University of Maribor Press

Jožef Ritonja

AUTOMATION WITH PROGRAMMABLE LOGIC CONTROLLERS

TEXBOOK

University of Maribor

Faculty of Electrical Engineering
and Computer Science

AUTOMATION WITH PROGRAMMABLE LOGIC CONTROLLERS

Textbook

Avtor
Jožef Ritonja

June 2026

Title **Automation With Programmable Logic Controllers**

Subtitle **Textbook**

Author Jožef Ritonja
(University of Maribor, Faculty of Electrical Engineering and Computer Science)

Review Bojan Grčar
(University of Maribor)

Jan Bilek
(Czech Technical University Prague)

Darius Andriukaitis
(Kaunas University of Technology)

Translation Jožef Ritonja
(University of Maribor, Faculty of Electrical Engineering and Computer Science)

Language editing Shelagh Hedges

Technical editors Jožef Ritonja
(University of Maribor, Faculty of Electrical Engineering and Computer Science)

Jan Perša
(University of Maribor, University of Maribor Press)

Cover designer Jan Perša
(University of Maribor, University of Maribor Press)

Published by **University of Maribor**
Založnik **University of Maribor Press**
Slomškov trg 15, 2000 Maribor, Slovenia
https://press.um.si, zalozba@um.si

Issued by **University of Maribor**
Izdajatelj **Faculty of Electrical Engineering and Computer Science**
Koroška cesta 46, 2000 Maribor, Slovenia
https://www.feri.um.si, feri@um.si

Edition First edition

Published at Maribor, Slovenia, June 2026

Publication type E-book

Available at https://press.um.si/index.php/ump/catalog/book/1133

CIP - Kataložni zapis o publikaciji
Univerzitetna knjižnica Maribor

681.5(0.034.2)

RITONJA, Jožef
Automation with programmable logic controllers [Elektronski vir] : textbook / Jožef Ritonja. - 1st ed. - E-publikacija. - Maribor : University of Maribor, University of Maribor Press, 2026

Način dostopa (URL): https://press.um.si/index.php/ump/catalog/book/1133

ISBN 978-961-299-159-3 (PDF)
doi: 10.18690/um.feri.6.2026
COBISS.SI-ID 281714691

ISBN 978-961-299-159-3 (pdf)

DOI https://doi.org/10.18690/um.feri.6.2026

Price Free copy

For publisher Prof. dr. Zdravko Kačič,
rektor Unvierze v Mariboru

Attribution Ritonja, J. (2026). *Automation With Programmable Logic Controllers: Textbook.* University of Maribor Press. doi: 10.18690/um.feri.6.2026

Contents

Preface

Automatic control is one of the fundamental pillars of modern society. Its principles are applied almost everywhere — in everyday life, in infrastructure, and, most notably, in industrial and energy environments where it serves as the foundation for automation of production processes. These control systems can be realized in various ways, but one of the most widely used and flexible methods is through the application of programmable logic controllers (PLCs).

PLCs are special-purpose computers designed to control industrial processes. Unlike personal computers, which are designed to communicate with humans, PLCs interact with machines and technical systems. Both types of computers rely on hardware and software, but they differ primarily in the nature of their interfaces. While a PC requires a keyboard, mouse, screen, and speakers to communicate with people, PLCs use specialized digital and analog I/O interfaces to connect with sensors and actuators, enabling real-time interaction with physical processes.

The development of PLCs was not immediate. Industrial automation began with electromechanical relays, continued with discrete electronic components such as transistors, and progressed through integrated circuits and purpose-built control computers. A major milestone occurred in 1968 when the General Motors Hydra-MaticTransmission Plant issued a request for a flexible control system. The contract was awarded to Bedford Associates of Massachusetts, whose project was internally referred to as “Project 84.” Recognizing the vast potential of their solution, the team founded MODICON (MOdular DIgital CONtrollers), which became one of the first manufacturers of PLCs. MODICON was later acquired by Gould Electronics and eventually by Schneider Electric, the current owner of the pioneering brand. Around the same time, the company Allen-Bradley (now part of Rockwell Automation) also made significant contributions to the field, including coining the now-universal term “PLC.”

When designing this textbook, I carefully considered how best to teach the fundamentals and applications of PLCs. A central dilemma was whether to approach the topic generically or to focus on a specific brand and platform from the outset. Through practical teaching experience, I concluded that students learn more effectively when they begin working with concrete hardware and software tools right away. In the field of PLCs, as with personal computing, there are many manufacturers, but in Central Europe — especially in our region — Siemens has emerged as the dominant player. For this reason, the textbook focuses on Siemens PLCs, providing both theoretical background and practical experience based on their platforms.

The evolution of industrial automation technology follows a somewhat different path from that of consumer electronics. In industry, frequent changes to equipment are not viable due to costs and long implementation cycles. As a result, new generations of hardware are introduced more slowly and with long-term support. Siemens began with relay-based logic controllers as early as 1958, introducing the brand SIMATIC (unofficially said to stand for "SIemens + Automatic") and the SIMATIC G series. The first true memory-programmable controllers — what we now call PLCs — appeared in 1973 with the SIMATIC S3 series. Their real breakthrough, however,

came with the SIMATIC S5, launched in 1979, which saw widespread adoption — particularly the versatile S5-95U.

In 1995, Siemens introduced three new series tailored to different complexity levels: S7-200 for basic tasks, S7-300 for medium complexity, and S7-400 for the most demanding applications. The S7-300 series quickly became the most commonly used PLCs in industrial and energy sectors across our region, due to their balance of performance, versatility, and cost. Instead of a next-generation "S9" series, Siemens opted to continue evolving the existing product lines. The S7-200 was replaced by the S7-1200, while the S7-300 and S7-400 gave way to the high-performance S7-1500 series. These newer controllers are supported by a new software environment, TIA Portal, replacing the legacy STEP 7 (SIMATIC Manager). While the S7-1200 and S7-1500 are set to dominate future applications, many industrial systems still rely on the tried-and-tested S7-300 series.

After careful consideration, and due to its widespread use in practice, I chose to base the practical sections of this textbook primarily on the SIMATIC S7-300 PLCs. I believe this approach provides a solid foundation for understanding automation concepts while equipping students with directly applicable skills for real-world industrial environments.

The textbook aligns well with the curriculum of technical tertiary-level education programs in the fields of automation, mechatronics, and industrial control systems. It can serve as a primary course material for subjects focusing on PLC programming and industrial process automation, and it also has value for self-study or as a reference for practicing engineers. The textbook is both timely and relevant. It addresses current technologies and methodologies used in industrial automation and PLC programming. While many books exist on similar topics, this publication distinguishes itself by focusing specifically on Siemens systems and by integrating real industrial examples that are often absent in more theoretical texts.

Compared to other educational resources on automation and PLCs, this book stands out for its practical orientation and detailed Siemens-specific content. In the context of the Slovenian or Central European educational landscape, such a textbook is rare and fills a notable gap in locally relevant English-language instructional material.
The content is extensive and comprehensive. The textbook is composed of 17 chapters, organized into 7 main parts. The first part introduces the fundamentals of automation, covering the history of automation, examples of automated systems, classification of automation systems, and an overview of the market.
The second part provides a detailed overview of Siemens hardware and software, as understanding the available equipment is essential for designing cost-effective and performance-optimized control systems.

The third part presents the basics of PLC programming, with a special emphasis on STL programming in the fourth part.
Parts five and six are more specialized, describing the installation of programming tools used to write PLC programs, as well as examples of creating project files for different hardware configurations.
Finally, the seventh part illustrates two practical examples of PLC applications: automation of a drilling machine and control of a robotic manipulator.

In this textbook, I have chosen to focus specifically on programming in the Statement List (STL) language. This decision was driven by several factors. STL is the most fundamental and universal of the available PLC programming languages. It offers a low-level, assembler-like view of how the controller processes instructions, making it particularly valuable for understanding the inner workings of PLC logic. Mastering STL provides learners with a solid foundation that enhances their ability to later work effectively with graphical languages like Ladder Diagram (LAD) and Function Block Diagram (FBD). Moreover, STL remains highly relevant in industrial environments where compact, efficient, and easily optimized code is required — especially for time-critical applications or in legacy systems that still dominate many plants.

By learning STL first, students not only gain insight into the logic structure of control systems but also acquire a skill set that is transferable across various Siemens platforms. This approach ensures they are well-prepared both for future learning and for solving practical problems in real-world industrial settings.

Part I – Introduction to the automation

1. Control systems, open loop control, closed loop control and automation

A control system is an interconnection of components forming a system configuration which will provide a desired system response [1]. The basis for analysis of a system is the foundation provided by linear system theory, which assumes a cause-effect relationship for the components of a system. Therefore, a controlled plant or process to be controlled can be represented as a block, as shown in Figure 1.1.

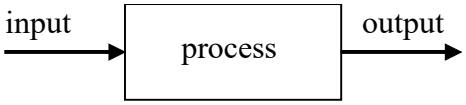

Figure 1.1: Block representation of a process with input and output variables

With regard to usage, an additional measure of the actual output, we divide control systems into open loop control systems and closed loop control systems.
An open-loop control system utilizes an actuating device to control the process directly without using feedback.

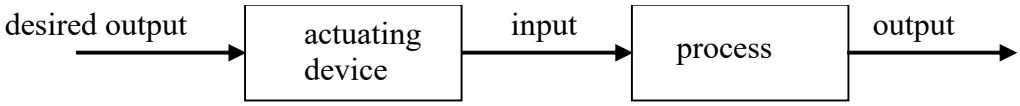

Figure 1.2: Open-loop control system

A closed-loop control system uses a measurement of the output and feedback of this signal to compare it with the desired input (reference or command).

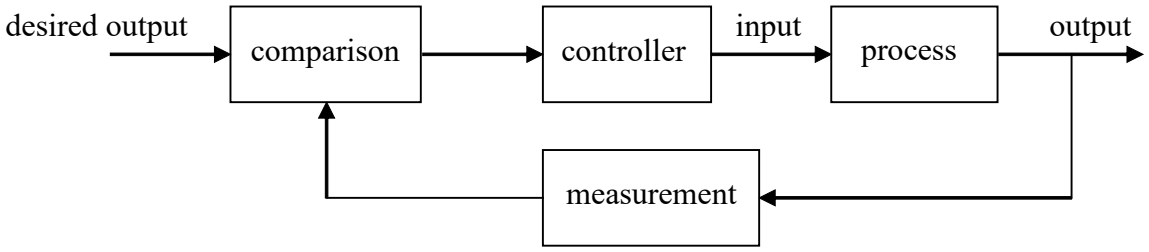

Figure 1.3: Closed-loop control systems

The control of an industrial process (manufacturing, production) by automatic rather than manual means is often called automation. Automation is prevalent in the chemical, electric, power, paper, automobile, and steel industries. The concept of automation is central to our industrial society. Automatic machines are used to increase the production of a plant per worker in order to offset rising wages and inflationary costs.

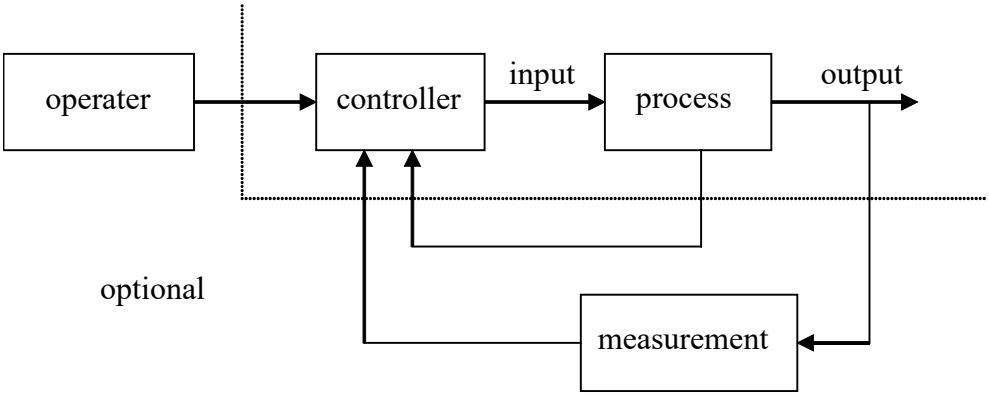

Figure 1.4: Manual and automatic process control

In its modern usage, automation can be defined as a technology that uses programmed commands to operate a given process, combined with feedback of information to determine that the commands have been executed properly [1]. Automation is often used for processes that were previously operated by humans. When automated, the process can operate without human assistance or interference.

2. Some examples of automation

The applications that a programmable logic controller can control are virtually limitless.
Some domestic and commercial applications:

- lighting in apartments, stairwells, shop windows,
- lluorescent lighting circuits,
- blinds, sunshades,
- heating and ventilation systems (HVAC),
- garage doors,
- alarm and bell systems,
- whirlpools,
- solariums,
- irrigation and sprinkler systems,
- ...

Some industrial applications:

- elevating platforms, lifts,
- cable-welding machines,
- etching and purifying plants,
- industrial saws, planers, bending machines, and cutters,
- cutting machines,
- strapping machines,
- boilers,
- heating, cooling, ventilation and air-conditioning systems,
- silo work,
- conveyor systems,
- fractional-horsepower motors, pumps, and valves,
- compressors,

- energy management systems,
- load transfer switches,
- access control/parking lot monitor systems,
- barrier and gate systems,
- ...

3. History and development of automation systems

Automation`s state of the art is the consequence of the evaluation in logic algebra theory, development in computer technology and inventions in control systems [1]. Therefore, the history of automation is connected closely with:

- the history of logic algebra,
- the history of computers, and
- the history of control systems.

3.1 The history of logic algebra

1614	John Napier	announced logarithmic tables, which simplified calculations with multiplication and divisions
1679	G. W. v. Leibnitz	made fundamentals of binary arithmetic – this arithmetic uses a counting system with basis 2
	de Morgan, Buye	extended Leibnitz's work to the theoretical logic – the subjects of this mathematical discipline are not numbers, geometrical objects etc., but methods
1815-1864	G. Boole	formulated logic algebra (or Bool's algebra), this is a part of theoretical logic
	Shannon	showed the connection between binary systems and technique and nature

3.2 The history of computers

3000 B. C.	China, Mesopotamia	abacus
1620	W. Oughtred	made the first logarithmic calculating machine (slide-rule, slipstick)
1642	Blaise Pascal	made a mechanical calculating machine for addition and subtraction
1674	G. W. v. Leibnitz	constructed (realized) a calculating machine which can also multiply and divide, but it was not robust enough for wide use
1834	Charles Babbage	designed a mechanical analytical calculating machine which was too complicated to be realized. This machine already stored results. The terms control unit, store, calculating independent from the nature of the problem were introduced with this machine
1860–1929	Herman Hollerith	sorting machine, tabulating machine company (today IBM)
1912-1954	Alan Turing	described an abstract computer (Turing's machine); this is the basis for theoretical computing

1910-1996	Konrad Zuse	made the first computers Z1, Z2, Z3 and Z4. Z3 was built in the year 1941 from 2400 relays, which enabled floating point operations, adding in 4seconds and 5 seconds for multiplication
1944	Howard Aiken	built in the year 1944 a mechanical electrical machine, 6 seconds for multiplication, calculated logarithmic and trigonometric functions
1903-1950	John von Neuman	American mathematician born in Hungary, described a scheme of a modern computer; the most important was the idea of the program stored together with data in the computer
1945	P. Eckert, J.Maunhly	made an electronic numerical integrator and calculating machine (ENIAC). This was a very powerful calculating machine but not a computer, because it did not store the results and data
1946	EDVAC	Electronic Discrete Variable Automatic Computer finished in the year 1952. It was the first computer with a built-in program, made up of a control unit, decoder, memory ad computer clock; the computer worked till 1961
1947	Bell's laboratory	J. Bardeen, W. Brattain and W. Shockley invented the transistor, a solid-state element similar to a triode, but much smaller – the computers became smaller
1948		Manchester Mark I (GB)
1951		Ferranti Mark I (GB), the first computer for commercial use
1958	Texas Instruments	integrated circuit
1971	T. Hoff	first microprocessor Intel 4004
1971	Intel Corp.	gave on the market the first handy calculating machines
1975	Altair (USA)	the first home computer, which was significantly smaller and cheaper than the previous computers

3.3 The history of control systems

300-1 B.C.	Greeks	a float regulator mechanism, used for the water clock of Ktesibios, an oil lamp by Philon
1572-1633	Cornelis Drebbel	a temperature regulator
1647-1712	Dennis Papin	a pressure regulator for steam boilers
1769	James Wattt	a flyball governor, the first automatic feedback controller used in an industrial process, designed for controlling the speed of the steam machine
1765	I. Polzunov	a water-level float regulator, the first historical feedback system, claimed by Russia
1868	J. C. Maxwell	he formulated a mathematical model for the governor control of a steam engine, and published the stability criterion for a third-order system based on the coefficients of the differential equation

1874	Henry Bessemer	stabilized a ship's saloon using a gyro and the ship's hydraulic system
1874	E. J. Routh	using a suggestion from William Kingdon Clifford that was ignored earlier by Maxwell, he was able to extend the stability criterion to a fifth-order system
1877		Routh submitted a paper entitled ¨A treatise on the stability of a given state of motion¨, won the Adams Prize which topic was ¨The criterion of dynamical stability¨, - this is the Routh-Hurwitz criterion
1885	Nicholas Minorsky	his theoretical development applied to the automatic steering of ships led to what we call today proportional - plus –integral – plus – derivative (PID), or three modes controller
1892	A. M. Lyapunov	extended the work of Routh to nonlinear systems; his doctoral thesis was entitled ¨The general problem of stability of motion¨
1927	H. W. Bode	analyzed feedback amplifiers – sinusoidal frequency analysis
1932	H. Nyquist	he developed a method for analyzing the stability of systems
1948	Evans	developed the graphic technique to plot the root of a characteristic equation of a feedback system whose parameters changed over the particular range of values;
1980		a robust control system design studied widely

3.4 Some important events in the history of automated production

1800	Eli Whitney	his concept of interchangeable parts` manufacturing demonstrated in the production of muskets. Whitney's development is often considered as the beginning of mass production
1913	Henry Ford	his mechanized assembly machine was introduced for automobile production
1952	MIT	numerical control (NC) developed at Massachusetts Institute of Technology for control of machine-tool axes
1954	George Devol	developed a ¨programmed article transfer¨, considered to be the first industrial robot design
1960		introduced the first Unimate robot, based on Devol's designs. Unimate installed it in 1961 for tending die-casting machines
1970		developed state-variable models and optimal control
1990		export-oriented manufacturing companies emphasized automation
1994		feedback control used widely in automobiles. Reliable, robust systems demanded in manufacturing.

4. Classification of automation systems

Automation systems can be classified in different ways. Broadly accepted are two divisions [2]:

- division of the automation system with regard to technical realization, and
- division of the automation system with regard to logical design.

4.1 Division of automation systems with regard to technical realization

Originally, automation systems were realized using mainly either electrical relays, pneumatic or hydraulic valves, or mechanical devices to implement the circuit. Today, the programmable logic controllers are used commonly. In our lectures we will focus our work on the automation systems using programmable logic controllers. The Figure 4.1 shows the division of electrical automation systems with regard to technical realization.

4.2 Division of automation systems with regard to logical design

The Figure 4.1 shows the division of binary control systems with regard to the logical design.

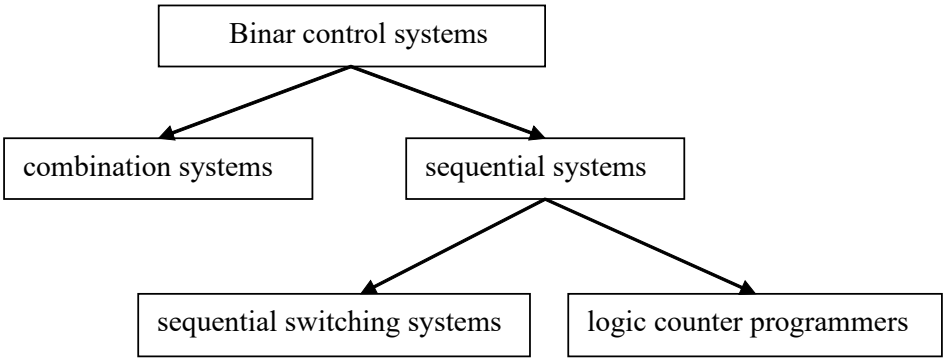

Figure 4.1: Classification of binary control systems

Because of their many advantages, logic counter programmers have come into wide use. Although we will also mention other type of control systems, we will stress the application of programmable logic controllers for logic counter programmers.

5. Controllers made of electronic logic gates – simple examples

In past times electronic logic gates were used for industrial automation to actuate various devices, such as cylinders, pumps, motors, timers, heaters. By means of combination logic elements (AND gate, OR gate, inverter, NOT gate, NOR gate) and memorial elements (set-reset flip flop), it was possible to realize different automation systems. The next examples show the use of electronic logic gates.

5.1 Simple combination logic system

The next scheme shows an electronic circuit for the addition of three 1-bit binary signals. X_1, X_2 and X_3 denote three binary input signals, S (sum) and C (carry) are binary output signals. The Truth Table shows the addition rules.

Table 5.1: Truth table of a three-input (X_1, X_2, X_3) two-output (C, S) logic system

X_1	X_2	X_3	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

The electronic circuit which executes this Truth table 5.1 is shown in Figure 5.1.

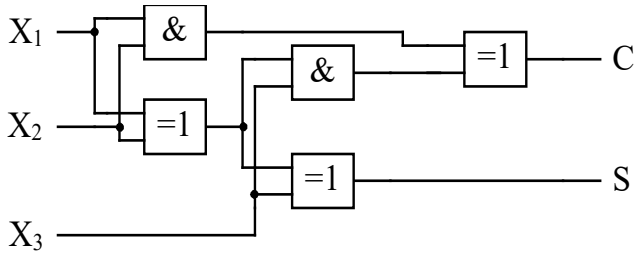

Figure 5.1: Logic circuit of a three-input (X_1, X_2, X_3) two-output (C, S) system

5.2 Simple sequential logic system

We try to develop a control system which will prevent the simultaneous switch on of two motors. Our control system has 3 binary inputs (switch on for motor 1, switch on for motor 2 and switch off for both motors), and 2 binary outputs (motor 1 and motor 2). The timing diagram of the control system signals is shown in Figure 5.2 and state transition diagram in Fig. 5.3.

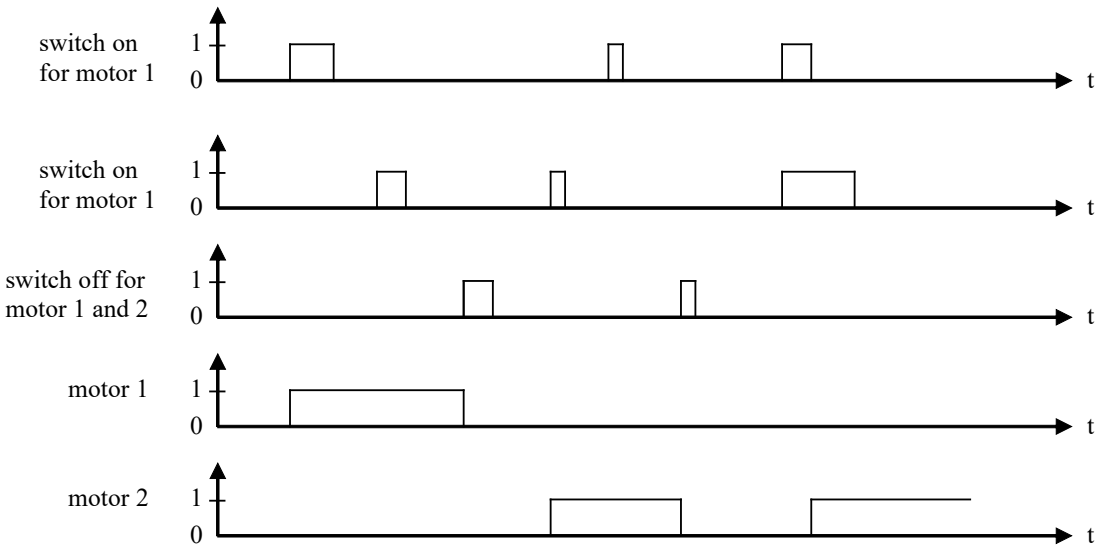

Figure 5.2: Timing diagram of control system signals preventing simultaneous motor operation

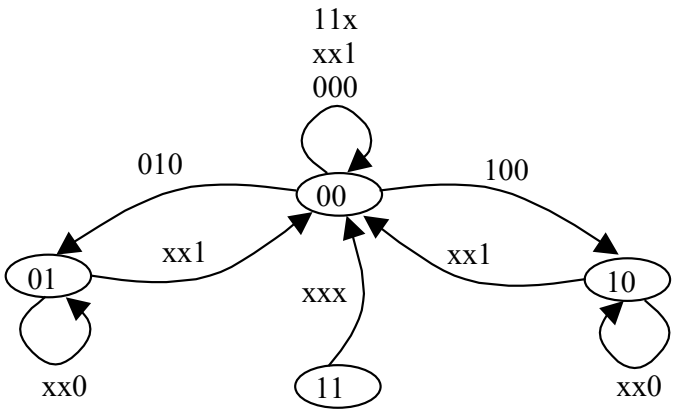

Figure 5.3: State transition diagram of a control system for preventing simultaneous motor operation

The corresponding logic circuit is shown in Figure 5.4. A denotes the signal for turning on motor 1, and B denotes the signal for the turning on motor 2.

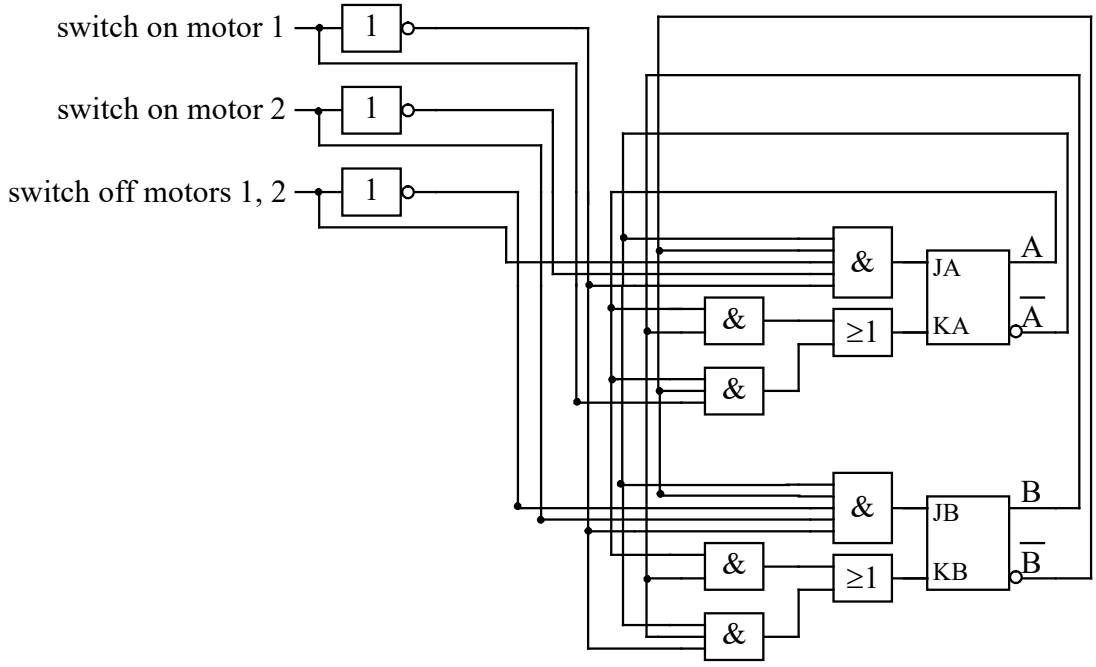

Figure 5.4: Logic circuit of a control system for preventing simultaneous motor operation

5.3 Simple closed loop system

With electronic logic gates we can also realize a simple closed loop control system. This example shows the control system which maintains the temperature of the controlled plant in an adequate range. The temperature of the controlled plant must be higher than the prescribed minimum temperature and lower than the prescribed maximum temperature. From the temperature sensors the control system receives two binary signals: one signal which indicates that the temperature is over (1) or under (0) the maximum temperature value, and one signal which indicates that the temperature is over (1) or under (0) the minimum temperature value. The controller output is a binary signal, which switches the heater for the controlled plant on or off.
A control system timing diagram is presented in Figure 5.5.

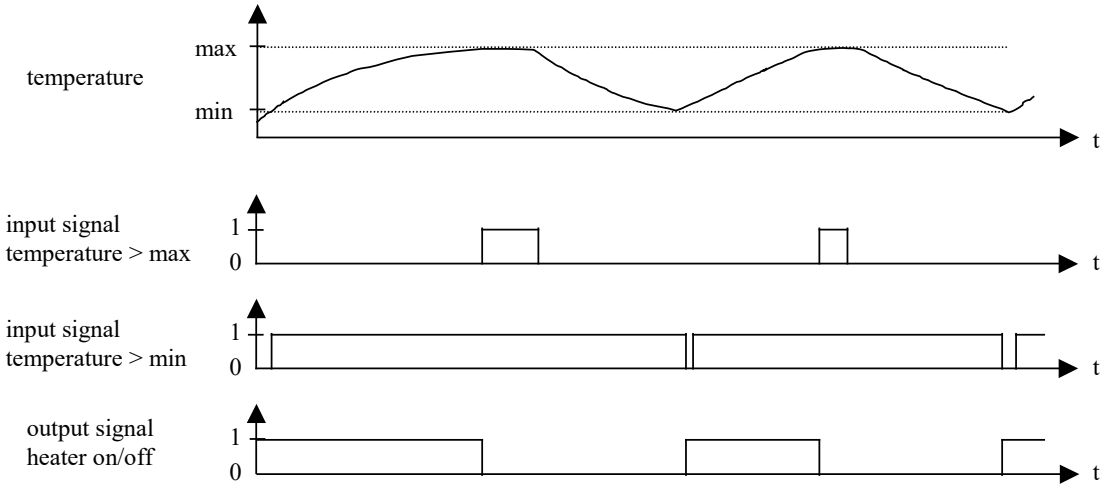

Figure 5.5: Timing diagram of control signals and controlled temperature in a simple closed-loop temperature control system

The corresponding simple logic circuit is shown in Figure 5.6.

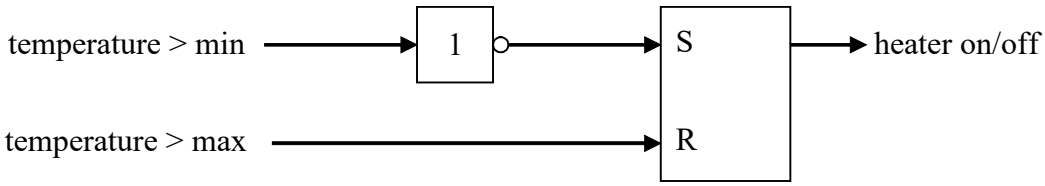

Figure 5.6: Logic circuit of a closed-loop temperature control system

6. Programmable logic controllers – market analysis

The realization of a control system with electronic logic gates is, in major cases, complicated and time consuming. Instead of the use of electronic logic gates it is recommended to use programmable logic controllers. In the next chapter we will show the application of programmable logic controllers (or, shorter, PLC) for realization of control systems [2]. In this case the controllers are software programmed, instead of circuit programmed controllers made of electronic logic gates.

A world market analysis showed that the five biggest automation companies produce more than two thirds of the programmable logic controllers [3]. These companies are:

- Siemens 22.1 %,
- Allen Bradley 16 %,
- AEG Schneider 10.8 %,
- Mitsubishi 9.9 %,
- Omron 8 %,
- Other producers 33.1 %.

The programmable logic controller's hardware and software of different producers are considerably different, therefore, we will focus our study in the next chapters on only one programmable logic controller's producer. Because Siemens has the major market share, we will restrict our attention only to the Siemens programmable logic controller's hardware and software.

Part II – Overview of Siemens Hardware and Software equipment for automation

The word SIMATIC was, in the past, often regarded as a synonym for Siemens` products intended for industry automation. SIMATIC covered a broad hardware and software assortment. Today, Siemens is introducing a more comprehensive automation concept under the name Totally Integrated Automation. All the hardware and software components are integrated in a single system. Such a concept is made possible through integration in data management, in configuring and programming, and in communication. In the next two chapters we will make a short overview of Totally Integrated Automation [2].

7. Hardware – entire assortment and equipment in a laboratory

The word SIMATIC was often used for programmable logic controllers. However, SIMATIC also denotes other hardware and software components.

7.1 SIMATIC controllers

Controllers are still the most recognized SMATIC components. Controllers are divided with regard to their performances in many groups.

7.1.1 SIMATIC S7-200

The SIMATIC S7-200 is a micro-PLC for open-loop and closed-loop control tasks in mechanical engineering and plant engineering. It combines maximum automation with minimum cost. SIMATIC S7-200 applications range from replacing relays and contactors to handling complex automation tasks in stand-alone mode, in network and within distributed configurations. The STEP 7-Micro programming package permits complete programming of all S7-200 CPUs.

The SIMATIC S7-200 offers various communications capabilities:

- Point-to-point interface; can be operated in PI, MPI mode (S7-200 CPUs are slaves in the MPI-network and cannot communicate with each other), and in user programmable interface mode.
- AS-interface connection; as a master using the CP 242-2 communications processor.
- Connection to PROFIBUS-DP as a slave; using the PROFIBUS-DP module EM 277.

Figure 7.1 shows a laboratory control system with two components:

- CPU 214 with 16 DI and 10 DO,
- EM 235 with 4 AI and 1 AO.

The price of the system: about 600 EURO.

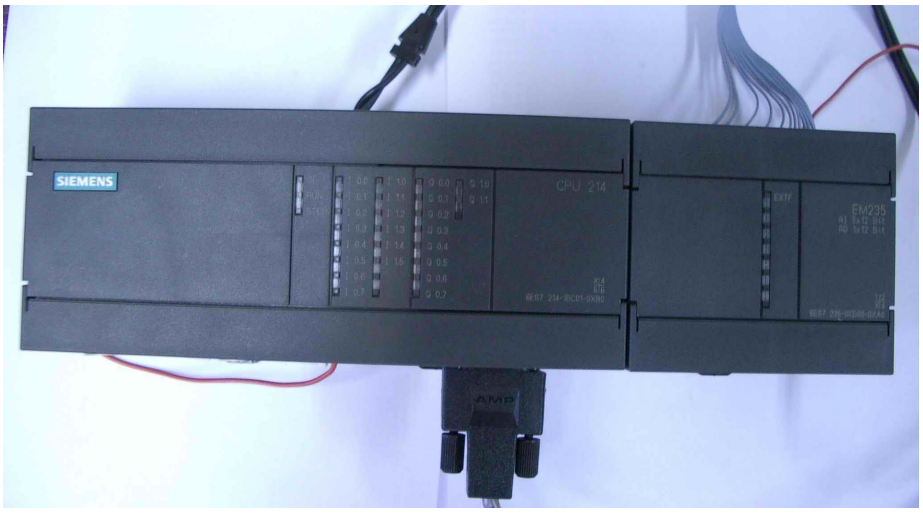

Figure 7.1: Laboratory control system based on PLC CPU 214 and EM 235 modules

7.1.2 SIMATIC S7-300

The SIMATIC S7-300 is suitable for universal use, with the focus on the production industry for the low and medium power ranges. The modular and fan-free design, the easy implementation of distributed structures and the user-friendly operation make the SIMATIC S7-300 an economic and convenient solution for the most varied tasks in the low and medium power ranges.
A system comprises:

- Power supplies (PS) for connecting the SIMATIC S7-300 to a supply voltage of 120/230 V AC.
- A central controller unit (CPU); Different CPUs are available for different power ranges, including CPUs with integrated inputs/outputs and their corresponding function, as well as CPUs, with an integrated PROFIBUS-DP interface.
- Interface modules (IM) for interconnecting the central controller (CC) and expansion unit (EU) wilt multi-tier configuration. The SIMATIC S7-300 can be operated with a max. 32 modules, distributed between the CC and three EUs. All the modules can be enclosed and operated without a fan.
- Signal modules (SM) for digital and analog inputs and output.
- Communication processors (CP) for bus coupling and point-to-point connections.
- Functional modules (FM) for rapid counting, positioning (controlled/regulated) and control.

The S-300 CPUs are programmed in LAD, FBD or STL using STEP 7-Basis.

The S7-300 has various communication interfaces:

- Communication modules for point-to-point connections.
- Multi point interface (MPI) integrated with the CPU is a cost-effective solution for the simultaneous connection of programming devices, human machine interfaces, and other SIMATIC S7 and C7 automation systems.
- Communication modules for interfacing to the AS-interface, PROFIBUS and industrial Ethernet bus systems.

Figure 7.2 shows a laboratory control system with two components:

- CPU 314 with 16 DI, 16 DO, 8 AI and 1 AO,
- power supply.

The price of the system: about 1300 EURO.

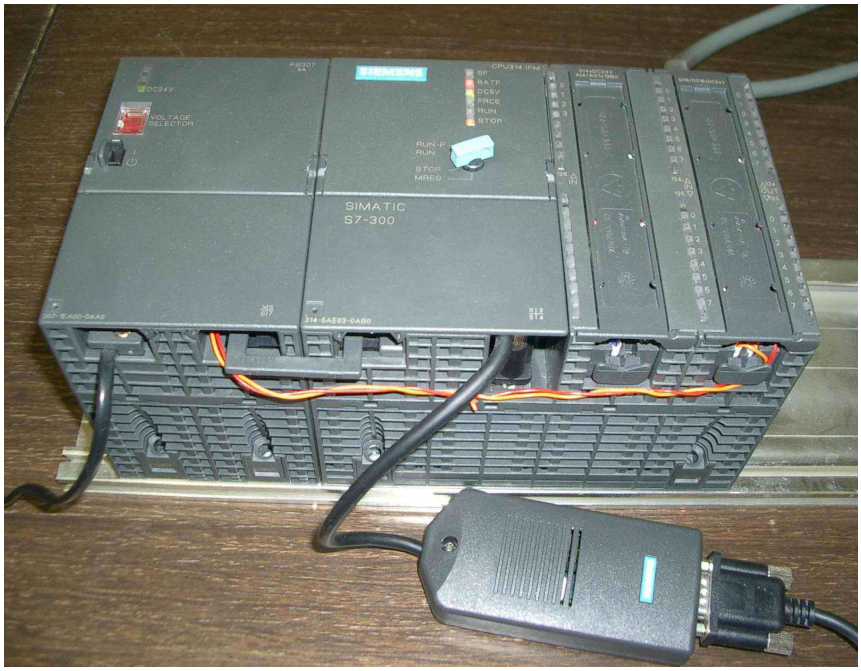

Figure 7.2: Laboratory control system with PLC CPU 314 and power supply

7.1.3 SIMATIC S7-400

The SIMATIC S7-400 is a power PLC for the mid- to high-end performance ranges. Applications for the SIMATIC S7-400 include:

- general mechanical engineering,
- automotive industry,
- warehousing,
- machine-tool construction,
- process engineering,
- instrumentation and control technology,
- textile machines,
- packaging machines,
- control-equipment construction,
- special machines.

Several classes of CPUs with graduated performance capabilities and an extensive module spectrum with many user-friendly functions allow the user to customize a solution to suit his automation task.

The S7-400 automation system is of modular design. It has at its disposal an extensive number of modules which can be combined as required individually.

A system comprises:

- A power supply module; used to connect the SIMATIC S7-400 to a mains voltage of 120/230 V AC.
- Central processing units (CPUs); various CPUs are available to different performance ranges, including some with an integrated PROFIBUS-DP interface. To upgrade performance even more, several CPUs can be used in one central controller.
- Interface modules (IMs) for connecting the central controllers and expansions units. The SIMATIC S7-400s central controller can be operated with as many as 21 expansions units.
- Signal modules (SMs) for digital (DI/DO) and analog (AI/AO) input/output.
- Communications modules (CPs) for bus links and point-to-point connections.
- Functions modules (FMs); the specialists for sophisticated tasks, such as counting, positioning, cam control.

The SIMATIC S7-400 provides various communications options:

- Combined multi-point-capable MPI and DP master interface, integrated in all CPUs; for simultaneous connection of PG/PC, HMI systems, S7-200 and S7-300 systems and additional S7-400 systems.
- Additional PROFIBUS-DP interface integrated in several CPUs for cost effective linking of the ET 200 distributed I/O.
- Communications modules for links to the PROFIBUS and Industrial Ethernet bus system.
- Communications modules for powerful point-to-point connection.
- Process communications; for cyclic addressing of I/O modules (exchange of process images) by using a bus (AS Interface or PROFIBUS-DP). Process communication is called by cyclic execution levels.

Laboratory working places consist of:

- 1 power supply,
- 1 CPU 414-3,
- 1 ET200M,
- 1 digital input module 321, 16 inputs,
- 1 digital output module 322, 16 outputs,
- 1 analog input module 331, 8 inputs,
- 1 analog output module 332, 4 outputs,
- 1 operating panel OP17,
- 1 communications processor for AS Interface CP 343-2, for communication with sensors and actuators, analog input 3RK1207, analog output 3RK1107,
- 1 communications processor CP 443-1, used to connect SIMATIC S7-400 to the Industrial Ethernet (Transfer rate 10/100 Mbit/).

The price of this system is about 10,000 EUR.
The laboratory working place is presented in Figure 7.3.

Figure 7.3: Laboratory control system based on SIMATIC S7-400 modules

7.1.4 SIMATIC C7

SIMATIC C7 are complete control systems for the low-end performance range. They combine a SIMATIC S7-300 programmable controller and a SIMATIC operator panel in one unit. Applications for the SIMATIC C7 include:

- general mechanical engineering,
- special purpose mechanical engineering,
- plastics mechanical engineering,
- textile machinery,
- wood-working machinery,
- food processing machinery.

The STEP 7 and ProTool software tools from the SIMATIC Industrial Software range are available for SIMATIC C7.

7.2 SIMATIC PG (Programming device)

SIMATIC PG (Programming devices) are industry standard programming devices intended for maintenance and servicing assignments, as well as for normal programming and configuring tasks. SIMATIC PGs are in notebook size, the installed software eliminates the need for time consuming preparation, and enables the user to concentrate on his automation tasks. The PGs have one MPI/PROFIBUS-DP interface up to 12 Mbauds for SIMATIC S7. The software supplied with the device is installed on the hard drive in one language. The remaining languages of the STEP software can be installed later from the STEP backup CD.

7.3 SIMATIC Industrial PC

SIMATIC Industrial PCs are robust and safe systems for PC based automation and industrial applications. SIMATIC PCs are available in four forms:

- Box PC; for installation where space is at a premium.
- Rack PC; for installation in switchgear cabinets and consoles.
- Tower PC; for installations in supervisory consoles and in engineering offices.
- Panel PC; for visualization of processes and operations in harsh industrial environments.

7.4 SIMATIC HMI (Human Machine Interface)

The HMI (Human Machine Interfaces) systems are products for operator control and visualization. They are part of every SIMATIC system, and are becoming increasingly important, even in the world of mini controllers. The HMI system requests the process data it needs for its configurable process displays from the SIMATIC controllers. The data are then forwarded to the HMI system automatically, so that forwarding need not be taken into account in the user program.
The SMATIC panels are configured with the SIMATIC ProTool configuring software, which executes under Windows. The systematic adherence to the principle “WYSIWYG”, easily identifiable icons, and drop down menus, have made the software clear and easy to read, reducing the assimilation and configuring time considerably.

7.5 SIMATIC NET

Communication is playing an increasingly important role, particularly in large plants. SIMATIC NET is the name of a whole family of networks. Widely used industrial Standards open the system to all sides. There are many SIMATIC NET bus systems:

- Industrial Ethernet is the widely accepted, powerful bus system for area- and cell networking to International Standards IEEE 802.3 / IEEE 802.3u.
- PROFIBUS is the bus system for the lower and mid performance range based on PROFIBUS Standard IEC 61 158 / EN 50 170.
- AS-Interface is a networking system for binary actuators and sensors in the lowest performance range.
- EIB (EN 50 090, ANSI EIA 776) is the internationally standardized building services system and the basis for building services automation.
- MPI (Multi Point Interface) is the successor to SINEC L1. MPI is the cost efficient choice for simple, but powerful networking of HMI systems, PGs/PCs and other SIMATIC systems.

- PPI (Point to Point Interface) is available for the SIMATIC S7-200. The S7-200 can communicate with a wide variety of peers over this interface.

SIMATIC NET has been based on Standards consistently right from the start. From the very beginning, Siemens has been a member of the standardization committees working on International Standards for industrial communication.

7.6 SIMATIC PCS 7 (Process Control System)

SIMATIC PCS 7 is the process control system which produced a unique, open platform for modern, future oriented and economical solutions in the processing industry. SIMATIC PCS 7 is an integrated system. Its control system characteristics, from engineering to operation, guarantee the fulfilment of all demands for future-oriented process engineering:

- modular and scalable design,
- fast, integrated, system-wide engineering,
- integrated fieldbus and link to the MES-/ERP level.

SIMATIC PCS 7 is based on SIMATIC components such as controller, PC, etc. Together with typical control system software, these standard products provide all the function, scope and performance of a modern process control system.

7.7 SIMATIC DP (Decentralized peripheries)

Distributed structures are highly acceptable, as they are more flexible, less complex, and, in many cases, cheaper. In conjunction with the PROFIBUS fieldbus, an integrated concept has been implemented for SIMATIC which enables the very highest system performance.
SIMATIC ET 200 modules are used for connections of sensors and actuators on site. They transmit signals between sensors or actuators and a higher-level controller or control system over the PROFIBUS-DP fieldbus, resulting in considerable saving on the wiring overhead.

SIMATIC ET 200 offers a suitable distributed solution for all sectors and applications:

- from simple I/O modules to multifunction systems,
- motor starters, frequency converters, pneumatic components, technology modules, security functions can be integrated, or even distributed intelligence (CPU),
- degree of protection IP 20 or IP 65/67.

SIMATIC ET 200 I/O devices are configured by means of STEP 7 or COM PROFIBUS. As standardized PROFIBUS slaves, however, they can be integrated into master devices from other manufacturers and their configuration tools by means of a GSD file.

7.8 Some PLC configurations

7.8.1 Typical PLC components

The programmable logic controller has a modular design. We can set up our own individual system by combining components from a comprehensive range of programmable logic controller modules.

The range of modules includes the following components:

- CPUs for various performance ranges,
- power supply modules for connecting the S7-300 to 120/230 VAC power supplies,
- signal modules for digital and analog input/output,
- modules for connecting: interface modules for the interconnection of the racks in multi-rack installations and communications processors,
- function modules for technological functions.

We also need a programming device to program the S7-300. Mainly, we connect the programming device to the CPU with a special programming device cable.

7.8.2 Typical connections of PLC components

SIMATIC offers a range of communication networks to suit different requirements. These requirements of the automation landscape can be categorized into the following four automation levels:

- **Management level:** At the management level supervisory tasks are processed, which affect the entire works (management functions). These include storing process values, as well as optimizing and analyzing processing functions and their output in the form of reports. The data required for these reports are collected and processed from various sites. From the management level, it is also possible to access other sites. The number of stations can exceed 1,000.
- **Cell level:** At the cell level, all automation and optimization functions are processed autonomously. At this cell level, programmable controllers, PCs and human-machine interfaces are connected to each other.
- **Field level:** The field level is the link between the installations and the programmable controllers. The field devices measure, signal, and transmit the commands from the cell level to the installations. Small data volumes are usually transferred. A hierarchic communication arrangement is typical for this level, i.e., several field devices communicate with one master.
- **Actuator/Sensor level:** At this level, a master communicates with the actuators and sensors that are connected to a subnet. Its characteristic feature is a fast response time for a small number of data bits.

SIMATIC offered the following possibilities, which meet the requirements of the different automation system levels (management, cell, field, and actuator/sensor level):

- Connection with Interface Modules,
- Point-to-point Link,
- MPI,
- PROFIBUS,
- Industrial Ethernet,
- AS-Interface.

Connection with interface modules

If we mount the programmable controller on several racks, we require interface modules. The task of the interface modules is to connect the S7-300 backplane bus from one rack to the next. The following rules apply to the arrangement of the modules:

- The interface module is always located in slot 3, to the left of the first signal module.

- No more than 8 modules (SM, FM, CP) are permitted per rack. The modules (SM, FM, CP) are always located to the right of the interface modules. Exception: In the case of the CPU 314 IFM, a module must not be plugged into slot 11 on rack 3.
- The number of modules (SM, FM, CP) that can be plugged in is limited by the permissible current drawn from the S7-300 backplane bus. The total current consumption per tier or rack must not exceed 1.2 A.
- We can connect up to 4 racks. The CPU is always in rack 0. For rack 0 an Interface module IM360 is used, and, for the racks 1 to 3, the interface module IM 361.

Point–to-point link

A point-to-point link is not technically a subnet. A point-to-point link allows data to be exchanged via a serial link. The point-to-point link can be used between our system and other programmable controllers, computers, or non-Siemens` systems with communication capability. In SIMATIC this link is implemented via point-to-point communication processors, whereby two stations are linked together [4].

Table 7.1: Characteristics of point-to-point communication

Point-to-point link	
Stations	2
Transfer medium	Serial interface-specific cable
Physical interfaces	RS 323C 20 mA RS 422/485
Transmission rates	From 300 bit/s to max. 76.8 Kbit/s with RS 232C and RS 422/485 max 19.2 Kbit/s with 20 mA
Max. length of network	10 m with RS 232C 1,000 m with 20 mA and 9.6 Kbit/s 1,200 m with RS 422/485 and 19.2 Kbit/s
Protocol drivers	ASCII driver 3964 (R) RK 512 Printer driver Loadable special drivers

Multipoint interface

The multipoint interface is suitable for the field level and cell level with low coverage. An MPI is a multipoint interface in SIMATIC S7 and C7 systems. It is designed as a programming device interface, and is intended for networking a small number of CPUs for the purpose of exchanging small volumes of data. The MPI is integral to the S7/M7 and C7 CPUs. This provides a simple networking capability.

Table 7.2: Characteristics of multipoint interface communication

Multipoint interface	
Standards	SIEMENS- specific
Stations	Maximum of 32 active stations
Access technique	Token passing
Transmission rate	187.5 Kbit/s
Transfer medium	Shielded 2-core cable, fiber optic (glass or plastic)
Max. length of network	Segment length 50 m via RS 485 repeaters up to 1,100 m, with fiber-optic cables via OLM > 100 km
Topology	Line, tree, star, ring
Services	S7 functions Global data communication

PROFIBUS

PROFIBUS is the network for the cell and the field level in the open, multi-vendor SIMATIC communication system.

Two versions of PROFIBUS are offered:

- PROFIBUS DP offers a standardized interface for the transfer of process input and process output data between the SIMATIC S7 stations and field devices (DP slaves). PROFIBUS-DP is characterized by high-speed, cyclic exchange of small quantities of data between the DP master and DP slaves.
- PROFIBUS subnet for the cell and field level supports the exchange of information between field devices and with systems at a higher system level. It is used to transfer small to medium quantities of data with communication partners that have equal rights. In SIMATIC S7, a communication processor is always required for PROFIBUS.

PROFIBUS is, physically, either a copper cable network based on a shielded 22-core cable or a fiber-optic cable network . The network access technique for PROFIBUS corresponds to the “Token bus” method. This access technique allows stations to be added and removed under operating conditions [5].

Table 7.3: Characteristics of PROFIBUS communication

PROFIBUS	
Standards	EN 50170 Volume 2 PROFIBUS
Stations	Maximum of 127 stations in the network
Access technique	Token bus for bus allocation between active stations Master/slave for communication with passive stations.
Transmission rate	9.6 Kbit/s to 12 Ambit/s
Transfer medium	Shielded 2-core cable, or fiber-optic cable
Max. length of network	Copper: Segment length up to 1,000 m with repeaters up to 10 km with fiber-optic cables depending on the type of OLM used > 100 km
Topology	line, tree, star, ring
Services	S7 functions, FDL, FMS, DP

Industrial Ethernet

Industrial Ethernet is the network for the management and cell level in the open, multi-vendor SIMATIC communication system. Industrial Ethernet supports communication between computers and programmable controllers. Industrial Ethernet is suitable for the high-speed exchange of large quantities of data, and facilitates communication between one site and another via a gateway. Physically, Ethernet is a copper cable network based on a shielded coaxial cable, a twisted-pair cable, or a fiber-optic network.

Table 7.4: Characteristics of Industrial Ethernet communication

Industrial Ethernet	
Standards	IEEE 802.3
Stations	More than 1,000
Access technique	CSMA/CD (carrier sense multiple access/collision detection)
Transmission rate	10 Mbit/s
Transfer medium	Copper: 2-core, shielded coaxial cable; Industrial Twisted Pair Fiber-optic cable
Max. length of network	Copper: 1.5 km Fiber-optic: 4.5 km
Topology	Line, tree, star, ring
Services	S7 functions ISO transport ISO-on-TCP

AS-Interface

The AS-interface or actuator/sensor interface is a subnet system for the lowest process level in automation systems. It is specially designed for the interconnection of binary sensors and actuators. The AS-Interface is a so-called “Single-master system”, i.e., only one master exists in each AS-subnet that controls data transfer. It calls all slaves in sequence, and reads or writes the data. The master/slave access with cyclic polling guarantees the response time. The AS-I

bus is not simply dedicated to the transfer of data between sensors/actuators and the master, it also supplies power to the sensors. No configuration is required before start-up. Slaves can be replaced without the need for configuration. Numerous devices (actuators/sensors) can be connected as a result of manufacturer-independent standardization. A power supply unit is required for supplying power via the bus.

Table 7.5: AS-Interface Communication Overview

AS-Interface	
Standards	AS-interface specification to IEC TG 178
Stations	1 master and max. 31 slaves
Access technique	Master/slave access technique
Transmission rate	167 Kbit/s
Response time	Max. 5 ms for 31 slaves
Transfer medium	Unshielded 2-core cable
Max. length of network	Cable length max. 300 m (with repeaters)
Topology	Line, tree
Services	AS-I functions

7.8.3 Programmable logic controller configuration 1

Figure 7.4 shows a single-module configuration, consisting of:

- CPU 214.

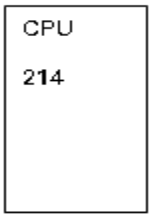

Figure 7.4: Single-module PLC configuration (CPU 214)

This module already has a power supply and digital signal modules with 16 DI and 10 DO integrated in it. For the programming of this system we have to connect the programming device and programmable logic controller with a programming device cable.

Programmable logic controller configuration 2

Figure 7.5 shows a configuration with 2 modules, also shown in Figure 7.2, consisting of:

- power supply,
- CPU 314 IFM.

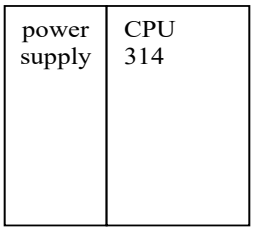

Figure 7.5: Two-module PLC configuration (CPU 314, power supply)

This module already has digital signal modules with 16 DI and 16 DO, and analog signal modules with 4 AI and 1 AO integrated in it. For the programming of this system we have to connect the programming device and programmable logic controller with a programming device cable with an MPI/RS232 adapter.

7.8.4 Programmable logic controller configuration 3

Figure 7.6 shows a possible PLC configuration with up to ten modules on a single rack.

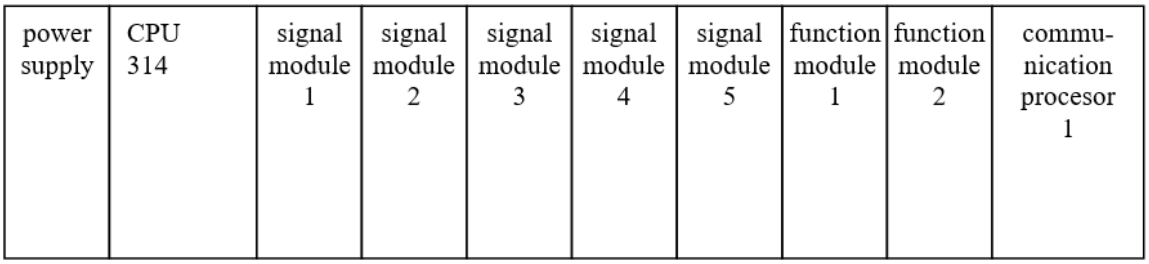

Figure 7.6: PLC configuration with up to ten modules on a single rack

The following modules are used:

- power supply,
- CPU 314 IFM,
- up to 8 signal, function or communication modules.

No more than 8 modules may be mounted to the right of the CPU.

7.8.5 Programmable logic controller configuration 4

Figure 7.7 shows a possible PLC configuration with up to 32 signal, function, and communication modules distributed across four racks.

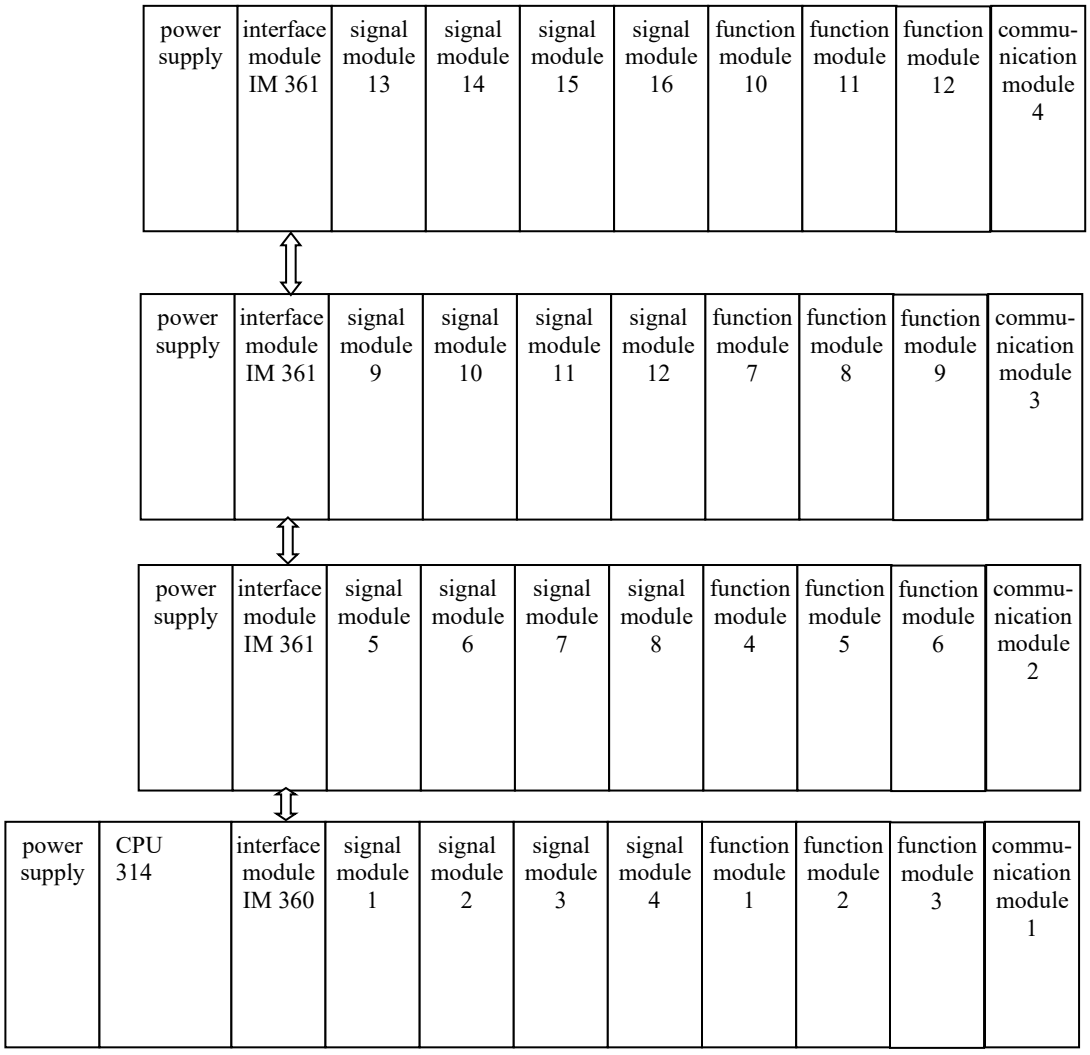

Figure 7.7: PLC configuration with up to 32 signal, function, and communication modules across four racks

The following modules are used:

- power supplies,
- CPU 314 (or 315 or 316),
- up to 32 signal, function or communication modules,
- interface modules (one interface module on every rack).

The connecting cables for the interface modules could be up to 10 m long.

7.8.6 Programmable logic controller configuration 5

Figure 7.8 shows a possible S7-400 PLC configuration with a PROFIBUS DP connection to an S7-300 module.

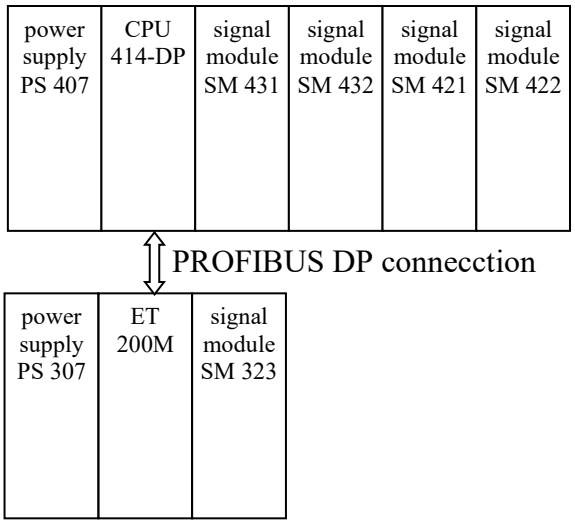

Figure 7.8: PLC configuration with an S7-400 CPU and an S7-300 module

The following modules are used:

- power supply PS 407 4A>6ES7 407-0DA00-0AA0,
- power supply PS 307,
- CPU 414-2DP>6ES7 414-2XJ00-0AB0,
- analog input module SM-400>AI8x13Bit>6ES7 431-1KF00-0AB0,
- analog output module SM-400>AO8x13Bit>6ES7 432-1HF00-0AB0,
- digital input module: SM-400>DI32xDC 24V>6ES7 421-1BL00-0AA0,
- digital output module SM-400>DO32xDC 24V/0.5A>6ES7 422-1BL00-0AA0,
- **PROFIBUS DP module ET 200M> IM 153-1>6ES7 153-1AA03-0XB0,**
- digital input/output module SM 323 DI8/DO8xDC24V/0.5A>6ES7 323-1BH01-0AA0.

7.8.7 Programmable logic controller configuration. 6

Figure 7.9 shows a possible S7-400 CPU configuration with Industrial Ethernet, PROFIBUS DP, and AS-Interface networks:

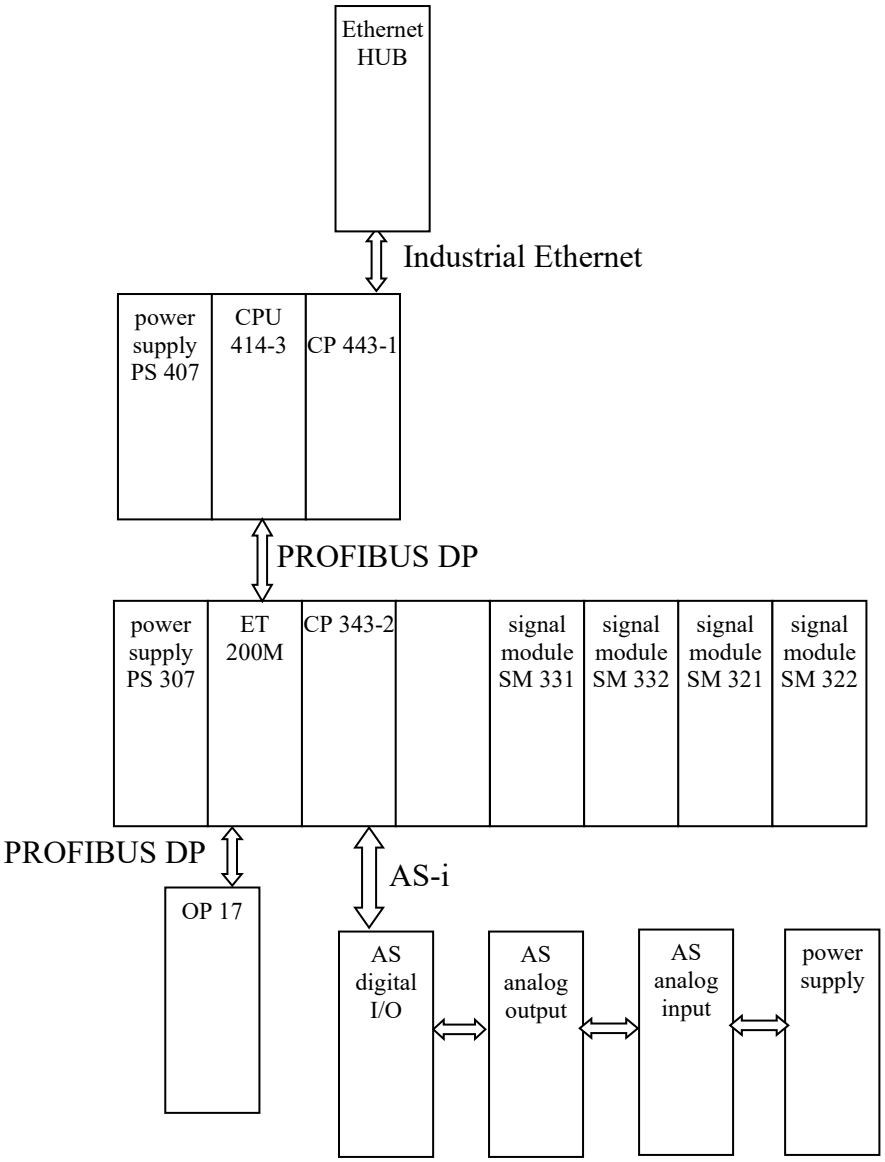

Figure 7.9: PLC configuration with an S7-400 CPU and multiple networks

The following modules are used:

- 3 power supplies,
- 1 CPU 414-3,
- 1 ET200M,
- 1 digital input module 321, 16 inputs,
- 1 digital output module 322, 16 outputs,
- 1 analog input module 331, 8 inputs,
- 1 analog output module 332, 4 outputs,
- 1 operating panel OP17,

- 1 communications processor for AS Interface CP 343-2 for communication with sensors and actuators, analog input 3RK1207, analog output 3RK1107,
- 1 communications processor CP 443-1, used to connect SIMATIC S7-400 to the Industrial Ethernet (Transfer rate 10/100 Mbit/),
- AS-I analog input module, analog output module, and digital input/output module.

8. Software − entire assortment and equipment in the laboratory

The SIMATIC industrial software is the system of seamlessly integrated software tools for the SIMATIC S/, C7 and Win AC automation systems. It provides convenient functions for all phases of an automation project [6]:

- planning, configuring, and parametrization of hardware and communication,
- creation of a user program,
- documentation,
- test, commissioning,
- service,
- process control,
- archiving.

The integration of all software packages into a common user interface supports an efficient and task-oriented workflow.

The SIMATIC industrial software uses the following Standards:

- DIN EN 6.1131-3 Standard,
- Windows 95/98/2000/XP.

The SIMATIC industrial software is unified:

- Common data maintenance; all data of a project (Symbol Table, configuration data, parametrizing data) are stored centrally in a database. They are available for all tools.
- Integrated tool system; convenient functions, which are integrated optimally, are available for each phase of an automation project.
- Open systems; the system platform of the SIMATIC industrial software is open to the world of PC.

SIMATIC industrial software is modular-designed. Four software classes are available:

- standard tools,
- engineering tools,
- runtime software,
- human-machine interfaces.

8.1 Standard tools

Standard tools form the base for programming the SIMATIC S7/C7 programmable controllers. They are always required for programming.

The following standard tools are available:

- STEP 7; the full version for all applications,
- STEP 7-Mini; the version for the lower performance range, usable for SIMATIC S7-300 and SIMATIC C7,
- STEP 7-Micro; the lean programming package for the SIMATIC S7-200.

STEP 7 basic software provides users with different tools for carrying out their automation projects:

- SIMATIC Manager; for joint, straightforward management of all tools and data for the SIMATIC S7, SIMATIC C7 and WinCC.
- Symbol editor; for defining symbolic designations, data types, and comments of global tags.
- Hardware configuration; for configuring the automation system and for parametrizing all modules that can be set.
- Communication; for configuring connections.
- Information functions; for a quick overview of CPU data and the causes of errors while a user program is running.

STEP 7 features the time-tested and standardized PLC programming languages for creating the user program:

- statement list (STL),
- ladder diagram (LAD),
- function block diagram (FBD).

8.2 Engineering tools

Engineering tools are task-oriented tools which can be used in addition to standard tools. They allow the user to focus on the actual project task, and to work according to his or her preferences. Engineering tools reduce engineering costs perceptibly and increase convenience substantially. Engineering tools include:

- high level languages for the programmer,
- graphics languages for the technology specialist,
- supplementary software for diagnosis, simulation, remote maintenance, plant documentation.

The following engineering tools are available:

- S7-SCL (Structured Control Language) is a PASCAL-like high-level program language optimized for programmable controllers. Target systems are S7-300 (from CPU-314 upwards), S7-400, C7, and WinCC.
- S7-GRAPH engineering tool enables configuring and programming of a sequence cascade. Target systems are S7-300 (from CPU-314 upwards), S7-400, C7, and WinCC. The S7-GRAPH offers the user an advanced Windows interface with comprehensive graphics, windows, and zoom capability. In the operation sequences, the individual steps of a process and the transitions to the next steps are represented as rectangles and lines. Within the steps, actions can be initiated by qualifiers (e.g., time-delay, or conditional execution).
- S7-HiGRAPH permits the description of asynchronous processes by the use of state transition diagrams. They serve to describe the states of processes and assemblies, and

the possible transition in graphical form. The user-positioned graphic elements ensure the required flexibility. Target systems are S7-300, S7-400, C7, and WinCC.

- CFC (Continuous Function Chart) enables graphic configuring and programming of automation applications in the form of technology-oriented diagrams. Target systems are S7-400 and WinCC. With the CFC engineering tool, automation applications can be created by drawing a process flow-chart (similar to a control system flowchart for PLC programming). With this graphic programming, the blocks can be arranged and interconnected on a kind of graphic sheet.
- S7-PDIAG enables configuring of process diagnostics for SIMATIC with STL, LAD or STL programming languages. Using process diagnostics, it is possible recognize improper states outside the automation system (e.g., limit switch not reached). Target systems are S7-300 (from CPU-314 upwards), S7-400, C7, and WinCC.
- S7-PLCSIM simulation software enables the functional test of SIMATIC S7 user blocks on a programming device/PC, even if the target hardware is not available.This permits troubleshooting and debugging at an earlier stage of the development process. S7-PLCSIM is suitable for all user blocks and a selection of existing system functions. It can be used for the LAD, FBD, STL, S7-GRAPH, S7-HiGRAPH, S7-SCL and CFC.
- TeleService enables the SIMATIC S7 and C7 automated system to be serviced remotely over a fixed or cellular telephone network using a programming device or PC.
- DOCPRO provides a tool to develop and manage plant documentation. DOCPRO permits structuring of project data, the preparation in the form of wiring manuals, and the printout in a unified print image.
- SIMATIC Micro Computing presents the bridge between the SIMATIC S7-200 and the PC world. It enables processing and visualization data from the S7-200 with standard Windows applications.

8.3 Runtime software

The runtime software includes ready-programmed solutions which are called by the user program. Runtime software is incorporated directly in the automation solution. There are two versions of runtime software [7]:

- Hardware bundled; software assigned to particular hardware, such as function blocks for function modules.
- Hardware unbundled; software with general hardware requirements, such as PRODAVE.

Runtime software includes:

- Control for SIMATIC S7, such as standard, modular, and fuzzy control.
- Tools for integrating automation systems into Windows applications.

8.3.1 Standard PID Control

Standard PID Control software package enables the integration of:

- continuous PID controllers,
- pulse controllers, and
- step action controllers.

in the user program. Standard PID Control can be used in the S7-300 (CPU 313 and higher), S7-400 and C7. The Standard PID Control consists of a parametrization tool and standard function blocks with the different controllers.

8.3.2 Modular PID Control

The Modular PID Control software package is used in those cases where a simple PID controller is not sufficient for solving an automation task. Practically all closed-loop control structures can be implemented by combining the supplied standard function blocks. Modular PID Control is the preferred tool for mid-range and high-end control applications and process engineering. Modular PID Control comprises a start-up tool and standard function blocks. The standard function block package contains 27 standard function blocks.

8.3.3 PID Self Tuner

The PID Self Tuner software package extends existing PID controllers into self-setting PI or PID controllers. The PID Self Tuner is suitable for optimizing:

- temperature controllers,
- level controllers,
- flow rate controllers.

The PID Self Tuner operates at an optimum in processes with:

- stable asymptotic response,
- delay times which are not excessive (delay < 0.1 x compensation time),
- sufficient linearity in the selected operating range,
- adequate quality of the measuring signals,
- a process gain which is not excessive.

8.3.4 Fuzzy Control++

Fuzzy Control++ is a configuring tool for creating fuzzy control systems for SIMATIC S7 and SIMATIC WinCC. Fuzzy control systems are used whenever a mathematical description of a process is difficult or impossible, where operations and processes are unpredictable and non-linearities occur, but know-how gained from experience of dealing with the process is available.

Fuzzy Control++ can be used at all levels of automation, from a single controller right up to the optimization of an entire plant. Fuzzy Control++ can be combined with conventional PID controllers, to utilize the advantages of both systems to obtain optimum control results.

8.3.5 NeuroSytems

Neural networks can be created and trained with the Neuro Systems software package. The spreading of a neural network is based on its learning capability, and on its ability to emulate a special process from a collection of data.

Neuro Systems can be used at all levels of automation, from the single closed-loop controller to the optimization of the plant. Target systems are S7-300 (from CPU-314 upwards) and S7-400.

8.3.6 Loadable drivers for CP 441-2 and CP 341

For the CP 441-2 and CP 341 drivers are available:

- MODBUS master; for communication with the MODBUS protocol with RTU format with SIMATIC S7 as the master.
- MODBUS slave; for communication with the MODBUS protocol with RTU format with SIMATIC S7 as the slave.
- Data Highway; for data transmission to a full duplex protocol Data Highway Asynchronous Link.

8.3.7 PRODAVE MPI

PRODAVE MPI is a toolbox for process data traffic between SIMATIC S7, C7 and PG/PC. It handles data traffic independently using the MPI interface (PPI for S7-200). Additional details on protocol handling are not required.

8.4 Human-machine interface

Human-machine interface is specially designed for operator control and process monitoring with SIMATIC. Human-machine interfaces include [8]:

- ProTool and ProTool/Lite for the configuration of an operator panel,
- ProTool/Pro for machine-level visualization with a PC,
- WinCC; visualization system based on Windows NT,
- ProAgent; the option package for process diagnostics.

8.4.1 ProTool and ProTool/Lite

ProTool and ProTool/Lite are continuous configuration software for all SIMATIC panels and the HMI part of SIMATIC C7. The software is capable of running under Windows 95/98 and NT/2000/XP.

8.4.2 ProTool/Pro

ProTool/Pro is the PC based HMI solution in the machine sector. The SIMATIC ProTool/Pro comprises:

- runtime software SIMATIC ProTool/Pro RT for PC based systems,
- configuration software SIMATIC ProTool/Pro Configuration.

Software packages run under Windows 95/98 and NT/2000/XP.

8.4.3 WinCC

SIMATIC WinCC is the PC based human-machine interface system that runs under Microsoft Windows 95/98/NT/2000/XP. The basic system configuration includes functions meeting industrial requirements for signaling of events, archiving of measured values, logging of all process and configuration data, user administration and visualization.

8.4.4 ProAgent

ProAgent is a standardized diagnostics concept for various SIMATIC components. It enables precise and rapid process fault diagnostics in plants and machines for SIMATIC S7 and SIMATIC HMI.

Part III – General about PLC programming

9. Programs in CPU

Two different types of programs run on a CPU:

- the operating system, and
- the user program.

9.1 Operating system

Every CPU has an operating system that organizes all the functions and sequences of the CPU that are not associated with a specific control task. The tasks of the operating system include the following [3, 4]:

- handling a warm restart and hot restart,
- updating the Process Image Table of the inputs and outputting the Process Image Table of the outputs,
- calling the user program,
- detecting interrupts, and calling the interrupt organization blocks,
- detecting and dealing with errors,
- managing the memory areas,
- communicating with programming devices and other communications partners.

If you change the operating system parameters (the operating system defaults settings), you can influence the activities of the CPU in certain areas.

9.2 User program

You yourself must create the user program and download it to the CPU. This contains all the functions required to process your specific automation task. The tasks of the user program include the following:

- Specifying the conditions for a warm restart and hot restart on the CPU (for example, initializing signals with a particular value).
- Processing process data (for example, combining binary signals logically, reading in and evaluating analog signals, specifying binary signals for output, outputting analog values).
- Specifying the reaction to interrupts.
- Handling disturbances in the normal running of the program.

9.3 Connection between the operating system and user program

PLCs software division in an operating system and user program enables simple PLCs` programming. With basic knowledge about conditions when, why, and which part of the user program will be called from the operating system, you can divide your program into different parts of the user program. Depending on the specific situation, the operating system will call appropriate parts of the user program. Such connections between operating systems makes writing user program simple and transparent.

There are two different types of programs executing:

- cyclic program processing,
- event driven program processing.

9.3.1 Cyclic program processing

Cyclic program processing is the normal type of program execution on programmable logic controllers, meaning the operating system runs in a program loop (the cycle) and calls the appropriate part of the user program - the organization block OB1 one in every loop in the main program. The user program in OB1 is, therefore, executed cyclically [3, 4].

The sequence of cyclic program processing in new CPUs (rom 10/98) can be divided into the following phases:

1. The operating system starts the cycle time monitoring.
2. The CPU writes the values from the Process Image Table of the outputs to the output modules.
3. The CPU reads the state of the inputs of the input modules and updates the Process Image Table of the inputs.
4. The CPU processes the user program, and executes the instructions contained in the program.
5. At the end of the cycle, the operating system executes any tasks that are pending, for example, downloading and deleting blocks, receiving and sending global data.
6. Finally, the CPU returns to the start of the cycle, and restarts the cycle monitoring time.

So that the CPU has a consistent image of the process signals during cyclic program processing, the CPU does not address the input (I) and output (Q) address areas directly on the I/O modules, but rather accesses an internal memory area of the CPU that contains an image of the inputs and outputs. These parts of the internal memory are entitled PII (Process Image Input) and PIQ (Process Image Output). During cyclic program processing by the CPU, the process image is updated automatically.
You program cyclic program processing by writing your user program in OB1, and in the blocks called within OB1 using STEP 7.

Cyclic program processing begins as soon as the startup program is completed without errors. Cyclic program processing can be interrupted by the following:

- an interrupt,
- a STOP command (mode selector, menu option of the programming device, SFC46STP, SFC20STOP),
- a power outage,
- the occurrence of a fault or program error.

The sequence of cyclic program processing in old CPUs has a different order of phases:

1. The operating system starts the cycle time monitoring.
2. The CPU reads the state of the inputs of the input modules, and updates the Process Image Table of the inputs.
3. The CPU writes the values from the Process Image Table of the outputs to the output modules.
4. The CPU processes the user program, and executes the instructions contained in the program.

5. At the end of the cycle, the operating system executes any tasks that are pending, for example, downloading and deleting blocks, receiving and sending global data.
6. Finally, the CPU returns to the start of the cycle and restarts the cycle monitoring time.

9.3.2 Event driven program processing

Cyclic program processing can be interrupted by certain events (interrupts). If such an event occurs, the block currently being executed is interrupted at a command boundary, and a different organization block is called, that is assigned to the particular event. Once the organization block has been executed, the cyclic program is resumed at the point at which it was interrupted [9].

This means it is possible to process parts of the user program that do not have to be processed cyclically only when needed. The user program can be divided up into subroutines, and distributed between different organization blocks. If the user program is to react to an important signal that occurs relatively seldomly (for example, a limit value sensor for measuring the level in a tank reports that the maximum level has been reached), the subroutine that is to be processed when the signal is output can be located in an OB whose processing is event-driven.

9.4 Process image input/output tables

If the input (I) and output (Q) address areas are accessed in the user program, the program does not scan the signal states on the digital signal modules, but accesses a memory area in the system memory of the CPU and distributed I/Os. This memory area is known as the process image. The process image is divided into two parts: the Process Image Input Table and the Process Image Output Table [10].
The CPU can only access the process image of the modules that you have configured with STEP 7, or that are obtainable using the default addressing.
The process image is updated cyclically by the operating system. At the beginning of cyclic program execution, the signal states are transferred from the Process Image Output Table to the output modules, and the signal states of the input modules are transferred to the Process Image Input Table.
Compared with direct access to the input/output modules, the main advantage of accessing the process image is that the CPU has a consistent image of the process signals for the duration of one process cycle. If a signal state on an input module changes while the program is being executed, the signal state in the process image is retained until the process image is updated again at the beginning of the next cycle. Access to the process image also requires far less time than direct access to the signal modules, since the process image is located in the internal memory of the CPU.

With the S7-400 CPU and the CPU 318 you can deselect the update of the process image if you want to:

- access the I/O directly instead, or
- update one or more process image input or output sections at a different point in the program using system functions SFC26UPDAT_PI and SFC27UPDAT_PO.

9.5 Blocks in the user program

The STEP 7 programming software allows you to structure your user program, in other words, to break down the program into individual, self-contained program sections. This has the following advantages:

- Extensive programs are easier to understand.
- Individual program sections can be standardized.
- Program organization is simplified.
- It is easier to make modifications to the program.
- Debugging is simplified, since you can test separate sections.
- Commissioning your system is made much easier.

There are several different types of blocks you can use within the STEP 7 user program:

- organizations blocks (OB),
- system Function Blocks (SFB) and System Functions (SFC),
- functions (FC),
- function blocks (FB),
- instance data blocks (instance DB),
- shared data blocks (DB).

OBs, FBs, FCs and SFCs contain sections of the program, and are therefore also known as logic blocks. The permitted number of blocks per block type and the permitted length of the blocks are CPU-specific.

9.5.1 Organization Blocks (OB)

Organization blocks determine the structure of the user program.

- They form the interface between the operating system and the user program.
- They are called by the operating system, and control the startup of the programmable logic controller, the cyclic and interrupt-driven program execution, and are responsible for handling errors [11].

By programming the organization blocks you specify the reaction of the CPU.
In most situations the predominant type of program execution on programmable logic controllers is cyclic execution. This means that the operating system runs in a program loop (the cycle), and calls the organization block OB1 once each time the loop is executed. The user program in OB1 is, therefore, executed cyclically.
Cyclic program execution can be interrupted by certain events (interrupts). If such an event occurs, the block currently being executed is interrupted at a command boundary, and a different organization block is called, that is assigned to the particular event. Once the organization block has been executed, the cyclic program is resumed at the point at which it was interrupted.

In STEP 7, the following non-cyclic types of program execution are possible:

- time driven program execution,
- process interrupt-driven program execution,
- diagnostic interrupt-driven program execution,
- processing of synchronous and asynchronous errors,

- processing of the different types of start-ups,
- multi computing-controlled program execution,
- background program execution.

The following Table shows the types of interrupts in STEP 7 and the priority of the organization blocks assigned to them. Not all the listed organization blocks and their priority classes are available in all STEP 7 CPUs [12].

Table 9.1: List of STEP 7 interrupts and their priority levels

Type of interrupt	Organization block	Priority class (default)
Main program scan	OB1	1
Time-of-day interrupts	OB10 to OB17	2
Time-delay interrupts	OB20 OB21 OB22 OB23	3 4 5 6
Cyclic interrupts	OB30 OB31 OB32 OB33 OB34 OB35 OB36 OB37 OB38	7 8 9 10 11 12 13 14 15
Hardware interrupts	OB40 OB41 OB42 OB43 OB44 OB45 OB46 OB47	16 17 18 19 20 21 22 23
Multi computing interrupt	OB60	25
Asynchronous error interrupts	OB80 Time error OB81 Power supply error OB82 Diagnostic interrupt OB 83 Insert/remove module interrupt OB84 CPU hardware error OB85 Priority class error OB86 Rack failure OB 87 Communication error	26
Background cycle	OB90	29
Startup	OB100 Warm restart OB101 Hot restart OB102 Cold restart	27 27 27
Synchronous error interrupts	OB121Programming error OB122 Access error	

The priority of organization blocks is fixed on the S7-300 CPU. With S7-400 CPUs, you can change the priority of the organization blocks OB10 through OB47, and the priority in the RUN mode of organization blocks OB81 through OB87 with STEP 7. Priority classes 2 through 23 are permitted for OB10 through OB87, and priority classes 24 through 26 for OB81 through OB87. You can assign the same priority to several OBs. OBs with the same priority are processed in the order in which their start events occur.

9.5.2 Functions (FC)

Functions (FC) belong to the blocks that you program yourself. A function is a logic block ¨without memory¨. Temporary variables belonging to the FC are saved in the local data stack. These data are then lost when the FC has been executed. To save data permanently, functions can also use shared data blocks [13].
Since an FC does not have any memory of its own, you must always specify actual parameters for it. You cannot assign initial values for the local data of an FC.

An FC contains a program section that is always executed when the FC is called by a different logic block. You can use functions for the following purposes:

- To return a function value to the calling block (example: math functions).
- To execute a technological function (example: single control function with a bit logic operation).

You must always assign actual parameters to the formal parameters of an FC. The input, output and in/out parameters user by the FC are saved as pointers to the actual parameters of the logic block that called the FC.

9.5.3 Function blocks (FB)

Functions blocks (FBs) belong to the blocks that you program yourself. A function block is a logic block ¨with memory¨. It is assigned a data block as its memory (instance data block). The parameters that are transferred to the FB and the static variables are saved in the instance DB. Temporary variables are saved in the local data stack.
Data saved in the instance DB are not lost when execution of the FB is complete. Data saved in the local data stack are, however, lost when execution of the FB is completed.

An FB contains a program that is always executed when the FB is called by a different logic block. Function blocks make it much easier to program frequently occurring, complex functions.

9.5.4 Instance data blocks (instance DB)

An instance data block is assigned to every function block call that transfers parameters. The actual parameters and the static data of the FB are saved in the instance DB. The variables declared in the FB determine the structure of the instance data block. Instance means a function block call. If, for example, a function block is called five times in the S7 user program, there are five instances of this block.

Before you create an instance data block, the corresponding FB must already exist. You specify the number of the FB when you create the instance data block.

9.5.5 Shared data blocks (DB)

In contrast to logic blocks, data blocks do not contain STEP 7 instructions. They are used to store user data, in other words, data blocks contain variable data with which the user program works. Shared data blocks are used to store user data that can be accessed by all other blocks.

The size of DBs can vary. Refer to the description of your CPU for the maximum possible size. You can structure shared data blocks in any way to suit your particular requirements.

9.5.6 System Function Blocks (SFB) and System Functions (SFC)

You do not need to program every function yourself. S7 CPUs provide you with preprogrammed blocks that you can call in your user program.
A system function block (SFB) is a function block integrated on the S7 CPU. SFBs are part of the operating system, and are not loaded as a part of the program. Like FBs, SFBs are blocks ¨with memory¨. You must also create instance data blocks for SFBs and load them on the CPU as part of the program [14].

The S7 CPU provides the following SFBs:

- for communication on configured connections,
- for integrated special functions (for example, SFB29¨HS_COUNT¨ on the CPU 312 IFM and the CPU 314 IFM).

A system function is a preprogrammed tested function that is integrated on the S7 CPU. You can call the SFC in your program. SFCs are part of the operating system, and are not loaded as a part of the program. Like FCs, SFCs are blocks ¨without memory¨.

S7-CPUs provide SFCs for the following functions:

- copying and block functions,
- checking the program,
- handling the clock and run-time meters,
- transferring data records,
- transferring events from a CPU to all other CPUs in the multicomputing mode,
- handling time-of-day and time-delay interrupts,
- handling synchronous errors, interrupts, and asynchronous errors,
- system diagnostics,
- process image updating and bit field processing,
- addressing modules,
- distributed peripheral I/O,
- global data communication,
- communication on non-configured connections,
- generating block-related messages.

9.6 Linear and structured programming

You can write your entire user program in OB1 (linear programming). This is only advisable with simple programs written for the S7-300 or S7-200 and requiring little memory.

Complex automation tasks can be controlled more easily by dividing them into smaller tasks reflecting the technological functions of the process, or that can be used more than once. These tasks are represented by corresponding program sections, known as the blocks (structured programming).

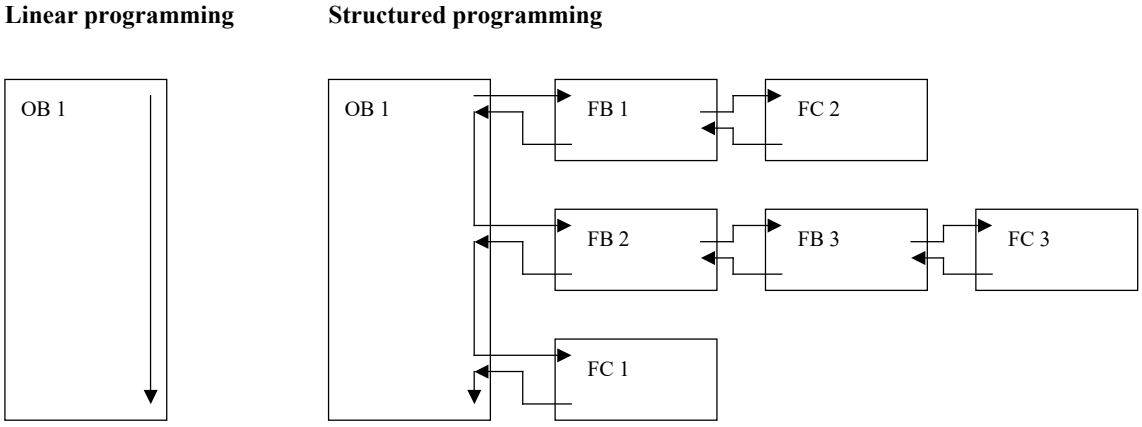

Figure 9.1: Linear vs. structured programming in STEP 7

9.7 Programming languages in STEP 7

Which language you choose depends largely on your own experience and which language you personally find easiest to use. There are many programming languages, and, among them, three are basic:

- statement list programming language,
- ladder logic programming language,
- function block diagram programming language.

9.7.1 Statement list programming language

The programming language representation type STatement List (STL) is a textual language similar to machine code. Each statement corresponds to a step as the CPU works with its way through a program. A number of statements can be linked together to from networks.
The Statement List programming language is included with the STEP 7 Standard software package. You can edit S7 Blocks in this language representation type using incremental editors, or create your program with a free-edit mode editor in a STL source file and then compile it into blocks.

9.7.2 Ladder logic programming language

The graphic programming language Ladder Logic (LAD) is based on the representation of circuit diagrams. The elements of a circuit diagram, such as normally open contacts and normally closed contacts, are grouped together in networks. One or more networks form the code section of a logic block.
The Ladder logic programming language type is included with the STEP 7 standard software package. Creating programs in Ladder Logic is done with an incremental editor.

9.7.3 Function block diagram programming language

The programming language Function Block Diagram uses the graphic logic symbols familiar from Boolean algebra to represent logic. Complex functions such as math functions can also be represented directly in conjunction with the logic boxes.
The FBD programming language type is included with the STEP 7 Standard software package.

9.8 Operating modes and mode changes

Operating modes describe the behavior of the CPU at any particular point in time. Knowing the operating modes of CPUs is useful when programming the startup, debugging and control program, and for troubleshooting.
There are four operating modes of the S7-300 and S7-400 computers:

- STOP,
- STARTUP,
- RUN,
- HOLD.

In the STOP mode, the CPU checks whether all the configured modules or module set by the default addressing actually exist, and sets the I/Os to a predefined initial status. The user program is not executed in the STOP mode.

In the STARTUP mode, a distinction is made between the startup types »Complete Restart« and »Restart«:

- In a complete restart, the program starts at the beginning with initial settings for the system data and user address areas (the non-retentive timers, counters, and bit memory are reset).
- In a restart, the program is resumed at the point at which it was interrupted (the timers, counters, and bit memory are not reset). A restart is only possible on S7-400 CPUs.

In the RUN mode, the CPU executes the user program, updates the inputs and outputs, services interrupts, and processes error messages.
In the HOLD mode execution of the user program is halted, and you can test the user program step by step. The HOLD mode is only possible when you are testing using the programming device.
In all these modes, the CPU can communicate on the MPI interface.

Part IV – STEP 7 Statement list programming

10. Language description

10.1 Structure of statements

With reference to the structure, an instruction statement belongs to either one of the following two basic groups [15]:

- a statement made up of an instruction alone (NOT),
- a statement made up of an instruction and an address (A I124.0).

The address of an instruction indicates a constant, or the location where the instruction finds a value (data object) on which to perform an operation. The address can have a symbolic name or an absolute designation. The address can point to any of the following items:

- a constant, the value of a timer or counter, an ASCII character string to be loaded into accumulator 1,
- a bit in the status word of the programmable logic controller,
- a symbolic name,
- a data block and the location within the data block area,
- a function or function block,
- an address identifier and a location within the memory area that is indicated by the address identifier.

Some addresses include an address identifier and a location within the memory area that is indicated by the address identifier. An address identifier can be one of the following three basic types:

- an address identifier that indicates the memory area and the size of a data object in that area,
- an address identifier that indicates the memory area but no size of a data object in that area,
- an address identifier that indicates the size of a data object but no memory area.

Most addresses in STL refer to memory areas:

- Bit memory – This area provides storage for intern results calculated in the program.
- Process image input – at the beginning of the scan cycles, the operating system reads the inputs from the process and records the values in this area. The program can use these values in its cyclic processing.
- Process image output – During the scan cycle, the program calculates output values and places them in this area. At the end of the scan cycle, the operating system reads the calculated output values from this area and sends them to the process outputs.
- Timers - are function elements of STL programming. This area provides storage for timer cells. In this area, clock timing accesses time cells, to update them by decrementing the time value, and the timer instructions access time cells.
- Counters - are function elements of STL programming. This area provides storage for counters. Counter instructions access them here.
- Data block - This area contains data that can be accessed from any block. If you need to have two different data blocks open oat the same time, you can open one with the

statement »OPN DB« and one with the statement »OPN DI«. In this way, the CPU can distinguish which of the two data blocks your program wants to access while both data blocks are open. While you can use the »OPN DI« statement to open any data block, the principal use of this statement is to open instance data blocks that are associated with function blocks (FBs) and system function blocks (SFBs).

- Local data – This area contains temporary data that are used within a logic block (OB, FB, or FC). These data are also called dynamic local data. They serve as an intermediate buffer. When the logic block is finished, these data are lost. The data are contained in the local data stack (L stack).
- I/O external input/output – These areas enable your program to have direct access to input and output modules (that is, peripheral inputs and outputs).

The following Table lists the memory areas. For the address range possible with your CPU, refer to the technical data of the CPU.

Table 10.1: Overview of STEP 7 memory areas

Name of area	Access to area	Abbrev.	Address range
Bit memory	Memory bit Memory byte Memory word Memory double word	M MB MW MD	0.0 – 255.7 0 – 255 0 – 254 0 - 252
Process image input	Input bit Input byte Input word Input double word	I IB IW ID	0.0 - 65535.7 0 – 65535 0 – 65534 0 – 65532
Process image output	Output bit Output byte Output word Output double word	Q QB QW QD	0.0 - 65535.7 0 – 65535 0 – 65534 0 – 65532
Timer	Timer T)	T	0 - 255
Counter	Counter (C)	C	0 - 255
Data block	Data bit Data byte Data word Data double word	DBX DBB DBW DBD	0.0 - 65535.7 0 – 65535 0 – 65534 0 – 65532
Local data	Temp. local data bit Temp. local data byte Temp. local data word Temp. local data double word	L LB LW LD	0.0 - 65535.7 0 – 65535 0 – 65534 0 – 65532
I/O external input	Peripheral input byte Peripheral input word Peripheral input double word	PIB PIW PID	0 – 65535 0 – 65534 0 – 65532
I/O external output	Peripheral output byte Peripheral output word Peripheral output double word	PQB PQW PQD	0 – 65535 0 – 65534 0 – 65532

10.2 Addressing

Statements made up of an instruction and an address can have different types of addressing:

- immediate addressing,
- direct addressing,
- memory indirect addressing,
- address registers,
- area-internal register indirect addressing,
- area-crossing register indirect addressing.

Most used are immediate addressing and direct addressing.

10.2.1 Immediate addressing

With immediate addressing the address is coded directly in the instruction; that is, it follows the value with which the instruction is to work directly (for example: LOAD). An instruction can also provide its own value (for example: SET).
Some examples of immediate addressing:

- SET — set the RLO to 1,
- L 27 — load the integer value 27 into accumulator 1,
- L 'ABCD' — load the ASCII character 'ABCD' into accumulator 1,
- L B#(100,12) — load the two bytes 100 and 12 into accumulator 1,
- L C#(0100) — load the BCD value 0100 into accumulator 1.

10.2.2 Direct addressing

An instruction that uses direct addressing has the following two-part address, that indicates the location of the value that the instruction is going to the process:

- an address identifier (for example, »IB« for »input byte«),
- an exact location within the memory area that is indicated by the address identifier.

The address points directly to the location of the value.
Some examples of direct addressing:

- A I0.0 — perform an AND logic operation on input bit I0.0,
- S L20.0 — set the local data bit L20.0,
- = M115.4 — assign the RLO to memory bit M115.4,
- L IB0 — load input byte IB0 into accumulator 1,
- L MW64 — load memory word MW64 into accumulator 1,
- T DBD12 — transfer the contents from accumulator 1 into double word DBD12.

10.3 Accumulators and status word

10.3.1 Accumulators

The two 32-bit accumulators are general purpose registers that you use to process bytes, words, and double words [16]. You can load constants or values from the memory as addresses into

the accumulator and perform a logic operation on them. You can also transfer the result of an operation from accumulator 1 o a memory location. The areas of an accumulator are shown in Figure 10.1.

word	1 (high word)																0 (low word)													
byte	3 (high byte)								2 (high byte)								1 (high byte)								0 (high byte)					
bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Figure 10.1: Structure of the 32-Bit Accumulators in STEP 7

The stack mechanism for accumulator administration is as follows:

- a LOAD instruction always acts on accumulator 1 and saves the old contents to accumulator 2,
- a TRANSFER instruction does not change the accumulators.

10.3.2 Status word

The status word contains bits that you can reference in the address of bit logic and word logic instructions. Figure 10.2 shows the structure of the status word. The sections that follow the Figure explain the significance of bits 0 through 8.

status word															
1								0							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
							BR	CC1	CC0	OV	OS	OR	STA	RLO	FC

Figure 10.2: Status word bit assignment in STEP 7

First check - FC

Bit 0 of the status word is called the first-check bit (FC). The signal state of 0 in the FC bit indicates that, following this point in your program, the next logic instruction begins a new logic string. (The bar under the FC indicates that it is negated.)
Each logic instruction checks the signal state of the FC bit, as well as the signal state of the location it addresses.

- If the FC bit is 0, the instruction stores the result of the signal state check in the result of the logic operation bit of the status word (RLO bit) and sets the FC bit to 1. This process is called a first check.
- If the signal state of FC bit is equal to 1, an instruction combines the result of its signal state check on the contact it addresses with the value stored in the previous RLO bit.

A string of logic instructions always ends with an output instruction (S, R or =), a jump instruction related to the result of logic operation (JC), or one of the nesting instructions A(, O(, X(, AN(, ON(, XN(. Such an output, jump instruction, or nesting instruction resets the FC bit to 0.

Result of logic operation – RLO

Bit 1 of the status word is called the RLO bit (RLO – Result of Logic Operation). This bit stores the result of a bit logic instruction or math comparison.
You can see the effect of the FC bit on logic instruction and RLO on the next example:

Table 10.2: Changes in Result of Logic Operation (RLO) and First check (FC) for a given program example

statement	input/output	RLO	FC	Explanation
		-	0	FC=0 indicates that the next instruction begins a logic string
A I1.0	I1.0=1	1	1	The result of the first check is stored in RLO, FC is set to 1
A I1.1	I1.1=1	1	1	RLO is obtained from the previous RLO and a value of I1.1 according to the AND Truth Table, FC remains 1
= Q1.0	Q1.0=1	-	0	RLO is assigned to the output Q1.0, FC is reset to 0

During the executing of the program both bit logic inputs I1.0 and I1.1 are equal to 1. The first instruction checks the input I0.0, and, according to the state of input1.0, sets or resets RLO. The second instruction performs the AND operation with two operands: RLO and I1.1 – the result is stored in RLO, replacing the former value in the RLO bit. Each subsequent instruction in the string performs a logic operation on two values: the result produced when the instruction checks the contact and the current RLO. A string of logic instructions ends with an output instruction =Q1.0 - the output Q1.0 is set to 1.
You can set the RLO to 1 unconditionally by using the SET instruction; you can reset the RLO to 0 unconditionally by using the CLR instruction. You can use a Boolean bit logic instruction on a first check to assign the state of the contents of a Boolean bit memory location to the RLO. You can use the RLO to trigger jump instructions.

Status bit – STA

The status bit (STA bit) stores the value of a bit that is referenced. The status of a bit instruction that has read access to the memory (A, O, AN, ON, X, XN) is always the same as the value of the bit that this instruction checks (the bit on which it performs its logic operation). The status of a bit instruction that has write access to the memory (S, R, =) is the same as the value of the bit to which the instruction writes, or, if no writing takes place, the same as the value of the bit that the instruction references. The status bit is not checked by an instruction. It is interpreted during a program test (program status) only.

OR bit (OR)

The OR bit is needed if you use the O instruction to perform a logical AND before OR operation. An AND function may contain the following instructions: A, AN, A(, AN(, and NOT. The OR bit shows these instructions that a previously executed AND function has supplied the value 1, thus forestalling the result of the logical OR operation. Any other bit-processing command resets the OR bit.

Overflow bit (OV)

The overflow bit indicates a fault. It is set by a math instruction or a floating-point comparison instruction after a fault occurs (overflow, illegal operation, illegal floating-point number). This bit is set according to the result of the next math instruction or comparison instruction.

Stored overflow bit (OS)

The stored overflow bit is set together with the OV bit when a fault occurs. Because the OS bit remains set after the fault has been eliminated, it stores the OV bit status, and indicates whether or not a fault occurred in one of the previously executed instructions. The following commands reset the OS bit: JOS (jump after stored overflow), the block call commands, and the block end commands.

Condition Code 1 and Condition Code 0 (CC1 and CC0)

The CC1 and CC0 bits (condition codes) provide information on the following results of bit:

- Result of a math operation (in math instructions without overflow, CC0 and CC1 indicate if the results is equal, greater to or less than zero, in math instructions with overflow, CC0 and CC1 indicate a range overflow).
- The result of comparison (in comparison, instructions CC0 and CC1 indicate the values of the accumulators 1 and 2).
- Result of digital operation.
- Bits that have been shifted out by a shift or rotate command.

Binary result Bit (BR)

The binary result bit (BR) forms a link between the processing of bits and words. It is an efficient means of interpreting the result of a word operation as a binary result, and integrates this result in a binary logic string. Viewed in this way, the BR bit represents a machine-internal memory bit to which the RLO is saved prior to a word operation that changes the RLO, so that the RLO will be available again after the operation to continue the interrupted bit string.

11. Instructions

11.1 Bit logic instructions

11.1.1 Boolean bit logic AND instruction

Table 11.1: Description and operation of instruction A (AND) in STEP 7

A n									
n (bit): I, Q, M, L, DBX, DIX, T, C									
Instruction AND (A) has two different functions: • In the case, when it is just before the execution of instruction A, the value of the FC bit of the status word equals ″0″, then instruction A checks the signal state of an address, to establish whether an address is set at ″1″ or ″0″, and then **instruction copies** this one bit logic value of signal state check to the RLO bit of the status word. At copying from digital inputs (I) or digital outputs (Q), instruction A really copies information from the process input image (PII) or process output image (PIQ). At copying from timers or counters, in fact, instruction A copies one bit timer or counter value momentarily. The value in the RLO bit of the status word before the executing of instruction A is, in this case, not important. • In the case, when it is just before the execution of instruction A, the value of the FC bit of the status word equals ″1″. Then, instruction A checks the signal state of an address, to establish whether an address is set at ″1″ or ″0″, and then **instruction performs a logical AND** operation with two operands: one operand is the signal state check and the other operand is the value stored in the RLO bit of the status word just before the execution of instruction A. The result of the AND operation will be stored in the RLO bit of the status word. The result overwrites the old value in RLO.									
Status word	BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
instruction depends on:	-	-	-	-	-	yes	-	yes	yes
instruction influence on:	-	-	-	-	-	yes	yes	yes	1
Instruction time for CPU 314 IFM: 0.2 µs (I, Q), 0.6 µs (M), 0.8 µs (L), 2.7 µs (DBX, DBI), 0.8 µs (T), 0.6 µs (C)									
Examples: A I0.0, A Q1.0, A M10.7, A M0.4, A T3, A C4, A L1.0									

11.1.2 Boolean bit logic OR instruction

Table 11.2: Description and operation of instruction O (OR) in STEP 7

O n									
n (bit): I, Q, M, L, DBX, DIX, T, C									
Instruction OR (O) has two different functions: • In the case, when it is just before execution of instruction O, the value of the FC bit of the status word equals ″0″. Then, instruction O checks the signal state of an address, to establish whether an address is set at ″1″ or ″0″, and then **instruction copies** this one bit logic value of the signal state check to the RLO bit of the status word. At copying from digital inputs (I) or digital outputs (Q), instruction O really copies information from the process input image (PII) or process output image (PIQ). At copying from timers or counters, instruction O copies, in fact, one bit timer or counter value momentarily. The value in the RLO bit of the status word before the executing of instruction O is, in this case, not important. • In the case, when it is just before execution of instruction O, the value of the FC bit of the status word equals ″1″, then instruction O checks the signal state of an address, to establish whether an address is set at ″1″ or ″0″, and then **instruction performs a logical OR** operation with two operands: one operand is the signal state check, and other operand is the value stored in the RLO bit of the status word just before the execution of instruction O. The result of the OR operation will be stored in the RLO bit of the status word. The result overwrites the old value in the RLO.									
Status word	BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
Instruction depends on:	-	-	-	-	-	-	-	yes	yes
Instruction influence on:	-	-	-	-	-	0	yes	yes	1
Instruction time for CPU 314 IFM: 0.2 µs (I, Q), 0.6 µs (M), 0.8 µs (L), 2.7 µs (DBX, DBI), 0.8 µs (T), 0.6 µs (C)									
Examples: O I0.0, O Q1.0, O M10.7, O M0.4, O T3, O C4, O L1.0									

11.1.3 Set and Reset Instructions

Table 11.3: Description and operation of instruction S (Set) in STEP 7

S n									
n (bit): I, Q, M, L, DBX, DIX, T, C									
Instruction SET (S) sets the signal state of an addressed bit to 1, depending on the value in the result of the logic operation before the execution of instruction S: • The S instruction sets the bit that the instruction addresses to 1 if the result of the logic operation from the previous statement is 1. • If the result of the logic operation from the previous statement is 0, the instruction S does not influence the signal state of the addressed bit. The instruction terminates the logic string.									
Status word	BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
Instruction depends on:	-	-	-	-	-	-	-	yes	-
Instruction influence on:	-	-	-	-	-	0	yes	-	0
Instruction time for CPU 314 IFM: 0.3 µs (I, Q), 0.8 µs (M), 1.2 µs (L), 3.3 µs (DBX, DBI)									
Examples: S I0.0, S Q1.0, S M10.7, S M0.4, S T3, S C4, S L1.0									

Table 11.4: Description and operation of instruction R (Reset) in STEP 7

R n									
n (bit): I, Q, M, L, DBX, DIX, T, C									
Instruction RESET (R) resets the signal state of an addressed bit to 0, depending on the value in the result of the logic operation before the execution of instruction S: • The R instruction resets the bit that the instruction addresses to 0 if the result of the logic operation from the previous statement is 1. • If the result of the logic operation from the previous statement is 0, the instruction R does not influence the signal state of the addressed bit. The instruction terminates the logic string.									
Status word	BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
Instruction depends on:	-	-	-	-	-	-	-	yes	-
Instruction influence on:	-	-	-	-	-	0	yes	-	0
Instruction time for CPU 314 IFM: 0.3 µs (I, Q), 0.8 µs (M), 1.2 µs (L), 3.3 µs (DBX, DBI)									
Examples: R I0.0, R Q1.0, R M10.7, R M0.4, R T3, R C4, R L1.0									

11.1.4 Assign Instruction

Table 11.5: Description and operation of instruction = (Assign) in STEP 7

= n									
n (bit): I, Q, M, L, DBX, DIX									
Instruction Assign (=) copies the value of the RLO from the previous statement into the bit that the instruction addresses. The instruction terminates the logic string.									
Status word	BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
Instruction depends on:	-	-	-	-	-	-	-	yes	-
Instruction influence on:	-	-	-	-	-	0	yes	-	0
Instruction time for CPU 314 IFM: 0.2 µs (I, Q), 0.9 µs (M), 1.0 µs (L), 3.3 µs (DBX, DBI)									
Examples: = I0.0, = Q1.0, = M10.7, = M0.4									

Part V – Installing STEP 7 and STEP 7 Standard Package description

12. Installing STEP 7 basis

12.1 Requirements for installation

12.1.1 Software requirements

STEP 7 software is available for Microsoft Windows 95, 98, Me, NT and XP (Version 5.5) operating systems. For the installation of newer versions Internet Explorer version 4.01 SP1 or newer is also required.

12.1.2 Hardware requirements

For STEP 7 running we need a programing device or PC with the following hardware components:

- processor 80486 or higher,
- minimum 16 MB RAM, recommended is 32 MB,
- color monitor, keyboard, mouse, CD device, floppy disc device, hard disc, and series interface.

STEP 7 Basis software is distributed on a CD. The STEP 7 Basis Version 5.1 CD has 359 MB. On the CD we can find 6 folders:

- Acroread (with Acrobat Reader),
- Authors (software for authorization),
- Manuals (with some basic manuals),
- Ncm and SimaticNetUpdate (folders with some network software), and
- Step7 (the main installation folder).

The Step 7 folder has 174 MB.

Memory capacity required on the hard disc:

- The standard package occupies ca. 230 MB. The memory required depends on the installation options selected for the standard software.
- STEP 7 should have approximately 60 MB, including the main memory available to create Swap files (meaning approximately 28 MB if there are 32 MB of main memory).
- For user data it is necessary to reserve at least 50 MB.
- The setup requires at least 1 MB of free memory on drive C: (the setup files are deleted when the installation is completed).

12.2 Installation process

STEP 7 Basis installation is simple and not problematic. STEP 7 contains a Setup program which executes the installation automatically. Prompts on the screen guide the user step by step through the whole installation process. The main stages in the installation are:

- copying the data to the programming device (PC),
- setting the drivers for EPROMs and communication,
- authorization.

If the installation program finds another version of STEP 7 on the programming device, it reports this and prompts you to decide how to proceed:

- abort the installation, and then uninstall the old STEP 7 version and then start installation again, or
- continue the installation and overwrite the old version with the new version.

The Setup.exe program is stored on the CD in the folder D:\Step7\Disk1. The installation process lasts on a P4 1.8 MHz notebook with 256 MB RAM approximately 20 minutes.
After running D:\Step7\Disk1\Setup.exe the blue Siemens welcome screen appears. In the continuation it is necessary to answer some questions [17]:

- Choice of Setup language.
- Warning that the program is protected by copyright – choice next.
- Possibility to read a readme file.
- The requirements for installing are shown (32 MB RAM, 200 MB HD, Internet Explorer).
- License Agreement.
- Fill in the Name, Company and ID number.
- Ask for a suitable installation folder, usually c:\siemens\Step7 is recommended.
- Ask for the version: Standard (all languages for the user interface, all applications, all examples), Minimum (only one language, no examples), or User dependent (selecting which programs, database, examples, and communications to install).
- Ask for languages to be installed (German, English, French, Spanish, Italian).
- Ask for the language for startup Step 7.
- During installation the program checks to see whether an authorization is installed on the hard disk. If no authorization is found you can run the authorization program immediately, or continue the installation and execute the authorization at a later date.
- During installation a dialog box is displayed, where you can assign parameters to memory cards,
    - if you are not using memory cards you do not need an EPROM driver. Select the option ¨No EPROM Driver¨,
    - otherwise, select the entry which applies to your programming device,
    - if you are using a PC, you can select a driver for an external programmer.Here, you must specify the port to which the programmer is connected (for example, LPT1).
- Ask for installing interfaces- when we intend to connect the PC with PLC via a PC Adapter which connect the PC's series RS 232 interface with the PLC's MPI interface, then select PC Adapter. In this case, the number of the adequate COM port (usually COM 2) and the transmission rate (usually 19200 bps), must be filled in.

If the installation was successful a message appears on the screen to confirm this. The installation process must be concluded with restarting the computer.

12.3 Authorization

The STEP 7 programming software requires a product-specific authorization (or license for use). The software is therefore copy-protected, and can be used only if the relevant authorization for the program or software package can been found on the hard disk of the respective PC or programming device. Different authorizations are required by different versions of STEP 7.
A read-only authorization disk is included with the scope of supply of the software. It contains the authorization's data. The number of authorizations possible is determined by an authorizations counter on the authorization disk. Every time you install an authorization, the counter is decremented by 1. When the counter value reaches zero, you cannot install any more authorizations using this disk.

An authorization may be lost, for example, if a hard disk defect occurs, and you did not have a chance to remove the authorization from the defective hard disk. If you lose your authorization you can use the emergency license also included on the authorization disk. The emergency license allows you to continue running the software for a limited period. In this case, the time remaining before the validity period runs out is displayed on the screen as you start. Within this period, you should make sure that you obtain a replacement for your lost authorization from your local Siemens representative.
The Siemens software authorization hotline telephone number is: 0049 911 895 7200.
If you attempt to start STEP 7 software and there is no authorization available for the software, a message appears to tell you this. The authorization could be installed during installing STEP 7 software, or before or after the STEP 7 installation. To install authorization out of the STEP 7 installation process you need the program AuthorsW.exe.
On the STEP 7 installation CD you can find in folder d:\Authors\Disk1 file Setup.exe. Executing this file installs the file AuthorsW.exe in folder C:\AuthorsW. Running the file an AuthorsW.exe window appears:

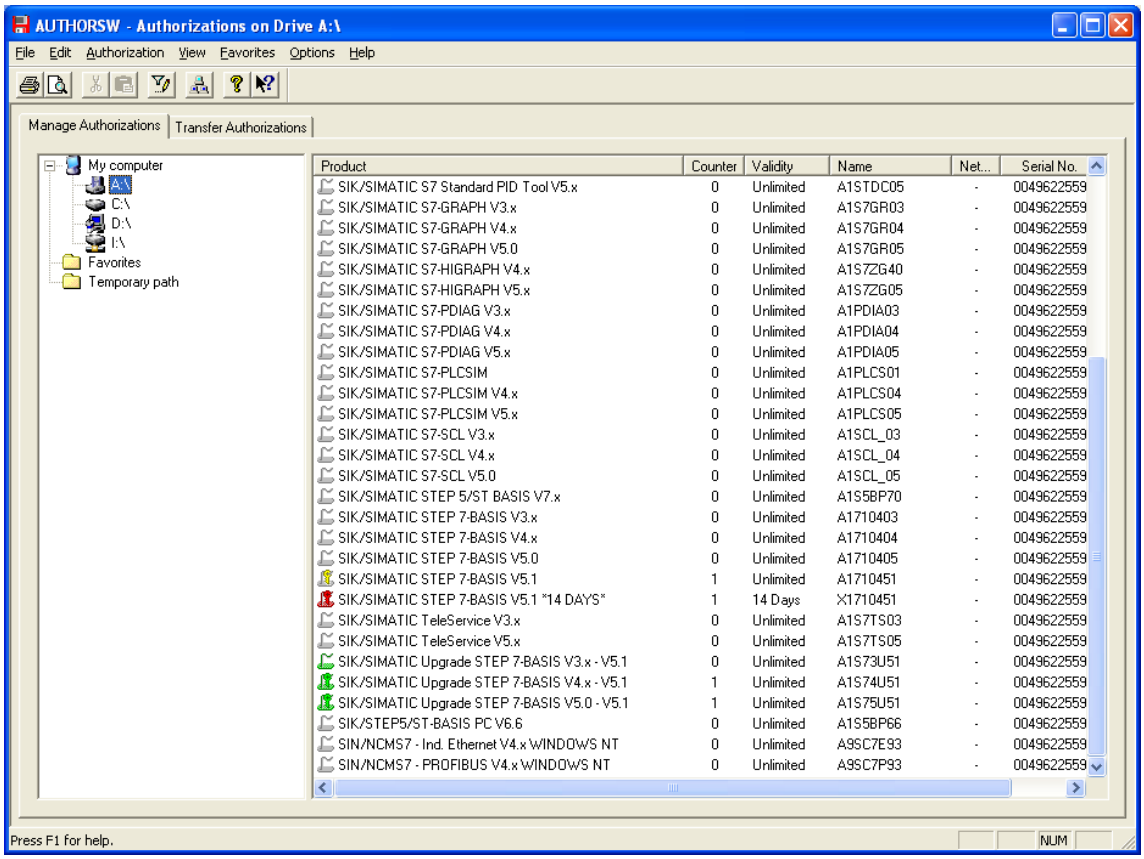

Figure 12.1: AuthorsW window for authorization management

In the right field you can see a number of available licenses on the authorizations floppy disk and the hard disk. For an authorizations license transfer you select the option Transfer Authorizations and transfer the authorization from the source to the target. The authorization is transferred to the physical drive, and your computer registers the fact that the authorization has been installed.

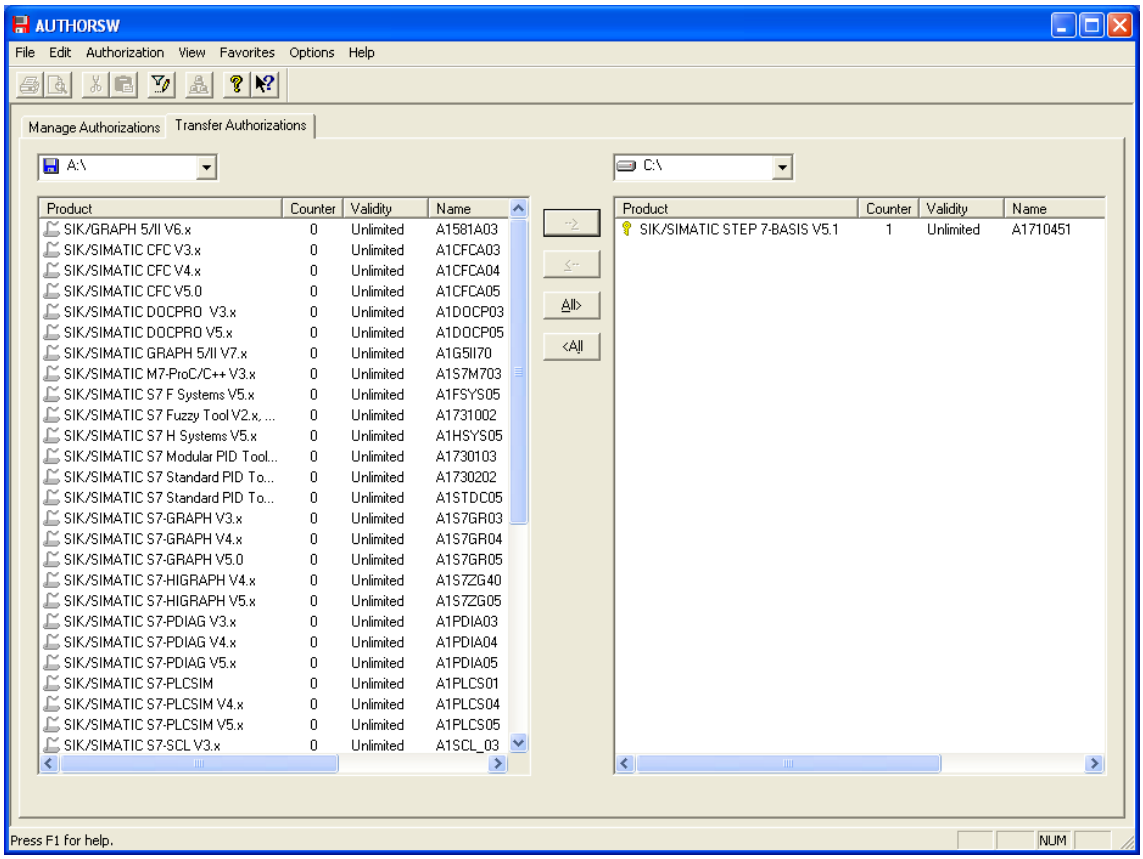

Figure 12.2: AuthorsW window for transferring authorizations

For STEP 7 V5.1, the authorization must be installed on the destination drive C:. If C: is a compressed drive (for example, DBLSPACE\DBLDRIVE), select the respective host drive. Hidden files are stored in the folder ¨C:\AX NF ZZ¨. These files and the folder must not be deleted, moved, or copied. They contain data required or the authorization of your software. If you do not adhere to these guidelines, the authorization may be lost irretrievably.

To help you install your authorization without errors, adhere to the guidelines in the README.TXT file supplied with the authorization diskette, which help you prevent loss of the authorization or any data [18]:

- The authorization disk must not be read-only. Because the authorization disk is to be used without write protection, there is a danger that a virus could be transferred from the hard disk to the disk. You should therefore run a virus check on your PC or programming device every time you install or remove an authorization.
- If you use an optimizer program which enables you to move fixed blocks of memory, only use this option once you have moved the authorizations from the hard disk back to the authorization disk.
- When you install an authorization, a cluster appears on the destination drive which is marked as defective. Do not attempt to restore the defective cluster.
- Before you format, compress or restore your hard disk drive, or before installing a new operating system, you must remove any existing authorizations.
- If a backup copy of your hard disk contains copies of authorizations, there is a danger that these copies may overwrite the valid installed authorizations when you restore your backup data to the hard disk, thereby destroying them.

- To prevent a valid authorization being overwritten by a back-up copy, you must remove all authorizations before you make a back-up copy, or exclude the authorizations from the backup.

Executing the authorization upgrade for STEP 7 V3.x, V4.x or V5.0 to V5.1 before installing STEP 7V5.1 is recommended. The requirements for the upgrade are as follows:

- You have an individual license or a copy license for STEP 7 V3.x, V4.x or V.0, which is either on an authorization diskette or on your computer.
- You have obtained an upgrade license for STEP 7 V5.1 in the form of a diskette labeled ¨Authorization Diskette Upgrade¨. When you upgrade the authorization from Version V3.x, V4.x or V5.0 to V5.1, your authorization for the previous version will be lost irretrievably.

12.4 Setting the PC/PLC interface

There are two possibilities to establish the connections between the PC used for programming and the programmable logic controller:

- you can equip the PC with a corresponding MPI card (Multi Point Interface), or
- you can use a cable with a PC adapter – an adapter between the RS 232 interface on the PC side and MPI interface on the PLC side.

In the majority of cases we use a cable with a PC adapter.

12.5 First running of STEP 7

After the first running of STEP 7, the graphic user interface is set to the English language, but STL programming is set to the German language. If we want to use the English version of the mnemonic programming language changing of the settings is necessary.

We use SIMATIC Manager:

Options-Customize-Language-Language/English, Mnemonics/IEC
Mnemonics SIMATIC means the German version of the mnemonic programming language STEP 7; mnemonics IEC means the English version. The changes are valid after the first restart of STEP 7.

13. The STEP 7 Standard Package

The Standard Package runs on the operating system Windows 95/98/NT/2000/XP, and matches the graphic and object-oriented operating philosophy of Windows.

The Standard software supports the user in all phases of the creation process of an automation task:

- setting up and managing projects,
- configuring and assigning parameters to hardware and communications,
- managing symbols,
- creating programs to the programmable controller,
- testing the automation system,
- diagnosing plant failures.

The STEP 7 Standard Package provides a series of tools (applications) within the software:

- SIMATIC Manager,
- Hardware Configurator,
- NETPRO Communication Configuration,
- Symbol Editor,
- Programming Languages LAD, FBD and STL,
- Hardware diagnostics.

13.1 SIMATIC Manager

The SIMATIC Manager manages all the data that belong to an automation project – regardless of which programmable control system (S7, C7, M7) they are designed for [18]. The tools needed to edit the selected data are started automatically by the SIMATIC Manager. Figure 13.1 shows the SIMATIC Manager window.

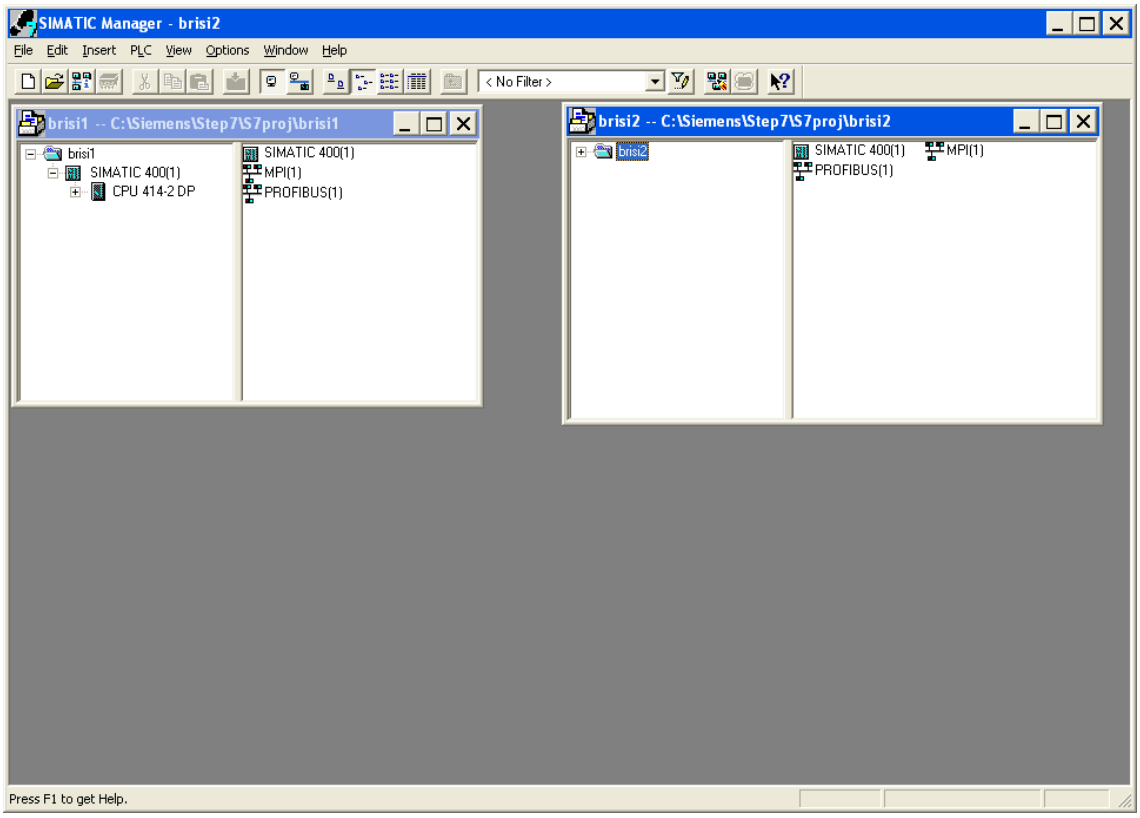

Figure 13.1: SIMATIC Manager main window

There are 8 groups of commands in the SIMATIC Manager menu bar:

- File: commands for opening, organizing, and printing projects.
- Edit: commands for editing blocks.
- Insert: commands for inserting hardware and software components.
- PLC: commands for downloading the program and monitoring the hardware.
- View: commands for setting the window display.
- Options: commands for selecting the language and making settings for process data.
- Window: commands for arrangement of the project windows.
- Help: commands for calling the STEP 7 online help.

The Project window is divided into two halves:

- The contents of the left-hand pane show the project structure.
- The contents of the right-hand pane show the objects and other folders for the folder selected on the left.

13.2 Hardware Configurator

The Hardware Configurator can be used to configure and assign parameters to the hardware of an automation project. The following functions are available:

- To configure the programmable controller, the user selects racks from an electronic catalog and arranges the selected modules in the required slots in the racks.
- Configuring the distributed I/O is identical to the configuration of the central I/O.
- In the course of assigning parameters to the CPU the user can set properties such as startup behavior and scan cycle time monitoring guided by the menus.
- In the course of assigning parameters to the modules, all the parameters the user can set are set using the dialog boxes. There are no settings to be made using DIP switches. The assignment of parameters to the module is done automatically during the startup of the CPU.
- Assigning parameters to function modules (FMs) and communications processors.

Figure 13.2 shows the Hardware Configuration window.

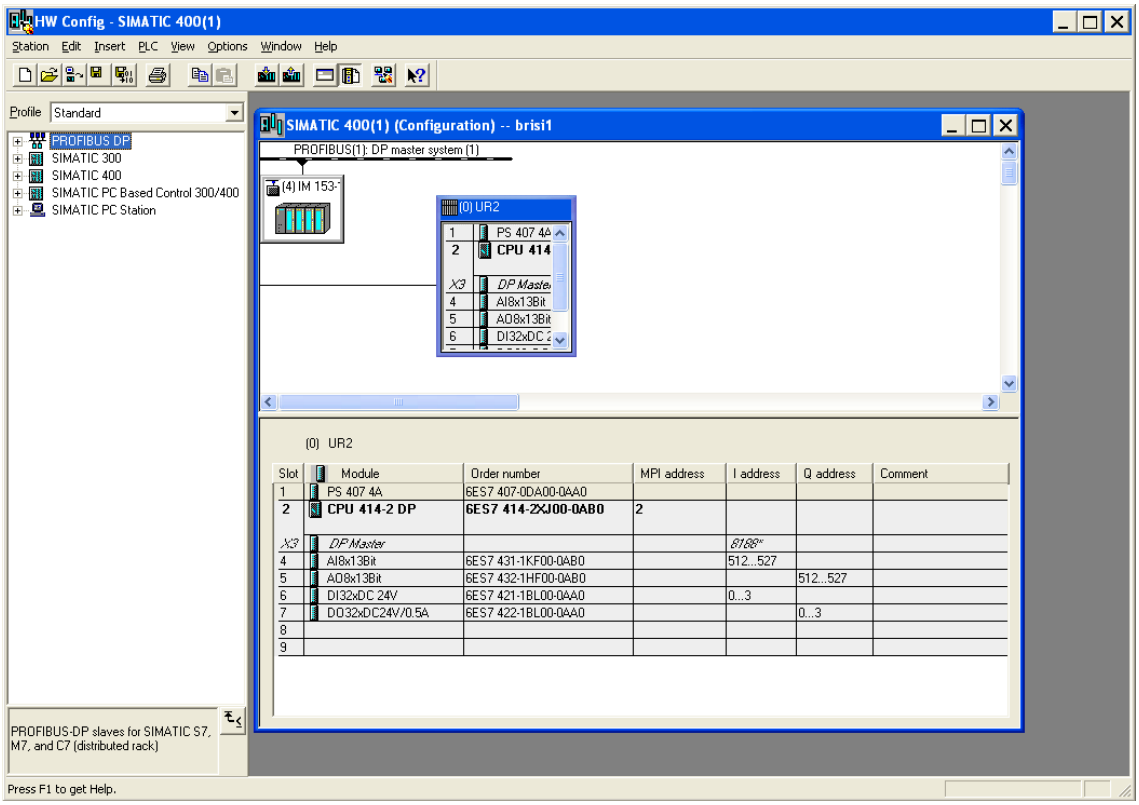

Figure 13.2: Hardware Configuration window

13.3 NETPRO Communication Configuration

The basis for communication is a pre-configured network. For this, the user will use NetPro to create the subnets required for automation networks, set the subnet properties, and set the network connection properties and any communication connections required for the network stations.

13.4 Symbol Editor

Each input and output has an absolute address predefined by the hardware configuration. This address is specified directly; that is, absolutely. The absolute address can be replaced by any symbolic name you choose. With Symbol Editor users manage all the shared symbols. The following functions are available:

- Setting symbolic names and comments for the process signals (inputs/outputs), bit memory, and blocks.
- Sort functions.
- Export/Import to/from other Windows programs.

The Symbol Table created with his tool is available to all other folders. Any changes to the properties of a symbol are therefore recognized automatically by all tools.
Figure 13.3 shows the Symbol Editor window:

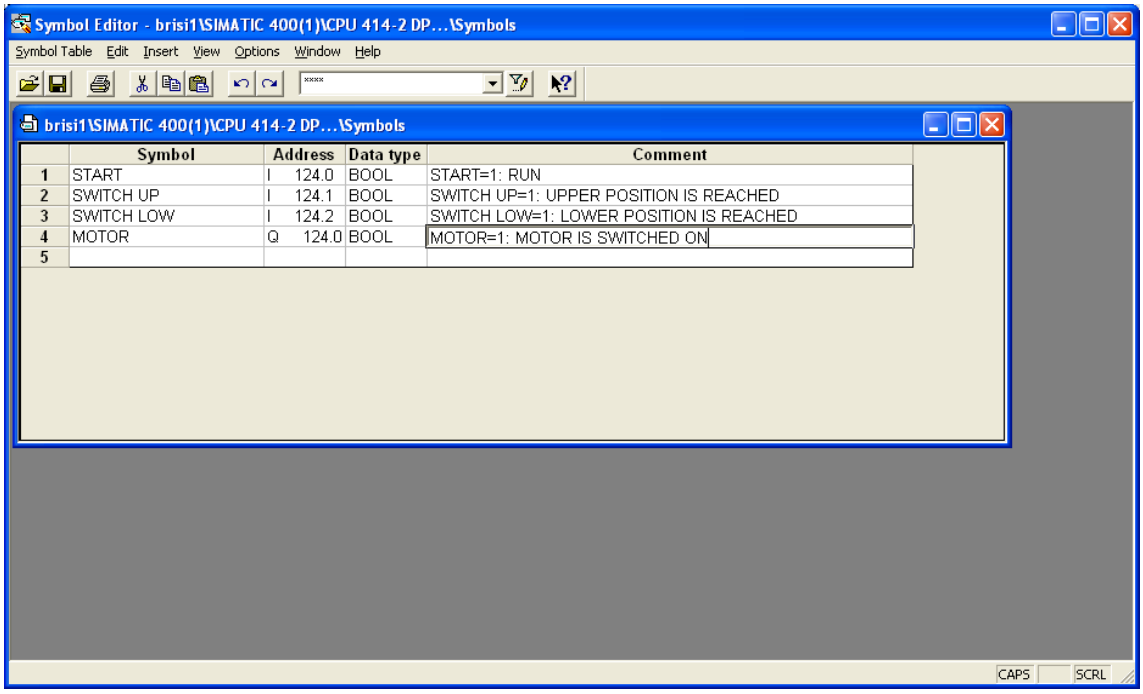

Figure 13.3: Symbol Editor window

13.5 Programming Languages LAD, FBD and STL

The Programming languages Ladder Logic, Statement List and Function Block Diagram for S7-300 and S7-400 are an integral part of the Standard package.

- Ladder Logic (LAD) is a graphic representation of the STEP 7 programming language. Its syntax for the instructions is similar to a relay ladder logic diagram: Ladder allows

to track the power flow between the power rails as it passes through various contacts, complex elements, and output coils.

- Function Block Diagram (FBD) is a graphic representation of the STEP 7 programming language and uses the logic boxes familiar from Boolean algebra to represent the logic. Complex functions (for example, math functions) can be represented directly in conjunction with the logic boxes.
- Statement List (STL) is a textual representation of the STEP 7 programming language, similar to machine code. If a program is written in Statement List, the individual instructions correspond to the steps with which the CPU executes the program. To make programming easier, Statement List has been extended to include some high-level language constructions (such as structured data access and block parameters).

Figure 13.4 shows the **Programming Languages LAD, FBD and STL** window:

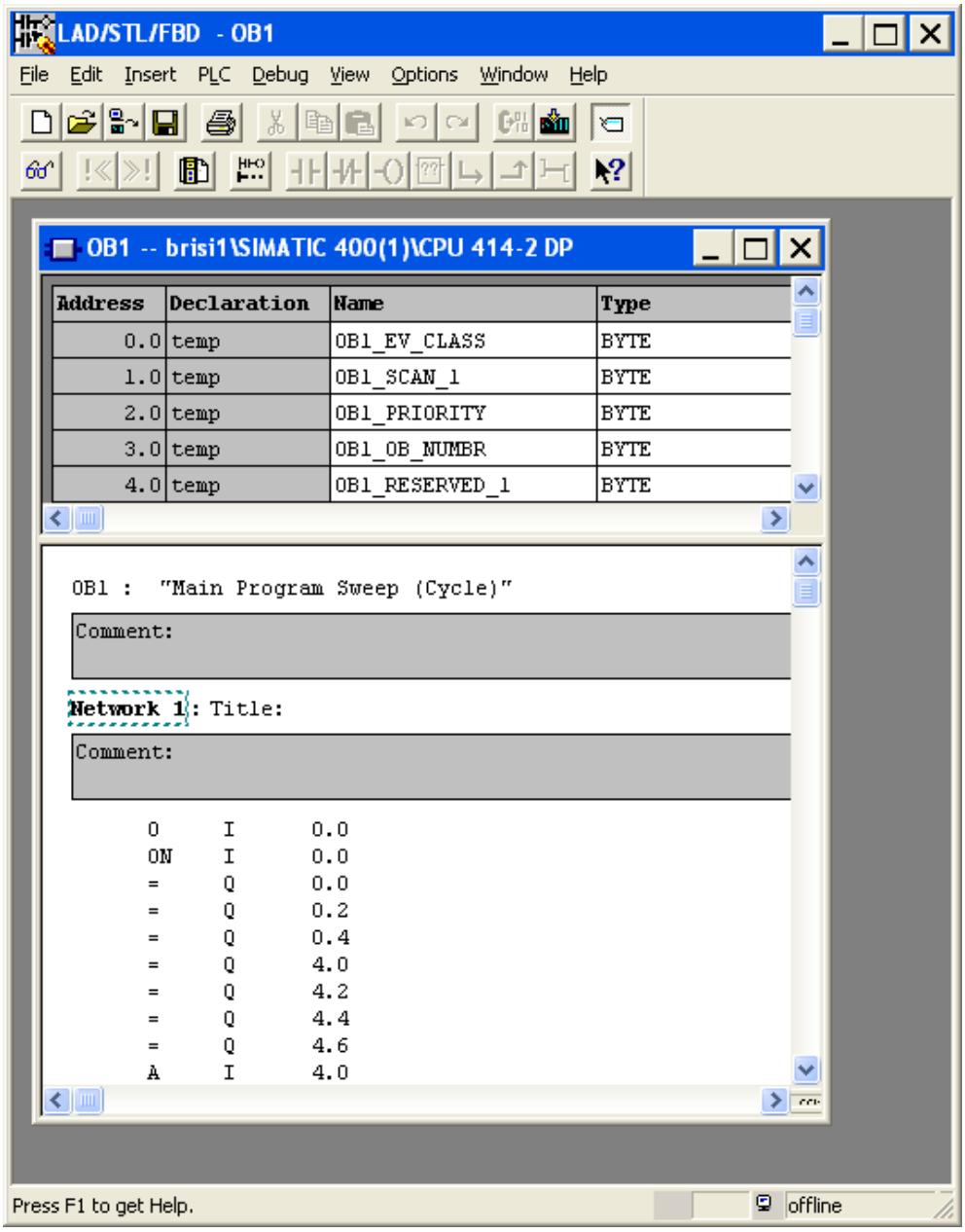

Figure 13.4: STEP 7 LAD/FBD/STL programming window

Other programing languages are available as optional packages.

13.6 Hardware diagnostics

These functions provide the user with an overview of the status of the programmable controller. An overview can display symbols to show whether every module has a fault or not. A double click on the faulty module displays detailed information about the fault. The scope of this information depends on the individual module.

Part VI – Creating STEP 7 projects for different PLC configurations

14. STEP 7 Projects

Projects represent the sum of all data and programs within the scope of an automation task. They are used to store the data and programs in an organized manner. The data collected together in a project include the following:

- configuration data on the hardware structure and parameters for modules,
- configuration data for communication networks,
- programs for programmable modules.

The main tasks involved in creating a project are, therefore, preparing the above data, and creating the programs. STEP 7 does not require that the components of a project are edited in a particular order.

Data are stored in a project in object form. The objects (Project, Station, ...) used in SIMATIC Manager have the following functions:

- carriers of object properties,
- folders,
- carriers of functions (for example, to start a particular application).

There are the following objects for configuring hardware and networks in a project:

- Network: The icons for networks appear when you select the project icon. We use network icon for starting the tool for network configuration and setting the network properties. The network icons represent the information about a network of the given type, and are used to set the network parameters. We can delete any network icons we do not require, and create them again, if necessary, using the menu command **Insert>Subnet**.
- Station: The icon for a station represents a hardware configuration. Stations are both: objects (project level) and object folder (station level). If we select a station in the left-hand half of the project window, we will see Hardware object in the right half of the window.
- Hardware: Hardware is a basic procedure for configuring hardware. Using Hardware object, we can create Programmable modules.
- Programmable Module: A Programmable module represents the parameters assignment data of a programmable module (CPUxxx, FMxxx, CPxxx). Programmable modules are both objects (station level) and object folders (Programmable modules level).

The objects in a project are arranged in a tree structure. The display of the hierarchy in the project window is similar to that of the Windows. Only the object icons have a different appearance. The top end of the project hierarchy is structured as follows:

- 1st level: Project,
- 2nd level: Subnets, stations, or S7 programs,
- 3rd level: depends on the object in level 2.

15. Using STEP 7

15.1 Using STEP 7 – for PLC configuration no. 2 (Part II, Chapter 7.8.4)

15.1.1 Starting the SIMATIC Manager and creating projects

The SIMATIC Manager is the central window which becomes active when STEP 7 is started. The default setting starts the STEP 7 Wizard, which supports the user when creating a STEP 7 project.

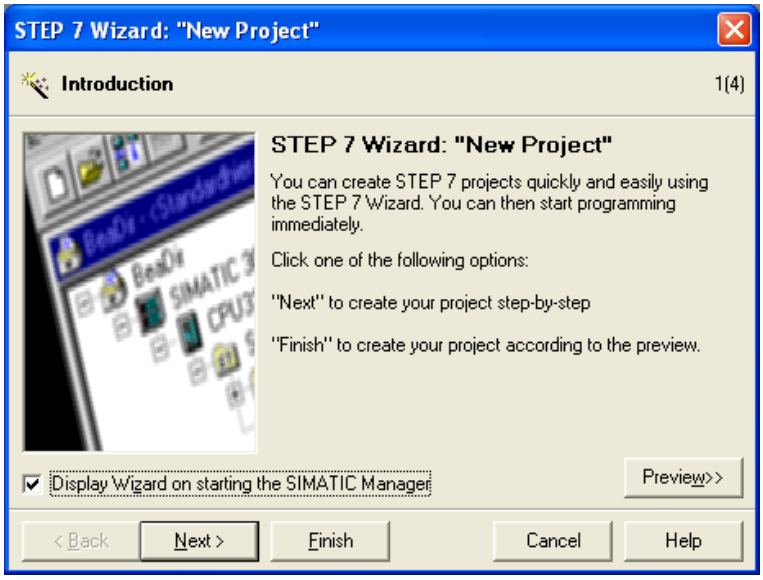

Figure 15.1: SIMATIC Manager start window with the STEP 7 wizard

We will write our project without wizard, therefore, we will close the STEP 7 Wizard window by pressing the Cancel button.
As soon as the STEP 7 Wizard window is closed, the SIMATIC Manager appears with the previous open projects. Because we will write a new project, we close the old projects by pressing x (Close) button (if old projects already exist).
After closing all the project windows the empty SIMATIC Manager window remains open, which is presented in Figure 15.2.

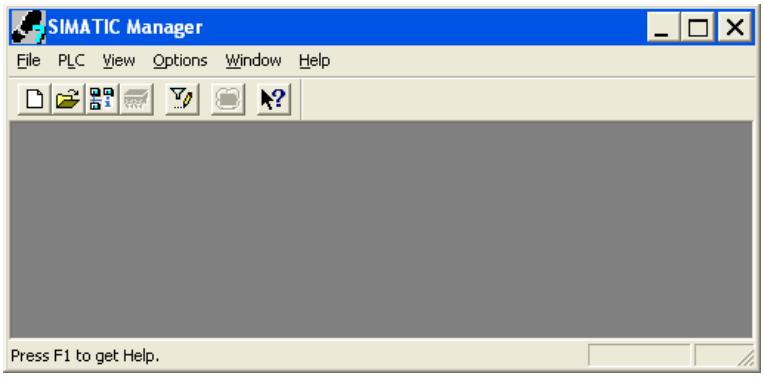

Figure 15.2: SIMATIC Manager main window after closing all project windows

We will create a new project using the menu bar command: **File>New**.
A New window appears:

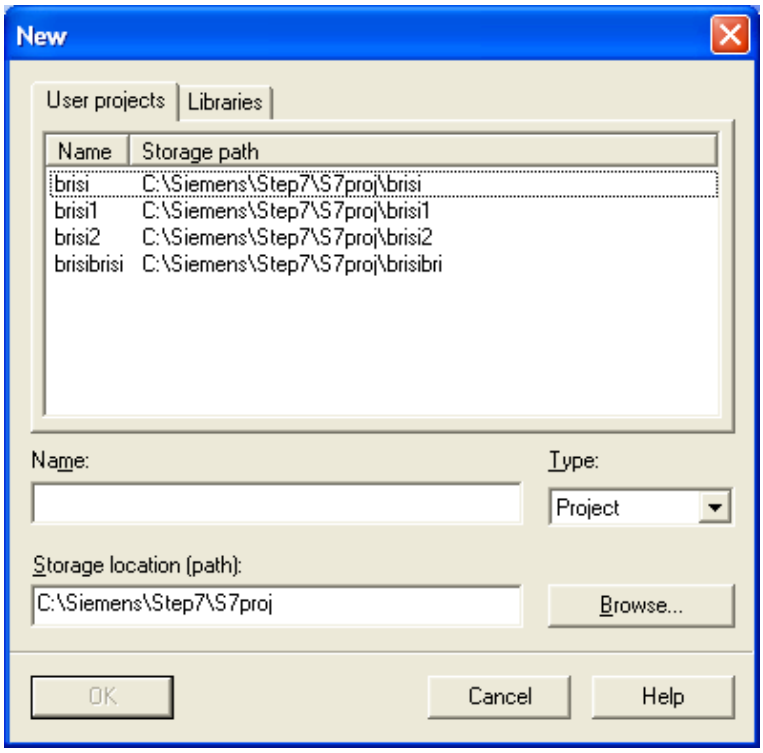

Figure 15.3: New Project and Libraries window

In the new window we fill in the project name. After pressing OK, the SIMATIC Manager window appears with a new created project window (in our case, we entitled the project with the title Bethune).

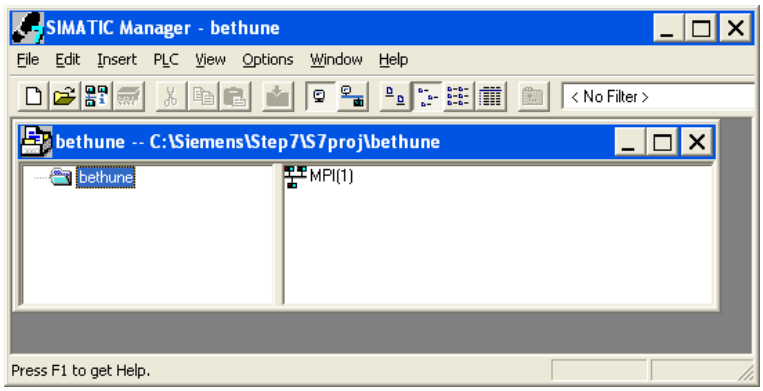

Figure 15.4: Newly created project window

15.1.2 Configuring hardware

We can configure the hardware as soon as we have created a project. In the Menu bar we choose **Insert>Station>SIMATIC 300 Station.**

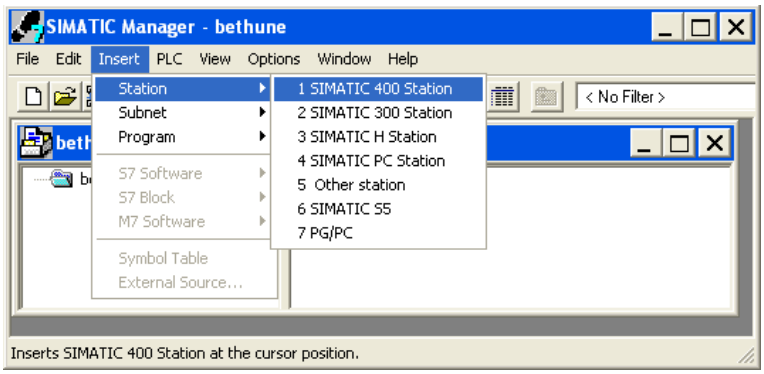

Figure 15.5: Hardware configuration window

In the right pane a new object appears – a station icon entitled SIMATIC 300(1). In the left pane, the project object Bethune changed into an object folder, Bethune, with a subfolder SIMATIC 300(1).

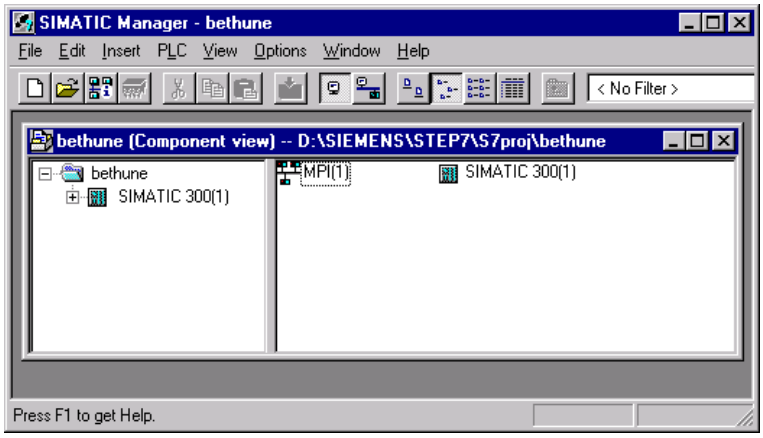

Figure 15.6: Station object SIMATIC 300(1) added to the project

To open Hardware Configurator we click twice on the SIMATIC 300(1) icon in the right pane, and, after that the Hardware icon appears. After double clicking on the Hardware icon a new window opens, HW Config – SIMATIC 300(1).

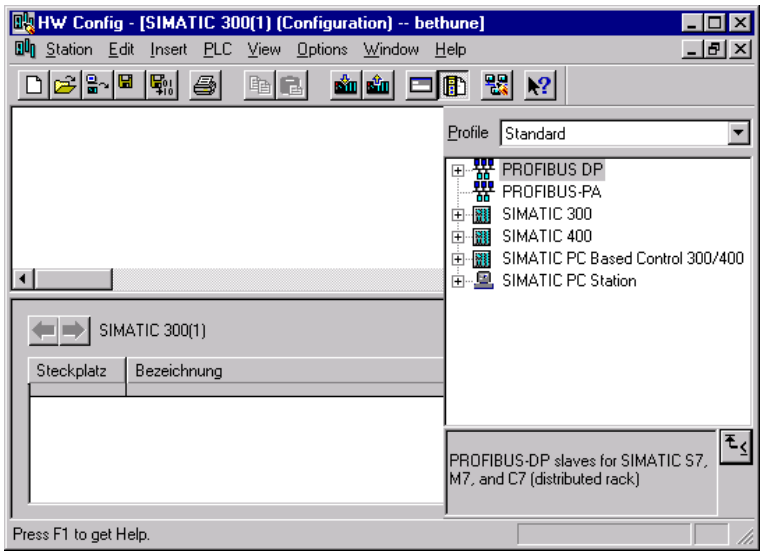

Figure 15.7: Hardware Configuration window for SIMATIC 300(1)

The window is divided into three parts:

- The right up pane contains the hardware catalog of SIMATIC modules.
- The right down pane shows the short information to the selected module.
- The left pane shows a window for a Configuration Table with MPI and I/O addresses.

First, we require a rack module. Navigate in the catalog until we reach the:
SIMATIC 300>RACK 300>Rail>6ES7 390-1???0-0AA0
After double clicking onto the selected rack, a Configuration Table opens in the left pane window.

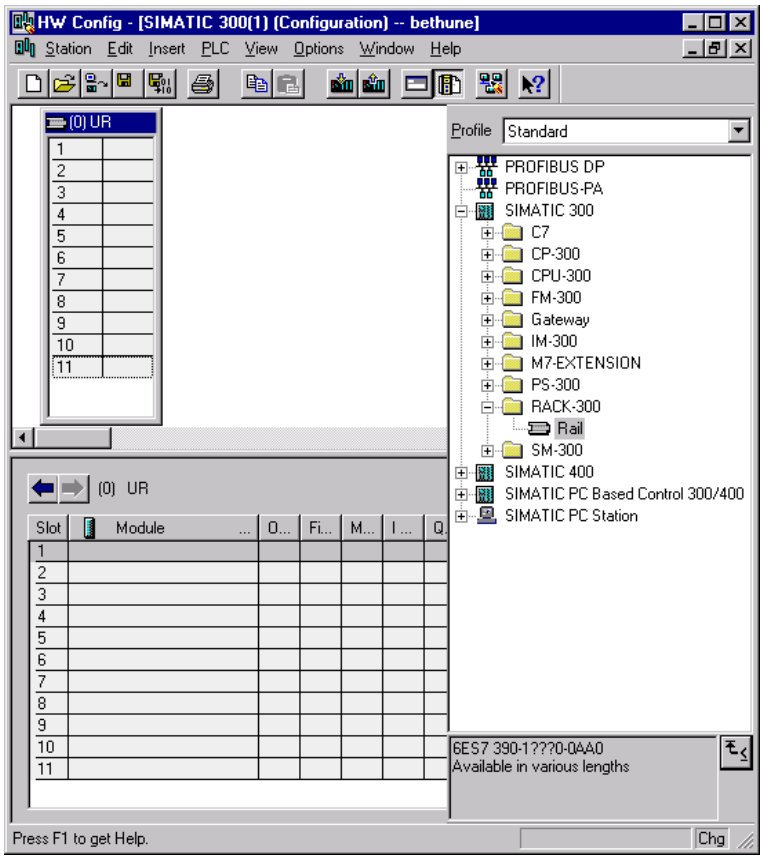

Figure 15.8: Rack module inserted into the Configuration Table

Then, we navigate until we find the power supply module:
SIMATIC 300>PS-300> PS-307 5A> 6ES7 307-1EA00-0AA0
We drag and drop the selected module into slot 1.

In the same way, we insert the CPU into slot 2:
SIMATIC 300>CPU 300>CPU 314 IFM>6ES7 314-5AE02-0AB0
The completed Configuration Table is shown in Figure 15.9.

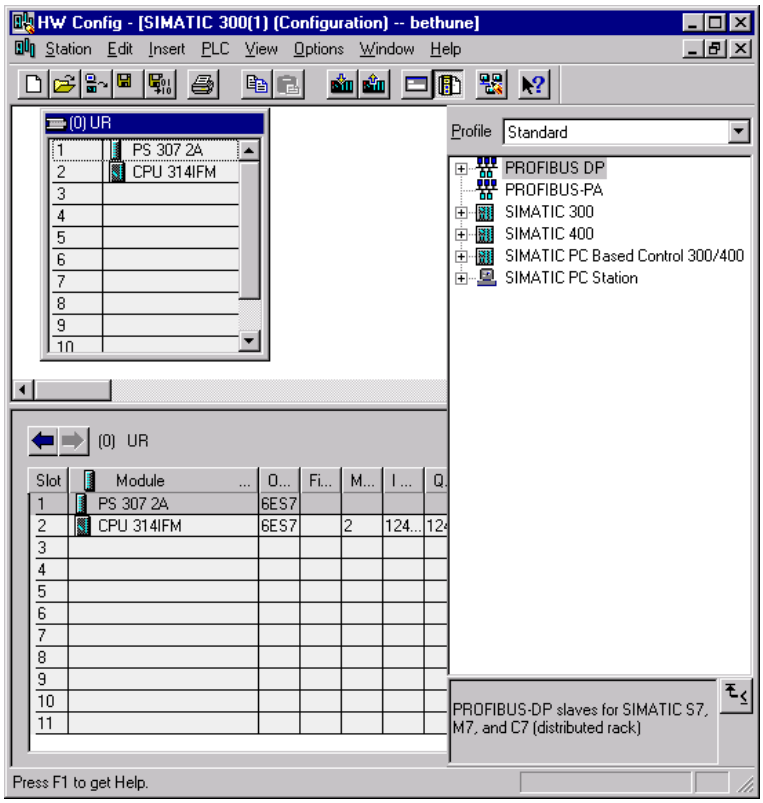

Figure 15.9: Completed Configuration Table (rack, PS, CPU).

In order to change the parameters (for example, address) of a module within a project, double-click the module. However, we should only change the parameters if we are sure we know what effects the changes will have on our programmable controller.

The data are prepared for transfer to the CPU using the menu bar command
Station>Save and Compile

We can also check our configuration for errors using the menu command
Station>Consistency Check
STEP 7 will provide you with possible solutions to any errors which may have occurred.

Finally, we close the HW Config window with the menu command: **Station>Exit**

15.1.3 Creating a Table of Symbols

In the Symbol Table, you assign a symbolic name and the data type to all the absolute addresses which you will address later on in your program; for example, I 124.0 the symbolic name SWITCH1. These names apply to all parts of the program, and are known as global variables.
To use the Table of Symbols you have to:

- Navigate in the project window until you reach **S7 Program (1),** and double-click to open the **Symbols** component.
- Enter SWITCH1 and I 124.0 in the first row.
- Save the entries or changes you have made in the Symbol Table and close the window: **Symbol Table>Save** and **Symbol Table>Exit**.

The Symbol Editor window is shown in Figure 15.10.

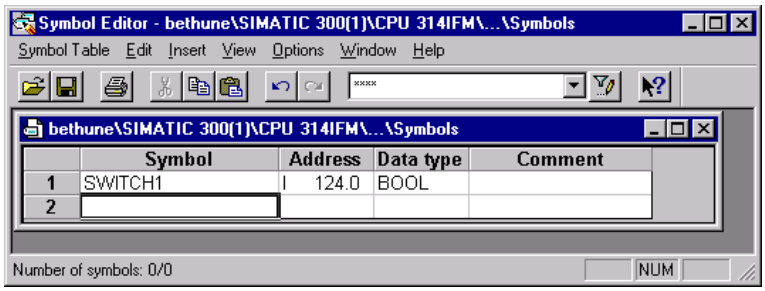

Figure 15.10: Symbol Editor window with entered symbol

15.1.4 Creating a Program

Navigate in the project window until you reach the **Block** folder. Double-click on it, in the right pane two objects appear: System data and OB1. In STEP 7, OB1 is processed cyclically by the CPU. The CPU reads line by line and executes the program commands. Double-clicking OB1 opens the LAD/STL/FBD – OB1 window, where we write the program.

For example:

```
A     I     SWITCH1
A     I     124.1
=     Q     124.0
```

The LAD/STL/FBD – OB1 Editor window is shown in Figure 15.11.

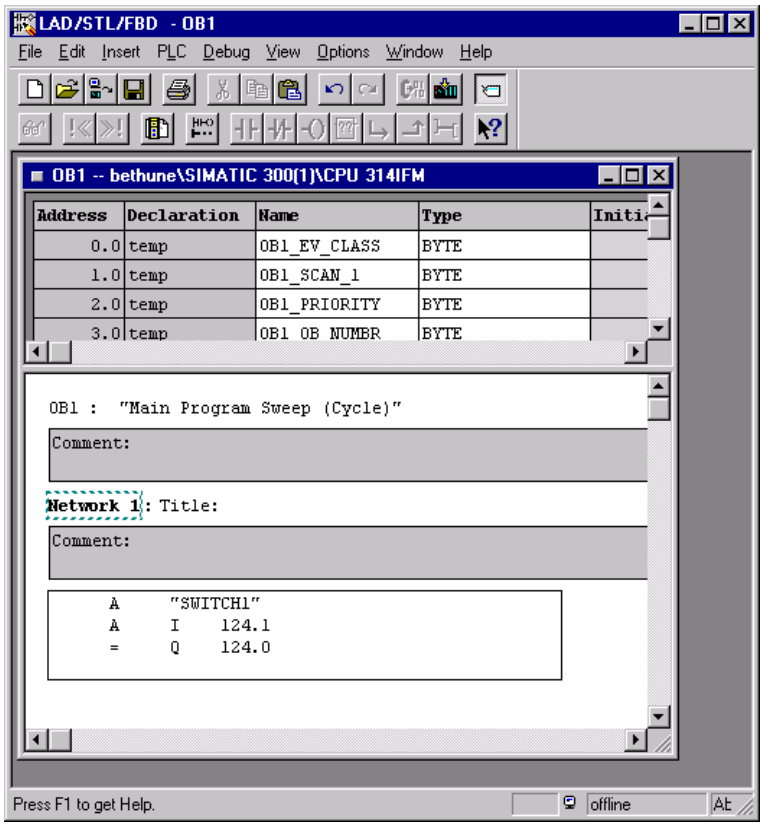

Figure 15.11: LAD/STL/FBD – OB1 Editor window

Save the block and exit the window: **File>Save** and **File>Exit**

15.1.5 Downloading and debugging the program

You must already have established an online connection in order to download the program.

- Switch on the power supply of the CPU using the ON/OFF switch. The diode »DC 5V« will light up on the CPU.
- Turn the operating mode switch to the STOP position (if not already in STOP). The red STOP light will light up.
- In the SIMATIC Manager select object SIMATIC 300(1) and execute the menu command **PLC>Download**.
- Turn the operating mode switch to **RUN-P**.

15.2 Using STEP 7 – for PLC configuration 5 (Part II, Chapter 7.8.7)

15.2.1 Starting the SIMATIC Manager and creating projects

The SIMATIC Manager is the central window which becomes active when STEP 7 is started. The default setting starts the STEP 7 Wizard, which supports the user when creating a STEP 7 project.

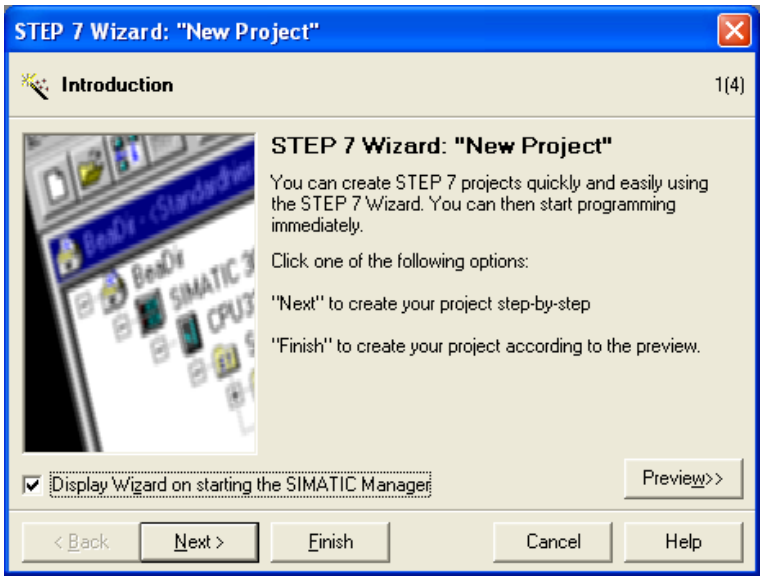

Figure 15.12: STEP 7 Wizard for creating a New Project

We will write our project without wizard, therefore, we will close the STEP 7 Wizard window with pressing the Cancel button.
As soon as the STEP 7 Wizard window is closed, the SIMATIC Manager appears with the previous open projects. Because we will write a new project, we close the old projects by pressing the x (Close) button (if old projects already exist).

After closing all project windows the empty SIMATIC Manager window remains open, which is presented in Figure 15.13.

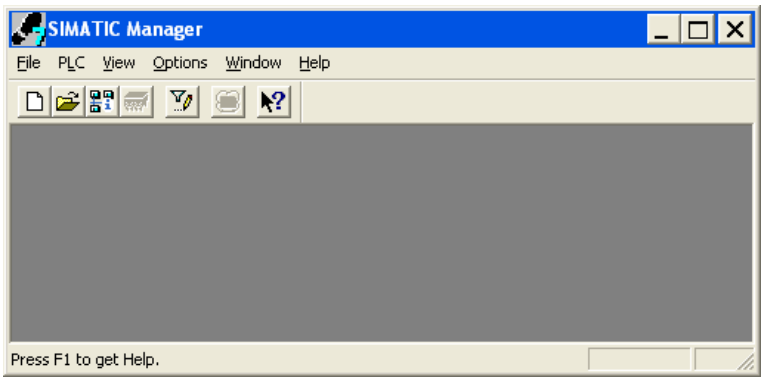

Figure 15.13: SIMATIC Manager window after all project windows have been closed

We will create a new project and libraries using the menu bar command: **File>New**.
A New window appears:

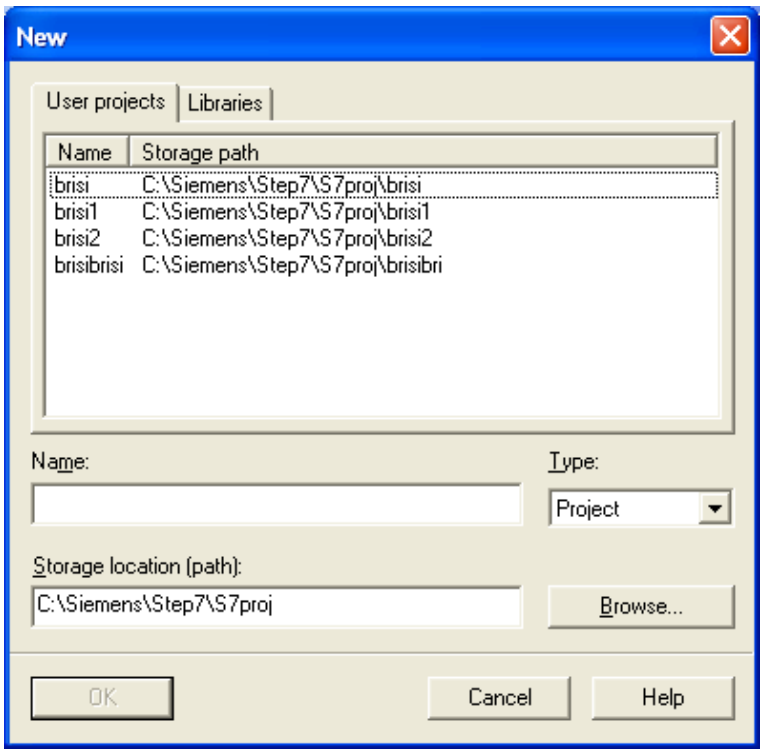

Figure 15.14: New project and library window

In the New window we fill in the project name. After pressing OK, the SIMATIC Manager window appears with a new created project window (in our case, we entitled the project with the title Bethune).

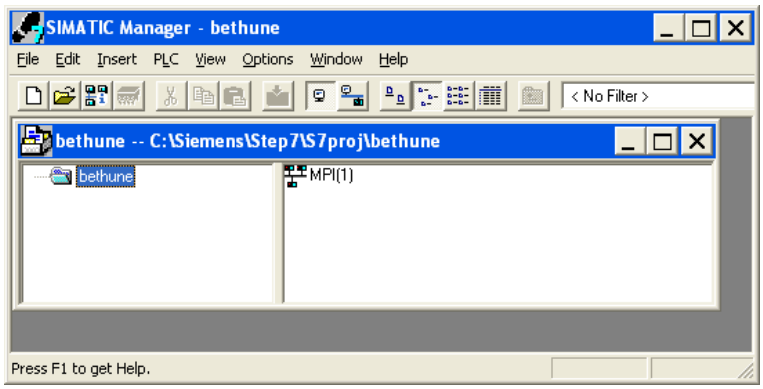

Figure 15.15: New project window

15.2.2 Configuring hardware

We can configure the hardware as soon as we have created a project. In the Menu bar we choose **Insert>Station>SIMATIC 400 Station.**

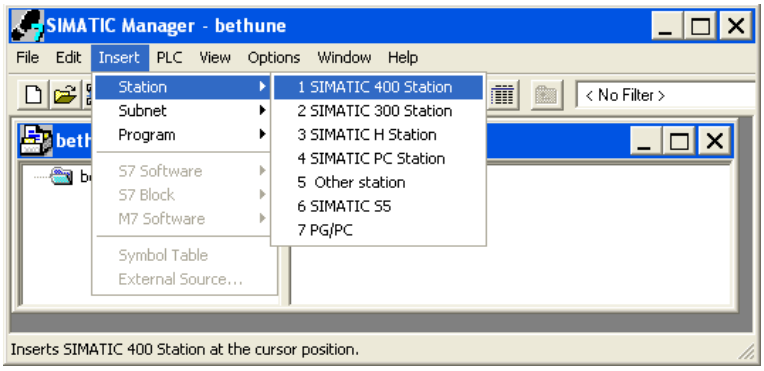

Figure 15.16: Hardware Configuration window

In the right pane a new object appears – a station icon entitled SIMATIC 400(1). In the left pane project the object Bethune changed into an object folder, Bethune, with a subfolder SIMATIC 400(1).

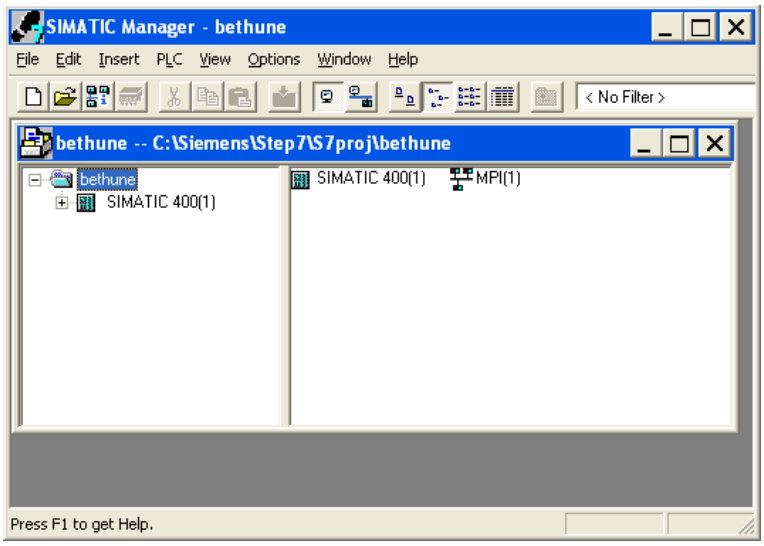

Figure 15.17: A new SIMATIC 400(1) station appears in the project tree

Then, we create a network. In the Menu bar we choose **Insert>Subnet>PROFIBUS.** In the right pane a new object appears – a network icon entitled PROFIBUS (1).
To open the Hardware Configurator, we click twice on the SIMATIC 400(1) icon in the right pane, and the Hardware icon appears after that. After double clicking on the Hardware icon a new window opens, HW Config – SIMATIC 400(1).

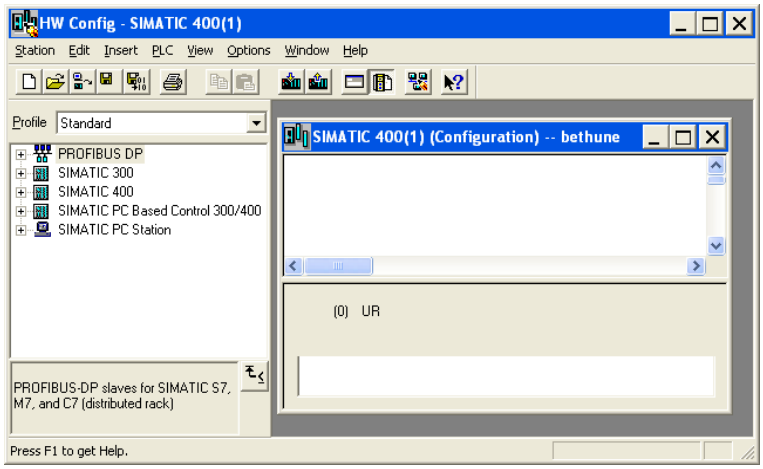

Figure 15.18: Opening the HW Config window for SIMATIC 400(1) station

The window is divided into three parts:

- The left upper pane contains the hardware catalog of the SIMATIC modules.
- The left down pane shows the short information to the selected module.
- The right pane shows a window for the Configuration Table with MPI and I/O addresses.

First, we require a rack module. Navigate in the catalog until we reach the:
SIMATIC 400>RACK 400>UR2>6ES7 400-1JA01-0AA0
After double clicking onto selected rack a Configuration Table opens in the right pane window.

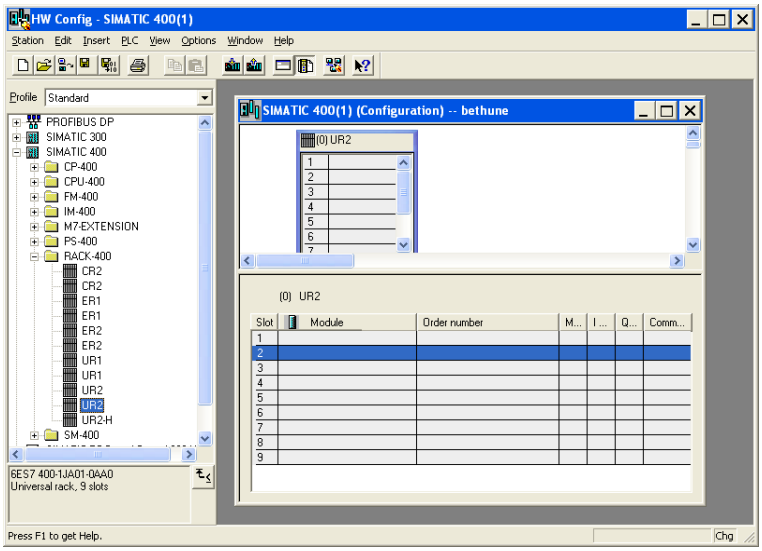

Figure 15.19: Rack and module configuration selection

Then, we navigate until we find the power supply module:
SIMATIC 400>PS-400>Standard PS-400>PS 407 4A>6ES7 407-0DA00-0AA0
We drag and drop the selected module into slot 1.

In the same way, we insert the CPU into slot 2:
SIMATIC 400>CPU 400>CPU 414-2DP>6ES7 414-2XJ00-0AB0
New window: Properties – PROFIBUS interface DP master, opens:

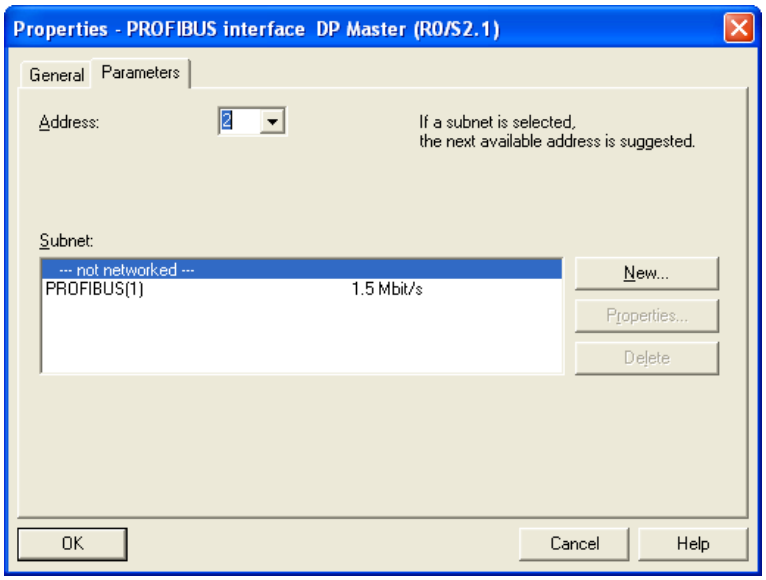

Figure 15.20: PROFIBUS DP Master interface properties

We select Address 2 and close the window with OK.

In the same way, we also insert other modules:

- analog input: SIMATIC 400>SM-400>AI8x13Bit>6ES7 431-1KF00-0AB0>slot 4,
- analog output: SIMATIC 400>SM-400>AO8x13Bit>6ES7 432-1HF00-0AB0>slot 5,
- digital input: SIMATIC 400>SM-400>DI32xDC 24V>6ES7 421-1BL00-0AA0>slot 6,

- digital output: SIMATIC 400>SM-400>DO32xDC 24V/0.5A>6ES7 422-1BL00-0AA0 >slot 7.

The completed Configuration Table is shown in Figure 15.21.

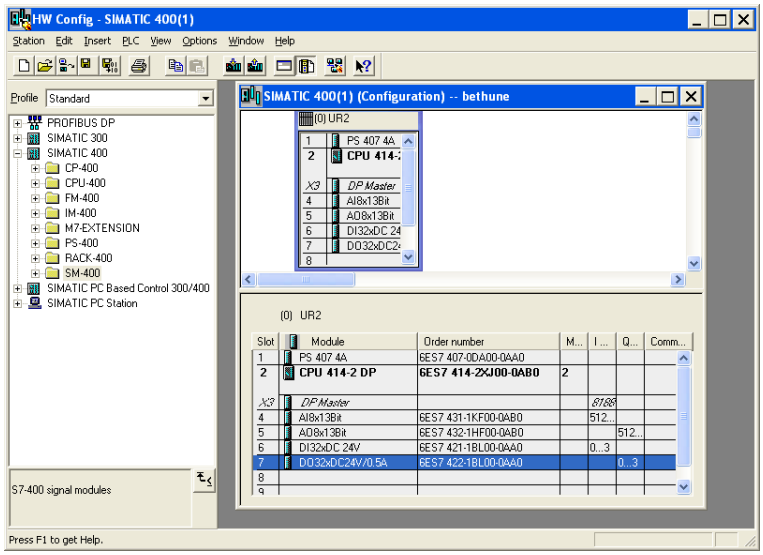

Figure 15.21: Completed HW Configuration Table

In order to change the parameters (for example, the address) of a module within a project, double-click the module. However, we should only change the parameters if we are sure we know what effects the changes will have on our programmable controller.

The data are prepared for transfer to the CPU using the menu bar command
Station>Save and Compile

We can also check our configuration for errors using the menu command
Station>Consistency Check

STEP 7 will provide you with possible solutions to any errors which may have occurred.
Finally, we close the HW Config window with the menu command: **Station>Exit**
The next step is to create a network. Double-clicking on the PROFIBUS (1) icon opens the NetPro window.

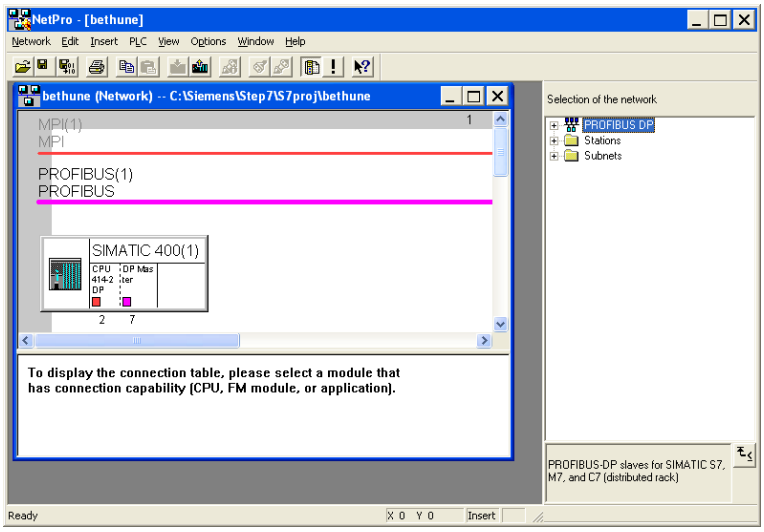

Figure 15.22: NetPro window for PROFIBUS configuration

In right half pane we can find the catalog of network modules, and, in the left pane, there is our network configuration. With the right mouse button we click on the DP Master field of SIMATIC 400(1), and a new window appears. In this window we select **Insert DP master**. In new opened window is:
Properties – PROFIBUS interface DP Master window
where we select **Address: 3** and **Subnet: PROFIBUS (1)**

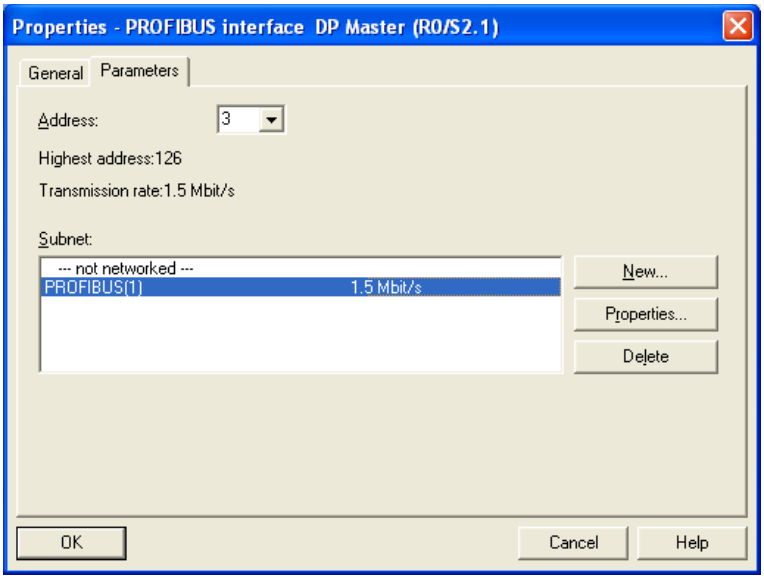

Figure 15.23: PROFIBUS DP Master interface properties

Then, close the window with OK.
SIMATIC 400(1) is now connected to the PROFIBUS network as a DP master.

The next step is to create the PROFIBUS slave. For this we have to navigate until we find the available bus interface module in the catalog:
PROFIBUS DP>ET 200M> IM 153-1>6ES7 153-1AA03-0XB0

Double-clicking on this module opens

Properties – PROFIBUS interface IM 153-1 window
where we select **Address: 4**. Then we close the window.
Before double-clicking on ... >IM 153> ... the DP Master of SIMATIC 400(1) must be selected!
The NetPro window with the network configuration is presented in Figure 15.24.

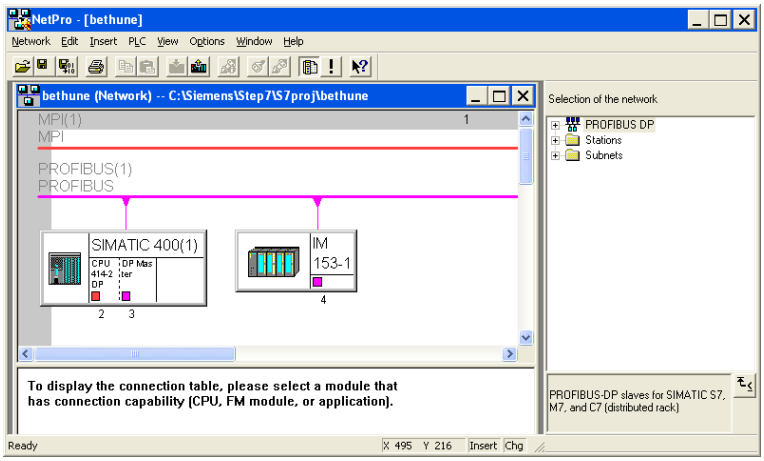

Figure 15.24: NetPro window with PROFIBUS slave configuration

The next step is to configure the bus interface module IM 153-1. We save the current configuration with the menu command **Network>Save,** and then double click on IM 153-1. The window **HW Config – Simatic 400(1)** opens. In the catalog we choose the available module:
SIMATIC 300>SM 300>DI/DO-300>SM 323 DI8/DO8xDC24V/0.5A>6ES7 323-1BH01-0AA0
and put it into slot 4. Then, we save the configuration:

- in the **HW Config – Simatic 400(1)** window with the menu command **Station>Save and Compile,** and then **Station>Exit**,
- **in the NetPro** window with the menu command **Network>Save and Compile,** and then **Network>Exit**.

15.2.3 Creating a Table of Symbols

In the Symbol Table, you assign a symbolic name and the data type to all the absolute addresses which you will address later on in your program; for example, I 4.0 the symbolic name SWITCH1. These names apply to all parts of the program, and are known as global variables.
To use the Table of Symbols you have to:

- Navigate in the project window until you reach **S7 Program (1),** and double-click to open the **Symbols** component.
- Enter SWITCH1 and I 4.0 in the first row.
- Save the entries or changes you have made in the Symbol Table and close the window: **Symbol Table>Save** and **Symbol Table>Exit**.

The Symbol Editor window is shown in Figure 15.25.

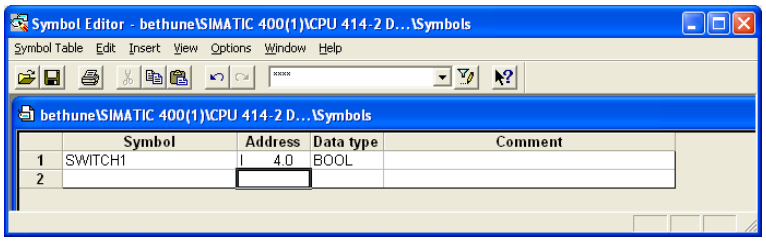

Figure 15.25: Symbol Editor window showing entered symbol

15.2.4 Creating a Program

Navigate in the project window until you reach the **Block** folder. Double-click on it, two objects appear in the right pane: System data and OB1. In STEP 7, OB1 is processed cyclically by the CPU. The CPU reads line by line, and executes the program commands. Double-clicking OB1 opens the LAD/STL/FBD – OB1 window where we write a program.

For example:

```
A     I     4.1
A     I     4.2
=     Q     4.0
```

Save the block and exit the window: **File>Save** and **File>Exit**

For using IM 153 we also need OB82. We generate it with the menu command in the Simatic manager window: Insert>S7 Block>Organization block>Name OB82>OK

15.2.5 Downloading and debugging the program

You must have established an online connection already in order to download the program.

- Switch on the power supply of the CPU and IM 153 using the ON/OFF switch. The diode »DC 5V« will light up on the CPU.
- Turn the operating mode switch to the STOP position (if not already in STOP). The red STOP light will light up.
- In SIMATIC Manager select object SIMATIC 400(1) and execute the menu command **PLC>Download**.
- Turn the operating mode switch to **RUN-P**.

15.3 Using STEP 7 – for PLC configuration 6 (Part II, Chapter 7.8.8)

Before we start to create a project for the PLC configuration 6 (introduced in Chapter II 1.8.8) it is reasonable to check:

- If all necessary hardware modules (i.e., all the modules mentioned in Chapter II 1.8.8) are available.
- If all the necessary hardware modules are connected properly.
- After that, we have to connect four modules to a 230 VAC power source: power supplies PS 407 and PS 307, a power supply for the AS network and the Ethernet hub for

connection of the Ethernet cables from the Ethernet communication processor and computer with installed STEP 7 (altogether there are four 230 VAC cables needed).

- Then we switch on the power supplies PS 407 and PS 307.
- Then it is useful to reset CPU 414-3.
- Then we switch the mode selector on CPU 414-3 to the position RUN-P, and the mode selector on the Ethernet communication processor module 443-1 to the position RUN, and.
- Finally, on the ET-200 module (the PROFIBUS DP slave module), we put the dip switches for the bus address to the value 4.

After all this, if everything is correct, the following status and fault LEDs must light:

- on PS 407 green LED DC 5V and green LED DC 24V,
- on CPU 414-3 red LED EXTF and orange LED STOP,
- on CP 443-1 green LED LINK, red LED INTF and orange LED STOP,
- on PS 307 green LED 24 V,
- on ET 200M red LED BF, green LED ON,
- on CP 343-2 green LED PWR, orange LED CM and blinking green LEDs B, 20+ and 10+,
- on SM 331 red LED SF,
- on SM 332 red LED SF.

15.3.1 Starting the SIMATIC Manager and creating projects

The SIMATIC Manager is the central window which becomes active when STEP 7 is started. The default setting starts the STEP 7 Wizard, which supports the user when creating a STEP 7 project.

Figure 15.26: STEP 7 Wizard for creating a New Project

We will write our project without the wizard, therefore, we will close the STEP 7 Wizard window by pressing the Cancel button.

As soon as the STEP 7 Wizard window is closed, the SIMATIC Manager appears with the previous open projects. Because we will write a new project, we close the old projects by pressing the x (Close) button (if old projects already exist).
After closing all the project windows the empty SIMATIC Manager window remains open, which is presented in Figure 15.27.

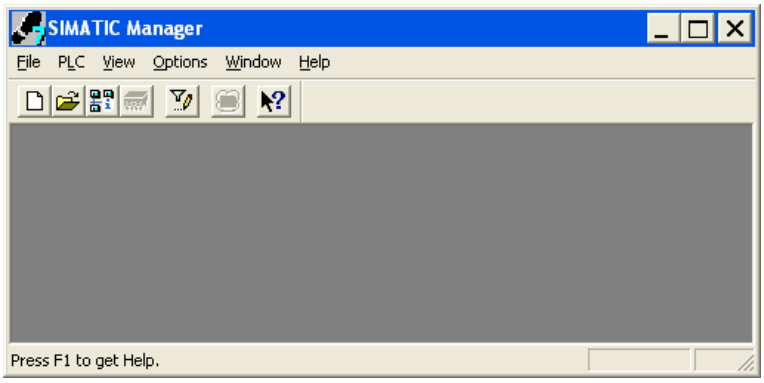

Figure 15.27: SIMATIC Manager main window after closing all project windows

We will create a new project and libraries using the menu bar command: **File>New**.
A New window appears:

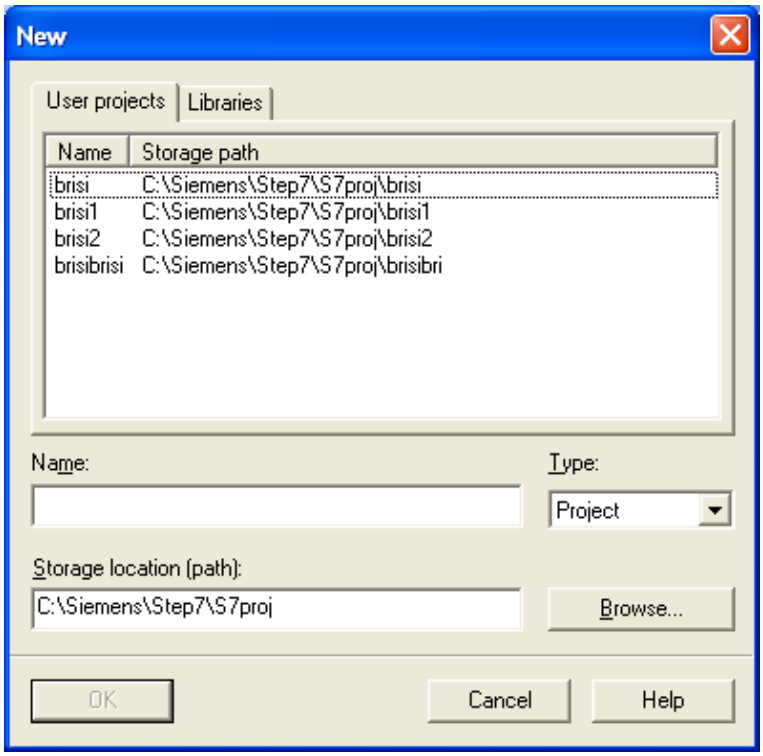

Figure 15.28: New project and libraries window

In New window we fill in the project name. After pressing OK, the SIMATIC Manager window appears with a new created project window (in our case, we entitled the project with the title Bethune).

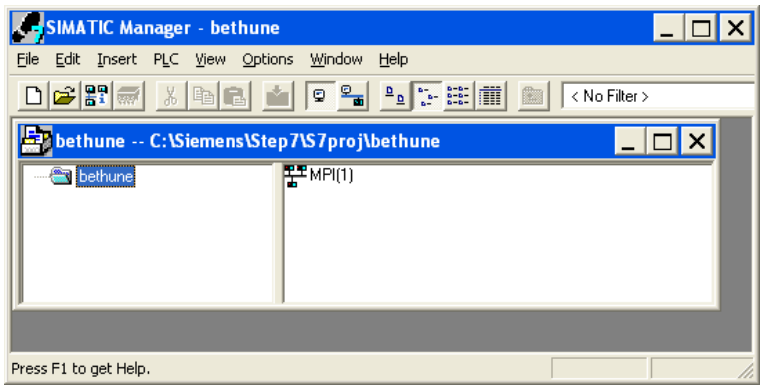

Figure 15.29: Newly created project window

15.3.2 Configuring hardware

We can configure the hardware as soon as we have created a project. In the Menu bar we choose **Insert>Station>SIMATIC 400 Station.**

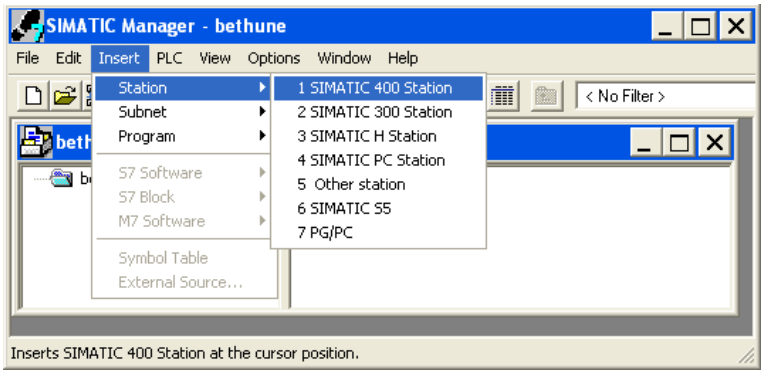

Figure 15.30: Hardware Configuration window

In the right pane a new object appears – a station icon entitled SIMATIC 400(1). In the left pane the project object Bethune changed into an object folder, Bethune, with a subfolder SIMATIC 400(1).

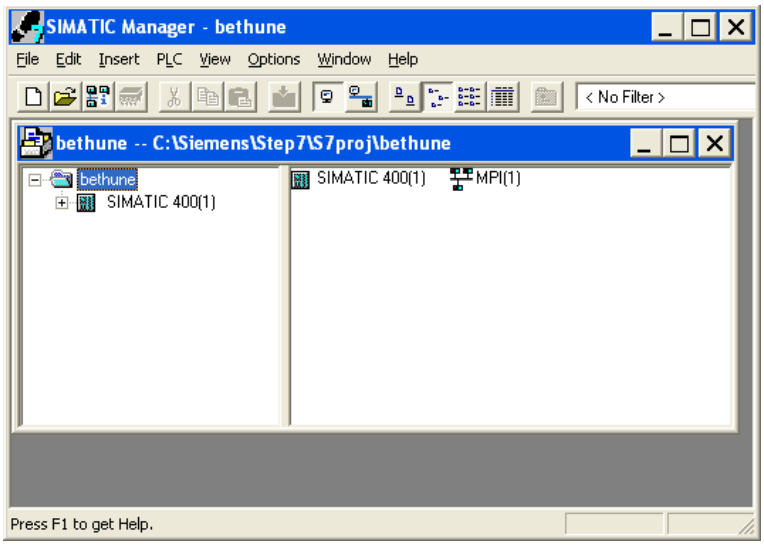

Figure 15.31: SIMATIC 400 (1) station object added to the project

Then, we create a configuration list for the SIMATIC 400(1) station. To do this we have to open the Hardware Configurator. To open the Hardware Configurator, we click twice on the SIMATIC 400(1) icon in the right pane, and, after that, the Hardware icon appears. After double clicking on the Hardware icon a new window opens, HW Config– SIMATIC 400(1).

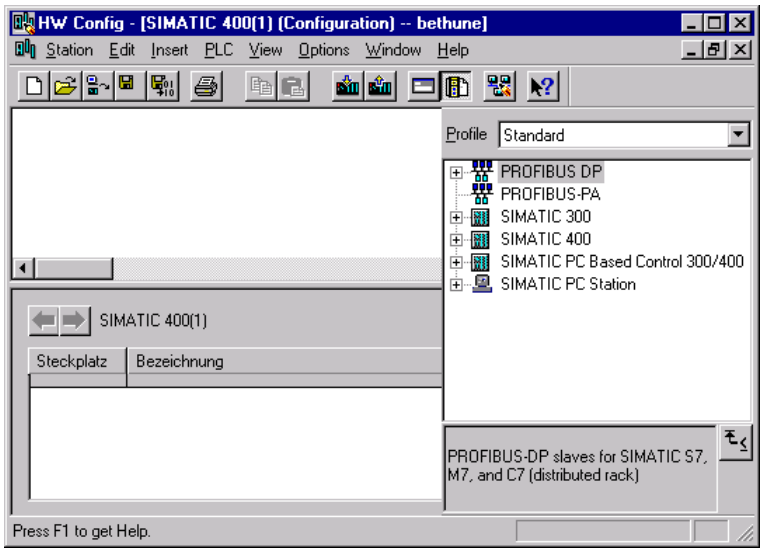

Figure 15.32: HW Config window for SIMATIC 400(1)

The window is divided into three parts:

- The right upper pane contains the hardware catalog of SIMATIC modules.
- The right down pane shows the short information to the selected module.
- The left pane shows the window for the Configuration Table.

First, we require a rack module. Navigate in the catalog until we reach:
SIMATIC 400>RACK 400>UR2>6ES7 400-1JA01-0AA0
After double clicking onto the selected rack, the Configuration Table opens in the left pane window.

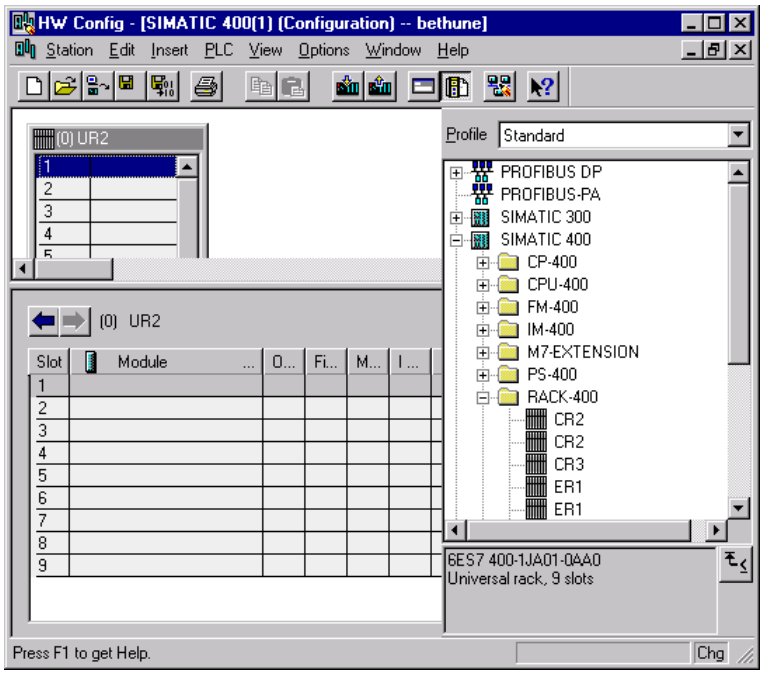

Figure 15.33: Rack module inserted into the Configuration Table

Then, we navigate until we find the power supply module:
SIMATIC 400>PS-400>Standard PS-400>PS 407 10A>6ES7 407-0KA010-0AA0
We drag and drop the selected module into slot 1. The power supply uses 2 slots (1 and 2).

In the same way, we insert the CPU into slot 3:
SIMATIC 400>CPU 400>CPU 414-3 DP>6ES7 414-3XJ00-0AB0>V3.0
A New window: Properties – PROFIBUS interface DP opens:

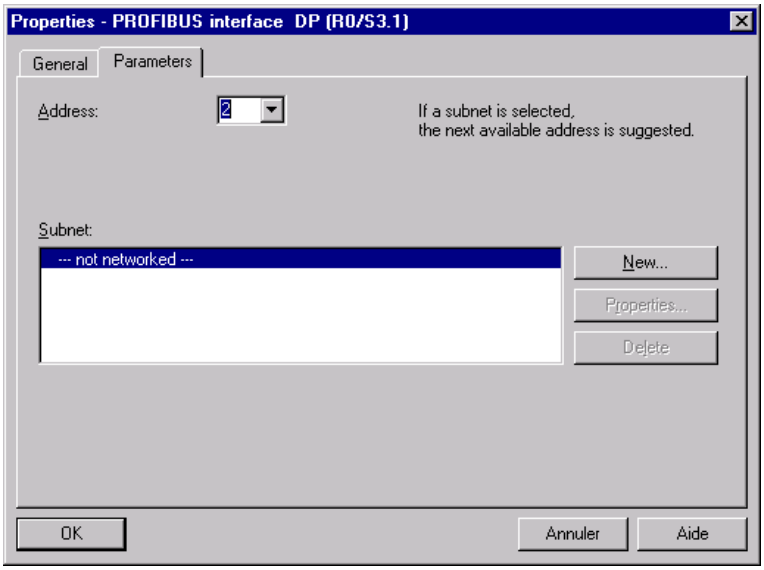

Figure 15.34: PROFIBUS DP interface properties

We select Address 2 and close the window with OK.

In the same way, we insert the Industrial Ethernet communication processor module into slot 5:
SIMATIC 400>CP 400>Industrial Ethernet>CP 443-1>6GK7 443-1EX11-0XE0>V2.0

A New window: Properties – Ethernet interface CP 443-1 opens.
In Figure 15.35:

- we select (mark) the square »Set MAC Address/use ISO protocol«,
- write the MAC address (it is written on the CP 443-1 Ethernet module under the cover), and
- clear (unmark) the field »IP protocol is being used«.

Modified windows: Properties – Ethernet interface CP 443-1 is shown in Figure 15.35.

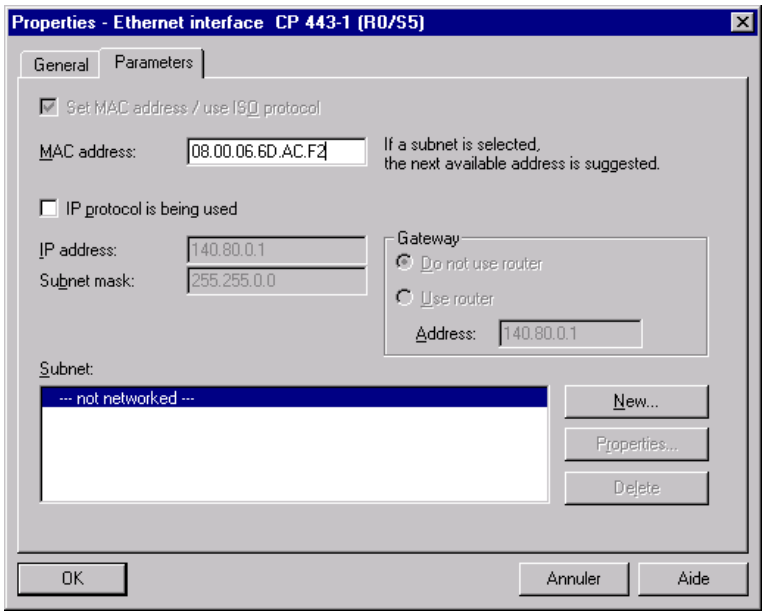

Figure 15.35: Ethernet interface CP 443-1 Properties

Then, with OK, we close the window Properties – Ethernet interface CP 443-1.
The completed Configuration Table is shown in Figure 15.36.

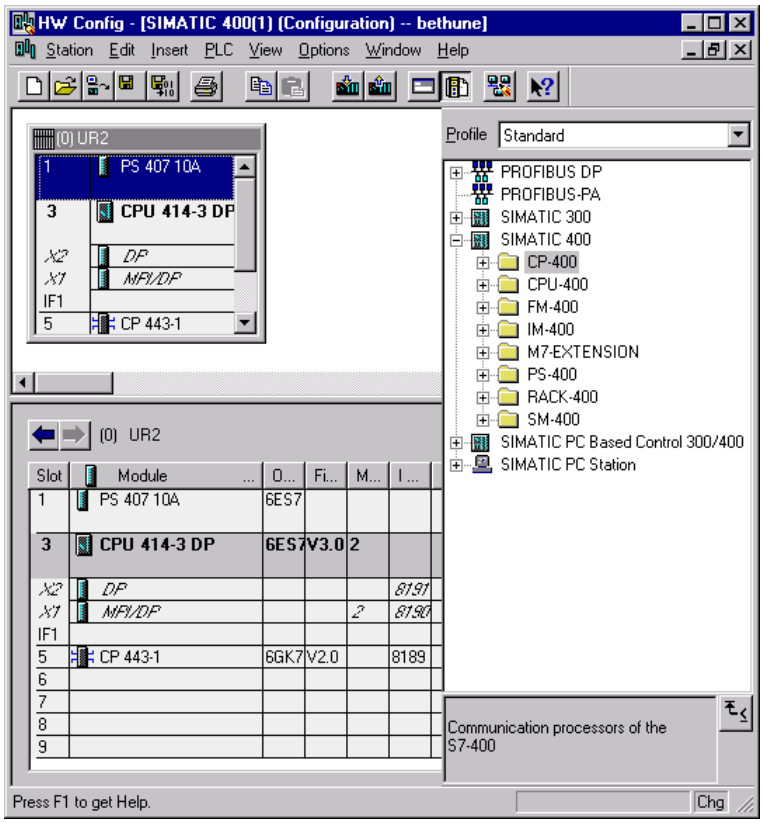

Figure 15.36: Completed HW Configuration Table

In order to change parameters (for example, the address) of a module within a project, double-click the module. However, we should only change the parameters if we are sure we know what effects the changes will have on our programmable controller.

We can check the consistency of the configuration with the bar command
Station>Consistency check
We get a message that the module 5 is not yet assigned to the network

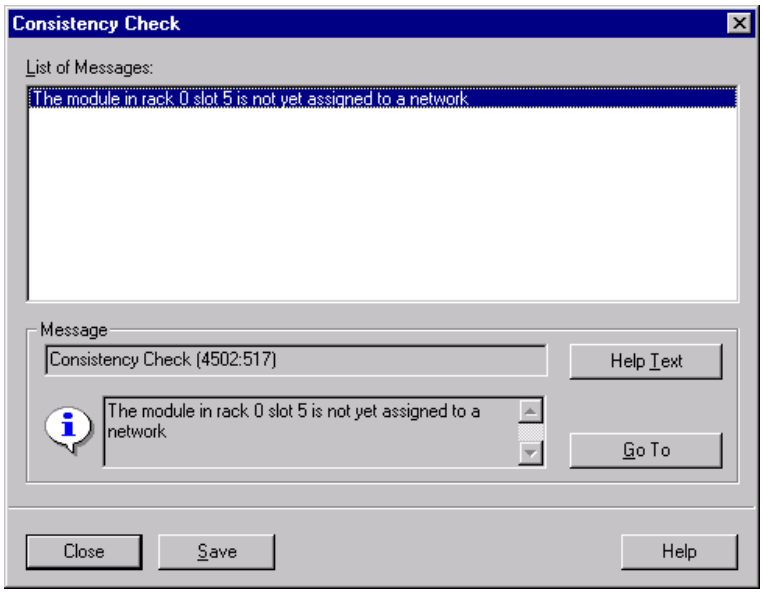

Figure 15.37: Consistency check – module 5 not assigned

STEP 7 will provide you with possible solutions to any errors which may have occurred.

We close the message window with Close.
The data are ready for save and compile – we use the menu bar command
Station>Save and Compile

Finally, we close the HW Config window with the menu command: **Station>Exit**

The next step is to create an Ethernet network. In the left half pane of the window »Bethune« we mark the folder Bethune with the left button on the mouse, and then in the Menu bar we choose **Insert>Subnet>Industrial Ethernet**
Double-clicking on the icon Ethernet (1) in the right half pane window »Bethune« opens the NetPro window.

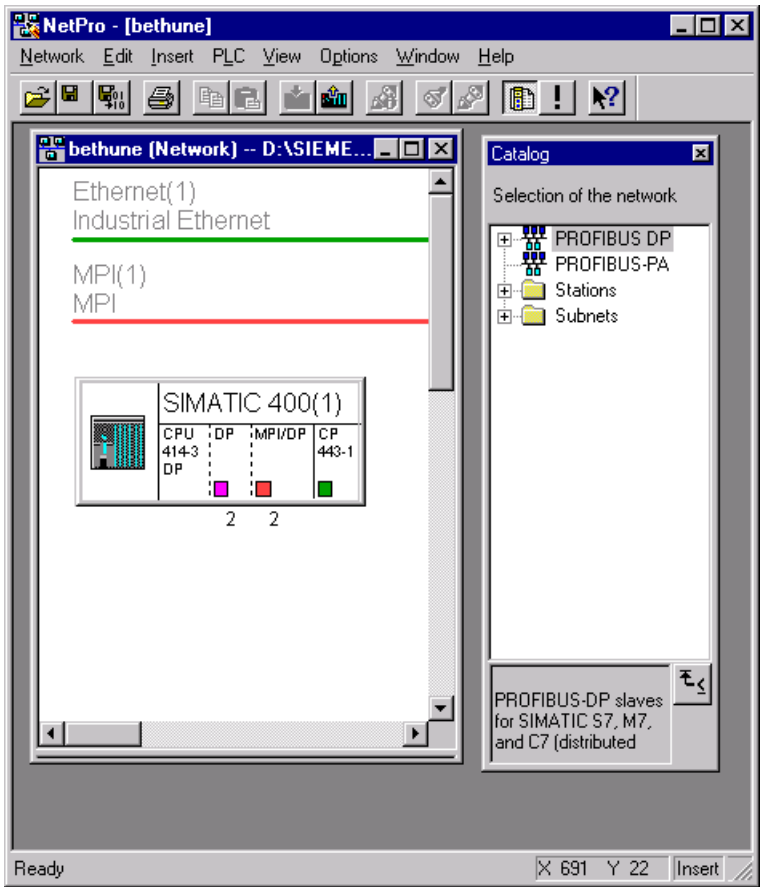

Figure 15.38: NetPro window after creating an Ethernet Network

In the right half pane we can find the catalog of network modules, and in the left pane there is our network configuration. With the right mouse button we click on the CP 443-1 field of SIMATIC 400(1) in the left half pane window, and a new small window appears. In this window we select
Object Properties ... Alt Return.
A new window **Properties – CP 443-1** opens:

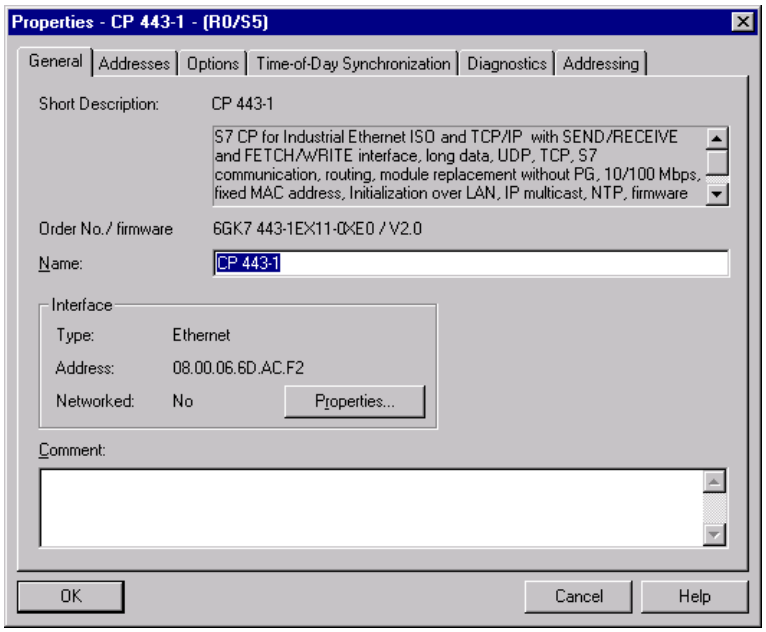

Figure 15.39: Accessing CP 443-1 Properties in NetPro

where we select field **Properties,**
which opens a new window **Properties – Ethernet interface CP 443-1**:

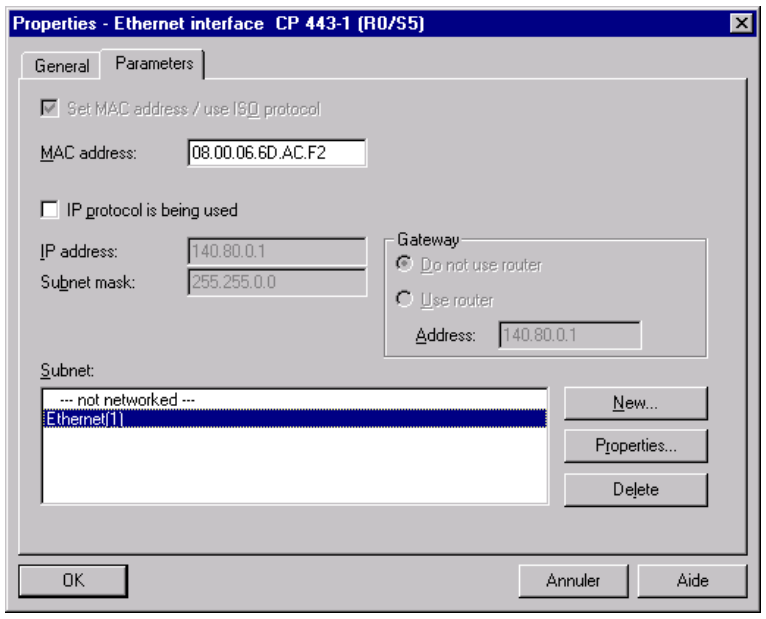

Figure 15.40: Opening the Ethernet Interface Properties for CP 443-1

where we select **Subnet: Ethernet (1).**
After that, we close both windows (window **Properties – Ethernet interface CP 443-1** and window **Properties – CP 443-1, OK** for every window).
Then, we have to change the MPI/DP address (because it is the same as the CPU 414-3 DP address). With the right mouse button we click on the MPI/DP field of SIMATIC 400(1) in the left half pane NetPro window, a new small window appears. In this window we select **Object Properties ... Alt Return**.

A new window **Properties – MPI/DP** opens:

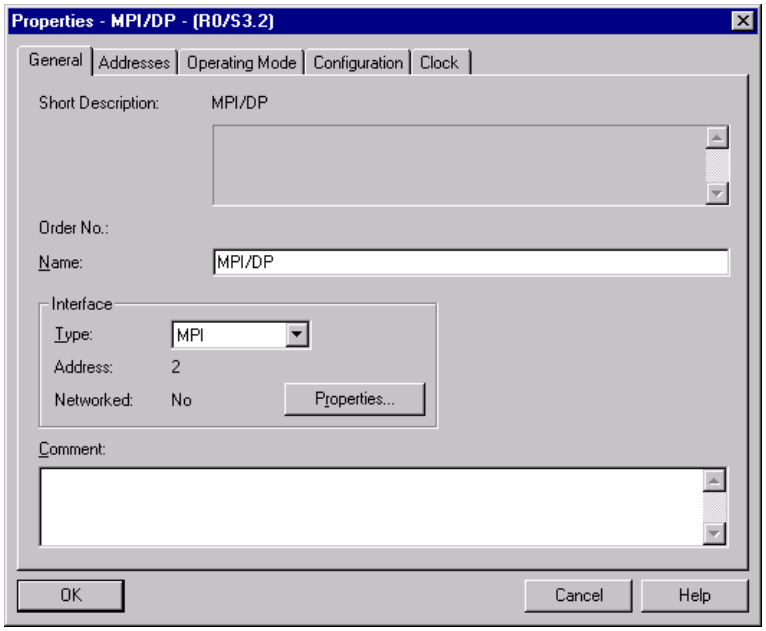

Figure 15.41: Accessing the MPI/DP Interface Properties

where we select the field **Properties**
which opens a new window **Properties – MPI interface MPI/DP**:
where we select Address**: 3** and **Subnet: ... not networked....**

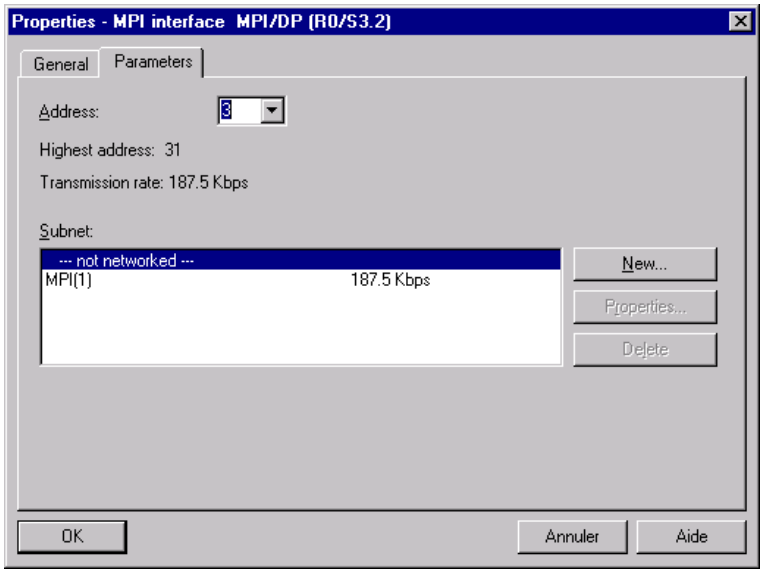

Figure 15.42: Accessing MPI/DP Interface Settings: Address 3 and Non-Networked Subnet

After that we close both windows (window **Properties – MPI interface MPI/DP** and window **Properties – MPI/DP, OK** for every window).
In the NETPRO window we can see that CP 443-1 is connected to the Ethernet (1), and that DP has address 2 and MPI/DP has address 3.

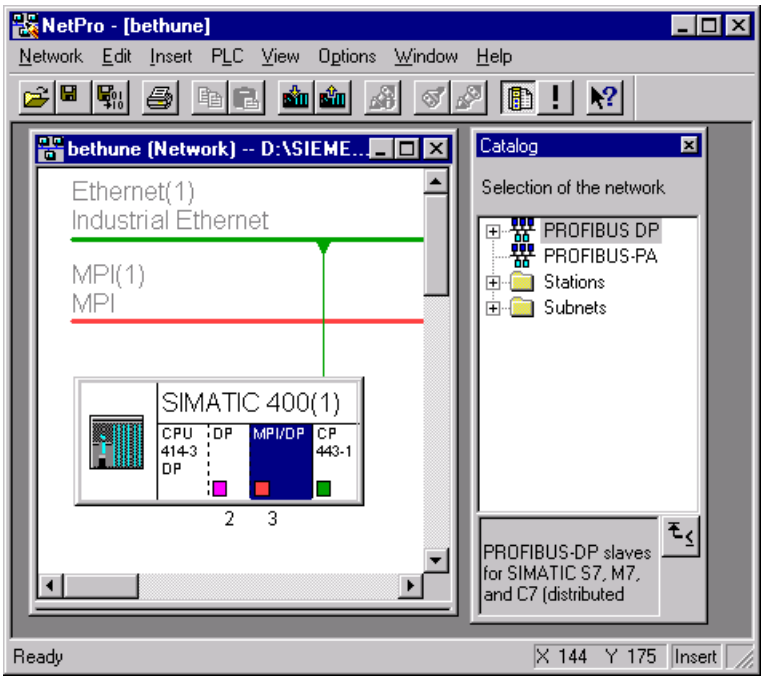

Figure 15.43: Final network view with updated CP 443-1 and MPI/DP settings

We can check the consistency of the configuration with the bar command
Network>Check Consistency
STEP 7 will provide you with possible solutions to any errors which may have occurred.

The data are ready for save and compile – we use the menu bar command
Network>Save and Compile>Compile and check everything

Finally, we close the NetPro window with the menu command: **Network>Exit**

The next step is to create a PROFIBUS DP network. In the left half pane of window »Bethune« we mark the folder Bethune with the left button on the mouse, and then, in the Menu bar, we choose **Insert>Subnet>PROFIBUS**
Double-clicking on the icon PROFIBUS (1) in the right half pane window »Bethune« opens the NetPro window.

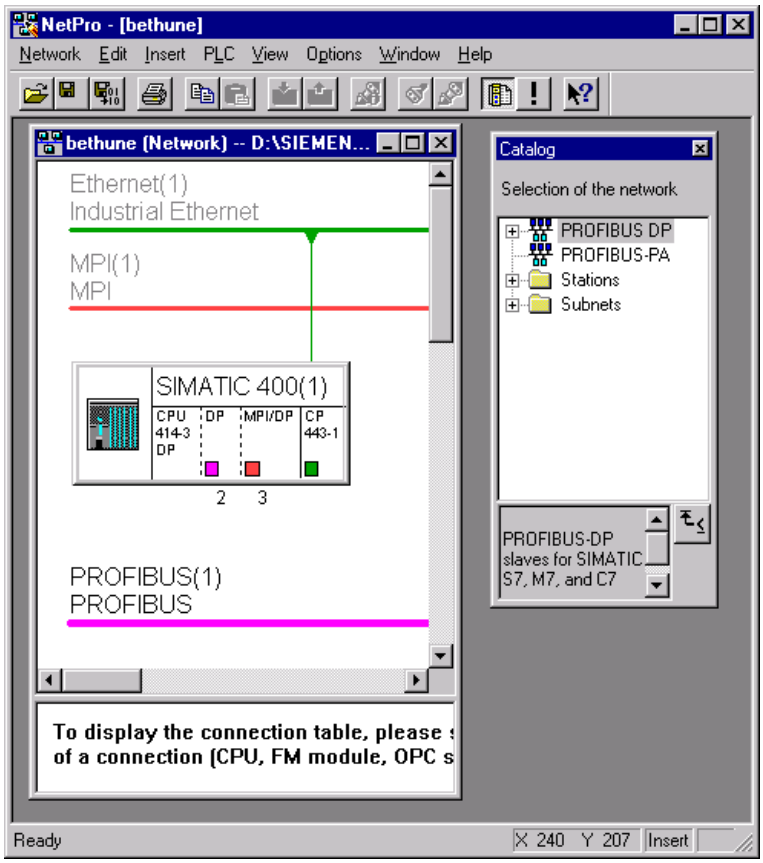

Figure 15.44: Created PROFIBUS DP network

In the right half pane we can find a catalog of network modules and in the left pane there is our network configuration. With the right mouse button we click on the DP field of SIMATIC 400(1) in left half pane window, and a new, small window appears. In this window we select **Object Properties ... Alt Return**.
A new window **Properties – DP** opens:

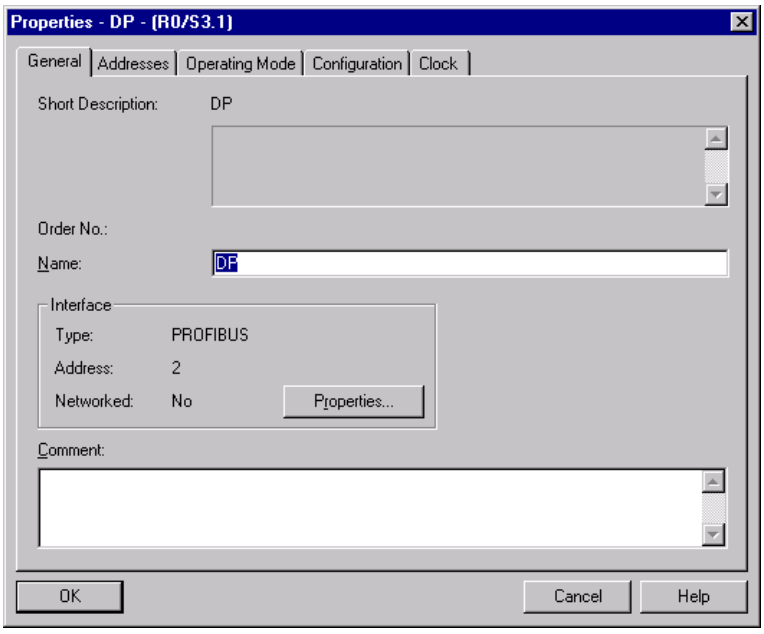

Figure 15.45: Accessing DP interface properties

where we select the field **Properties**
which opens a new window **Properties – PROFIBUS interface DP**:

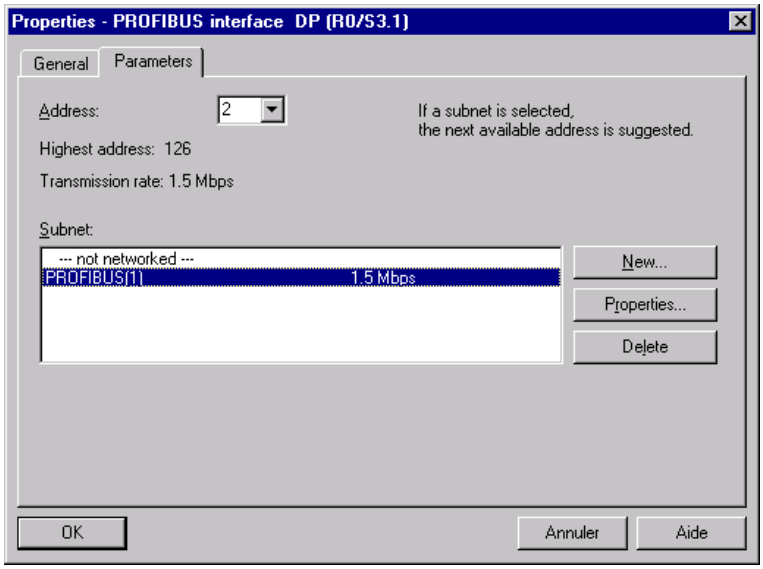

Figure 15.46: PROFIBUS DP interface properties

where we select **Address: 2** and **Subnet: PROFIBUS (1).**
After that, we close both windows (window **Properties – PROFIBUS interface DP** and window **Properties – DP, OK** for every window).
In the NETPRO window we can see that DP is connected to the PROFIBUS (1).

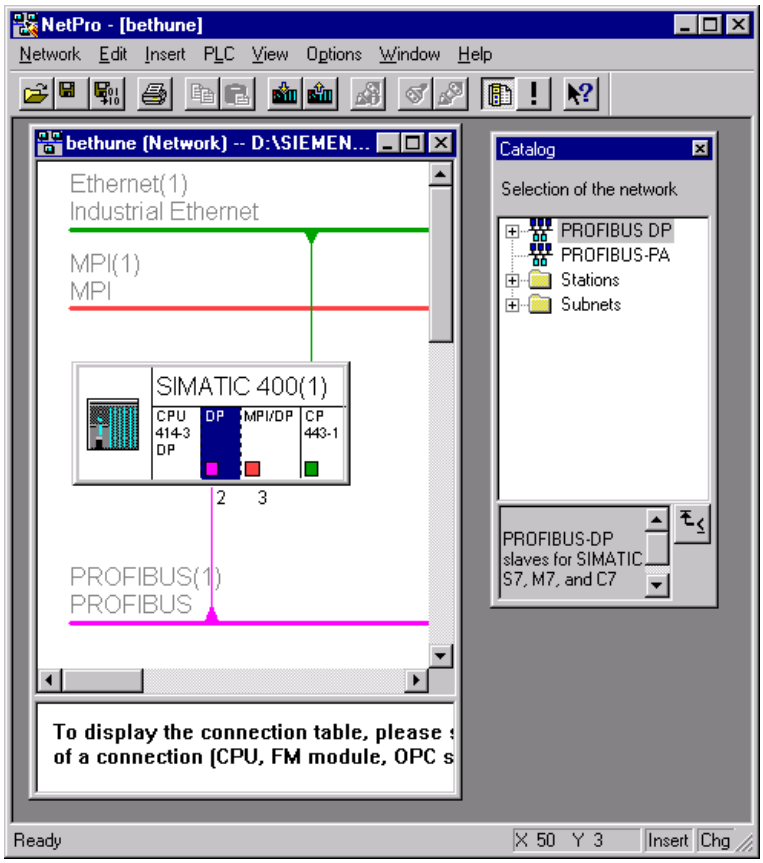

Figure 15.47: NetPro window showing DP connected to PROFIBUS (1)

We can check the consistency of the configuration with the bar command
Network>Check Consistency
STEP 7 will provide you with possible solutions to any errors which may have occurred.

The data are ready for save and compile – we use the menu bar command
Network>Save and Compile>Compile and check everything

Then, we have to add to our configuration a PROFIBUS DP slave module ET 200M. First, we mark the DP field in the SIMATIC 400(1) box in the left half pane of the NetPro window with 1 click to the left button of the mouse (the field must be blue). After that, in the Catalog window in the right half pane of the NetPro window, select the correct module:
PROFIBUS DP>ET 200M>IM 153-2>6ES7-2AA02-0XB0

A new window »PROFIBUS interface IM 153-2« opens. The right address of the module ET 200M must be filled in. We can see this address on the dip switches under the cover of the ET 200M module. In our case, the switches have the value 4. In Figure 15.48 we can see the correct values in the »PROFIBUS interface IM 153-2« window:

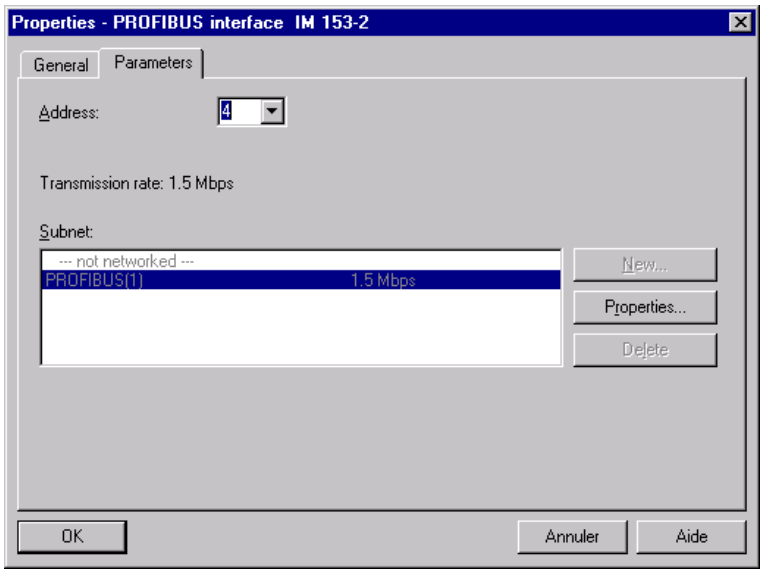

Figure 15.48: IM 153-2 PROFIBUS interface Address Configuration

After that we Close the »PROFIBUS interface IM 153-2« window with OK. In the NETPRO window we can see that IM 153-2 is now connected to the PROFIBUS (1).

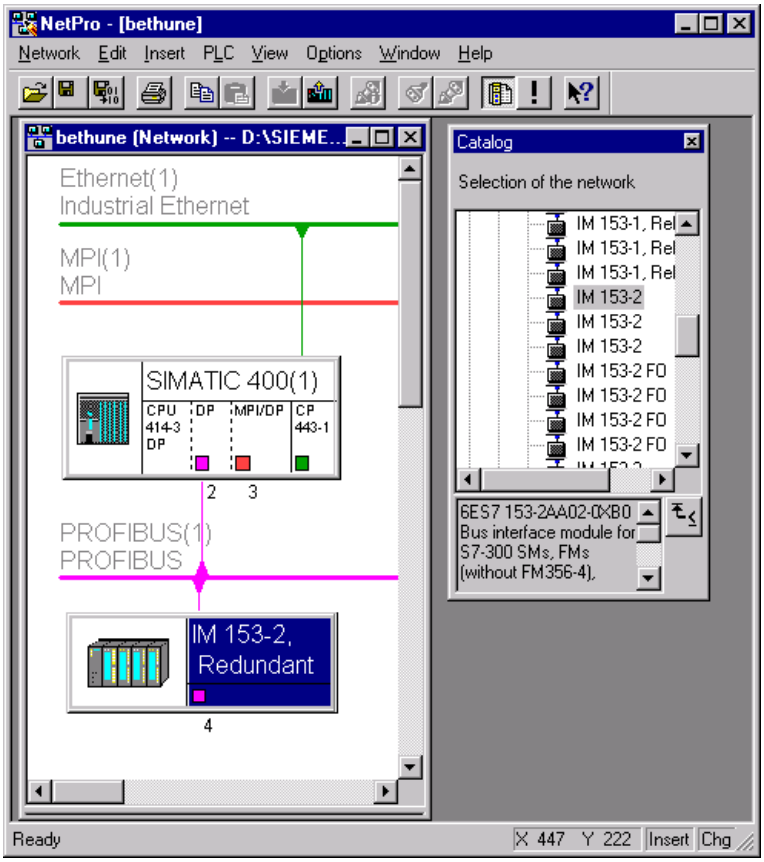

Figure 15.49: NetPro window showing IM 153-2 connected to PROFIBUS (1)

We can check the consistency of the configuration with the bar command **Network>Check Consistency**

We get the message that the DP slave with the PROFIBUS address 4 of the DP master system 1 has to contain at least one module. In our case, 5 modules are connected to the DP slave ET 200M . We also have to configure these additional modules.

The data are ready for save and compile – we use the menu bar command
Network>Save and Compile>Compile and check everything
and close the Error window.

To insert in our configuration also the modules which are connected to ET 200M, we click twice on the IM 153-2 box in the left half pane window of the NetPro window. The Hardware Configurator is started. A new window appears, »HW Config – SIMATIC 400(1) «. With the left mouse bottom we click once on the IM 153-2 box in left upper half window of the »HW Config – SIMATIC 400(1)« window. We get the window shown in Figure 15.50.

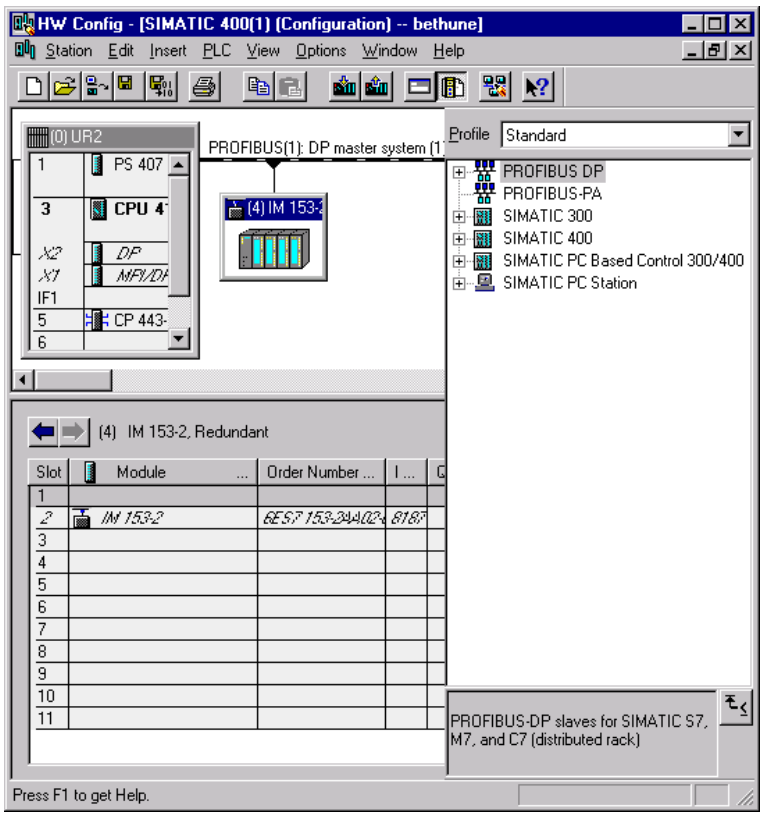

Figure 15.50: Opening HW Config via IM 153-2

Our task is to find the correct modules in the Hardware Catalog in the right half pane of the window, and put them to the right places in the configuration list in the left half pane of the window.
First, we require a CP 343-2 module communication with an AS network. Navigate in the catalog until we reach:
SIMATIC 300>CP-300>AS-Interface>CP 343-2 AS-i >6GK7 343-2AH00-0XA0
We drag and drop the selected module into slot 4.

In the same way, we also insert the other signal modules:
SIMATIC 300>SM 300>AI 300>SM331 AI8x12Bit>6ES7 331-7KF02-0AB0 in slot 6
SIMATIC 300>SM 300>AO 300>SM332 AO4x12Bit>6ES7 332-5HD01-0AB0 in slot 7

SIMATIC 300>SM 300>DI 300>SM321 DI16xDC24V>6ES7 321-1BH02-0AA0 in slot 8
SIMATIC 300>SM 300>DO 300>SM322 DO16xDC24V/0.5A>6ES7 322-1BH01-0AA0 in slot 9
The completed Configuration Table is shown in Figure 15.51.

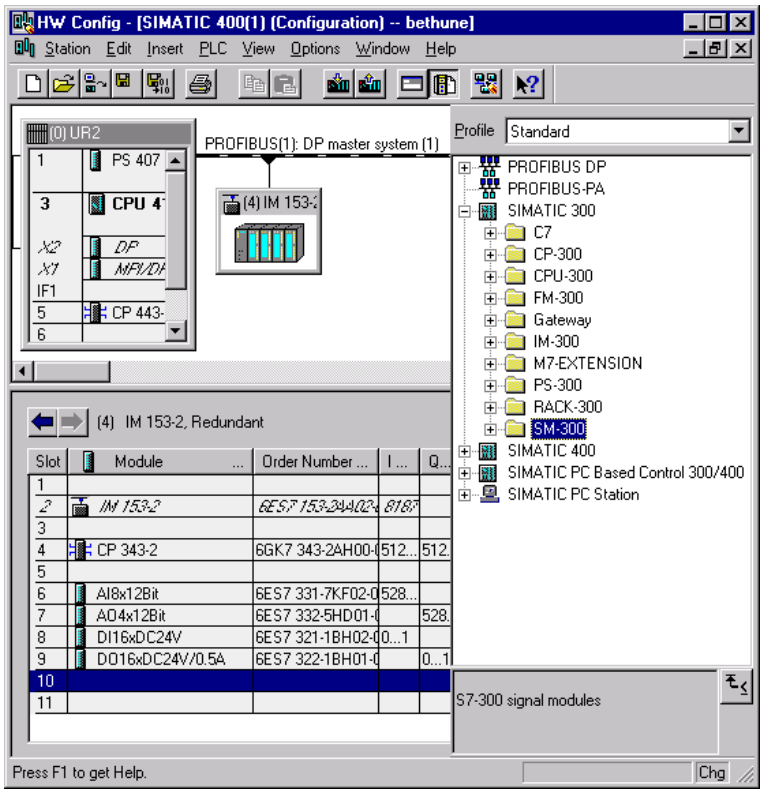

Figure 15.51: Completed HW Config table

Because of the empty slot 5 we have to set the possibility to replace modules during operation. Therefore, with the right mouse button, select the IM 153-2 box in the left upper half pane of the »HW Config – SIMATIC 400(1) « window. A small window appears. We select the option **Object Properties ... Alt+Return,** and, after that, a new window »**DP Slave properties**« appears. In this window we select in the menu the possibility »**Special«.** The obtained window is shown in Figure 15.52.

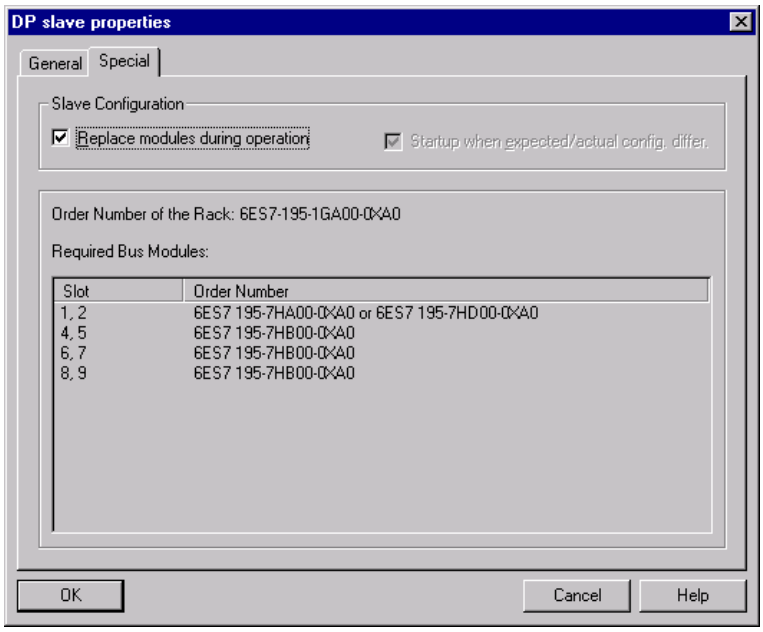

Figure 15.52: Opening the DP slave properties – special settings

In this window we select (mark the square) the option **Replace modules during operation.** After that, we close this window (with OK).

We can check the consistency of the configuration with the bar command
Station>Consistency check
STEP 7 will provide you with possible solutions to any errors which may have occurred.

The data are ready for save and compile – we use the menu bar command
Station>Save and Compile

Then, we close the HW Config window with the menu command: **Station>Exit.**

Finally, we check the consistency of the entire configuration with the bar command
Network>Check Consistency

After that, the data are ready for save and compile – we use the menu bar command
Network>Save and Compile>Compile and check everything

and we close the NetPro window with the menu command: **Network>Exit**

Finally, we can download the project configuration to the CPU. Therefore, we mark in the right half pane of the project window »**bethune**« the object **SIMATIC 400(1)** (with the left mouse button to become the blue object) and then we use the bar command in the »**SIMATIC Manager – bethune**« window:
PLC>Download
We agree with the following messages (Yes, OK, Yes, Yes):

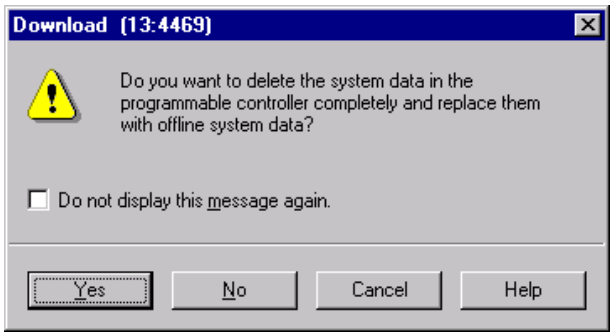

Figure 15.53: First warning message during CPU download

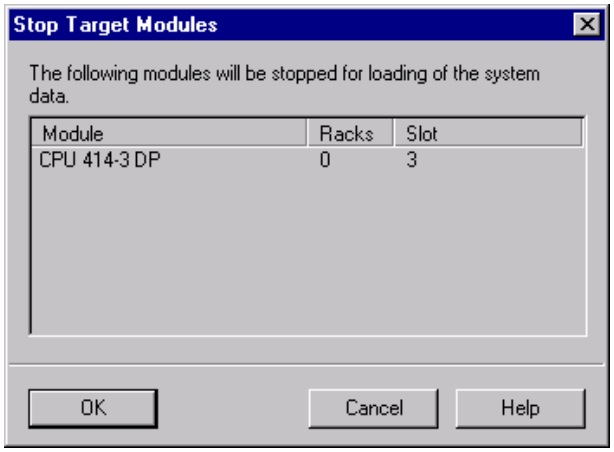

Figure 15.54: Second information message during CPU download

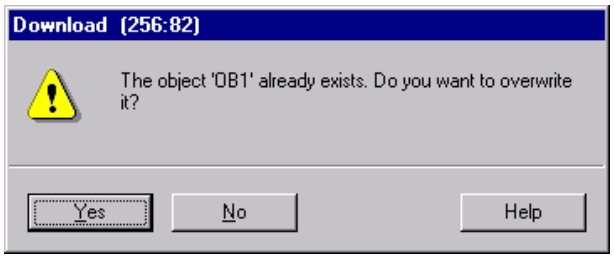

Figure 15.55: Third warning message during CPU download

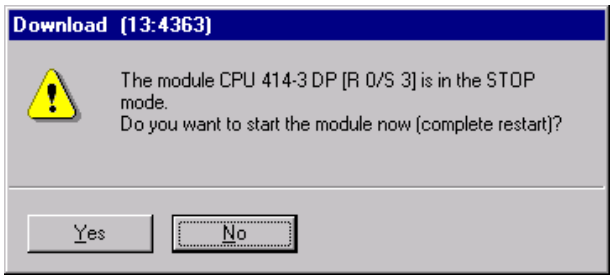

Figure 15.56: Fourth warning message during CPU download

After the downloading, the following status and fault LEDs light:

- on PS 407 green LED DC 5V and green LED DC 24V,
- on CPU 414-3 red LED EXTF and green LED RUN,
- on CP 443-1 green LED LINK and green LED RUN,
- on PS 307 green LED 24 V,
- on ET 200M red LED BF, green LED ON,
- on CP 343-2 green LED PWR, orange LED CM and blinking green LEDs B, 20+ and 10+,
- on SM 331 red LED SF,
- on SM 332 red LED SF.

15.3.3 Creating the Table of Symbols

In the Symbol Table you assign a symbolic name and the data type to all the absolute addresses which you will address later on in your program; for example, I 0.0 the symbolic name SWITCH1. These names apply to all parts of the program, and are known as global variables.
To use the Table of Symbols you have to:

- Navigate in the project window until you reach **S7 Program(1)** and double-click to open the **Symbols** component
- Enter SWITCH1 and I 0.0 in the first row
- Save the entries or changes you have made in the Symbol Table and close the window: **Symbol Table>Save** and **Symbol Table>Exit**

The Symbol Editor window is shown in Figure 15.57.

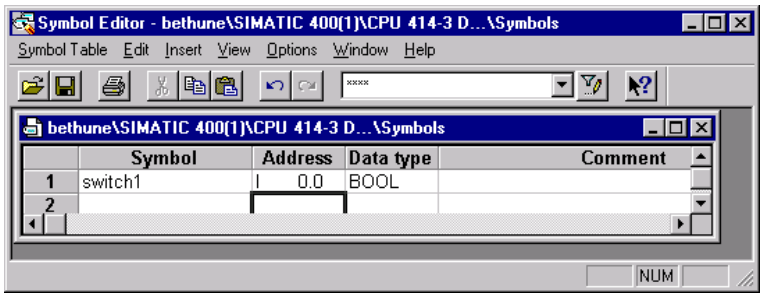

Figure 15.57: Symbol Editor window with entered symbol

15.3.4 Creating a Program

Navigate in the project window until you reach the **Block** folder. Double-click on it, and two objects appear in the right pane : System data and OB1. In STEP 7, OB1 is processed cyclically by the CPU. The CPU reads line by line, and executes the program commands. Double-click OB1 opens LAD/STL/FBD – OB1 window where we write the program. For example:

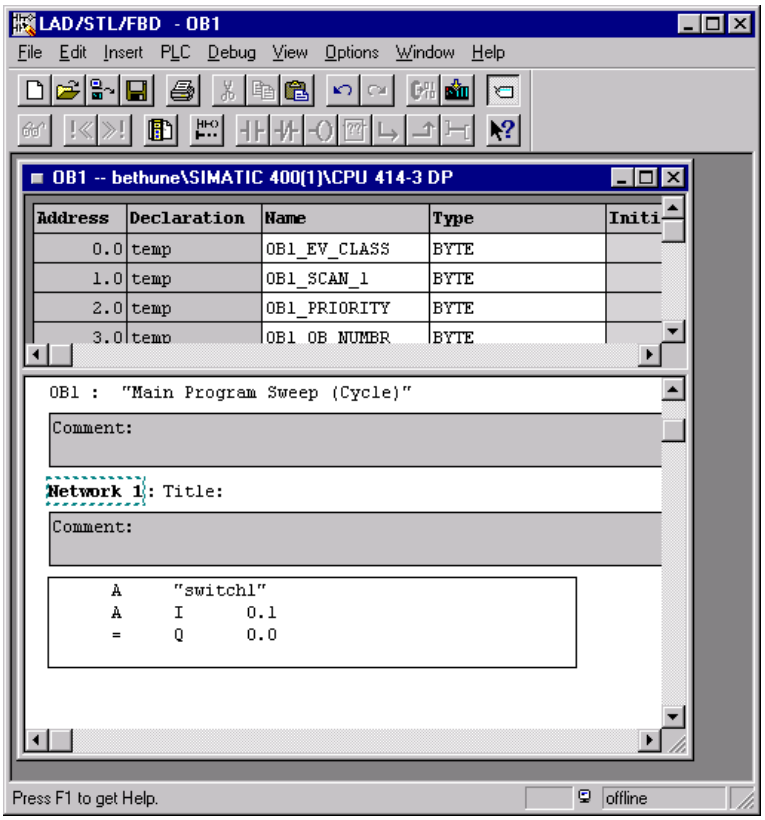

Figure 15.58: Creating a program in OB1

Save the block and exit the window: **File>Save** and **File>Exit**

15.3.5 Downloading and debugging the program

To download the program, we use the bar command in the "**LAD/STL/FBD – OB1**« window:
PLC>Download
and allow that the newly created organization blocks overwrite the old organization blocks.

Part VII – Machines applications

16. Automation of the drilling machine

16.1 Description of the machine

Our task is to control the movement of the drilling machine. The drilling machine must move from the position right above, via the positions left above, to the position left below to drill the borehole. After that, it must return to the initial position. The operating process is shown schematically in Figure 16.1.

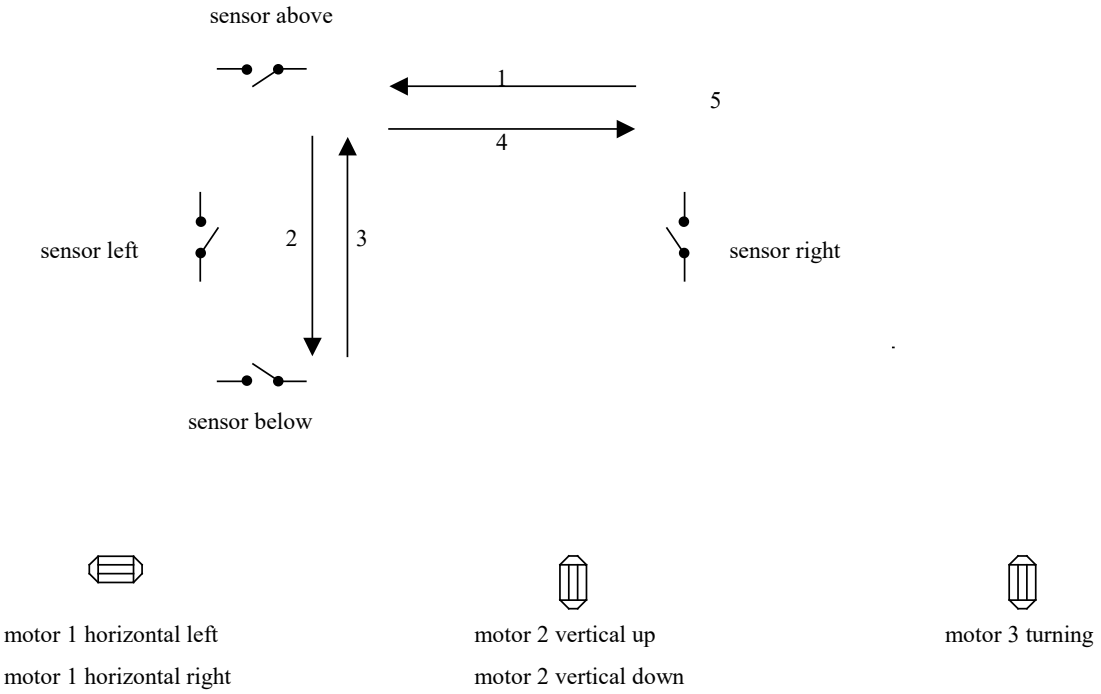

Figure 16.1: Drilling machine operating sequence

For moving we use two electrical drives: one for vertical movement and one for horizontal movement. There is also one electrical drive for turning of the drilling machine. The information about the position of the machine is obtained by means of four sensors (switches). The first sensor is positioned right, and is intended to notify the correct horizontal right position, the second sensor is positioned left, and is intended to notify the correct left position, the third sensor is positioned above, and is intended to notify the correct upper position, the fourth sensor is positioned below, and is intended to notify the correct lower position. The goal of the automation system is to actuate in a proper sequence.
At beginning of a working cycle the tool must be found in the upper right position. The working cycle left begins with turning on the switch for the execution of the working cycle. The first phase of the working cycle represents movement of tool left, as long as it has not achieved the utmost left position. In the left position the motor 3 starts with rotating the drilling machine. At the same time, motor 2 also starts with the vertical movement down. At the moment when the drilling machine makes the borehole, the motor 2 must be switched to the up movement. When the correct upper position is reached the movement up is stopped, and the horizontal movement right is started. The machine must be returned to the correct position in the upper right. When

the switch for the execution of the working cycle is still on, then the next working cycle will be executed. In other cases, the tool will stay in the upper left position as long as the switch for the execution of the working cycle is turned on.
For the design of an automation system, it is important to define the inputs and outputs of the automation system (controller) and phases of the working cycle (steps of sequential control), precisely.
We can describe the automation system working with the following Time diagram:

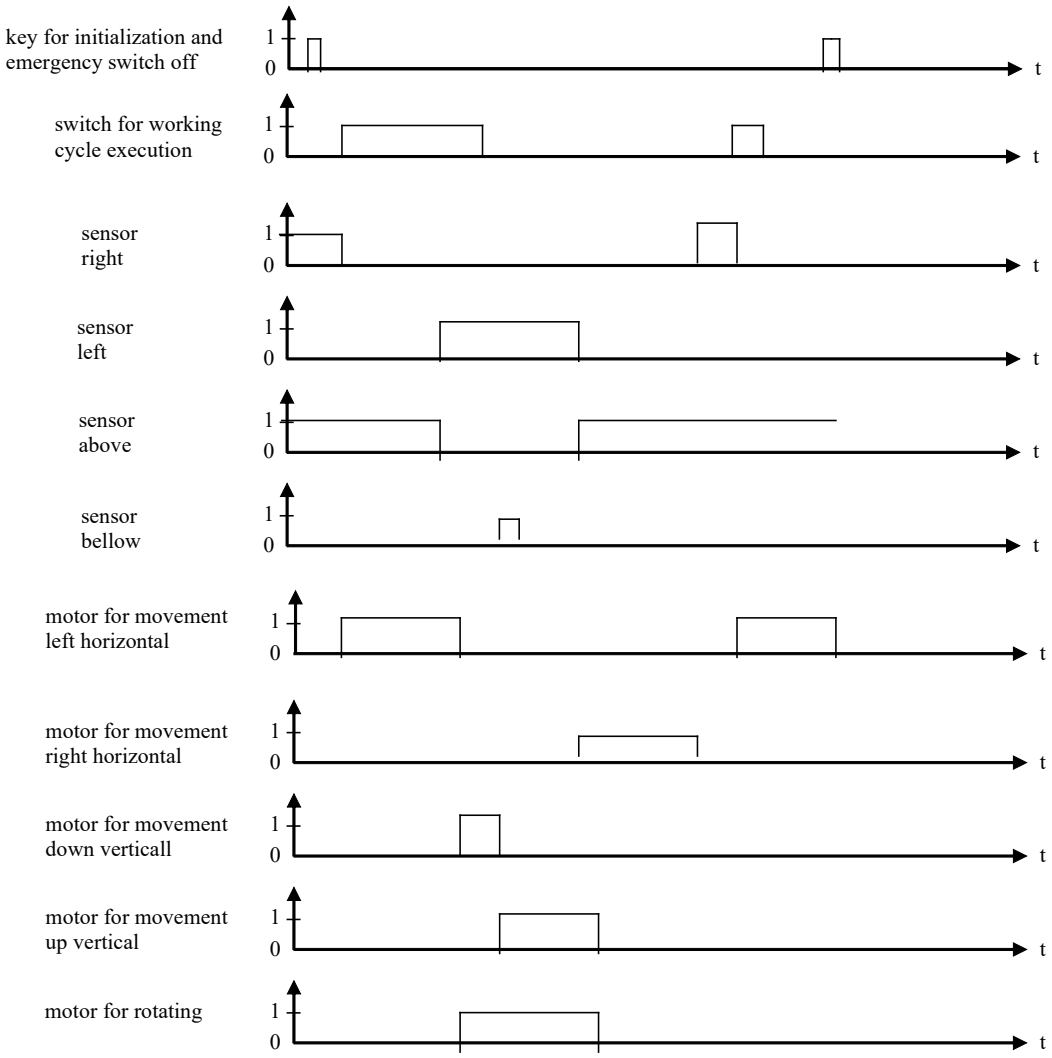

Figure 16.2: Time diagram of the signals in the drilling machine automation system

The inputs in the automation system are the outputs from a machine and the operator`s command:

- Signals from four sensors: sensor left, sensor right, sensor above, sensor below.
- Signal from the switch for execution of the working cycle.
- Signals from the key for the emergency switch off of all the motors, this key will be also used for initializing of the automation system.

The outputs from the automation system are the inputs in the machine:
- One signal for switching on/off the drive for horizontal movement left.
- One signal for switching on/off the drive for horizontal movement right.
- One signal for switching on/off the drive for vertical movement up.
- One signal for switching on/off the drive for vertical movement down.
- Oignal for switching on/off the motor for rotating drilling machine.

The working cycle consists of five steps:
- Movement from position right above to position left above.
- Movement from position left above to position left below with simultaneous rotating.
- Movement from position left below to position left above with simultaneous rotating.
- Movement from position left above to position right above.
- Waiting till the beginning for the movement left.

16.2 Choice of the kind of control system

With regards to the working cycle, we select the counting sequence control system.

16.3 Control system synthesis

16.3.1 Choice of control system inputs and outputs, Symbol Table

The control system has:
- 6 input signals,
- 5 output signals,
- 5 steps (states).

Table 16.1: Symbol Table for the drilling machine automation system

Symbol	Address	Comment
k_init	I124.0	Key for initialization and emergency switch off
s_work	I124.1	switch for working cycle execution
s_left	I124.2	sensor left
s_right	I124.3	sensor right
s_up	I124.4	sensor up
s_down	I124.5	sensor down
m_rotate	Q124.0	switch for motor for rotation
m_left	Q124.1	motor for movement horizontal left
m_right	Q124.2	motor for movement horizontal right
m_up	Q124.3	motor for movement vertical up
m_down	Q124.4	motor for movement vertical down
step_1	M0.0	step 1
step_2	M0.1	step 2
step_3	M0.2	step 3
step_4	M0.3	step 4
step_5	M0.4	step 5

16.3.2 Function diagram

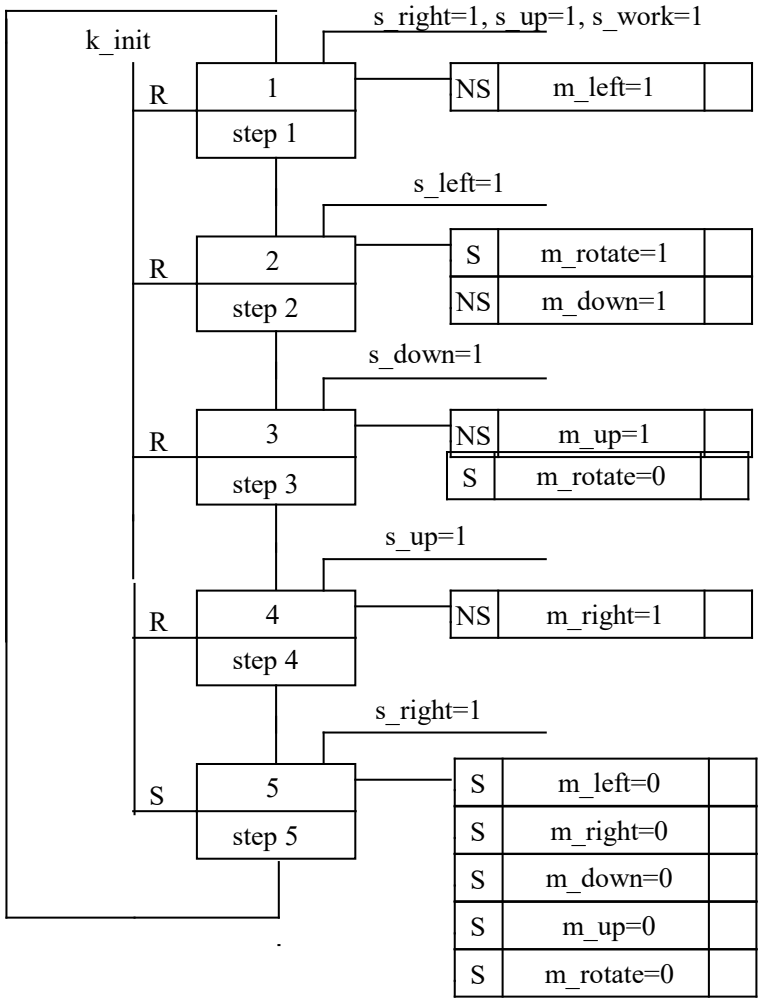

Figure 16.3: Function Diagram for the drilling machine automation system

16.4 Program

a “k_init” with binary input “s_init” set the controller in initial state (set “step_5” and reset steps “step_1” to “step_4

```
r step_1
r step_2
r step_3
r step_4
s step_5

 a “step_5”          //condition for step 1
 a “s_right”         //condition for step 1
 a “s_work”          //condition for step 1
 s “step_1”          //set step 1
r “step_5”           //reset step 5 (previous)

 a “step_1”          //condition for step 2
```

```
a “s_left”          //condition for step 2
s “step_2”          //set step 2
r “step_1”          //reset step 1 (previous)

a “step_2”          //condition for step 3
a “s_down”          //condition for step 3
s “step_3”          //set step 3
r “step_2”          //reset step 2 (previous)

a “step_3”          //condition for step 4
a “s_up”            //condition for step 4
s “step_4”          //set step 4
r “step_3”          //reset step 3 (previous)

a “step_4”          //condition for step 5
a “s_right” ”       //condition for step 5
s “step_5”          //set step 5
r “step_4”          //reset step 4 (previous)

a “step_1”          //active step 1
s “m_left”

a “step_2”          //active step 2
r “m_left”
s “m_down”
s “m_rotate”

a “step_3”          //active step 3
r “m_down”
s “m_up”

a “step_4”          //active step 4
r “m_up”
s “m_right”

a “step_5”          //active step 5
r “m_right”
r “m_left”
r “m_rotate”
r “m_up”
r “m_down”
```

17. Automatic variable sequence manipulator

17.1 Description of the industrial process

The task of the machine is to move the material from the position left below via the positions left above and right above to the position right below. The operating process is shown schematically in Figure 17.1.

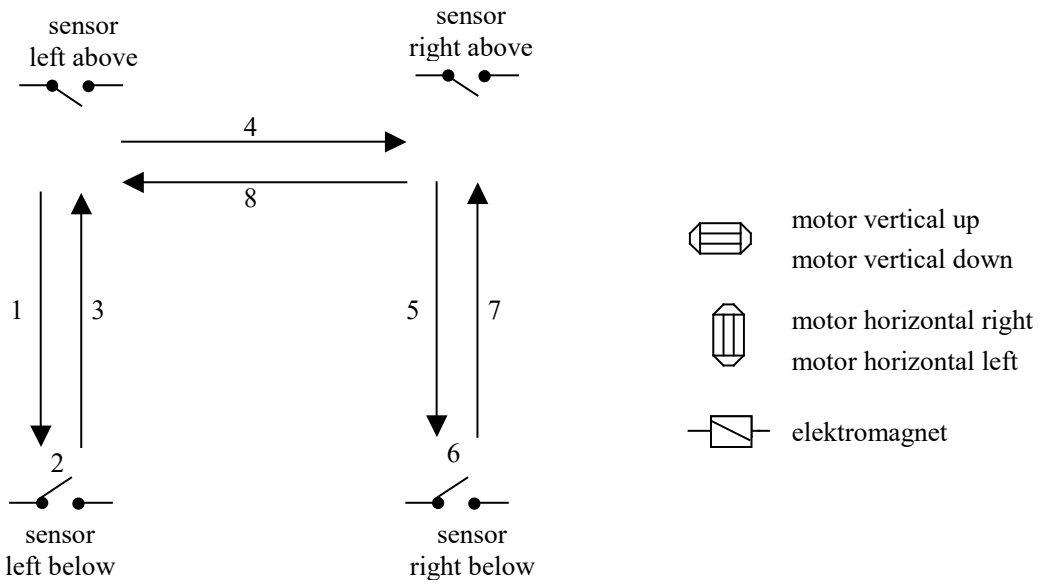

Figure 17.1: Robotic manipulator operating sequence

For transport we use two electrical drives: one for vertical movement and one for horizontal movement. The material is caught by an electromagnet, which serves to hold on and to hold off. The information about the position of the machine is obtained by means of four sensors (switches). The first sensor is left below, the second sensor is left above, the third sensor is right above, and the fourth sensor is right below. The goal of the automation system is to actuate the motors and electromagnet in a proper sequence.
At the beginning of the working cycle the tool must be found in the upper left position. The working cycle begins with turning on the switch for the execution of the working cycle. The first phase of the working cycle represents movement of tool downward, as long as has not achieved the utmost lower position. In the lower position the electromagnet is switched on. The electromagnet attracts the material to carry over. Because the phase of attraction lasts some seconds, and because we do not have a sensor which could inform about the successful catching of material, therefore, it is necessary to use some time delay before the beginning of the next phase. In our example, the motor for vertical movement can begin to lift the material not earlier than five seconds after turning on the electromagnet. Lifting of the material lasts, as long as has not achieved the upper position. After that, the horizontal movement right must be turned on. In correct right position the horizontal drive must be switched off, and the drive for moving downward must be turned on. After reaching a suitable lower position, the current through the electromagnet is interrupted. Because the phase of loosening of the material lasts some time, and because we do not have a sensor which could inform us about the end of the loosening process, we must, with an appropriate delay, prevent the incorrect activity of the drive for vertical movement. After the prescribed time delay the machine must be returned to the correct upper left position. This position will be reached in two steps: first, vertical movement up until

the upper sensor, and then horizontal movement left until the left sensor. When the switch for the execution of the working cycle is still on, then the next working cycle will be executed. In any case, the tool will stay in the upper left position as long as the switch for the execution of the working cycle will be turned on.
For the design of an automation system, it is important to define the inputs and outputs of the automation system (controller) and phases of working cycle (steps of sequential control) precisely.
We can describe the automation system working with the next time diagram:

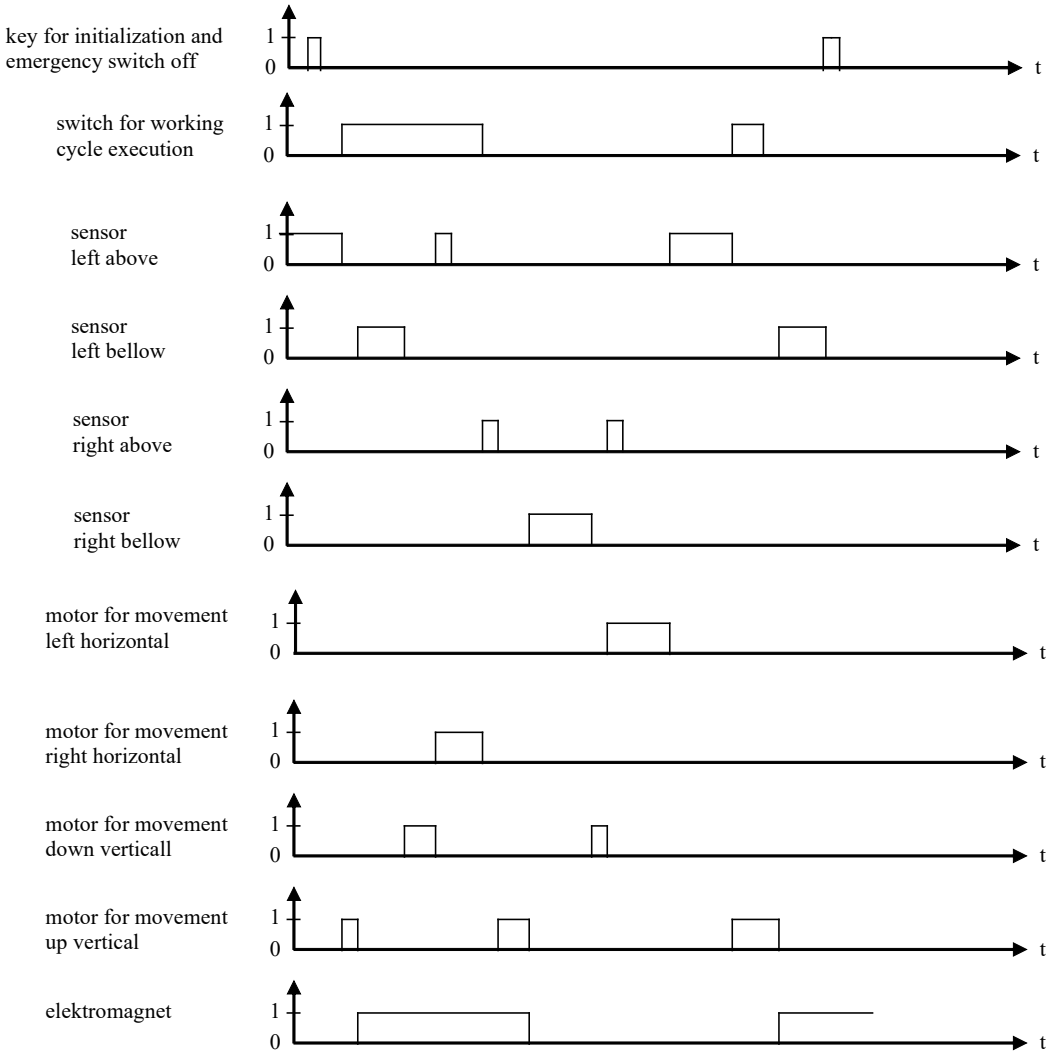

Figure 17.2: Time diagram of the signals in the robotic manipulator automation system

The inputs in the automation system are the outputs from machine and the operator`s commands:

- Signals from four sensors: sensor left below, sensor left above, sensor right above, sensor right below.
- Signal from switch for execution of the working cycle.
- Signals from the key for emergency switch off of all motors. This key will also be used for initializing of the automation system.

The outputs from the automation system are the inputs in the machine:

- Four signals for switching the drives on/off: a motor for left horizontal movement, a motor for right horizontal movement, a motor for vertical up movement, and a motor for down vertical movement.
- A signal for switching the electromagnet on/off.

The working cycle consists of nine steps:

- Movement from position left above to position left below.
- Switching on the electromagnet.
- Movement from position left below to position left above.
- Movement from position left above to position right above.
- Movement from position right above to position right below.
- Switching off the electromagnet.
- Movement from position right below to position right above.
- Movement from position right above to position left above.

17.2 Choice of kind of control system

With regard to the working cycle we select the counting sequence control system.

17.3 Control system synthesis

17.3.1 Choice of control system inputs and outputs, Symbol Table

The control system has:

- 6 input signals,
- 5 outputs,
- 9 steps (states).

Table 17.1: Symbol Table for the robotic manipulator automation system

Symbol	Address	Comment
k_init	I124.0	Key for initialization and emergency switch off
s_work	I124.1	switch for working cycle execution
s_le_ab	I124.2	sensor left above
s_le_be	I124.3	sensor left belove
s_ri_ab	I124.4	sensor right above
s_ri_be	I124.5	sensor right belove
s_mag	Q124.0	switch for electromagnet
m_ho_le	Q124.1	motor for movement horizontal left
m_ho_ri	Q124.2	motor for movement horizontal right
m_ve_od	Q124.3	motor for movement vertical down
m_ve_up	Q124.4	motor for movement vertical up
step_1	M0.0	step 1
step_2	M0.1	step 2
step_3	M0.2	step 3
step_4	M0.3	step 4
step_5	M0.4	step 5
step_6	M0.5	step 6
step_7	M0.6	step 7
step_8	M0.7	step 8
step_9	M1.0	step 9

17.3.2 Function diagram

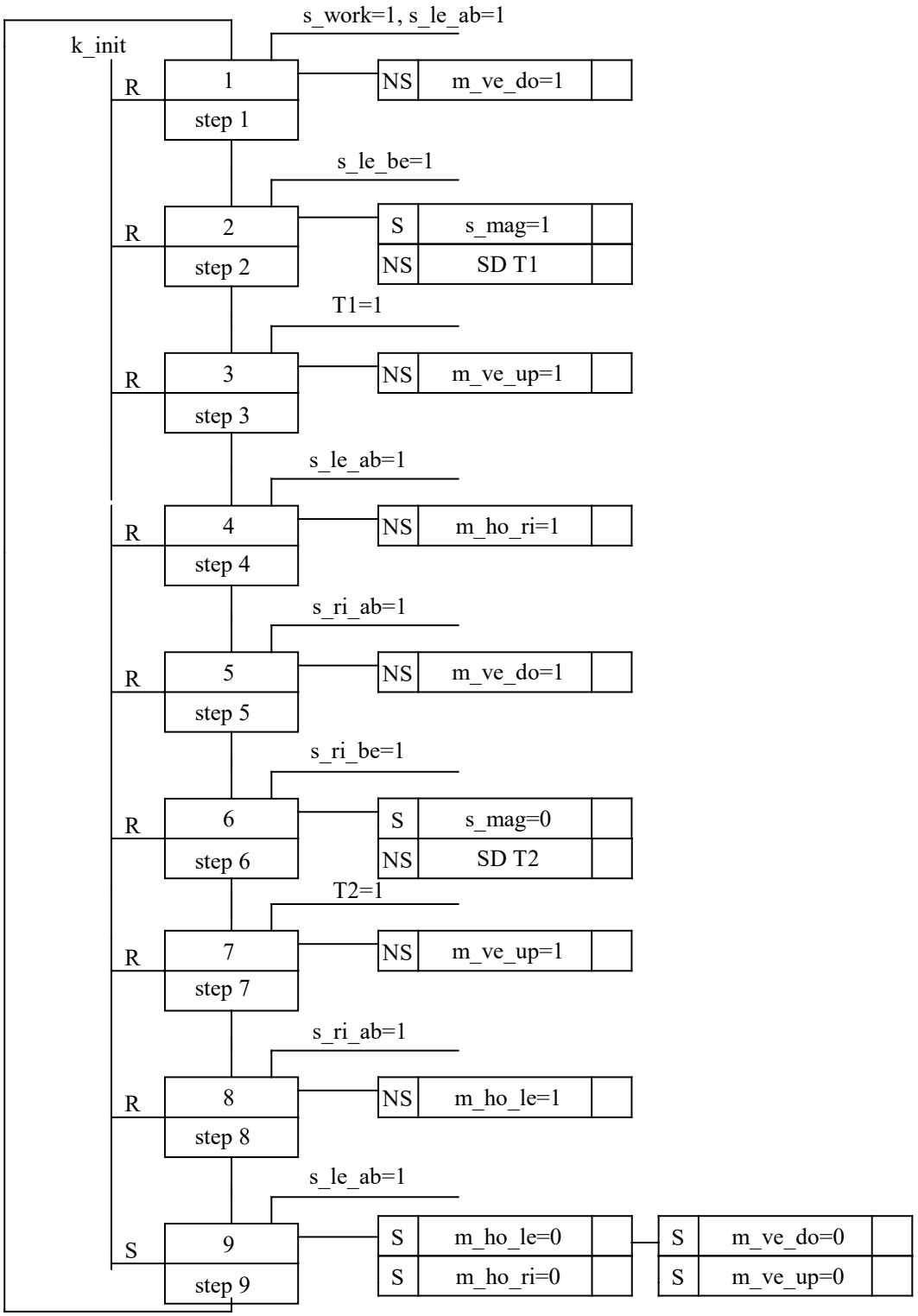

Figure 17.3: Function Diagram for the robotic manipulator automation system

17.4 Program

```
a “k_init”   with binary input “s_init” set the controller in initial state (set “step_9” and reset
             steps “step_1” to “step_8
r step_1
r step_2
r step_3
r step_4
r step_5
r step_6
r step_7
r step_8

 a “step_9”          //condition for step 1
 a “s_le_ab”         //condition for step 1
 a “s_work”          //condition for step 1
 s “step_1”          //set step 1
r “step_9            //reset step 9 (previous)

 a “step_1           //condition for step 2
 a “s_le_be          //condition for step 2
 s “step_2”          //set step 2
 r “step_1”          //reset step 1 (previous)
 a “step_2           //condition for timer 1
l S5T#5S
 sd T1

 a “step_2           //condition for step 3
 a “T1               //condition for step 3
 s “step_3”          //set step 3
 r “step_2”          //reset step 2 (previous)

 a “step_3”          //condition for step 4
 a “s_le_ab          //condition for step 4
 s “step_4”          //set step 4
 r “step_3”          //reset step 3 (previous)

 a “step_4”          //condition for step 5
 a “s_ri_ab” ”       //condition for step 5
 s “step_5”          //set step 5
 r “step_4”          //reset step 4 (previous)

 a “step_5” ”        //condition for step 6
 a “s_ri_be”         //condition for step 6
 s “step_6”          //set step 6
 r “step_5”          //reset step 5 (previous)

 a “step_6”          //condition for timer 2
l S5T#5S
 sd T2
```

```
a “step_6”          //condition for step 7
a “T2” ”            //condition for step 7
s “step_7”          //set step 7
r “step_6”          //reset step 6 (previous)

a “step_7” ”        //condition for step 8
a “s_ri_ab” ”       //condition for step 8
s “step_8”          //set step 8
r “step_7”          //reset step 7 (previous)

a “step_8” ”        //condition for step 9
a “s_le_ab” ”       //condition for step 9
s “step_9”          //set step 9
r “step_8”          //reset step 8 (previous)

a “step_1”          //active step 1
s “m_ve_do”

a “step_2”          //active step 2
r “m_ve_do”
s “s_mag”

a “step_3”          //active step 3
s “m_v_up”

a “step_4”          //active step 4
r “m_ve_up”
s “m_ho_ri”

a “step_5”          //active step 5
r “m_ho_ri”
s “m_ve_do”

a “step_6”          //active step 6
r “m_ve_do”
r “s_mag”

a “step_7”          //active step 7
s “m_ve_up”

a “step_8”          //active step 8
r “m_ve_up”
s “m_ho_le”

a “step_9”          //active step 9
r “m_ve_do”
r “m_ve_up”
r “m_ho_ri”
r “m_ho_le”
```

References

[1] Richard C. Dorf, Robert H. Bishop: Modern Control Systems; 1998 Addison Wesley Longman, Inc.

[2] Siemens, Simatic: Komponenten fuer die Vollintegrierte Automation, Katalog ST 70

[3] Hans Berger: Automating with STEP 7 in STL and SCL; 2001 Publicis MCD Corporate Publishing Erlangen and Munich

[4] Hans Berger: Automatisieren mit STEP 7 in KOP und FUP; 2003 Publicis MCD Corporate Publishing Erlangen

[5] Siemens, Simatic: Statement List (STL) for S7-300 and S7-400 Programming, Manual

[6] Siemens, Simatic: S7-300 Programmable Controller CPU 314IFM, Instruction List

[7] Siemens, Simatic: System Software for S7-300 and S7-400 Program Design, Programming Manual

[8] Siemens, Simatic: System Software for S7-300 and S7-400, System and Standard Functions, Reference Manual

[9] Siemens, Simatic: S7-300 Programmable Controller, Installation and Hardware

[10] Siemens, Simatic: Standard Software for S7 and M7, STEP 7, User Manual

[11] Siemens, Simatic: Automation and Drives, Katalog CA01 04/2003, on CD

[12] Siemens, Simatic: Components for Totally Integrated Automation, Catalog ST 70 2001

[13] Siemens, Simatic: Working with STEP 7 V5.1, Getting started, Edition 08/2000

[14] Siemens, Simatic: Programming with STEP 7 V5.1, Manual, Edition 08/2000

[15] Siemens, Simatic: Distributed I/O Device ET 200M, Manual, Edition 10/2002

[16] Siemens, Simatic:Automation System S7-400 CPU Specifications, Reference manual, Edition 12/2002

[17] Siemens, Simatic: S7-400 and M7-400 Programmable Controllers Hardware and Installation, Installation manual, Edition 12/2002

[18] Siemens, Simatic: SIMATIC NET NCM S7 for PROFIBUS, Manual Volume 1 of 2, Edition 11/2002

[19] Siemens, Simatic: Comunication with SIMATIC, Manual, Edition 2

Automation With Programmable Logic Controllers: Textbook

Jožef Ritonja
University of Maribor, Faculty of Electrical Engineering and Computer Science, Maribor, Slovenia
jozef.ritonja@um.si

The textbook *Automation with Programmable Logic Controllers* covers essential concepts and practical methods in industrial automation, with a strong focus on Siemens hardware and software. The textbook focuses primarily on the SIMATIC S7-300 family of controllers, which still represent the widely used technology in many industrial environments. Special attention is given to the Statement List (STL) programming language, offering deep insights into how PLCs operate. The content is structured into 17 chapters, ranging from automation fundamentals to real-world industrial case studies. The book is intended for engineering students, practicing professionals, and anyone seeking to acquire or enhance knowledge in PLC technology. With its practical orientation and regionally relevant focus, the publication fills a gap in the existing literature and serves as a valuable resource for both education and professional development in the field of industrial automation.

DOI
https://doi.org/10.18690/um.feri.6.2026

ISBN
978-961-299-159-3

Keywords:
programmable logic controllers, industrial automation, automation education, Siemens SIMATIC, STEP 7, statement list language programming

University of Maribor Press

University of Maribor

Faculty of Electrical Engineering
and Computer Science