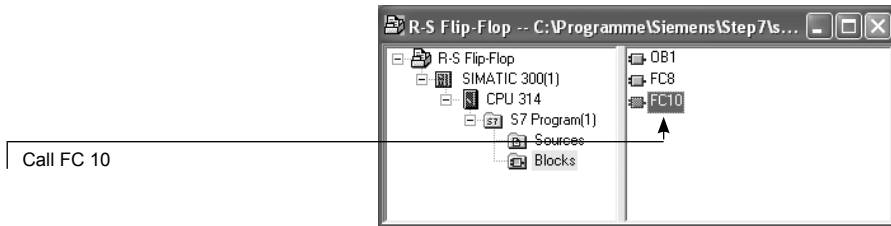
Figure 6.23 R-S function for the task: a) Connection to the PLC, b) Reset dominant (FBD), c) Set dominant (FBD)

Table 6.18

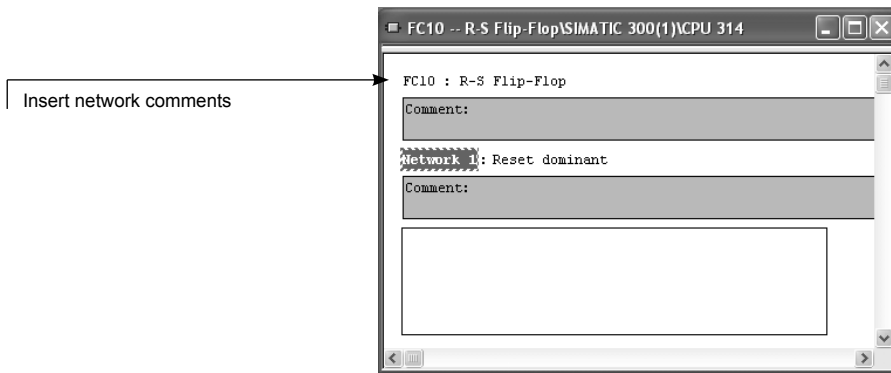
Assignment list		
Symbol	Operand	Comment
S1	I1.1	Button «NO contact»
S2	I1.0	Button «NO contact»
S3	IO.1	Button «NO contact»
S4	IO.0	Button «NO contact»
P1	Q4.0	Indicator light
P2	Q4.1	Indicator light

The previous steps for creating a project will no longer be explained in detail here.

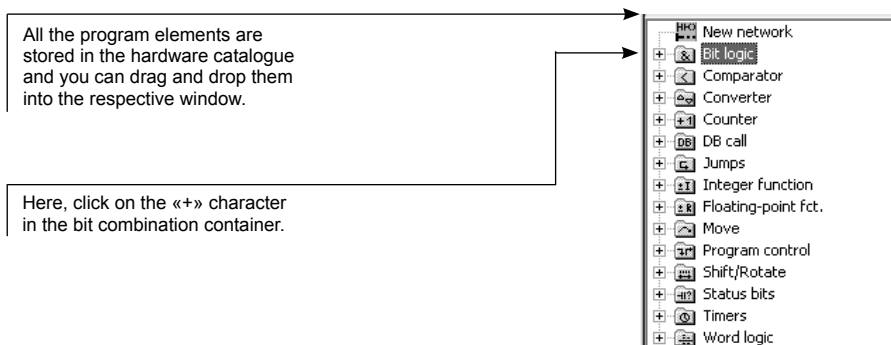
⇒ The following window is displayed:



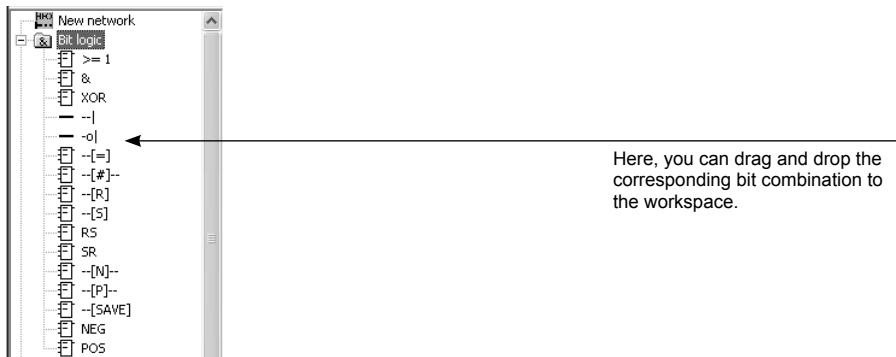
⇒ The following window is displayed:



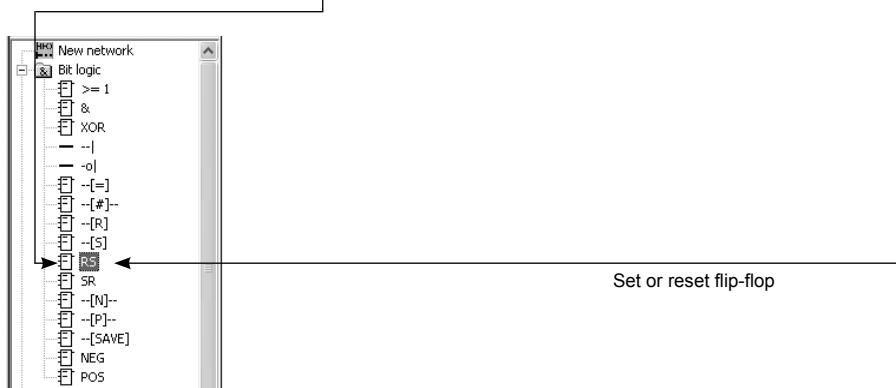
⇒ The following window is displayed:



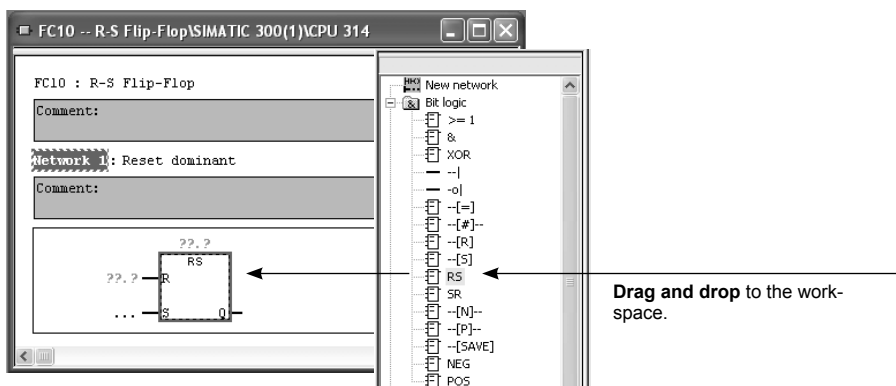
⇒ The following window is displayed:



For our R-S function (Reset dominant), we need the following function (SR):

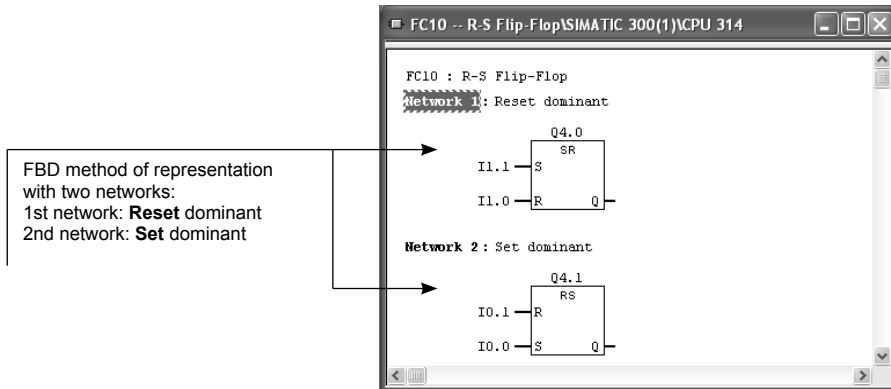


⇒ The following window is displayed:



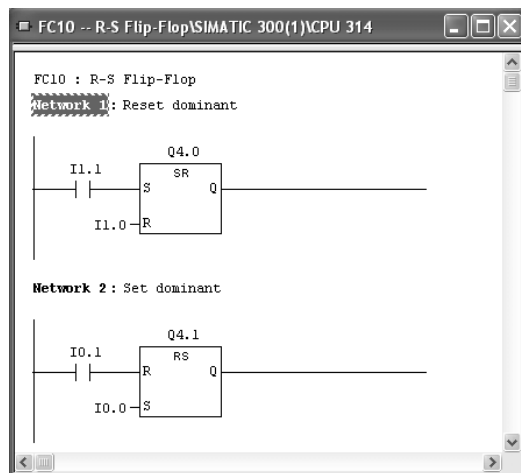


⇒ After completing the program, the following window is displayed:

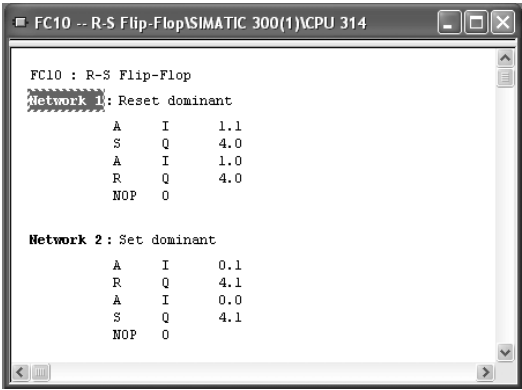


The subsequent windows show the program in the LAD and STL methods of representation as a solution.

Method of representation: **LAD**



Method of representation: STL



Test the program.

6.9.3 Pump controller

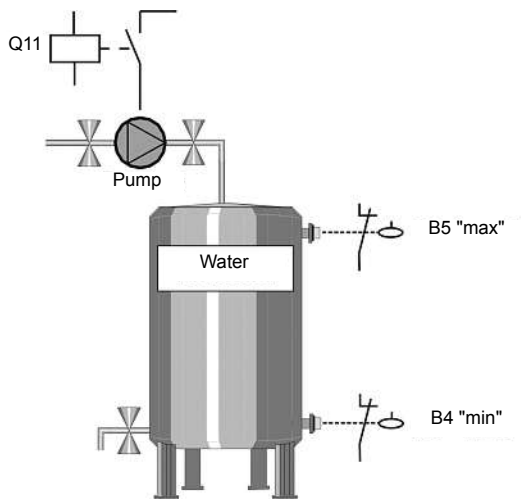


Figure 6.24  
Pump controller:  
Technology scheme

### Description of Function

Filling of a water container (Pump On) is carried out automatically after switching on the plant if the bottom level has been reached. On reaching the top fill level, the pump switches off.

To prevent switching off due to water "sloshing", the **Fill level max.** condition must be present for at least 1 second. The liquid-level switches are emulated by sensors.

Automatic-manual switching is carried out using button S1. Indicator light P1 shows manual operation. In manual mode, it is only possible to switch on the pump using button S2 **Manual On**. Switch off is carried out using button S3 **Manual Off**, or when the maximum fill level "B5 max." is reached.

After switch on, the plant is in automatic mode. The two fill level sensors, "B4 min." and "B5 max.", can now switch the pump motor on and off respectively via contactor Q11. The "min." level condition only needs to be pending briefly; the "max." level condition must be present for at least 1 second. S2 and S3 have no effect in automatic mode. Using button S1, manual -automatic switching is carried out via a T flip-flop (pulse circuit – binary scaler). In manual mode, "S4 min." has no effect and the pump can only be switched on by activating S2 **Manual On**. After this, it is possible to switch off using S3 **Manual Off** or by activating S5 "max." The associated logic diagram is shown in Figure 6.25, with Figure 6.26 showing the connection to the PLC.

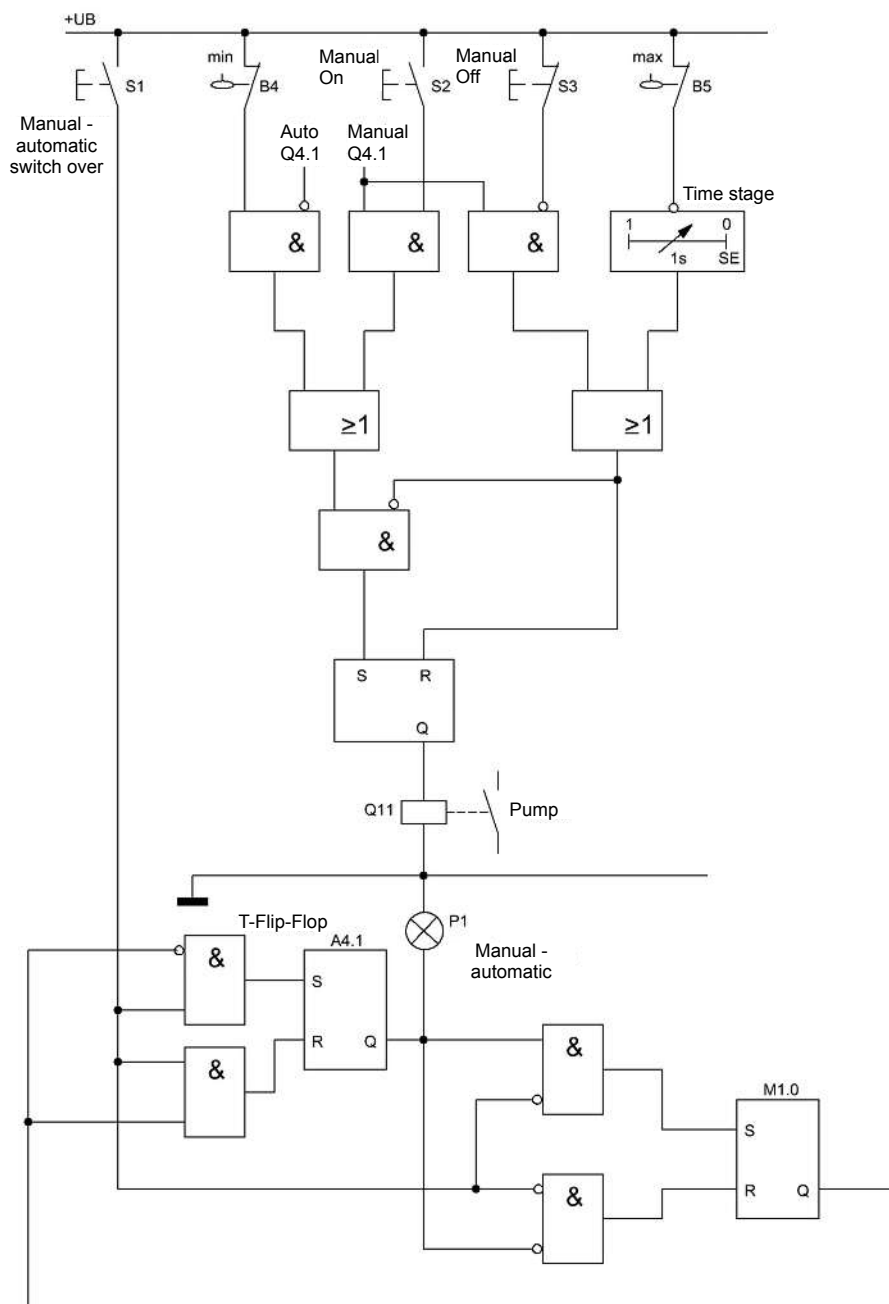
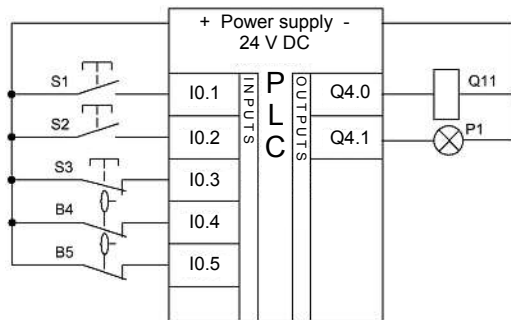


Figure 6.25 Logic diagram

Table 6.19

Assignment list		
Symbol	Operand	Comment
S1	I0.1	Button «NO contact» – Manual/automatic switchover
S2	I0.2	Button «NO contact» – Pump Manual On
S3	I0.3	Button «NC contact» – Pump Manual Off
B4	I0.4	Liquid-level switch «NC contact» min. level
B5	I0.5	Liquid-level switch «NC contact» max. level
Q11	Q4.0	Power contactor «Pump motor»
P1	Q4.1	Indicator light «Manual/automatic»

Figure 6.26  
Connection to the PLC

Proceed as follows:

- ☐ Program Pump controller in FBD, LAD and STL
- ☐ Choose a function (we suggest FC 29)
- ☐ Organize OB 1
- ☐ Download the program into the CPU
- ☐ Test the program (**Switch on the CPU to RUN-P:** In this switch position, it is possible to change programs in online mode without needing to put the switch into the STOP position).
- ☐ Change the methods of representation: > FBD > LAD > STL



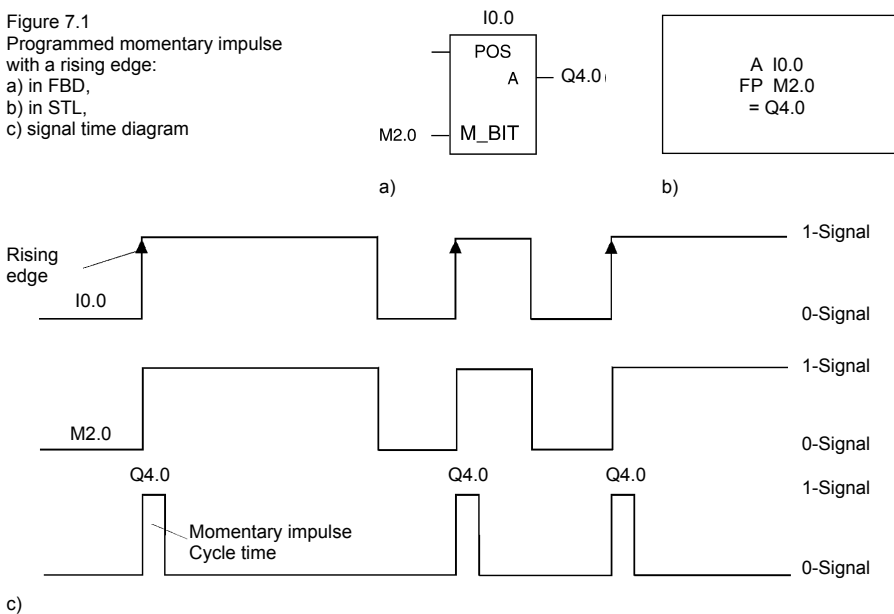
## 7 Creating Momentary Impulses (Edge Instructions)

Rather than acquiring a static signal state 0 and 1, edge instructions acquire a signal change, e.g. of an input. The program of an edge instruction corresponds to an edge-detecting contact in a relay circuit.

### 7.1 Momentary impulse with a rising edge (FP)

Figure 7.1  
Programmed momentary impulse  
with a rising edge:

a) in FBD,  
b) in STL,  
c) signal time diagram



If the system detects a rising (positive) edge (change from 0 to 1) at I0.0 (see Figure 7.1), it sets output Q4.0 to 1 for one OB 1 cycle. A rising edge is detected by the automation system saving the RLO (that supplied operation "A") in edge trigger memory M2.0 and comparing it with the RLO of the previous cycle.

## 7.2 Momentary impulse with a falling edge (FN)

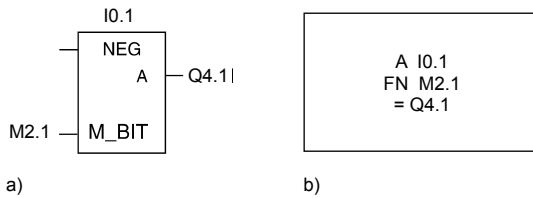
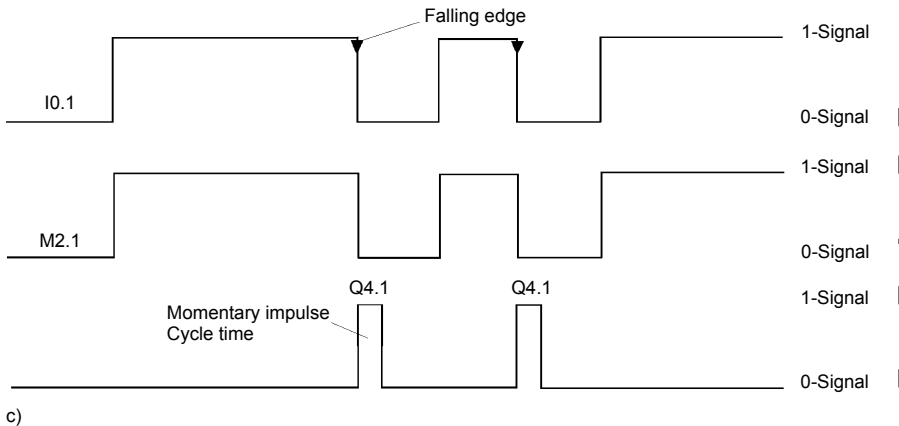


Figure 7.2  
Programmed momentary  
impulse with a falling edge:  
a) in FBD,  
b) in STL,  
c) signal time diagram



If the system detects a falling edge (see Figure 7.2) at I0.1 (change from 1 to 0), it sets output Q4.1 to 1 for one OB 1 cycle. A falling edge is detected by the automation system saving the RLO (that supplied operation "A") in edge trigger memory M2.1 and comparing it with the RLO of the previous cycle.

## 7.3 Program input

Task:

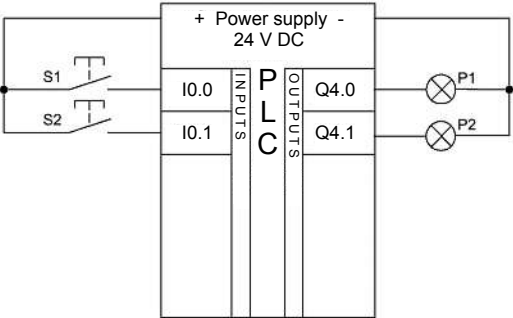
- ☐ Program the momentary impulses in FBD
- ☐ Use function FC 11
- ☐ Organize OB 1
- ☐ Download the program into the CPU
- ☐ Test the program
- ☐ Change the methods of representation: > FBD > LAD > STL



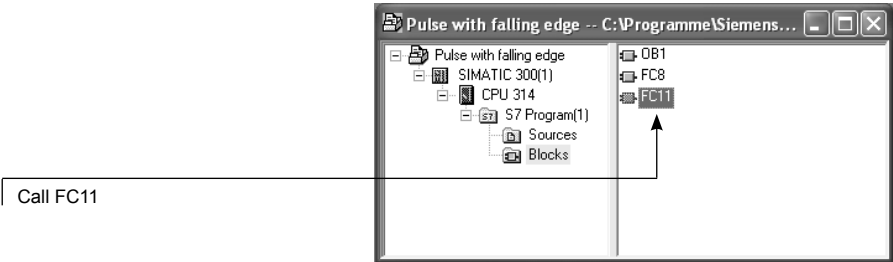
Table 7.1

Assignment list		
Symbol	Operand	Comment
S1	I0.0	Button «NO contact»
S2	I0.1	Button «NO contact»
P1	Q4.0	Indicator light
P2	Q4.1	Indicator light

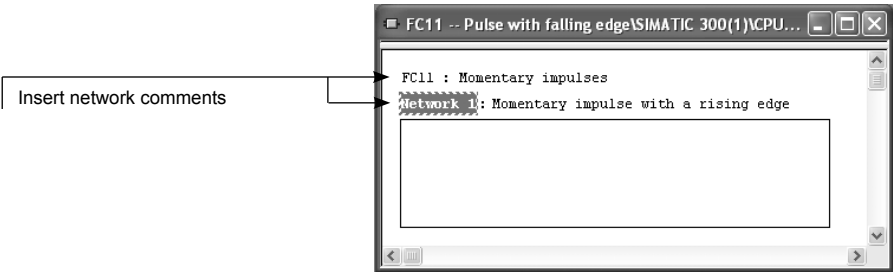
Figure 7.3  
Momentary impulse program  
input: Connection to the PLC



The preceding steps for configuration will no longer be explained in detail here.

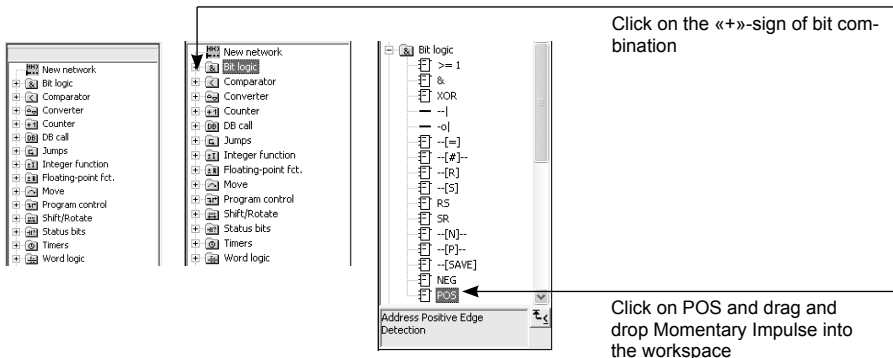


⇒ The following window is displayed:



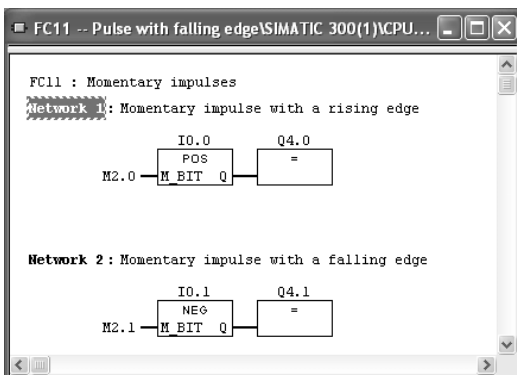
Click on Program elements: Overview on/off

⇒ The following window is displayed:



Drag the momentary impulse for the falling edge (NEG) into the second network of the workspace too.

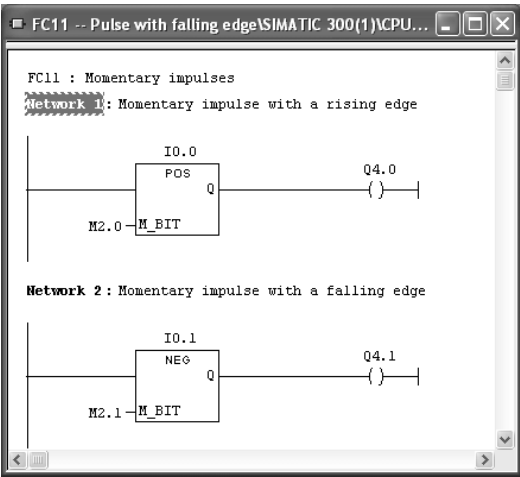
⇒ The following window is displayed:



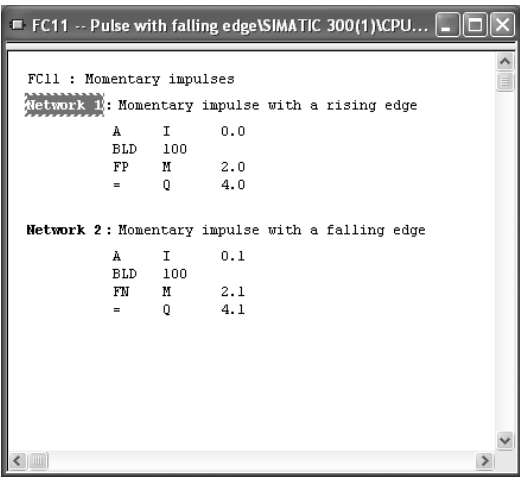
The window shows the FBD with two networks:  
 1st network: momentary impulse with a rising edge  
 2nd network: momentary impulse with a falling edge

In the following windows, the program is shown with LAD and STL as a solution.

Method of representation LAD:



Method of representation STL:



Test the program.

## 7.4 Acknowledgement circuit

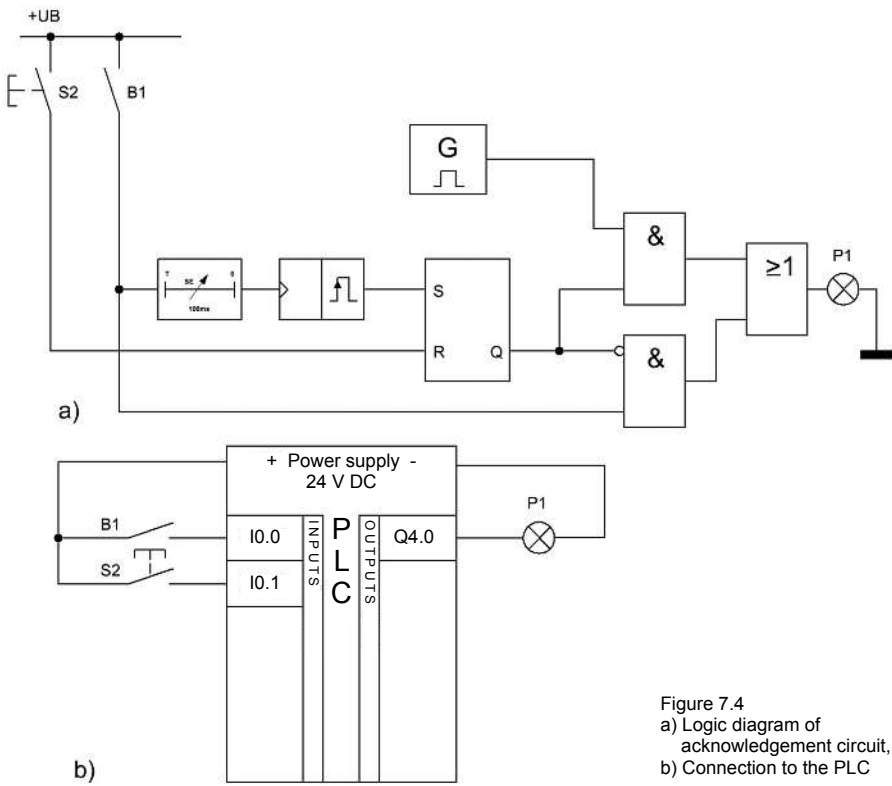


Figure 7.4  
a) Logic diagram of  
acknowledgement circuit,  
b) Connection to the PLC

### Description of Function

In the case of a digital circuit, it is intended for a warning lamp to flash at a frequency of 2 Hz if a false signal (disturbance) occurs.

After activating an acknowledgement key S2, it is intended for the warning lamp to change from flashing to steady light that goes out when the false signal disappears (see Figure 7.4).

Table 7.2

Assignment list		
Symbol	Operand	Comment
B1	I0.0	Disturbance «NO contact»
S2	I0.1	Acknowledge disturbance «NO contact»
P1	Q4.0	Indicator light

Proceed as follows:

- ☐ Program the acknowledgement circuit in STL, LAD and FBD
- ☐ Choose a function (we suggest FC 28)
- ☐ Organize OB 1
- ☐ Download the program into the CPU
- ☐ Test the program (**Switch on the CPU to RUN-P:** In this switch position, it is possible to change programs in online mode without needing to put the switch into the STOP position).
- ☐ Change the methods of representation: STL > FBD > LAD



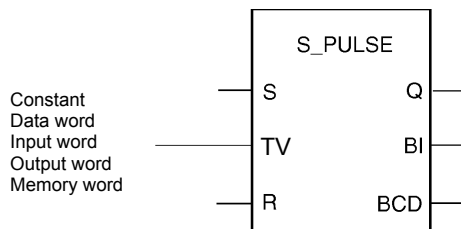
## 8 Timing Functions

The S7-300 programmable controller with the CPU 314 has 128 timers (0...127) that you can use for different functions. The program specifies the runtime and it can be between 0.01 seconds and 9990 seconds or 2 hours, 46 minutes and 30 seconds. To implement control jobs, it is very often necessary to use different timing functions. The timing functions are integrated into the central processing unit of the programmable controller (PLC). Setting of the desired time and starting of the timing function must be carried out via the user program. A 16-bit word is assigned to each of the different timing functions.

### 8.1 Timing value specification

You can specify the time duration TV (runtime) as a predefined constant, as an input word IW, as an output word QW, as a data word DBW/DIW or as a memory word MW (see Figure 8.1).

Figure 8.1  
FBD for TV



Starting from the left, a timer word has two status bits that the processor needs for processing the timing functions. They are irrelevant for specifying the time duration, i.e., they are of no significance for the time sequence. After these, there are two bits for the time base and four bits each for the BCD-coded current value. Figure 8.2 presents the current value of 137 with the time base 1 s.

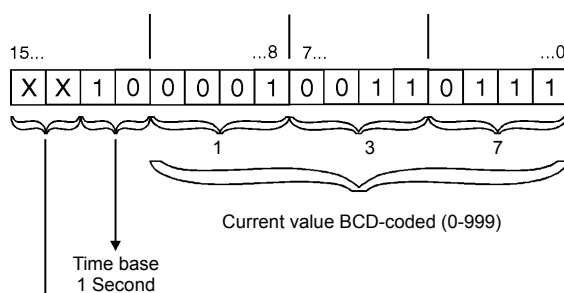


Figure 8.2  
Current value

## 8.2 Release a time (FR)

A positive edge change from 0 to 1 in the RLO of the release operation (FR) releases a time. The release operation (FR) is only present in STL type of representation. The release (FR) is not needed for starting up or for the normal function of a time. This function is only used to post-trigger a running time or to start it up again. Restarting the timer only leads to a function if the system continues to process the start operation with a 1 signal.

## 8.3 Current value

Bits 12 and 13 of the timer word contain the time base in binary-coded form. This defines the interval in which the timer word is reduced by 1 unit. The smallest time base is 10 ms, and the largest is 10 s. Values whose resolutions are too big for the desired range are rounded-down (FBD see Figure 8.3).

Table 8.1

Time base	Time base	Time range
0.01 s	00	10 ms - 9 s and 990 ms (9.990 sec.)
0.1 s	01	100 ms - 1 m and 39 s and 900 ms (1:39.900 min.)
1 s	10	1 s - 16 m and 39 s (16:39 min.)
10 s	11	10 s - 2 h and 46 m und 30 s (2:46:30 hours)

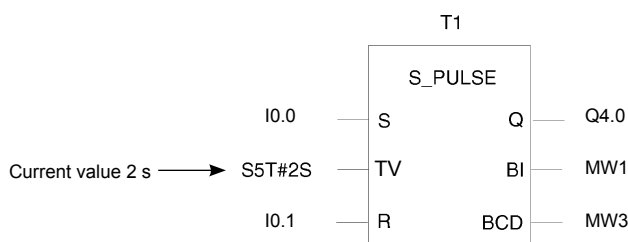


Figure 8.3  
FBD for FR



## 8.4 Reset time

A signal at the reset input (R) of the timer resets the timer; the time sequence is ended. The output (Q) of the timer is reset.

### Poll current value

A current value is stored in binary-coded form in a timer word. This value can be loaded as a dual number (BI) or as a BCD number (BCD) into the accumulator and transferred from there to other operand areas. With STL programming, you use L T1 for dual polling or LC T1 for BCD polling.

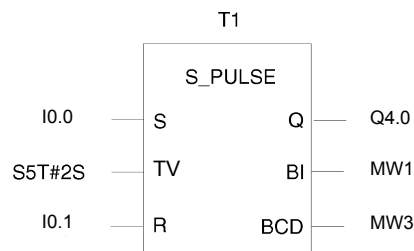
### Polling the signal state of the timer (binary) Q

You can poll a time for its signal state 0 or 1. You can poll using A T1, AN T1, ON T1 etc. and use the result for further logic operations.

## 8.5 Selection of times (five different ones)

### 8.5.1 Pulse timer (SP)

Figure 8.4  
FBD for SP



Output Q (see Figure 8.4) carries a 1 signal after starting of timer **pulse (SP)**. The output is reset when the specified time has expired, if the start signal is reset to 0 or if a 1 signal is pending at the reset input (see Figure 8.5).

### Signal status diagram for a pulse timer (SP) with release (FR)

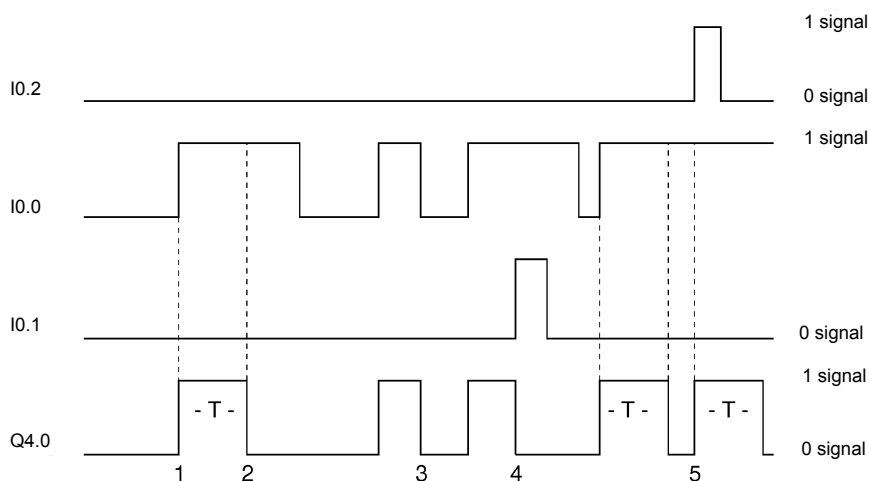


Figure 8.5 Signal status diagram SP with FR

### Explanation of the signal status diagram for pulse timer (SP)

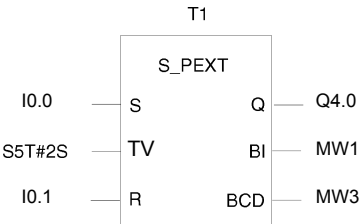
- 1 Output Q4.0 carries a 1 signal after start-up.
- 2 Output Q4.0 is reset when the time has expired.
- 3 Output Q4.0 is reset if the start signal goes to 0.
- 4 Output A4.0 is reset if the reset input has a 1 signal.
- 5 A positive edge change from 0 to 1 of the release operation (FR) restarts the time completely. Restarting is only possible if the start operation continues to carry a 1 signal.

### Statement list for Figure 8.4

A	I0.2	
FR	T1	release of timer T1 (only possible in STL)
A	I0.0	
L	S5T#2S	load start time 2 s in accumulator 1
SP	T1	start time T1 as pulse
A	I0.1	
R	T1	reset T1
A	T1	
=	Q4.0	binary polling of time T1
L	T1	load time T1 in binary-coded form
T	MW1	transfer the value into memory word MW1
LC	T1	load time T1 BCD-coded
T	MW3	transfer the value into memory word MW3

### 8.5.2 Extended pulse timer (SE)

Figure 8.6  
FBD for SE



Output Q (Figure 8.6) carries a 1 signal after starting of the **extended pulse timer (SE)** (Figure 8.7). Output Q is reset if the specified time duration expires or the reset input has a 1 signal. During the time sequence, switching on the start input does not reset output Q (latch circuit). If another signal change occurs at the signal input while the time is running, the time is restarted (retriggered).

#### Signal status diagram for an extended pulse timer (SE)

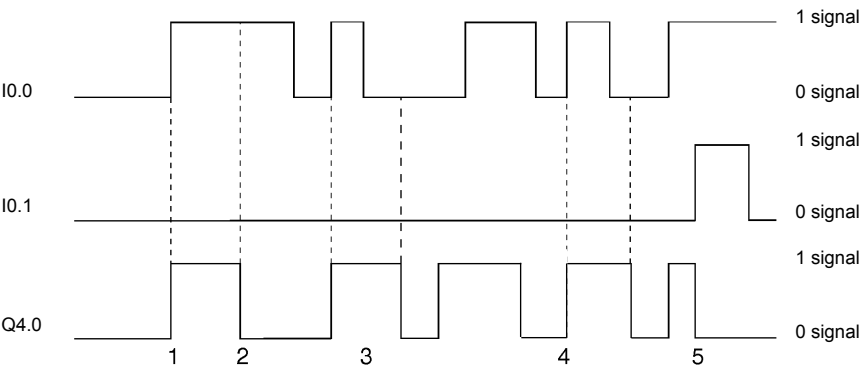


Figure 8.7 Signal status diagram SE

#### Explanation of the signal status diagram for an extended pulse timer SE

- 1 Output Q4.0 carries a 1 signal after start-up.
- 2 Output Q4.0 is reset when the time has expired.
- 3 Output Q4.0 is reset if the start signal goes to 0.
- 4 Timer is restarted (retriggered).
- 5 Output Q4.0 is reset if the reset input has a 1 signal.

Statement list for Figure 8.6

A	I0.0	
L	S5T#2S	load start time 2 s in accumulator 1
SE	T1	start time T1 as extended pulse
A	I0.1	
R	T1	reset T1
A	T1	
=	Q4.0	binary polling of time T1
L	T1	load time T1 in binary-coded form
T	MW1	transfer the value into memory word MW1
LC	T1	load time T1 BCD-coded
T	MW3	transfer the value into memory word MW2

8.5.3 Switch-on delay timer (SD)

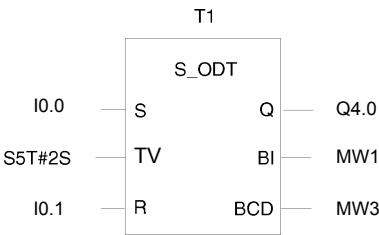


Figure 8.8  
FBD for SD

After starting the **switch-on delay timer (SD)**, output Q (Figure 8.8) does not carry a 1 signal until the programmed time has expired and RLO = 1 is still pending at the start input. Output Q is reset if the start input is switched-off or a 1 signal is pending at the reset input of the timer.

### Explanation of the signal status diagram for a switch-on delay timer SD

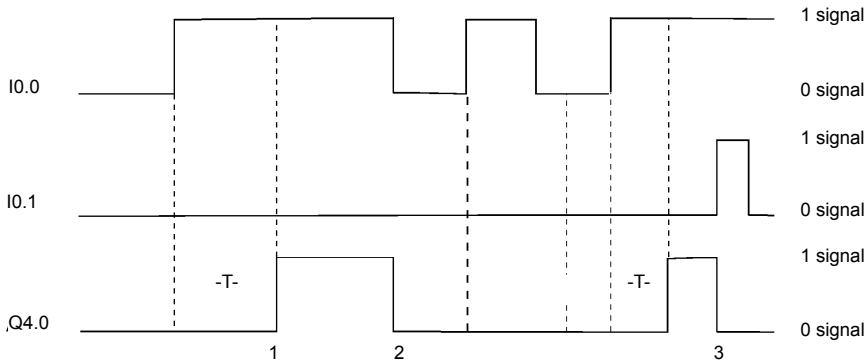


Figure 8.9 Signal status diagram SD

### Explanation of signal status diagram for timer SD

- 1 Output Q4.0 carries a 1 signal when the time has expired.
- 2 Output Q4.0 is reset if the start input is switched-off.
- 3 Output Q4.0 is reset if the reset input has a 1 signal.

### Statement list for Figure 8.8

```

A      I0.0
L      S5T#2S load start time 2 s in accumulator 1
SD     T1      start time T1 as a switch-on delay
A      I0.1
R      T1      reset timer T1
A      T1
=      Q4.0    binary polling of time T1
L      T1      load time T1 in binary-coded form
T      MW1     transfer the value into memory word MW1
LC     T1      load time T1 BCD-coded
T      MW3     transfer the value into memory word MW3

```

8.5.4 Retentive switch-on delay timer (SS)

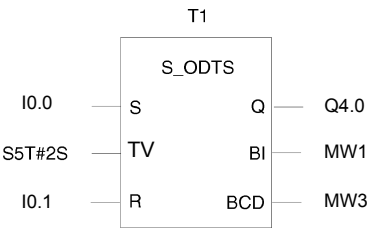


Figure 8.10  
FBD for SS

After starting the **retentive switch-on delay timer (SS)**, output Q (Figure 8.10) does not carry a 1 signal until the programmed time has expired (Figure 8.11). The function no longer needs a 1 signal after starting at the start input, which means that this input can be switched-off (latch circuit). Output Q is reset if a 1 signal is pending at the timer's reset input. Switching the start input off and back on again has the effect of restarting (retriggering) the timing function while the time is running.

Signal status diagram for a retentive switch-on delay timer (SS)

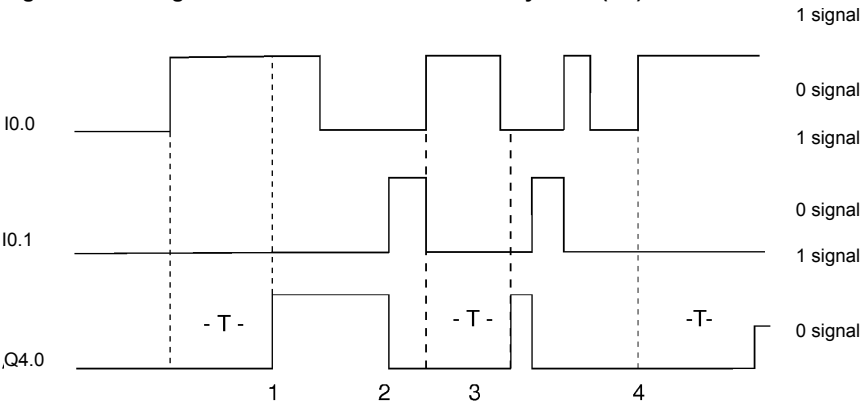


Figure 8.11 Signal status diagram SS

Explanation of signal status diagram for timer SS

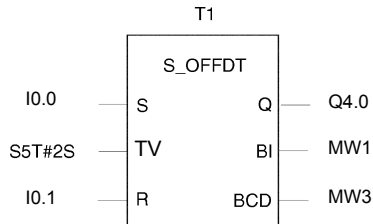
- 1 Output Q4.0 carries a 1 signal when the time has expired.
- 2 Output Q4.0 is reset if the reset input has a 1 signal.
- 3 The function no longer needs a 1 signal after starting at the start input.
- 4 Switching the start input off and back on again has the effect of restarting (retriggering) the timing function while the time is running.

Statement list for Figure 8.10

A	I0.0	
L	S5T#2S	load start time 2 s in accumulator 1
SS	T1	start time T1 as a latching switch-on delay
A	I0.1	
R	T1	reset timer T1
A	T1	
=	Q4.0	binary polling of time T1
L	T1	load time T1 in binary-coded form
T	MW1	transfer the value into memory word MW1
LC	T1	load time T1 BCD-coded
T	MW3	transfer the value into memory word MW3

8.5.5 Switch-off delay timer (SF)

Figure 8.12  
FBD for SF



If there is a signal change from 0 to 1 at the input, output Q (Figure 8.12) switches on directly. If there is a signal change from 1 to 0 at the input, output Q continues to supply a signal state of 1 until the programmed time has expired. This means that switching off the start input has the effect of delaying switch off of the output by the specified time duration. Output Q is also switched-off if the reset input is carrying a 1 signal. Switching on the timer again while the time is running has the effect of pausing the expiring time and not restarting it again until the next switch-off of the start input. If the start and reset inputs of the timing function are carrying a 1 signal, the output of the timer is not set until the dominant reset is switched-off (Figure 8.13).

### Signal status diagram for a switch-off delay timer (SF)

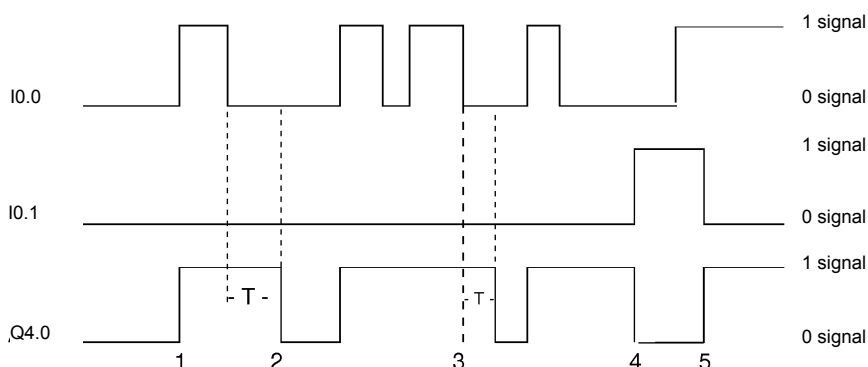


Figure 8.13 Signal status diagram SF

### Explanation of signal status diagram for timer SF

- 1 Output Q4.0 carries a 1 signal if the start input is carrying a 1 signal.
- 2 Output Q4.0 keeps carrying a 1 signal until the time has expired.
- 3 Switching on again has the effect of pausing the expiring time.
- 4 The output of the timer is switched-off at resetting.
- 5 If the start and reset inputs are carrying a 1 signal, the output of the timer is not set until the dominant reset is switched-off.

### Statement list for Figure 8.12

A	I0.0	
L	S5T#2S	load start time 2 s in accumulator 1
SF	T1	start time T1 as a switch-off delay
A	I0.1	
R	T1	reset timer T1
A	T1	
=	Q4.0	binary polling of time T1
L	T1	load time T1 in binary-coded form
T	MW1	transfer the value into memory word MW1
LC	T1	load time T1 BCD-coded
T	MW3	transfer the value into memory word MW3

## 8.6 PC program input of timing functions

You must test all five timing functions SP, SE, SD, SS and SF with the following specified current values:



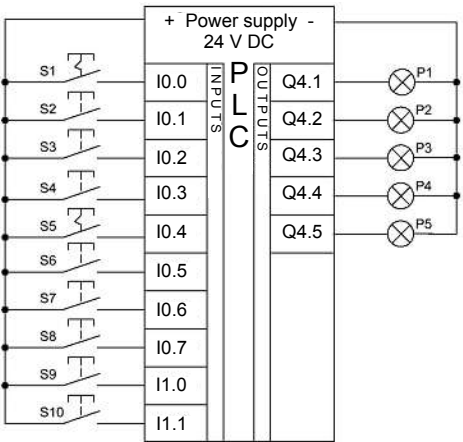
SP	T1	specified time = 1.5 s
SE	T2	specified time = 1.75 s
SD	T3	specified time = 2 s
SS	T4	specified time = 2.3 s
SF	T5	specified time = 3 s

- ☐ Program the timing functions in FBD
- ☐ Choose the function
- ☐ Organize OB 1
- ☐ Download the program into the CPU
- ☐ Test the program
- ☐ Change the methods of representation: > FBD > LAD > STL

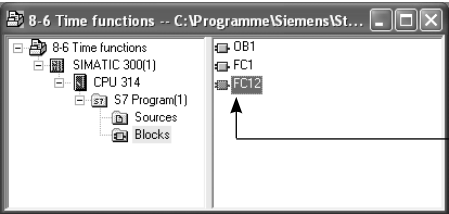
Table 8.2

Assignment list		
Symbol	Operand	Comment
S1	I0.0	Timer «SP»(T1) Switch ON
S2	I0.1	Timer «SP» (T1) Button reset
P1	Q4.1	Timer «SP»(T1) Indicator light
S3	I0.2	Timer «SE» (T2) Button ON
S4	I0.3	Timer «SE» (T2) Button reset
P2	Q4.2	Timer «SE» (T2) Indicator light
S5	I0.4	Timer «SD» (T3) Switch ON
S6	I0.5	Timer «SD» (T3) Button reset
P3	Q4.3	Timer «SD» (T3) Indicator light
S7	I0.6	Timer «SS» (T4) Button ON
S8	I0.7	Timer «SS» (T4) Button reset
P4	Q4.4	Timer «SS» (T4) Indicator light
S9	I1.0	Timer «SF» (T5) Button ON
S10	I1.1	Timer «SF» (T5) Button reset
P5	Q4.5	Timer «SF» (T5) Indicator light

Figure 8.14  
Connection to the PLC

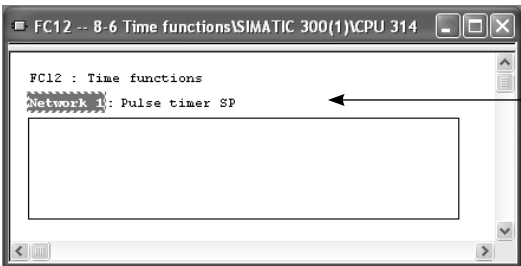


The preceding steps for configuration will no longer be explained in detail here.



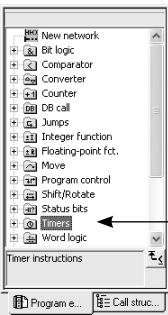
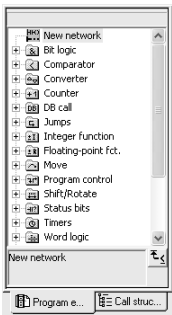
e.g. call FC 12

⇒ The following window is displayed:

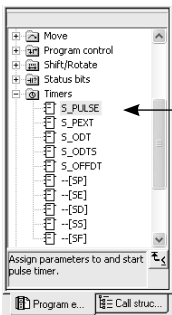


Insert network comments

Click on the **Program element: Overview on/off**

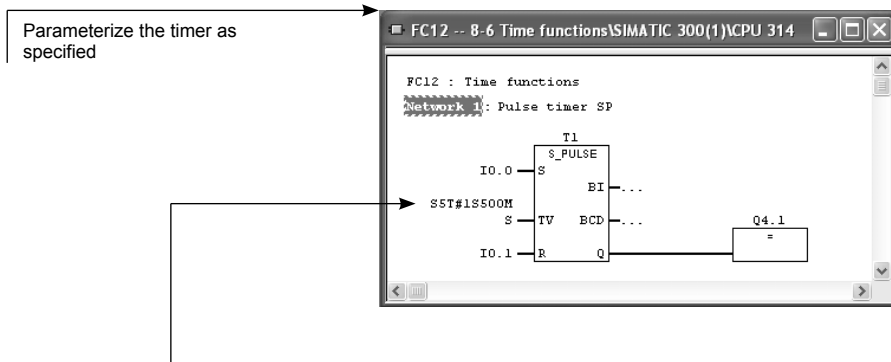


Click on the «+»-sign of bit combination **Timers**.



Click on **S\_PULSE** and drag and drop timer to the network 1 workspace.

## Parameterize timer SP



Enter the 1.5 s current value using the following characters:

**S5T#1.5s**

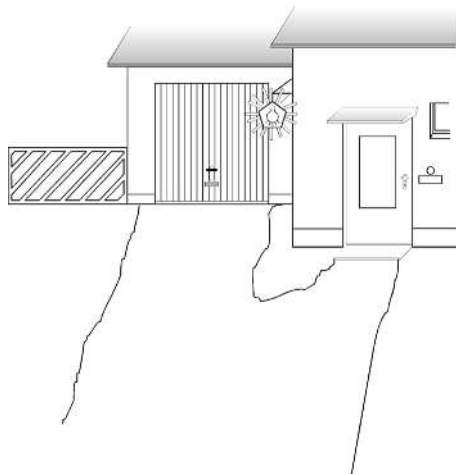
The important thing to remember when doing this is that you must enter a **period** after the number 1 and not a comma.

**Drag and drop** timers SE (S\_PEXT), SD (S\_ODT), SF (S\_OFFDT) and SS (S\_ODTS) to the workspace (insert one network for each timer).

Test the timing functions in online mode.

### 8.6.1 Garage lighting

Figure 8.15  
Garage lighting:  
Technology scheme



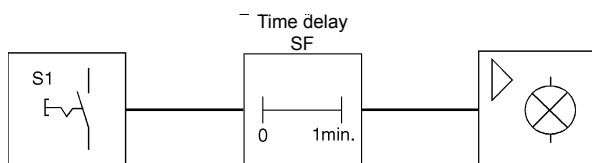


Figure 8.16  
Garage lighting:  
block diagram

### Description of Function

It is intended that the garage lighting (Figure 8.15) should not go out as soon as you press the switch; rather, it should go out a short while later so that users are not left stumbling in the dark. In this example, we choose a time of 1 minute (see Figure 8.16).

Table 8.3

Assignment list		
Symbol	Operand	Comment
S1	I0.0	System «On» – NO contact
P1	Q4.0	Indicator light

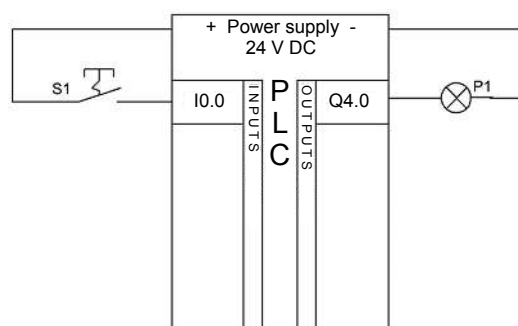


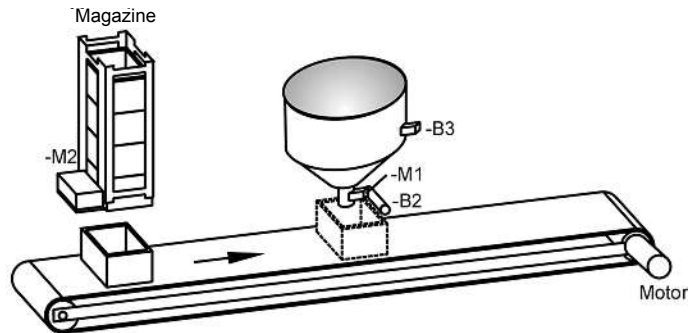
Figure 8.17  
Connection to the PLC

Proceed as follows:

- ☐ Program the garage lighting in FBD, LAD and STL
- ☐ Choose a function (we suggest FC 19)
- ☐ Organize OB 1
- ☐ Download the program into the CPU
- ☐ Test the program (**Switch on the CPU to RUN-P**: In this switch position, it is possible to change programs in online mode without needing to put the switch into the STOP position).
- ☐ Change the methods of representation: > FBD > LAD > STL

8.6.2 Filling system

Figure 8.18  
Filling system:  
Technology scheme



Description of Function

When you press the **Start** pushbutton, the conveyor belt transports crates that have been released from a store to below a silo. If sensor **B2** is contacted, the conveyor belt stops and the crate is filled with liquid on a time-dependent basis. If time-restricted supply of liquid is completed, i.e. the crate is full, it is transported on. While the **Start** pushbutton is pressed, the process keeps repeating until the silo is empty. When the silo is empty, a warning light lights up until the silo is filled up again.

Table 8.4

Assignment list		
Symbol	Operand	Comment
S1	I0.0	Switch «Start» - NO contact
B2	I0.1	Sensor «Crate under silo»
B3	I0.2	Message «Silo empty»
M1	Q4.0	Valve on silo
M2	Q4.1	Release for crates
Q11	Q4.2	Conveyor belt motor (power contactor)
P1	Q4.7	Warning light «Silo empty»

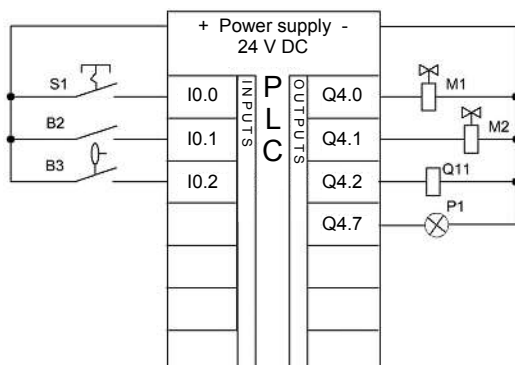


Figure 8.19  
Connection to the PLC

Proceed as follows:

- ☐ Program the filling system in FBD
- ☐ Choose a function (we suggest FC 20)
- ☐ Organize OB 1
- ☐ Download the program into the CPU
- ☐ Test the program (**Switch on the CPU to RUN-P**: In this switch position, it is possible to change programs in online mode without needing to put the switch into the STOP position).
- ☐ Change the methods of representation: > FBD > LAD > STL

### 8.6.3 Compressor system

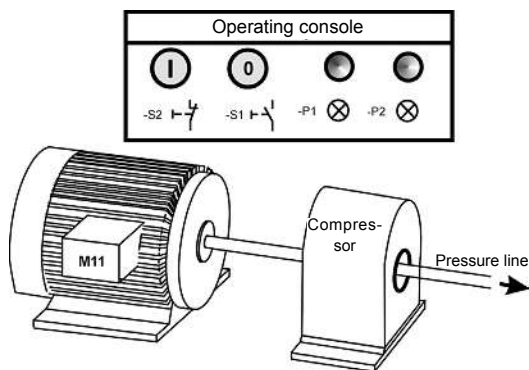


Figure 8.20  
Compressor system:  
Technology scheme

#### Description of Function

The compressor system (see Figure 8.20) is supplied with 400 V AC. An automatic star-delta connection ensures that the motor M11 starts up evenly. The compressor system is monitored by a temperature probe R10 and a pressure monitor for

water (S3) and a pressure monitor for air (S4). The monitoring circuit R10, S3 and S4 forms a safety chain. If one of the sensors takes on another switch position, motor M1 switches off, and indicator light P2 lights up (see Figure 8.21 and Figure 8.22).

Figure 8.21  
Main electric circuit

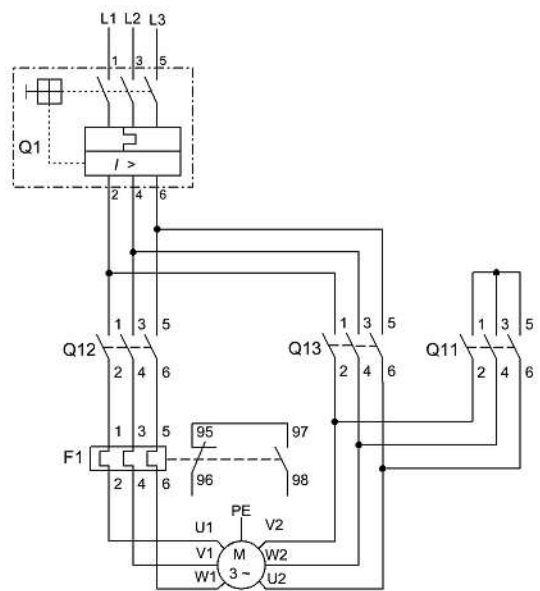


Figure 8.22  
Control circuit

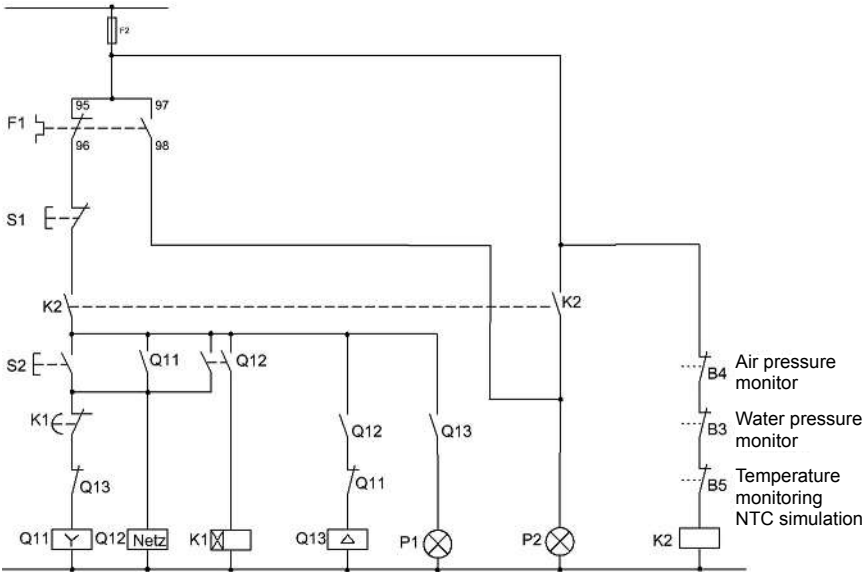


Table 8.5

Assignment list		
Symbol	Operand	Comment
S1	I0.0	Pushbutton NC contact - Plant «Off»
S2	I0.1	NO contact - Plant «On»
B3	I0.2	NC contact (water pressure monitor)
B4	I0.3	NC contact (air pressure monitor)
B5	I0.4	NC contact (temperature monitoring - NTC)
F1	I0.5	Overcurrent release for motor
Q11	Q4.0	Star contactor
Q12	Q4.1	Mains contactor
Q13	Q4.2	Delta contactor
P1	Q5.0	Indicator light for operation
P2	Q5.1	Indicator light for fault
K1	T1	Timer for switch-on delay of 2.5 seconds
K2	Bit memory	Auxiliary contact

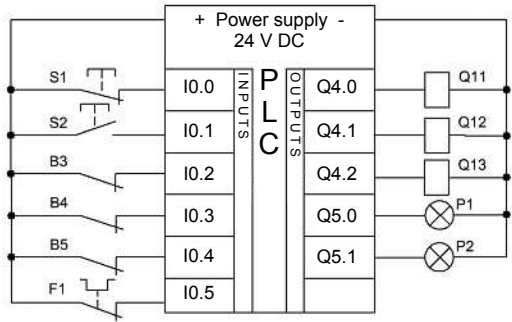


Figure 8.23  
Connection to the PLC

Proceed as follows:

- ☐ Program the compressor system in FBD
- ☐ Choose a function (we suggest FC 21)
- ☐ Organize OB 1
- ☐ Download the program into the CPU
- ☐ Test the program (**Switch on the CPU to RUN-P**: In this switch position, it is possible to change programs in online mode without needing to put the switch into the STOP position).
- ☐ Change the methods of representation: > FBD > LAD > STL



# 9 Clock Generators

Clock generators are used for a range of different inspection, monitoring and control jobs. In digital technology, they are referred to as **astable** flip-flops. In practical applications, you often need a flashing frequency for disturbance and operational messages. In the CPU S7-300, a parameterizable clock memory is already present that you can set at CPU configuration (see Section 5.1.9).

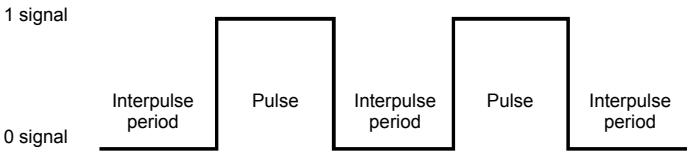
Clock memories are bit memories within a clock memory byte. If we consider memory byte MB100, for example, bit memories M100.0, M100.1, M100.2, M100.3...M100.7 are present in it. This means that the memory byte has 8 bits. Each of the individual bits in the memory byte can be polled and further-processed. Here, this memory byte is given the name clock memory byte MB100, since memory byte MB100 was stated at CPU configuration. A clock memory changes its value periodically as shown in the following table.

Table 9.1

Bit	.7	.6	.5	.4	.3	.2	.1	.0
Period (S)	2	1.6	1	0.8	0.5	0.4	0.2	0.1
Frequency (Hz)	0.5	0.625	1	1.25	2	2.5	5	10

A frequency of 1 Hz, for example, would mean that when using memory byte MB100 the system would have to poll bit memory M100.5. The clock memory clocks at a ratio of 1:1 (see Figure 9.1), i.e. the interpulse period is the same as the pulse length.

Figure 9.1  
Duty cycle 1:1



## 9.1 PC program input with clock generator

We want to test all 8 clock frequencies of memory byte MB100, which was configured as a clock memory byte in the CPU configuration (see chapter 5.1.9). Each pulse frequency is to be released with a switch and displayed on the module output byte QB4 (8 clock generators with a start-stop device and different frequencies).

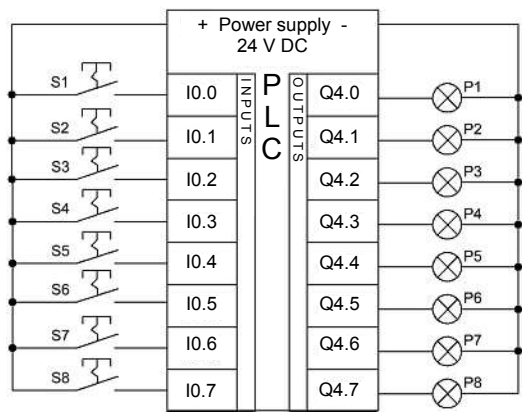
Proceed as follows:

- ☐ Program the clock generator in FBD
- ☐ Choose a function (we suggest FC 13)
- ☐ Organize OB 1
- ☐ Download the program into the CPU
- ☐ Test the program (**Switch on the CPU to RUN-P**: In this switch position, it is possible to change programs in online mode without needing to put the switch into the STOP position).
- ☐ Change the methods of representation: > FBD > LAD > STL

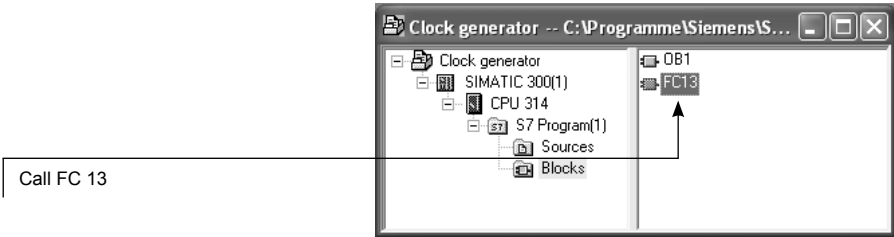
Table 9.2

Assignment list		
Symbol	Operand	Comment
S1	I0.0	Release 0.5 Hz
S2	I0.1	Release 0.625 Hz
S3	I0.2	Release 1 Hz
S4	I0.3	Release 1.25 Hz
S5	I0.4	Release 2 Hz
S6	I0.5	Release 2.5 Hz
S7	I0.6	Release 5 Hz
S8	I0.7	Release 10 Hz
P1	Q4.0	Indicator light for clock frequency 0.5 Hz
P2	Q4.1	Indicator light for clock frequency 0.625 Hz
P3	Q4.2	Indicator light for clock frequency 1 Hz
P4	Q4.3	Indicator light for clock frequency 1.25 Hz
P5	Q4.4	Indicator light for clock frequency 2 Hz
P6	Q4.5	Indicator light for clock frequency 2.5 Hz
P7	Q4.6	Indicator light for clock frequency 5 Hz
P8	Q4.7	Indicator light for clock frequency 10 Hz

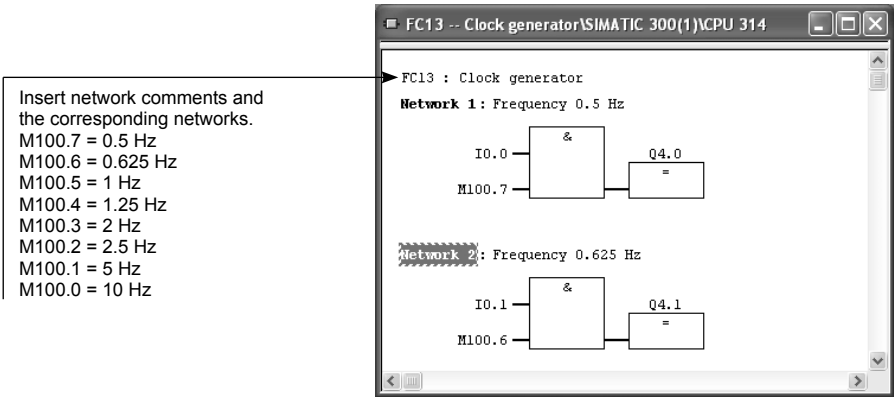
Figure 9.2  
Connection to the PLC



The preceding steps for creating a project will no longer be explained in detail here.



⇒ The following window is displayed:



All clock frequencies are tested in online mode.

9.1.1 Channel switch

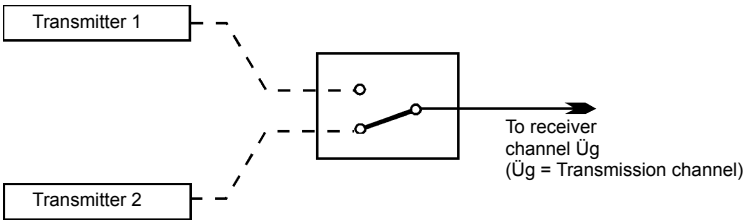


Figure 9.3 Channel switch: Technology scheme

Description of Function

Transmitting information on a digital basis is very secure and economic. Serial data transfer yields particularly favourable plant and operating costs. Figure 9.3 shows several pieces of information that come from different sources and are transmitted in chronological order via a transmission channel  $\bar{U}_g$  to the receiver. Figures 9.4 a) and 9.4 b) show the FBD and the connection to the PLC.

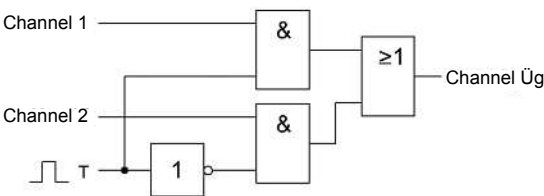


Figure 9.4  
a) FBD  
b) Connection to the PLC

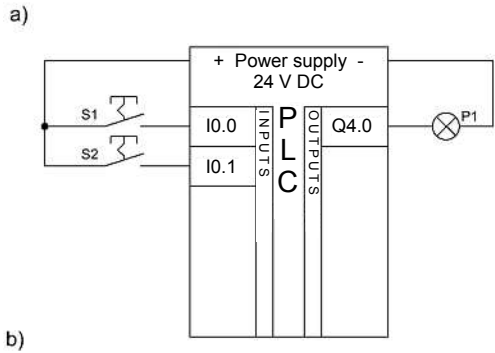


Table 9.3

Assignment list		
Symbol	Operand	Comment
S1	I0.0	Channel 1
S2	I0.1	Channel 2
T	M100.0	Pulse frequency
P1	Q4.0	Indicator light

Proceed as follows:

- ☐ Program the channel switch in FBD
- ☐ Choose a function (we suggest FC 22)
- ☐ Organize OB 1
- ☐ Download the program into the CPU
- ☐ Test the program (**Switch on the CPU to RUN-P:** In this switch position, it is possible to change programs in online mode without needing to put the switch into the STOP position).
- ☐ Change the methods of representation: > FBD > LAD > STL

### 9.1.2 Paging system

#### Description of Function

By means of a paging system with only one lamp, it is possible to page three different people. Using switch S1, you switch the lamp on; person 1 is paged. Using switch S2 and switch S3, you switch the lamp on **intermittent**. A clock generator determines the light on and light off periods. Figures 9.5 a) and 9.5 b) show the associated FBD and the PLC connection.

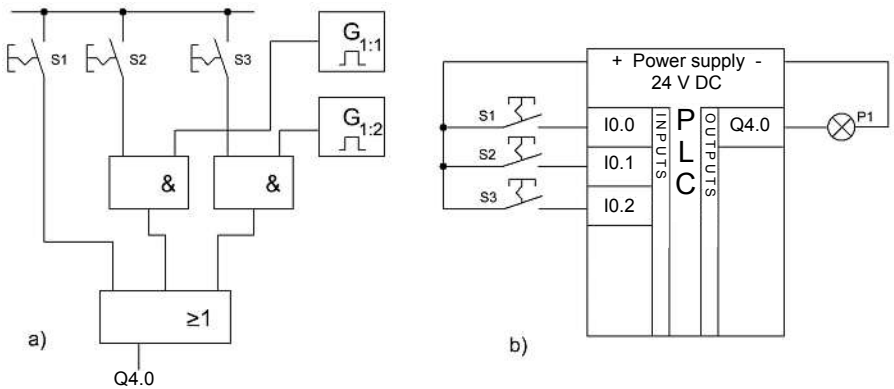


Figure 9.5 a) FBD, b) Connection to the PLC

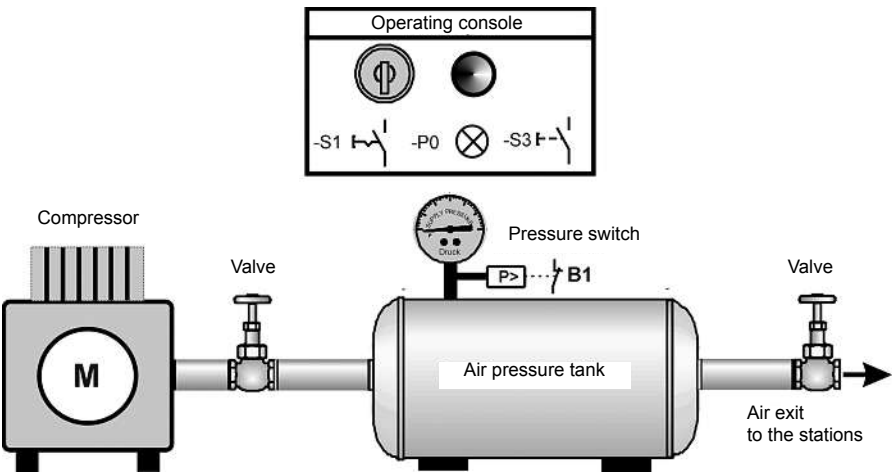
Table 9.4

Assignment list		
Symbol	Operand	Comment
S1	I0.0	«NO contact» switch
S2	I0.1	«NO contact» switch
S3	I0.2	«NO contact» switch
P1	Q4.0	Indicator light

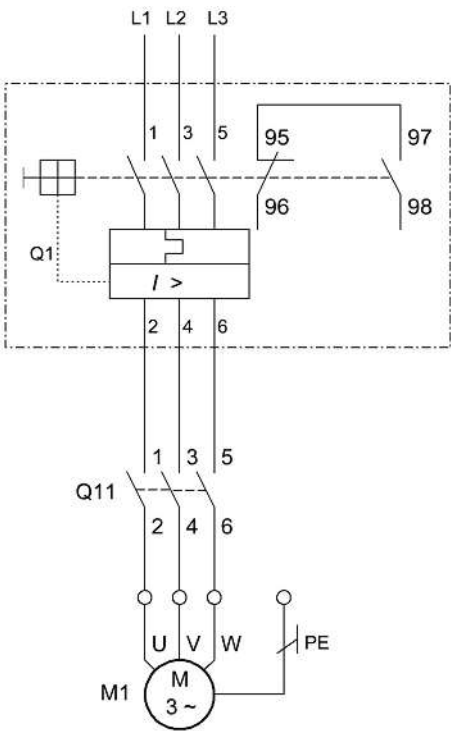
Proceed as follows:

- ☐ Program the paging system in FBD
- ☐ Choose a function (we suggest FC 23)
- ☐ Organize OB 1
- ☐ Download the program into the CPU
- ☐ Test the program (**Switch on the CPU to RUN-P**: In this switch position, it is possible to change programs in online mode without needing to put the switch into the STOP position).
- ☐ Change the methods of representation: > FBD > LAD > STL

9.1.3 Air supply



a)



b)

Figure 9.6 Air supply: Technology scheme, b) Main electric circuit

**Description of Function**

The system is for supplying air to an assembly line. The plant is switched-on with switch S1. The compressor pumps air into the container. The pressure in this container is monitored by pressure switch S2. At the maximum pressure, S2 opens; at the minimum pressure, S2 closes; when air is removed, the pressure drops in the air container. If the pressure switch falls below S2 min. (S2 closed), an air request is received. The compressor starts up and switches off again on reaching the maximum pressure (S2 open). If the compressor is switched-on, this is shown by indicator light P0. Using pushbutton S3, you carry out the lamp test. If there is a compressor fault, the steady light changes to a flashing light with a frequency of 1 Hz (interpulse period 1:1).

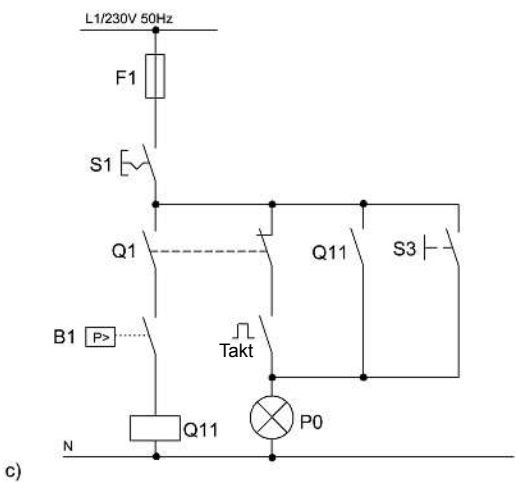


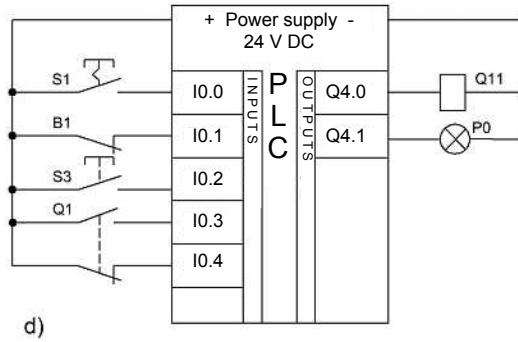
Figure 9.6  
Air supply:  
c) Circuit diagram

Table 9.5

Assignment list		
Symbol	Operand	Comment
S1	I0.0	Air supply system «On» – NO contact
B1	I0.1	Pressure switch P – NC contact
S3	I0.2	Lamp test – NO contact
Q1	I0.3	Motor relay switch auxiliary contactor – NO contact
Q1	I0.4	Motor relay switch auxiliary contactor – NC contact
Pulse frequency	M100.0	Clock memory
P0	Q4.1	Indicator light for fault – steady light/flashing light
Q11	Q4.0	Power contactor for compressor



Figure 9.6  
Air supply:  
d) Connection to the PLC



Proceed as follows:

- ☐ Program the air supply in FBD
- ☐ Choose a function (we suggest FC 24)
- ☐ Organize OB 1
- ☐ Download the program into the CPU
- ☐ Test the program (**Switch on the CPU to RUN-P**: In this switch position, it is possible to change programs in online mode without needing to put the switch into the STOP position).
- ☐ Change the methods of representation: > FBD > LAD > STL

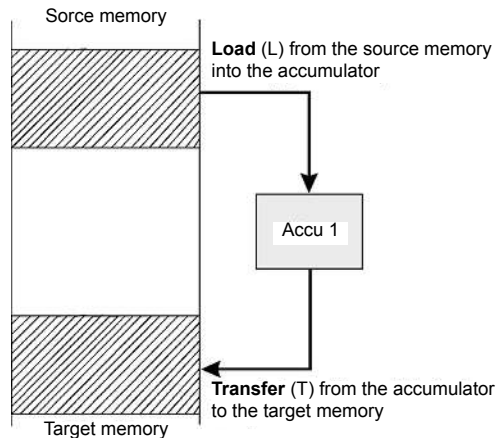


## 10 Counters

If you want to exchange information in the STEP 7 programming language between the process input image (PII) and the process output image (PIQ) as well as to further-process the timer and counter values, the comparison and calculation operations, you must work with the load command (L) and the transfer command (T). Exchanging information does not function directly; rather, it always flows via the accumulator. The accumulator is a special register in the processor and functions as a buffer. The load and transfer functions (see Figure 10.1) are always executed independent of the result of logic operation (RLO).

### 10.1 Load and transfer functions

Figure 10.1  
Principle:  
directional flow of information



At loading, the system copies the contents of the addressed source memory and writes it to the accumulator. When doing this, the contents of the accumulator up to now are overwritten.

At transfer, the system copies the contents of the accumulator and writes them to the addressed target memory (see Figure 10.2). Since the accumulator is only copied, it is available for further transfer operations.

Example STL:

L IW0

T QW4

L +5

T QW6

BE

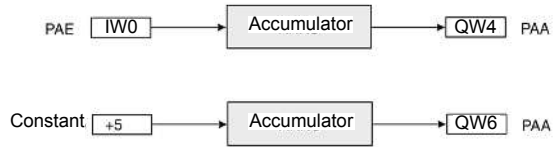


Figure 10.2 Flow of information: Schematic representation

Loading and transferring are unconditional operations that are executed in every cycle regardless of the RLO.

## 10.2 Counter functions

In control engineering, counter functions are needed to acquire piece counts or pulses and to evaluate times and distances. With the SIMATIC S7, counters are integrated in the CPU. The counting range of the counter is between 0 and 999. In the S7-CPU 314 used here, 64 counters are integrated to which the functions listed below (see Figures 10.3a and 10.3b) can be assigned.

### 10.2.1 Release a counter (FR)

A positive edge change from 0 to 1 in the RLO of the release operation (FR) releases one counter. No release is requested for setting a counter or for normal counter operations. However, if you want to set a counter before the corresponding count operation (CU, CD or S) or count forwards or backwards, you can do this with a release. This is only possible, however, if the RLO bit has a signal state of 1 before the corresponding operation (CU, CD or S). The release operation (FR) is only present in the STL type of representation.

### 10.2.2 Counting forwards

The value of the addressed counter is incremented by 1. The function only becomes effective with a positive edge change of the logic operation programmed before CU. If the count value reaches the top limit of 999, it is no longer incremented. A carry is not formed.

### 10.2.3 Counting backwards

The value of the addressed counter is decremented by 1. The function only becomes effective with a positive edge change of the logic operation programmed before CD. If the count value reaches the bottom limit of 0, it is no longer decremented.

### 10.2.4 Set counter

You can set the counter to a specific value. From this value onwards, it is possible to count forwards or backwards.

If, for example, you want to set the counter to the value 8:

#### STL

A	I0.0	Polling the signal state
L	+5	Loading the count value
S	C1	Setting the counter with the loaded count value (with a positive edge change)

### 10.2.5 Count value definition

If a counter is set, the system uses the contents of accumulator 1 as the count value. It is possible to load the count value either in binary-coded form or BCD-coded. The following operations are possible:

<input type="checkbox"/> Input word	IW..
<input type="checkbox"/> Output word	QW..
<input type="checkbox"/> Memory word	MW..
<input type="checkbox"/> Data word	DBW/DIW..
<input type="checkbox"/> Local data word	LW..
<input type="checkbox"/> Constants	DW#16#, +8, B#16#, 2#...etc.

### 10.2.6 Reset counter (R)

In the case of a result of logic operation RLO 1, the counter is set to 0 (reset). An RLO 0 does not affect the counter. Resetting of a counter has a static effect. If a reset condition is fulfilled, it is not possible to carry out setting or counting.

### 10.2.7 Poll count value (L/LC)

A count value is stored in binary-coded form in a counter word. The value in the counter can be loaded as a dual number (BI) or as a BCD number (BCD) into the accumulator and transferred from there to other operand areas.

With STL programming, it is possible to choose between **L Z1** (load counter 1) for polling as a dual number and **LC C1** (load counter 1 in coded form) for polling the BCD number.

10.2.8 Poll signal state of counter (binary)

It is possible to poll the counter for its signal state:

- ❑ Signal state 0: Counter is at a value of 0
- ❑ Signal state 1: Counter is ready to count

You can poll the signal status conditions with A C1, AN C1, ON C1 etc. and use them for further logic operations.

Release FR (only in STL)  
A I0.7  
FR C1

Counting forwards  
A I0.0  
CU C1

Counting backwards  
A I0.1  
CD C1

Set  
A I0.2  
L8  
S 1

Reset  
A I0.3  
R C1

Requests (digital)  
L C1 (dual coded)  
T MW2  
L C1  
T QW4

Request (binary)  
A C1  
= Q4.0

Figure 10.3a  
Programming the counter

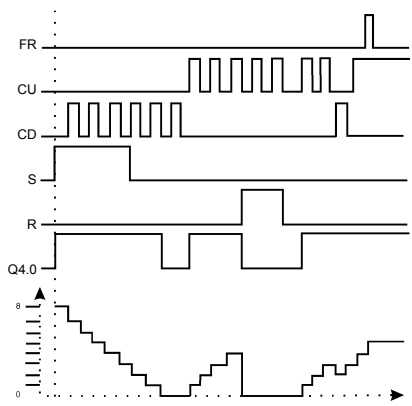
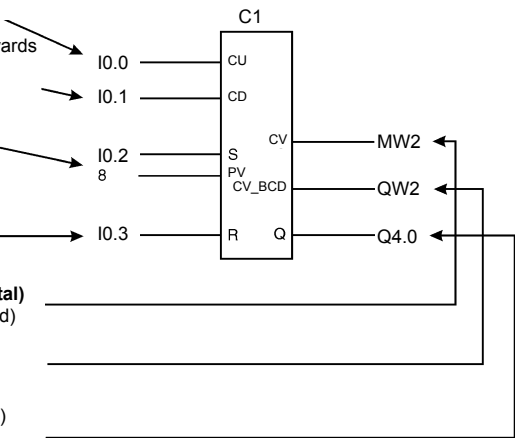
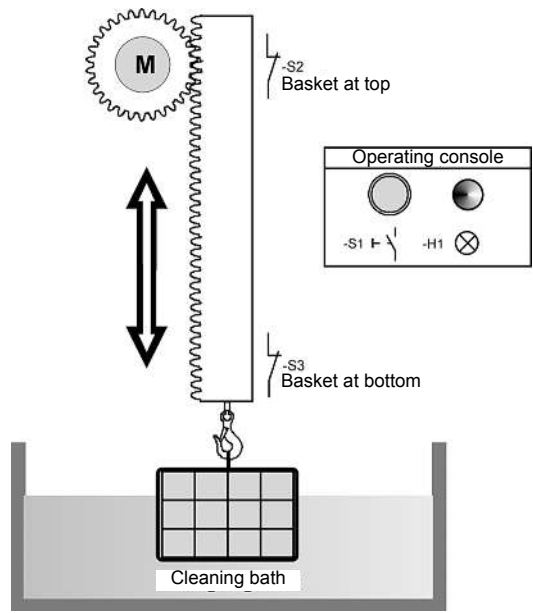


Figure 10.3b Function chart

### 10.3 PC program input cleaning bath

Figure 10.4  
Cleaning bath:  
Technology scheme



#### Description of Function

A cleaning bath (see Figure 10.4) is intended to remove the flux from soldered PCBs. A basket is hooked in and unhooked manually. If you activate pushbutton S1, the basket automatically enters the cleaning bath five times and stays there for 15 s in each case to clean the boards. After the cleaning process, the basket can be removed or be filled up and lowered again. The signal lamp shows automatic mode.

Table 10.1

Assignment list		
Symbol	Operand	Comment
S1	I0.0	«Start» pushbutton
B2	I0.1	Limit switch at top – NC contact
B3	I0.2	Limit switch at bottom – NC contact
Q11	Q4.0	Motor downwards
Q12	Q4.1	Motor upwards
P1	Q4.2	Indicator light for automatic mode

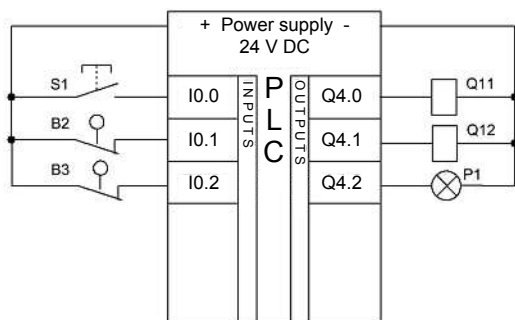


Figure 10.5  
Connection to the PLC

Proceed as follows:

- ☐ Program the cleaning bath in FBD
- ☐ Choose a function (we suggest FC 25)
- ☐ Organize OB 1
- ☐ Download the program into the CPU
- ☐ Test the program (**Switch on the CPU to RUN-P**: In this switch position, it is possible to change programs in online mode without needing to put the switch into the STOP position).
- ☐ Change the methods of representation: > FBD > LAD > STL



# 11 Comparators

Using the comparison functions, the system directly compares two digital values with one another that are located in accumulators 1 and 2.

However, the values to be compared must first be saved to the accumulator using load instructions.

The length of the operands (byte, word) is stated in conjunction with the operands, e.g. IB, IW, QB, MB, MW, etc.

The result of the comparison is binary. A comparison is conditional upon both numbers having the same format.

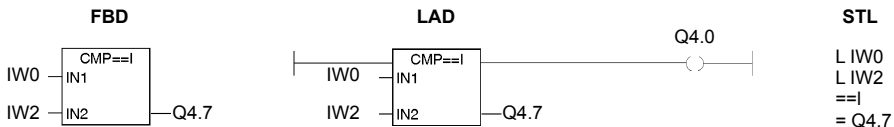
The following pairs of numeric values can be compared:

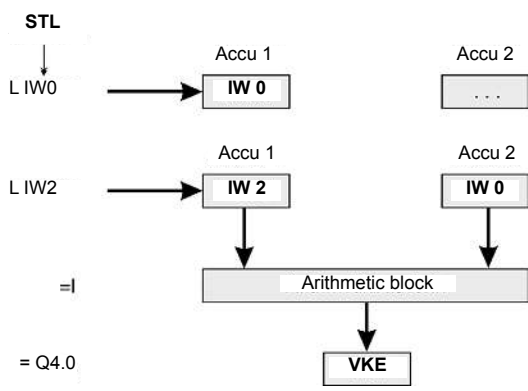
- ☐ 2 integers (16-bit, symbol: I)
- ☐ 2 integers (32-bit, symbol: D)
- ☐ 2 real numbers (floating point numbers 32-bit, symbol: R)

## 11.1 Comparison functions

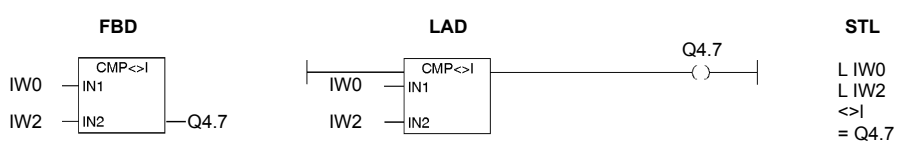
In the STEP 7 programming language, you can choose six different comparisons:

### 11.1.1 Equal to ==

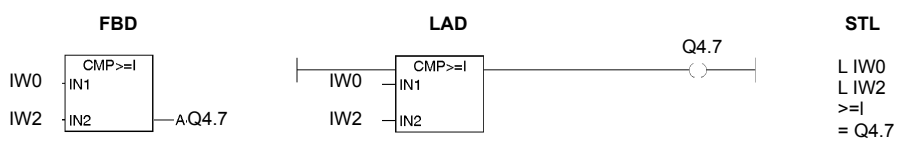




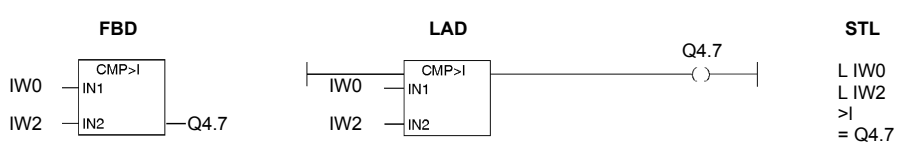
### 11.1.2 Not equal <>



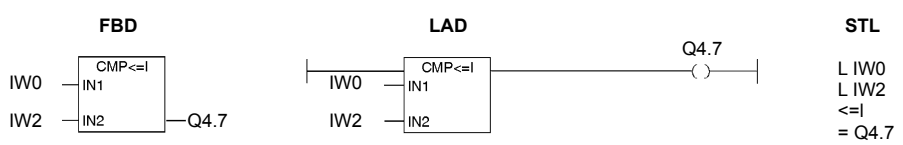
### 11.1.3 Greater than equal to >=



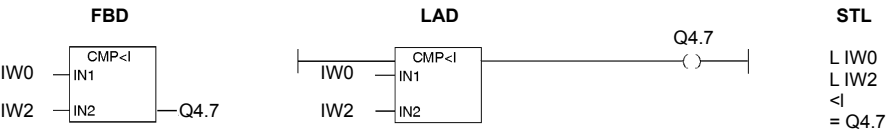
### 11.1.4 Greater than >



### 11.1.5 Less than equal to <=



11.1.6 Less than <



11.2 PC program input runway light

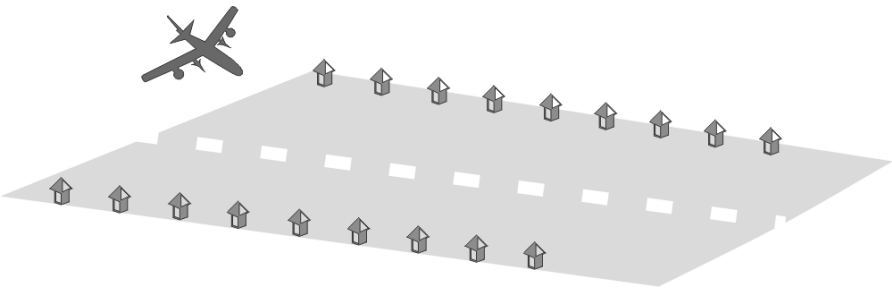


Figure 11.1 Runway light: Technology scheme

Description of Function

At an airstrip, it is intended to place nine lamps (runway lights) on the left and on the right next to the runway. These runway lights are intended to allow pilots to land safely after dark. The runway lights are switched-on from the control tower using switch S1.

Table 11.1

Assignment list		
Symbol	Operand	Comment
S1	E0.0	Runway lights "ON" – NO contact
P1	A4.0	Indicator light P1 for runway lights
P2	A4.1	Indicator light P2 for runway lights
P3	A4.2	Indicator light P3 for runway lights
P4	A4.3	Indicator light P4 for runway lights
P5	A4.4	Indicator light P5 for runway lights
P6	A4.5	Indicator light P6 for runway lights
P7	A4.6	Indicator light P7 for runway lights
P8	A4.7	Indicator light P8 for runway lights
P9	A5.0	Indicator light P9 for runway lights

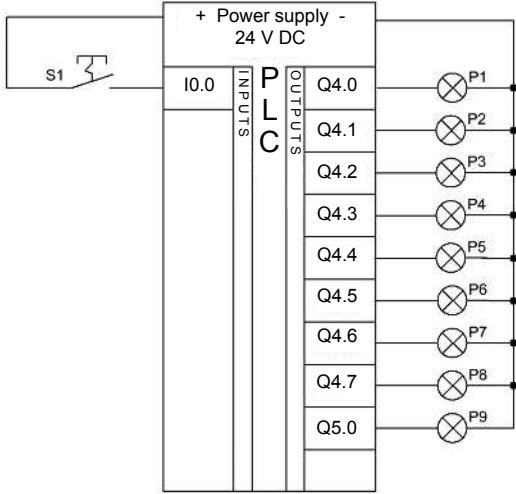


Figure 11.2  
Connection to the PLC

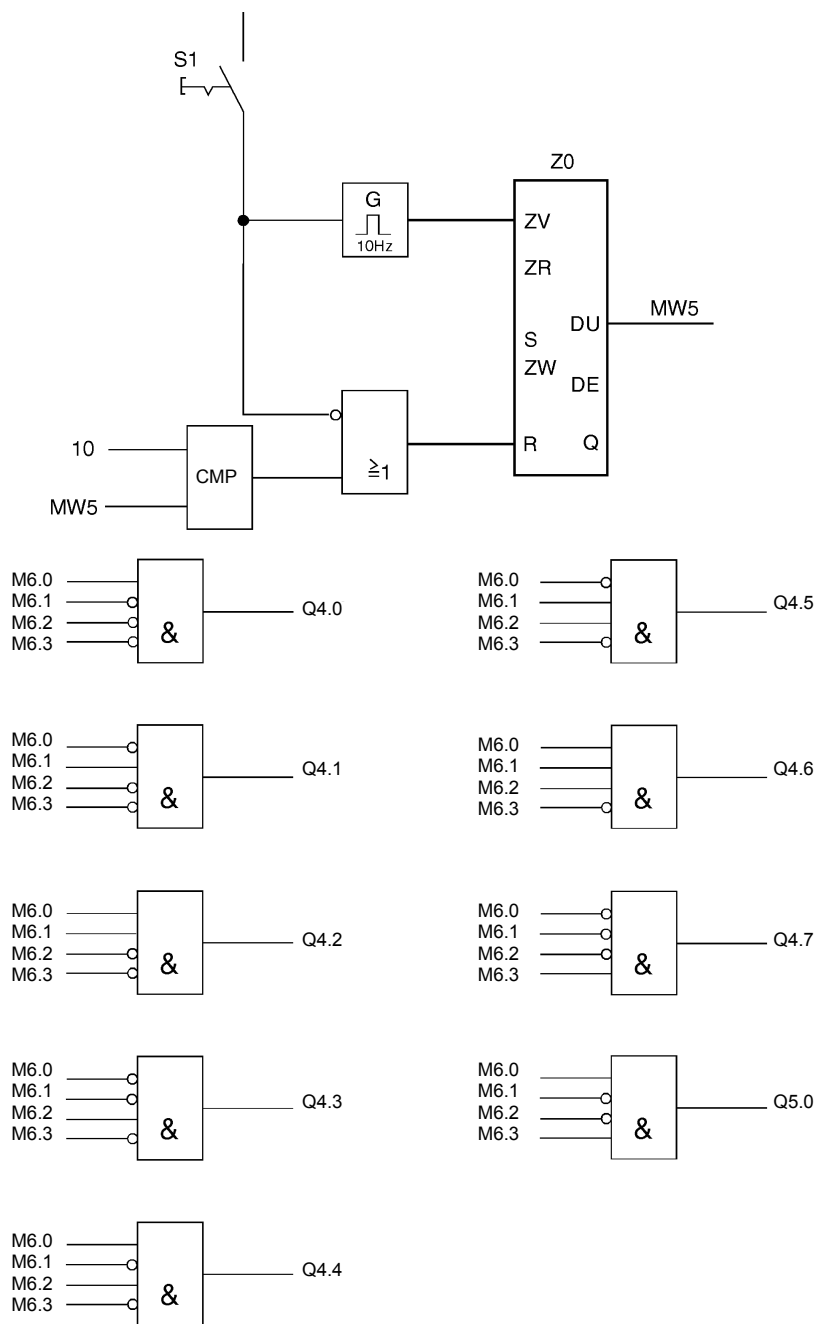


Figure 11.3 FBD

Proceed as follows:

- ☐ Program the runway lights in FBD
- ☐ Choose a function (we suggest FC 26)
- ☐ Organize OB 1
- ☐ Download the program into the CPU
- ☐ Test the program (**Switch on the CPU to RUN-P**: In this switch position, it is possible to change programs in online mode without needing to put the switch into the STOP position).
- ☐ Change the methods of representation: > FBD > LAD > STL

### 11.3 Program input sequence function

#### Description of Function

Using switch S1, you activate the clock memory of the CPU 1 Hz (Start-Stop facility). The cycle is counted in a counter C1. On reaching three clock pulses, it is intended that output Q4.0 carries a 1 signal. On reaching five clock pulses, it is intended that output Q4.1 carries a 1 signal and on reaching seven clock pulses, it is intended that output Q4.2 carries a 1 signal. On switching off switch S1, it is intended that all the outputs be set to 0. The process can start again from the beginning.

Table 11.2

Assignment list		
Symbol	Operand	Comment
S1	I0.0	«Start» switch
P1	Q4.0	Indicator light P1
P2	Q4.1	Indicator light P2
P3	Q4.2	Indicator light P3

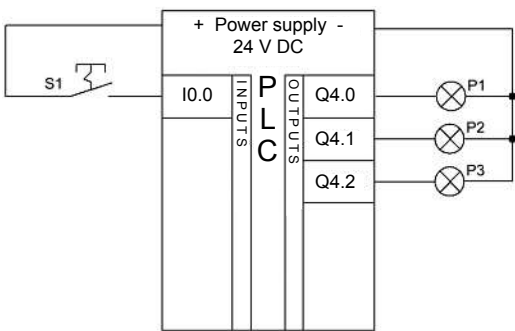


Figure 11.4  
Connection to the PLC

Proceed as follows:

- ☐ Program the sequence function in FBD
- ☐ Choose a function (we suggest FC 27)
- ☐ Organize OB 1
- ☐ Download the program into the CPU
- ☐ Test the program (**Switch on the CPU to RUN-P:** In this switch position, it is possible to change programs in online mode without needing to put the switch into the STOP position).
- ☐ Change the methods of representation: > FBD > LAD > STL

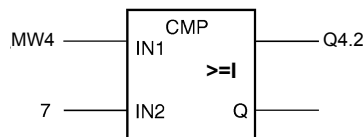
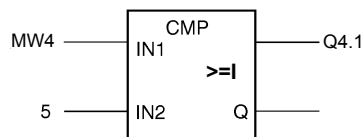
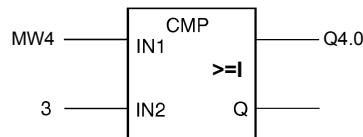
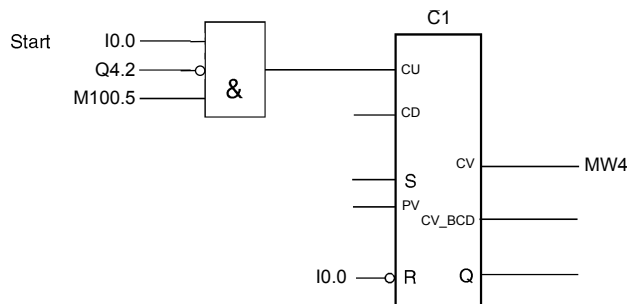


Figure 11.5 FBD





# 12 Practical Examples with Simulators

## 12.1 Seven-segment display

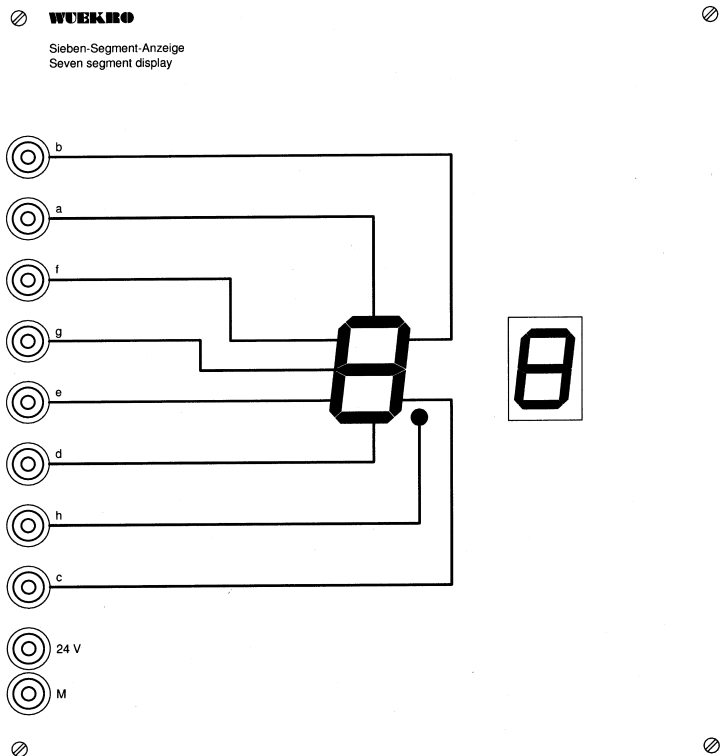


Figure 12.1 Seven-segment display: Technology scheme

### Description of Function

The control procedures for converting the numbers 0...9 to BCD code (8-4-2-1 code) for representation using a seven-segment display are intended to be implemented using a programmable logic controller. The signals of numbers 0...9 are entered in the PLC using switches S1...S4 in the BCD code. It is intended to display

on the seven-segment display the corresponding digits 0...9. To do this, the system must trigger the appropriate segments for each digit.

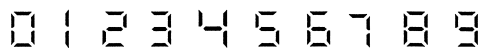


Figure 12.2  
Assignment of segments to the decimal digits

Table 12.1

Assignment list		
Symbol	Operand	Comment
S1	I0.0	Binary position 1-value 1
S2	I0.1	Binary position 2-value 2
S3	I0.2	Binary position 4-value 4
S4	I0.3	Binary position 8-value 8
a	Q4.0	Seven-segment display - Segment a
b	Q4.1	Seven-segment display - Segment b
c	Q4.2	Seven-segment display - Segment c
d	Q4.3	Seven-segment display - Segment d
e	Q4.4	Seven-segment display - Segment e
f	Q4.5	Seven-segment display - Segment f
g	Q4.6	Seven-segment display - Segment g
M0.0	M0.0	Evaluation of number 0
M0.1	M0.1	Evaluation of number 1
M0.2	M0.2	Evaluation of number 2
M0.3	M0.3	Evaluation of number 3
M0.4	M0.4	Evaluation of number 4
M0.5	M0.5	Evaluation of number 5
M0.6	M0.6	Evaluation of number 6
M0.7	M0.7	Evaluation of number 7
M1.0	M1.0	Evaluation of number 8
M1.1	M1.1	Evaluation of number 9

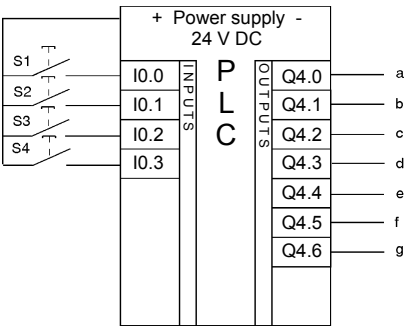
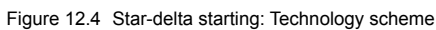


Figure 12.3  
Connection to the PLC

- ❑ Program the seven-segment display in FBD, LAD and STL
- ❑ Choose a function (we suggest FC 0)
- ❑ Organize OB 1
- ❑ Download the program into the CPU
- ❑ Test the program (**Switch on the CPU to RUN-P:** In this switch position, it is possible to change programs in online mode without needing to put the switch into the STOP position).
- ❑ Change the methods of representation: > FBD > LAD > STL

The control procedures for automatic star-delta starting of a three-phase asynchronous motor are intended to be implemented using a programmable logic controller.



**Function or problem description**

A three-phase asynchronous motor is intended to be started up via a star-delta connection. On activating pushbutton S1 (NO contact), the motor is intended to start up in a star connection (contactor Q11 and star contactor Q12 ON), and then switch to a delta connection (star contactor Q12 OFF and delta contactor Q13 ON). On activating pushbutton S0 (NC contact), the controller is immediately switched to the idle state. When the overcurrent relay F2 (NC contact) operates, the controller is also deactivated via the PLC (see Figure 12.4).

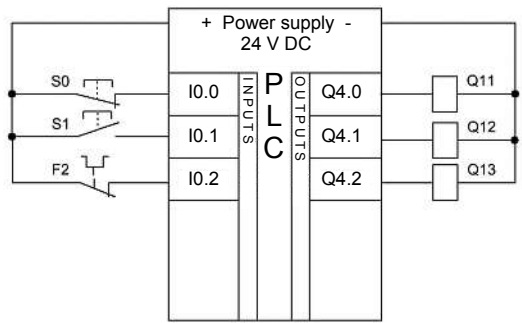


Figure 12.5  
Connection to the PLC

Table 12.2

Assignment list		
Symbol	Operand	Comment
S0	I0.0	OFF button (NC contact)
S1	I0.1	ON button (NO contact)
F2	I0.2	Overcurrent relay (NC contact)
Q11	Q4.0	Mains contactor
Q12	Q4.1	Star contactor
Q13	Q4.2	Delta contactor
T1	T1	Wait time
M1	M0.0	Bit memory for start of T1

Proceed as follows:

- ☐ Program the star-delta connection in FBD, LAD and STL
- ☐ Choose a function (we suggest FC 1)
- ☐ Organize OB 1
- ☐ Download the program into the CPU
- ☐ Test the program (**Switch on the CPU to RUN-P:** In this switch position, it is possible to change programs in online mode without needing to put the switch into the STOP position).
- ☐ Change the methods of representation: > FBD > LAD > STL

## 12.3 Traffic light controller

The control procedures for the traffic lights at a pedestrian crossing are intended to be implemented using a programmable logic controller for daytime and nighttime operation.

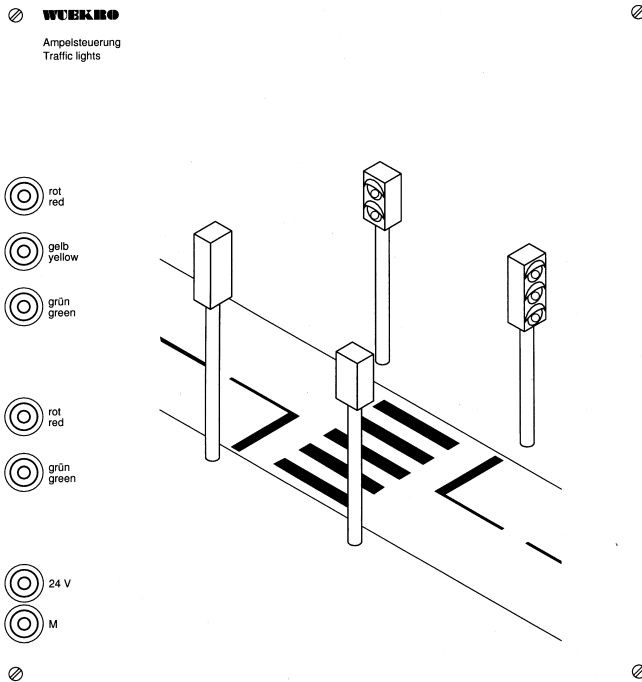


Figure 12.6 Traffic light controller: Technology scheme

### Description of Function

The traffic lights for a pedestrian crossing are controlled in two operating modes:

- ☐ Daytime operation,
- ☐ Nighttime operation.

### Daytime operation

The traffic lights switch automatically. In this connection, a set switching cycle repeats continuously. The switchboards of the traffic lights within a switching cycle are shown in the following pulse time diagram (see Figure 12.7). The assumed clock frequency is 0.5 Hz.

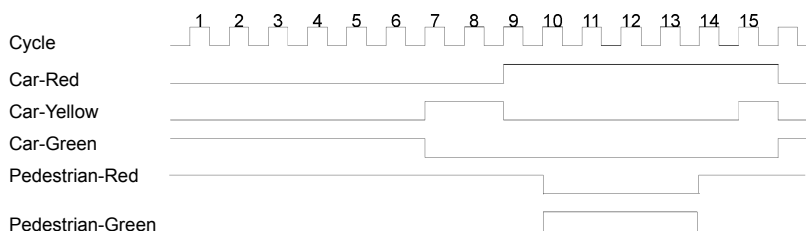


Figure 12.7 Pulse time diagram

### Nighttime operation

You set nighttime operation using switch S0. When you do this, the switching cycle of daytime operation is interrupted immediately and the yellow signal lamp of the driver's traffic light operates at a flashing rate of 0.5 Hz.

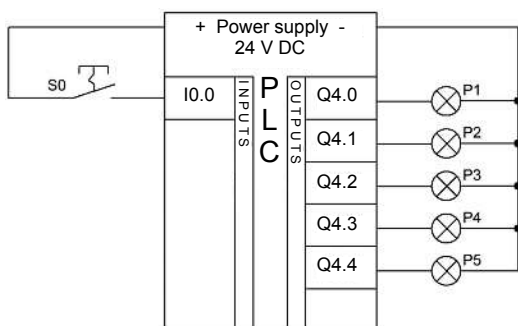


Figure 12.8  
Connection to the PLC

Table 12.3

Assignment list		
Symbol	Operand	Comment
S0	I0.0	Day-Night switch
P1	Q4.0	Car – Red
P2	Q4.1	Car– Yellow
P3	Q4.2	Car– Green
P4	Q4.3	Pedestrian – Red
P5	Q4.4	Pedestrian – Green
Q5.0	Q5.0	Counter value 1
Q5.1	Q5.1	Counter value 2
Q5.2	Q5.2	Counter value 3
Q5.3	Q5.3	Counter value 4
C1	C1	Cycle counter
MB1	MB1	Cycle memory
M2	M2.0	Bit memory for reset counter
M3	M100.7	Clock memory 0.5 Hz

Proceed as follows:

- ☐ Program the traffic light controller in FBD, LAD and STL
- ☐ Choose a function (we suggest FC 2)
- ☐ Organize OB 1
- ☐ Download the program into the CPU
- ☐ Test the program (**Switch on the CPU to RUN-P:** In this switch position, it is possible to change programs in online mode without needing to put the switch into the STOP position).
- ☐ Change the methods of representation: > FBD > LAD > STL

## 12.4 Conveyor belt controller

The control procedures for a conveyor belt system (see Figure 12.9) are intended to be implemented using a programmable logic controller.

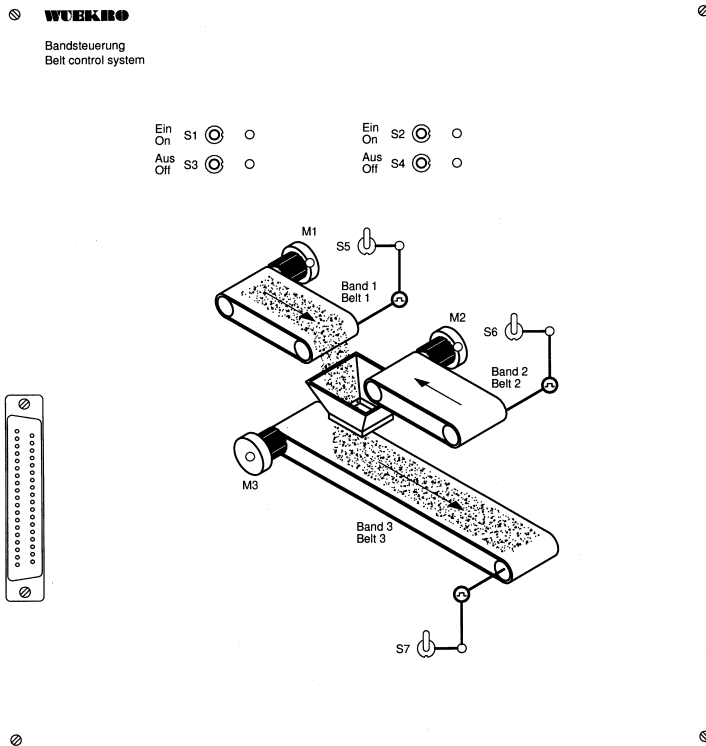


Figure 12.9 Conveyor belt controller: Technology scheme

**Description of Function**

Using pushbuttons **S1...S4**, it is intended to switch on and off conveyor belts 1 and 2. One ON and OFF lamp each will show the operating status. Belts 1 and 2 may not carry out conveying at the same time. Belt 3 is intended to automatically start up whenever belt 1 or 2 is carrying out conveying.

Belt monitors signal the movement of the conveyor belts at a pulse frequency of 10 Hz. This frequency is specified by a clock memory in the CPU and transferred to the simulator. If the belt monitor pulses fail (standstill or conveyor break), the encoders supply the signal state 0.

During the three-second long start-up stage, the belt monitor pulses should not be evaluated. After activating the OFF buttons, you should run empty belts 1 and 2 for another 2 s and belt 3 for another 6 s before switching off. If the monitor signals of belt 1 or 2 fail during operation, you should immediately switch off the respective drive while belt 3 keeps running empty for another 6 s before being switched off. The OFF lamp of belt 1 or 2 should indicate the disturbance by flashing in a 2-Hz-cycle (clock memory in the CPU!) If the belt monitor pulses of belt 3 fail, all the drives must switch off immediately.

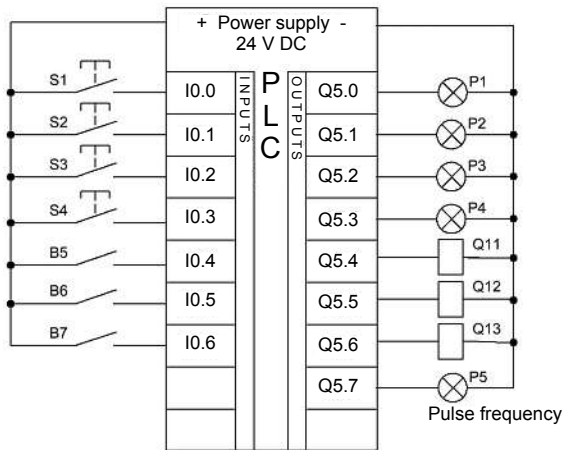


Figure 12.10  
Connection to the PLC



Table 12.4

Assignment list		
Symbol	Operand	Comment
S1	I0.0	ON button belt 1
S2	I0.1	ON button belt 2
S3	I0.2	OFF button belt 1
S4	I0.3	OFF button belt 2
B5	I0.4	Monitor belt 1
B6	I0.5	Monitor belt 2
B7	I0.6	Monitor belt 3
P1	Q5.0	ON lamp belt 1
P2	Q5.1	ON lamp belt 2
P3	Q5.2	OFF lamp belt 1
P4	Q5.3	OFF lamp belt 2
Q11	Q5.4	Drive M1 belt 1
Q12	Q5.5	Drive M1 belt 2
Q13	Q5.6	Drive M1 belt 3
P5 (Pulse frequency)	Q5.7	10-Hz clock-pulse rate
T0	T0	Time for starting time: 3 seconds
T1	T1	Time for belt monitoring belt 1/2: 0.12 seconds
T2	T2	Time for belt monitoring belt 3: 0.12 seconds
T5	T5	Time for switch-off delay belt 1: 2 seconds
T6	T6	Time for switch-off delay belt 2: 2 seconds
T7	T7	Time for switch-off delay belt 3: 6 seconds
Memory 1	M100.3	Memory for 2-Hz clock generator
Memory 2	M100.0	Memory for 10-Hz clock generator

Proceed as follows:

- ☐ Program the conveyor belt controller in FBD, LAD and STL
- ☐ Choose a function (we suggest FC 4)
- ☐ Organize OB 1
- ☐ Download the program into the CPU
- ☐ Test the program (**Switch on the CPU to RUN-P**: In this switch position, it is possible to change programs in online mode without needing to put the switch into the STOP position).
- ☐ Change the methods of representation: > FBD > LAD > STL

## 12.5 Reaction vessel

The control procedures for a chemical process in a reaction vessel (see Figure 12.11) are intended to be implemented using a programmable logic controller.

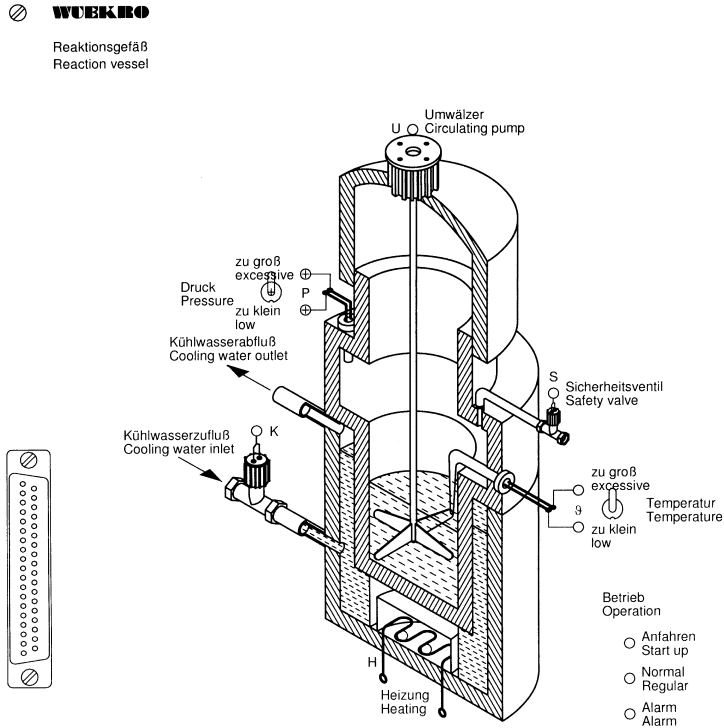


Figure 12.11 Reaction vessel: Technology scheme

### Description of Function

In a reaction vessel, it is intended that a chemical process run at a specific temperature and at a specific pressure. The reaction vessel has a temperature probe and a pressure gauge for measuring the temperature and pressure. The temperature and pressure are controlled by a heater H, a cooling water intake K and a safety valve S. The actuators of the reaction vessel should allow the following switch-on conditions:

Safety valve S:  
Cooling water intake K:  
Heater H:

pressure P too high  
temperature too high  
temperature too low or P too low and temperature normal

If actuators cooling water intake K or heater H are switched-on, the circulator must be activated.

The sequence of the reaction is divided into three operating status conditions:

1. Start-up:
2. Normal:
3. Alarm:
- P too low  
P normal  
P too high

The technology scheme shows the arrangement of the encoders and actuators.

Figure 12.12  
Connection to the PLC

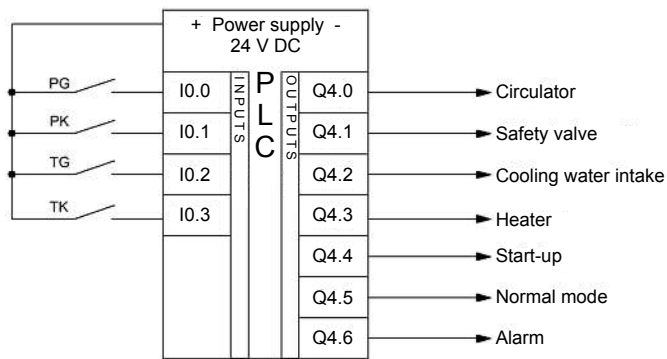


Tabelle 12.5

Assignment list		
Symbol	Operand	Comment
PG	I0.0	Pressure too high
PK	I0.1	Pressure too low
TG	I0.2	Temperature too high
TK	I0.3	Temperature too low
U	Q4.0	Circulator
S	Q4.1	Safety valve
K	Q4.2	Cooling water intake
H	Q4.3	Heater
AN	Q4.4	Start-up
NB	Q4.5	Normal mode
AL	Q4.6	Alarm

Proceed as follows:

- ☐ Program the reaction vessel in FBD, LAD and STL
- ☐ Choose a function (we suggest FC 0)
- ☐ Organize OB 1
- ☐ Download the program into the CPU
- ☐ Test the program (**Switch on the CPU to RUN-P**: In this switch position, it is possible to change programs in online mode without needing to put the switch into the STOP position).
- ☐ Change the methods of representation: > FBD > LAD > STL

## 12.6 Container filling system

The control procedures for a container filling system (see Figure 12.13) are intended to be implemented using a programmable logic controller.

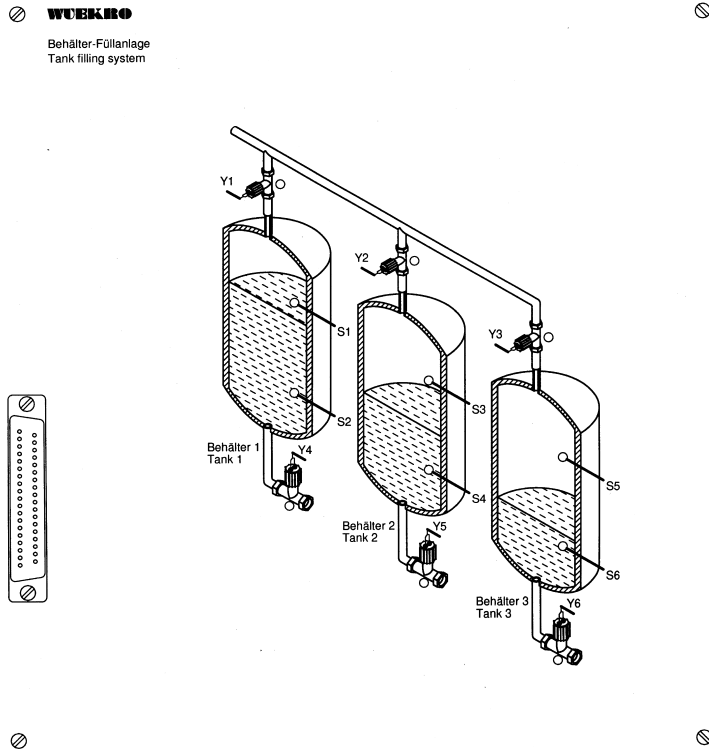


Figure 12.13 Container filling system: Technology scheme

Description of Function

Three supply containers with signal transmitters B1, B3, B5 for the full signal (NO contact) and B2, B4, B6 for the empty signal (NC contact) can be emptied manually in any order you like. Using the controller, it is intended that only one container with an empty signal will ever keep being filled until the full signal is triggered.

The containers are to be filled in the order in which they were emptied. If, for example, the containers were emptied in the order 2-1-3, they must be filled in this order too.

For filling, the PLC controls valves M1, M2 and M3. Using switches S10, S11 and S12 (on the PLC), the containers are emptied via valves M4, M5 and M6.

Figure 12.14  
Connection to the PLC

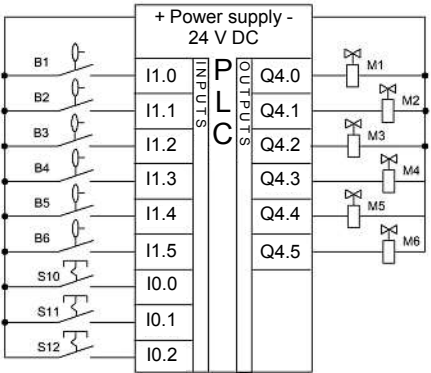


Table 12.6

Assignment list		
Symbol	Operand	Comment
B1	I1.0	Container 1 full
B2	I1.1	Container 1 empty
B3	I1.2	Container 2 full
B4	I1.3	Container 2 empty
B5	I1.4	Container 3 full
B6	I1.5	Container 3 empty
S10	I0.0	Empty container 1
S11	I0.1	Empty container 2
S12	I0.2	Empty container 3
M1	Q4.0	Solenoid valve 1
M2	Q4.1	Solenoid valve 2
M3	Q4.2	Solenoid valve 3
M4	Q4.3	Solenoid valve 4
M5	Q4.4	Solenoid valve 5
M6	Q4.5	Solenoid valve 6
M0.0	M0.0	Container 1 empty
M0.1	M0.1	Container 2 empty
M0.2	M0.2	Container 3 empty

Proceed as follows:

- ☐ Program the container filling system in FBD, LAD and STL
- ☐ Choose a function (we suggest FC 0)
- ☐ Organize OB 1
- ☐ Download the program into the CPU
- ☐ Test the program (**Switch on the CPU to RUN-P**: In this switch position, it is possible to change programs in online mode without needing to put the switch into the STOP position).
- ☐ Change the methods of representation: > FBD > LAD > STL

## 12.7 Automatic tablet filler

The control procedures for an automatic tablet filler (see Figure 12.15) are intended to be implemented using a programmable logic controller.

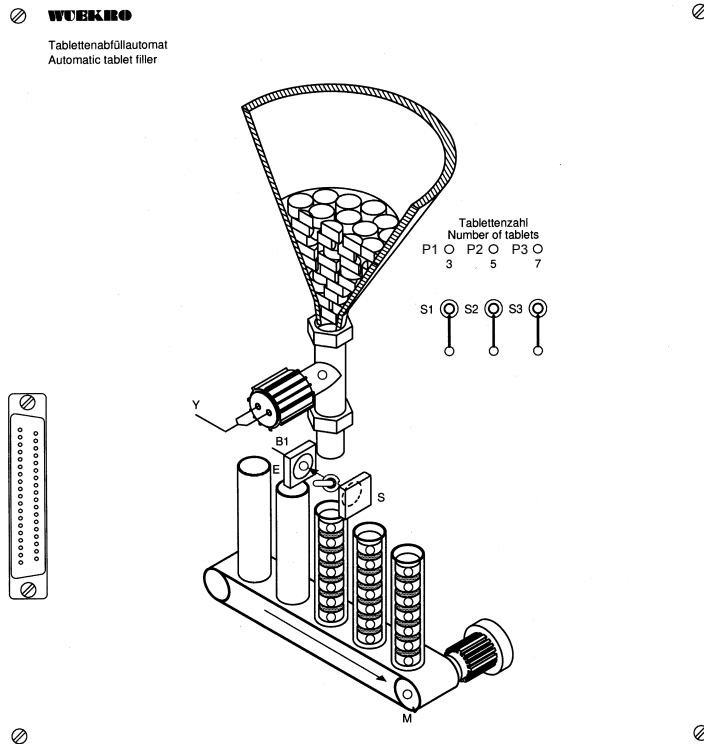


Figure 12.15 Automatic tablet filler: Technology scheme

**Description of Function**

An automatic tablet filler is intended to continuously fill tubes with a selectable number of tablets. After switching on the system using the ON switch, you must choose the desired number of tablets. A belt drive motor M drives the conveyor belt until a tablet tube arrives at the filling location, which encoder GA on the PLC must confirm.

Valve M1 then opens the supply hopper and light barrier B1 counts the tablets. Once the set number of tablets has been reached, the valve closes again and the belt drive motor is set in motion. This process is repeated continuously.

If you want to set a different number of tablets by pressing the corresponding pushbutton, an ongoing filling process must still be completed with the old number. When switching off the system, a filling process that has been started is completely finished before all the actuators are deactivated.

Control requires signal processing:

- ❑ The memory states TSP3, TSP5, TSP7 must result from the briefly pending input signals S1, S2, S3 of the desired number of tablets.
- ❑ Intermediate memories TA3, TA5, TA7 are intended to prevent unclear number selections (pressing S1, S2, S3 several times).
- ❑ It should always be possible to switch directly from one number of tablets to another at any time.
- ❑ If the system is switched-off, storing of the number of tablets must be cleared.

Figure 12.16  
Connection to the PLC

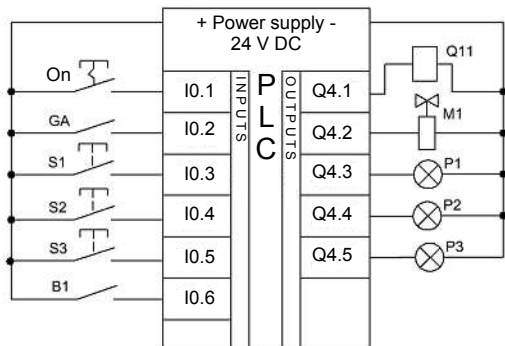


Table 12.7

Assignment list		
Symbol	Operand	Comment
On	I0.1	Switch system On – On = 1 on
GA	I0.2	Encoder of filling location – GA = 1 reached
S1	I0.3	Pushbutton number 3 – S1 = 1 pushed
S2	I0.4	Pushbutton number 5 – S2 = 1 pushed
S3	I0.5	Pushbutton number 7 – S3 = 1 pushed
B1	I0.6	Light barrier – B1 = 0 free
Q11	Q4.1	Belt drive motor – M = 1 running
M1	Q4.2	Valve Y = 1 open
P1	Q4.3	Lamp number of tablets 3 – P1 = 1 lit-up
P2	Q4.4	Lamp number of tablets 5 – P2 = 1 lit-up
P3	Q4.5	Lamp number of tablets 7 – P3 = 1 lit-up
Memory 0	M40.0	Status bit memory
Memory 1	M40.1	Status bit memory
Memory 2	M40.2	Status bit memory
Memory 3	M40.3	Status bit memory
Memory 4	M40.4	Status bit memory
Memory 5	M40.5	Status bit memory
Memory 6	M40.6	Status bit memory
TA3	M50.0	Intermediate memory 3
TSP3	M50.1	Tablet memorizing counter 3
TA5	M50.2	Intermediate memory 5
TSP5	M50.3	Tablet memorizing counter 5
TA7	M50.4	Intermediate memory 7
TSP7	M50.5	Tablet memorizing counter 7
MY	M60.6	Switch-on memory
MX	M60.7	Initializing pulse memory (pulse 1 = cycle time)
N3	C1	Decrementer for 3 tablets
N5	C2	Decrementer for 5 tablets
N7	C3	Decrementer for 7 tablets

Proceed as follows:

- ☐ Program the automatic tablet filler in FBD, LAD and STL
- ☐ Choose a function (we suggest FC 1)
- ☐ Organize OB 1
- ☐ Download the program into the CPU
- ☐ Test the program (**Switch on the CPU to RUN-P**: In this switch position, it is possible to change programs in online mode without needing to put the switch into the STOP position).
- ☐ Change the methods of representation: > FBD > LAD > STL



## 12.8 Door access control system

The control procedures for an access control system (see Figure 12.17) are intended to be implemented using a programmable logic controller.

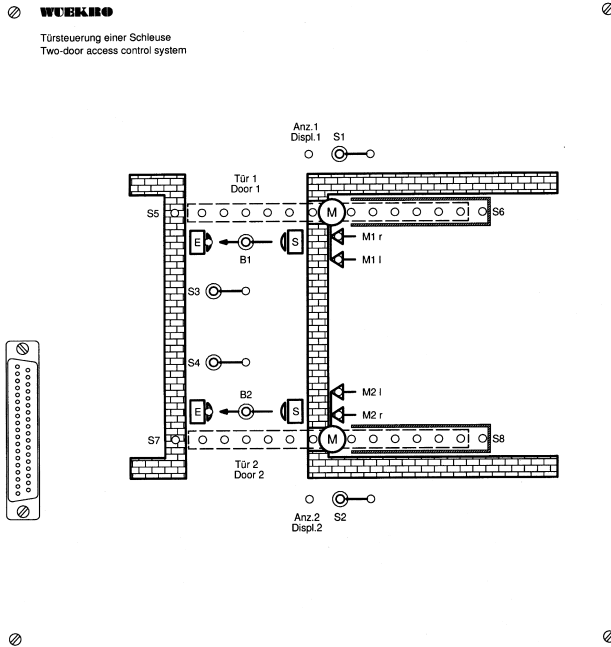


Figure 12.17 Door access control system: Technology scheme

### Description of Function

It is only possible to enter or leave a clean room through an air-lock with two automatically closing doors, A and B. The first door is opened using an external door opener, whereas the second door opens automatically but only after the first door has closed. Both doors close automatically after three seconds have expired.

If during closing of a door, the associated light barrier is interrupted or the associated external or internal door opener is activated, the door opens again immediately. The internal door openers are mounted as a safety measure in case somebody enters the air-lock without first having activated one of the external door openers after they found the door open when somebody else had just left the air-lock.

Two limit switches are provided for each door that signal whether the doors are open or closed. Next to the external door openers, indicator lights are mounted that show whether the controller has detected the pushbutton being pressed.

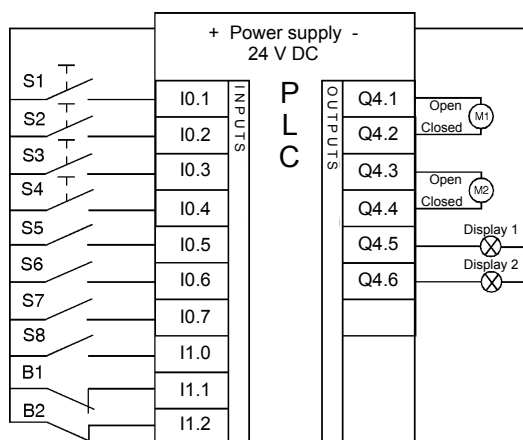


Figure 12.18  
Connection to the PLC

Table 12.8

Assignment list		
Symbol	Operand	Comment
S1	I0.1	Door opener door 1 outside NO contact
S2	I0.2	Door opener door 2 outside NO contact
S3	I0.3	Door opener door 1 inside NO contact
S4	I0.4	Door opener door 2 inside NO contact
S5	I0.5	Limit switch door 1 closed NO contact
S6	I0.6	Limit switch door 1 open NO contact
S7	I0.7	Limit switch door 2 closed NO contact
S8	I1.0	Limit switch door 2 open NO contact
B1	I1.1	Light barrier door 1 NC contact
B2	I1.2	Light barrier door 2 NC contact
M1A	Q4.1	Motor door 1 open
M1Z	Q4.2	Motor door 1 closed
M2A	Q4.3	Motor door 2 open
M2Z	Q4.4	Motor door 2 closed
Display 1	Q4.5	Display pushbutton S1
Display 2	Q4.6	Display pushbutton S2

Control requires signal processing:

- ❑ Since the controller cannot necessarily respond immediately to a button being pressed by opening an air-lock door, display memory states DISPL.1 or DISPL.2 must result from the briefly pending input signals S1 or S2 of the external door openers. The display memories are reset in each case when the corresponding air-lock door opens.
- ❑ For the controller to remember that it has to open the other door in each case despite the display memory being cleared, two key memories TSP1 or TSP2 are

needed in addition. The key memories are reset when the second door in each case is opened.

Proceed as follows:

- ☐ Program the door access control system in FBD, LAD and STL
- ☐ Choose a function (we suggest F 5)
- ☐ Organize OB 1
- ☐ Download the program into the CPU
- ☐ Test the program (**Switch on the CPU to RUN-P**: In this switch position, it is possible to change programs in online mode without needing to put the switch into the STOP position).
- ☐ Change the methods of representation: > FBD > LAD > STL

## 12.9 Pump controller

The control procedures for a pump controller (see Figure 12.19) are intended to be implemented using a programmable logic controller.

### Description of Function

Four pumps pump liquid from a suction container into a network. By switching the pumps in or off on a step-by-step basis, it is intended to maintain the pressure in the network at a constant level within a specific range (see Figure 12.20).

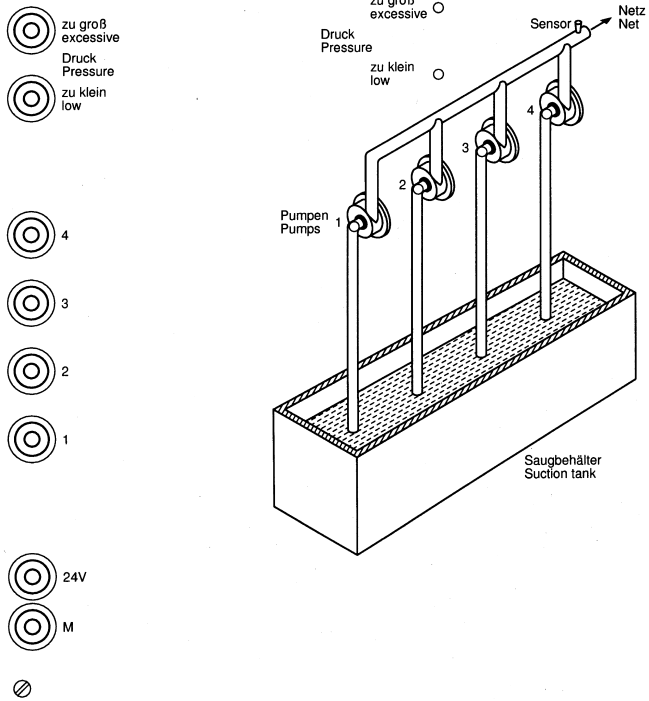
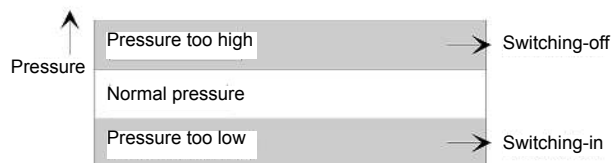


Figure 12.19 Pump controller: Technology scheme

Figure 12.20



- ☐ At switching-in as well as at switching-off, it is intended that the system waits for a specific response time to ensure that no switching pulses are triggered in the case of brief changes in pressure.
- ☐ Whenever the pressure is below the normal value for more than 5 s, it is intended that a switch-in pulse is triggered.
- ☐ Whenever the pressure is above the normal value for more than 5 s, it is intended that a switch-off pulse is triggered.
- ☐ In this connection, it is intended to guarantee that all four pumps have as similar a runtime and switching rate as possible.
- ☐ At switch-off, the pump with the longest runtime must always be deactivated; at switching-in, the pump with the longest idle time must be switched-on.

Figure 12.21  
Connection to the PLC

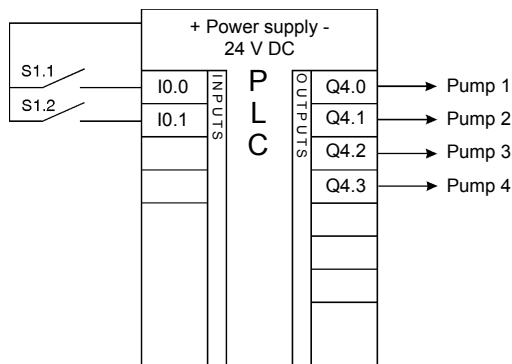


Table 12.9

Assignment list		
Symbol	Operand	Comment
S1.1	I0.0	Pressure too low
S1.2	I0.1	Pressure too high
P1	Q4.0	Pump 1
P2	Q4.1	Pump 2
P3	Q4.2	Pump 3
P4	Q4.3	Pump 4
T1	T1	Switch-on delay SD
T2	T2	Switch-off delay SF
M1	M0.1	Pulse flag for pump On
M2	M1.1	Pulse flag for pump Off
M3	M0.0	Flag for Start T1
M4	M1.0	Flag for Start T2
M5	M2.0	Register pumps On
M6	M2.1	Register pumps On
M7	M2.2	Register pumps On
M33.0	M33.0	Register pumps On
M33.1	M33.1	Register pumps On
M33.2	M33.2	Register pumps On
M8	M2.3	Register pumps Off
M9	M2.4	Register pumps Off
M10	M3.3	Register pumps Off
M11	M3.4	Register pumps Off

Proceed as follows:

- ☐ Program the pump controller in FBD, LAD and STL
- ☐ Choose a function (we suggest FC 3)
- ☐ Organize OB 1
- ☐ Download the program into the CPU
- ☐ Test the program (**Switch on the CPU to RUN-P**: In this switch position, it is possible to change programs in online mode without needing to put the switch into the STOP position).
- ☐ Change the methods of representation: > FBD > LAD > STL

# 13 Sequence Control Systems

## 13.1 Introduction

Processing or process sequences that comply with a specific sequence (see Figure 13.1) are controlled by sequence control systems.

Sequence control systems have the following advantages:

- ❑ Quick and easy configuration and programming,
- ❑ Clear program structure,
- ❑ Easy changing of control procedures,
- ❑ Easy troubleshooting in the case of disturbances,
- ❑ Different operating modes.

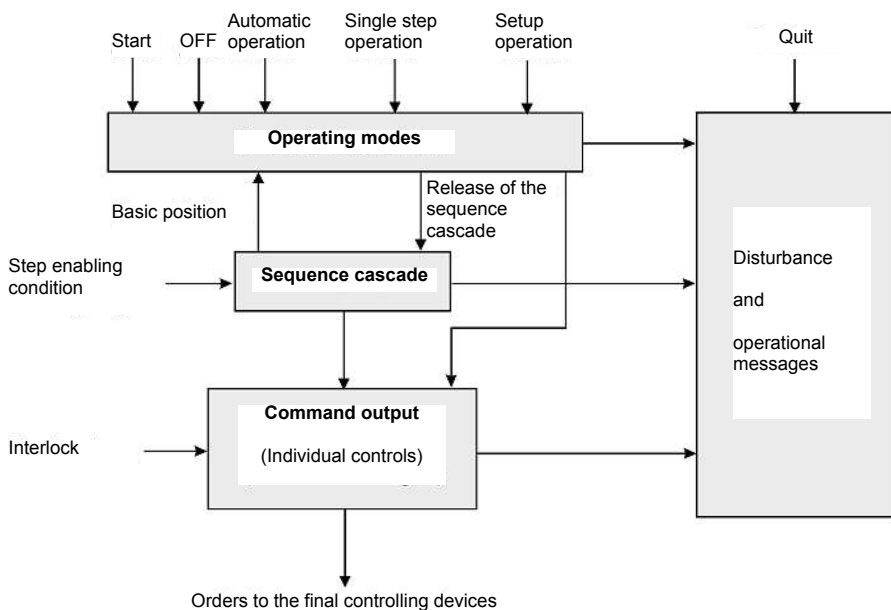


Figure 13.1 Complete control programmed with a sequence control system

## 13.2 Components of a sequence control system

The functions of a sequence control system are divided into four parts (see Figure 13.2).

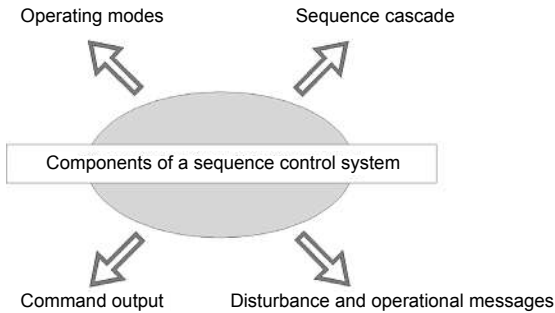


Figure 13.2  
Sequence control system and its components

### Operating modes

In the mode section, you edit the different conditions for the operating modes. You set the operating mode, e.g. manual or automatic mode using selector switches or a corresponding pushbutton on an operator panel. The result is displayed and signalled to the sequence cascade and the command output in the form of release signals.

The following operating modes are customary:

- |                    |  |
|--------------------|--|
| <b>AUTOMATIC</b>   | In this operating mode, the system processes the sequence cascade of the control procedures without the operating staff intervening. |
| <b>SINGLE STEP</b> | In this operating mode, the sequence cascade is processed manually step by step, e.g. it is possible to advance at commissioning.    |
| <b>SET UP</b>      | In this operating mode, it is possible to activate manually the individual actuators independently of the controller's program.      |

### Sequence cascade

In the sequence cascade, the program for the step by step function cycle of the controller is processed. The steps of the sequence are run through one after another with only one step ever being set.



**Command output**

Here, the commands that are activated by the individual steps of the sequence are combined with the release signals of the mode section and the interlock signals from the process. Apart from this, the system takes into account the commands for controlling the individual actuators manually. This command output function corresponds to an individual controller.

**Disturbance and operational messages**

Here, the disturbance and operational messages are generated and the current sequence step is displayed.

**13.3 Type of representation**

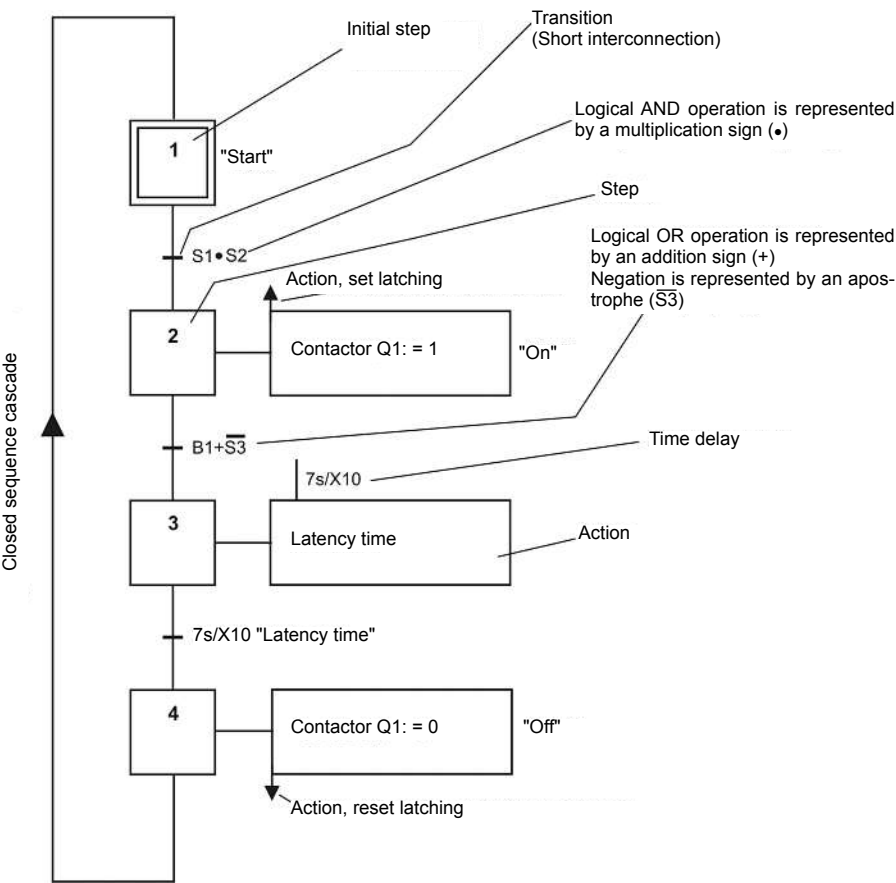


Figure 13.3 Sequence cascade

### 13.4 Linear sequence cascade

With the linear sequence cascade, the steps are run through one after another with only one step ever being set.

A step is set if the one before it is switched-on and the step enabling condition for the step is fulfilled.

Step enabling conditions include, for example, limit switches, timers, sensors, operating modes, etc.

The step enabling condition can be process- or time-dependent. The individual steps can output commands to the actuators.

### 13.5 Sheet metal bending device

The control procedures for a sheet metal bending device are intended to be implemented using a programmable logic controller (see Figure 13.4).

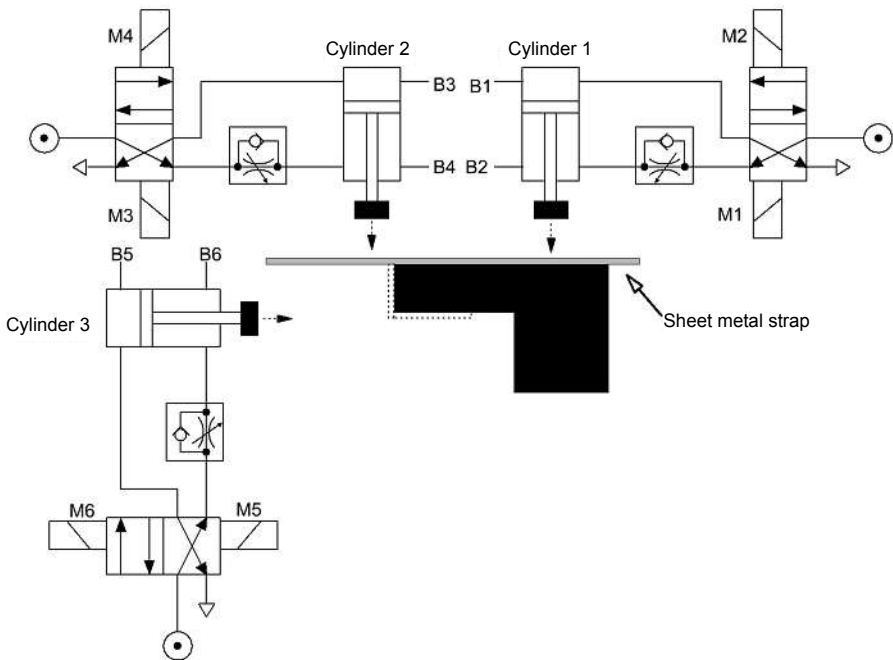


Figure 13.4 Sheet metal bending device: Technology scheme

Description of Function

Metal sheet is bent on a sheet metal bending device. Once the metal sheet has been inserted manually (F5 key) and the start key has been pressed, cylinder 1 extends and secures the metal sheets. Cylinder 2 first bends the metal sheet by 90° before cylinder 3 gives it its final shape. The technology scheme shows which solenoid must be controlled in each case.

Figure 13.5  
Connection to the PLC

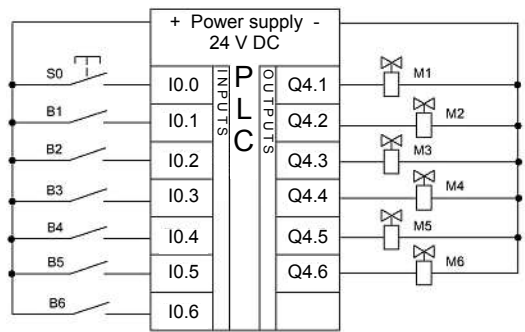


Table 13.1

Assignment list		
Symbol	Operand	Comment
S0	I0.0	Start/single step
B1	I0.1	Back limit position cylinder 1
B2	I0.2	Front limit position cylinder 1
B3	I0.3	Back limit position cylinder 2
B4	I0.4	Front limit position cylinder 2
B5	I0.5	Back limit position cylinder 3
B6	I0.6	Front limit position cylinder 3
M1	Q4.1	Solenoid valve cylinder 1 forwards
M2	Q4.2	Solenoid valve cylinder 1 backwards
M3	Q4.3	Solenoid valve cylinder 2 forwards
M4	Q4.4	Solenoid valve cylinder 2 backwards
M5	Q4.5	Solenoid valve cylinder 3 forwards
M6	Q4.6	Solenoid valve cylinder 3 backwards

Mode section

The mode section makes it possible to operate the sheet metal bending device in automatic single-step mode or, alternatively, in manually controlled single-step mode.

### **Automatic mode**

You preselect automatic mode using switch S10 ( $S10 = 0$ ). In this operating mode, it is possible to trigger the automatic sequence of the bending process using the start key S0 if the plant and the sequence cascade are in the basic position. Pushbutton S12 is used to end automatic mode in step "0".

### **Single-step mode with step enabling conditions**

You set single-step mode using switch S10 ( $S10 = 1$ ). In this operating mode, activating pushbutton S0 switches on the sequence step that follows next in each case, assuming that the step enabling conditions are fulfilled.

The output commands assigned to the sequence step are only executed while pushbutton S9 is switched. If you switch to single step during automatic mode, the controller comes to a halt at the current sequence step. Step enabling is then carried out using pushbutton S0.

Proceed as follows:

- ☐ Program the sheet metal bending device in FBD, LAD and STL
- ☐ Choose a function (we suggest FC 1)
- ☐ Organize OB 1
- ☐ Download the program into the CPU
- ☐ Test the program (**Switch on the CPU to RUN-P:** In this switch position, it is possible to change programs in online mode without needing to put the switch into the STOP position).
- ☐ Change the methods of representation: > FBD > LAD > STL

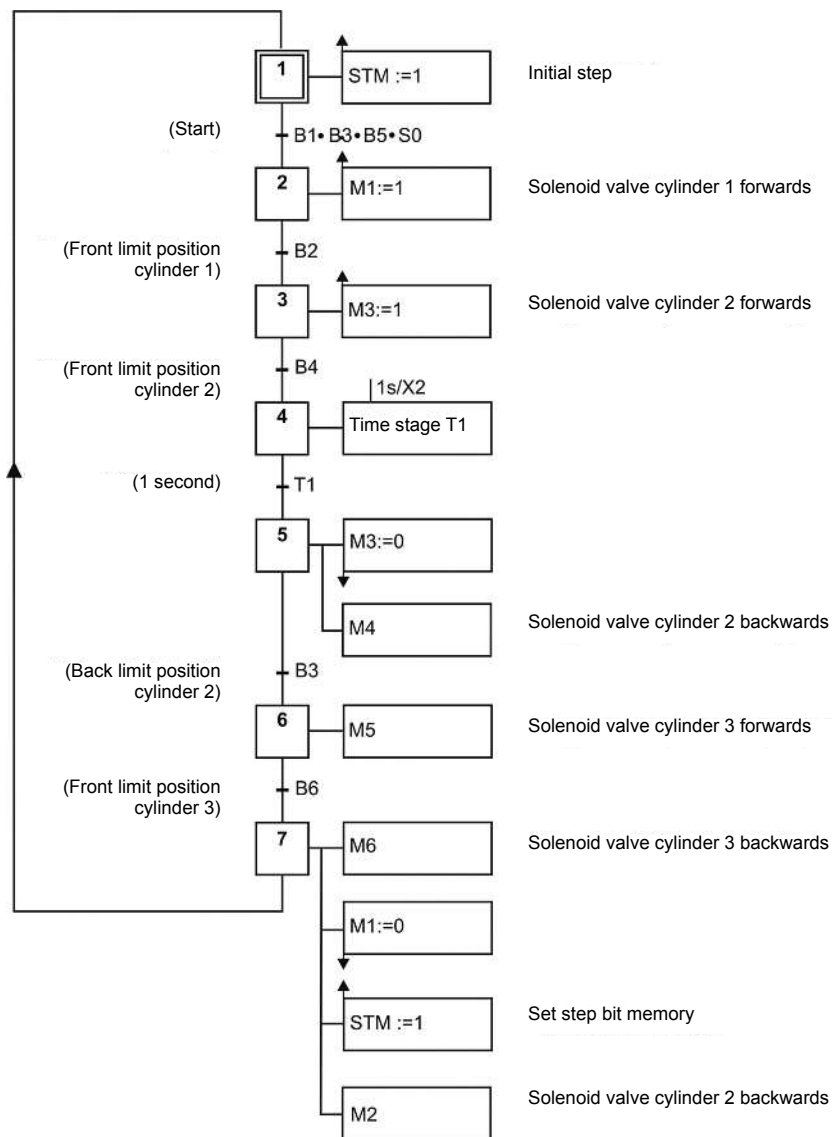


Figure 13.6 Sequence cascade



## 14 Safety Regulations

When programming programmable logic controllers, you must comply with the (DIN VDE 0113) rules in common use in contactor controls.

### 14.1 Rules

You must prevent status conditions that could endanger people or damage machines.

After restoring the mains voltage that has failed, machines must not be able to start up automatically. If there are disturbances in the PLC, the commands from the emergency STOP switch and from the safety limit switches must always be effective. This means that the protective devices must be effective directly on the actuators in the power circuit (manual level).

Deactivation must not be blocked due to faults in the encoder circuits (e.g. wire breaks or earth faults). Switching on is carried out with the working current (NO contact); switching off is carried out with the standby current (NC contact). These rules apply by analogy to sequence control systems too.

## 14.2 Emergency STOP control release

Figure 14.1 shows the principle of the entire power supply of a controller.

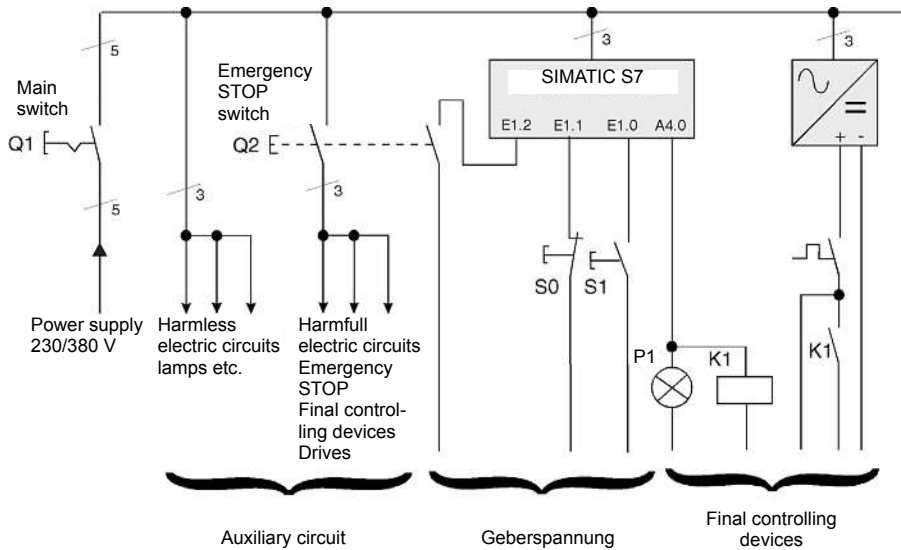


Figure 14.1 Power supply of a controller

The entire plant is activated using main switch **Q1**. Facilities must not be switched off by the emergency STOP switch which may endanger people due to these facilities having been switched off (e.g. presses, clamping devices, auxiliary supplies, etc.).

The emergency STOP switch **Q2** must only switch off drives and actuators that can cause a dangerous situation for people and facilities (VDE 0113). If the emergency STOP switch is activated, this is communicated to the PLC (I1.2) via the auxiliary contactor. When programming, you should note that the input is programmed accordingly as an emergency STOP switch.

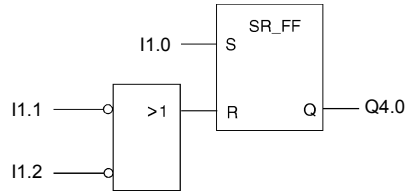
The emergency STOP switch can also be arranged in the supply line instead of the main switch. However, this is only allowed if after an emergency STOP command some of the actuators do not still have to be supplied with energy. The emergency STOP status cannot be saved either.

After switching on the power supply and restoring the voltage after a power failure, no commands must be output to the actuators via the outputs of the programmable controller if this leads to people or machinery being endangered.



### 14.3 Example of a control release

Figure 14.2  
Control release in FBD



Output Q4.0 is switched on in a latching manner using pushbutton **S1** via input I1.0. Output Q4.0 is switched off:

- ☐ Manually using pushbutton **S0** at input I1.1,
- ☐ Via input I1.2 in the case of an emergency **STOP**,
- ☐ In the case of a disturbance of the encoder voltages,
- ☐ In the case of a power failure due to the process image being deleted.

Output Q4.0 can be used in a controller program for switch on or switch off conditions. Using contactor K1 allows you to specify that specific actuators and electric circuits can only be switched on in the case of a control release.



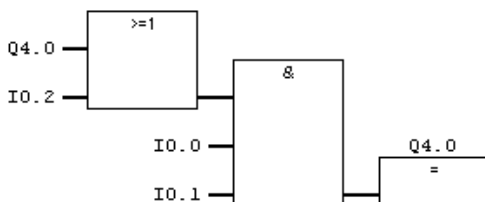
# Appendix

Solutions according to the examples

## Example of a solution according to chapter 6.8.1

FC9 : Solution according to chap. 6.8.1 Temperature difference

Network 1: Control

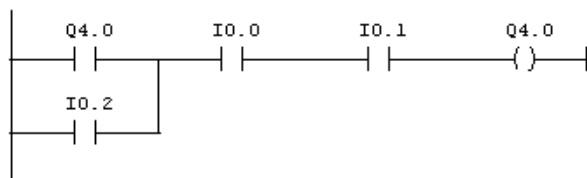


Network 2: Assignment of the pilot lamp

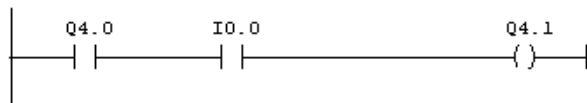


FC9 : Solution according to chap. 6.8.1 Temperature difference

Network 1: Control



Network 2: Assignment of the pilot lamp



FC9 : Solution according to chap. 6.8.1    Temperature difference

**Network 1:** Control

```
A(  
  O      Q      4.0  
  O      I      0.2  
)  
A      I      0.0  
A      I      0.1  
=      Q      4.0
```

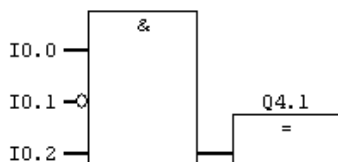
**Network 2:** Assignment of the pilot lamp

```
A      Q      4.0  
A      I      0.0  
=      Q      4.1
```

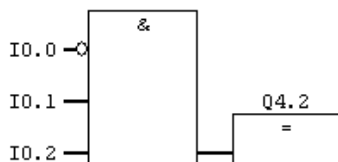
## Example of a solution according to chapter 6.8.2

FC10 : Solution according to chap. 6.8.2 Drinks machine

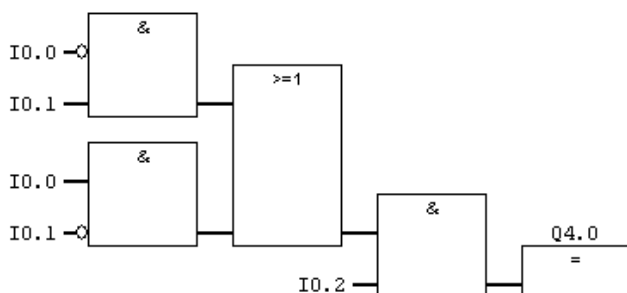
**Network 1:** Tea powder runs out



**Network 2:** Coffee powder runs out

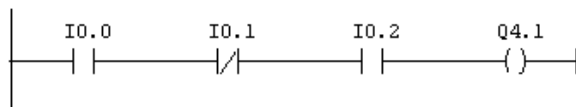


**Network 3:** Hot water runs out

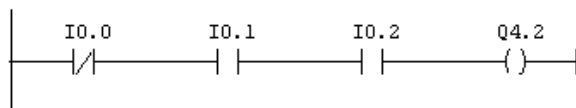


FC10 : Solution according to chap. 6.8.2 Drinks machine

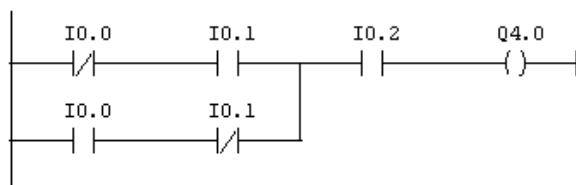
**Network 1:** Tea powder runs out



**Network 2:** Coffee powder runs out



**Network 3:** Hot water runs out



FC10 : Solution according to chap. 6.8.2 Drinks machine

**Network 1:** Tea powder runs out

A	I	0.0
AN	I	0.1
A	I	0.2
=	Q	4.1

**Network 2:** Coffee powder runs out

AN	I	0.0
A	I	0.1
A	I	0.2
=	Q	4.2

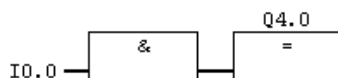
**Network 3:** Hot water runs out

A(		
AN	I	0.0
A	I	0.1
O		
A	I	0.0
AN	I	0.1
)		
A	I	0.2
=	Q	4.0

## Example of a solution according to chapter 6.8.3

FC11 : Solution according to chap. 6.8.3 Intercom

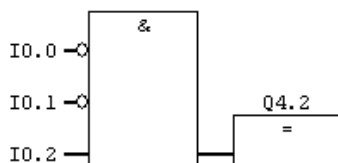
**Network 1:** Managing director is speaking



**Network 2:** Works manager is speaking



**Network 3:** Factory supervisor is speaking



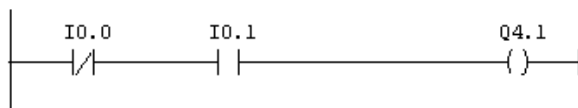


FC11 : Solution according to chap. 6.8.3 Intercom

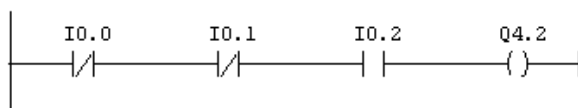
**Network 1:** Managing director is speaking



**Network 2:** Works manager is speaking



**Network 3:** Factory supervisor is speaking



FC11 : Solution according to chap. 6.8.3 Intercom

**Network 1:** Managing director is speaking

A	I	0.0
=	Q	4.0

**Network 2:** Works manager is speaking

AN	I	0.0
A	I	0.1
=	Q	4.1

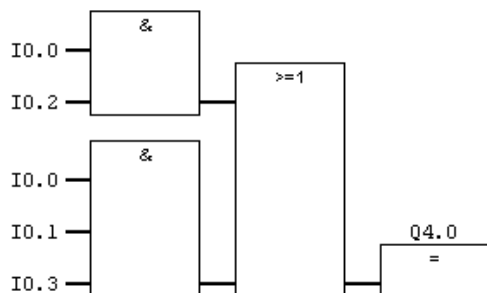
**Network 3:** Factory supervisor is speaking

AN	I	0.0
AN	I	0.1
A	I	0.2
=	Q	4.2

## Example of a solution according to chapter 6.8.4

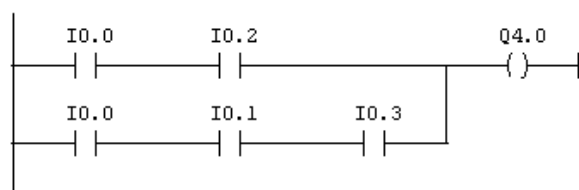
FC12 : Solution according to chap. 6.8.4 Generator

Network 1: Generator monitoring



FC12 : Solution according to chap. 6.8.4 Generator

Network 1: Generator monitoring



FC12 : Solution according to chap. 6.8.4 Generator

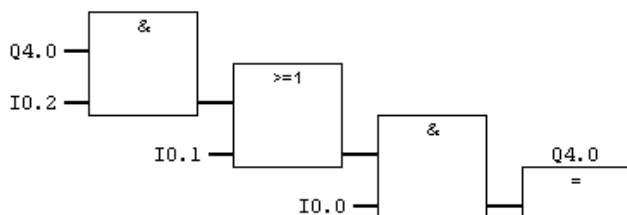
**Network 1:** Generator monitoring

A	I	0.0
A	I	0.2
O		
A	I	0.0
A	I	0.1
A	I	0.3
=	Q	4.0

## Example of a solution according to chapter 6.8.5

FC13 : Solution according to chap. 6.8.5 Boiler control

**Network 1:** Auxiliary contact



**Network 2:** Water inflow valve

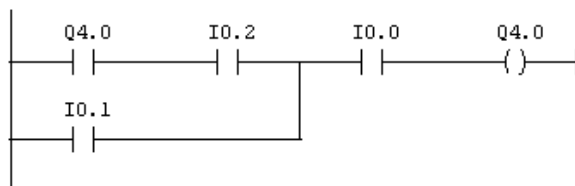


**Network 3:** Indicator light for water inflow

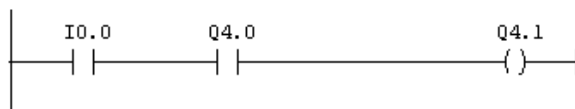


FC13 : Solution according to chap. 6.8.5 Boiler control

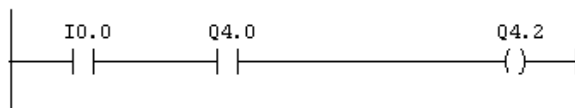
**Network 1:** Auxiliary contact



**Network 2:** Water inflow valve



**Network 3:** Indicator light for water inflow



FC13 : Solution according to chap. 6.8.5 Boiler control

**Network 1:** Auxiliary contact

```

A(
  A    Q    4.0
  A    I    0.2
  O    I    0.1
)
A    I    0.0
=    Q    4.0
  
```

**Network 2:** Water inflow valve

```

A    I    0.0
A    Q    4.0
=    Q    4.1
  
```

**Network 3:** Indicator light for water inflow

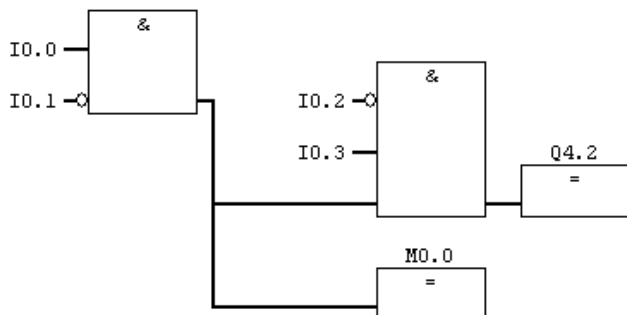
```

A    I    0.0
A    Q    4.0
=    Q    4.2
  
```

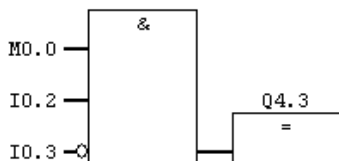
## Example of a solution according to chapter 6.8.6

FC14 : Solution according to chap. 6.8.6 Smelting furnaces

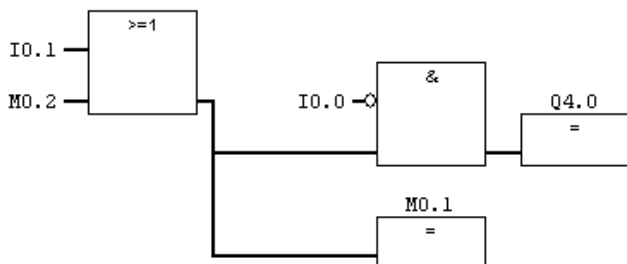
**Network 1:** Indicator light P3 Furnace "C" Vc = 25 %



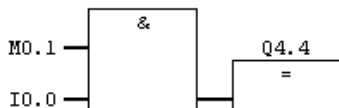
**Network 2:** Indicator light P4 Furnace "D" Vd = 25 %



**Network 3:** Indicator light P1 Furnace "A" Va = 65 %



**Network 4:** Indicator light P5 Alarm



**Network 5:** Indicator light P2    Furnace "E"    Vb = 45 %



**Network 6 :** Auxiliary bit memory

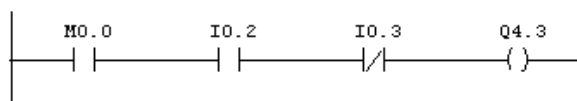


FC14 : Solution according to chap. 6.8.6 Smelting furnaces

**Network 1:** Indicator light P3 Furnace "C" Vc = 25 %

A	I	0.0
AN	I	0.1
=	L	0.0
AN	I	0.2
A	I	0.3
A	L	0.0
=	Q	4.2
A	L	0.0
BLD	102	
=	M	0.0

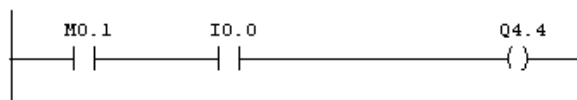
**Network 2:** Indicator light P4 Furnace "D" Vd = 25 %



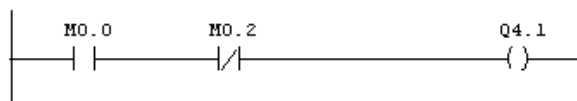
**Network 3:** Indicator light P1 Furnace "A" Va = 65 %

O	I	0.1
O	M	0.2
=	L	0.0
AN	I	0.0
A	L	0.0
=	Q	4.0
A	L	0.0
BLD	102	
=	M	0.1

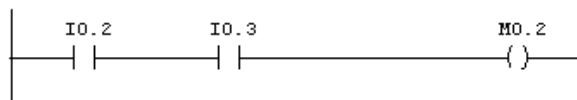
**Network 4:** Indicator light P5 Alarm



**Network 5:** Indicator light P2 Furnace "B" Vb = 45 %



**Network 6:** Auxiliary bit memory





FC14 : Solution according to chap. 6.8.6 Smelting furnaces

**Network 1:** Indicator light P3    Furnace "C"    Vc = 25 %

A	I	0.0
AN	I	0.1
=	L	0.0
AN	I	0.2
A	I	0.3
A	L	0.0
=	Q	4.2
A	L	0.0
BLD	102	
=	M	0.0

**Network 2:** Indicator light P4    Furnace "D"    Vd = 25 %

A	M	0.0
A	I	0.2
AN	I	0.3
=	Q	4.3

**Network 3:** Indicator light P1    Furnace "A"    Va = 65 %

O	I	0.1
O	M	0.2
=	L	0.0
AN	I	0.0
A	L	0.0
=	Q	4.0
A	L	0.0
BLD	102	
=	M	0.1

**Network 4:** Indicator light P5    Alarm

A	M	0.1
A	I	0.0
=	Q	4.4

**Network 5:** Indicator light P2    Furnace "B"    Vb = 45 %

A	M	0.0
AN	M	0.2
=	Q	4.1

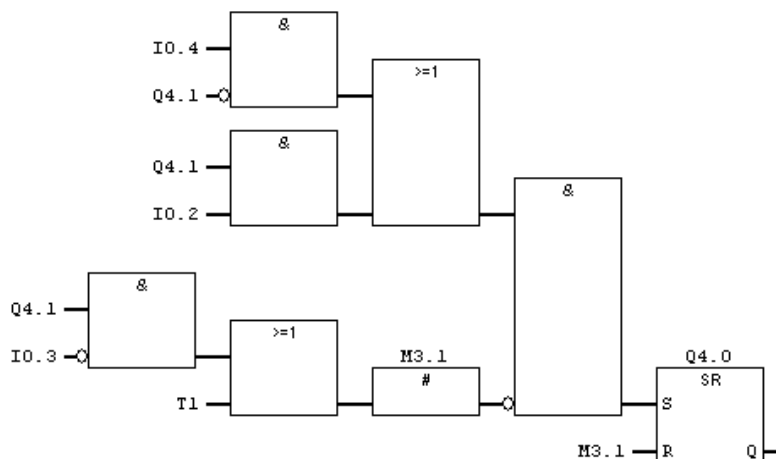
**Network 6:** Auxiliary bit memory

A	I	0.2
A	I	0.3
=	M	0.2

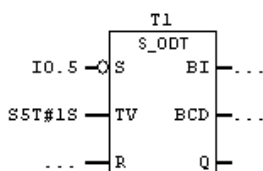
## Example of a solution according to chapter 6.9.3

FC29 : Solution according to chap. 6.9.3 Pump controller

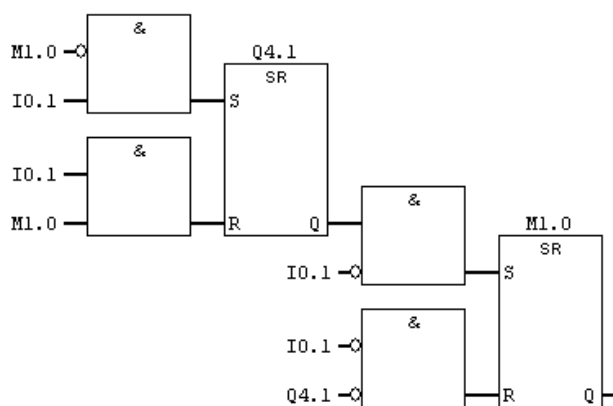
Network 1: Logical operation for the pump



Network 2: Time stage SD "water sloshing"



**Network 3 : T-Flip-Flop**



FC29 : Solution according to chap. 6.9.3 Pump controller

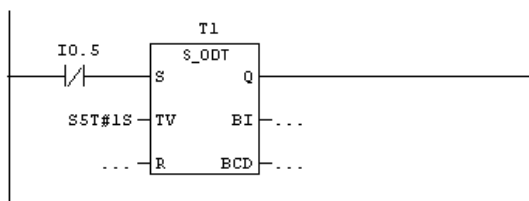
**Network 1:** Logical operation for the pump

```

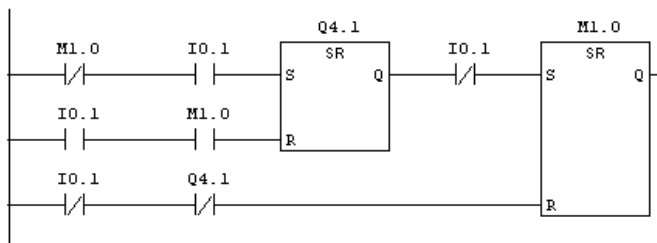
A(
  A    I    0.4           //Fill level min.
  AN   Q    4.1           //Pump
  O
  A    Q    4.1           //Indicator light manual/automatic
  A    I    0.2           //Button "NO contact" Pump manual On
)
A(
  A(
  A(
  A    Q    4.1           //Indicator light manual/automatic
  AN   I    0.3           //Button "NO contact" Pump manual Off
  O    T    1
  )
  =    M    3.1           //Auxiliary bit memory
  A    M    3.1
  NOT
  )
  )
  S    Q    4.0           //Pump
  A    M    3.1           //Auxiliary bit memory
  R    Q    4.0           //Pump
  NOP  O

```

**Network 2:** Time stage SD "water sloshing"



**Network 3:** T-Flip-Flop



FC29 : Solution according to chap. 6.9.3 Pump controller

Network 1: Logical operation for the pump

```

A(
A  I    0.4          //Fill level min.
AN Q    4.1          //Pump
O
A  Q    4.1          //Indicator light manual/automatic
A  I    0.2          //Button "NO contact" Pump manual On
)
A(
A(
A(
A  Q    4.1          //Indicator light manual/automatic
AN I    0.3          //Button "NO contact" Pump manual Off
O  T    1
)
)
=  M    3.1          //Auxiliary bit memory
A  M    3.1
NOT
)
)
S  Q    4.0          //Pump
A  M    3.1          //Auxiliary bit memory
R  Q    4.0          //Pump
NOP O

```

Network 2: Time stage SD "water sloshing"

```

AN  I    0.5
L   SST#1S
SD  T    1
NOP O
NOP O
NOP O
NOP O

```

Network 3: T-Flip-Flop

```

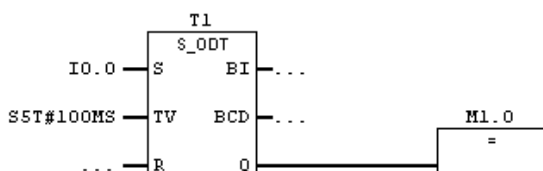
A(
AN  M    1.0          //T-Flip-Flop memory
A  I    0.1          //Button "NO contact" manual/automatic switchover
S  Q    4.1          //Indicator light manual/automatic
A  I    0.1
A  M    1.0
R  Q    4.1
A  Q    4.1
)
AN  I    0.1
S  M    1.0
AN  I    0.1
AN  Q    4.1
R  M    1.0
NOP O

```

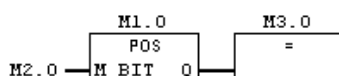
## Example of a solution according to chapter 7.4

FC28 : Solution according to chap. 7.4 Acknowledgement circuit

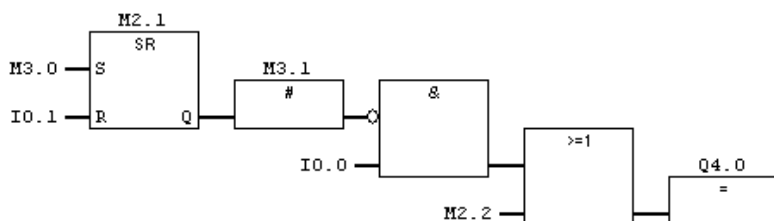
Network 1: Switch-on delay 100 ms



Network 2: Momentary impulse with a rising edge



Network 3: Logical connection

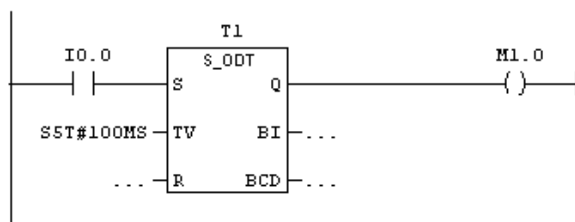


Network 4: Clock memory

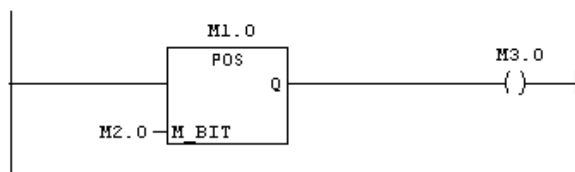


FC28 : Solution according to chap. 7.4 Acknowledgement circuit

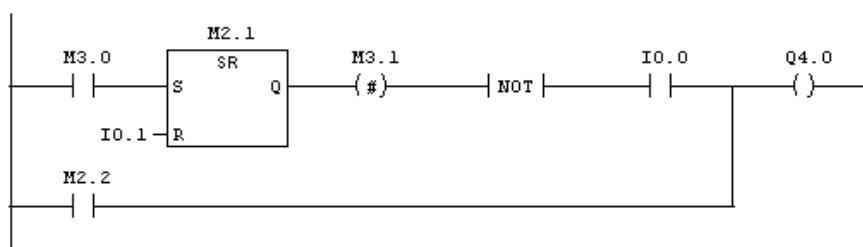
**Network 1:** Switch-on delay 100 ms



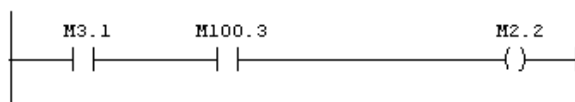
**Network 2:** Momentary impulse with a rising edge



**Network 3:** Logical connection



**Network 4:** Clock memory



FC28 : Solution according to chap. 7.4 Acknowledgement circuit

**Network 1:** Switch-on delay 100 ms

```

A      I      0.0
L      S5T#100MS           //Time 100 ms
SD     T      1
NOP    O
NOP    O
NOP    O
A      T      1
=      M      1.0

```

**Network 2:** Momentary impulse with a rising edge

```

A      M      1.0           //Auxiliary bit memory
BLD    100
FP     M      2.0           //Edge bit memory
=      M      3.0           //Auxiliary bit memory

```

**Network 3:** Logical connection

```

A(
A      M      3.0
S      M      2.1
A      I      0.1
R      M      2.1
A      M      2.1
)
=      M      3.1
A      M      3.1
NOT
A      I      0.0
O      M      2.2
=      Q      4.0           //Indicator light

```

**Network 4:** Clock memory

```

A      M      3.1
A      M      100.3         //Clock memory 2 Hz
=      M      2.2

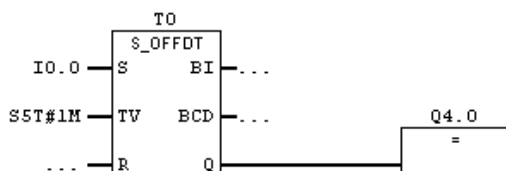
```



## Example of a solution according to chapter 8.6.1

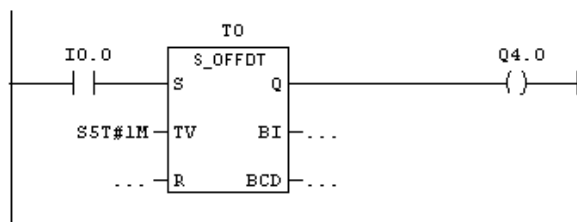
FC19 : Solution according to chap. 8.6.1 Garage lighting

Network 1: Time stage "SF"



FC19 : Solution according to chap. 8.6.1 Garage lighting

Network 1: Time stage "SF"



FC19 : Solution according to chap. 8.6.1 Garage lighting

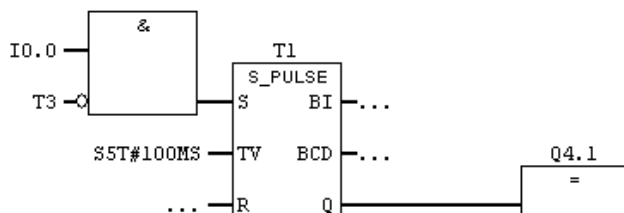
Network 1: Time stage "SF"

A	I	0.0
L	S5T#1M	
SF	T	0
NOP	0	
NOP	0	
NOP	0	
A	T	0
=	Q	4.0

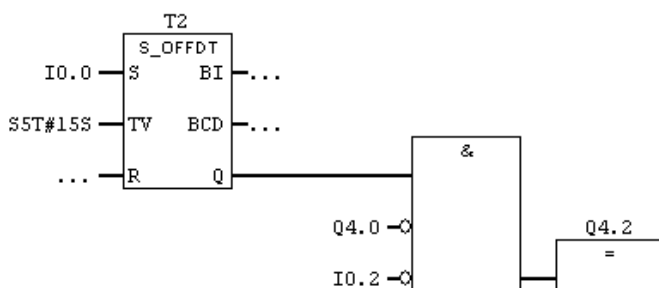
## Example of a solution according to chapter 8.6.2

FC20 : Solution according to chap. 8.6.2 Filling system

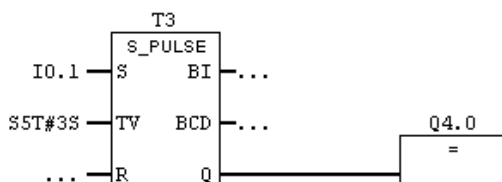
**Network 1:** Release for crates



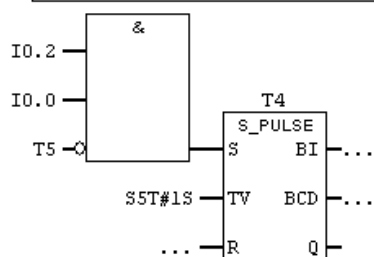
**Network 2:** Conveyor belt motor



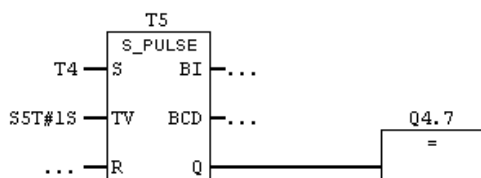
**Network 3:** Valve on silo



**Network 4:** Warning pause

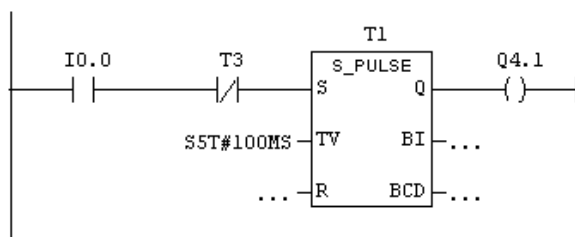


**Network 5:** Warning light "Silo empty"

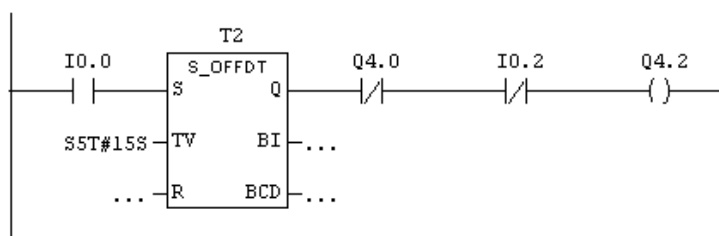


FC20 : Solution according to chap. 8.6.2 Filling system

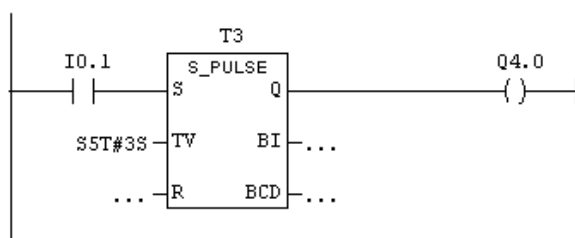
**Network 1:** Release for crates



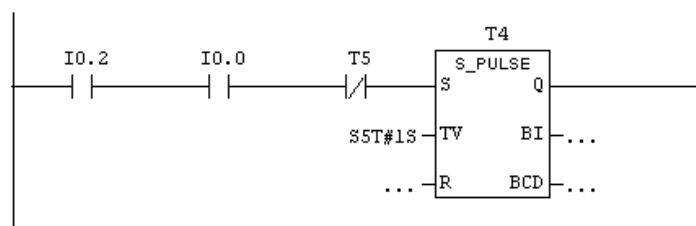
**Network 2:** Conveyor belt motor



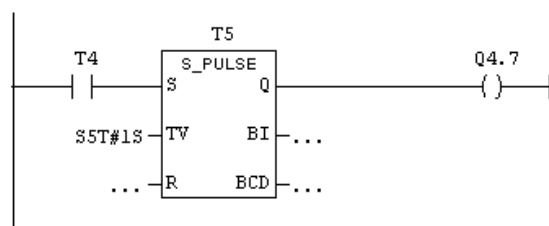
**Network 3:** Valve on silo



**Network 4:** Warning pause



**Network 5:** Warning light "Silo empty"



FC20 : Solution according to chap. 8.6.2 Filling system

**Network 1:** Release for crates

```
A      I      0.0
AN     T      3
L      SST#100MS
SP     T      1
NOP    0
NOP    0
NOP    0
A      T      1
=      Q      4.1
```

**Network 2:** Conveyor belt motor

```
A(
A      I      0.0
L      SST#15S
SF     T      2
NOP    0
NOP    0
NOP    0
A      T      2
)
AN     Q      4.0
AN     I      0.2
=      Q      4.2
```

**Network 3:** Valve on silo

```
A      I      0.1
L      SST#3S
SP     T      3
NOP    0
NOP    0
NOP    0
A      T      3
=      Q      4.0
```

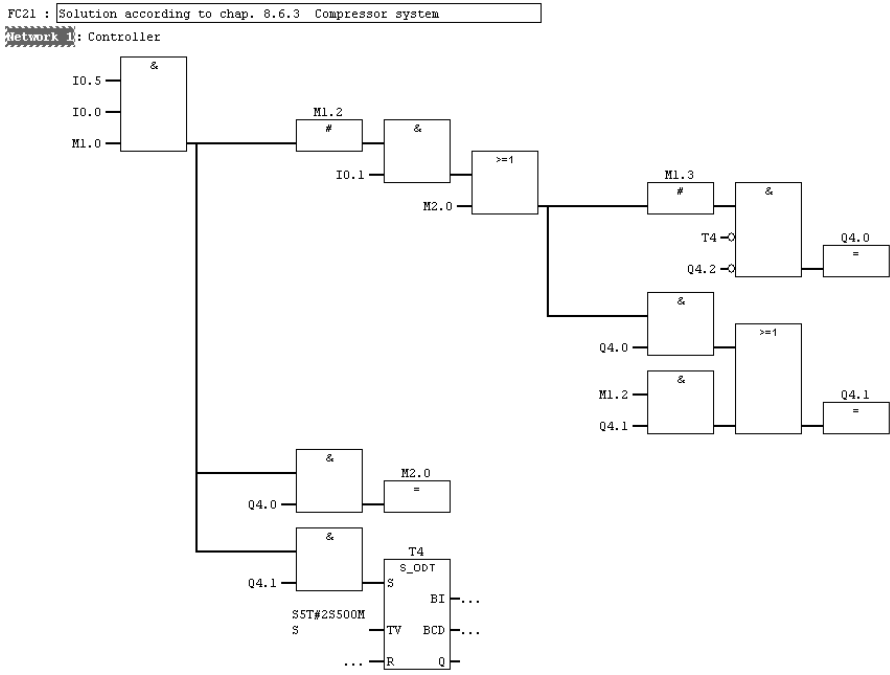
**Network 4:** Warning pause

A	I	0.2
A	I	0.0
AN	T	5
L	S5T#1S	
SP	T	4
NOP	0	
NOP	0	
NOP	0	
NOP	0	

**Network 5:** Warning light "Silo empty"

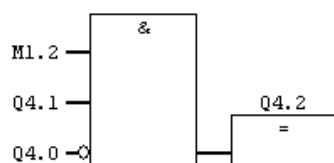
A	T	4
L	S5T#1S	
SP	T	5
NOP	0	
NOP	0	
NOP	0	
A	T	5
=	Q	4.7

# Example of a solution according to chapter 8.6.3





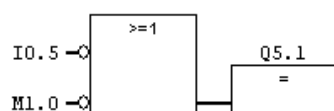
**Network 2 :** Delta contactor



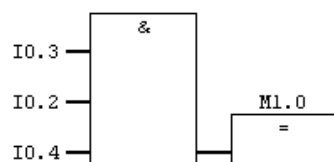
**Network 3 :** Indicator light for operation (delta contactor)



**Network 4 :** Indicator light for fault



**Network 5 :**



FC21 : Solution according to chap. 8.6.3 Compressor system

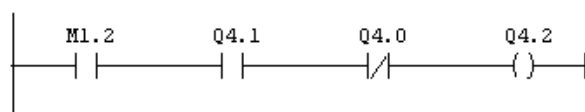
**Network 1:** Controller

```

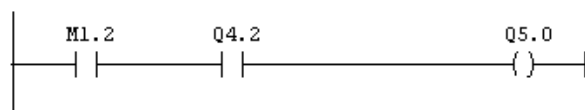
A      I      0.5
A      I      0.0
A      M      1.0
=      L      0.0
A(
A      L      0.0
=      M      1.2
A      M      1.2          //Auxiliary bit memory
)
A      I      0.1
O      M      2.0
=      L      0.1
A(
A      L      0.1
=      M      1.3          //Auxiliary bit memory
A      M      1.3
)
AN      T      4
AN      Q      4.2          //Star contactor
=      Q      4.0
A      L      0.1
A      Q      4.0
O
A      M      1.2
A      Q      4.1          //Mains contactor
=      Q      4.1
A      L      0.0
A      Q      4.0
=      M      2.0
A      L      0.0
A      Q      4.1
L      SST#2S500MS
SD      T      4
NOP      O
NOP      O
NOP      O
NOP      O

```

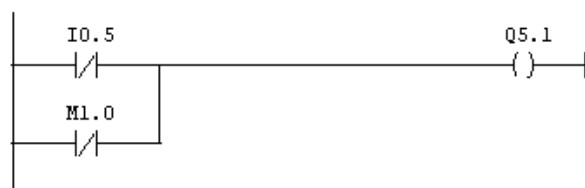
**Network 2 : Delta contactor**



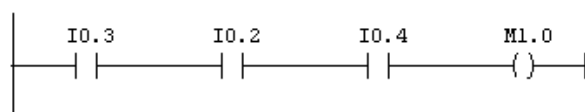
**Network 3 : Indicator light for operation (delta contactor)**



**Network 4 : Indivator light for fault**



**Network 5 :**



FC21 : Solution according to chap. 8.6.3 Compressor system

Network 1: Controller

```

A      I      0.5
A      I      0.0
A      M      1.0
=      L      0.0
A(
A      L      0.0
=      M      1.2
A      M      1.2          //Auxiliary bit memory
)
A      I      0.1
O      M      2.0
=      L      0.1
A(
A      L      0.1
=      M      1.3          //Auxiliary bit memory
A      M      1.3
)
AN      T      4
AN      Q      4.2          //Star contactor
=      Q      4.0
A      L      0.1
A      Q      4.0
O
A      M      1.2
A      Q      4.1          //Mains contactor
=      Q      4.1
A      L      0.0
A      Q      4.0
=      M      2.0
A      L      0.0
A      Q      4.1
L      SST#2S500MS
SD      T      4
NOP      O
NOP      O
NOP      O
NOP      O

```

**Network 2 : Delta contactor**

A	M	1.2	
A	Q	4.1	
AN	Q	4.0	
=	Q	4.2	//Delta contactor

**Network 3 : Indicator light for operation (delta contactor)**

A	M	1.2	
A	Q	4.2	
=	Q	5.0	//Indicator light delta contactor

**Network 4 : Indivator light for fault**

ON	I	0.5	
ON	M	1.0	
=	Q	5.1	//Indicator light fault

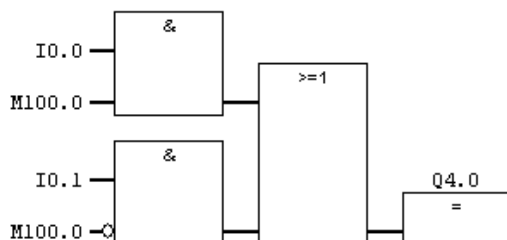
**Network 5 :**

A	I	0.3	
A	I	0.2	
A	I	0.4	
=	M	1.0	//Auxiliary bit memory

## Example of a solution according to chapter 9.1.1

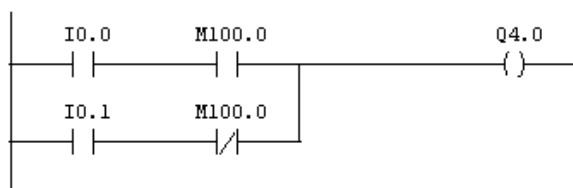
FC22 : Solution according to chap. 9.1.1 Channel switch

Network 1: Network



FC22 : Solution according to chap. 9.1.1 Channel switch

Network 1: Network



FC22 : Solution according to chap. 9.1.1 Channel switch

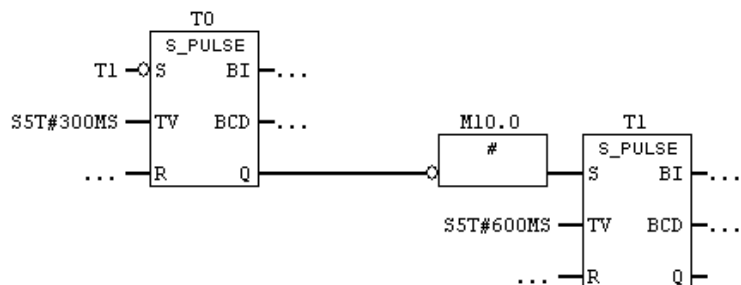
Network 1: Network

A	I	0.0
A	M	100.0
O		
A	I	0.1
AN	M	100.0
=	Q	4.0

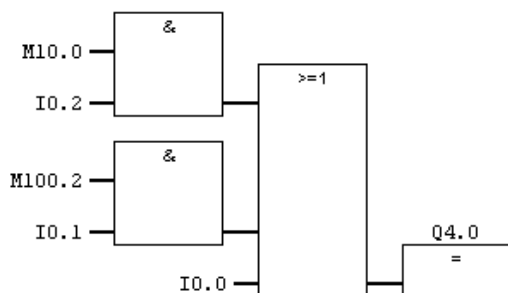
## Example of a solution according to chapter 9.1.2

FC23 : Solution according to chap. 9.1.2 Paging system

**Network 1:** Clock generator 1:2

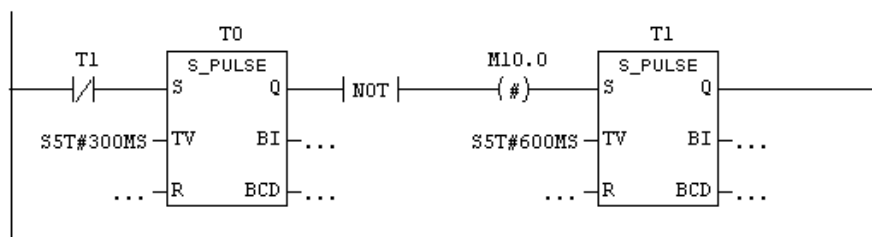


**Network 2:** Logical connection

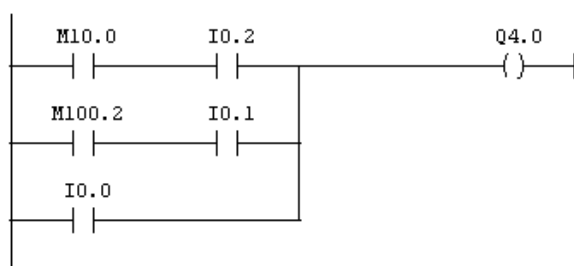


FC23 : Solution according to chap. 9.1.2 Paging system

**Network 1:** Clock generator 1:2



**Network 2:** Logical connection





FC23 : Solution according to chap. 9.1.2 Paging system

Network 1: Clock generator 1:2

```
A(
  AN    T      1
  L      SST#300MS
  SP    T      0
  NOP   0
  NOP   0
  NOP   0
  A      T      0
)
NOT
=      M      10.0           //Clock from clock generator with two times
A      M      10.0
L      SST#600MS
SP     T      1
NOP    0
NOP    0
NOP    0
NOP    0
```

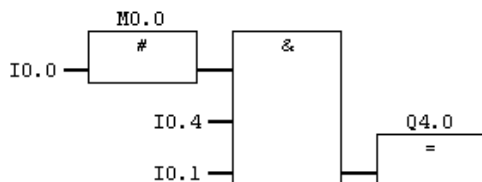
Network 2 : Logical connection

```
A      M      10.0
A      I      0.2
O
A      M      100.2         //Clock memory from CPU
A      I      0.1
O      I      0.0
=      Q      4.0
```

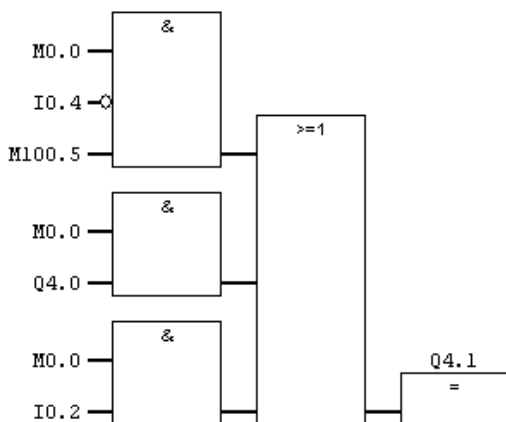
## Example of a solution according to chapter 9.1.3

FC24 : Solution according to chap. 9.1.3 Air supply

**Network 1:** Power contactor for compressor

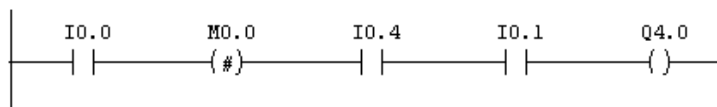


**Network 2:** Indicator light operation / fault

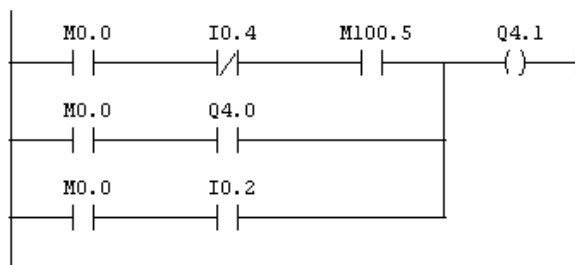


FC24 : Solution according to chap. 9.1.3 Air supply

**Network 1:** Power contactor for compressor



**Network 2:** Indicator light operation / fault



FC24 : Solution according to chap. 9.1.3 Air supply

**Network 1:** Power contactor for compressor

A	I	0.0	//Plant ON
=	M	0.0	//Auxiliary bit memory
A	M	0.0	
A	I	0.4	//Motor relay switch "NC contact"
A	I	0.1	
=	Q	4.0	//Power contactor compressor

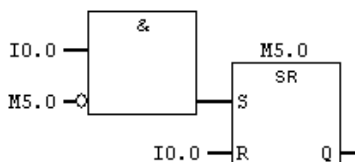
**Network 2:** Indicator light operation / fault

A	M	0.0	//Auxiliary bit memory
AN	I	0.4	//Motor relay switch "NO contact"
A	M	100.5	//Clock memory from CPU 1 Hz
O			
A	M	0.0	
A	Q	4.0	
O			
A	M	0.0	
A	I	0.2	
=	Q	4.1	//Indicator light operation /fault

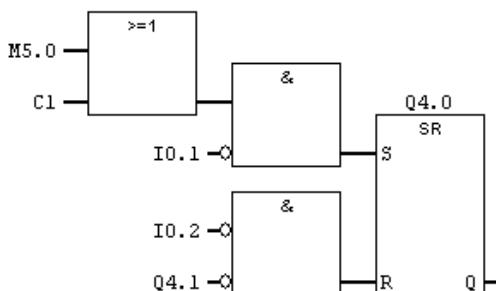
## Example of a solution according to chapter 10.3

FC25 : Solution according to chap. 10.3 Cleaning bath

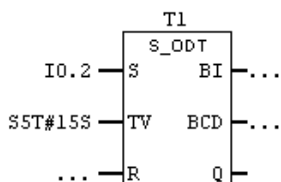
**Network 1:** Release for start



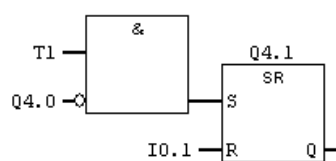
**Network 2:** Motor downwards



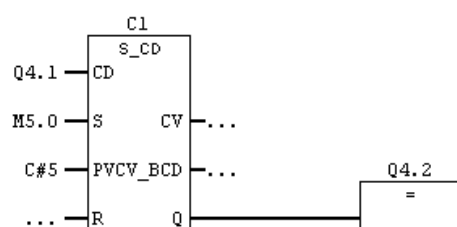
**Network 3:** Dwell time 15 seconds



**Network 4 : Motor upwards**

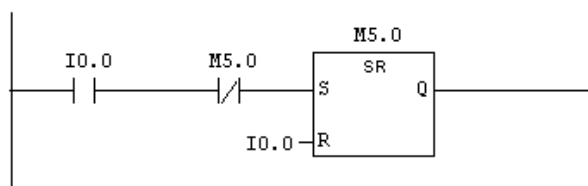


**Network 5 : Counter**

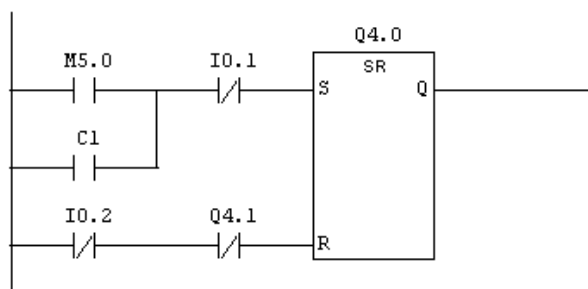


FC25 : Solution according to chap. 10.3 Cleaning bath

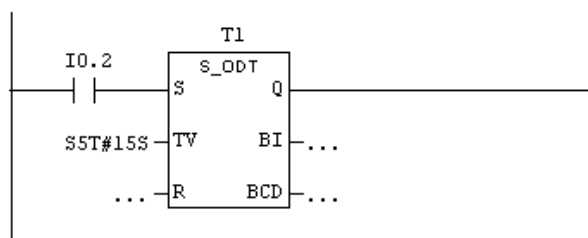
**Network 1:** Release for start



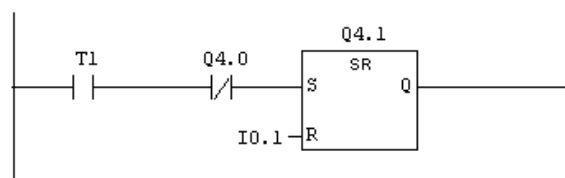
**Network 2:** Motor downwards



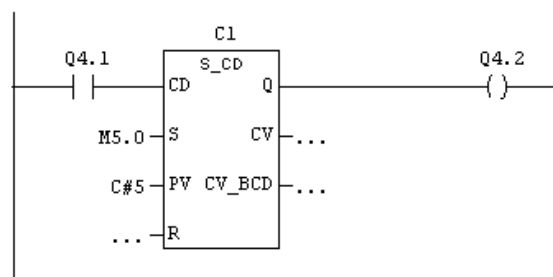
**Network 3:** Dwell time 15 seconds



**Network 4 : Motor upwards**



**Network 5 : Counter**



FC25 : Solution according to chap. 10.3 Cleaning bath

**Network 1:** Release for start

A	I	0.0
AN	M	5.0
S	M	5.0
A	I	0.0
R	M	5.0
NOP	O	

**Network 2:** Motor downwards

A(		
O	M	5.0
O	C	1
)		
AN	I	0.1
S	Q	4.0
AN	I	0.2
AN	Q	4.1
R	Q	4.0
NOP	O	

**Network 3:** Dwell time 15 seconds

A	I	0.2
L	\$ST#15S	
SD	T	1
NOP	O	
NOP	O	
NOP	O	
NOP	O	



**Network 4 : Motor upwards**

A	T	1
AN	Q	4.0
S	Q	4.1
A	I	0.1
R	Q	4.1
NOP	0	

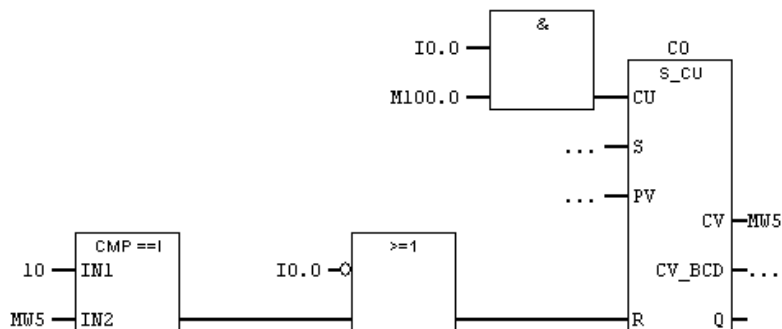
**Network 5 : Counter**

A	Q	4.1
CD	C	1
BLD	101	
A	M	5.0
L	C#5	
S	C	1
NOP	0	
NOP	0	
NOP	0	
A	C	1
=	Q	4.2

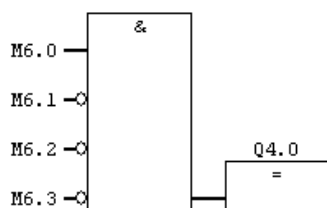
## Example of a solution according to chapter 11.2

FC26 : Solution according to chap. 11.2 Runway light

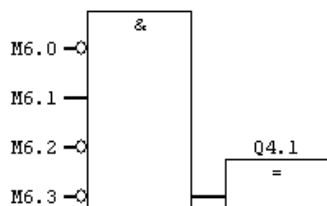
**Network 1:** Comparator with counter and clock pulse 10 Hz



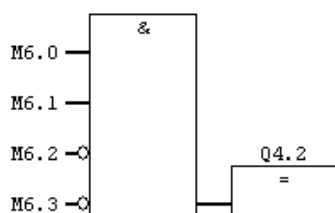
**Network 2:** Number 1



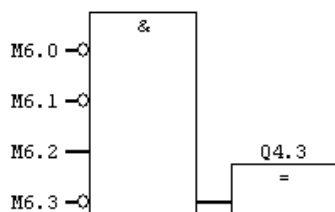
**Network 3:** Number 2



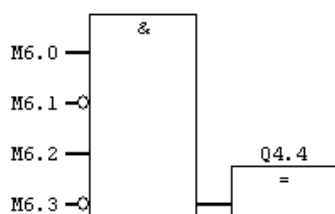
**Network 4 : Number 3**



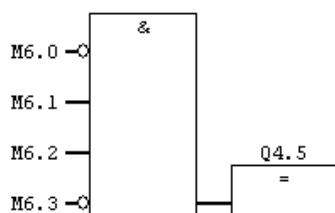
**Network 5 : Number 4**



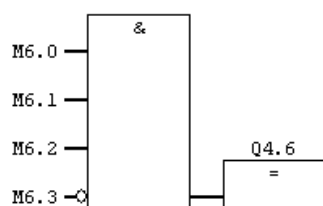
**Network 6 : Number 5**



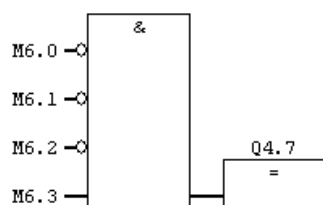
**Network 7 : Number 6**



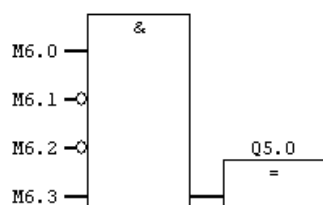
**Network 8 : Number 7**



**Network 9 : Number 8**

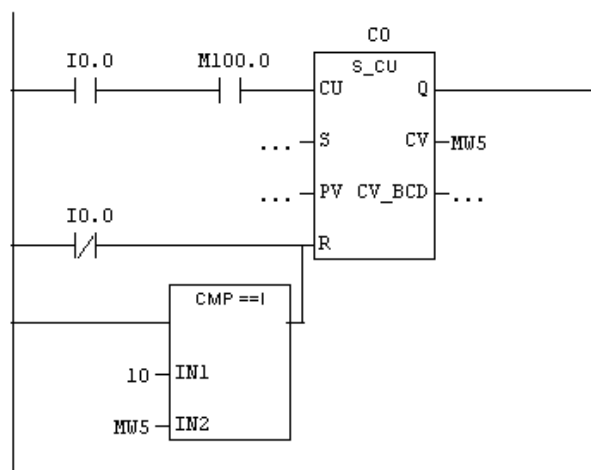


**Network 10 : Number 9**

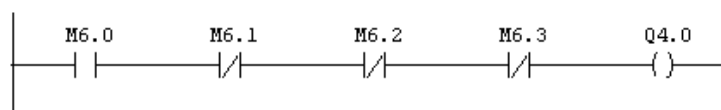


FC26 : Solution according to chap. 11.2 Runway light

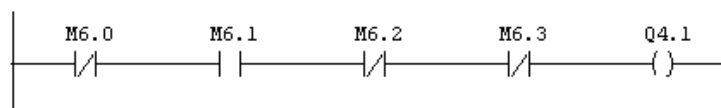
**Network 1:** Comparator with counter and clock pulse 10 Hz



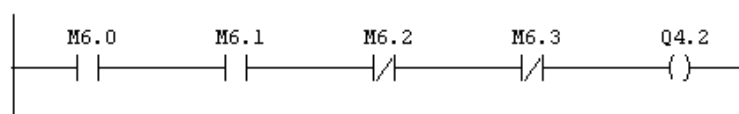
**Network 2:** Number 1



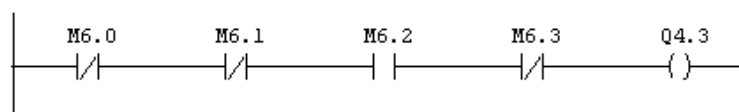
**Network 3:** Number 2



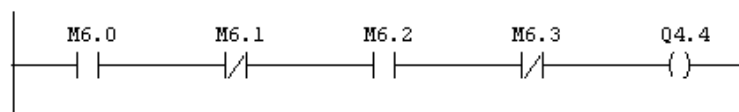
**Network 4 : Number 3**



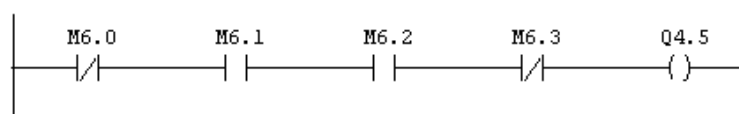
**Network 5 : Number 4**



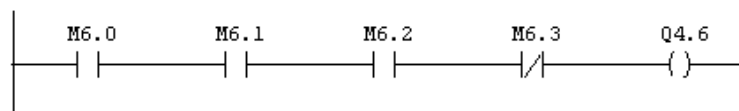
**Network 6 : Number 5**



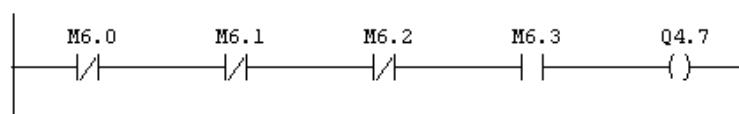
**Network 7 : Number 6**



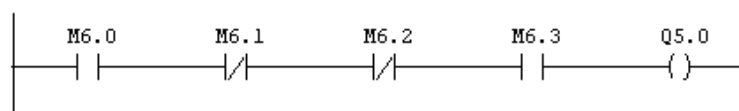
**Network 8 : Number 7**



**Network 9 : Number 8**



**Network 10 : Number 9**



FC26 : Solution according to chap. 11.2 Runway light

**Network 1:** Comparator with counter and clock pulse 10 Hz

```
A      I      0.0
A      M      100.0
CU      C      0
BLD     101
NOP     0
NOP     0
A(
  ON     I      0.0
  O(
    L     10
    L     MW     5
    ==I
  )
)
R      C      0
L      C      0
T      MW     5
NOP     0
NOP     0
```

**Network 2:** Number 1

```
A      M      6.0
AN     M      6.1
AN     M      6.2
AN     M      6.3
=      Q      4.0
```

**Network 3:** Number 2

```
AN     M      6.0
A      M      6.1
AN     M      6.2
AN     M      6.3
=      Q      4.1
```

**Network 4:** Number 3

```
A      M      6.0
A      M      6.1
AN     M      6.2
AN     M      6.3
=      Q      4.2
```

**Network 5 : Number 4**

AN	M	6.0
AN	M	6.1
A	M	6.2
AN	M	6.3
=	Q	4.3

**Network 6 : Number 5**

A	M	6.0
AN	M	6.1
A	M	6.2
AN	M	6.3
=	Q	4.4

**Network 7 : Number 6**

AN	M	6.0
A	M	6.1
A	M	6.2
AN	M	6.3
=	Q	4.5

**Network 8 : Number 7**

A	M	6.0
A	M	6.1
A	M	6.2
AN	M	6.3
=	Q	4.6

**Network 9 : Number 8**

AN	M	6.0
AN	M	6.1
AN	M	6.2
A	M	6.3
=	Q	4.7

**Network 10 : Number 9**

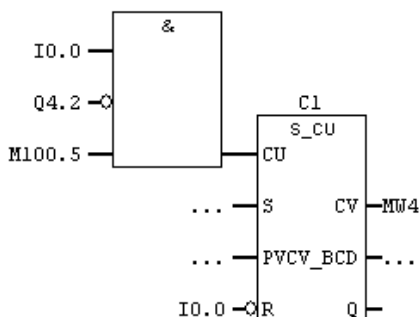
A	M	6.0
AN	M	6.1
AN	M	6.2
A	M	6.3
=	Q	5.0



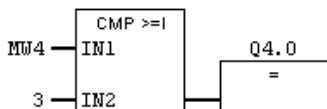
## Example of a solution according to chapter 11.3

FC27 : Solution according to chap. 11.3 Sequence function

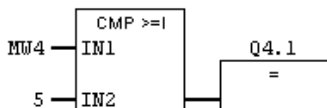
**Network 1:** Counter with stop function



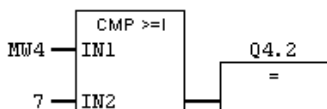
**Network 2:** Indicator light P1



**Network 3:** Indicator light P2

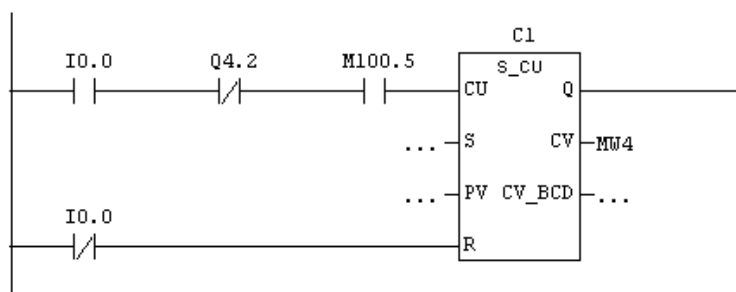


**Network 4:** Indicator light P3

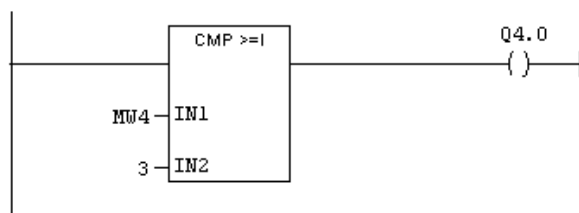


FC27 : Solution according to chap. 11.3 Sequence function

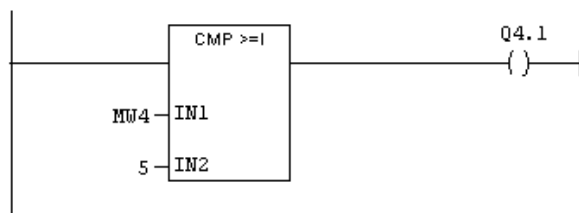
**Network 1:** Counter with stop function



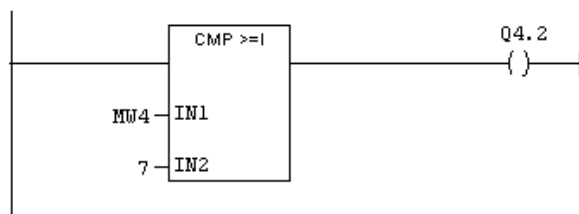
**Network 2:** Indicator light P1



**Network 3:** Indicator light P2



**Network 4:** Indicator light P3



FC27 : Solution according to chap. 11.3 Sequence function

**Network 1:** Counter with stop function

```
A      I      0.0
AN     Q      4.2
A      M      100.5           //Clock memory 1 Hz
CU     C      1
BLD    101
NOP    0
NOP    0
AN     I      0.0
R      C      1
L      C      1
T      MW     4
NOP    0
NOP    0
```

**Network 2:** Indicator light P1

```
L      MW     4
L      3
>=I
=      Q      4.0
```

**Network 3:** Indicator light P2

```
L      MW     4
L      5
>=I
=      Q      4.1
```

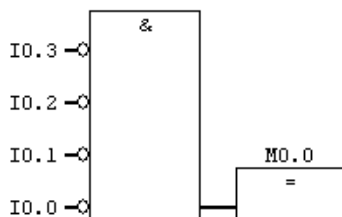
**Network 4:** Indicator light P3

```
L      MW     4
L      7
>=I
=      Q      4.2
```

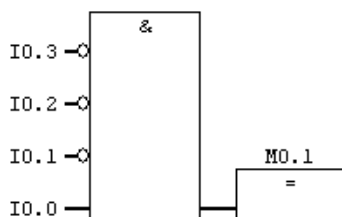
## Example of a solution according to chapter 12.1

FC0 : Solution according to chap. 12.1 Seven-segment display

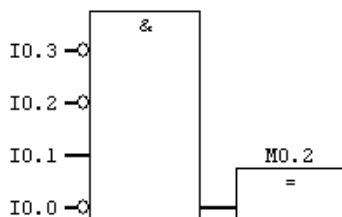
**Network 1:** Evaluation of number 0



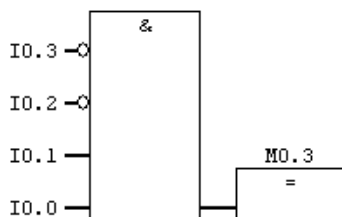
**Network 2:** Evaluation of number 1



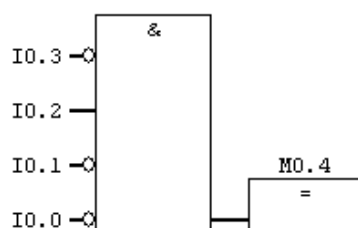
**Network 3:** Evaluation of number 2



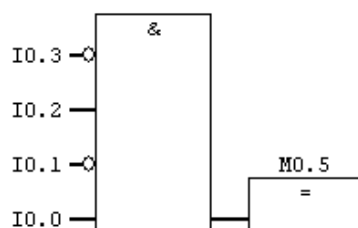
**Network 4:** Evaluation of number 3



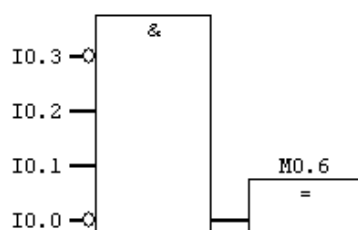
**Network 5 :** Evaluation of number 4



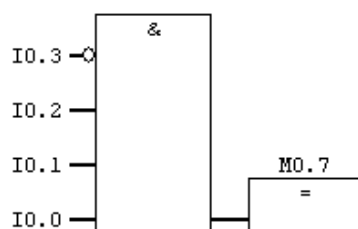
**Network 6 :** Evaluation of number 5



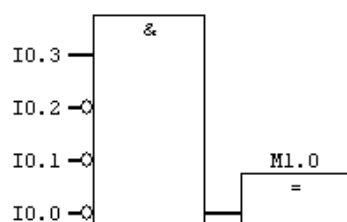
**Network 7 :** Evaluation of number 6



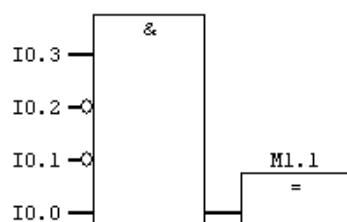
**Network 8 :** Evaluation of number 7



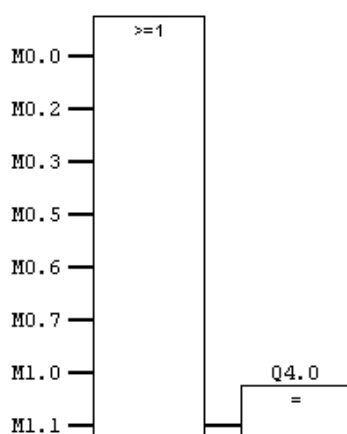
**Network 9:** Evaluation of number 8



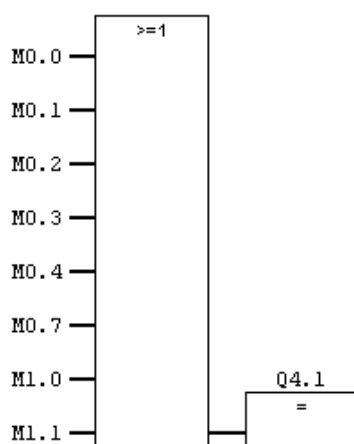
**Network 10:** Evaluation of number 9



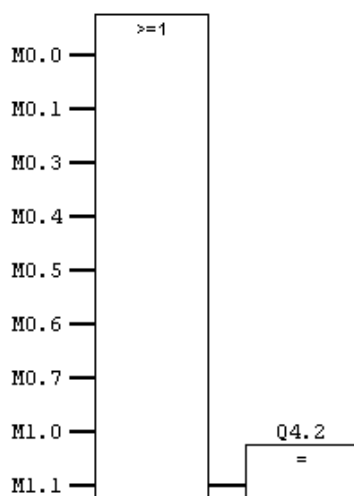
**Network 11:** Control of a segment



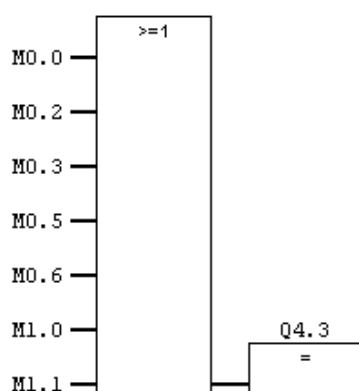
**Network 12 : Control of a segment**



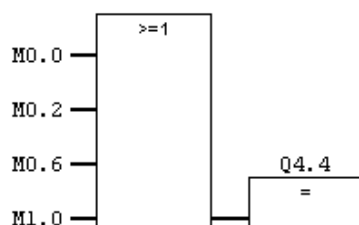
**Network 13 : Control of a segment**



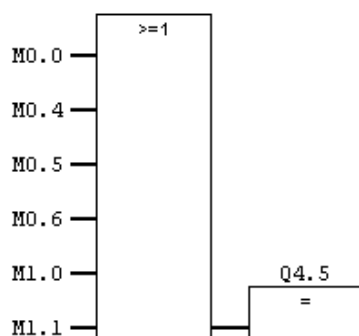
**Network 14 : Control of a segment**



**Network 15 : Control of a segment**

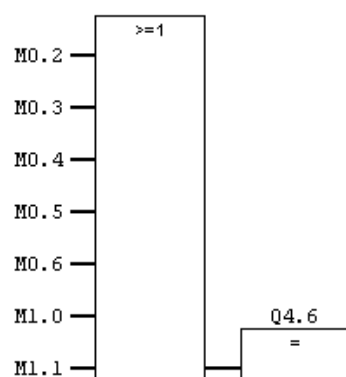


**Network 16 : Control of a segment**



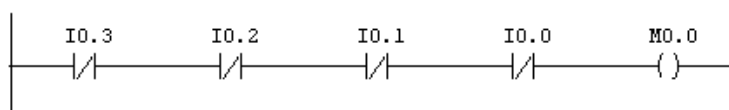


**Network 17:** Control of a segment

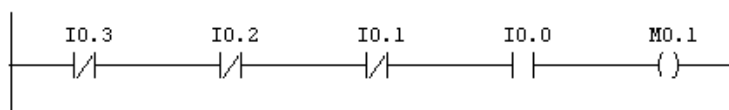


FC0 : Solution according to chap. 12.1 Seven-segment display

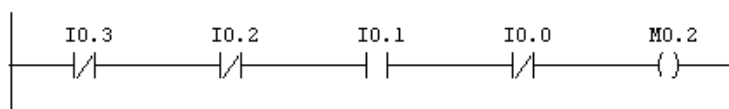
**Network 1:** Evaluation of number 0



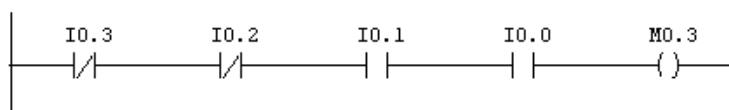
**Network 2:** Evaluation of number 1



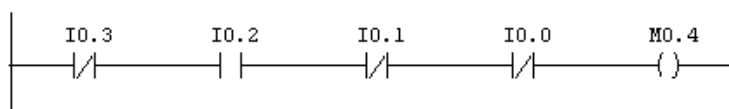
**Network 3:** Evaluation of number 2



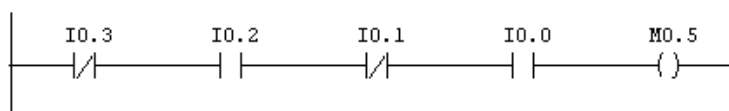
**Network 4:** Evaluation of number 3



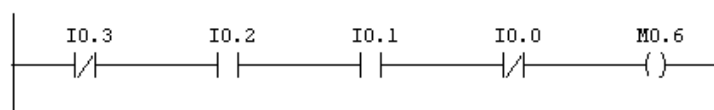
**Network 5:** Evaluation of number 4



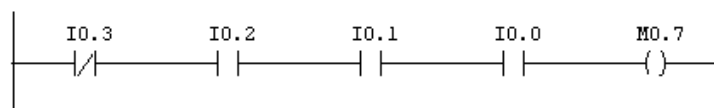
**Network 6:** Evaluation of number 5



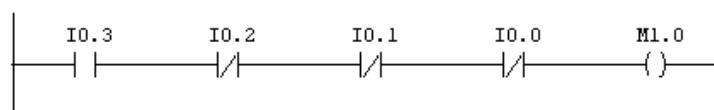
**Network 7 :** Evaluation of number 6



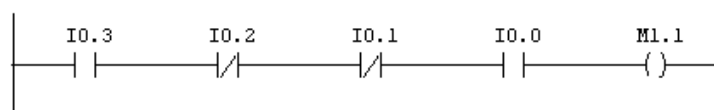
**Network 8 :** Evaluation of number 7



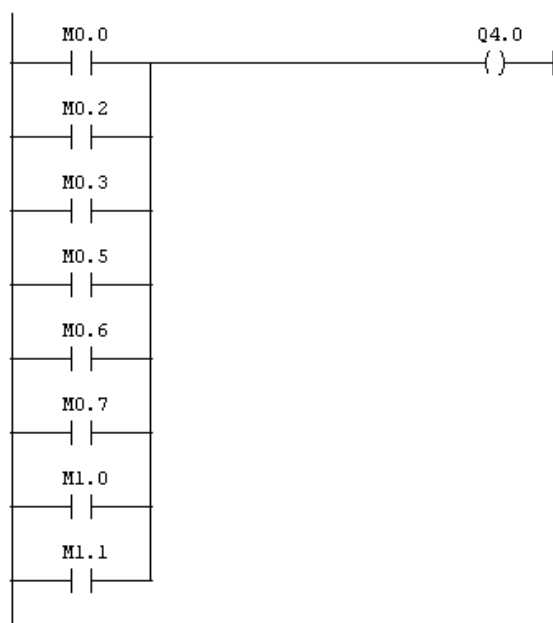
**Network 9 :** Evaluation of number 8



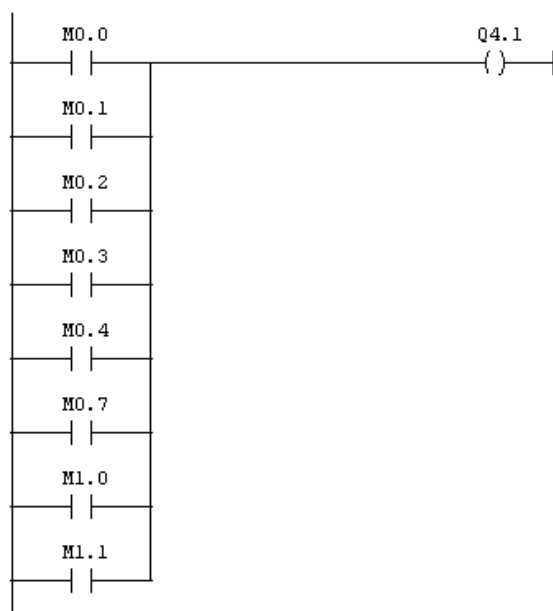
**Network 10 :** Evaluation of number 9



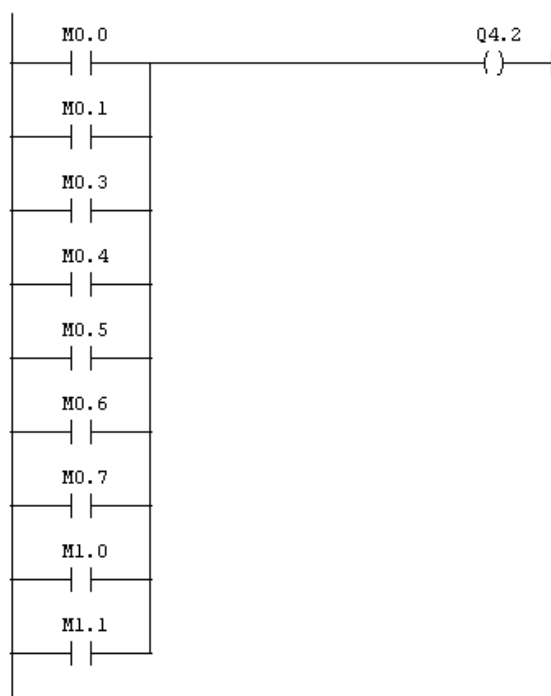
**Network 11:** Control of a segment



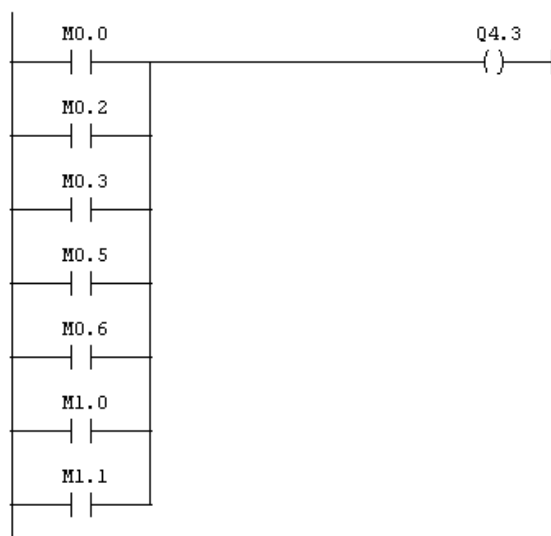
**Network 12:** Control of a segment



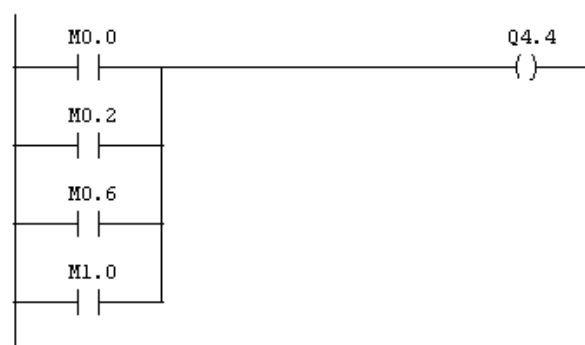
**Network 13:** Control of a segment



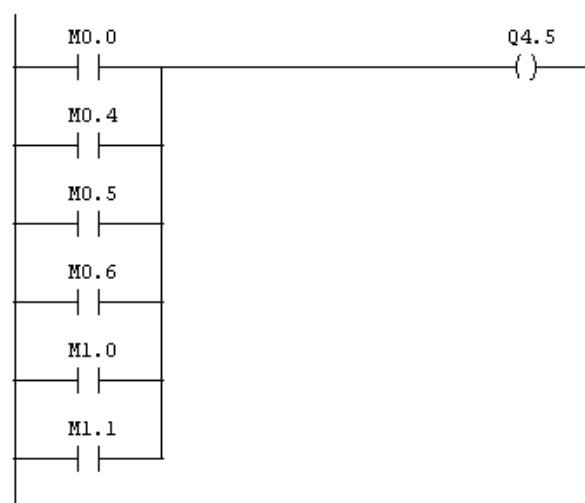
**Network 14:** Control of a segment



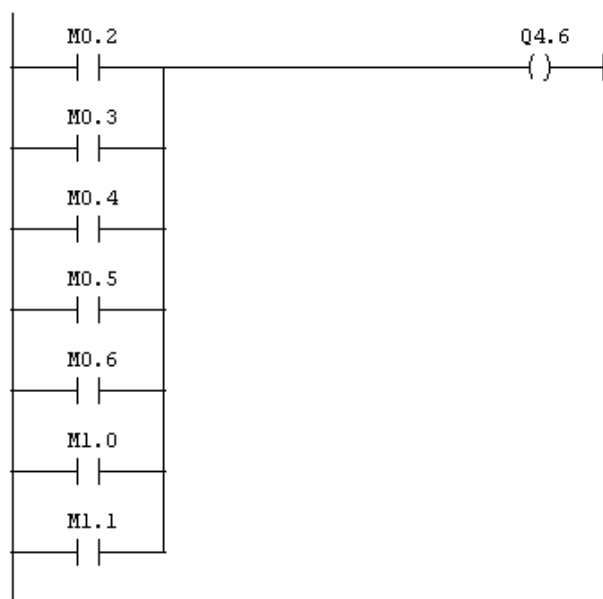
**Network 15 :** Control of a segment



**Network 16 :** Control of a segment



**Network 17:** Control of a segment



FC0 : Solution according to chap. 12.1 Seven-segment display

**Network 1:** Evaluation of number 0

AN	I	0.3
AN	I	0.2
AN	I	0.1
AN	I	0.0
=	M	0.0

**Network 2:** Evaluation of number 1

AN	I	0.3
AN	I	0.2
AN	I	0.1
A	I	0.0
=	M	0.1

**Network 3:** Evaluation of number 2

AN	I	0.3
AN	I	0.2
A	I	0.1
AN	I	0.0
=	M	0.2

**Network 4:** Evaluation of number 3

AN	I	0.3
AN	I	0.2
A	I	0.1
A	I	0.0
=	M	0.3

**Network 5:** Evaluation of number 4

AN	I	0.3
A	I	0.2
AN	I	0.1
AN	I	0.0
=	M	0.4



**Network 6 :** Evaluation of number 5

AN	I	0.3
A	I	0.2
AN	I	0.1
A	I	0.0
=	M	0.5

**Network 7 :** Evaluation of number 6

AN	I	0.3
A	I	0.2
A	I	0.1
AN	I	0.0
=	M	0.6

**Network 8 :** Evaluation of number 7

AN	I	0.3
A	I	0.2
A	I	0.1
A	I	0.0
=	M	0.7

**Network 9 :** Evaluation of number 8

A	I	0.3
AN	I	0.2
AN	I	0.1
AN	I	0.0
=	M	1.0

**Network 10 :** Evaluation of number 9

A	I	0.3
AN	I	0.2
AN	I	0.1
A	I	0.0
=	M	1.1

**Network 11:** Control of a segment

0	M	0.0
0	M	0.2
0	M	0.3
0	M	0.5
0	M	0.6
0	M	0.7
0	M	1.0
0	M	1.1
=	Q	4.0

**Network 12:** Control of a segment

0	M	0.0
0	M	0.1
0	M	0.2
0	M	0.3
0	M	0.4
0	M	0.7
0	M	1.0
0	M	1.1
=	Q	4.1

**Network 13:** Control of a segment

0	M	0.0
0	M	0.1
0	M	0.3
0	M	0.4
0	M	0.5
0	M	0.6
0	M	0.7
0	M	1.0
0	M	1.1
=	Q	4.2

**Network 14:** Control of a segment

0	M	0.0
0	M	0.2
0	M	0.3
0	M	0.5
0	M	0.6
0	M	1.0
0	M	1.1
=	Q	4.3

**Network 15 :** Control of a segment

0	M	0.0
0	M	0.2
0	M	0.6
0	M	1.0
=	Q	4.4

**Network 16 :** Control of a segment

0	M	0.0
0	M	0.4
0	M	0.5
0	M	0.6
0	M	1.0
0	M	1.1
=	Q	4.5

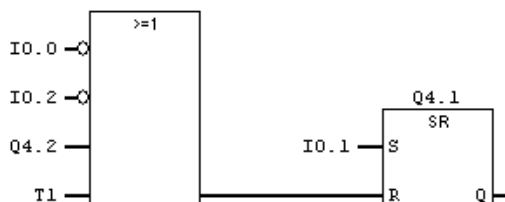
**Network 17 :** Control of a segment

0	M	0.2
0	M	0.3
0	M	0.4
0	M	0.5
0	M	0.6
0	M	1.0
0	M	1.1
=	Q	4.6

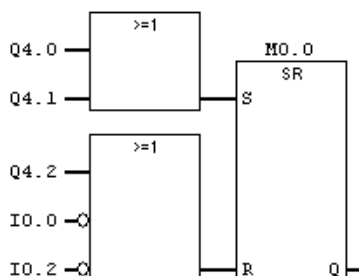
## Example of a solution according to chapter 12.2

FC1 : Solution according to chap. 12.2 Star-delta starting

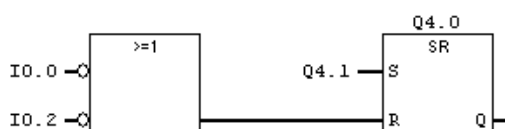
**Network 1:** Star starting



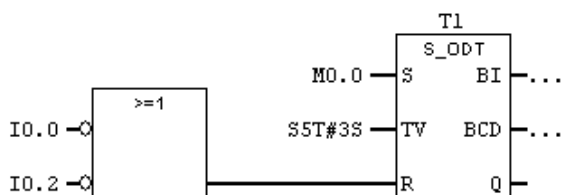
**Network 2:** Bit memory for start of T1



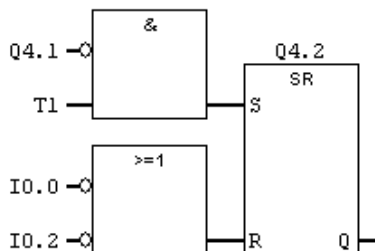
**Network 3:** Star starting



**Network 4 : Star starting**

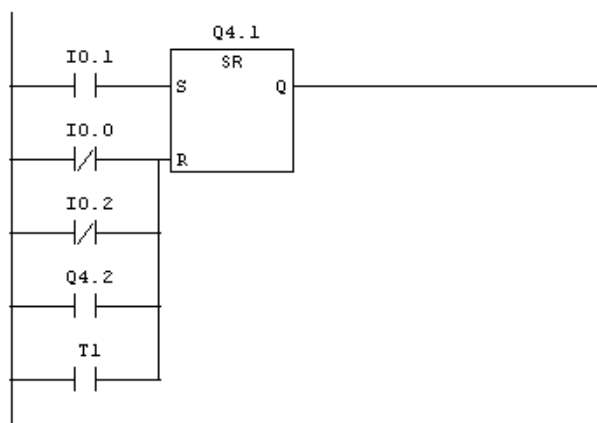


**Network 5 : Delta starting**

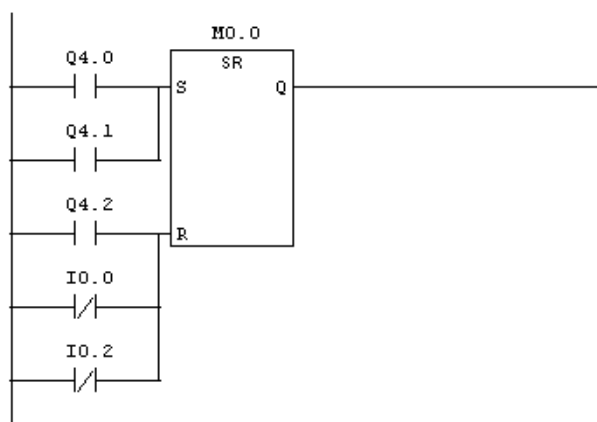


FC1 : Solution according to chap. 12.2 Star-delta starting

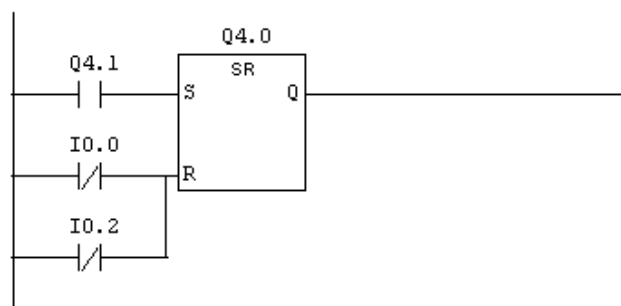
**Network 1:** Star starting



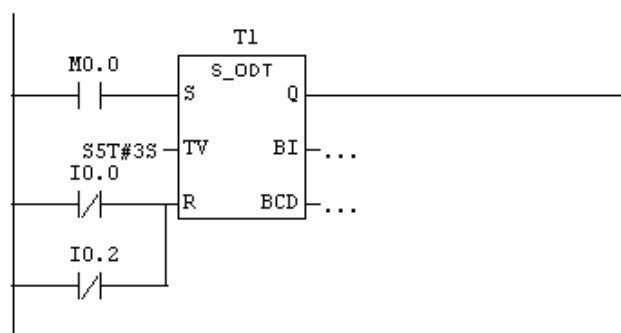
**Network 2:** Bit memory for start of T1



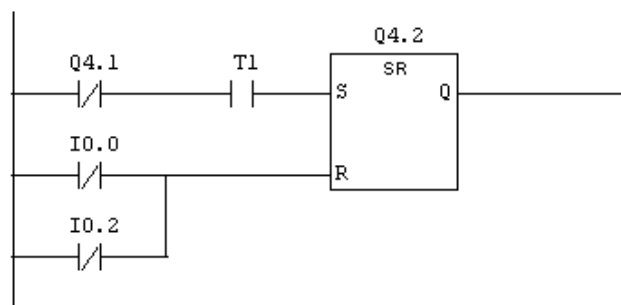
**Network 3 : Star starting**



**Network 4 : Star starting**



**Network 5 : Delta starting**



FC1 : Solution according to chap. 12.2 Star-delta starting

**Network 1:** Star starting

```

A      I      0.1           //ON button
S      Q      4.1           //Star contactor
A(
ON     I      0.0           //OFF button
ON     I      0.2           //Overcurrent relay
O      Q      4.2           //Delta contactor
O      T      1             //Wait time
)
R      Q      4.1           //Star contactor
NOP    O

```

**Network 2:** Bit memory for start of T1

```

A(
O      Q      4.0           //Mains contactor
O      Q      4.1           //Star contactor
)
S      M      0.0           //Bit memory for start of T1
A(
O      Q      4.2           //Delta contactor
ON     I      0.0           //OFF button
ON     I      0.2           //Overcurrent relay
)
R      M      0.0
NOP    O

```

**Network 3:** Star starting

```

A      Q      4.1           //Star contactor
S      Q      4.0           //Mains contactor
A(
ON     I      0.0           //OFF button
ON     I      0.2           //Overcurrent relay
)
R      Q      4.0           //Mains contactor
NOP    O

```



**Network 4 : Star starting**

A	M	0.0	
L	S5T#3S		
SD	T	1	//Wait time
A{			
ON	I	0.0	//OFF button
ON	I	0.2	
}			
R	T	1	
NOP	0		
NOP	0		
NOP	0		

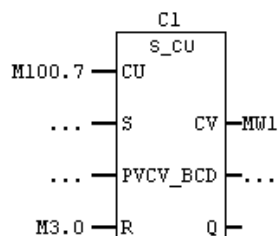
**Network 5 : Delta starting**

AN	Q	4.1	//Star contactor
A	T	1	//Wait time
S	Q	4.2	//Delta contactor
A{			
ON	I	0.0	//OFF button
ON	I	0.2	
}			
R	Q	4.2	//Delta contactor
NOP	0		

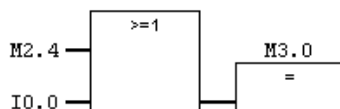
## Example of a solution according to chapter 12.3

FC2 : Solution according to chap. 12.3 Traffic light controller

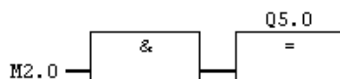
**Network 1:** Counter with clock frequency of 0.5 Hz



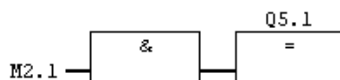
**Network 2:** Reset counter



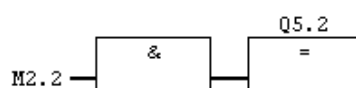
**Network 3:** Counter value 1



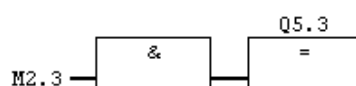
**Network 4:** Counter value 2



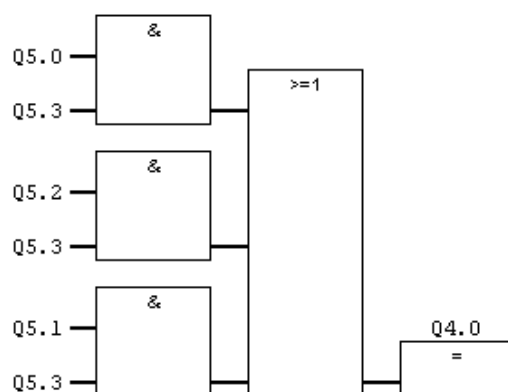
**Network 5 :** Counter value 4



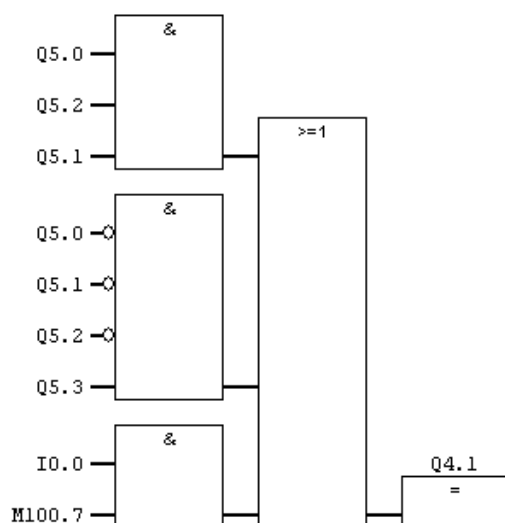
**Network 6 :** Counter value 8



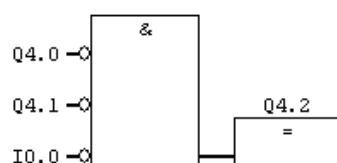
**Network 7 :** Coding of the traffic light: red for a car



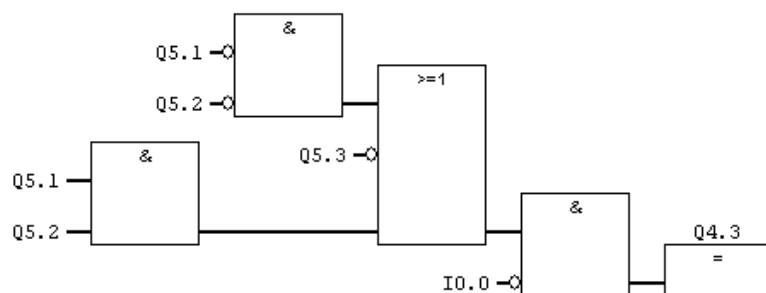
**Network 8:** Coding of the traffic light: yellow for a car



**Network 9:** Coding of the traffic light: green for a car



**Network 10:** Coding of the traffic light: red for a pedestrian

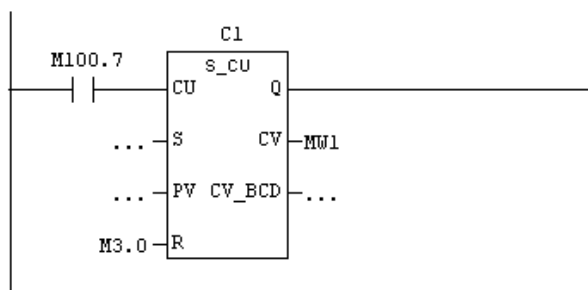


**Network 11:** Coding of the traffic light: green for a pedestrian

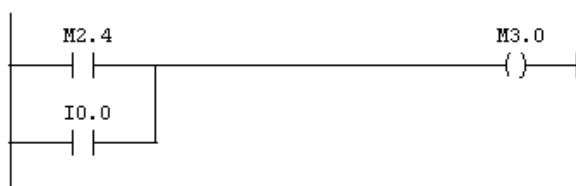


FC2 : Solution according to chap. 12.3 Traffic light controller

**Network 1:** Counter with clock frequency of 0.5 Hz



**Network 2:** Reset counter



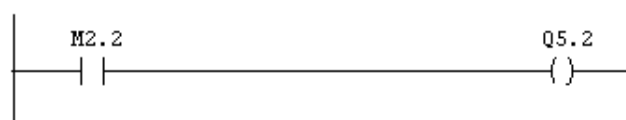
**Network 3:** Counter value 1



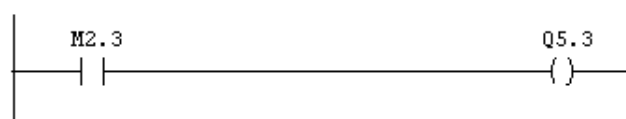
**Network 4:** Counter value 2



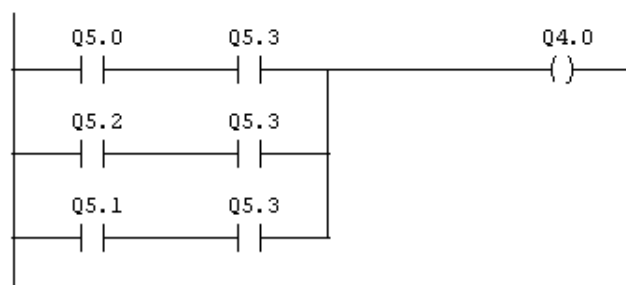
**Network 5 :** Counter value 4



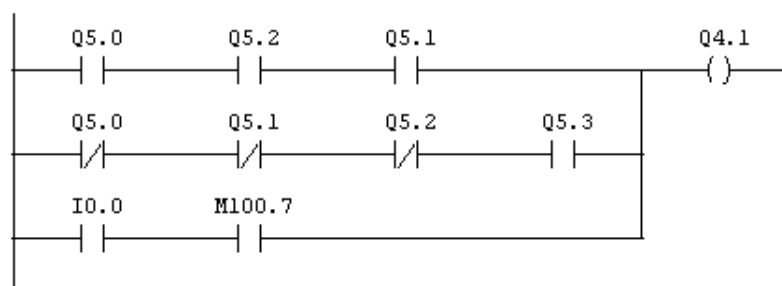
**Network 6 :** Counter value 8



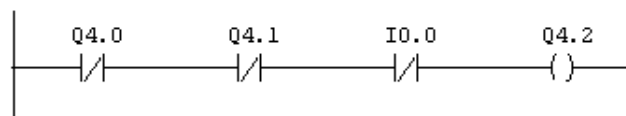
**Network 7 :** Coding of the traffic light: red for a car



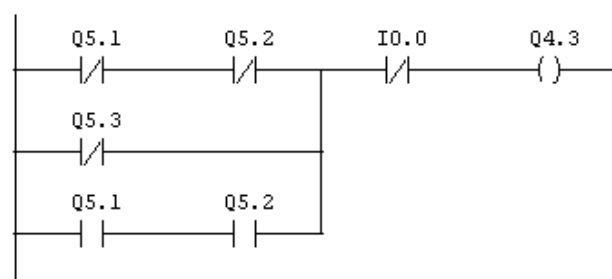
**Network 8 :** Coding of the traffic light: yellow for a car



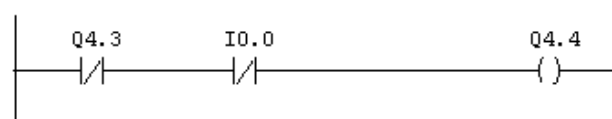
**Network 9 :** Coding of the traffic light: green for a car



**Network 10 :** Coding of the traffic light: red for a pedestrian



**Network 11 :** Coding of the traffic light: green for a pedestrian



FC2 : Solution according to chap. 12.3 Traffic light controller

**Network 1:** Counter with clock frequency of 0.5 Hz

A	M	100.7	//Clock memory 0.5 Hz
CU	C	1	//Cycle counter
BLD	101		
NOP	0		
NOP	0		
A	M	3.0	//Flag for reset counter
R	C	1	
L	C	1	
T	MW	1	//Cycle memory
NOP	0		
NOP	0		

**Network 2:** Reset counter

0	M	2.4	//Counter value 16
0	I	0.0	//Day-night switch
=	M	3.0	//Bit memory for reset counter

**Network 3:** Counter value 1

A	M	2.0	//Counter value 1
=	Q	5.0	

**Network 4:** Counter value 2

A	M	2.1	//Counter value 2
=	Q	5.1	

**Network 5:** Counter value 4

A	M	2.2	//Counter value 4
=	Q	5.2	

**Network 6:** Counter value 8

A	M	2.3	//Counter value 8
=	Q	5.3	



**Network 7 :** Coding of the traffic light: red for a car

```

A      Q      5.0      //Counter value 1
A      Q      5.3      //Counter value 8
O
A      Q      5.2      //Counter value 4
A      Q      5.3      //Counter value 8
O
A      Q      5.1      //Counter value 2
A      Q      5.3      //Counter value 8
=      Q      4.0

```

**Network 8 :** Coding of the traffic light: yellow for a car

```

A      Q      5.0      //Counter value 1
A      Q      5.2      //Counter value 4
A      Q      5.1      //Counter value 2
O
AN     Q      5.0      //Counter value 1
AN     Q      5.1      //Counter value 2
AN     Q      5.2      //Counter value 4
A      Q      5.3      //Counter value 8
O
A      I      0.0      //Day-night switch
A      M      100.7    //Clock memory 0.5 Hz
=      Q      4.1      //Car yellow

```

**Network 9 :** Coding of the traffic light: green for a car

```

AN     Q      4.0      //Car not red
AN     Q      4.1      //Car not yellow
AN     I      0.0      //Day-night switch
=      Q      4.2      //Car green

```

**Network 10 :** Coding of the traffic light: red for a pedestrian

```

A(
AN     Q      5.1      //Counter value 2
AN     Q      5.2      //Counter value 4
ON     Q      5.3      //Counter value 8
O
A      Q      5.1      //Counter value 2
A      Q      5.2      //Counter value 4
)
AN     I      0.0      //Day-night switch
=      Q      4.3      //Pedestrian red

```

**Network 11 :** Coding of the traffic light: green for a pedestrian

```

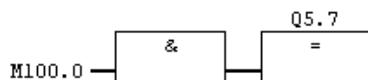
AN     Q      4.3      //Pedestrian red
AN     I      0.0      //Day-night switch
=      Q      4.4      //Pedestrian green

```

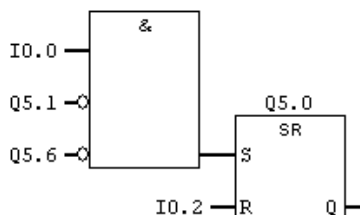
## Example of a solution according to chapter 12.4

FC4 : Solution according to 12.4 Conveyor belt controller

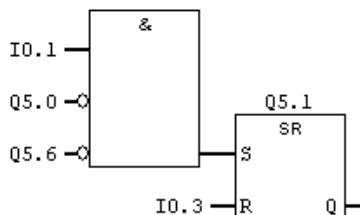
Network 1: 10 Hz clock-pulse rate



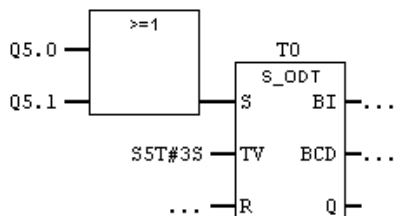
Network 2: ON lamp belt 1



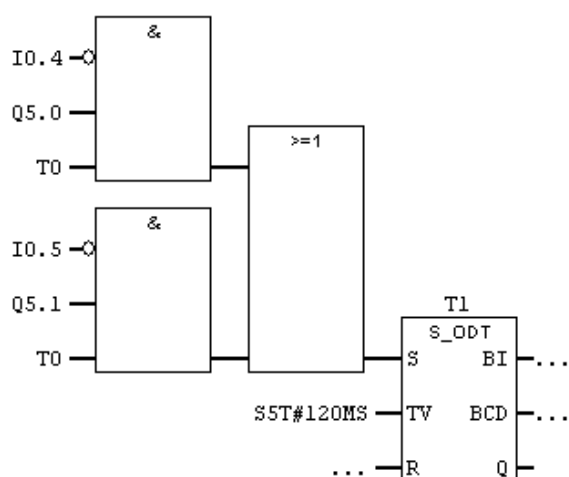
Network 3: ON lamp belt 2



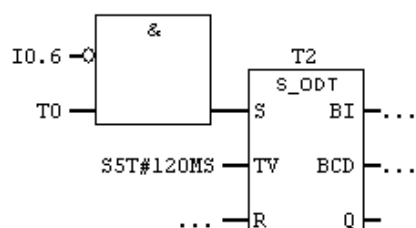
Network 4: Time for starting time: 3 seconds



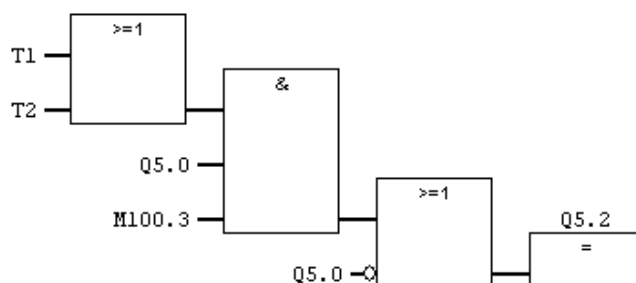
**Network 5:** Time for belt monitoring belt 1/2: 0.12 seconds



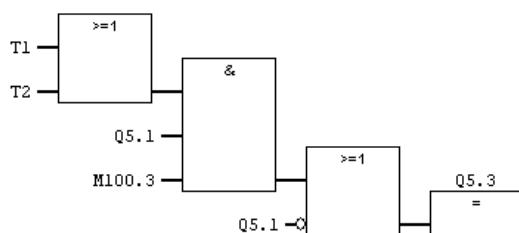
**Network 6:** Time for belt monitoring belt 3: 0.12 seconds



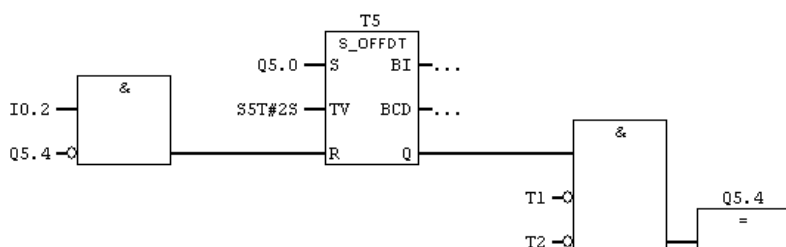
**Network 7:** OFF lamp belt 1



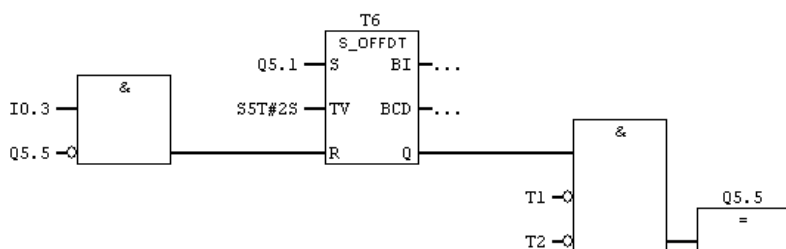
**Network 8 :** OFF lamp belt 2



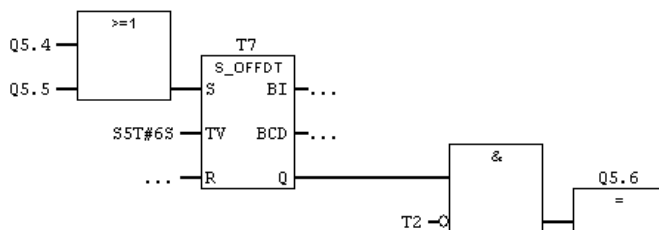
**Network 9 :** Drive M1 Switch-off delay



**Network 10 :** Drive M2 Switch-off delay



**Network 11 :** Drive M3 Switch-off delay

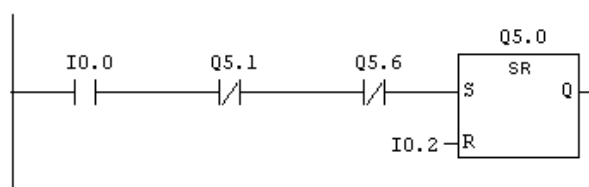


FC4 : Solution according to 12.4 Conveyor belt controller

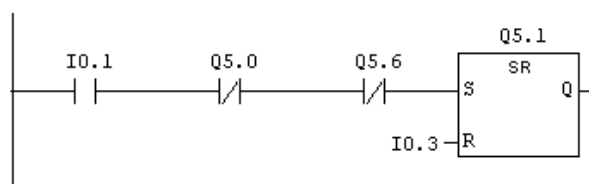
**Network 1:** 10 Hz clock-pulse rate



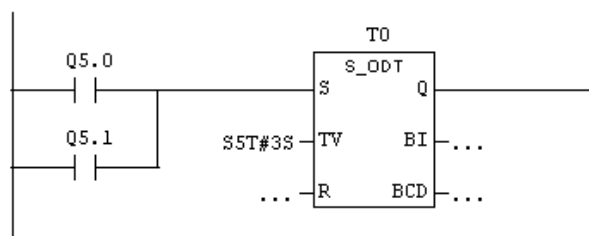
**Network 2:** ON lamp belt 1



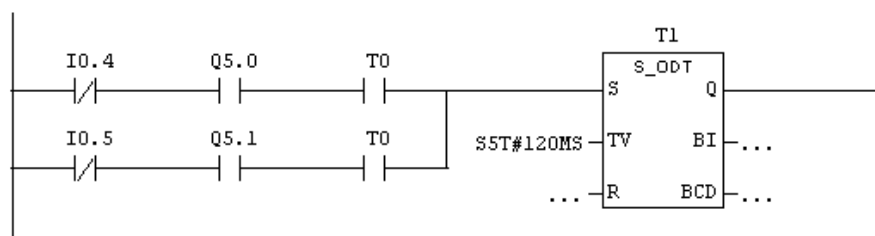
**Network 3:** ON lamp belt 2



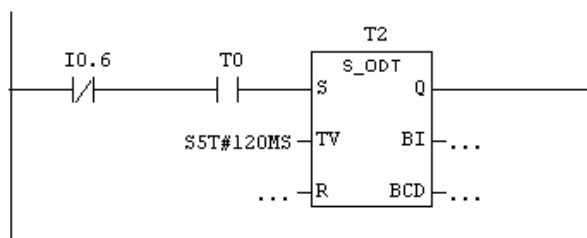
**Network 4:** Time for starting time: 3 seconds



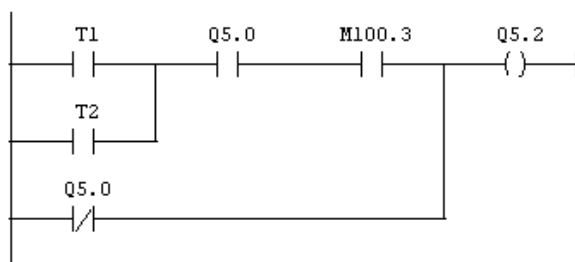
**Network 5:** Time for belt monitoring belt 1/2: 0.12 seconds



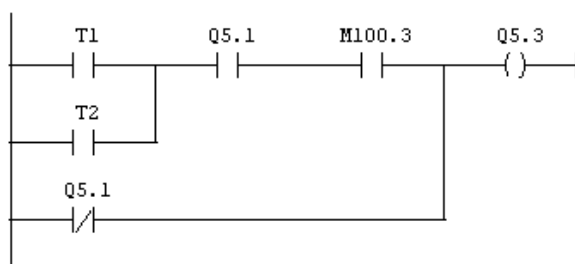
**Network 6:** Time for belt monitoring belt 3: 0.12 seconds



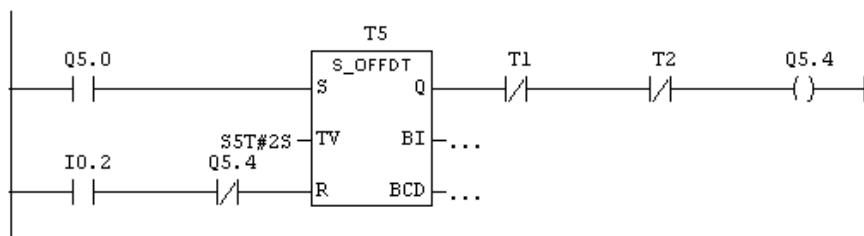
**Network 7:** OFF lamp belt 1



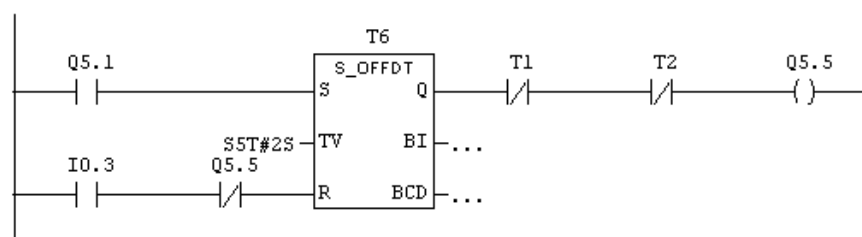
**Network 8:** OFF lamp belt 2



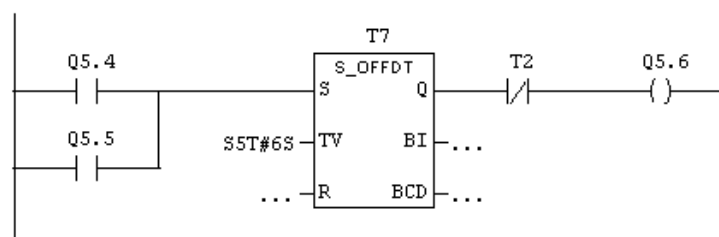
**Network 9:** Drive M1 Switch-off delay



**Network 10:** Drive M2 Switch-off delay



**Network 11:** Drive M3 Switch-off delay



FC4 : Solution according to 12.4 Conveyor belt controller

Network 1: 10 Hz clock-pulse rate

A	M	100.0	//Bit memory for 10-Hz clock generator
=	Q	5.7	//10-Hz clock for simulator

Network 2: ON lamp belt 1

A	I	0.0	//ON button belt 1
AN	Q	5.1	//ON lamp belt 2
AN	Q	5.6	//Drive M3
S	Q	5.0	//ON lamp belt 1
A	I	0.2	//OFF button belt 1
R	Q	5.0	
NOP	O		

Network 3: ON lamp belt 2

A	I	0.1	//ON button belt 2
AN	Q	5.0	//ON lamp belt 1
AN	Q	5.6	//Drive M3
S	Q	5.1	//ON lamp belt 2
A	I	0.3	//OFF button belt 2
R	Q	5.1	
NOP	O		

Network 4: Time for starting time: 3 seconds

A(			
O	Q	5.0	//ON lamp belt 1
O	Q	5.1	//ON lamp belt 2
)			
L	S5T#3S		//Starting time
SD	T	0	
NOP	O		
NOP	O		
NOP	O		
NOP	O		



**Network 5 :** Time for belt monitoring belt 1/2: 0.12 seconds

```

A(
AN  I      0.4           //Monitor belt 1
A   Q      5.0           //ON lamp belt 1
A   T      0             //Starting time
)
AN  I      0.5           //Monitor belt 2
A   Q      5.1           //ON lamp belt 2
A   T      0
)
L   SST#120MS           //Belt monitoring belt 1/2
SD  T      1
NOP 0
NOP 0
NOP 0
NOP 0

```

**Network 6 :** Time for belt monitoring belt 3: 0.12 seconds

```

AN  I      0.6           //Monitor belt 3
A   T      0             //Starting time
L   SST#120MS
SD  T      2
NOP 0
NOP 0
NOP 0
NOP 0

```

**Network 7 :** OFF lamp belt 1

```

A(
O   T      1             //Belt monitoring belt 1/2
O   T      2             //Belt monitoring belt 3
)
A   Q      5.0           //ON lamp belt 1
A   M      100.3         //Bit memory for 2-Hz clock generator
ON  Q      5.0           //ON lamp belt 1
=   Q      5.2           //OFF lamp belt 1

```

**Network 8 :** OFF lamp belt 2

```

A(
O   T      1             //Belt monitoring belt 1/2
O   T      2             //Belt monitoring belt 3
)
A   Q      5.1           //ON lamp belt 2
A   M      100.3         //Bit memory for 2-Hz clock generator

ON  Q      5.1           //ON lamp belt 2
=   Q      5.3           //OFF lamp belt 2

```

**Network 9 : Drive M1 Switch-off delay**

```

A(
A      Q      5.0          //ON lamp belt 1
L      S5T#2S
SF     T      5          //Switch-off delay belt 1
A      I      0.2        //OFF button belt 1
AN     Q      5.4        //Drive M1
R      T      5
NOP    O
NOP    O
A      T      5
)
AN     T      1
AN     T      2
=      Q      5.4        //Drive M1

```

**Network 10 : Drive M2 Switch-off delay**

```

A(
A      Q      5.1          //ON lamp belt 2
L      S5T#2S
SF     T      6          //Switch-off delay belt 2
A      I      0.3        //OFF button belt 2
AN     Q      5.5        //Drive M2
R      T      6
NOP    O
NOP    O
A      T      6          //Switch-off delay belt 2
)
AN     T      1          //Belt monitoring belt 1/2
AN     T      2          //Belt monitoring belt 3
=      Q      5.5        //Drive M2

```

**Network 11 : Drive M3 Switch-off delay**

```

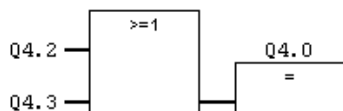
A(
A(
O      Q      5.4          //Drive M1
O      Q      5.5          //Drive M2
)
L      S5T#6S
SF     T      7          //Switch-off delay belt 3
NOP    O
NOP    O
NOP    O
A      T      7          //Switch-off delay belt 3
)
AN     T      2
=      Q      5.6        //Drive M3

```

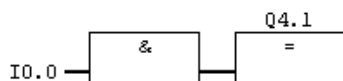
## Example of a solution according to chapter 12.5

FC0 : Solution according to chap. 12.5 Reaction vessel

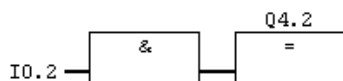
**Network 1:** Operation circulator



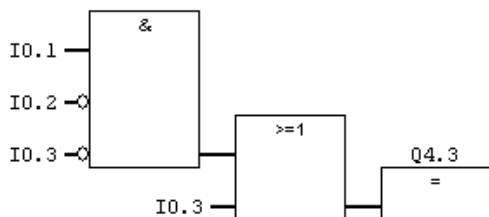
**Network 2:** Operation safety valve



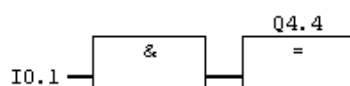
**Network 3:** Operation cooling water intake



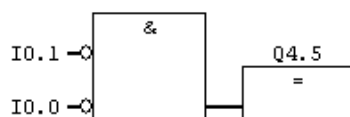
**Network 4:** Operation heater



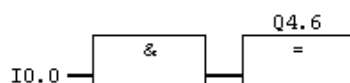
**Network 5 :** Operation state: Start-up



**Network 6 :** Operation state: Normal mode

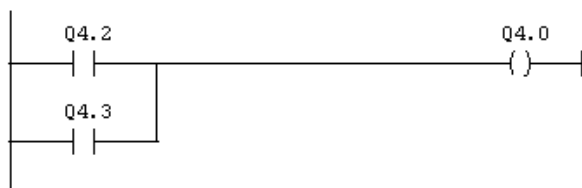


**Network 7 :** Operation state: Alarm

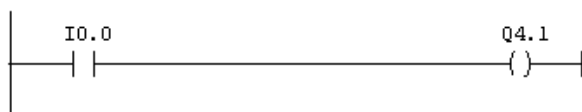


FC0 : Solution according to chap. 12.5 Reaction vessel

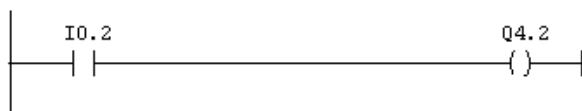
**Network 1:** Operation circulator



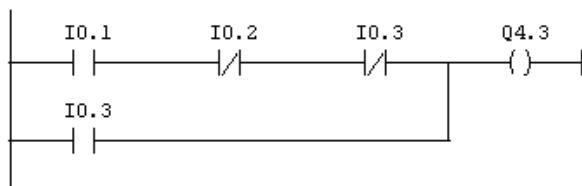
**Network 2:** Operation safety valve



**Network 3:** Operation cooling water intake



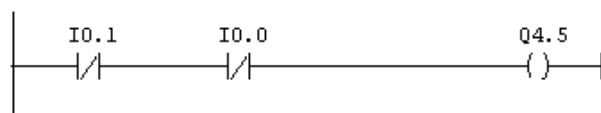
**Network 4:** Operation heater



**Network 5 :** Operation state: Start-up



**Network 6 :** Operation state: Normal mode



**Network 7 :** Operation state: Alarm



FCO : Solution according to chap. 12.5 Reaction vessel

**Network 1:** Operation circulator

0	Q	4.2	//Cooling water intake
0	Q	4.3	//Heater
=	Q	4.0	//Circulator

**Network 2:** Operation safety valve

A	I	0.0	//Pressure is too high
=	Q	4.1	//Safety valve

**Network 3:** Operation cooling water intake

A	I	0.2	//Temperature is too high
=	Q	4.2	//Cooling water intake

**Network 4:** Operation heater

A	I	0.1	//Pressure is too low
AN	I	0.2	//Temperature is not too high
AN	I	0.3	//Temperature is not too low
0	I	0.3	//Temperature is too low
=	Q	4.3	

**Network 5:** Operation state: Start-up

A	I	0.1	//Pressure is too low
=	Q	4.4	//Start-up

**Network 6:** Operation state: Normal mode

AN	I	0.1	//Pressure is not too low
AN	I	0.0	//Pressure is not too high
=	Q	4.5	//Normal mode

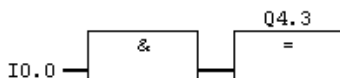
**Network 7:** Operation state: Alarm

A	I	0.0	//Pressure is too high
=	Q	4.6	//Alarm

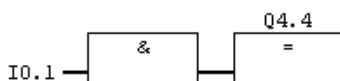
## Example of a solution according to chapter 12.6

FC0 : Solution according to chap. 12.6 Container filling system

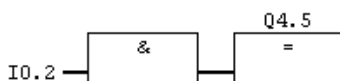
**Network 1:** Empty container 1



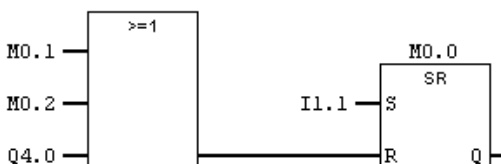
**Network 2:** Empty container 2



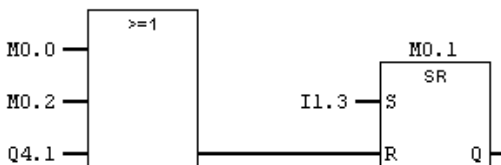
**Network 3:** Empty container 3



**Network 4:** Container 1 empty

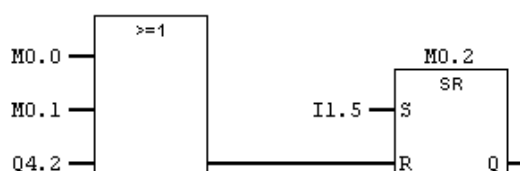


**Network 5:** Container 2 empty

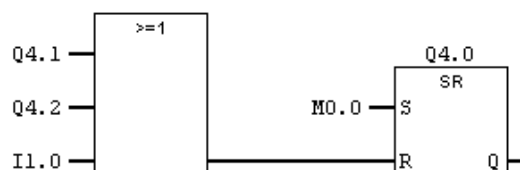




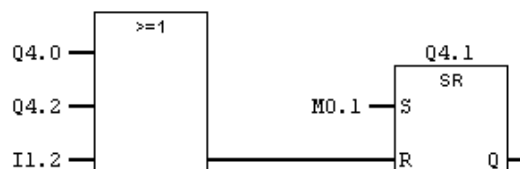
**Network 6 : Container 3 empty**



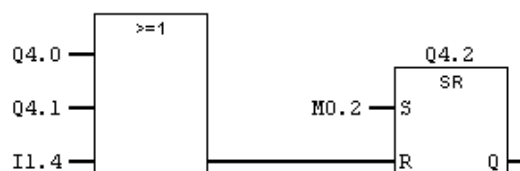
**Network 7 : Solenoid valve 1**



**Network 8 : Solenoid valve 2**



**Network 9 : Solenoid valve 3**

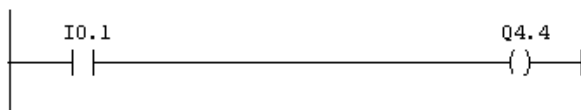


FC0 : Solution according to chap. 12.6 Container filling system

**Network 1:** Empty container 1



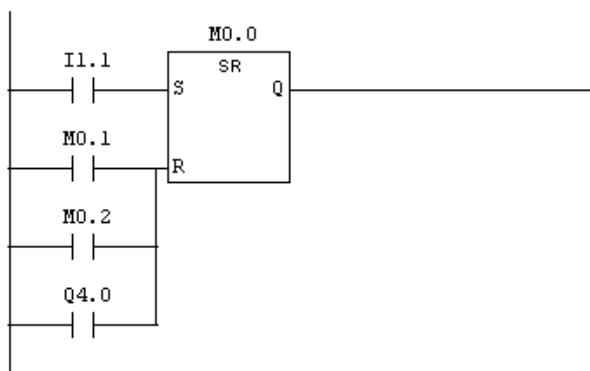
**Network 2:** Empty container 2



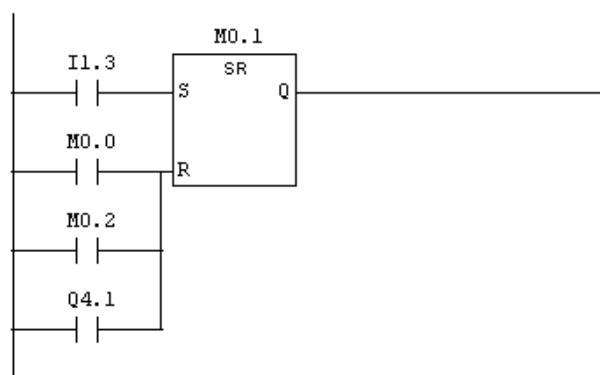
**Network 3:** Empty container 3



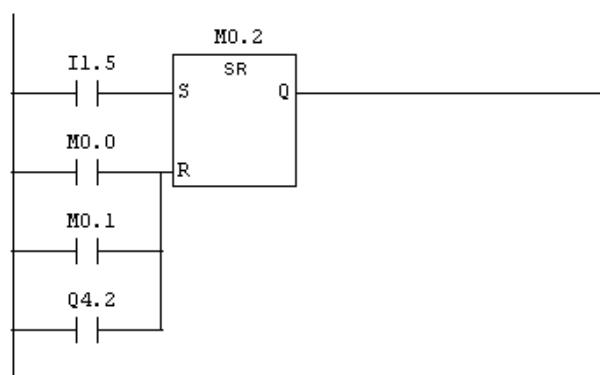
**Network 4:** Container 1 empty



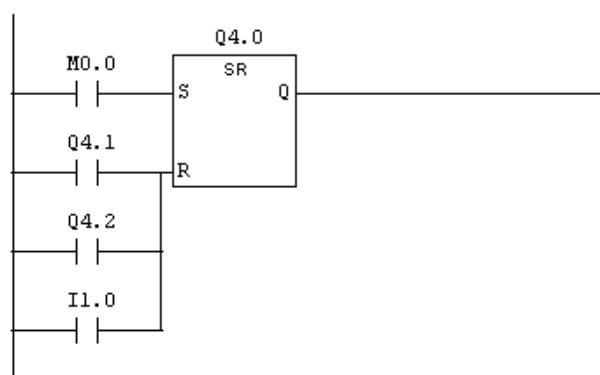
**Network 5 :** Container 2 empty



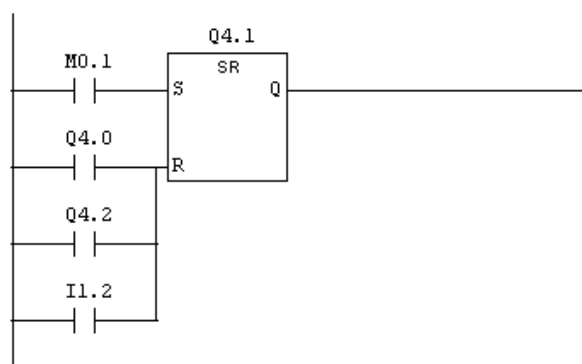
**Network 6 :** Container 3 empty



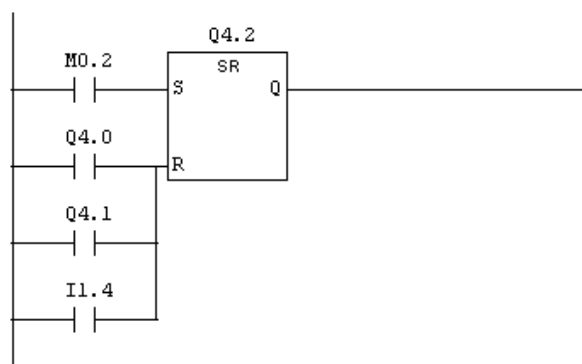
**Network 7 :** Solenoid valve 1



**Network 8 :** Solenoid valve 2



**Network 9 :** Solenoid valve 3



**Network 1:** Empty container 1

```

A      I      0.0      //Empty container 1
=      Q      4.3      //valve 4

```

**Network 2:** Empty container 2

```

A      I      0.1      //Empty container 2
=      Q      4.4      //Valve 5

```

**Network 3:** Empty container 3

```

A      I      0.2      //Empty container 3
=      Q      4.5      //Valve 6

```

**Network 4:** Container 1 empty

```

A      I      1.1      //Container 1 empty
S      M      0.0      //Container 1 empty
A(
O      M      0.1      //Container 2 empty
O      M      0.2      //Container 3 empty
O      Q      4.0      //Valve 1
)
R      M      0.0      //Container 1 empty
NOP    O

```

**Network 5:** Container 2 empty

```

A      I      1.3      //Container 2 empty
S      M      0.1      //Container 2 empty
A(
O      M      0.0      //Container 1 empty
O      M      0.2      //Container 3 empty
O      Q      4.1      //Valve 2
)
R      M      0.1      //Container 2 empty
NOP    O

```

**Network 6 : Container 3 empty**

A	I	1.5	//Container 3 empty
S	M	0.2	//Container 3 empty
A(			
O	M	0.0	//Container 1 empty
O	M	0.1	//Container 2 empty
O	Q	4.2	//Valve 3
)			
R	M	0.2	//Container 3 empty
NOP	O		

**Network 7 : Solenoid valve 1**

A	M	0.0	//Container 1 empty
S	Q	4.0	//Valve 1
A(			
O	Q	4.1	//Valve 2
O	Q	4.2	//Valve 3
O	I	1.0	//Container 1 full
)			
R	Q	4.0	//Valve 1
NOP	O		

**Network 8 : Solenoid valve 2**

A	M	0.1	//Container 2 empty
S	Q	4.1	//Valve 2
A(			
O	Q	4.0	//Valve 1
O	Q	4.2	//Valve 3
O	I	1.2	//Container 3 full
)			
R	Q	4.1	//Valve 1
NOP	O		

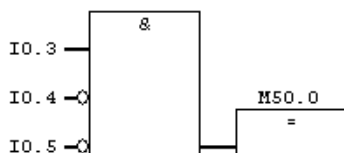
**Network 9 : Solenoid valve 3**

A	M	0.2	//Container 3 empty
S	Q	4.2	//Valve 3
A(			
O	Q	4.0	//Valve 1
O	Q	4.1	//Valve 2
O	I	1.4	//Container 3 full
)			
R	Q	4.2	//Valve 3
NOP	O		

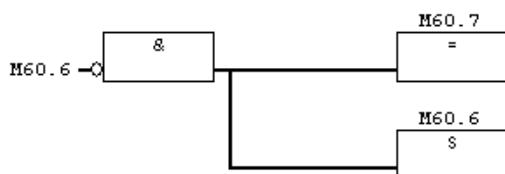
## Example of a solution according to chapter 12.7

FC1 : Solution according to chap. 12.7 Automatic tablet filler

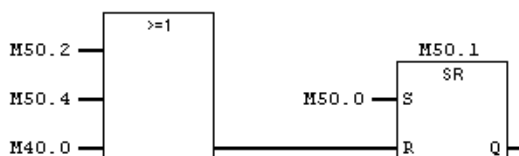
**Network 1:** Intermediate bit memory 3 (signal preparation)



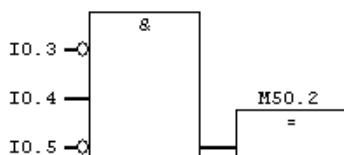
**Network 2:** Switch-on bit memory



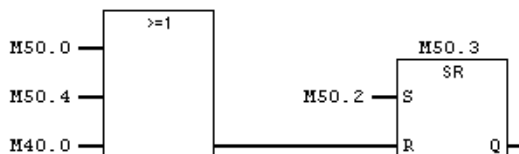
**Network 3:** Tablet memorizing counter 3 (signal preparation)



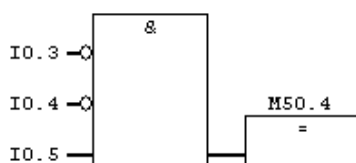
**Network 4:** Intermediate bit memory 5 (signal preparation)



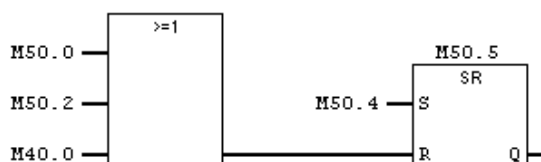
**Network 5:** Tablet memorizing counter 5 (signal preparation)



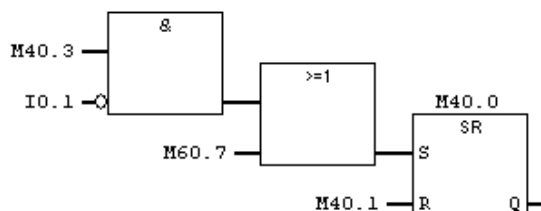
**Network 6 :** Intermediate bit memory 7 (signal preparation)



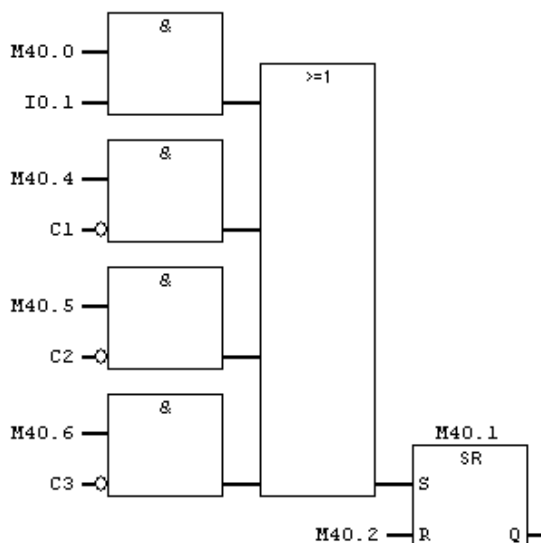
**Network 7 :** Tablet memorizing counter 7 (signal preparation)



**Network 8 :** State 0

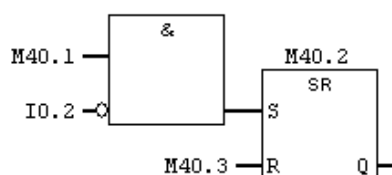


**Network 9 :** State 1

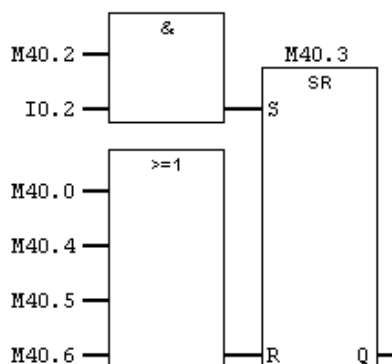




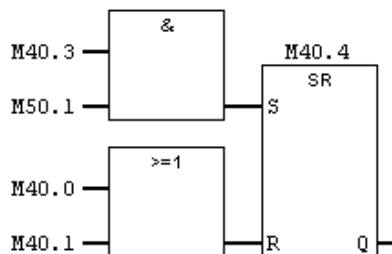
**Network 10 : State 2**



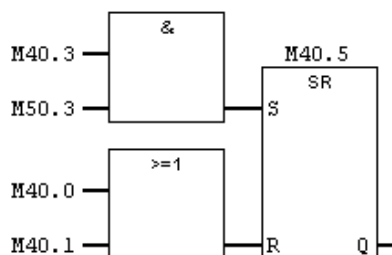
**Network 11 : State 3**



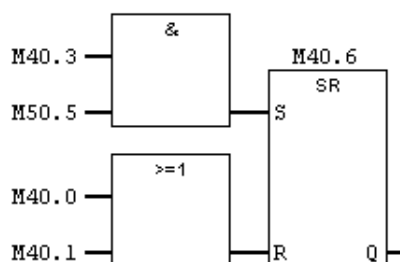
**Network 12 : State 4**



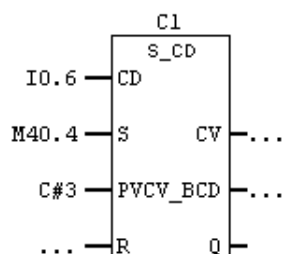
**Network 13 : State 5**



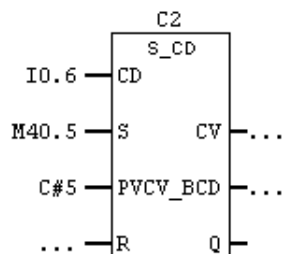
**Network 14 :** State 6



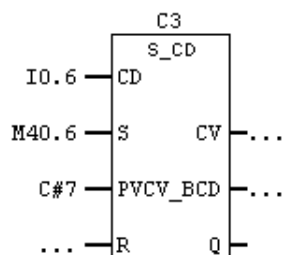
**Network 15 :** Down counter for 3 tablets



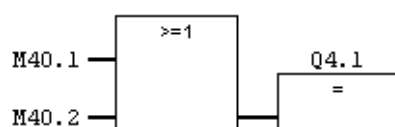
**Network 16 :** Down counter for 5 tablets



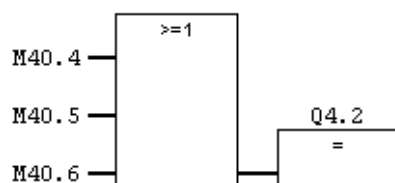
**Network 17 :** Down counter for 7 tablets



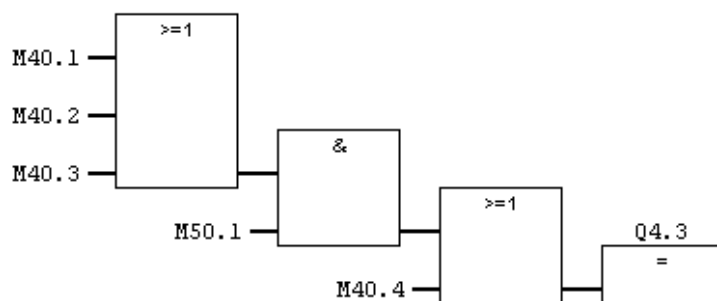
**Network 18 : Belt drive motor**



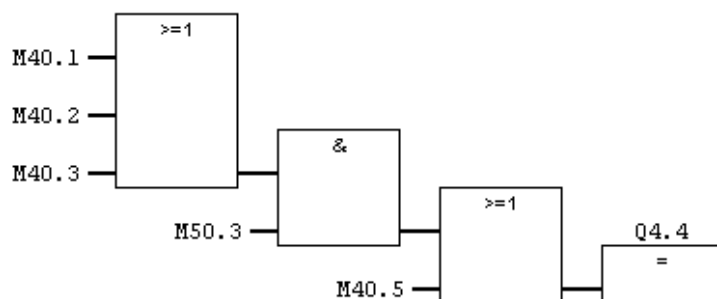
**Network 19 : Valve**



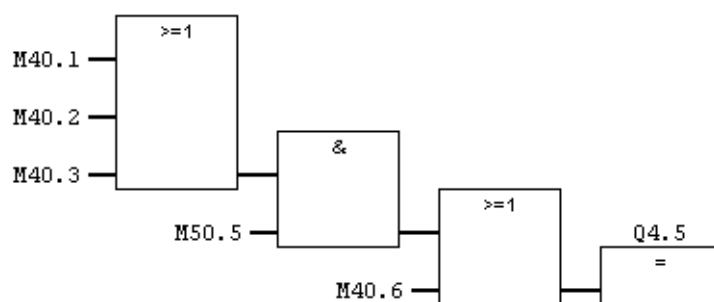
**Network 20 : Lamp P1 for 3 tablets**



**Network 21 : Lamp P2 for 5 tablets**

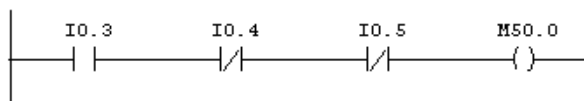


**Network 22:** Lamp P3 for 7 tablets

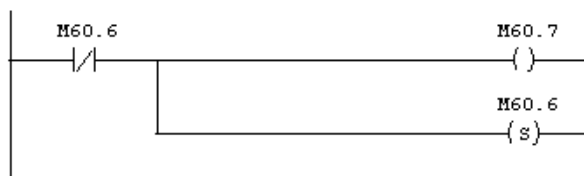


FC1 : Solution according to chap. 12.7 Automatic tablet filler

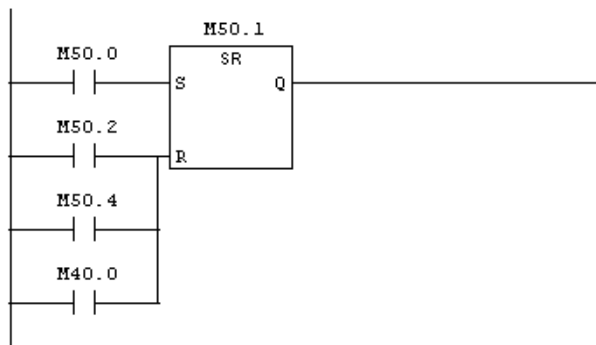
**Network 1:** Intermediate bit memory 3 (signal preparation)



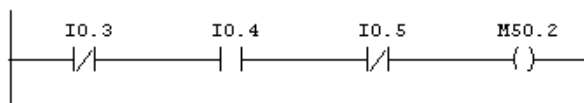
**Network 2:** Switch-on bit memory



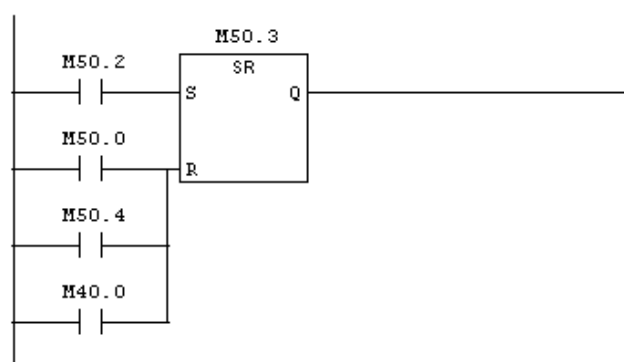
**Network 3:** Tablet memorizing counter 3 (signal preparation)



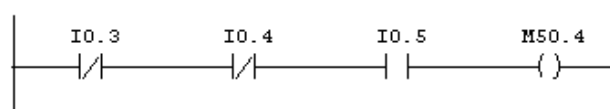
**Network 4:** Intermediate bit memory 5 (signal preparation)



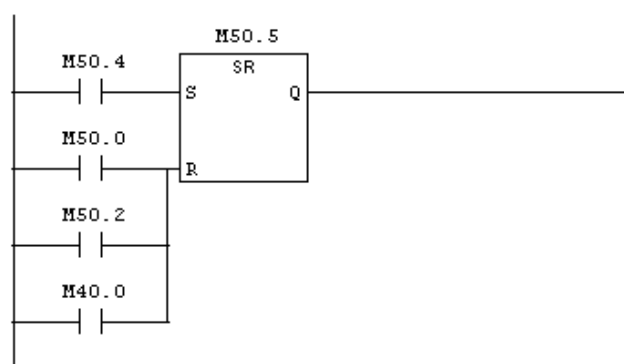
**Network 5:** Tablet memorizing counter 5 (signal preparation)



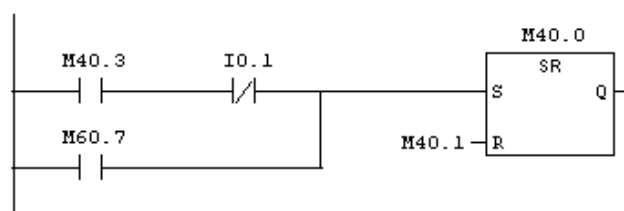
**Network 6:** Intermediate bit memory 7 (signal preparation)



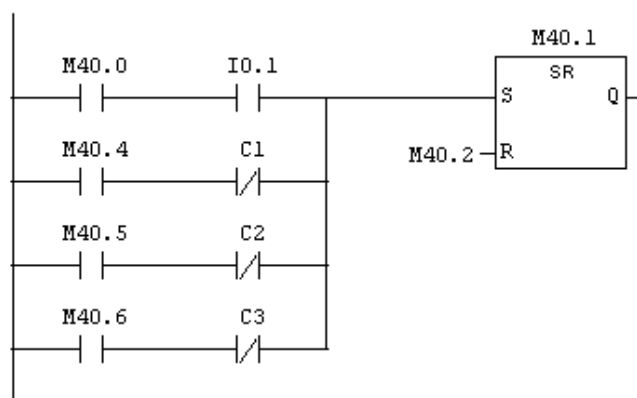
**Network 7:** Tablet memorizing counter 7 (signal preparation)



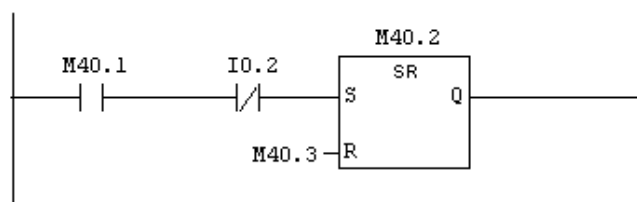
**Network 8:** State 0



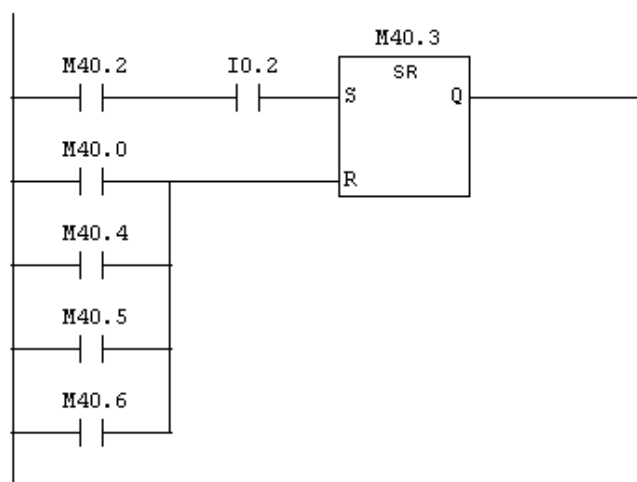
**Network 9 : State 1**



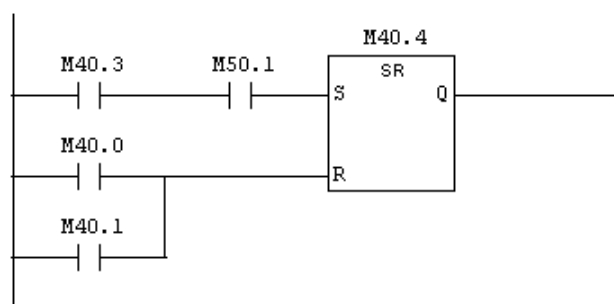
**Network 10 : State 2**



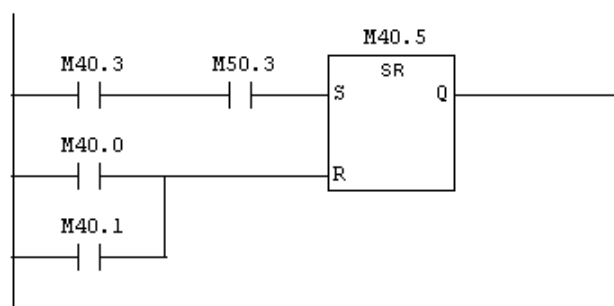
**Network 11 : State 3**



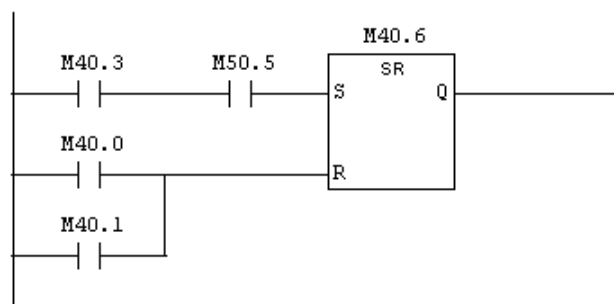
**Network 12 : State 4**



**Network 13 : State 5**

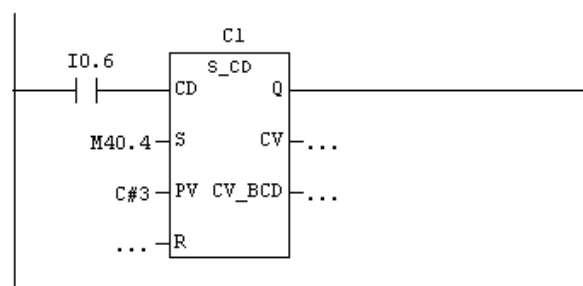


**Network 14 : State 6**

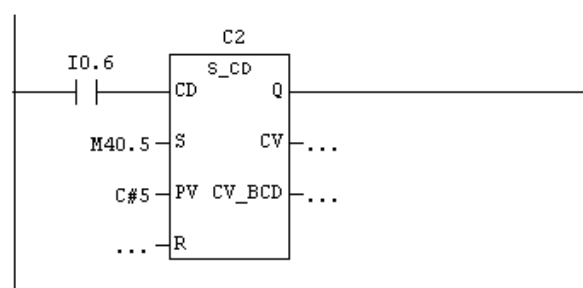




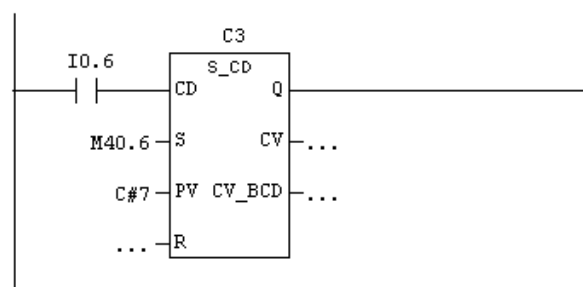
**Network 15:** Down counter for 3 tablets



**Network 16:** Down counter for 5 tablets



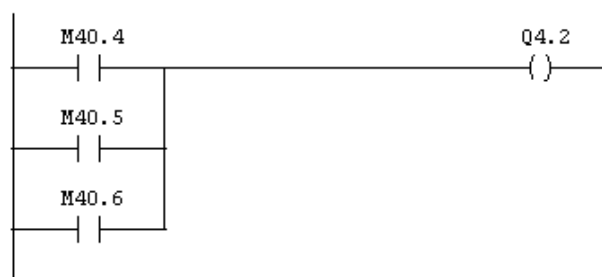
**Network 17:** Down counter for 7 tablets



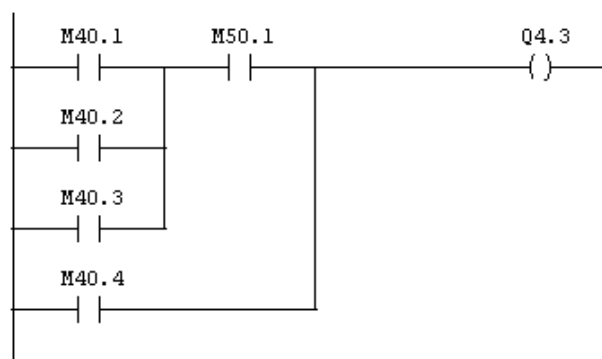
**Network 18:** Belt drive motor



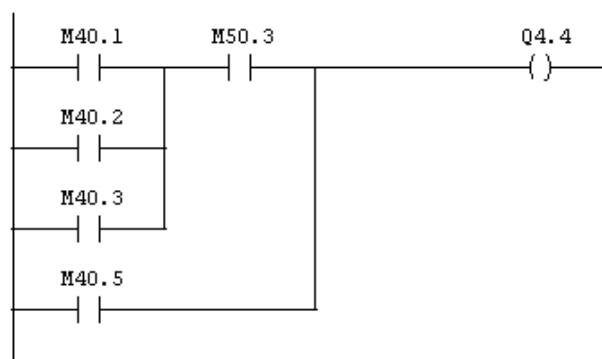
**Network 19 : Valve**



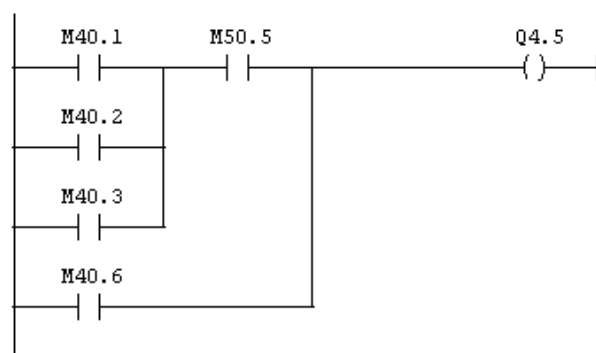
**Network 20 : Lamp P1 for 3 tablets**



**Network 21 : Lamp P2 for 5 tablets**



**Network 22 :** Lamp P3 for 7 tablets



FC1 : Solution according to chap. 12.7 Automatic tablet filler

**Network 1:** Intermediate bit memory 3 (signal preparation)

A	I	0.3
AN	I	0.4
AN	I	0.5
=	M	50.0

**Network 2:** Switch-on bit memory

AN	M	60.6
=	M	60.7
S	M	60.6

**Network 3:** Tablet memorizing counter 3 (signal preparation)

A	M	50.0
S	M	50.1
A{		
O	M	50.2
O	M	50.4
O	M	40.0
}		
R	M	50.1
NOP	O	

**Network 4:** Intermediate bit memory 5 (signal preparation)

AN	I	0.3
A	I	0.4
AN	I	0.5
=	M	50.2

**Network 5:** Tablet memorizing counter 5 (signal preparation)

A	M	50.2
S	M	50.3
A{		
O	M	50.0
O	M	50.4
O	M	40.0
}		
R	M	50.3
NOP	O	

**Network 6 :** Intermediate bit memory 7 (signal preparation)

AN	I	0.3
AN	I	0.4
A	I	0.5
=	M	50.4

**Network 7 :** Tablet memorizing counter 7 (signal preparation)

A	M	50.4
S	M	50.5
A(		
O	M	50.0
O	M	50.2
O	M	40.0
)		
R	M	50.5
NOP	O	

**Network 8 :** State 0

A(		
A	M	40.3
AN	I	0.1
O	M	60.7
)		
S	M	40.0
A	M	40.1
R	M	40.0
NOP	O	

**Network 9 :** State 1

A(		
A	M	40.0
A	I	0.1
O		
A	M	40.4
AN	C	1
O		
A	M	40.5
AN	C	2
O		
A	M	40.6
AN	C	3
)		
S	M	40.1
A	M	40.2
R	M	40.1
NOP	O	

**Network 10 : State 2**

A	M	40.1
AN	I	0.2
S	M	40.2
A	M	40.3
R	M	40.2
NOP	0	

**Network 11 : State 3**

A	M	40.2
A	I	0.2
S	M	40.3
A(		
O	M	40.0
O	M	40.4
O	M	40.5
O	M	40.6
)		
R	M	40.3
NOP	0	

**Network 12 : State 4**

A	M	40.3
A	M	50.1
S	M	40.4
A(		
O	M	40.0
O	M	40.1
)		
R	M	40.4
NOP	0	

**Network 13 : State 5**

A	M	40.3
A	M	50.3
S	M	40.5
A(		
O	M	40.0
O	M	40.1
)		
R	M	40.5
NOP	0	

**Network 14 : State 6**

A	M	40.3
A	M	50.5
S	M	40.6
A(		
O	M	40.0
O	M	40.1
)		
R	M	40.6
NOP	O	

**Network 15 : Down counter for 3 tablets**

A	I	0.6
CD	C	1
BLD	101	
A	M	40.4
L	C#3	
S	C	1
NOP	O	
NOP	O	
NOP	O	
NOP	O	

**Network 16 : Down counter for 5 tablets**

A	I	0.6
CD	C	2
BLD	101	
A	M	40.5
L	C#5	
S	C	2
NOP	O	
NOP	O	
NOP	O	
NOP	O	

**Network 17 : Down counter for 7 tablets**

A	I	0.6
CD	C	3
BLD	101	
A	M	40.6
L	C#7	
S	C	3
NOP	O	
NOP	O	
NOP	O	
NOP	O	

**Network 18 :** Belt drive motor

0	M	40.1
0	M	40.2
=	Q	4.1

**Network 19 :** Valve

0	M	40.4
0	M	40.5
0	M	40.6
=	Q	4.2

**Network 20 :** Lamp P1 for 3 tablets

A(		
0	M	40.1
0	M	40.2
0	M	40.3
)		
A	M	50.1
0	M	40.4
=	Q	4.3

**Network 21 :** Lamp P2 for 5 tablets

A(		
0	M	40.1
0	M	40.2
0	M	40.3
)		
A	M	50.3
0	M	40.5
=	Q	4.4

**Network 22 :** Lamp P3 for 7 tablets

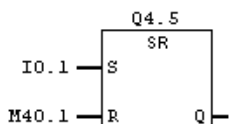
A(		
0	M	40.1
0	M	40.2
0	M	40.3
)		
A	M	50.5
0	M	40.6
=	Q	4.5



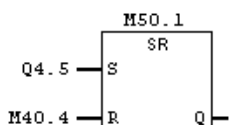
## Example of a solution according to chapter 12.8

FC5 : Solution according to chap. 12.8 Door access control system

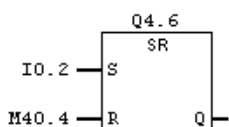
**Network 1:** Button S1



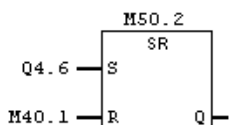
**Network 2:** Button S1



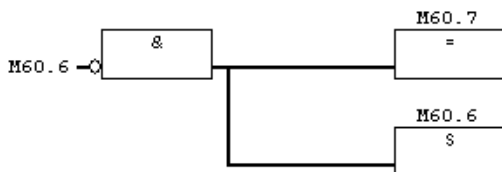
**Network 3:** Button S2



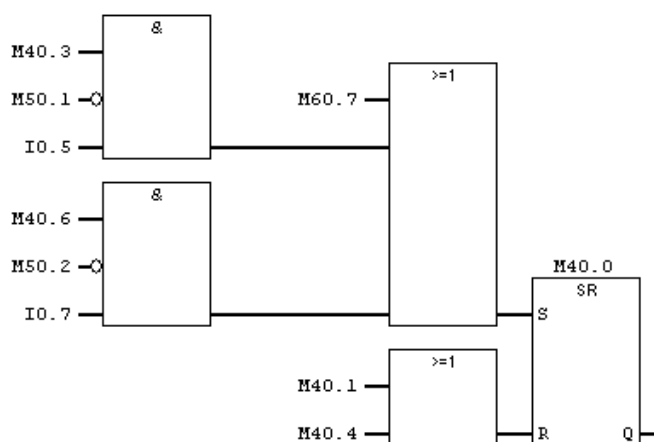
**Network 4:** Button S2



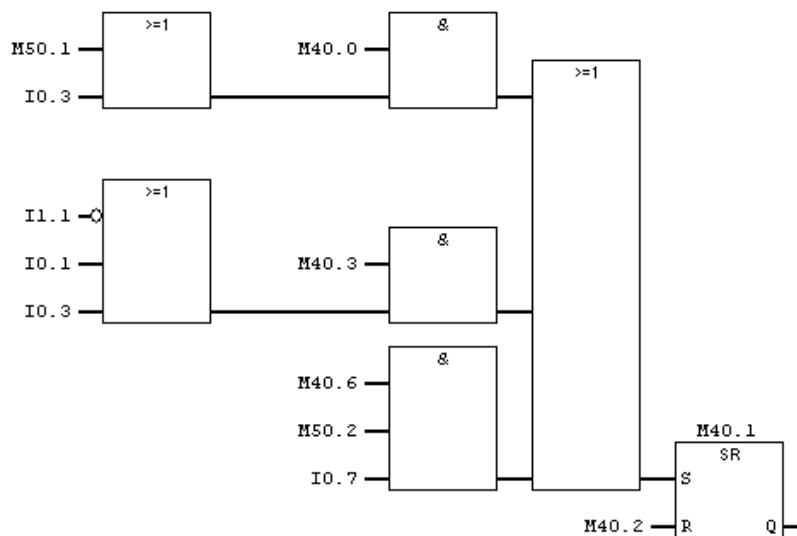
**Network 5:** Switch-on/setting pulse bit memory



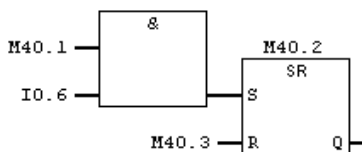
**Network 6 : Status bit memory 0**



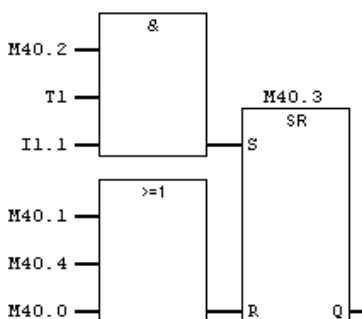
**Network 7 : Status bit memory 1**



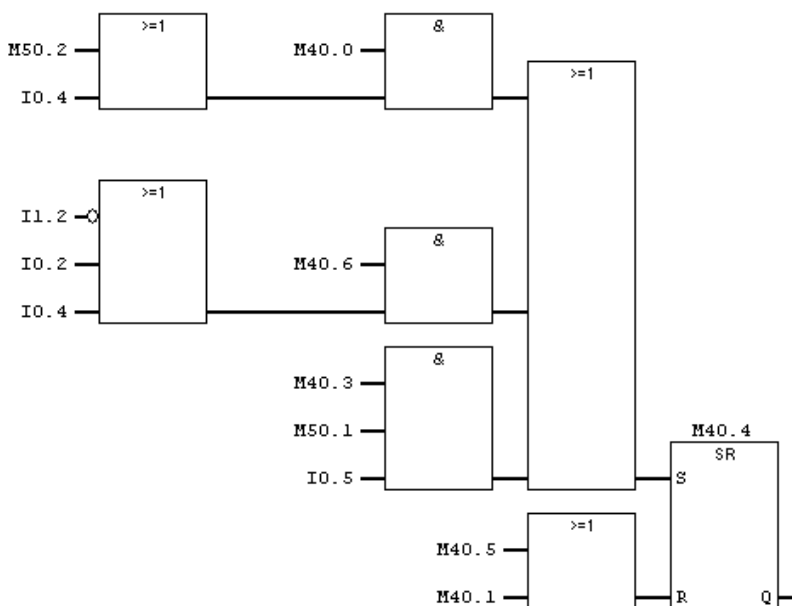
**Network 8 : Status bit memory 2**



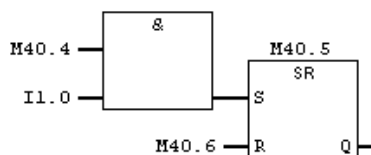
**Network 9 : Status bit memory 3**



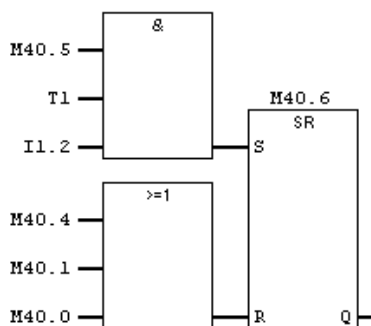
**Network 10 : Status bit memory 4**



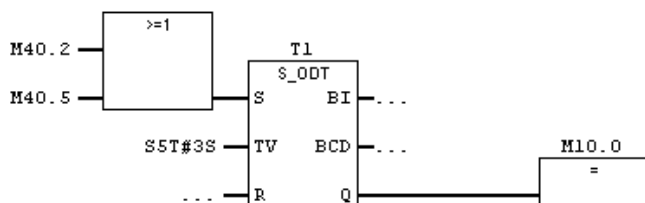
**Network 11:** Status bit memory 5



**Network 12:** Status bit memory 6



**Network 13:** Wait time



**Network 14:** Motor door 1 open



**Network 15 :** Motor door 1 closed



**Network 16 :** Motor door 2 open

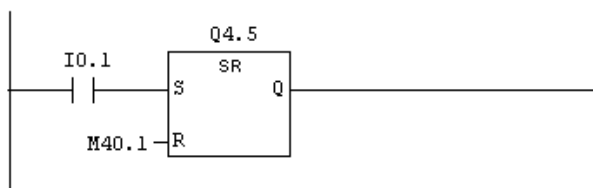


**Network 17 :** Motor door 2 closed

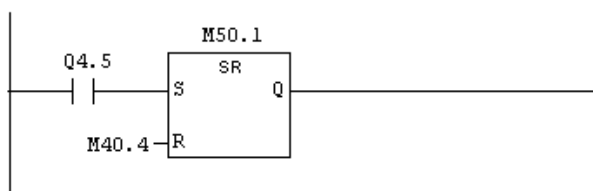


FC5 : Solution according to chap. 12.8 Door access control system

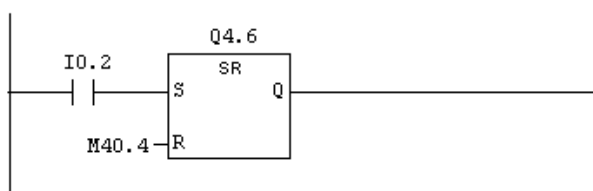
**Network 1:** Button S1



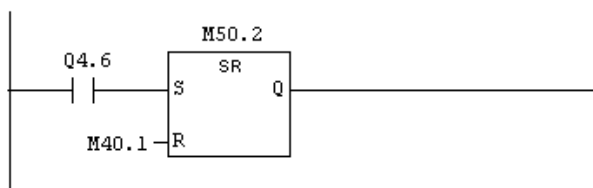
**Network 2:** Button S1



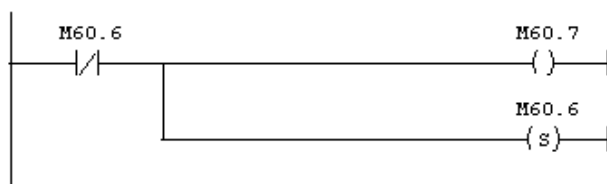
**Network 3:** Button S2



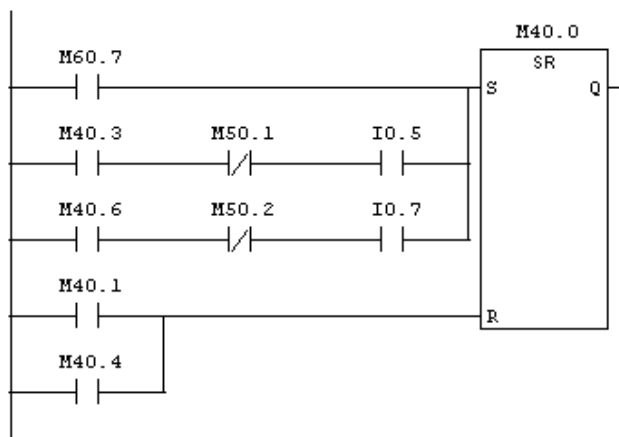
**Network 4:** Button S2



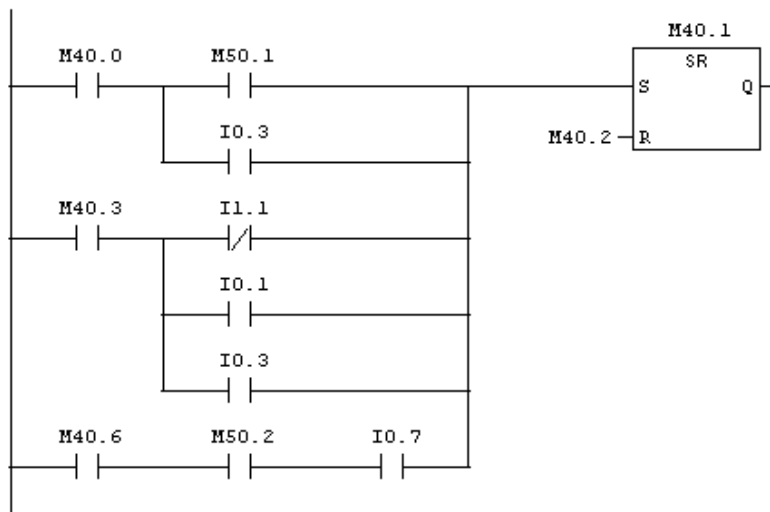
**Network 5 : Switch-on/setting pulse bit memory**



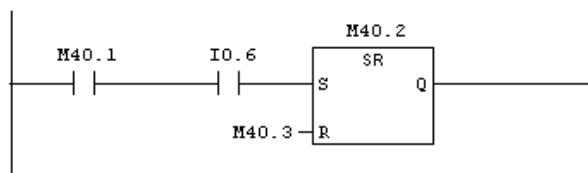
**Network 6 : Status bit memory 0**



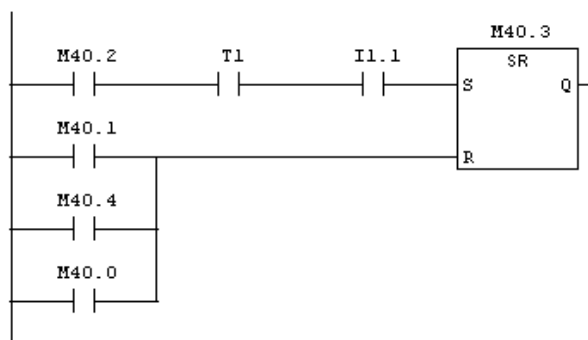
**Network 7 : Status bit memory 1**



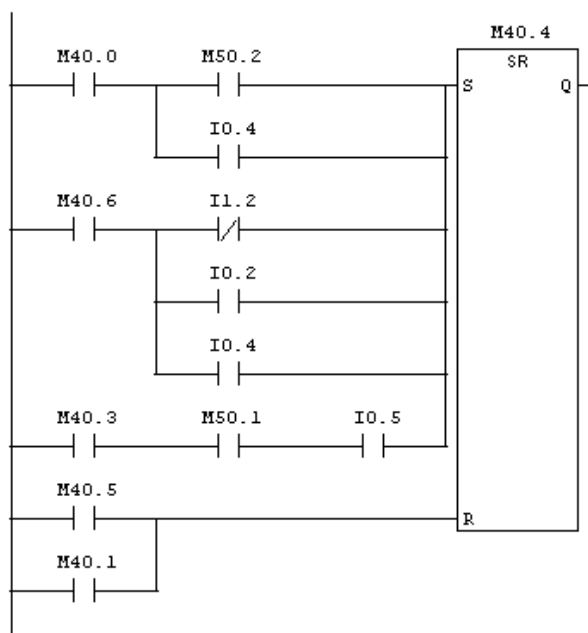
**Network 8 : Status bit memory 2**



**Network 9 : Status bit memory 3**

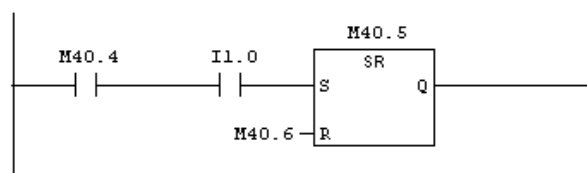


**Network 10 : Status bit memory 4**

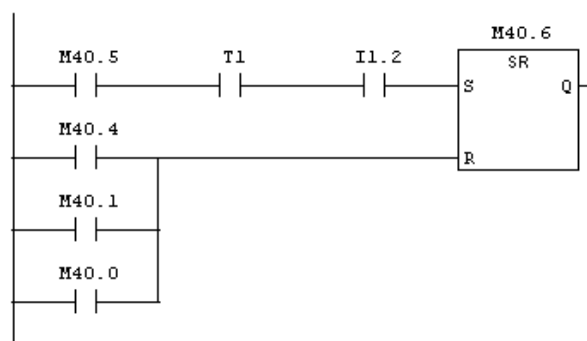




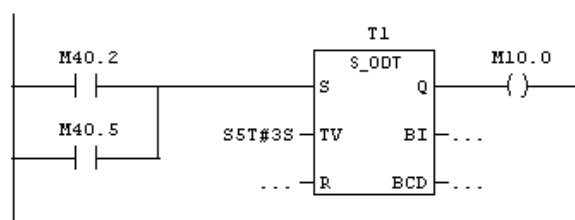
**Network 11:** Status bit memory 5



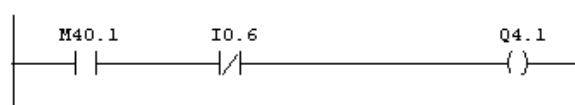
**Network 12:** Status bit memory 6



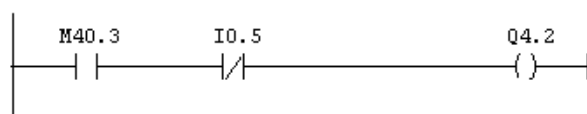
**Network 13:** Wait time



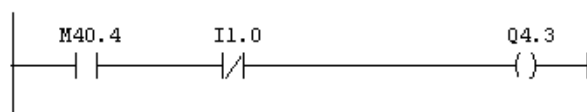
**Network 14:** Motor door 1 open



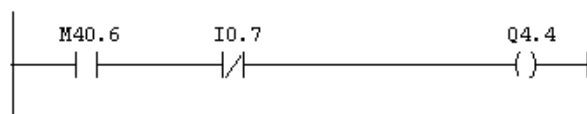
**Network 15:** Motor door 1 closed



**Network 16:** Motor door 2 open



**Network 17:** Motor door 2 closed



FC5 : Solution according to chap. 12.8 Door access control system

**Network 1:** Button S1

A	I	0.1	//Button S1 "Door open"
S	Q	4.5	//Display button S1
A	M	40.1	
R	Q	4.5	
NOP	O		

**Network 2:** Button S1

A	Q	4.5	
S	M	50.1	
A	M	40.4	
R	M	50.1	//Button memory TSP1
NOP	O		

**Network 3:** Button S2

A	I	0.2	//Button S2 "Door open"
S	Q	4.6	//Display button S2
A	M	40.4	
R	Q	4.6	
NOP	O		

**Network 4:** Button S2

A	Q	4.6	
S	M	50.2	
A	M	40.1	
R	M	50.2	//Button memory TSP2
NOP	O		

**Network 5:** Switch-on/setting pulse bit memory

AN	M	60.6
=	M	60.7
S	M	60.6

**Network 6 :** Status bit memory 0

```
A(
O      M      60.7
O
A      M      40.3
AN     M      50.1
A      I      0.5
O
A      M      40.6
AN     M      50.2
A      I      0.7
)
S      M      40.0
A(
O      M      40.1
O      M      40.4
)
R      M      40.0
NOP    O
```

**Network 7 :** Status bit memory 1

```
A(
A      M      40.0
A(
O      M      50.1
O      I      0.3
)
O
A      M      40.3
A(
ON     I      1.1
O      I      0.1
O      I      0.3
)
O
A      M      40.6
A      M      50.2
A      I      0.7
)
S      M      40.1
A      M      40.2
R      M      40.1
NOP    O
```

**Network 8 :** Status bit memory 2

```
A      M      40.1
A      I      0.6
S      M      40.2
A      M      40.3
R      M      40.2
NOP    O
```

**Network 9 : Status bit memory 3**

A	M	40.2
A	T	1
A	I	1.1
S	M	40.3
A(		
O	M	40.1
O	M	40.4
O	M	40.0
)		
R	M	40.3
NOP	O	

**Network 10 : Status bit memory 4**

A(		
A	M	40.0
A(		
O	M	50.2
O	I	0.4
)		
O		
A	M	40.6
A(		
ON	I	1.2
O	I	0.2
O	I	0.4
)		
O		
A	M	40.3
A	M	50.1
A	I	0.5
)		
S	M	40.4
A(		
O	M	40.5
O	M	40.1
)		
R	M	40.4
NOP	O	

**Network 11 : Status bit memory 5**

A	M	40.4
A	I	1.0
S	M	40.5
A	M	40.6
R	M	40.5
NOP	O	

**Network 12 :** Status bit memory 6

```
A      M      40.5
A      T      1
A      I      1.2
S      M      40.6
A(
O      M      40.4
O      M      40.1
O      M      40.0
)
R      M      40.6
NOP    O
```

**Network 13 :** Wait time

```
A(
O      M      40.2
O      M      40.5
)
L      S5T#3S
SD     T      1
NOP    O
NOP    O
NOP    O
A      T      1
=      M      10.0
```

**Network 14 :** Motor door 1 open

```
A      M      40.1
AN     I      0.6
=      Q      4.1
```

**Network 15 :** Motor door 1 closed

```
A      M      40.3
AN     I      0.5
=      Q      4.2
```

**Network 16 :** Motor door 2 open

```
A      M      40.4
AN     I      1.0
=      Q      4.3
```

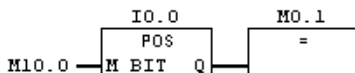
**Network 17 :** Motor door 2 closed

```
A      M      40.6
AN     I      0.7
=      Q      4.4
```

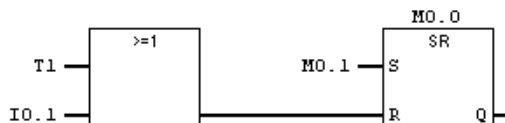
## Example of a solution according to chapter 12.9

FC3 : Solution according to chap. 12.9 Pump controller

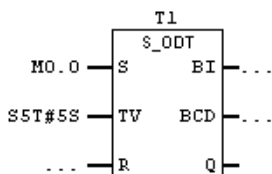
**Network 1:** Pulse for pump On



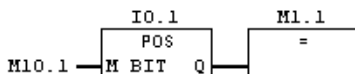
**Network 2:** Pulse for pump On



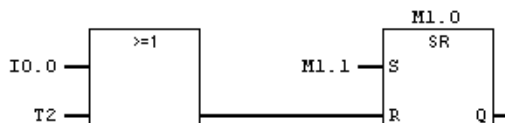
**Network 3:** Bit memory for Start T1



**Network 4:** Pulse for pump Off



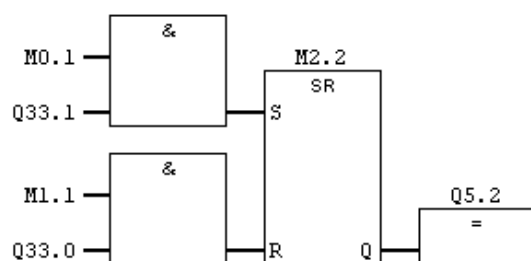
**Network 5:** Pulse for pump Off



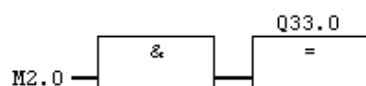




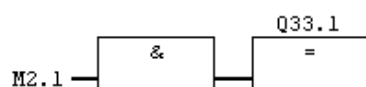
**Network 9 :** Register pumps On



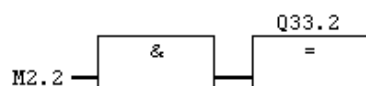
**Network 10 :** Register pumps On



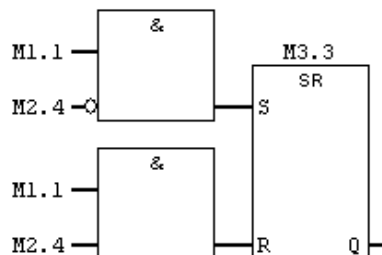
**Network 11 :** Register pumps On



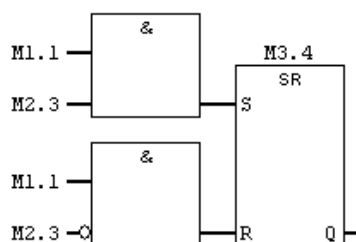
**Network 12 :** Register pumps On



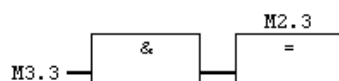
**Network 13 :** Register pumps Off



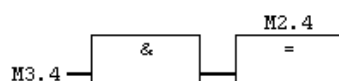
**Network 14 : Register pumps Off**



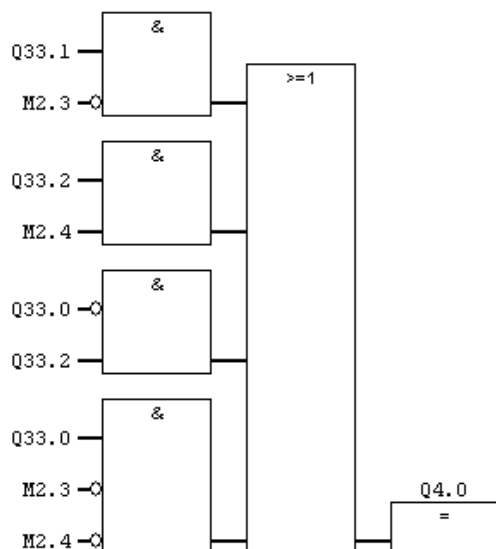
**Network 15 : Register pumps Off**



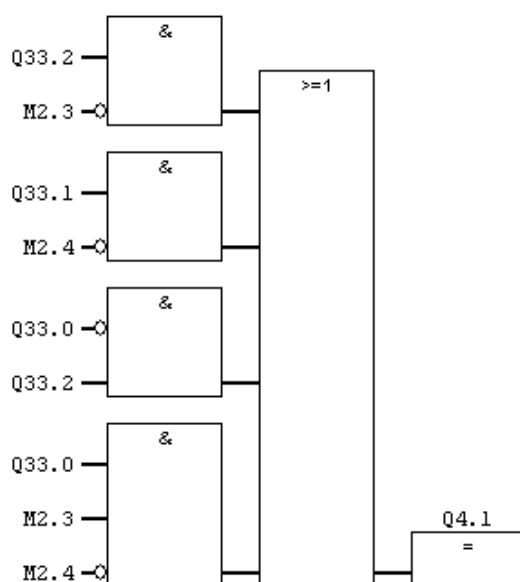
**Network 16 : Register pumps Off**



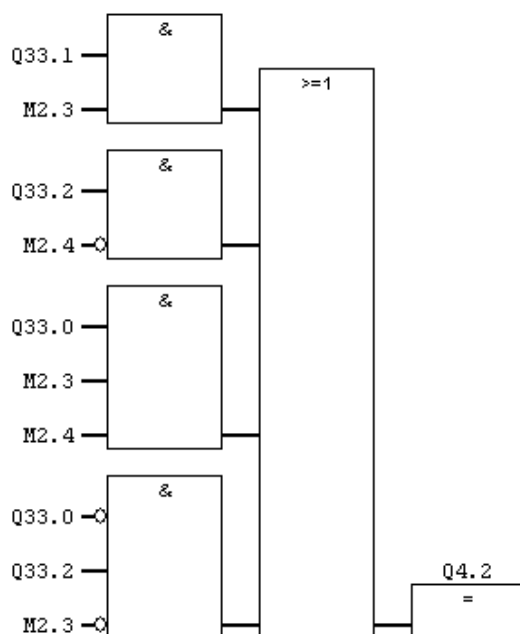
**Network 17 : Pump control**



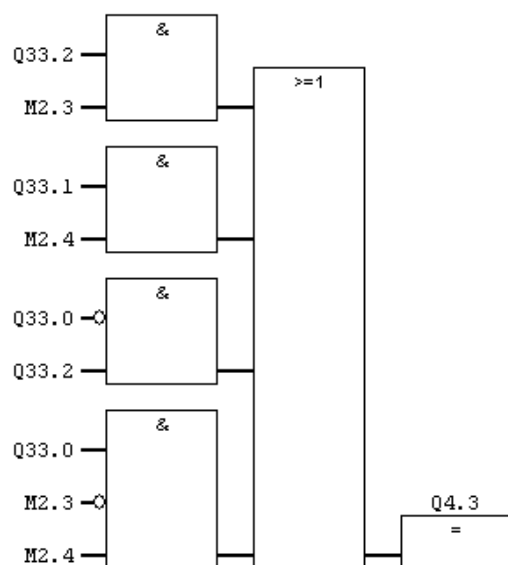
**Network 18 : Pump control**



**Network 19 : Pump control**

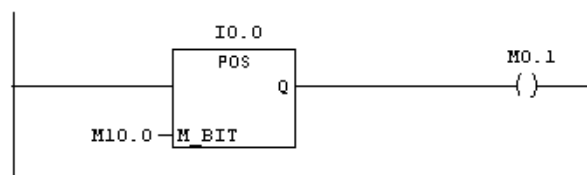


**Network 20 : Pump control**

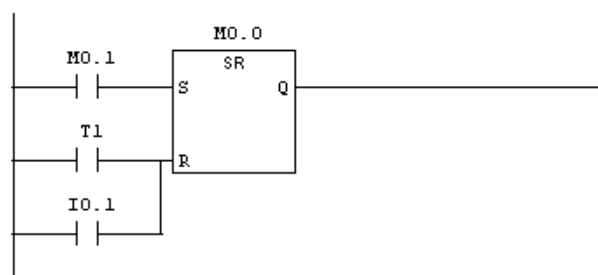


FC3 : Solution according to chap. 12.9 Pump controller

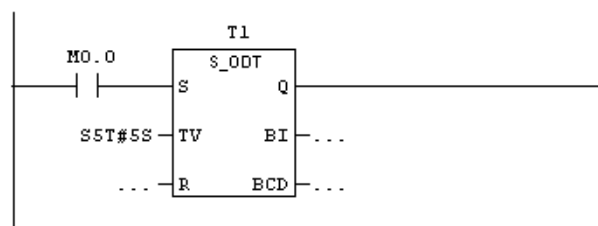
**Network 1:** Pulse for pump On



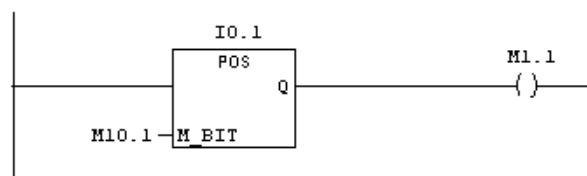
**Network 2:** Pulse for pump On



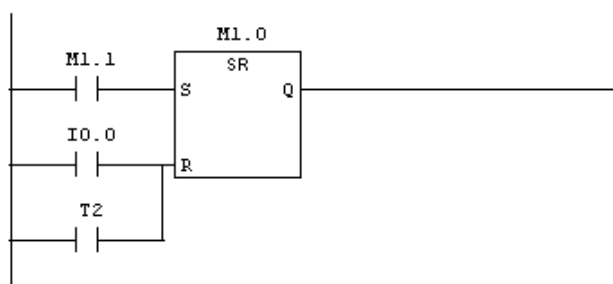
**Network 3:** Bit memory for Start T1



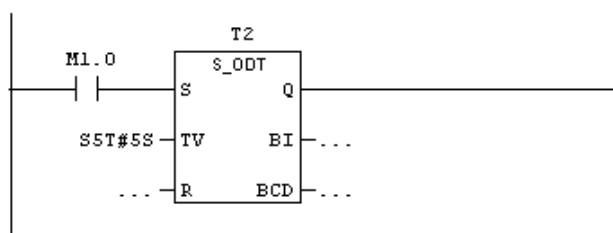
**Network 4:** Pulse for pump Off



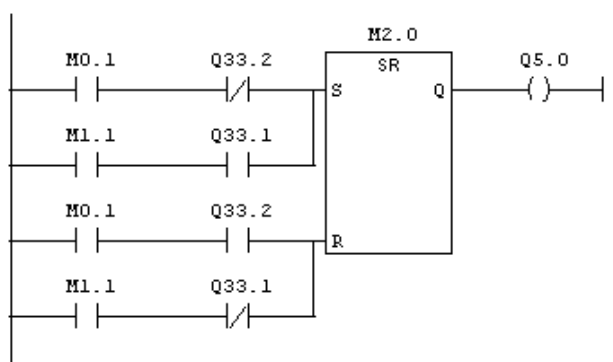
**Network 5 :** Pulse for pump Off



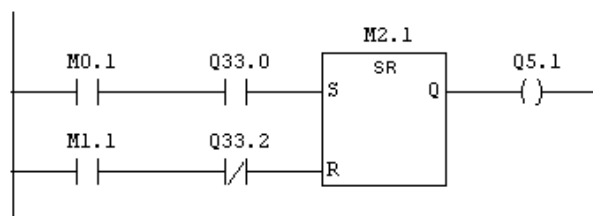
**Network 6 :** Bit memory for Start T2



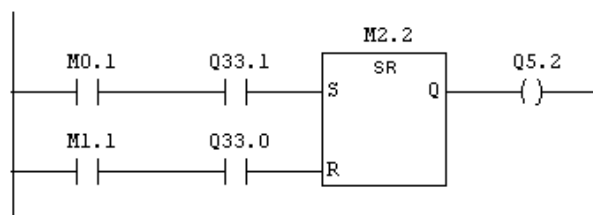
**Network 7 :** Register pumps On



**Network 8 :** Register pumps On



**Network 9 :** Register pumps On



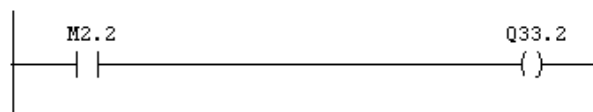
**Network 10 :** Register pumps On



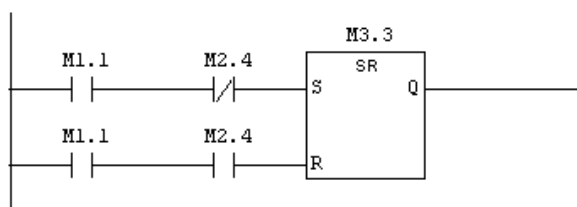
**Network 11 :** Register pumps On



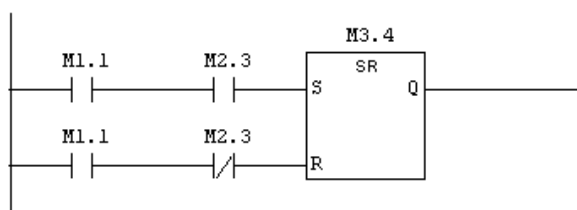
**Network 12 :** Register pumps On



**Network 13 : Register pumps Off**



**Network 14 : Register pumps Off**



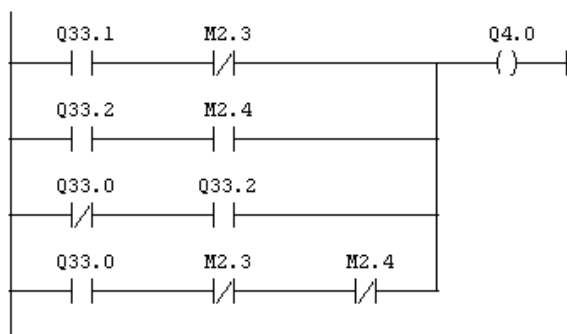
**Network 15 : Register pumps Off**



**Network 16 : Register pumps Off**

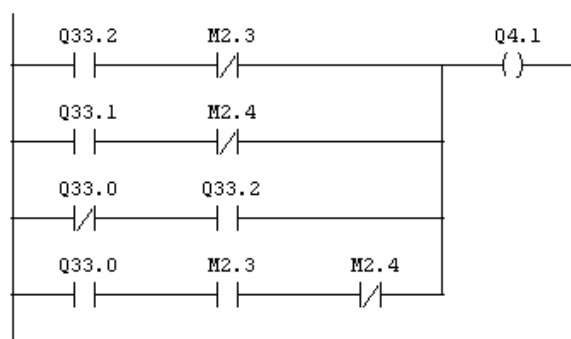


**Network 17 : Pump control**

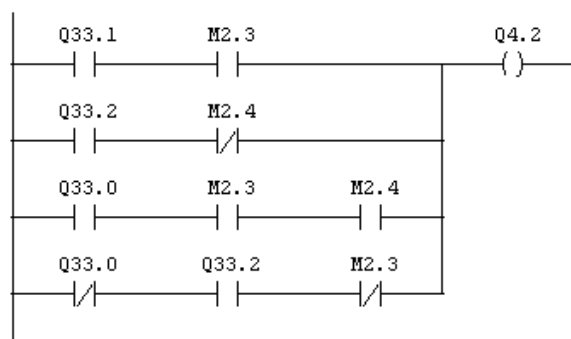




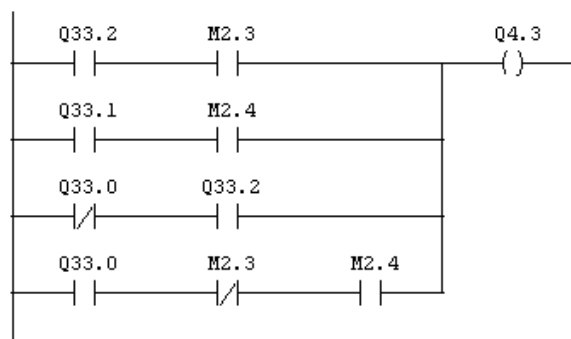
**Network 18 : Pump control**



**Network 19 : Pump control**



**Network 20 : Pump control**



FC3 : Solution according to chap. 12.9 Pump controller

**Network 1:** Pulse for pump On

```
A      I      0.0
BLD    100
FP     M      10.0
=      M      0.1
```

**Network 2:** Pulse for pump On

```
A      M      0.1
S      M      0.0
A(
O      T      1
O      I      0.1
)
R      M      0.0
NOP    0
```

**Network 3:** Bit memory for Start T1

```
A      M      0.0
L      SST#5S
SD     T      1
NOP    0
NOP    0
NOP    0
NOP    0
```

**Network 4:** Pulse for pump Off

```
A      I      0.1
BLD    100
FP     M      10.1
=      M      1.1
```

**Network 5:** Pulse for pump Off

```
A      M      1.1
S      M      1.0
A(
O      I      0.0
O      T      2
)
R      M      1.0
NOP    0
```

**Network 6 :** Bit memory for Start T2

A	M	1.0
L	S5T#5S	
SD	T	2
NOP	O	
NOP	O	
NOP	O	
NOP	O	

**Network 7 :** Register pumps On

A(		
A	M	0.1
AN	Q	33.2
O		
A	M	1.1
A	Q	33.1
)		
S	M	2.0
A(		
A	M	0.1
A	Q	33.2
O		
A	M	1.1
AN	Q	33.1
)		
R	M	2.0
A	M	2.0
=	Q	5.0

**Network 8 :** Register pumps On

A	M	0.1
A	Q	33.0
S	M	2.1
A	M	1.1
AN	Q	33.2
R	M	2.1
A	M	2.1
=	Q	5.1

**Network 9 :** Register pumps On

A	M	0.1
A	Q	33.1
S	M	2.2
A	M	1.1
A	Q	33.0
R	M	2.2
A	M	2.2
=	Q	5.2

**Network 10 :** Register pumps On

A	M	2.0
=	Q	33.0

**Network 11 :** Register pumps On

A	M	2.1
=	Q	33.1

**Network 12 :** Register pumps On

A	M	2.2
=	Q	33.2

**Network 13 :** Register pumps Off

A	M	1.1
AN	M	2.4
S	M	3.3
A	M	1.1
A	M	2.4
R	M	3.3
NOP	O	

**Network 14 :** Register pumps Off

A	M	1.1
A	M	2.3
S	M	3.4
A	M	1.1
AN	M	2.3
R	M	3.4
NOP	O	

**Network 15 :** Register pumps Off

A	M	3.3
=	M	2.3

**Network 16 :** Register pumps Off

A	M	3.4
=	M	2.4

**Network 17 : Pump control**

A	Q	33.1
AN	M	2.3
O		
A	Q	33.2
A	M	2.4
O		
AN	Q	33.0
A	Q	33.2
O		
A	Q	33.0
AN	M	2.3
AN	M	2.4
=	Q	4.0

**Network 18 : Pump control**

A	Q	33.2
AN	M	2.3
O		
A	Q	33.1
AN	M	2.4
O		
AN	Q	33.0
A	Q	33.2
O		
A	Q	33.0
A	M	2.3
AN	M	2.4
=	Q	4.1

**Network 19 : Pump control**

A	Q	33.1
A	M	2.3
O		
A	Q	33.2
AN	M	2.4
O		
A	Q	33.0
A	M	2.3
A	M	2.4
O		
AN	Q	33.0
A	Q	33.2
AN	M	2.3
=	Q	4.2

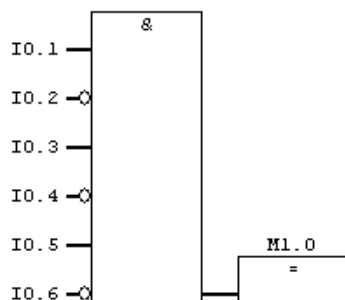
**Network 20 :** Pump control

A	Q	33.2
A	M	2.3
O		
A	Q	33.1
A	M	2.4
O		
AN	Q	33.0
A	Q	33.2
O		
A	Q	33.0
AN	M	2.3
A	M	2.4
=	Q	4.3

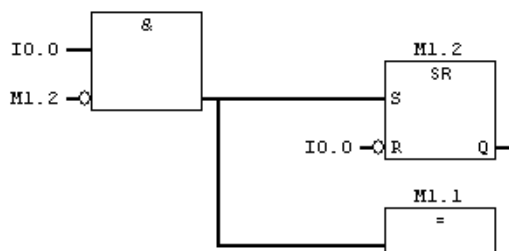
## Example of a solution according to chapter 13.5

FC1 : Solution according to chap. 13.5 Sheet metal bending device

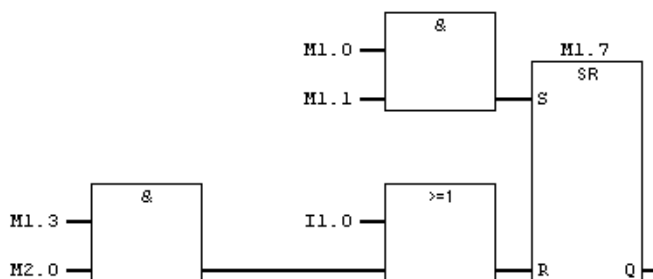
**Network 1:** Basic position of the plant



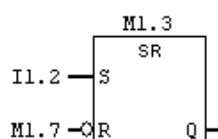
**Network 2:** Bit memory for start/single step



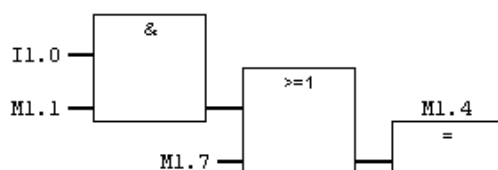
**Network 3:** Automatic operation



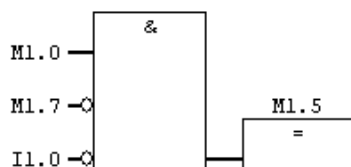
**Network 4 : Stop automatic operation**



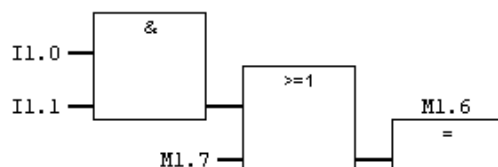
**Network 5 : Release of the sequence cascade**



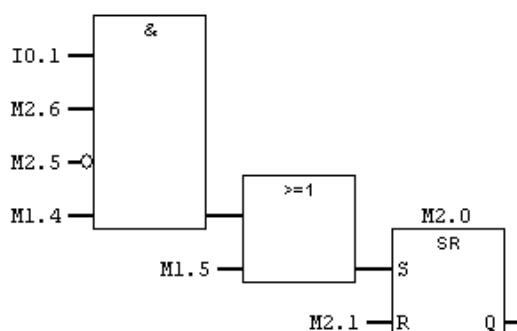
**Network 6 : Sequence cascade in basic position**



**Network 7 : Release of the issue of orders**

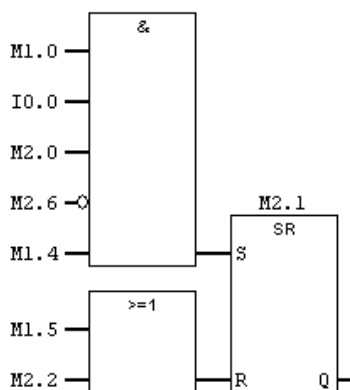


**Network 8 : Sequence cascade - Step 0**

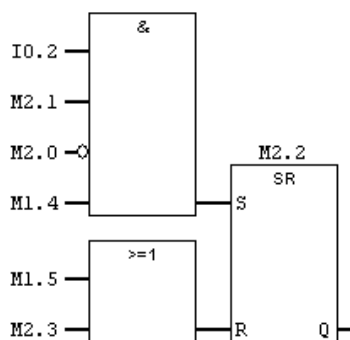




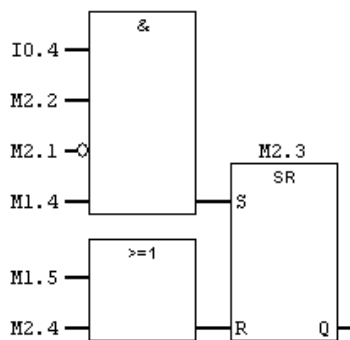
**Network 9 :** Sequence cascade - Step 1



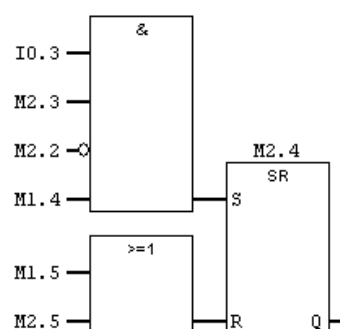
**Network 10 :** Sequence cascade - Step 2



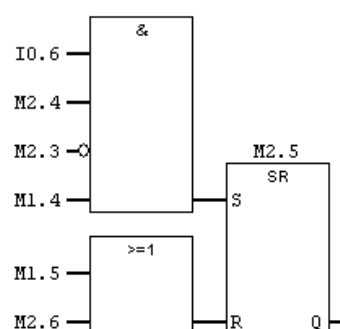
**Network 11 :** Sequence cascade - Step 3



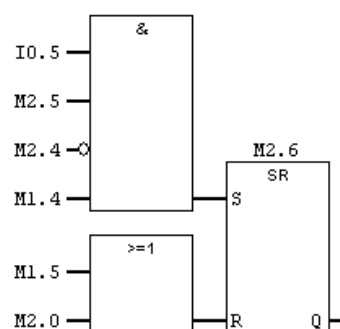
**Network 12 :** Sequence cascade - Step 4



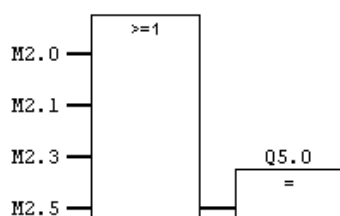
**Network 13 :** Sequence cascade - Step 5



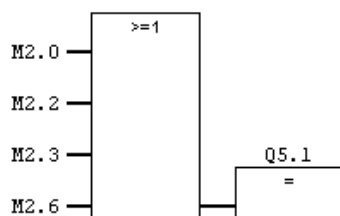
**Network 14 :** Sequence cascade - Step 6



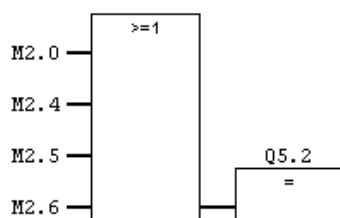
**Network 15 :** Step indicator - Value 1



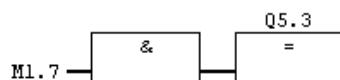
**Network 16 :** Step indicator - Value 2



**Network 17 :** Step indicator - Value 4



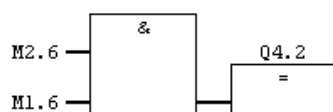
**Network 18 :** Indication of automatic operation



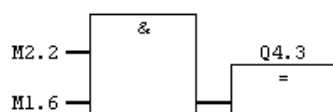
**Network 19 :** Issue of orders



**Network 20 : Issue of orders**



**Network 21 : Issue of orders**



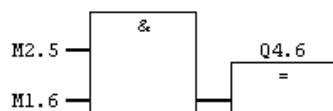
**Network 22 : Issue of orders**



**Network 23 : Issue of orders**

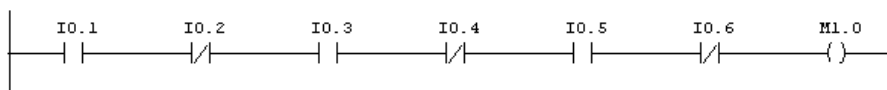


**Network 24 : Issue of orders**



FC1 : Solution according to chap. 13.5 Sheet metal bending device

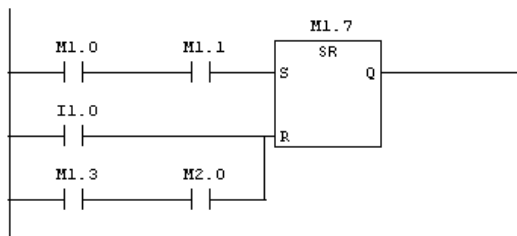
Network 1 : Basic position of the plant



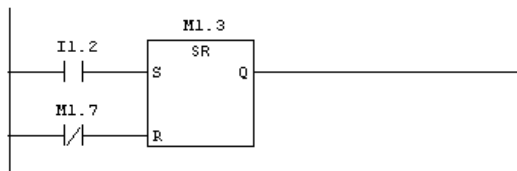
Network 2 : Bit memory for start/single step

A	I	0.0
AN	M	1.2
=	L	0.0
A	L	0.0
S	M	1.2
AN	I	0.0
R	M	1.2
NOP	O	
A	L	0.0
BLD	102	
=	M	1.1

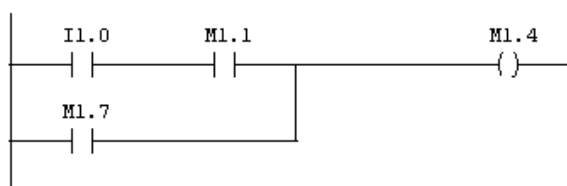
Network 3 : Automatic operation



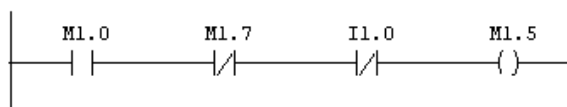
Network 4 : Stop automatic operation



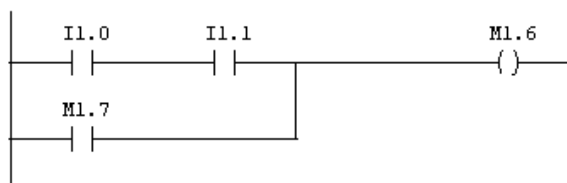
**Network 5:** Release of the sequence cascade



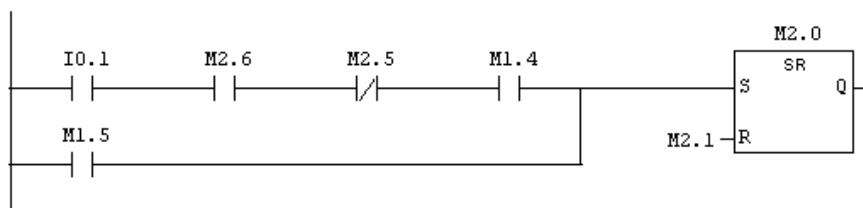
**Network 6:** Sequence cascade in basic position



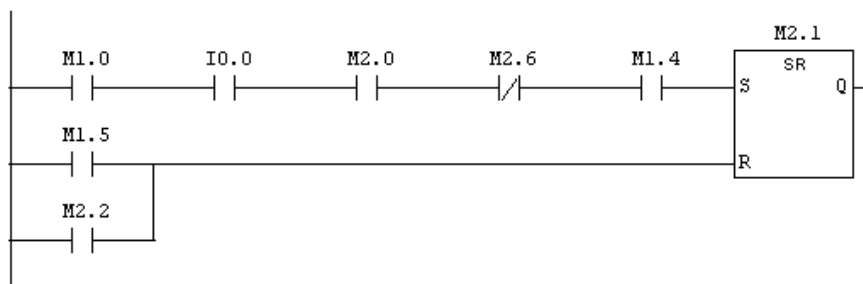
**Network 7:** Release of the issue of orders



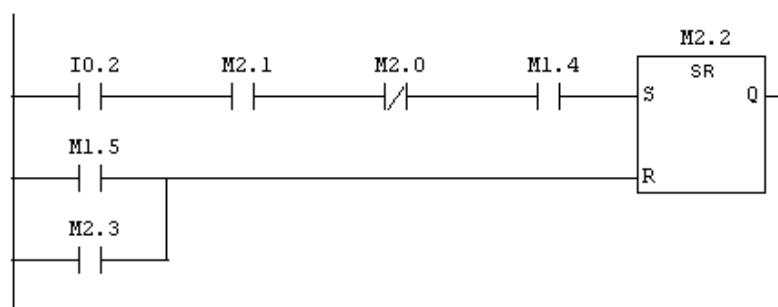
**Network 8:** Sequence cascade - Step 0



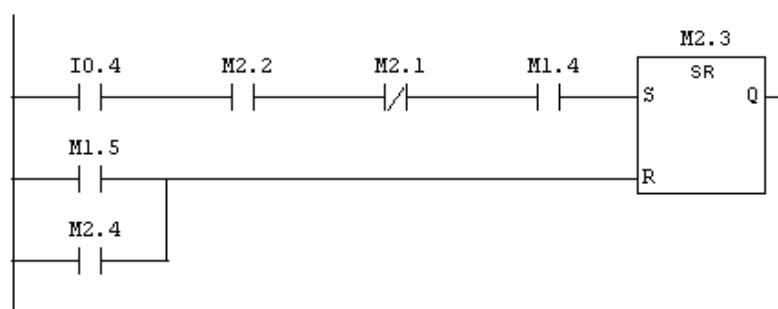
**Network 9:** Sequence cascade - Step 1



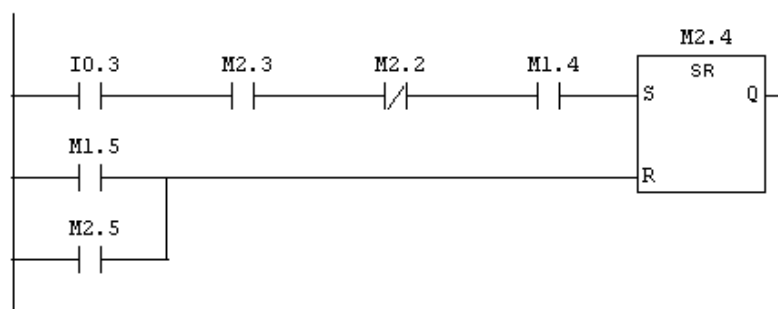
**Network 10 :** Sequence cascade - Step 2



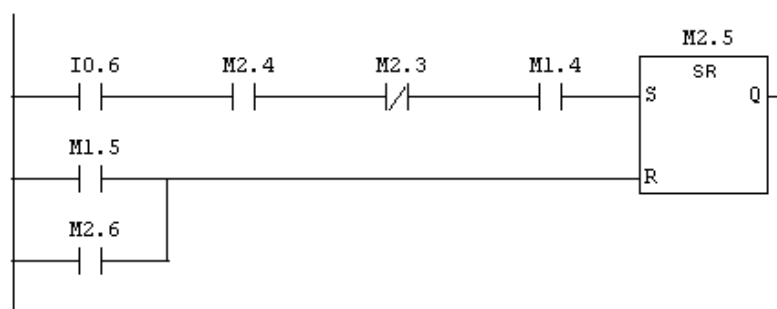
**Network 11 :** Sequence cascade - Step 3



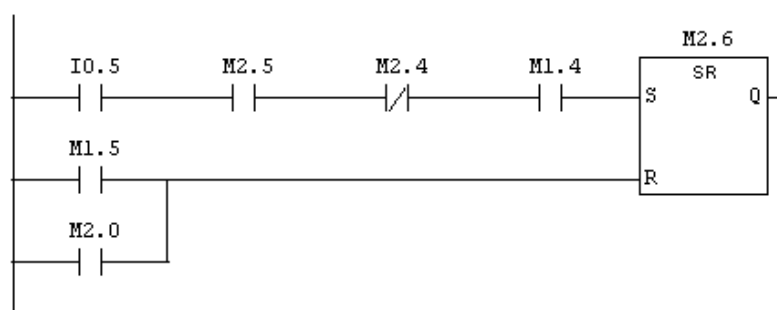
**Network 12 :** Sequence cascade - Step 4



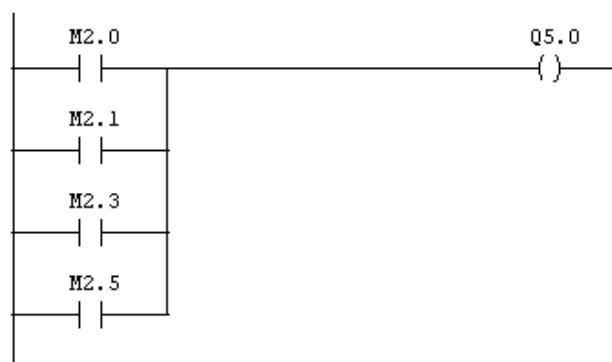
**Network 13 :** Sequence cascade - Step 5



**Network 14 :** Sequence cascade - Step 6

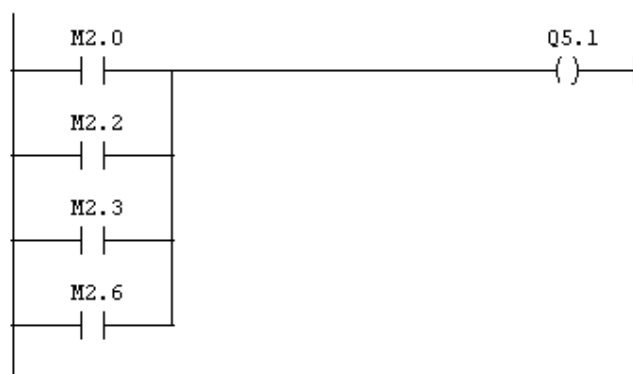


**Network 15 :** Step indicator - Value 1

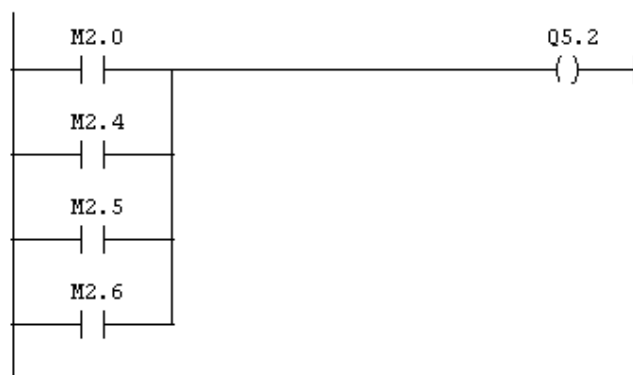




**Network 16 :** Step indicator - Value 2



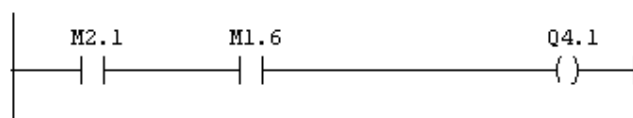
**Network 17 :** Step indicator - Value 4



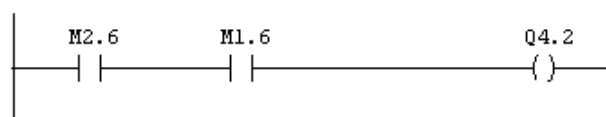
**Network 18 :** Indication of automatic operation



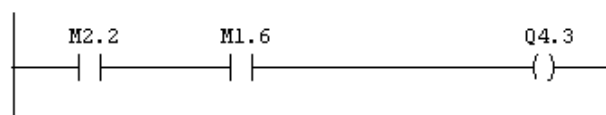
**Network 19 :** Issue of orders



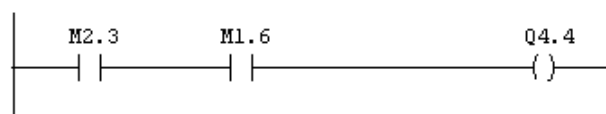
**Network 20 : Issue of orders**



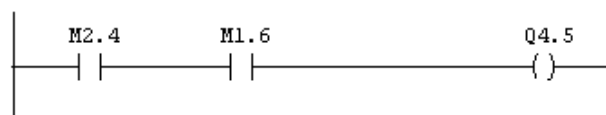
**Network 21 : Issue of orders**



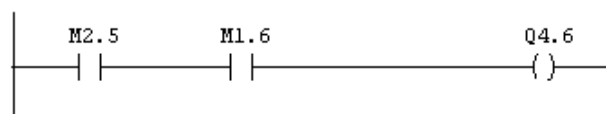
**Network 22 : Issue of orders**



**Network 23 : Issue of orders**



**Network 24 : Issue of orders**



FC1 : Solution according to chap. 13.5 Sheet metal bending device

**Network 1:** Basic position of the plant

A	I	0.1
AN	I	0.2
A	I	0.3
AN	I	0.4
A	I	0.5
AN	I	0.6
=	M	1.0

**Network 2:** Bit memory for start/single step

A	I	0.0
AN	M	1.2
=	L	0.0
A	L	0.0
S	M	1.2
AN	I	0.0
R	M	1.2
NOP	O	
A	L	0.0
BLD	102	
=	M	1.1

**Network 3:** Automatic operation

A	M	1.0
A	M	1.1
S	M	1.7
A{		
O	I	1.0
O		
A	M	1.3
A	M	2.0
}		
R	M	1.7
NOP	O	

**Network 4:** Stop automatic operation

A	I	1.2
S	M	1.3
AN	M	1.7
R	M	1.3
NOP	O	

**Network 5 :** Release of the sequence cascade

A	I	1.0
A	M	1.1
O	M	1.7
=	M	1.4

**Network 6 :** Sequence cascade in basic position

A	M	1.0
AN	M	1.7
AN	I	1.0
=	M	1.5

**Network 7 :** Release of the issue of orders

A	I	1.0
A	I	1.1
O	M	1.7
=	M	1.6

**Network 8 :** Sequence cascade - Step 0

A(		
A	I	0.1
A	M	2.6
AN	M	2.5
A	M	1.4
O	M	1.5
)		
S	M	2.0
A	M	2.1
R	M	2.0
NOP	O	

**Network 9 :** Sequence cascade - Step 1

A	M	1.0
A	I	0.0
A	M	2.0
AN	M	2.6
A	M	1.4
S	M	2.1
A(		
O	M	1.5
O	M	2.2
)		
R	M	2.1
NOP	O	

**Network 10 : Sequence cascade - Step 2**

A	I	0.2
A	M	2.1
AN	M	2.0
A	M	1.4
S	M	2.2
A(		
O	M	1.5
O	M	2.3
)		
R	M	2.2
NOP	0	

**Network 11 : Sequence cascade - Step 3**

A	I	0.4
A	M	2.2
AN	M	2.1
A	M	1.4
S	M	2.3
A(		
O	M	1.5
O	M	2.4
)		
R	M	2.3
NOP	0	

**Network 12 : Sequence cascade - Step 4**

A	I	0.3
A	M	2.3
AN	M	2.2
A	M	1.4
S	M	2.4
A(		
O	M	1.5
O	M	2.5
)		
R	M	2.4
NOP	0	

**Network 13 : Sequence cascade - Step 5**

A	I	0.6
A	M	2.4
AN	M	2.3
A	M	1.4
S	M	2.5
A(		
O	M	1.5
O	M	2.6
)		
R	M	2.5
NOP	0	

**Network 14 :** Sequence cascade - Step 6

A	I	0.5
A	M	2.5
AN	M	2.4
A	M	1.4
S	M	2.6
A{		
0	M	1.5
0	M	2.0
}		
R	M	2.6
NOP	0	

**Network 15 :** Step indicator - Value 1

0	M	2.0
0	M	2.1
0	M	2.3
0	M	2.5
=	Q	5.0

**Network 16 :** Step indicator - Value 2

0	M	2.0
0	M	2.2
0	M	2.3
0	M	2.6
=	Q	5.1

**Network 17 :** Step indicator - Value 4

0	M	2.0
0	M	2.4
0	M	2.5
0	M	2.6
=	Q	5.2

**Network 18 :** Indication of automatic operation

A	M	1.7
=	Q	5.3

**Network 19 :** Issue of orders

A	M	2.1
A	M	1.6
=	Q	4.1

**Network 20 :** Issue of orders

A	M	2.6
A	M	1.6
=	Q	4.2

**Network 21 :** Issue of orders

A	M	2.2
A	M	1.6
=	Q	4.3

**Network 22 :** Issue of orders

A	M	2.3
A	M	1.6
=	Q	4.4

**Network 23 :** Issue of orders

A	M	2.4
A	M	1.6
=	Q	4.5

**Network 24 :** Issue of orders

A	M	2.5
A	M	1.6
=	Q	4.6



**SIEMENS**

E20001-190-0250-X-7600



## Applicable practical know-how

Comprehensive teaching support for educational institutions

Siemens Automation Cooperates with Education

Siemens Automation Cooperates with Education (SCE) focuses on the needs of the students by providing lectures the tools and training to build confidence and applied expertise. The SCE program delivers value to learning institutions with instructor training, learning curriculum and exceptional hardware and software Trainer Packages. Through partnerships we are driven to share knowledge, resources and tools for teaching automation and drive system technologies.

[siemens.com/sce](https://www.siemens.com/sce)



This PLC basic course explains the logic control with the SIMATIC S7-300. For all programming examples from the practice, also a solution is provided. This book is suitable for trade schools, technical colleges and others, as well as for private study. It contains:

- Arrangement and functioning of a PLC
- Program processing and programming
- Logic operations and program input
- Momentary impulses, timing functions, clock generators, counters, comparators
- Practical examples with simulators
- Sequence control systems
- Safety Regulations
- Appendix with solutions

