

MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE
NATIONAL UNIVERSITY "ODESSA MARITIME ACADEMY"

Gorb S.I., Nikolskyi V.V., Shapo V.F., Khniunin S.H.

Programming controllers in the integrated development environment

TRAINING MANUAL

PRACTICE

Odessa – 2017

Г 67 Gorb S.I., Nikolskyi V.V., Shapo V.F., Khniunin S.H. Programming controllers in the integrated development environment: training manual. Practice. – Odessa: National University "Odessa Maritime Academy", 2017. – 164 p.

Programming of controllers was considered using laboratory benches developed by a group of universities within the framework of the project TEMPUS 544010-TEMPUS-1-2013-1-DE-TEMPUS-JPHES – "Trainings in Automation Technologies for Ukraine" (TATU). These benches are distinguished by a full set of advanced technologies that are used in automation systems and modular construction, minimal set of technical means for organizing comprehensive training in the use of controllers, adaptation of equipment for the organization of learning process.

The PC Worx development environment was chosen as the programming environment.

Examples of solving the problems of technological processes automation of different complexity are given.

The tutorial is intended for bachelors, specialists and masters of disciplines "information technology", "mechanical engineering", "electrical engineering", "automation and instrument engineering", "transport", as well as postgraduate education of engineers providing design and operation of automation systems (including marine systems).

Approved by the Academic Council of the National University "Odessa Maritime Academy" as a training manual for disciplines "Automation and Computer-Integrated Technologies", "River and Maritime Transport", minutes dated October 27, 2016, No. 3.

Approved by the Academic Council of Post Graduate Institute of Specialists in Maritime and Inland Water Transport (Odessa) as a training manual for specialists and masters of disciplines "information technology", "mechanical engineering", "electrical engineering", "automation and instrumentation, "transport", minutes dated October 28, 2016, No. 3.

Reviewers: V.Y. Voropaeva, Candidate of Technical Sciences, Associate Professor, Vice-Rector for Scientific and Pedagogical Work of Donetsk National Polytechnic University;
I.I. Klyuchnik, Candidate of Technical Sciences, Professor, Head of Department of design and operation of electronic devices of Kharkov National University of Radio Electronics;
R.A. Shaporin, Candidate of Technical Sciences, Associate Professor, Head of Department of Computer Intelligent Systems and Networks of Odessa National Polytechnic University.

ISBN 978-966-7591-73-1

© S.I. Gorb, V.V. Nikolskyi, V.F. Shapo, S.H. Khniunin, 2017

Contents

INTRODUCTION	4
1. TRAINING LABORATORY FACILITIES TATU SMART LAB	
1.1. TATU Smart Lab hardware	5
1.2. Hardware module "Programmable controllers and PROFINET"	8
1.3. "PROFIBUS" hardware module	24
1.4. Hardware Module "Process Modeling"	30
1.5. Switching of hardware modules	34
2. CONTROLLERS SOFTWARE	
2.1. AUTOMATIONWORX Software Suite environment	38
2.2. Hardware requirements for the AUTOMATIONWORX Software Suite environment	41
2.3. Installing the AUTOMATIONWORX Software Suite environment	42
3. PC WORX INTEGRATED DEVELOPMENT ENVIRONMENT	
3.1. PC Worx interface and modes of operation	50
3.2. Creating a new project	56
3.3. Configuring PC Worx when working with ILC 151 GSM/GPRS controller	67
3.4. Configuring PC Worx when working with AXC 3050 controller	76
4. PROGRAMMING IN PC WORX INTEGRATED DEVELOPMENT ENVIRONMENT	
4.1. Function Block Diagram (FBD)	97
4.2. Ladder Diagram (LD)	118
4.3. Structured Text (ST)	127
5. EXAMPLES OF CONTROL SYSTEMS PROJECTS	
5.1. Simulation model of ship auxiliary boiler control system	143
5.2. Control system for reciprocating compressor unit	145
5.3. Scarecrow for airports and gardens	156
CONCLUSION	159
ANNEX. Elements of the LD language according to IEC 61131-3	160
List of references	163

INTRODUCTION

Controllers are a present-day element base of process automation systems. Currently, they are assembled on microcircuit die and they perform the functions of microcomputer. Apart from processor, inside a single chip various memory devices (RAM and ROM), I/O ports, communication interfaces, timers, system clock and peripheral devices are mounted, that allow the components to work and interact with each other and external devices using special microprograms stored inside the controller. This enables the use of controllers in various applications, from power plants to household appliances.

Controller that contains an operating system and software and has standard input and output signals is called programmable. The term "Programmable Logic Controller" is also used, which appeared when programmable controllers were used to control sequential logic operations. These controllers had only binary inputs and outputs, where values of the signals could correspond only to logical zero or logical unit. Present-day programmable controllers have analog inputs and outputs and in addition to logical operations are able to perform almost any arithmetic operations and implement complex control laws. Therefore, the use of the term "logical" in the name of programmable controllers can be considered obsolete.

Compared to microprocessors with unconditional logic, programmable controllers have more prospects in "alternative" control systems, especially if the latter require adaptation to controlled objects. Whenever programmable controllers are used, the time for designing, creating and configuring control systems is reduced. They can also be considered the most advanced means of process automation among unified systems.

The world's largest producers of programmable controllers are Allen-Bradley, Omron, Schneider Electric, Siemens, Advantech, Delta, VIPA, Mitsubishi Electric, WAGO I/O, Phoenix Contact, Segnetics, Fastwel, Owen, Kontat and Tecon.

During training of specialists in disciplines "information technology", "mechanical engineering", "electrical engineering", "automation and instrument engineering", "transportation" knowledge and skills in programming controllers, organization of data transmission in digital control networks, including wireless, real-time process control and integration of control systems using devices from different manufacturers should be provided. This will allow specialists to design, develop, operate and repair modern automation systems, as well as to organize pre-project work on the automation of various technological processes and plants.

In the proposed training manual only questions of programming controllers are considered. In this case, PC Worx was chosen as the integrated development environment – it has good functionality and quite a lot of elements, similar to other programming environments.

1. TRAINING LABORATORY FACILITIES TATU SMART LAB

1.1. TATU Smart Lab hardware

TATU Smart Lab (TSL) is a flexibly configurable mobile set of devices for teaching modern automation technologies. It contains devices from different manufacturers and was developed within the framework of Industry 4.0 (the fourth stage of industrial revolution). It is a German public-private program in which large German IT companies create fully automated industries where individual devices and their nodes can communicate with each other and consumers using wireless data transmission technologies.

TSL allows studying a number of modern technologies for building automation systems. Within the framework of TATU project, these technologies are divided into the following training modules.

1. Programming controllers in the PC Worx integrated development environment and hardware-independent CoDeSys development environment.
2. Use of PROFINET and Modbus TCP standards and integration of automation systems with PROFIBUS networks.
3. Wireless data transmission technologies.
4. Management of real-time processes.
5. Introduction to real time data exchange standard OPC DA.

The first training module is focused on exploring the PC Worx integrated development environment, offered by Phoenix Contact for programming and modeling automation systems. This software is aimed at devices and modules manufactured by Phoenix Contact. Other companies that develop programmable controllers also produce their own programming environments, designed only for proprietary models of programmable controllers. Typically, development environments are distributed on a paid basis, but there are also "limited" editions with limited capabilities that can be obtained for free.

Integrated development environments provide the use of libraries of standard elements and procedures, quick input of standard elements, convenient debugging of programs with control of input values and step-by-step execution of programs, visualization of programs with comments of its modules, loading programs into controllers.

CoDeSys (COntroller DEvelopment SYStem) development environment is distributed by 3S-Smart Software Solutions GmbH (Kempten, Germany) for free and works with many models of programmable controllers produced by different manufacturers.

PC Worx and CoDeSys support all five programming languages of the third version of the International Standard of the International Electrotechnical Commission (IEC) 61131-3, which was released in 2012:

IL – Instruction List (a text language similar to assembler. It is rarely used by applications engineers);

FBD – Function Block Diagram (graphical language using function blocks that are "written" in other languages.) Typical function blocks include filter, trigger, PID controller, timer, pulse generator, etc. Its specific feature is illustrative purpose);

LD – Ladder Diagram (graphical language that forms logical conditions and result of operation in the form of electrical circuit, through which the current either flows or does not flow. The circuit consists of contacts and coils of electromagnetic relays. For performing arithmetic operations, it contains function blocks of operations of addition, multiplication, average computation, etc. It is convenient for specialists who are used to trace a signal on a ladder diagram in case of debugging programs and equipment fail, but it is inconvenient for implementing complex algorithms, since it does not support functions and programs);

SFC – Sequential Function Chart (graphical high-level language, which allows describing technological process as transitions between sets of states. It is intended for programming the sequence of actions at specified intervals or when certain events occur. SFC is based on mathematical apparatus of networks; it is rarely used);

ST – Structured Text (a high-level text language that contains constructions of the type IF ... THEN ... ELSE, WHILE ... DO, Boolean and arithmetic operators that are similar to classical programming languages and are the closest to Pascal; it is convenient for implementing complex algorithms) .

In Russia the IEC 61131-3 standard is implemented as State Standard R IEC 61131-3-2016 "Programmable controllers. Part 3. Programming Languages". The latter can be used as a translation of IEC 61131-3 into Russian.

CoDeSys has also a number of implemented extensions to the IEC 61131-3 specification, the most significant of which is the support of object-oriented programming.

The programs (projects) compiled into machine code are loaded into controller. CoDeSys includes a set of software tools for preparing and debugging programs, compilers, configurators, visualization editors, etc. The project created in CoDeSys can be stored on computer or in programmable controller.

The enhanced professional version of this development environment is called CoDeSys Professional Developer Edition. It is distributed under license and includes support for UML class and state diagrams, connection of Subversion version control system (distributed freely), static analyzer and code profiler.

The second training module was designed for studying modern data transfer technologies PROFINET and Modbus.

PROFINET can use the TCP/IP (Transmission Control Protocol/Internet Protocol) network protocols and real-time Ethernet. TCP/IP protocols allow managing automation devices remotely over the Internet.

Modbus can be used for data transmission through serial communication lines and through networks using TCP/IP protocols (Modbus TCP). Developed by Modicon, it is supported by a non-profit organization Modbus-IDA. The advantage of this technology is that virtually all control and monitoring systems have software drivers for working with Modbus.

PROFIBUS networks, which belong to the previous generation of data transmission networks, are considered in the training module. At the same time, they have a wide scope of application and are supported in millions of automation devices that have been installed worldwide over the past 20 years (since the early 1990s). PROFIBUS network technologies are largely based on Ethernet technologies and industrial Ethernet implementations, which are the most popular all over the world.

The third training module provides a wide range of technologies for wireless data transmission. This direction is extremely important in connection with the widespread introduction of wireless networks in recent years.

In the fourth training module tools for real-time process management are studied: real-time operating systems, software and hardware systems for building real-time systems, applied and system software, hardware solutions.

The fifth training module covers the main approaches used for real-time data exchange in networks within the framework of the OPC group of standards (OLE – Object Linking and Embedding for Process Control, implementation and linking of objects for process control). OPC technology defines two types of software: OPC server that directly communicates with devices, and OPC client that receives data from the OPC server and passes control commands to the OPC server. Software developers can organize data acquisition for processing from various external technological systems via unified interface. Thus, it is possible to avoid binding to specific models of equipment of specific manufacturers, and the data exchange processes are simplified and unified.

OPC DA (Data Access) is one of the eight standards included in the OPC group. OPC DA is used much more often than other group standards, as it provides modes of synchronous and asynchronous real-time data exchange with programmable controller, human-machine interface systems, numerical control systems and distributed control systems.

TSL consists of three hardware modules, each of which is housed in a separate portable box (suitcase), designed to study specific topics and can be used independently of the rest. TSL modules, their interaction and possible cable connection are shown in Fig. 1.1.

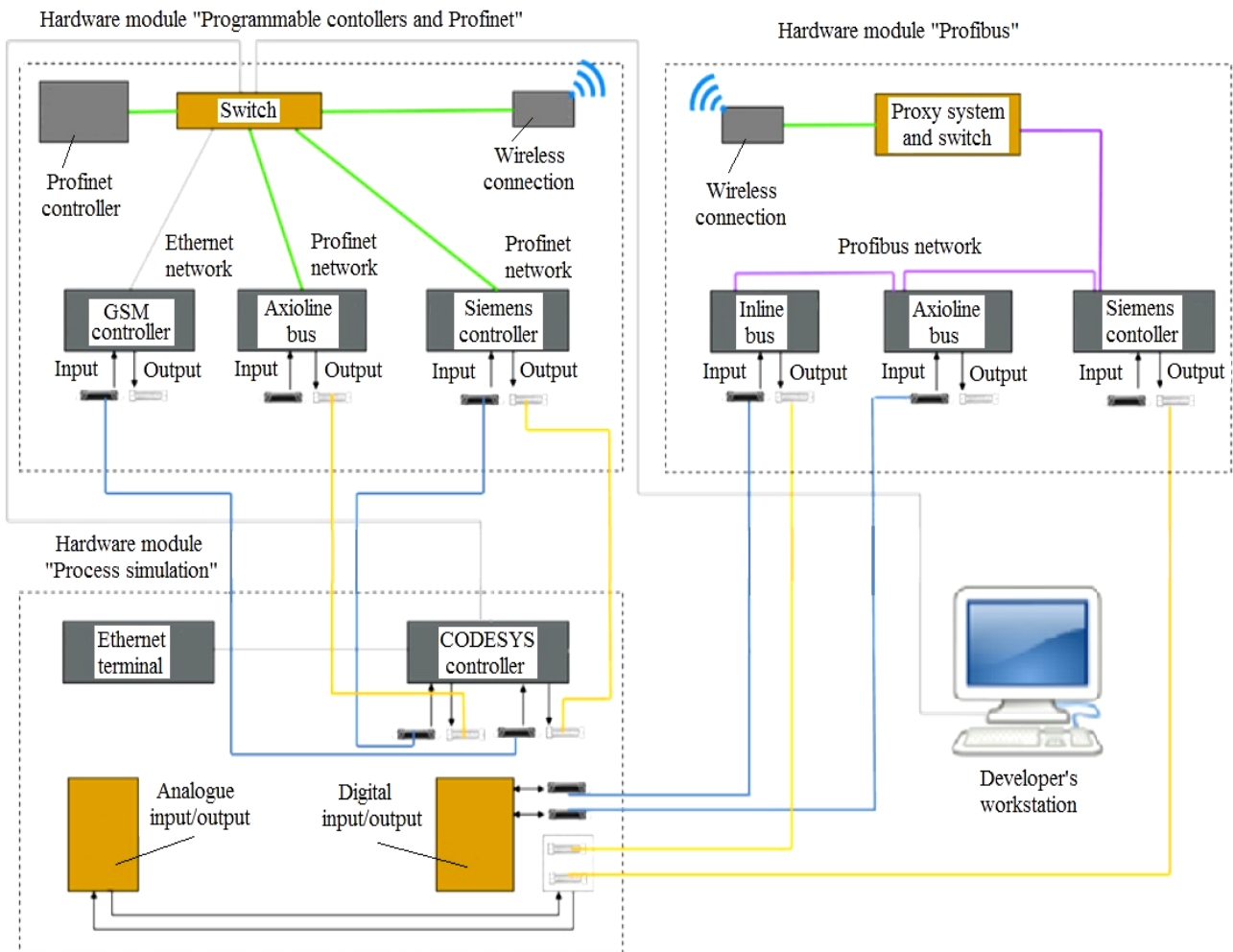


Fig. 1.1. The structure of TSL hardware modules and their connection to computer

1.2. Hardware module "Programmable controllers and PROFINET"

The first hardware module (Fig. 1.2) is called "Programmable Controllers and PROFINET", since it contains two programmable controllers (AXC 3050 and ILC 151 GSM/GPRS), PROFINET I/O devices, managed network switch and wireless access point for wireless LAN.

This hardware module can be used for studying a wide range of topics, from basic programming of controllers in PC Worx to working with data transfer technologies developed on the basis of Ethernet technology and wireless data transmission technologies. Two programmable controllers can communicate using TCP/IP or Modbus TCP data transfer protocols. The PROFINET I/O devices can only be connected to the AXC 3050 controller and the ILC 151 GSM/GPRS Inline controller does not have PROFINET functionality. Thus, the devices and technologies implemented in this module allow studying the material of training modules 1, 2 and 3.

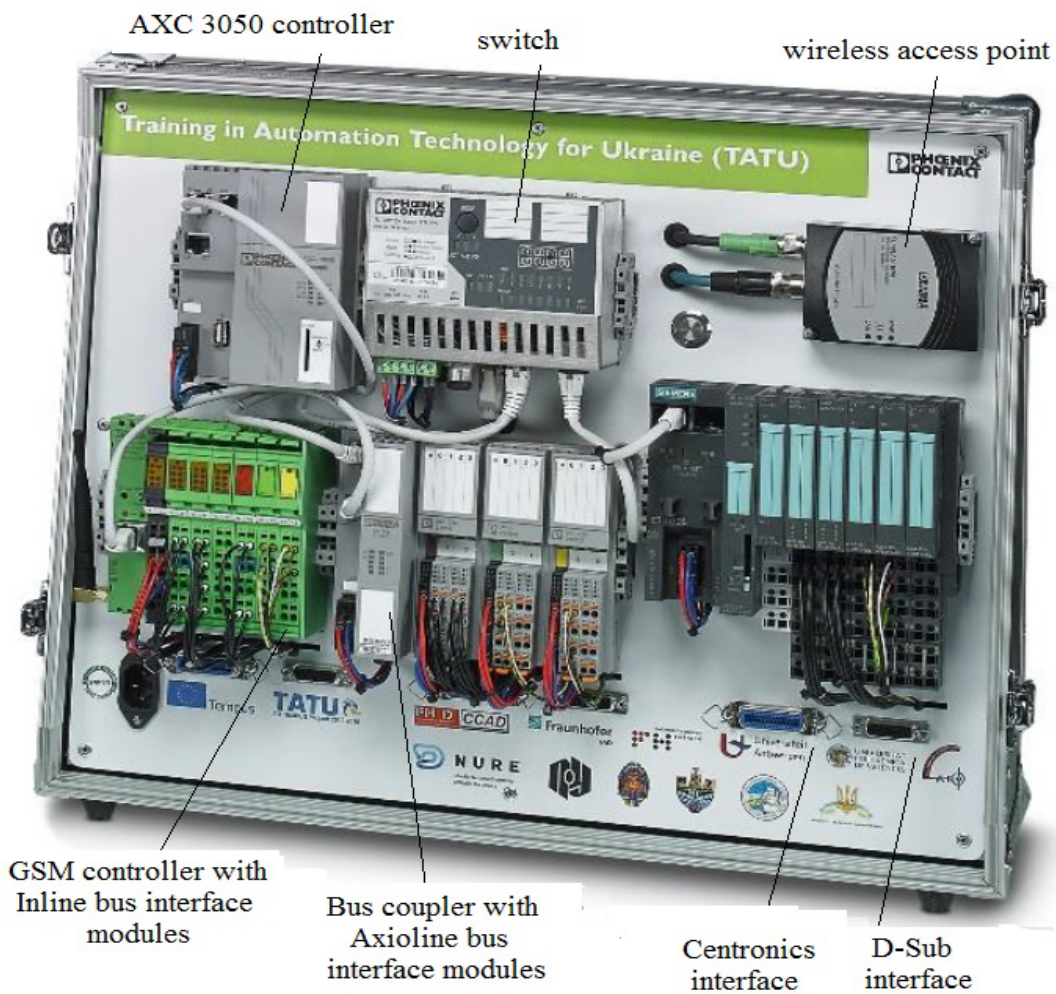


Fig. 1.2. Appearance of the first TSL hardware module "Programmable controllers and PROFINET"

Fig. 1.3 shows the connection diagram of devices in the first hardware module.

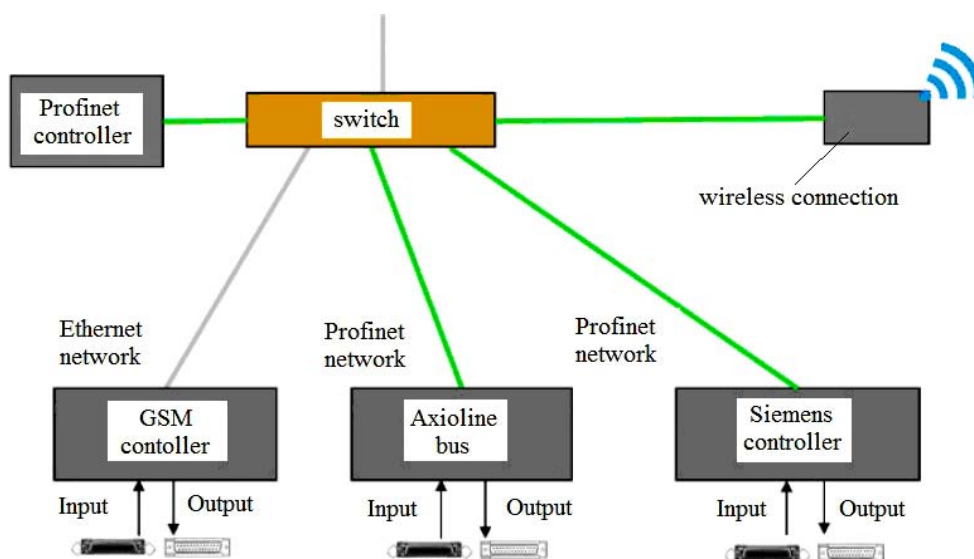


Fig. 1.3. Connecting devices in the first TSL hardware module "Programmable controllers and PROFINET"

The **AXC 3050 programmable controller** (Fig. 1.4) can work with Ethernet family networks and the Axioline F local bus, which supports any Ethernet-based data transfer protocols. The Axioline station can be created by connecting Axioline modules to the controller. The Axioline F local bus, described in more detail below, can be used for the sequential installation of various modules (devices) one closely to the other.

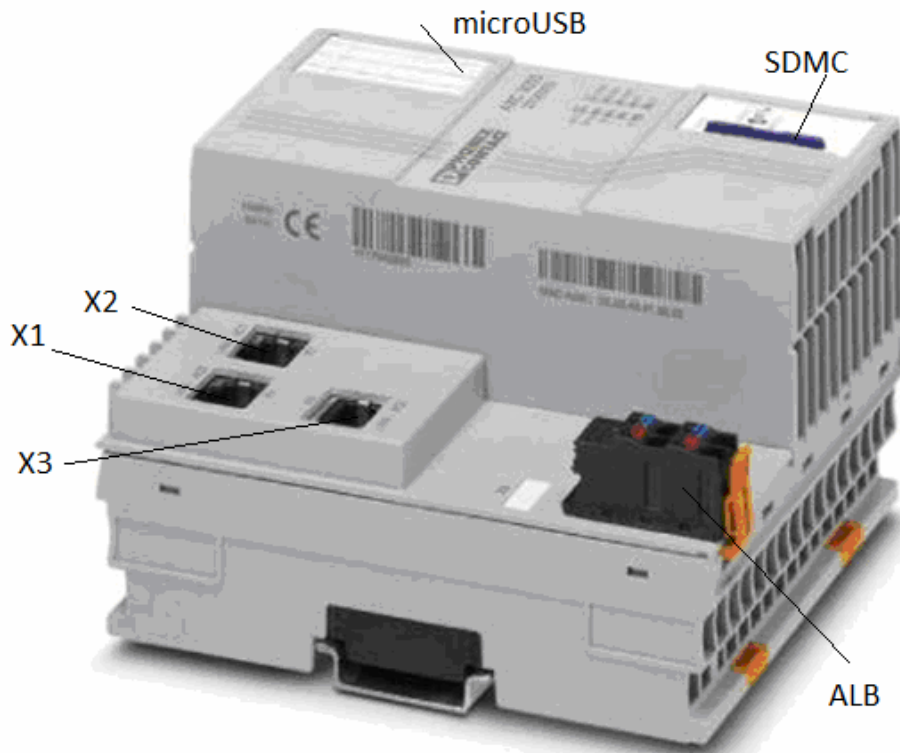


Fig. 1.4. Controller AXC 3050: X1, X2, X3 – interfaces for connecting to Ethernet family (8-pin RJ45 connectors); SDMC (Secure Digital Memory Card – a slot for connecting flash cards such as Secure Digital); MicroUSB – a connector for connecting USB devices, hidden under a paper tag for information labels; ALB – Axioline bus

Fig. 1.5 shows the connection of supply voltages.

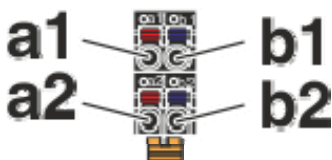


Fig. 1.5. Contacts pin map on remote panel: a1, a2 (red terminals) – supply voltage 24 V DC; contacts b1, b2 (blue terminals) – grounding

The AXC 3050 controller can be fully configured and programmed in one of five programming languages in accordance with IEC 61131-3 standard with PC Worx when connected over Ethernet network.

The AXC 3050 controller has built-in interfaces for connecting devices over

Ethernet network. It allows you to configure the controller using TCP/IP or UDP (User Datagram Protocol) protocols.

The controller has three integrated Ethernet ports X1, X2, X3 (see Fig. 1.4).

Using function blocks IP_USEND (sending user data via TCP/IP protocol) and IP_URCV (receiving user data via TCP/IP protocol) in PC Worx, it is possible to organize data exchange (i.e., values of variables corresponding to the measured process parameters and physical quantities) between the controllers. This approach allows implementing distributed and configurable automation solutions.

By using the AX OPC server (Object Linking and Embedding for Process Control, a collection of software technologies that provide a single interface for managing automation objects and technological processes), the controller is accessible over Ethernet network and can be used in software visualization packages.

The PROFINET technology can be implemented by connecting to the Ethernet interfaces of the AXC 3050 controller. The PROFINET controller is always available when connected via the 8-pin RJ45 connector of the X3 interface. The PROFINET functionality can be activated on the Ethernet interfaces X1, X2, X3 (see Fig. 1.4). By default, this function is disabled and can be activated in PC Worx.

Modbus TCP technology can also be implemented by connecting to the Ethernet interfaces of the AXC 3050 controller. This controller can act as a Modbus client, and can be configured as a MODBUS TCP server when using its corresponding function blocks.

At the bottom of the AXC 3050 controller there is communication interface with the local Axioline F bus (see Fig. 1.4) for connecting various modules.

Up to 63 devices can be connected to the controller. The actual number of devices depends on the total current consumption of all devices, which should not exceed the maximum current that the controller provides to the local bus.

Due to the Web-based management interface integrated into the controller, the user can visualize the status and diagnostic information from the controller in the browser.

The AXC 3050 controller is equipped with two USB interfaces (see Fig. 1.4).

The AXC 3050 controller has internal memory. It can be used for storing programs and configurations for a custom project. If the internal memory is insufficient for the created application, the AXC 3050 can work with external memory in the form of SD format flash memory (Secure Digital) or USB-drive.

The controller has 4 MB of internal memory for program storage, and 8 MB of memory for data storage; 128 kB is used for storing data after power off. The minimum controller cycle time is 1 ms, the number of control tasks performed simultaneously is 16.

The **ILC 151 GSM/GPRS controller** (Fig. 1.6) is a small, scalable, modular controller with integrated ports for connecting Ethernet and Interbus networks and an integrated quad-band modem.

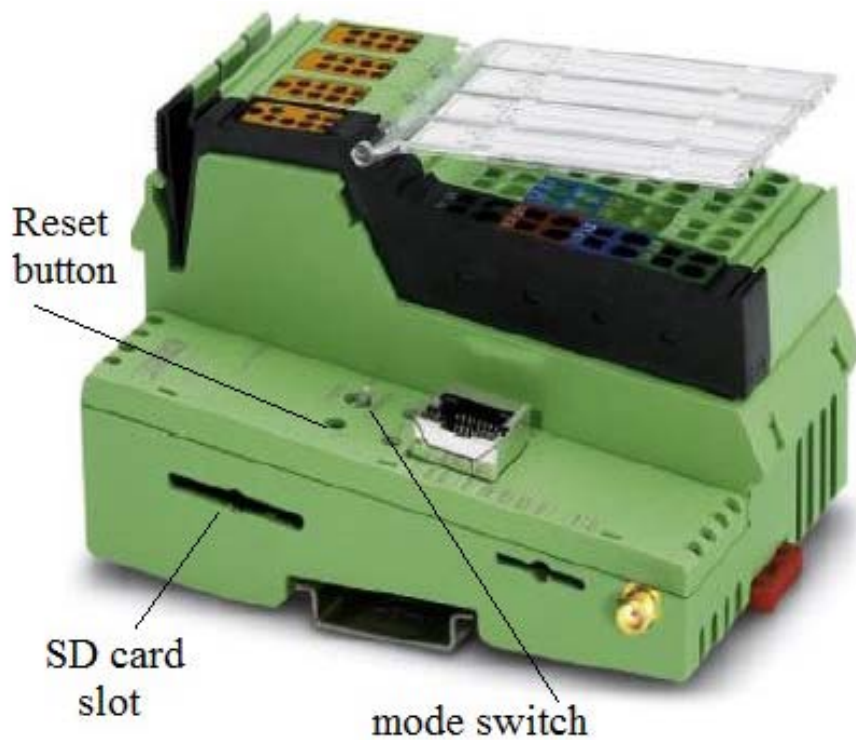


Fig. 1.6. ILC 151 GSM/GPRS controller

The controller can be configured and programmed in PC Worx using Ethernet connection in all five programming languages in accordance with IEC 61131-3. The connection to the Ethernet network is done via a twisted-pair cable, and TCP/IP and UDP/IP data protocols are used to access the controller.

Built-in communication functions allow you to exchange data over Ethernet network. Using function blocks IP_USEND (sending user data via TCP/IP protocol) and IP_URCV (receiving user data via TCP/IP protocol) in PC Worx, it is possible to organize data exchange (i.e., values of variables corresponding to the measured process parameters and physical quantities) between controllers. This approach allows implementing distributed and configurable automation solutions.

When using the AX OPC server, the controller is accessible over Ethernet network and can be used in various visualization packages.

The Modbus TCP communication protocol can be enabled via the Ethernet interfaces of the ILC 151 GSM/GPRS controller. The controller can act as a Modbus client.

The local Inline bus and the remote Interbus bus are activated via the appropriate connection. In this case, the developer can create a full Interbus system (maximum four levels of remote bus), using the controller as a distributed one.

The I/O level is connected to this controller via Interbus bus.

The controller can work with external memory in the form of SD format flash memory. It can be used for storing programs and configurations for a custom project. This external memory is optional and is not required for normal operation of controller.

GSM-modem, integrated in this controller, allows you to perform the following functions:

- sending and receiving SMS;

- control of remote controller via GPRS (General Packet Radio Service) or CSD (Circuit Switched Data);

- permanent GPRS connection for working with this protocol without executing programs.

The TCP/IP connection in the user project must use the appropriate function blocks. PC Worx has built-in communication function blocks MOBILE_CONNECT, SMS_SEND and SMS_RECEIVE to implement communication by sending SMS over the GSM network using the ILC 151 GSM/GPRS controller.

PC Worx has a standard GPRS_CONNECT function block for setting up a GPRS connection with the ILC 151 GSM/GPRS controller. The TCP/IP blocks allow the transfer of data over a GPRS connection using the TCP/IP protocol.

The ILC 151 GSM/GPRS controller has only two analog outputs.

In order to extend the capabilities of the controller, the following additional hardware modules can be used.

The **IB IL DO 4-ME module** (Fig. 1.7) has 4 digital outputs. The following abbreviations are used for the module labeling: IB – InterBus, IL – InLine, DO – Digital Output, ME – Machine Edition (designed for engineering, positioning and motion control).

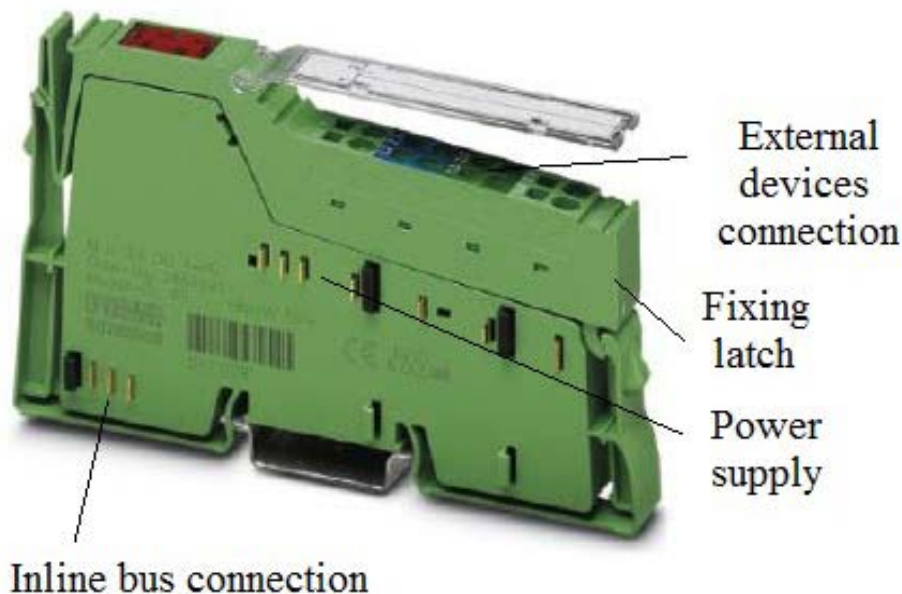


Fig. 1.7. The appearance of the IL DO 4-ME module of the ILC 151 GSM/GPRS Interbus controller bus

The module has 4 outputs 24 V, 500 mA DC. It is designed to output digital signals and has the following characteristics:

connection for 4 digital actuators;
connection of actuators by two- and three-wire technologies;
rated output current 500 mA;
total module current 2 A;
outputs are protected from short circuit and overload;
diagnostic and status indicators.

The **IB IL AI 4/U-PAC module** (Fig. 1.8) has 4 analog inputs. The following abbreviations are used for the module labeling: IB – InterBus, IL – InLine, AI – Analogue Input.

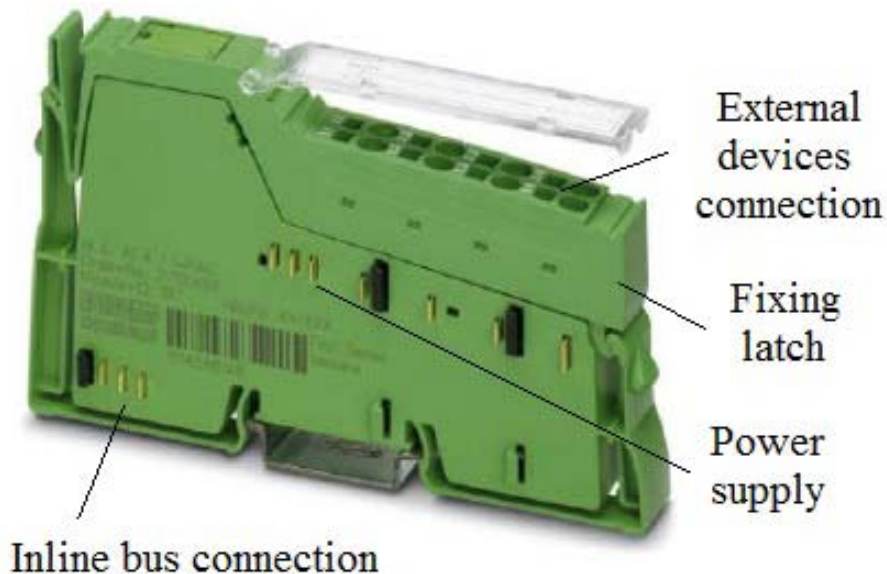


Fig. 1.8. The appearance of the IB IL AI 4/U-PAC module of the ILC 151 GSM/GPRS Interbus controller bus

The module has 4 inputs for connecting sources with varying voltage or current. Its characteristics are as follows:

4 analog, bipolar input channels;
connection of sensors by two-wire technology;
ranges of voltages from 0 to 10 V and from –10 V to +10 V;
generation of average values at the inputs;
updating data at inputs every 250 ms;
diagnostic and status indicators.

The **IB IL AO 2/UI-PAC module** (Fig. 1.9) has 2 outputs for outputting analog signals with varying currents and voltages. Its characteristics are as follows:

connection of actuators by two-wire circuit;
ranges of currents change from 0 to 20 mA, from 4 to 20 mA and from –20 to +20 mA;
ranges of voltage change from 0 to 10 V and from –10 to +10 V;
diagnostic and status indicators.

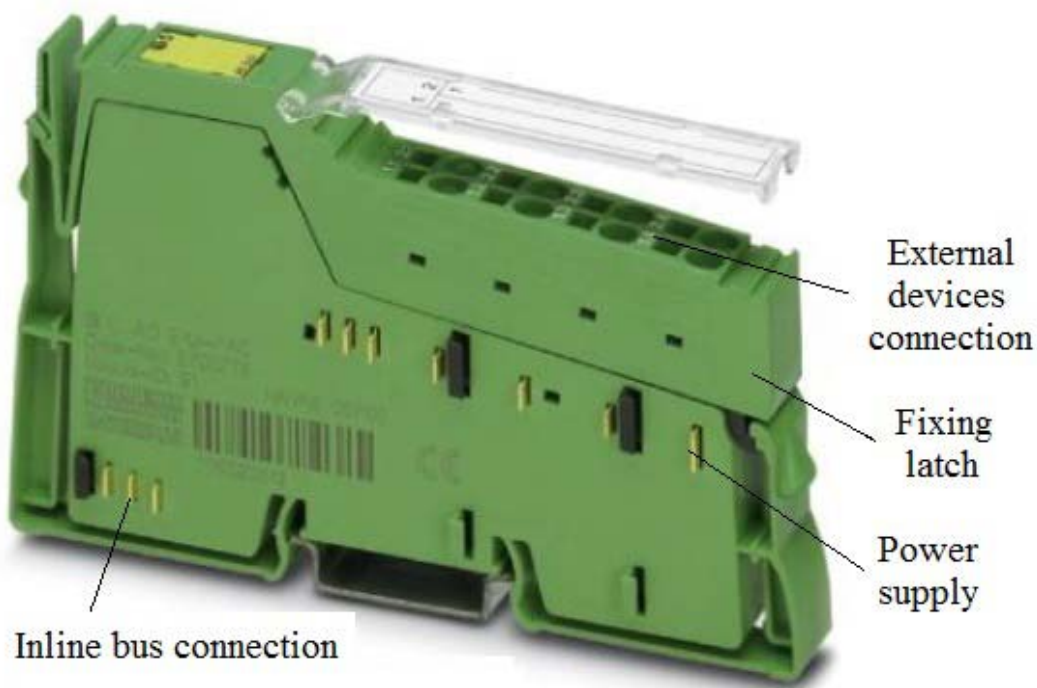


Fig. 1.9. The appearance of the IB IL AO 2/UI-PAC module of the Interbus ILC 151 GSM/GPRS controller bus

Bus connector AXL F BK PN Axioline F (Fig. 1.10) combines the PROFINET network with the Axioline F system. By using it up to 63 Axioline F devices can be connected to the existing PROFINET network. Its characteristics:

- 2 Ethernet ports and an integrated switch;

- typical cycle time for the local Axioline F bus is 10 ms;

- support of PROFINET RT (Real Time) and IRT (Isochronous Real Time);

- support of PROFIsafe standard (extension of PROFIBUS and PROFINET technologies, by which it's possible to create freely programmable security functions and exchange the necessary input and output data with safe input-output devices);

- the minimum PROFINET cycle time for RT and IRT is 250 ms;

- the execution time in the bus coupler is negligible;

- firmware can be updated;

- MRP (Media Redundancy Protocol) client, which ensures reliable operation of the network regardless of topology and even failing of some cables;

- settings are managed in the browser;

- I & M (Identification and Maintenance) functions;

- access of different users to the device through the network;

- diagnostic and status indicators.

The PC Worx 6.30 version allows you to configure PROFINET devices and contains an online description of the functions of devices with technical data and configuration file (if several versions of configuration files are available, you need to make sure that the version of the file corresponds to the version of the hardware and firmware).

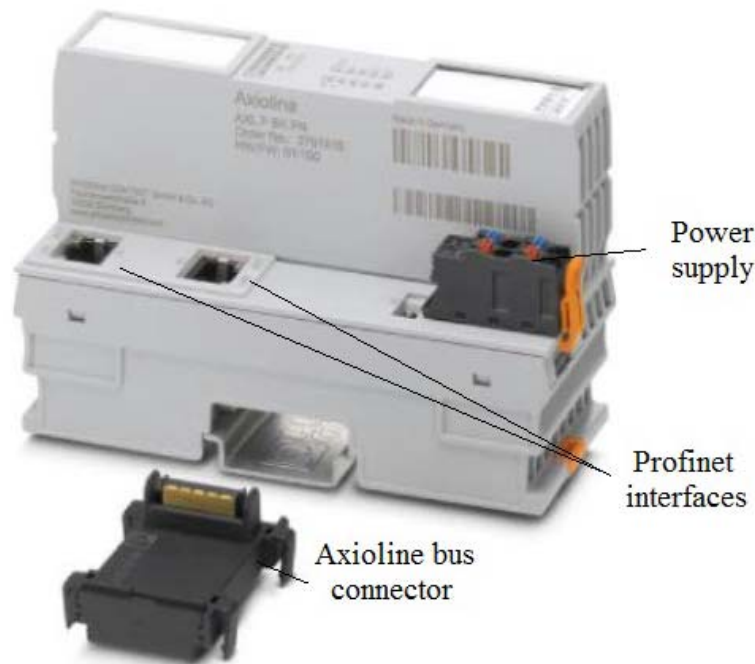


Fig. 1.10. Bus connector AXL F BK PN Axioline F for PROFINET

Modern I/O modules perform many functions that have previously been performed only by controllers. To perform these functions, the devices require setting during system installation, maintenance and parameterization. Therefore, it is necessary to have accurate and complete information about the devices: the type of functions to be performed, the number of inputs/outputs, the range of variables, the units of measurement, the default values that identify the device parameters, etc. PROFIBUS offers several approaches for the unified description of devices. The simplest is the use of GSD text files containing general and device-specific information. The GSD file is loaded into the PROFIBUS Configurator configuration tool and is used when it is installed.

The AXL F BK PN Axioline F has a web server that creates the required pages for web-based management and, if necessary, loads them into the browser. Web-based management can be used to access static information (technical data, MAC address) or dynamically changing information (IP address, status information).

Access to the device's web server can be obtained by using an IP address if it is properly configured. The home (main) web page of the device is downloaded after entering the address in the following format: `http://IP address` (for example, `http://192.36.133.58`).

Bus connector supports the Simple Network Management Protocol (SNMP) for controlling devices in IP networks using TCP and UDP protocols. SNMP supports routers, switches, servers, workstations, printers, modem racks, etc.

To extend the capabilities of the AXC 3050 controller, additional modules can be connected to the AXL F BK PN Axioline F bus coupler when building complex automation systems. The modules included in the TSL package are described below.

The **AXL F DI8/1 DO8/1 1H module** (Fig. 1.11) is used for digital input and output; it has 8 inputs (24 V DC) and 8 outputs (24 V DC), maximum current 500 mA, single-wire connection technology.



Fig. 1.11. Module AXL F DI8/1 DO8/1 1H

The module is designed to work as part of the Axioline F station and is used for acquiring and outputting digital signals. Time filters at the inputs can be adjusted to increase the immunity to noise. A 100 ms time filter makes it possible to implement a counter function with the maximum frequency 5 kHz. The outputs are protected against short circuits and overloads.

Characteristics of the input subsystem:

8 input signals;

24 V, 2,4 mA direct current;

connection of sensors using single-wire technology;

time filters can be tuned to three possible incremental values – less than 100 ms, 1000 ms and 3000 ms;

the maximum frequency of the inputs is 5 kHz.

Characteristics of the output subsystem:

8 output signals, 24 V, 500 mA direct current;

connection of actuators based on a single-wire circuit.

The minimum module update time is less than 100 ms. There are diagnostic and status indicators available.

The **AXL F AI4 U 1H** analog input module (Fig. 1.12), intended for use inside the Axioline F station, serves for the input of analog voltage signals and has an integrated power supply unit for sensors.

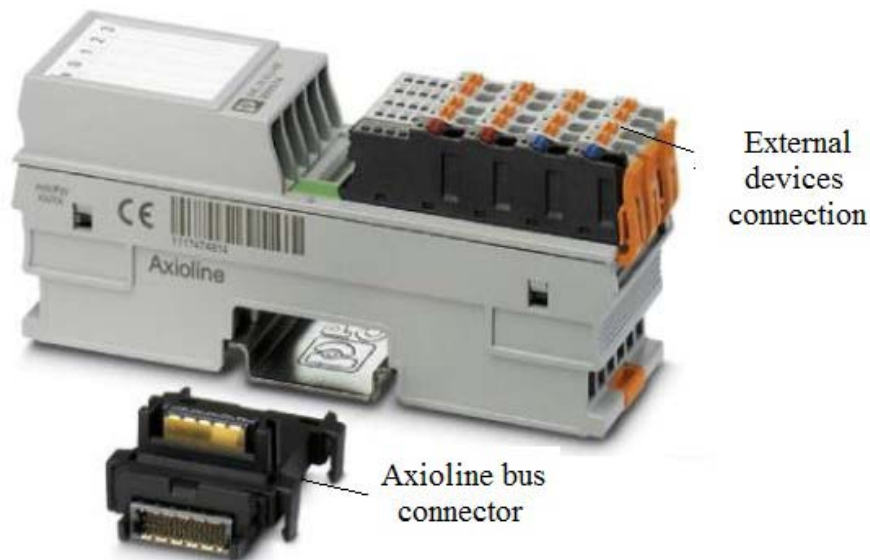


Fig. 1.12. Module AXL F AI4 U 1H

Module characteristics:

- 4 analog bipolar input channels for connecting voltage signals;
- two-, three- and four-wire connection of sensors;
- ranges of input voltages from 0 to 10 V, from -10 to $+10$ V, from 0 to 5 V and from -5 to $+5$ V;
- simultaneous polling of all channels using the simultaneous sampling function;
- high coefficient of transient attenuation between channels due to separate signal circuits;
- high immunity to interference caused by electromagnetic radiation;
- status and diagnostic indicators.

The AXL F AO4 1H analog output module (Fig. 1.13) is intended for use within the Axioline F station. It serves for the output of analog voltage and current signals.

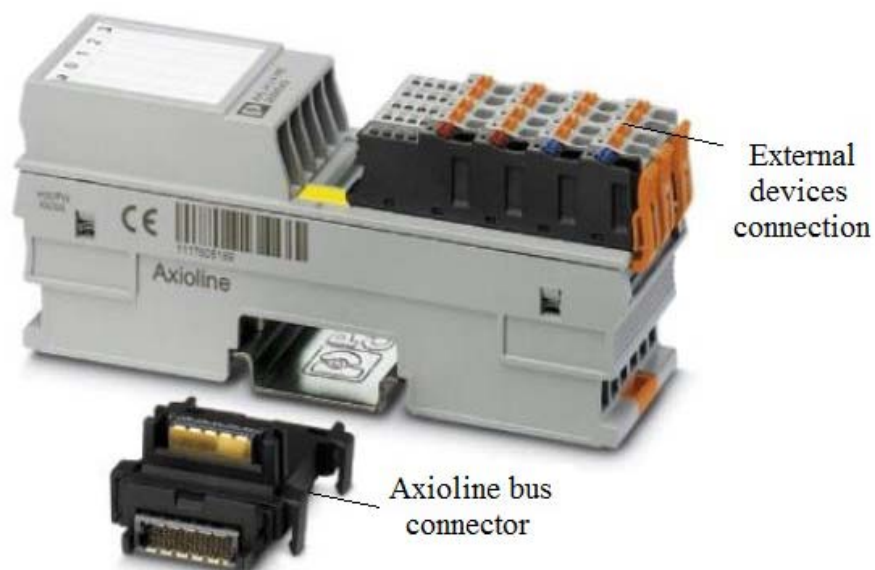


Fig. 1.13. Module AXL F AO4 1H

Module characteristics:

4 analog output channels for connecting voltage or current signals;

bipolar voltage outputs, unipolar current outputs;

two-wire connection of actuators;

voltage ranges from 0 to 10 V, from -10 to +10 V, from 0 to 5 V and from -5 to +5 V;

current ranges from 0 to 20 mA and from 4 to 20 mA;

outputs are protected from short circuits;

specification is saved in the device's own memory;

status and diagnostic indicators.

Siemens ET 200S is a peripheral I/O device that works with the PROFINET IO protocol. It consists of the IM151-3 PN interface module, power module, digital input module, two digital output modules inserted in the terminal module.

ET 200S distributed I/O system is a modular DP-slave system with flexible configuration for connection to signal processing in the central controller or in the PROFINET IO network. ET 200S has an IP 20 protection rating.

The IM151-3 PN interface module (Fig. 1.14) with an integrated two-port switch connects the ET 200S to the PROFINET IO network. The device name and backup data can be stored on a memory card. It supports Ethernet services such as ping, arp (two standard Windows command line utilities), Net diagnostics (SNMP)/MIB-2, LLDP (Link Layer Discovery Protocol) and has built-in LEDs for device status diagnostics. The network topology in which LLDP is used can be obtained from control computer by sequential traversing and polling of each device.

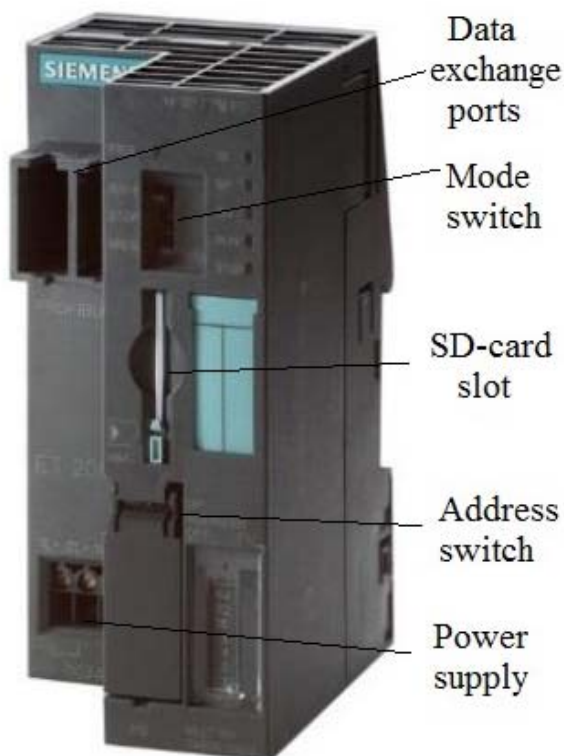


Fig. 1.14. Bus coupler Siemens IM151-3 PN HF PROFINET

Up to 63 modules can be connected to IM151-3 PN. Communication in isochronous real-time mode (IRT) can be organized. The minimum update time in IRT mode is 250 ms. The maximum length of the bus connection is 2 m.

The IM151-3 PN HF interface module allows updating the firmware using memory card or via the PROFINET IO bus. During operation the following interrupts are allowed:

- diagnostic;
- working process;
- for adding/removing modules;
- maintenance.

The maximum addressing space for I/O data is 256 bytes.

The module is connected to the bus from the rear panel; the maximum length of the bus is 2 m.

Terminal modules TM-P15S23-A1 and TM-E15S26-A1 (Fig. 1.15) implement electrical and mechanical communication of the I/O modules with the interface module and the end module.

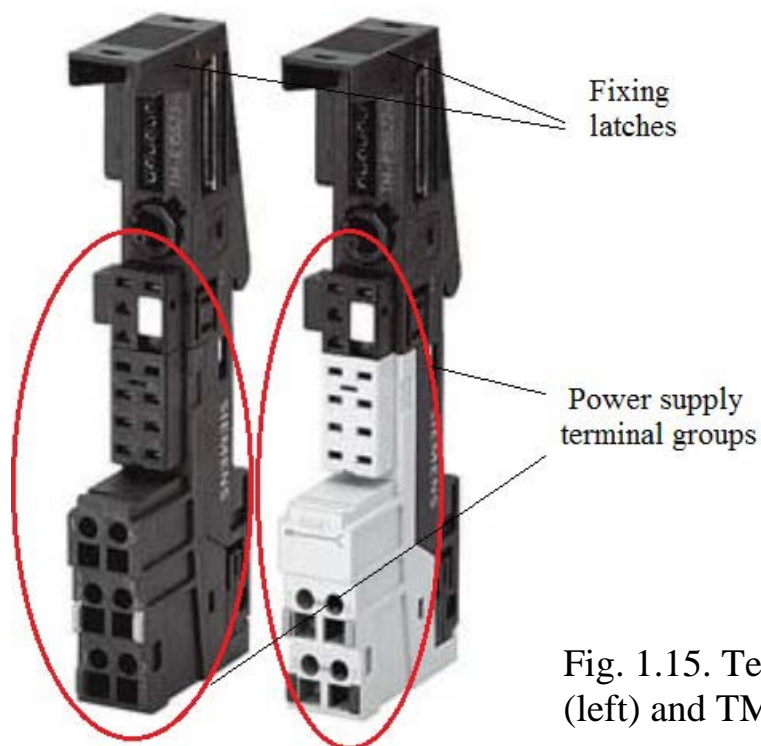


Fig. 1.15. Terminal modules TM-P15S23-A1 (left) and TM-E15S26-A1 (right)

The installed I/O module outputs signals to terminals 1 – 16, A3, A4, A7, A8, A11, A12, A15, A16.

The required terminal module is selected depending on the voltages necessary for the application. The AUX1 secondary bus is integrated into the terminal modules (it can be supplied with auxiliary supply voltage up to ~230 V or used as PE protective earth). Secondary bus can be used as a guiding protective bar or for supplying additional voltage.

Characteristics of TM-P15S23-A1:

- provision of new voltage group for the next TM-P terminal module;

three mounting types – mounted by screwing, spring retention and fast connect without dismantling;

solid firm AUX1 bus with electrical connection to the next group of voltages on the left;

access to the AUX1 bus voltage from terminals A4 and A8.

Characteristics of TM-E15S26-A1:

universal terminal module for all electronic modules 15 mm wide;

three mounting types – mounted by screwing, spring retention and fast connect without dismantling;

solid firm AUX1 bus with electrical connection to the next group of voltages on the left;

access to the AUX1 bus voltage from terminals A4, A8 and A3, A7.

The **ET 200S PM-E DC24V distributed I/O module** (Fig 1.16) monitors the feeding voltage at all electronic modules. The feeding voltage is supplied by means of the terminal module TM-P.



Fig. 1.16. Distributed module ET 200S for power supply I/O PM-E DC24V

Electronic modules, excluding 2DI AC120V ST, 2DI AC230V ST, 2DO AC24..230V/1A, can be used in the voltage group of the PM-E DC 24V power supply module. The current status of the power supply module is stored in the status byte in special memory area (Process Image of the Inputs, PII), because direct access to the I/O modules is not possible. It is updated independently or when the voltage failure diagnostics is enabled. The PM-E DC 24V power supply module can be used for self-disconnecting (fault-protected) modules. Temperature range was expanded for vertical installation: 0 – 55 °C.

The characteristics of the modules connected to the ET 200S PM-E DC24V are listed below.

Module 8DI DC 24V has 8 digital inputs; the nominal input voltage is 24 V DC; allows connecting two-wire sensors; isochronous operations are supported.

Module 8 DO DC 24V/0.5A has 8 digital outputs; the output current is 0,5 A per output (channel), the total current is 4 A; the nominal load voltage is 24 V DC; integrated short-circuit protection; isochronous operations are supported. The module is compatible with solenoid valves, DC contactors, indicating systems.

Module 2 AI STANDARD U has 2 outputs for measuring voltage with the following input ranges: ± 10 V with resolution capability 13 bits + sign bit, ± 5 V with resolution capability 13 bits + sign bit, 1 – 5 V with resolution capability 13 bits. There is also isolation from the load voltage L+. Voltage up to 5 V AC (Soft Start) is allowed. Temperature range was expanded for vertical installation: 0 – 50 °C.

Module 2 AO U has 2 outputs for output voltage with an output range of ± 10 V for resolution capability 13 bits + sign, 1 – 5 V with resolution capability 12 bits. There is also isolation from the load voltage L+. Temperature range was expanded for vertical installation: 0 – 50 °C.

FL SMCS 8TX-PN (Smart Managed Compact Switch) is an intelligent, controlled, compact Ethernet industry standard switch (Fig. 1.17). The device does not use a fan, but it has reduced power consumption and meets all industry standards in terms of electromagnetic compatibility, temperature conditions, mechanical load and ensuring the highest possible level of availability. Redundancy of the network topology can be created with the help of STP (Spanning Tree Protocol), RSTP (Rapid STP – advanced, accelerated implementation of the STP protocol), MRP. There are special stickers on the device for writing comments for each port.

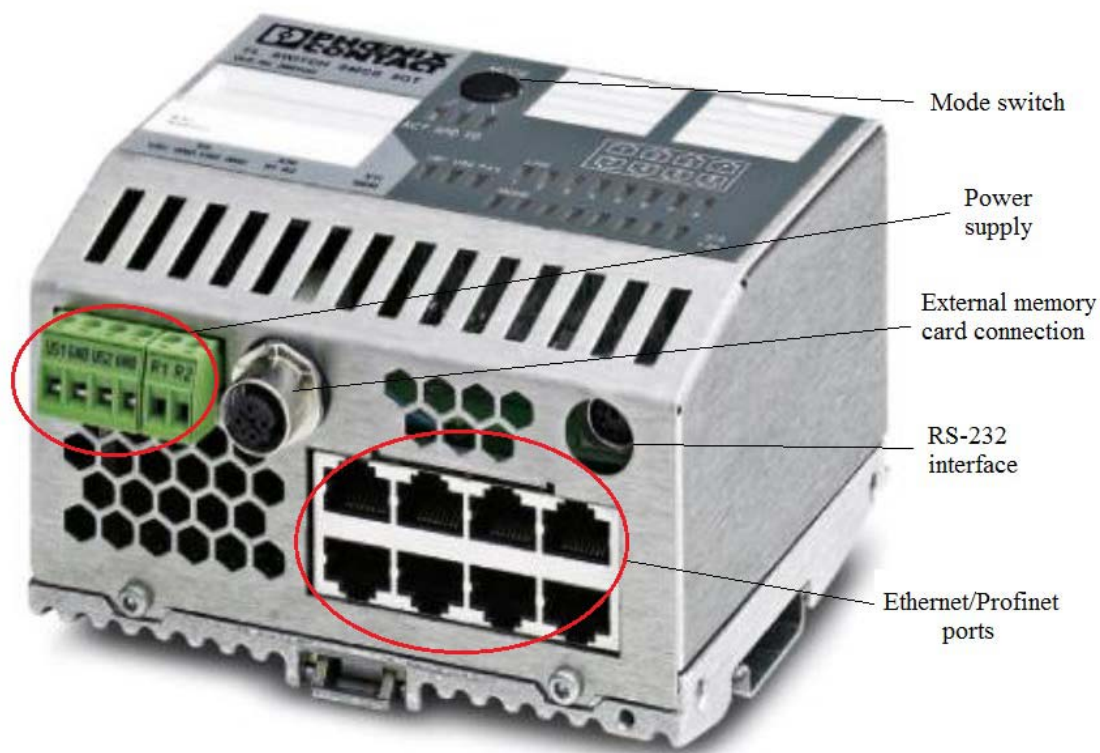


Fig. 1.17. FL SMCS 8TX-PN switch

Web server and SNMP agent are provided for diagnostics, maintenance and configuration via the network. Terminal access point can be used to transfer data over short distances.

Port mirroring can be used for monitoring traffic on network connections or as an important service function.

Features and applications of the switch:

- maximum performance of all ports up to 1 Gbit/s;

- increased network performance by filtering traffic (local traffic is not transferred to the external network and the amount of data in the network segments is reduced);

- simple expansion of the network and its configuration;

- unification of cable network segments with different data rates;

- automatic detection of data rates of 10, 100 or 1000 Mbit/s with autocrossing of RJ45 ports;

- flexible use of fiber optic modules in SFP slots (Small Form-factor Pluggable – industry standard for modular compact transceivers for data transmission). SFP modules are used to connect a network device card (switch, router) to an optical fiber or unshielded twisted pair;

- increased availability due to the use of redundant data paths with the minimum switching time and the use of the RSTP protocol, as well as the definition of the fast ring update mode (rapid detection of ring topology, expansion of RSTP protocol);

- support of various topologies and cell structures, as well as ring topologies with an emergency determination of the presence of a ring;

- configuring the switch in a web-based management system via SNMP or locally via the RS232 interface;

- port mirroring;

- determining the topology and use of LLDP protocol;

- address allocation by BootP protocols (Bootstrap Protocol), DCP (Discovery and Configuration Protocol, used for configuring device names and IP addresses in small and medium-sized applications without installing a DHCP server on PROFINET networks) or statically;

- Support of MRP protocol as a client (thus the MRP ring can be created using any switch ports);

- the switch can be used in the PROFINET environment;

- settings can be easily changed by using the Smart mode.

Wireless access point FL WLAN EPA (EPA – Ethernet Port Adapter) is a high-performance interface between the Ethernet and PROFINET cable and wireless networks (Fig. 1.18). Transparent protocol is used to transfer data to the second layer (L2) of the OSI model (Open System Interconnection, a seven-level open system interconnection model designed to standardize devices and data transmissions from different manufacturers), which allows easy integration with the networks that are based on industrial Ethernet (PROFINET and Modbus/TCP).

FL WLAN EPA complies with Class A PROFINET requirements and has a PROFI-safe profile to avoid data transmission failures.

FL WLAN EPA is certified for compatibility with wireless LANs of IEEE 802.11 b/g/n standard operating at 2,4 GHz. This standard is supported by stationary computers and laptops, mobile wearable devices, bar code scanners, RFID tags readers, weighing systems.



Fig. 1.18. Wireless Access Point FL WLAN EPA

WLAN (IEEE 802.11 b/g/n/a) is a standard wireless technology that provides reliable data transmission where there are metal objects and in environments with high levels of interference. WLAN has become the standard for wireless transmission of control data in automation networks. Factoryline wireless LANs are optimized as part of the standard for process automation.

1.3. "PROFIBUS" hardware module

The second hardware module contains three PROFIBUS I/O devices, a PROFINET proxy for I/O for PROFIBUS DP technology with an integrated switch and a wireless access point (Fig. 1.19). This module can be used for studying the PROFIBUS technology. It can be used in combination with the first hardware module or directly with a controller running on PROFINET or PROFIBUS technologies.

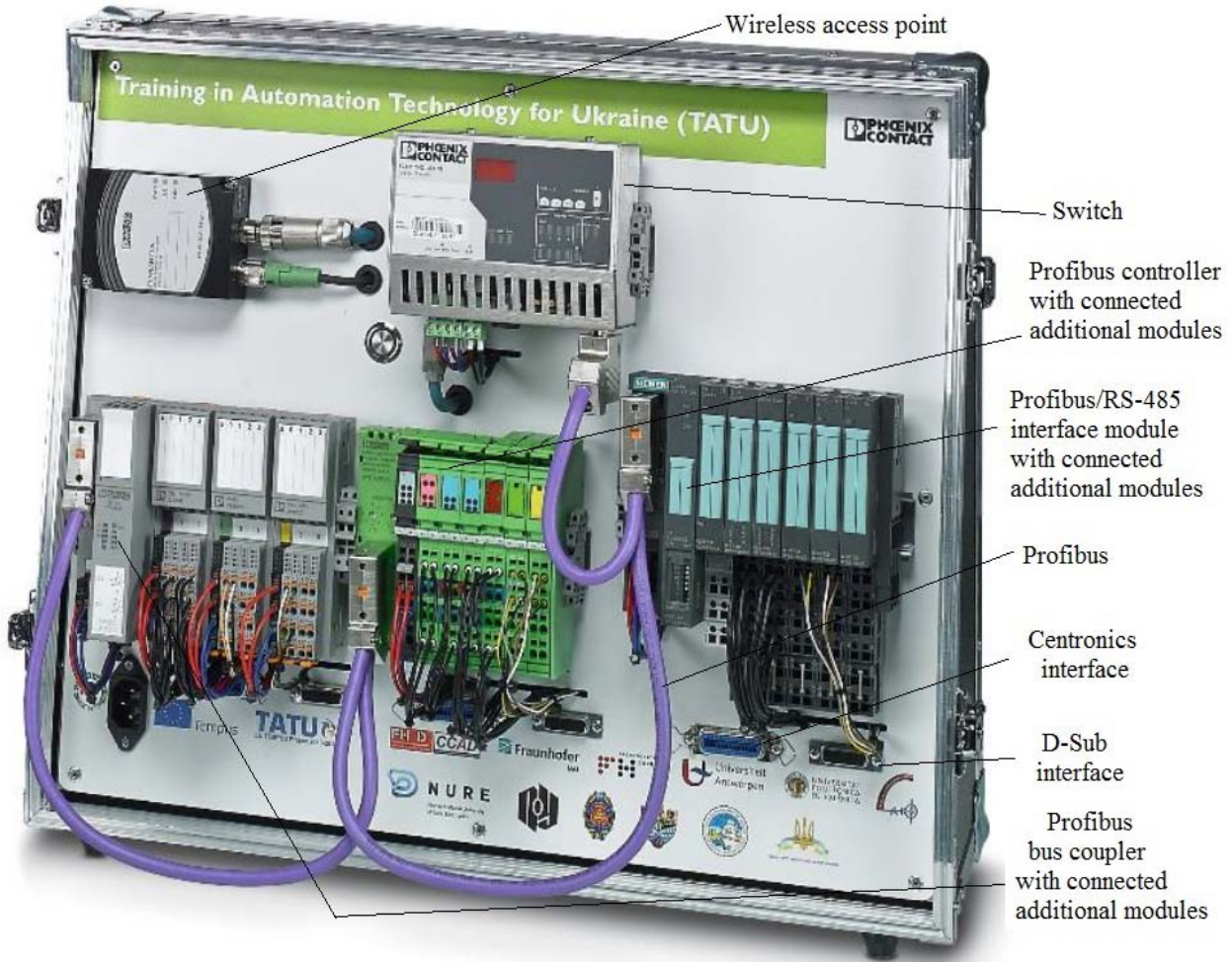


Fig. 1.19. Second hardware module TSL "PROFIBUS"

Fig. 1.20 shows the connection diagram of devices in the second hardware module.

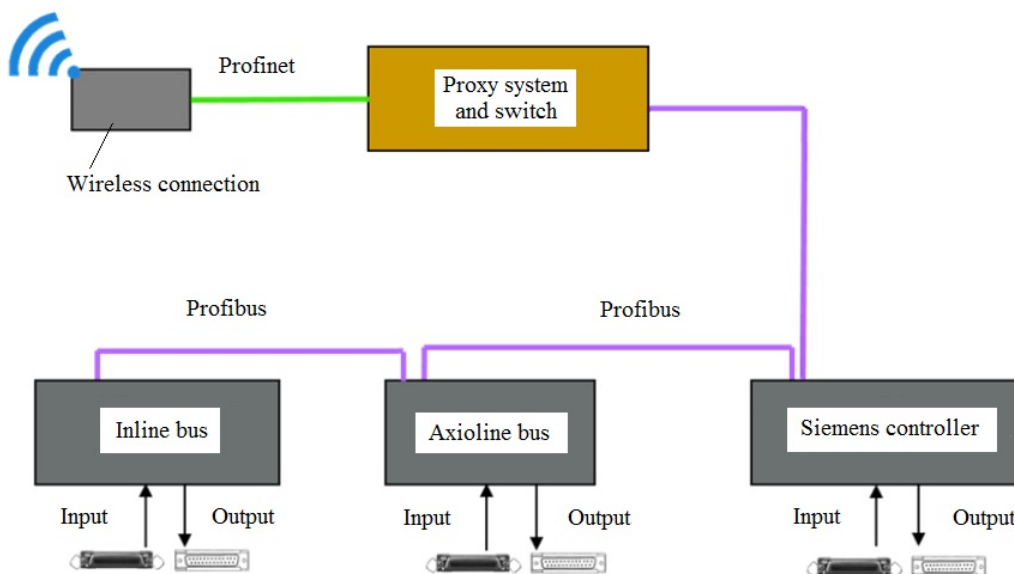


Fig. 1.20. Connection diagram of devices in the second hardware module TSL "PROFIBUS"

In the second hardware module the following devices are installed.

The FL NP PND-4TX PB proxy system with built-in 4-port RJ45 switch (Fig. 1.21) has the following characteristics:

- support of PROFINET I/O devices;
- Ethernet 10/100Base-T (X) for twisted pair;
- built-in PROFINET IO proxy for PROFIBUS;
- support of negotiation with Class B PROFINET;
- built-in managed four-port switch;
- class 1 PROFIBUS DP master device;

PROFIBUS DP master connection with the speed of up to 12 Mbps (connection via copper cable and RS-485 standard);

full configuration using PC Worx;

connection of the PROFIBUS DP system to the PROFINET IO controller;

use in small control devices for easy integration of existing PROFIBUS DP solutions in the PROFINET network.

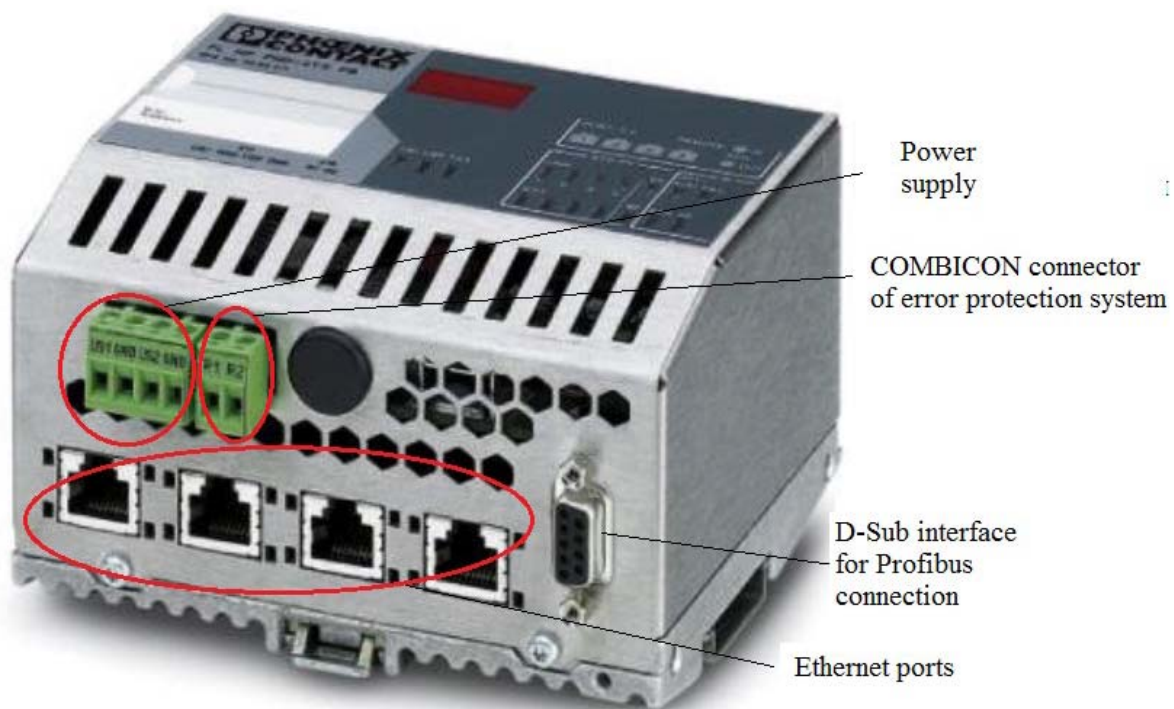


Fig. 1.21. Proxy system FL NP PND-4TX PB with built-in switch

The PROFIBUS DP proxy system can be configured with PC Worx.

Parameterized memory simplifies the replacement of devices if the PROFINET IO controller does not support automatic device discovery based on the topology. Port mirroring can be used for packet analysis.

The second hardware module has the **wireless access point FL WLAN EPA**, similar to the first hardware module.

Bus coupler AXL F BK PB Axioline F (Fig. 1.22) connects the PROFIBUS network and the Axioline F. It can connect up to 63 Axioline F devices to the existing PROFIBUS network.

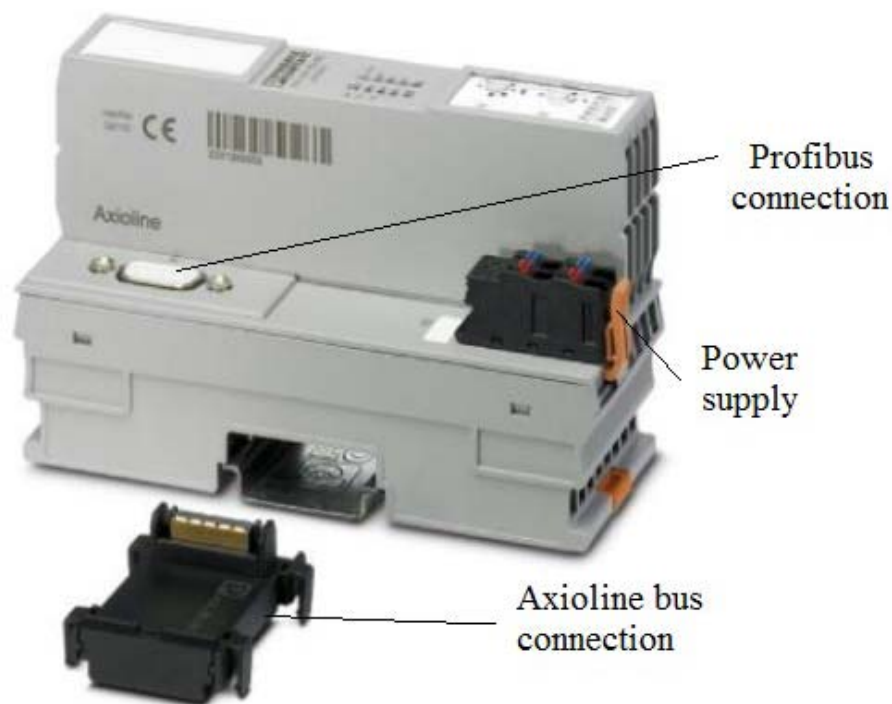


Fig. 1.22. Axioline F bus coupler for PROFIBUS DP

Main characteristics of the bus coupler:

the connection is setup using 9-pin D-SUB Female connectors;

physical interface RS-485 is used;

in case master devices of classes 1 and 2 are installed, PROFIBUS DP-V1 mode is used (the DP-V0 sub-standard is used for cyclic data exchange and diagnostics, DP-V1 is used for acyclic and cyclic data exchange and alarm processing, DP-V2 is used for mode synchronization and data exchange between the subordinated units of equipment);

autodetection of the data transmission speed in the range of 9,6 kbit/s – 12 Mbit/s is performed;

coded rotary switches are used for setting PROFIBUS addresses;

support of dynamic configuration;

PROFIBUS addresses in the range 0 – 126 are supported;

the device description is stored in the GSD file;

I & M (Identification and Maintenance) functions.

The **Axioline F** bus coupler for PROFIBUS DP has the same plug-in modules as the **AXL F BK PN** bus connector for PROFINET in the first hardware module.

The **IL PB BK DI8 DO4/EF-PAC bus coupler** for PROFIBUS DP (Fig. 1.23) has 8 digital inputs and 4 digital outputs. It is necessary for connecting PROFIBUS DP devices with the Inline installation system. In addition, it is used for collection and output of digital signals. The bus coupler allows connecting 61 Inline devices to any point in the existing PROFIBUS DP network. The bus coupler and Inline series devices form one station with a maximum of 63 devices connected to the local bus. The input and output devices of the bus coupler are the first and second local bus devices.



Fig. 1.23. Inline bus coupler for PROFIBUS DP

Characteristics of the bus coupler:

connection to PROFIBUS is carried out with the help of 9-pin D-SUB Female connectors;

physical interface RS-485 for PROFIBUS is used;

up to 61 additional Inline-devices can be connected;

up to 16 PCP devices can be connected (Port Control Protocol, which in IP networks controls the redirection of incoming data packets in the router). The router filters data packets or works in accordance with the NAT (Network Address Translation) mechanism, allowing IP-networks to resolve IP-addresses of transit packages). This makes it easier to manage and configure network traffic so that systems located behind NAT devices or firewalls are reachable via Internet (i.e. they can also function as servers), which is necessary for many applications;

PROFIBUS DP/V1 class 1 and 2 master devices can be used;

autodetection of the data transmission speed in the range of 9,6 kbit/s – 12 Mbit/s is performed;

coded rotary switches are used to set PROFIBUS addresses;

PROFIBUS addresses in the range 0 – 126 are supported;

the device description is stored in the GSD file;

I & M (Identification and Maintenance) functions;

there are 8 digital inputs and 4 digital outputs;

autodetection of the local bus speed (500 kbit/s or 2 Mbit/s);

calling of IO-Link firmware – version 2.0 and higher (IO-Link is the first standardized input/output technology that allows data exchange at all levels – from control system to the lowest level of automation).

The device is approved for use in the zone of two potentially explosive areas. The PROFIBUS bus coupler configures the station and controls the data exchange with the PROFIBUS master. It also supports the power supply to the connected Inline terminals. PROFIsafe I/O modules can also be used.

The bus coupler can work with the same plug-ins as the ILC 151 GSM/GPRS controller discussed in 1.2.

The IL PB BK DI8 DO4/EF-PAC Inline bus coupler has only 2 analog outputs that are connected to the terminal.

The IM151-1 HIGH FEATURE interface module (Fig. 1.24) has the following characteristics:

It connects the ET200S I/O peripheral device to the PROFIBUS DP network via the RS 485 interface;

Simatic 7 limits are eliminated for the maximum length of parameter (usually 244 bits);

the maximum address space is 244 bytes for inputs and 244 bytes for outputs;

it can be used as a slave device DPV0 or DPV1;

the maximum number of serviced modules is 63;

the maximum length of the bus is 2 m;

support of the service option and using of status byte for power modules;

it can be synchronized with PROFIBUS DP cycles;

firmware can be updated via PROFIBUS DP using Siemens Simatic Step 7 HW Config;

the data exchange is performed in accordance with Siemens Safety-related I-slave-I-slave technology for PROFIBUS DP;

it can be used as DPV1 for Y-switching technology;

ProfiSafe modules are used.

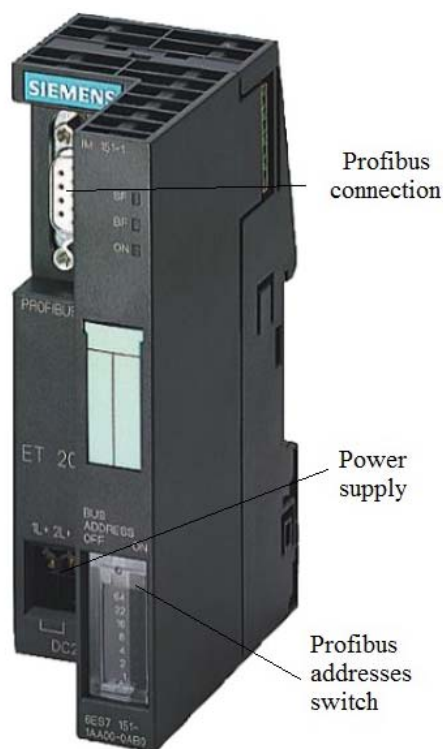


Fig. 1.24. Interface module IM151-1 HIGH FEATURE

This interface module works with the same plug-in modules as the Siemens PROFINET IM151-3 PN HF bus coupler described in cl. 1.2.

1.4. Hardware Module "Process Modeling"

The third hardware module (Fig. 1.25) is intended for modeling of technological processes. Models should be developed in CoDeSys of 3.x versions and loaded into the EtherCAT EC2250 controller. The graphical interface can be seen in the browser by entering the link <http://xxx.xxx.xxx.xxx:8080/webvisu.htm>, where xxx.xxx.xxx.xxx is the IP address of the EtherCAT EC2250 controller. For visualization it's possible also use the built-in graphic terminal Ethernet ET1007 WT. This module also allows you to study the programming of controllers in CoDeSys of 3.x versions.

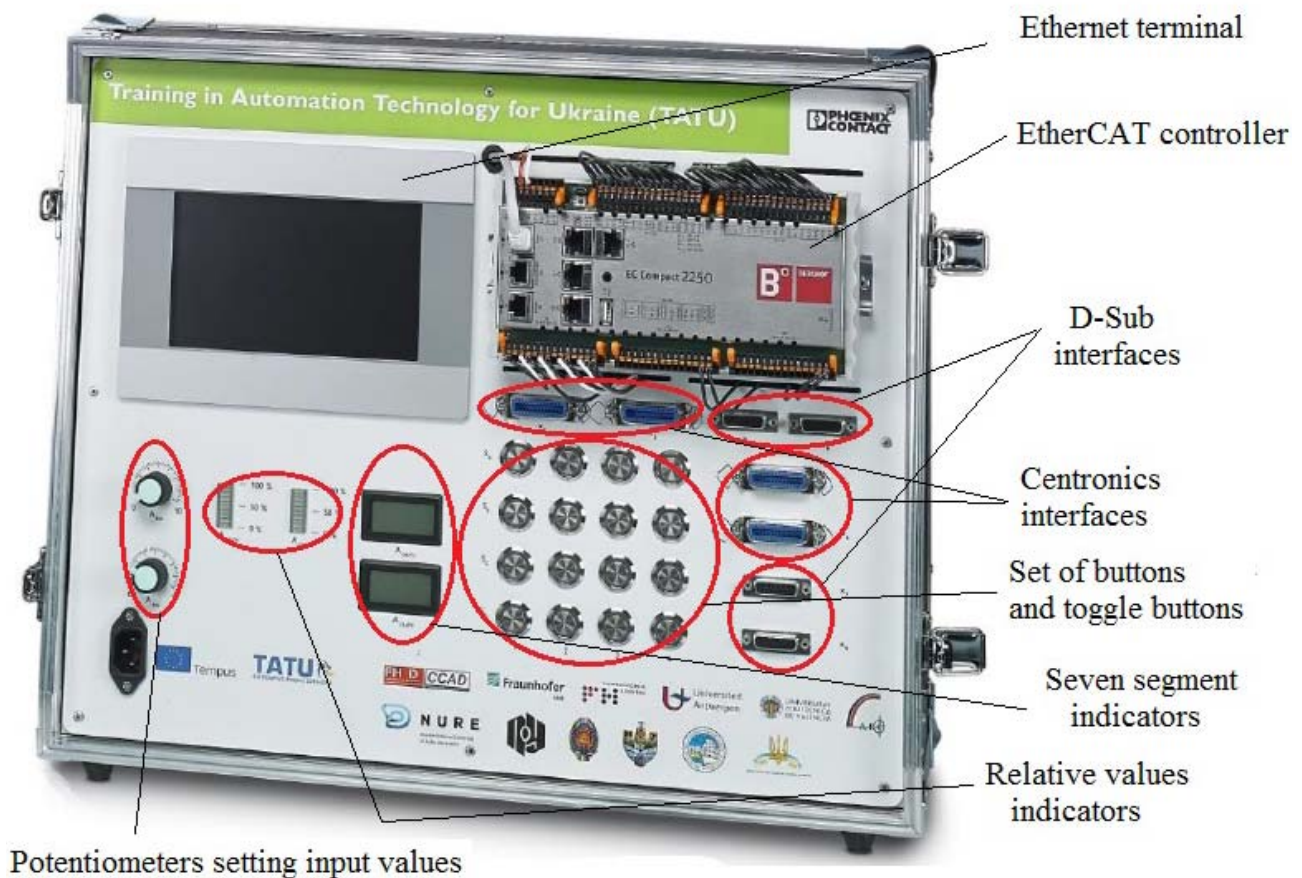


Fig. 1.25. The third hardware module TSL "Process Modeling"

The third hardware module has various analog and digital inputs and outputs, as well as buttons that can be used for testing and simulating the operation of a highly complex control system. Analog inputs/outputs are located on the left and are connected to terminal X7 (Fig. 1.26). Each button has a built-in LED that is connected to the digital output. Nonlocking keys in rows SA and SB are connected to terminal X3. Keys in rows SC and SD are the usual locking keys, connected to terminal X4. Thus, the devices and technologies implemented in this module allow studying the material of training modules 1, 4 and 5.



Fig. 1.26. Location of physical inputs and outputs

Fig. 1.27 shows the connection diagram of devices in the third hardware module.

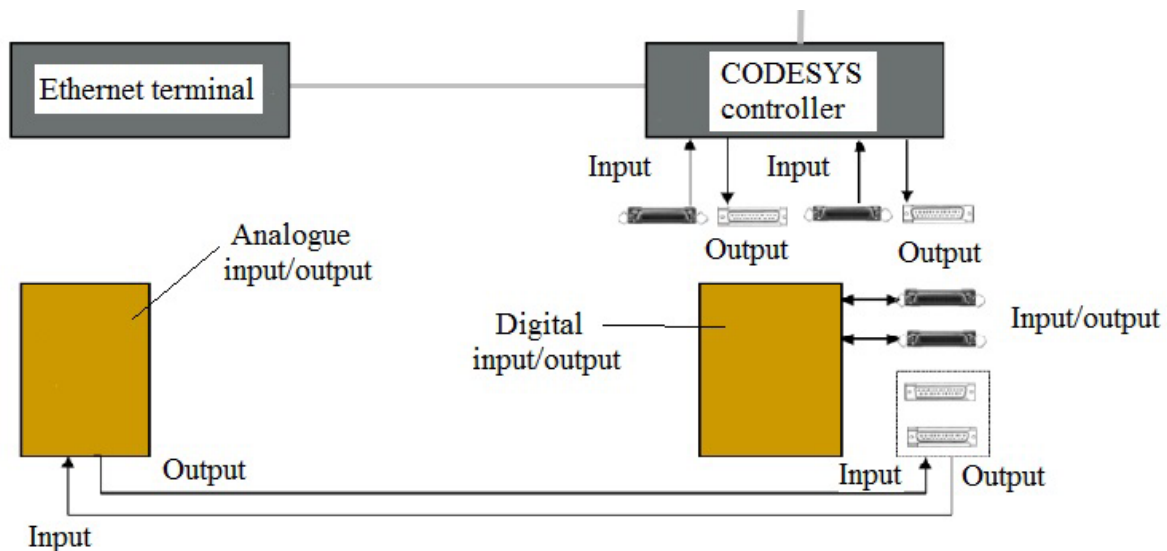


Fig. 1.27. Connection diagram of devices in the third TSL hardware module "Process Modeling"

The connection table is shown in Fig. 1.28.

In the third hardware module the following devices are installed.

Compact controller EC2250 EtherCAT (Fig. 1.29) has a short cycle time and is designed for hard real-time systems. The controller uses a high-performance scalable ARM processor with Cortex TM-A9 core and frequency of 800 MHz.

EtherCAT is an Ethernet based bus standardized by SEMI (Semiconductor Equipment and Materials International), IEC and ISO. The principle of operation of EtherCAT differs significantly from other Ethernet based solutions. In the EtherCAT network a single Ethernet packet includes the input and output data of several devices. Real bandwidth usage can reach more than 90 %. EtherCAT is much faster than traditional buses and industrial Ethernet based solutions. The typical EtherCAT cycle time is 50 ... 250 ms, while in traditional buses 5 ... 15 ms are required for each update.

Hardware module 3, connector X ₃ (8 digital outputs, 8 digital inputs)			Hardware module 3, connector X ₄ (8 digital outputs, 8 digital inputs)		
Signal name	Contact number in connector X ₃	Visualization/control element	Signal name	Contact number in connector X ₄	Visualization/control element
Output 0	1	LED S _A 1	Output 0	1	LED S _C 1
Output 1	2	LED S _A 2	Output 1	2	LED S _C 2
Output 2	3	LED S _A 3	Output 2	3	LED S _C 3
Output 3	4	LED S _A 4	Output 3	4	LED S _C 4
Output 4	5	LED S _B 1	Output 4	5	LED S _D 1
Output 5	6	LED S _B 2	Output 5	6	LED S _D 2
Output 6	7	LED S _B 3	Output 6	7	LED S _D 3
Output 7	8	LED S _B 4	Output 7	8	LED S _D 4
Input 0	13	Button S _A 1	Input 0	13	Button S _C 1
Input 1	14	Button S _A 2	Input 1	14	Button S _C 2
Input 2	15	Button S _A 3	Input 2	15	Button S _C 3
Input 3	16	Button S _A 4	Input 3	16	Button S _C 4
Input 4	17	Button S _B 1	Input 4	17	Button S _D 1
Input 5	18	Button S _B 2	Input 5	18	Button S _D 2
Input 6	19	Button S _B 3	Input 6	19	Button S _D 3
Input 7	20	Button S _B 4	Input 7	20	Button S _D 4
0 V	11, 12, 23, 24		0 V	11, 12, 23, 24	
24 V	9, 10, 21, 22		24 V	9, 10, 21, 22	

Fig. 1.28. Connection of digital inputs and outputs (keys S_A1, S_A2, S_A3, S_A4, S_B1, S_B2, S_B3, S_B4 are locking, keys S_C1, S_C2, S_C3, S_C4, S_D1, S_D2, S_D3, S_D4 are nonlocking)

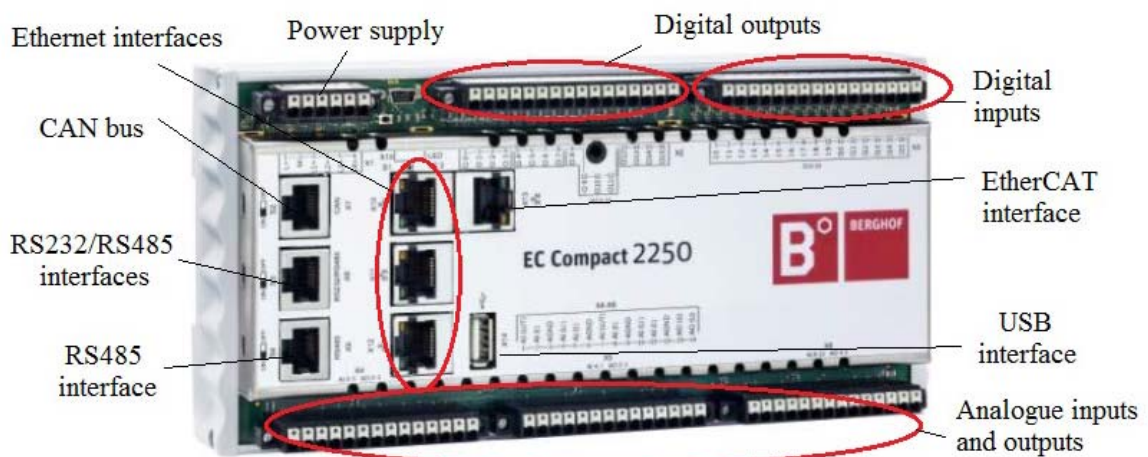


Fig. 1.29. Compact controller EC2250 EtherCAT

A lot of digital and analog inputs and outputs, as well as built-in programming languages for working with version 3 CoDeSys, are located in the controller itself. Together with the CoDeSys SoftMotion package, complex applications for controlling multi-axis actuators can be operated. The device integrates the following communication interfaces: Ethernet, EtherCAT, CAN, RS 232 and RS 485. PROFINET, BACnet and Modbus protocols are available. EC2250 uses CoDeSys, Web Visu and the corresponding Ethernet terminals for visualization.

CoDeSys SoftMotion is a functional set of motion control tools built into the development environment and the CoDeSys execution system. It controls motions from the simplest displacements along one axis to the complex multidimensional interpolation of advanced systems with numerical program control (NPC).

The EC2250 controller has 32 digital inputs/outputs and 18 analog inputs/outputs and supports the following:

- programming, visualization, communication, motion control in CoDeSys and SoftMotion;

- master modes in EtherCAT and CANopen technologies;

- serial data transfer interfaces;

- use of extension cards.

In the third hardware module the EC2250 controller is used primarily for process modeling.

Ethernet terminal ET1007 WT (Fig. 1.30) was designed specifically for visualization in CoDeSys – it has simple control and high speed of operation. It can work with various equipment, perform CoDeSys web visualization or target CoDeSys visualization with the help of VNC protocol (Virtual Network Computing – system of remote access to resources (for example, desktop) using the RFB (Remote Frame Buffer) protocol. It does not matter if the visualization data comes from Berghof controller or from any other controller that works with CoDeSys. The performance and memory capacity of the terminal allow quick change of the pages of the visualization project.



Fig. 1.30. Ethernet terminal ET1007 WT (as an example, arbitrarily selected controls and visualization tools that can be implemented in a specific project, are shown)

The terminal receives data over the Ethernet interface at the speeds of 10/100 Mbps. Entering IP address and HTTP address ensures a simple and quick connection to the CoDeSys controller (visualization server). The Ethernet terminal visualizes the file running on the EC2250.

The terminal has high brightness (more than 400 cd/sq. m.).

To operate the target visualization the "Target-Visualization" option located on the "General" tab of the CoDeSys system must be activated in the system settings. If allowed in the target file, this option can be turned on or off by the user.

1.5. Switching of hardware modules

To ensure physical connections of devices of hardware modules, Centronics IEEE 488 connector with 24 outputs is used (Fig. 1.31). Each set of 8 digital inputs and outputs is connected to the pins of connector in accordance with the requirements of IEEE 488.



Fig. 1.31. 24-pin Centronics IEEE 488 connector

Also a 15-pin D-Sub connector is used (Fig. 1.32).



Fig. 1.32. 15-pin D-Sub connector

Up to 6 analog signals are connected to the D-Sub connector. The analog-to-digital conversion is performed at resolution capacity of 12 bits. The sample rate is 0,5 kHz.

Only the third hardware module can be connected to other modules by standard cables, as their inputs and outputs are mounted symmetrically. Fig. 1.33 shows an example of using Centronics connectors for connecting hardware modules. Fig. 1.34 shows an example of using D-Sub connectors for connecting hardware modules.

Hardware modules 1 and 2 (8 digital inputs,
8 digital outputs)

Hardware module 3 (8 digital outputs,
8 digital inputs)

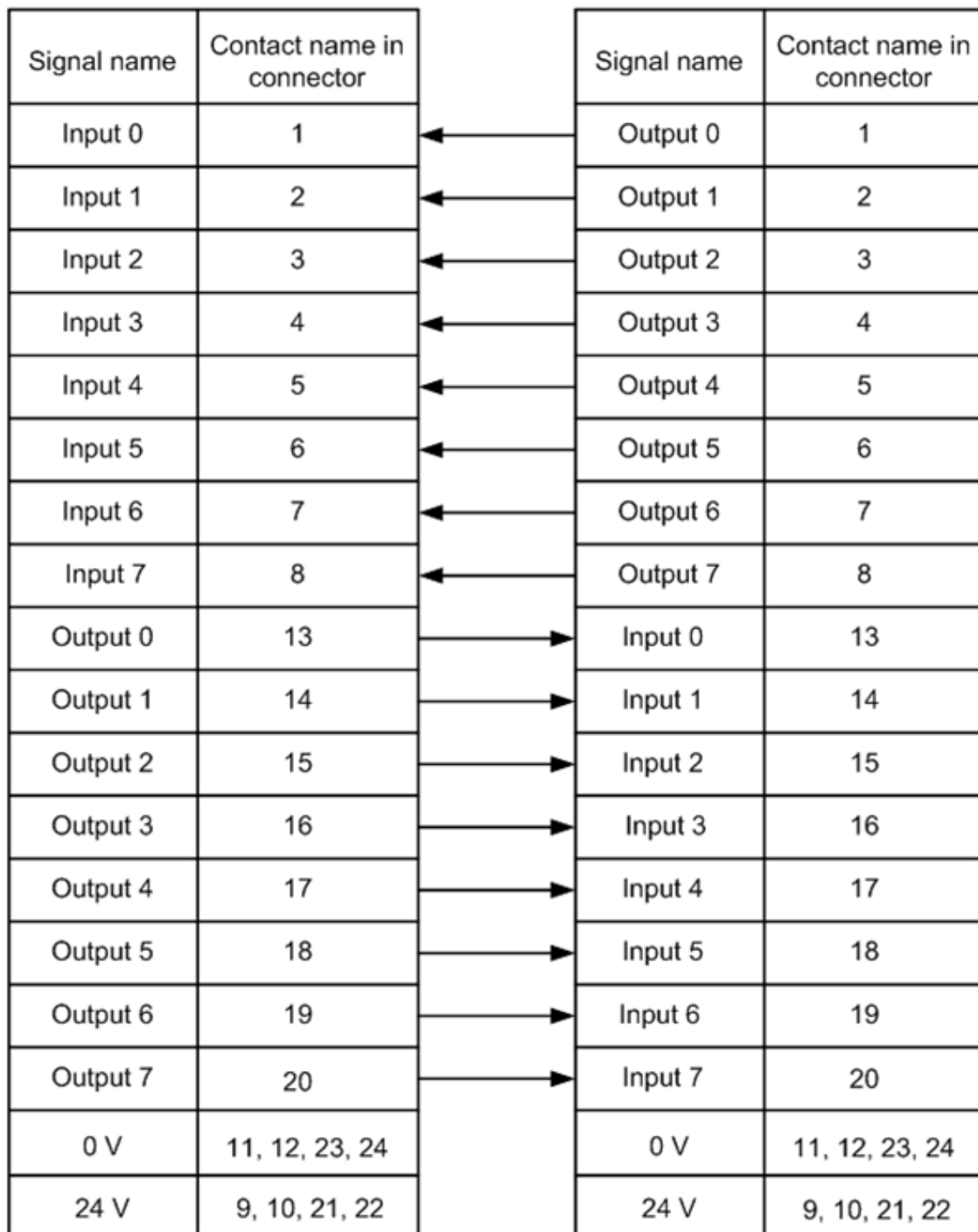


Fig. 1.33. Contacts pin map when connecting hardware modules using Centronics connectors

Hardware modules are connected to each other using standard cables packed with TSL in accordance with the diagram shown in Fig. 1.1. These cables can also be used for connecting other external devices that do not have IP addresses. Devices that have IP addresses are connected to RJ45 ports of Ethernet switches using a standard twisted-pair cable of category 5 and higher.

Hardware modules 1 and 2 (2 analogue inputs, 4 analogue outputs)

Hardware module 3 (2 analogue outputs, 4 analogue inputs)

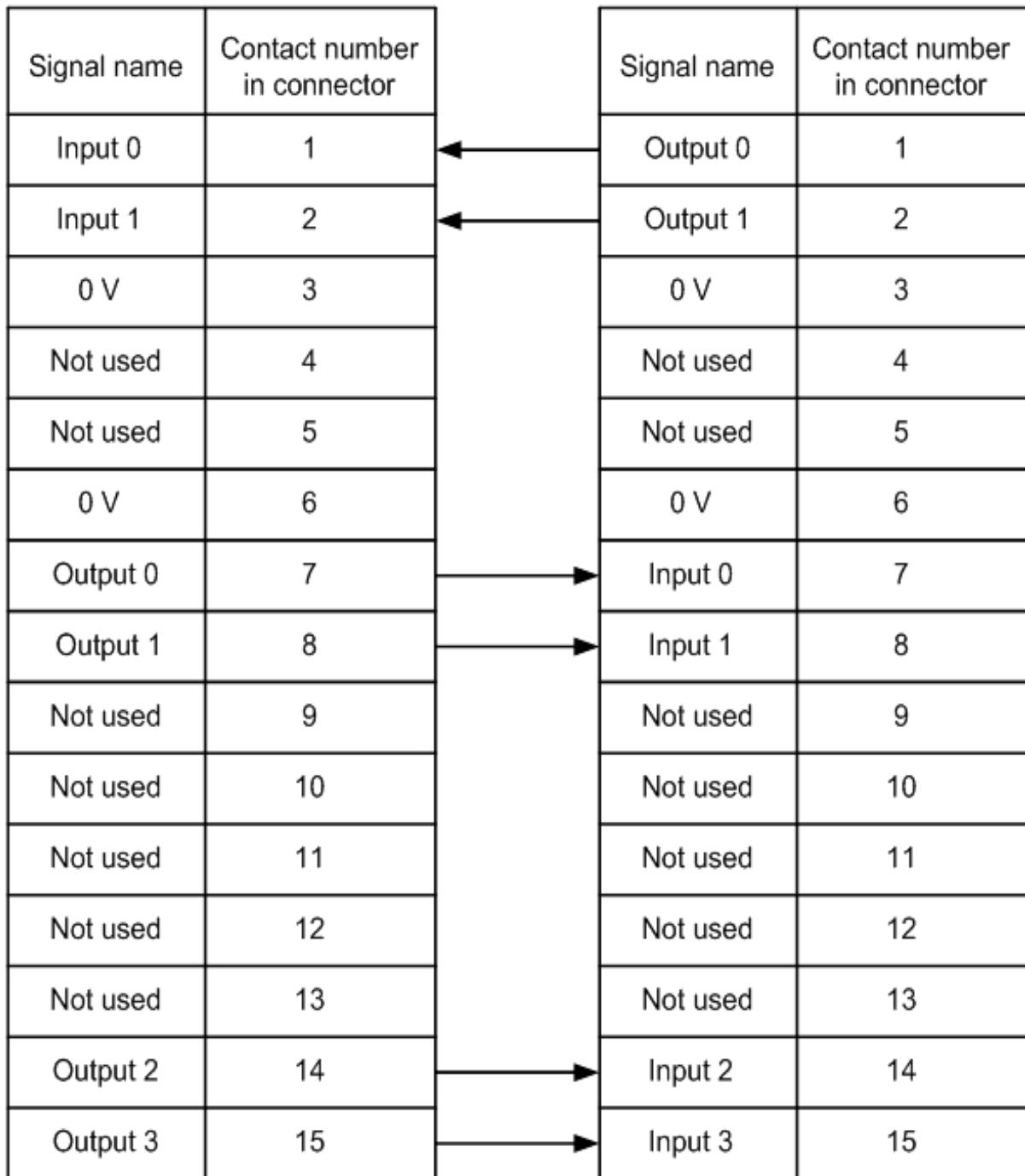


Fig. 1.34. Contacts pin map when connecting hardware modules using D-Sub connectors

In order to work through the tasks within this training manual, two standard cables are used (Fig. 1.35) for connecting the first and the third hardware modules ("Programmable Controllers and PROFINET" and "Process Modeling").

In the first hardware module (see Fig. 1.2) the slots located under the ILC 151 controller or under the AXC 3050 controller are used, and in the third hardware module (see Fig. 1.25) – the slots X4 and X7 are used.

After the modules are switched, the controls located in the third hardware module become available.



Fig. 1.35. Cables for connecting hardware modules

Test questions

1. What hardware modules does the TATU Smart Lab training laboratory facilities consist of?
2. What data transfer technologies are used in the TATU Smart Lab training laboratory facilities?
3. What are the bus connectors used for?
4. What local buses are used for connecting peripherals to controllers?
5. What interfaces are used for connecting peripherals to the TATU Smart Lab training laboratory facilities?
6. What are the differences between the controllers used in the TATU Smart Lab training laboratory facilities?
7. How can wireless data transmission in the TATU Smart Lab training laboratory facilities be organized and what are the differences of these methods?
8. What software development tools can be used when working with the TATU Smart Lab?
9. What hardware is used in the TATU Smart Lab training laboratory facilities for simulating the operation of external equipment?
10. What is the purpose of the Ethernet terminal ET 1007 WT?

2. CONTROLLERS SOFTWARE

2.1. AUTOMATIONWORX Software Suite environment

The AUTOMATIONWORX Software Suite environment was developed by Phoenix Contact and is a package of coordinated programs for configuring and programming controllers, diagnostics of I/O modules, as well as tools for creating controllers Web-interface.

AUTOMATIONWORX Software Suite includes the following components:

- PC Worx;
- PC Worx Express;
- Config+;
- Diag+;
- Diag+ NetScan;
- Visu+;
- WebVisit;
- AX OPC Server.

PC Worx is universal software for project development and is suitable for all Phoenix Contact process control devices.

Depending on the number of supported process inputs/outputs, one of two software versions can be used: PC Worx BASIC or PC Worx PRO.

The software supports the following functions:

- compiling network topologies using the Drag & Drop method;
- controlling the device using the integrated configuration and addressing PROFINET (the standard for building automation networks based on the Ethernet network technology with real-time mode and using the TCP/IP protocol), PROFIBUS (the previous generation technology that uses the principle of master/slave and data transmission rate from 9,6 kbit/s to 12 Mbit/s), INTERBUS, as well as Modbus-TCP;

- testing programs in the early stages using the simulation function;

- diagnostics of all components of the system;

- multiple users with password protection;

- conducting several projects and comparing projects.

The software contains:

- catalogs of devices and modules;

- controller change assistant in case of the change of models.

PC Worx Express is a free, simplified version of PC Worx. It supports modular mini controllers of 100 and 1000 series, as well as programmable controllers PC Worx SRT with programming languages ST and LD according to IEC 61131-3. The maximum number of input-output data is 128 kB.

PC Worx Express supports the following: features for diagnostics of control points or individual programming stages, for example, debugging of libraries and structure; making changes to the program while the controller is running; comfortable control of the device by the PROFIBUS, INTERBUS and also Modbus-TCP integrated configuration and addressing; quick commissioning and programming using an extensive collection of standard libraries.

Config+ is software for configuring the INTERBUS network. It performs the following functions:

- data reading and comparison of actual and project topologies;
- assigning addresses automatically or by Drag & Drop method;
- parameterization of several leading connection blocks and modules in one project;
- assigning IP addresses using the BootP server;
- parameterization of devices regardless of the manufacturer by means of the FDT/DTM concept;
- configuring the settings of multiple devices using the MDC wizard;
- monitoring of wiring control;
- diagnostics of INTERBUS networks by graphical indication of errors in the network topology, text messages with tips for troubleshooting errors, displaying device states online, providing statistical data on the quality of the transmission, storing comments on error messages;

Config+ software has an integrated Diag+ diagnostic tool for troubleshooting.

Diag+ is diagnostic software for PROFINET and INTERBUS, which reports both network errors and the current status of control devices. It includes the following diagnostic functions:

- start and stop of data transfer via INTERBUS;
- acknowledgment of INTERBUS error messages;
- bypassing, enabling and disabling of INTERBUS modules;
- displaying error messages with tips for their elimination and detailed information about the type of device and its status;
- output of color characters for indication of errors and device states;
- quality control of fiber optic lines transmission for preventive diagnostics;
- comparison and processing of time-varying LWL diagnostic data packets;
- creation of reception protocols as a PDF file;
- connection to other software tools, for example, to visualization tools;
- display of saved messages from the messages archive of the control device;
- overview of the topology of Ethernet/PROFINET devices in the form of two-dimensional graphics;
- displaying message about the availability of Ethernet/PROFINET devices;
- management of individual user rights for various users.

Diag+ NetScan is software for cyclic diagnostics of INTERBUS networks. It provides synchronous monitoring of INTERBUS networks with several connection modules/control devices. For example, continuous monitoring of transmission quality in all optical channels of the system can be carried out. It is possible to control low-level buses connected via systems interface devices. Moreover, a 32-bit application with the programming interface as an ActiveX control element can be included.

Main functions of the software:

execution of basic commands (start/stop/...);

input of data on the bus structure;

recognition/presentation of alarm conditions (text messages from the database);

storage of diagnostic information in flash memory or memory for controller parameters;

diagnostics of fiber-optic channels (transmission quality control);

cyclical reading of diagnostic data from all controllers or INTERBUS controllers of the same network (the number of control groups is unlimited);

mapping of the network structure, where all the controllers (INTERBUS control devices) of one system are visually displayed as a tree structure (detailed diagnostic information is triggered by the button click);

simultaneous monitoring of up to 10 controllers/INTERBUS control devices.

Visu+ is visualization software which provides expanded monitoring by displaying diagrams or setting values for alarms. It also allows registering and logging all process parameters and exchanging data with databases or enterprise resource planning systems.

Visualization elements can be selected using Drag & Drop function. Data can be viewed using a web client, such as a mobile phone. The software can work both on the ARM platform and with multiprocessor server systems.

WebVisit is a software tool for visualization of devices and processes in the network. It helps to create graphical interfaces without programming, which allows for visual control of works at the factory and onsite. The free version of WebVisit 6 Express allows organizing visualization on the basis of web technologies. Moreover, up to 10 visualization pages can be created and up to 60 process variables can be used.

The biggest advantage of WebVisit is that it is a graphical editor itself (Java or HTML programming is not required). WebVisit visualization pages can be displayed in any standard browser.

AX OPC SERVER is communication interface for SCADA-system with OPC support and control system based on PC Worx. OPC (OLE for Process Control) is a family of software technologies that provide a single interface for managing automation objects and technological processes. The AX OPC server provides all OPC DA clients with variables of controllers, programmed with PC Worx. The interface provides the ability to access local and global data. Variable names are automatically synchronized with the current project.

Main functions:

- support of standard OPC functions, as well as all additional interfaces (according to OPC specification DA 1.0a and DA 2.04/2.05);

- simultaneous support of several control devices;

- built-in client for OPC testing and diagnostics.

The following types of controllers are supported: ILC 1xx, AXC 1xxx, ILC 3xx, AXC 3xxx, RFC 4xx, PC Worx RT BASIC/SRT.

The following standard data types are supported: BOOL, BYTE, INT, DINT, UINT, SINT, USINT, UDINT, WORD, DWORD, REAL, LREAL, STRING, TIME (according to IEC 61709).

The following user data types are supported: ARRAY OF, STRUCTS, ARRAY OF STRUCT, STRUCT in STRUCT (according to IEC 61131-3).

2.2. Hardware requirements for the AUTOMATIONWORX Software Suite environment

The AUTOMATIONWORX Software Suite environment requires the following hardware:

- RAM – at least 2 GB;

- processor – at least Pentium 4/Celeron 1.6 GHz;

- interfaces – COM-port, Ethernet (PCI for the INTERBUS-Master main board in a personal computer);

- monitor resolution – XGA (1024 x 768).

Also, operating system – MS Windows 7 and higher – is required.

The following functions are supported:

- languages – German, English, French, Italian, Spanish, Chinese;

- browsers – Internet Explorer 8 and higher, Opera.

2.3. Installing the AUTOMATIONWORX Software Suite environment

The AUTOMATIONWORX Software Suite environment can be installed from installation discs or from Phoenix Contact official website – www.phoenixcontact.com. In the latter case, the sequence of actions is as follows.

1. In the "Products" section, select the "Software" section (Fig. 2.1).

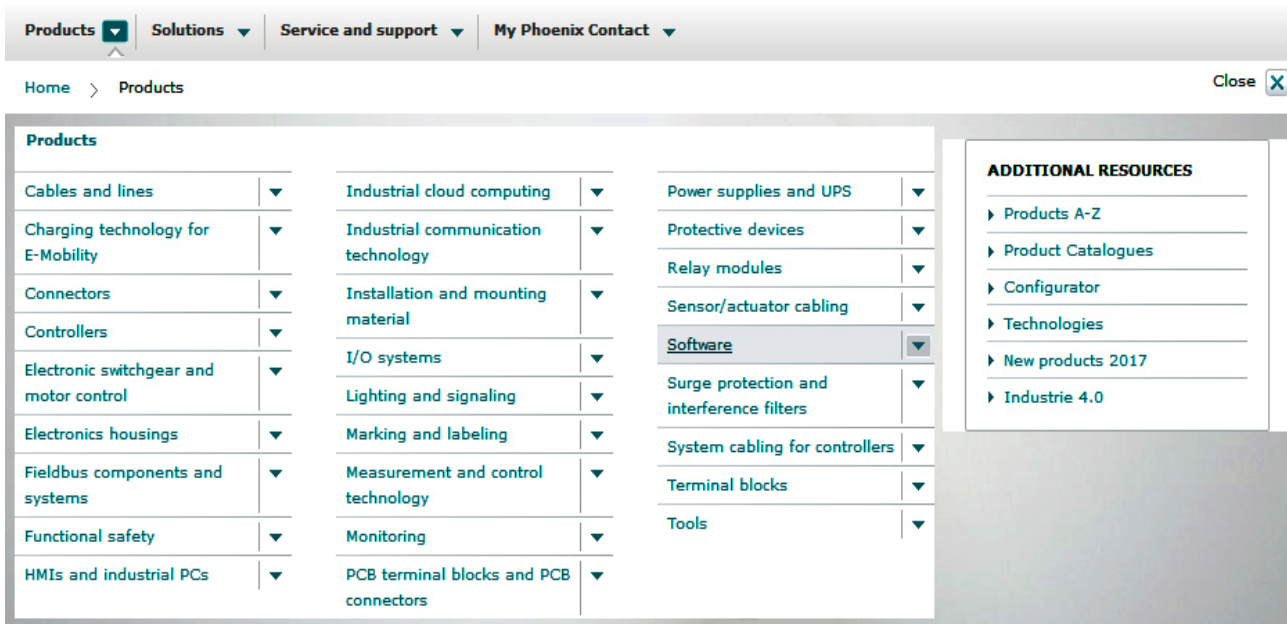


Fig. 2.1. Phoenix Contact component page

2. In the "Category" section, select "Programming" (Fig. 2.2).



Fig. 2.2. Category selection

3. In the new window, expand the item "Product list Programming" (Fig. 2.3).

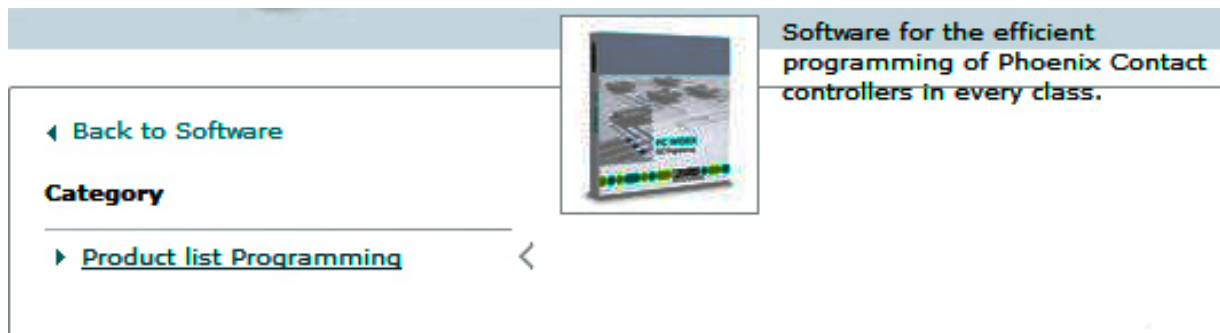


Fig. 2.3. Software selection

4. In the "Downloads" section, check the items and additional components necessary for downloading the AUTOMATIONWORX Software Suite environment (Fig. 2.4).

Demo Software

	Description	Language	Revision
<input checked="" type="checkbox"/>	[zip, 67 MB] Demo Software Post installation for AX SW Suite 1.82 to support firmware 4.4x of class 1x1 modular small-scale controllers. AX_SW_Suite_1.82_AddOn_V1.zip	International	1.82 V1
<input checked="" type="checkbox"/>	[zip,] Demo Software AUTOMATIONWORX Software Suite 1.82 demo programming, configuration, parameterization, diagnostics. The Automation Software Suite is a comprehensive collection of optimally coordinated software tools for the Automation Worx automation system consisting of PC Worx; PC Worx EXPRESS; DIAG+; DIAG+ NetScan; CONFIG+; WEBVISIT; AX OPC SERVER. Visu+ download under article 2988544. AX_SW_Suite_2015_182.zip	International	1.82

Fig. 2.4. Window for selecting the required software version

5. Confirm the terms of the license agreement by clicking on the "Download" button and wait for the download to finish (Fig. 2.5).

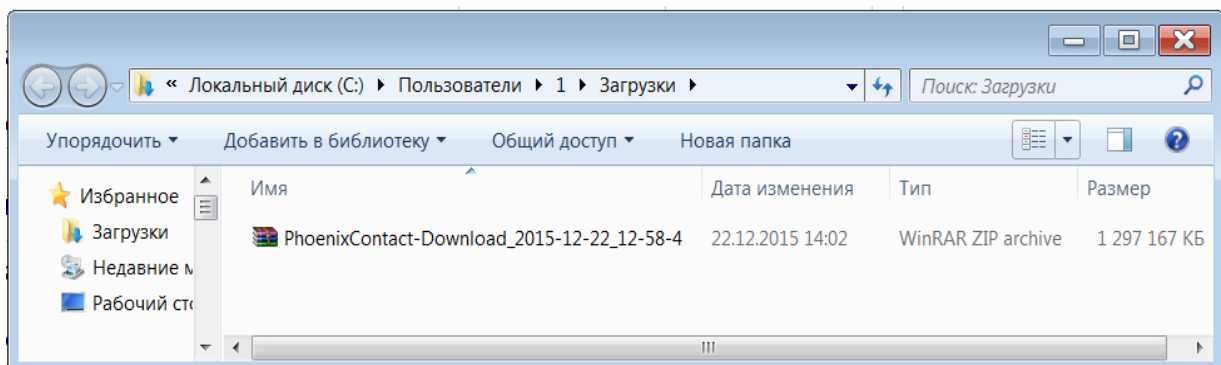


Fig. 2.5. Location of the downloaded installation file

The downloaded archive shown in Fig. 2.5 can be unzipped using WinZip or WinRAR. Let us consider the option of unzipping the archive using WinRAR (Fig. 2.6).

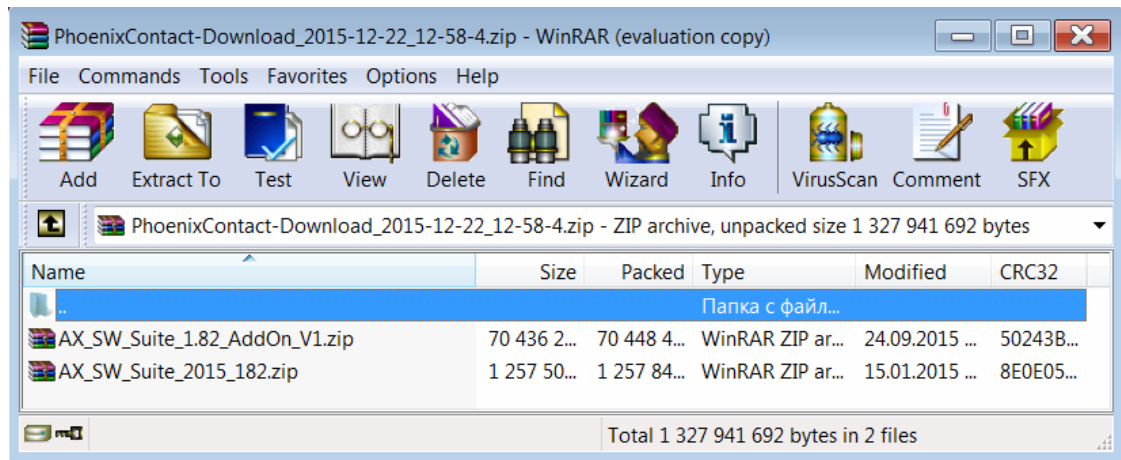


Fig. 2.6. WinRAR window

As it's possible to see from the figure, the archive contains AUTOMATIONWORX Software Suite version 1.82 and its additional components. In order to start the installation, unpack them to a separate folder. After the program is extracted, Setup.exe is launched.

In the first window the installation wizard will prompt you to select the language (Fig. 2.7).

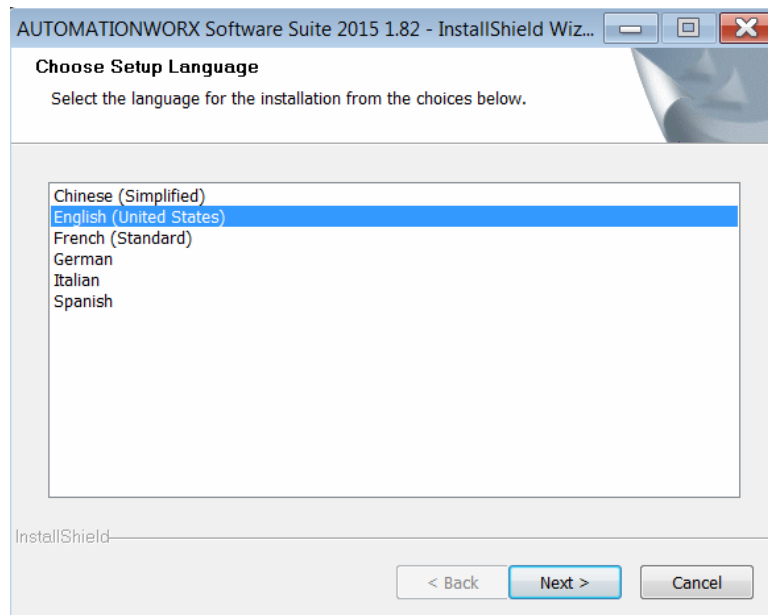


Fig. 2.7. Language selection

During the next step of installation the wizard checks the availability of WinPcap on the computer and suggests its installation in case of absence (Fig. 2.8 and 2.9).

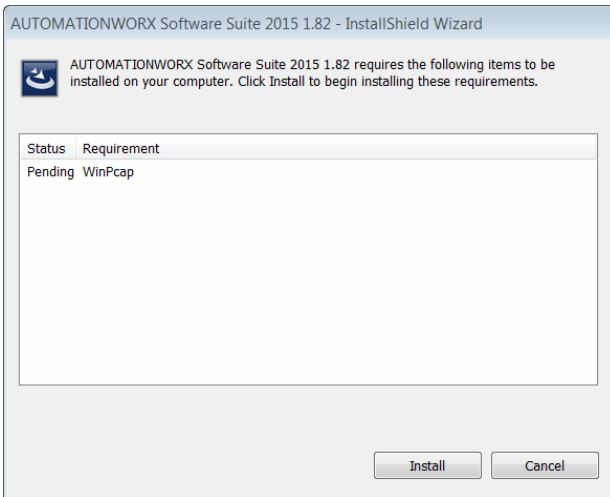


Fig. 2.8. WinPcap selection window and installation



Fig. 2.9. License agreement

WinPcap supplements the standard functions of the Win32 family operating systems with the ability to receive and transmit data over the network, bypassing the operating system protocol stack and interacting directly with the network adapter of the computer, and also provides high-level API applications with low-level processes management. WinPcap consists of three components: packet capture device driver (pasket.vxd), low-level dynamic library (packet.dll), and high-level static library (libpcap).

During the next step it is recommended to check the automatic startup box for the WinPcap driver at booting (Fig. 2.10).

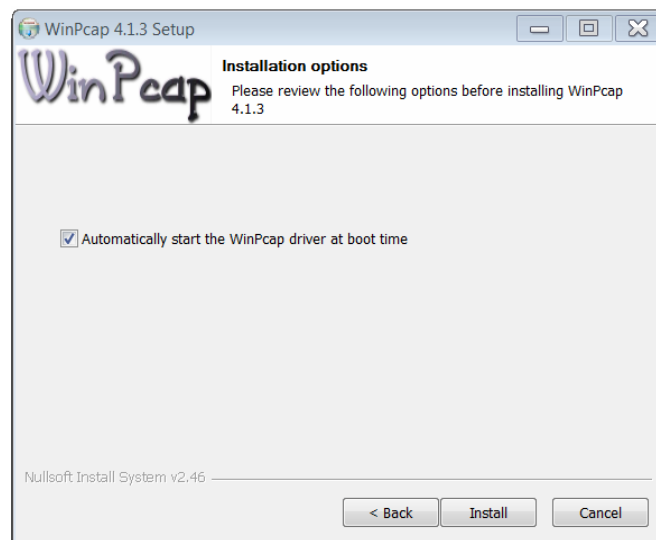


Fig. 2.10. Checking the automatic startup box

After the WinPcap installation is complete, the installation wizard returns to the AUTOMATIONWORX Software Suite installation and prompts you to review and accept the terms of the license agreement (see Fig. 2.11).

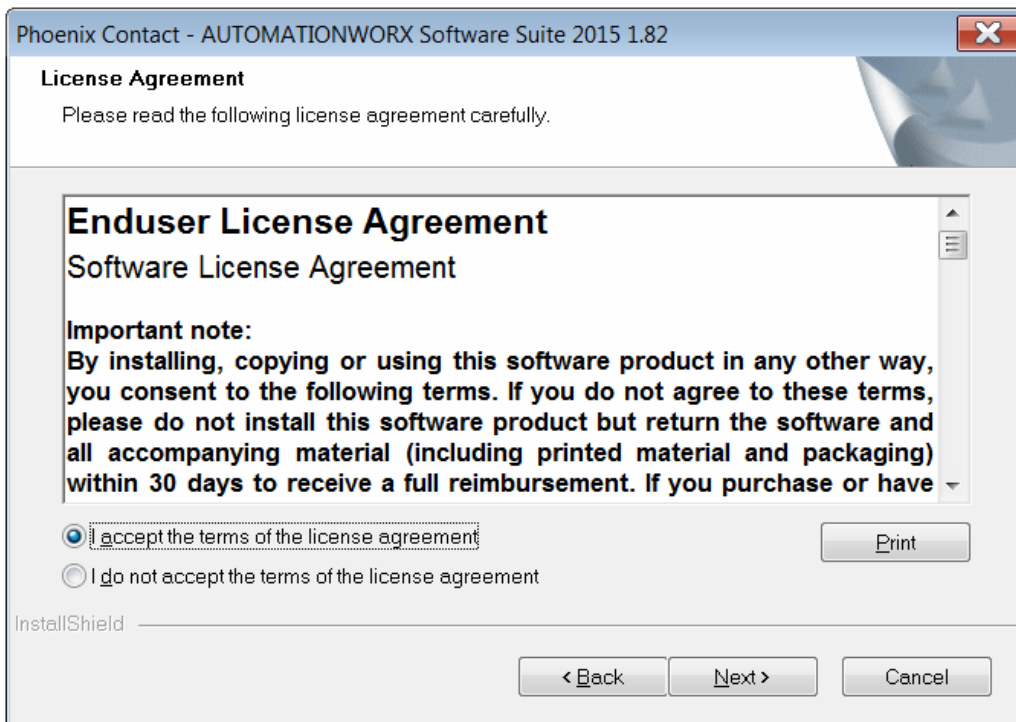


Fig. 2.11. AX SW Suite 1.82 license agreement window

If necessary, the location of this software can be changed by clicking the "Change" button (Fig. 2.12).

During the next step of the software installation the list of installed applications is selected (Fig. 2.13).

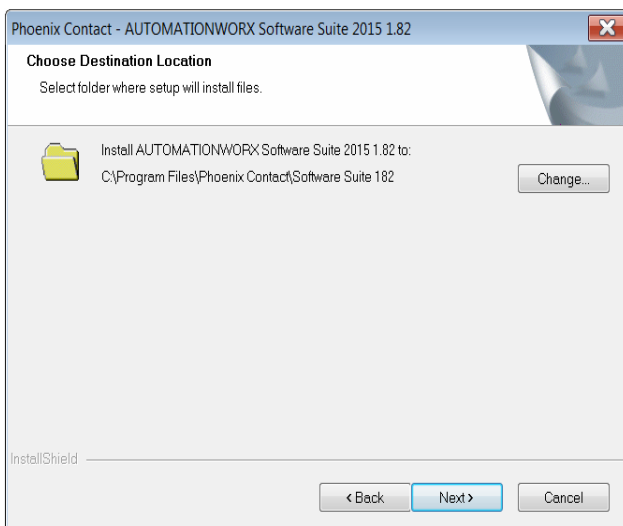


Fig. 2.12. Selecting the location folder for AX SW Suite 1.82

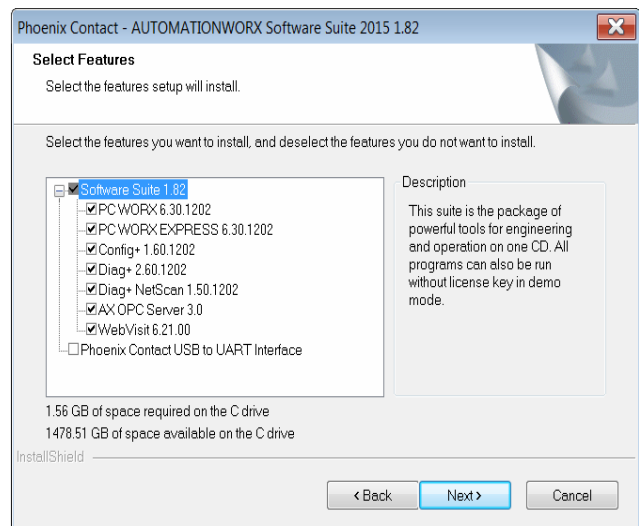


Fig. 2.13. Selecting software components

After that the process of copying and installing files on the computer will begin. After the process is completed, security settings should be configured (Fig. 2.14).

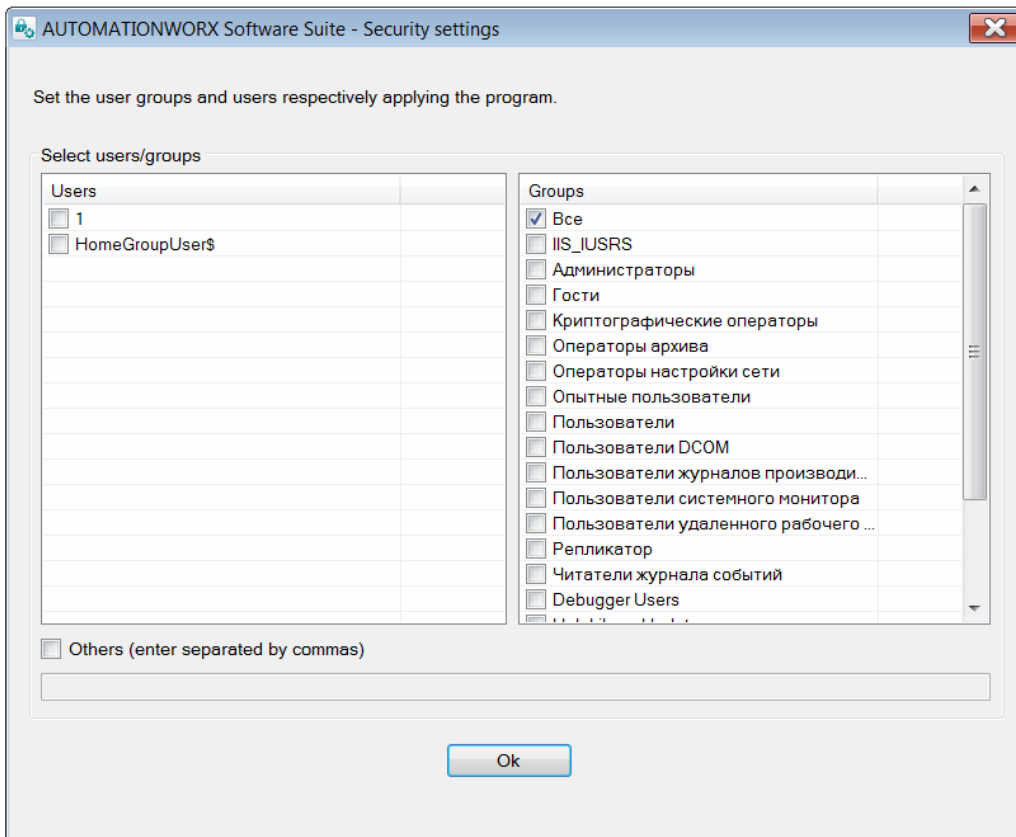


Fig. 2.14. Configuring of security settings

When installation is complete, the final window of the installation wizard appears (Fig. 2.15).

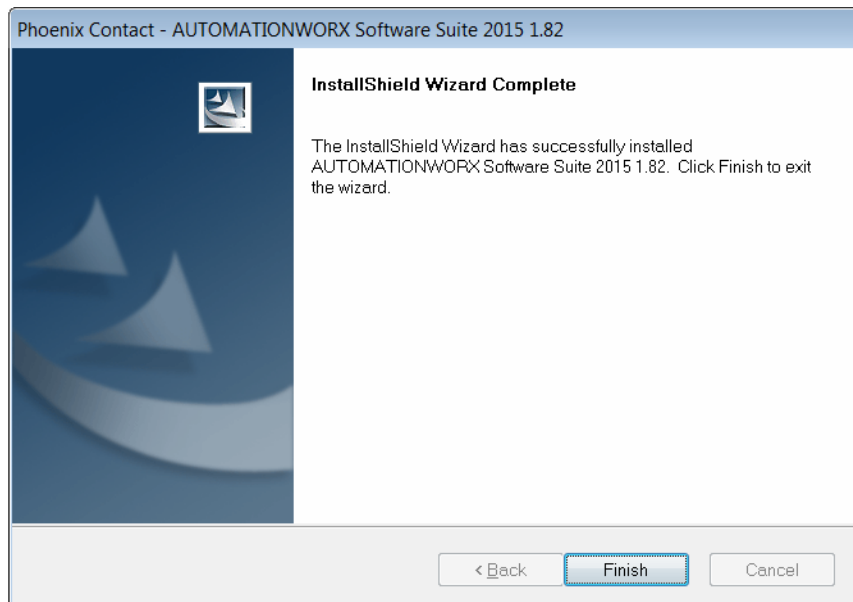


Fig. 2.15. The final window of the installation wizard

The next stage is the installation of additional components. For this purpose run the file AX_Software_Suite_X.XX_AddOn_V1.exe (Fig. 2.16).

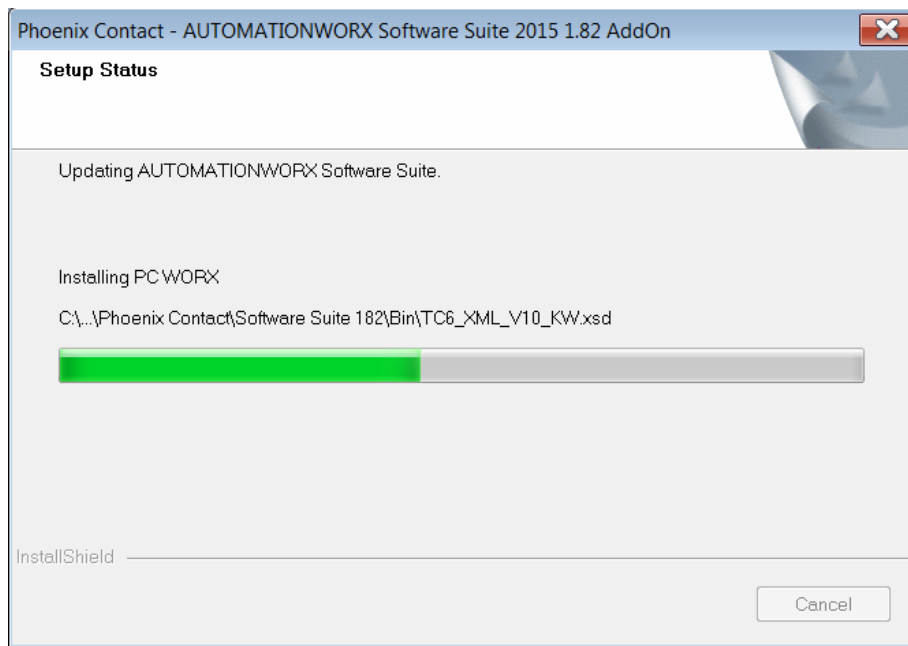


Fig. 2.16. Installing additional components

Before completing the installation of additional components security settings shown in Fig. 2.14 should be configured.

At the end of installation the "Phoenix Contact" folder containing references to the software being a part of AUTOMATIONWORX Software Suite will be created in the Start menu (Fig. 2.17).

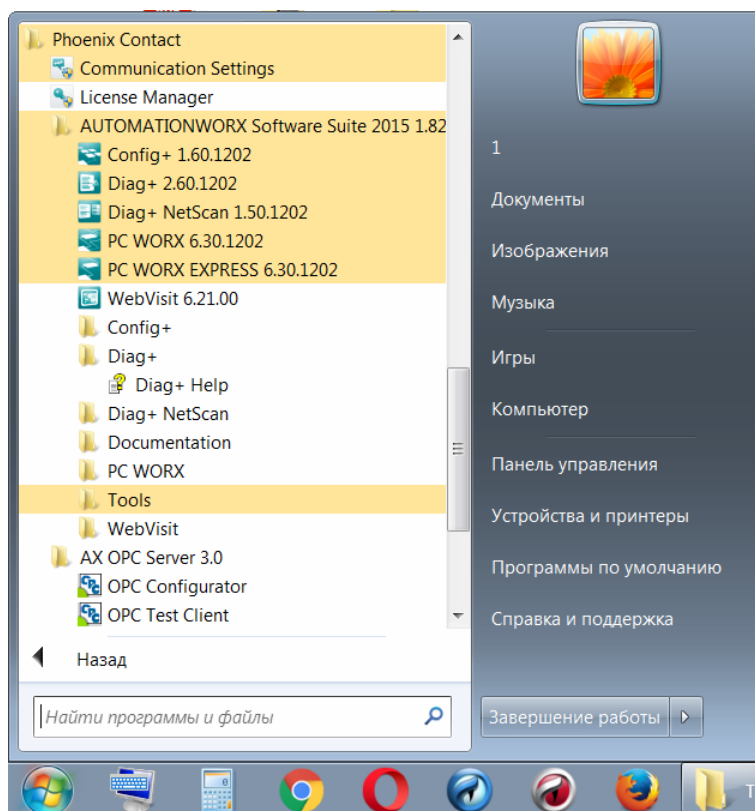


Fig. 2.17. Start menu

Test questions

1. What is the purpose of the PC Worx development environment?
2. What are the hardware requirements for the AUTOMATIONWORX Software Suite environment?
3. What is the purpose of WinPcap?
4. Which folder is selected for the location of the AUTOMATIONWORX Software Suite environment?
5. What folder contains the links to the software being a part of AUTOMATIONWORX Software Suite?

3. PC WORX INTEGRATED DEVELOPMENT ENVIRONMENT

3.1. PC Worx interface and modes of operation

The PC Worx development environment has a standard Windows interface. The title indicates the name of the software and the name of the project. If the project was not saved and its name did not change, the program will have the appearance presented in Fig. 3.1. The color of the title bar indicates whether the program is active or running in the background.

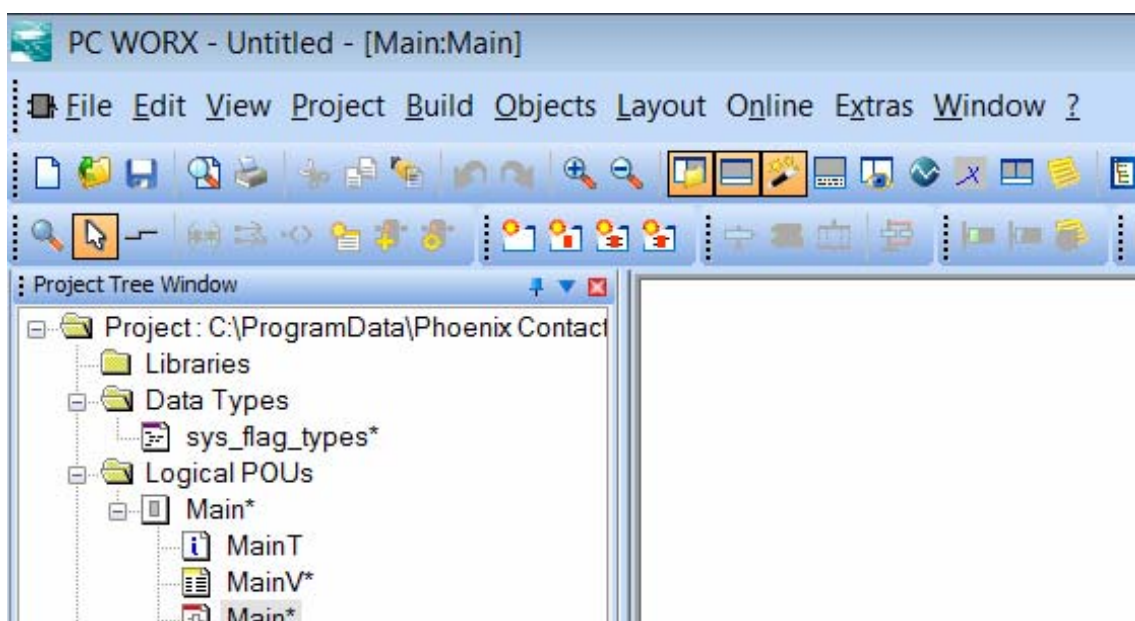


Fig. 3.1. PC Worx interface

Below the title, the top menu is displayed. Depending on the mode of operation, certain menu items may not be available.

Let us consider each menu item in the programming mode.

"File" – allows to create, open, save, close and delete projects. In addition, the menu item contains commands and print settings, as well as print preview.

This menu item contains commands for exporting and importing projects, which can be exported to a CSV file containing comments and descriptions.

In order to increase the security of the project, password protection is provided ("Enter password ...").

"Edit" contains the commands necessary for editing: copying, cutting or pasting, searching and replacing text lines in the text. Depending on the active editor, the respective commands for working with objects are activated.

"View" is used to display or hide various windows and user interface controls (project tree, message window, cross-reference window, etc.), as well as the state and switching between program modes.

"Project" is used for inserting data and declaring POU libraries. It also contains the command for comparing the current project with another project.

"Build" consists of various commands for starting the compilation of the project after editing, displaying errors found during compilation, building cross-references and deleting declared unused local variables.

"Objects" is available in the editor mode. The sub-menu "Variable" can be used for inserting new variables into the list of the current POU. When editing graphic worksheets, menu sub-items are available for inserting and editing graphic objects, such as connectors, jumps, contacts, coils, etc. Depending on the graphic language used, some sub-menus may be inactive.

Depending on the type of worksheet (graphic or text), "Layout" contains elements for scaling and adjusting the sheet size, displaying page boundaries or grid, setting auto-scrolling speed and object size.

"Online" contains commands for debugging the project, managing dialog resource and activating display mode.

In the **"Extras"** menu item it's possible to call the dialog boxes "Shortcut Keys" and "Options" and additional tools, such as the editor "Pagelayout Editor". Dialog Boxes: "Shortcut Keys" allows you to define your own keyboard shortcuts or default key combinations; "Options" provides the ability to customize menus, toolbars, text editors and text colors. Using the menu items "Export Options ..." and "Import Options ..." it's possible to export the current settings to an XML file on the local computer.

"Window" is used for placing windows and symbols on the screen and closing all open windows with one click.

"?" contains commands for using the Help option.

Toolbars are located under the top menu. The number of toolbars can be set by the user. Using the items of these toolbars the user can access the frequently used functions of the program. These functions can also be called up through the menu or through predefined shortcuts.

By default, all toolbars are visible. The user can hide or show the toolbar using the "Options" tab of the dialog box.

When you hover the mouse over the symbol, it's possible to see a brief description of this item displayed as a tooltip.

Each panel can be pulled, docked to other panels in another position and moved.

To open a workspace sheet, double-click the associated sheet icons in the project tree. These sheets can be edited using a graphical or text editor.

By default, the working display area has the value "Workbook Style" – when you open multiple sheets, a shortcut is assigned to each opened sheet. In order to activate a certain sheet, click on the corresponding tab or go to opened sheets by repeatedly pressing <Ctrl> + <TAB>.

In order to maximize the workspace (it is useful when working with a small display), it's possible to hide unused windows by clicking the appropriate button on the toolbar.

The message window "Message Window" displays the detected errors and warnings of the compiler.

One of the main advantages of the "Message Window" window is the ability to directly access a worksheet in which errors were found during compilation.

In order to display the message window, select "Message Window" in the "View" menu item. If the "Message Window" window is closed, it will automatically appear when compilation starts. The user can customize the message window by resizing it and moving it to another location on the screen.

The status bar "Status bar" displays various messages when the user is working with the programming environment.

The left part of the status bar displays messages about the operations that are being performed or system messages. If you place the cursor on top of the symbol icon or menu item (without performing the operation), a short description of the function from the selected character or menu item will be displayed in the status bar.

When working in the graphical editor the field to the right shows the position of the cursor, and when working in the text editor – the current row and column. Under the cursor free space on hard disk is displayed. If there is not enough disk space, this field is colored red.

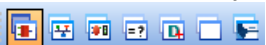
When sending a project to a programmable controller (upload/download), the progress indicator shows the progress of the operation and is displayed in the status bar.

Different modes of the programming environment are indicated by the different colors of the status bar:

gray indicates that the work is done offline, i.e. there is no connection between the programming system and the controller;

green indicates that the connection is established and data can be uploaded/downloaded;

red indicates errors.

During operation, PC Worx allows switching to different modes of operation. Modes are switched by the top menu "View" or using a group of buttons on the toolbar .

The interface of the IEC Programming mode is shown in Fig. 3.2.

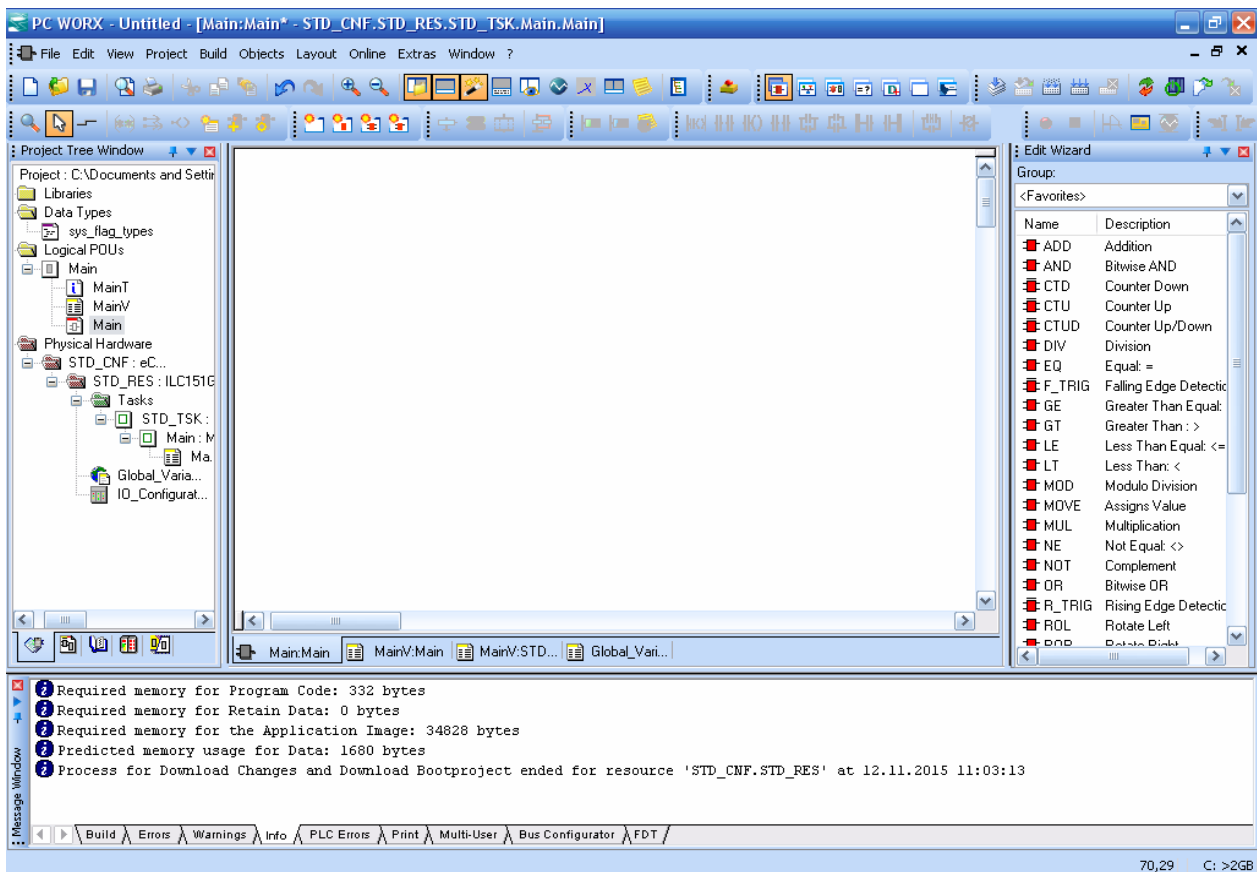


Fig. 3.2. Basic interface of the IEC Programming operating mode

This interface consists of four main windows. The name of each window is indicated at the top or on the left.

"Project Tree Window" has a tree structure and serves to organize the program and project management. It consists of the following branches:

"Libraries" – includes project libraries;


"Data Types" – serves for declaring data types;

"Logical POU's" – stores the main program and programs added by the user;

"Physical Hardware" – contains the configuration of the elements, global and network variables.

The contents of the selected branches "Project Tree Window" is displayed in the central part and is called the "Working display area". The white field in Fig. 3.2 is called the "WorksSheet" of the main program. At the bottom of the working display area it's possible to see the tabs that display the contents of the "Project Tree Window". Switch tabs by clicking the mouse or using the keyboard by repeatedly pressing <Ctrl> + <TAB>.

The tabs at the bottom of the Project Tree Window allow sorting the contents of the window. When you hover the mouse pointer over the tab (without pressing a button), a tooltip appears that shows the name of the displayed branch.

For this and other windows it's possible to use the automatic hiding function , located in the title bar. When activated, the window will be automatically hidden if it is not used.

The "Edit Wizard" window is located on the right side of the interface. It lists the functions used for programming in the FBD language.

Elements are installed in the worksheet from the Edit Wizard by simply dragging them to the workspace.

All elements of the language are grouped together. The group selection is possible in the drop-down menu under the window title. It includes the following sections.

The group "<all FUs and FBs>" contains all functions. The color of the elements is as follows:

red denotes system elements (Firmware);

blue denotes library items (Library);

user elements are displayed in green.

These colors are the default system colors and can be changed by the user with the help of the "Options" dialog box.

In the "Function blocks" group only the functional blocks of the colors described above are highlighted.

The "Functions" group contains only functions.

The "Network Templates" group offers already saved network templates that can be inserted into the current sheet.

The "String FUs" group offers only string functions.

The "Favorites" group contains the blocks added to the "Favorites" by the user.

The "Type conv. FUs" group contains conversion functions, such as the "Bool" type in "Integer" or "Bool" in "Word".

At the bottom "Message Window" described earlier is located.

The next mode of operation is Bus Configuration. Its interface is shown in Fig. 3.3.

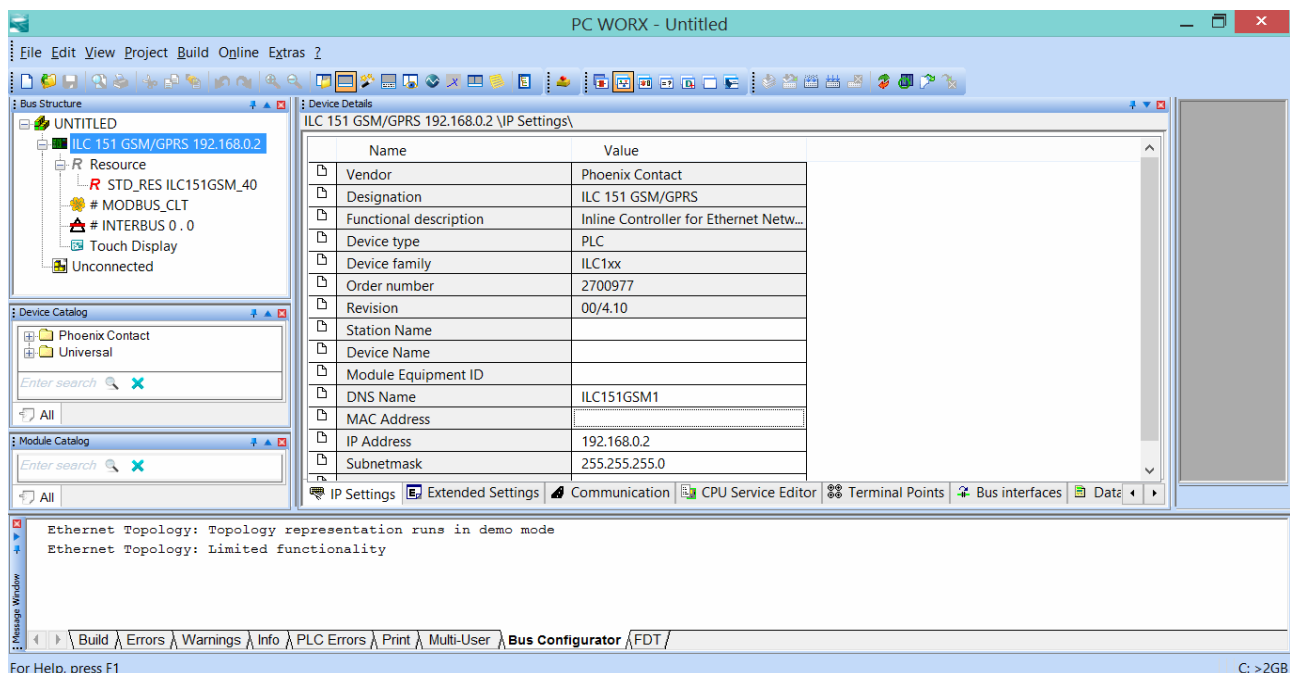


Fig. 3.3. Basic Bus Configuration interface

In the "Bus Structure" window, which has a tree structure, the list of devices used in the project is displayed. The detailed information on each selected device is displayed in the central part of the "Device Detail" window. At the bottom of the "Device Detail" window tabs that display properties by section are located.

The "Device Catalog" window is a constantly updated catalog of devices from both Phoenix Contact and the third-party manufacturers.

The "Module Catalog" window is the catalog of modular devices.

The Process Data interface is shown in Fig. 3.4.

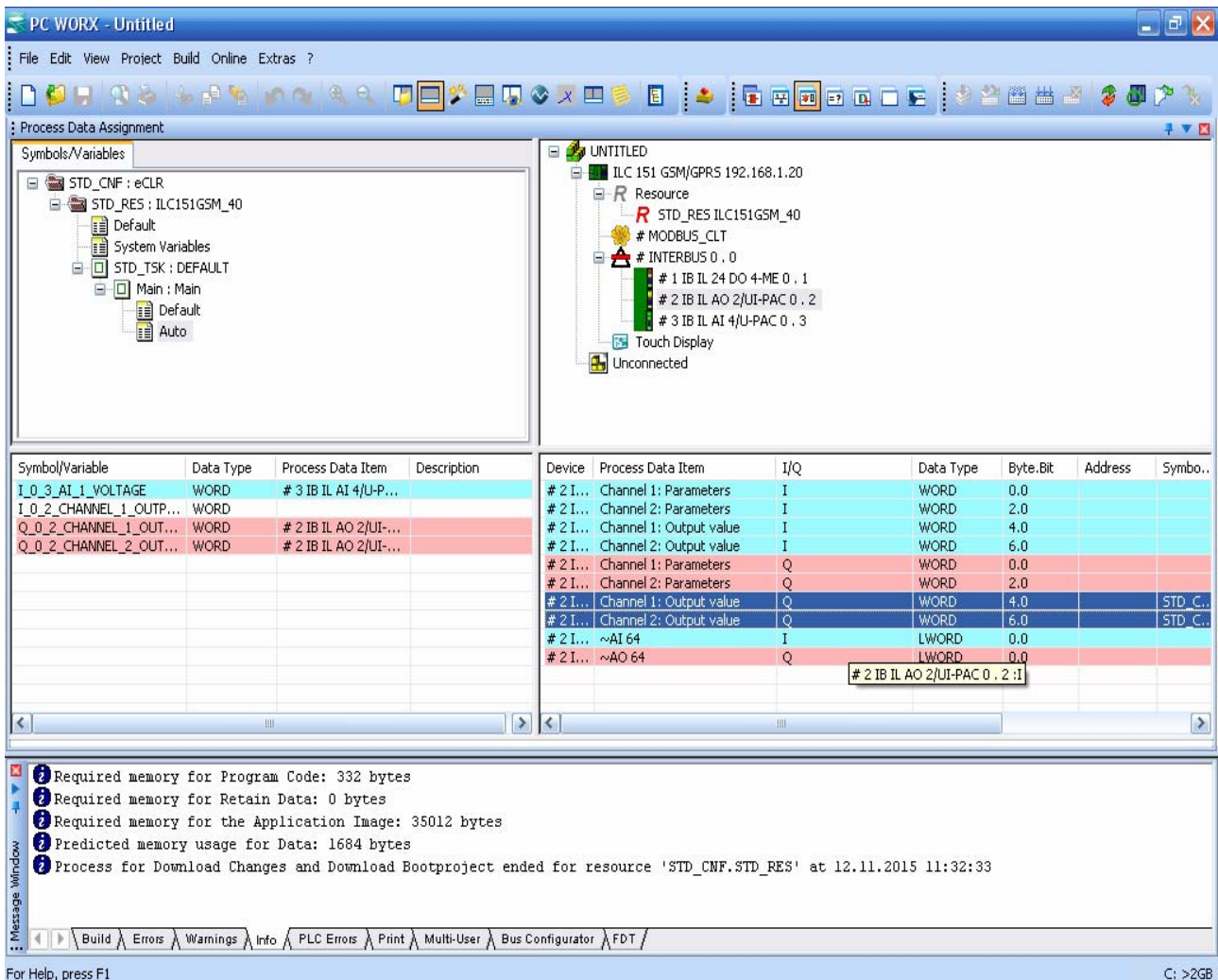


Fig. 3.4. Basic Process Data interface

It is the mode of working with data. All data is displayed in the "Process Data Assignment" window and is divided into four parts. In the upper right-hand quarter the devices connected to the controller using this or that protocol are displayed. The lower part shows the variables that are in the connected blocks. The variables that are to be used in the project are added to the project by dragging them to the left side of the window (it is described in detail in specific examples in the following sections).

Test questions

1. By what color of the status bar are the modes of programming environment distinguished?
2. List the main operating modes of PC Worx.
3. How is the function of displaying/hiding various windows and UI controls implemented?
4. How is project protection with a password implemented?
5. How can help system be called?
6. What information is displayed in the status bar?
7. List the main interface windows when the program is running in Bus Configuration mode.

3.2. Creating a new project

In order to create a new automation system project consistently perform the following steps.

1. Select File – New Project from the menu (Fig. 3.5).

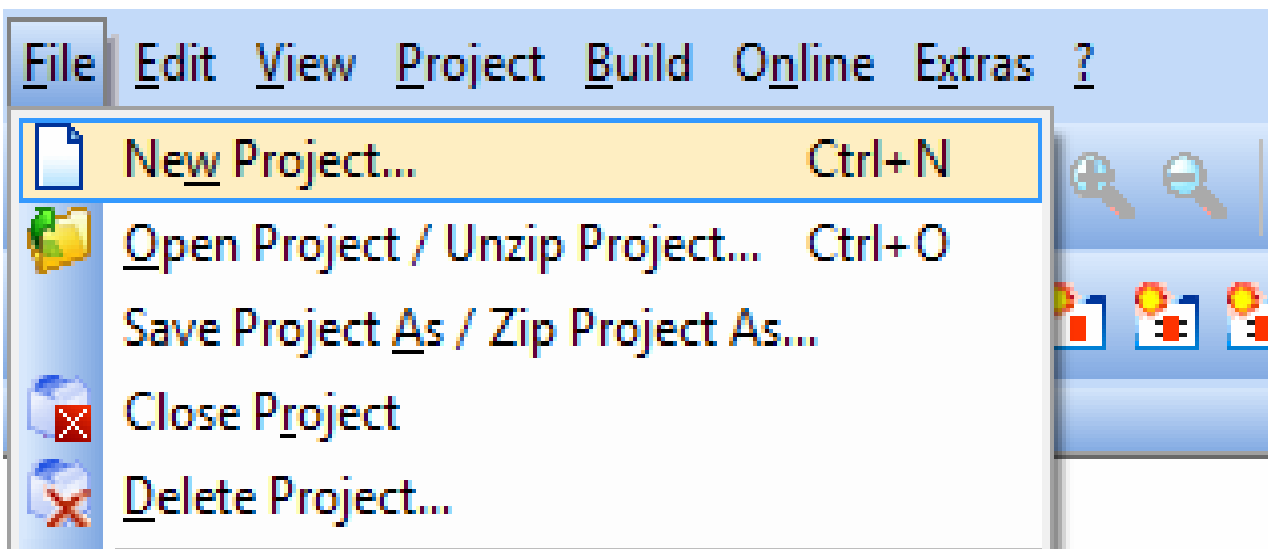


Fig. 3.5. The window for creating a new project

In the appearing window (Fig. 3.6) on the General tab select the Project Wizard item.

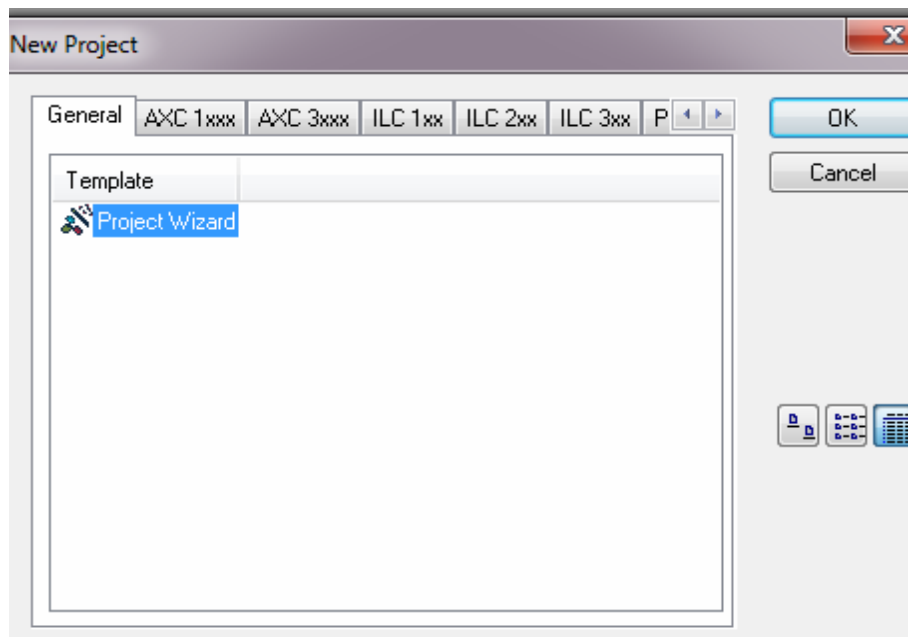


Fig. 3.6. Selection window for the Project wizard

Then consistently follow six stages, described with the help of six separate windows.

In the first stage (Fig. 3.7) specify the project name (in this case, `st_project_1`) and the folder where the project files will be stored.

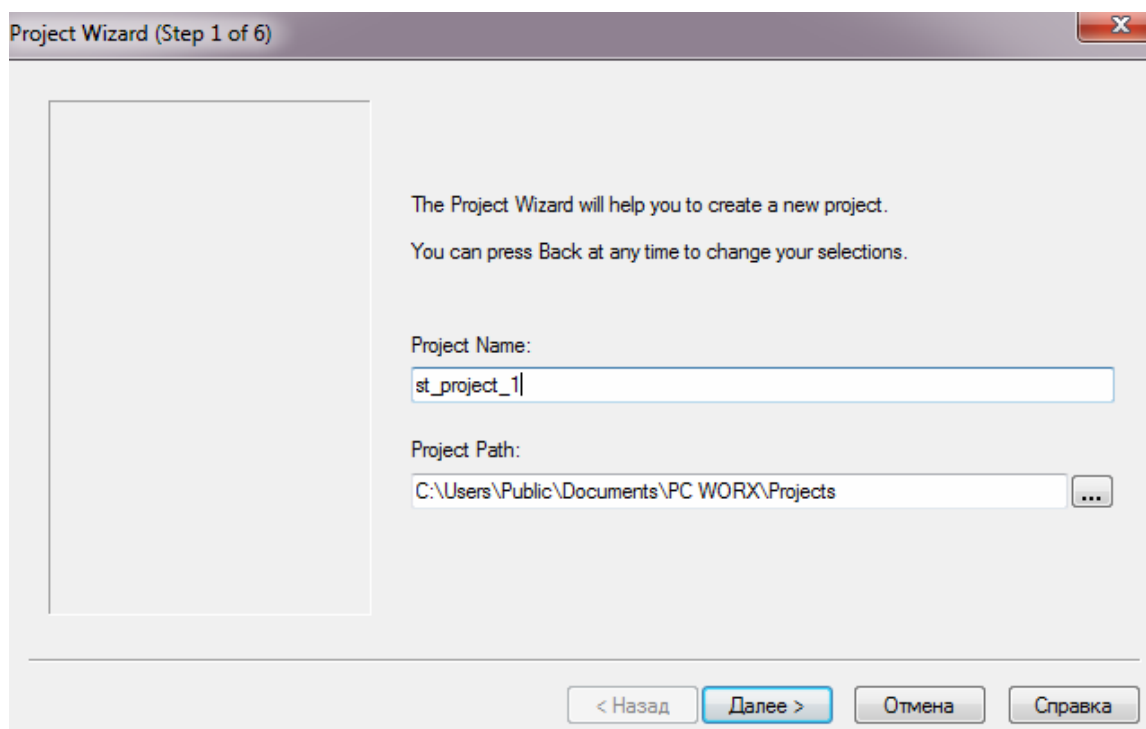


Fig. 3.7. Name and storage folder for the project

The second stage is to select the name of the program to be created and the programming language to be used (Fig. 3.8).

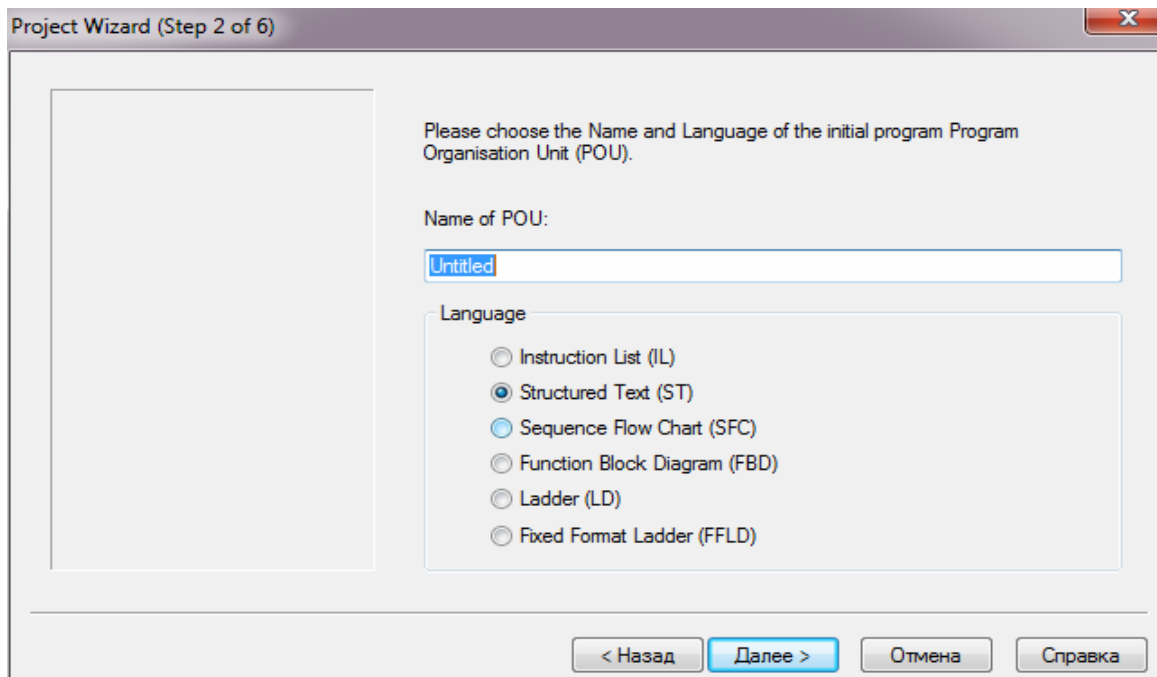


Fig. 3.8. The window for entering the name of the program and the programming language

In the third stage (Fig. 3.9) select the name of the hardware configuration you are creating and the family of processors based of which a particular controller is built.

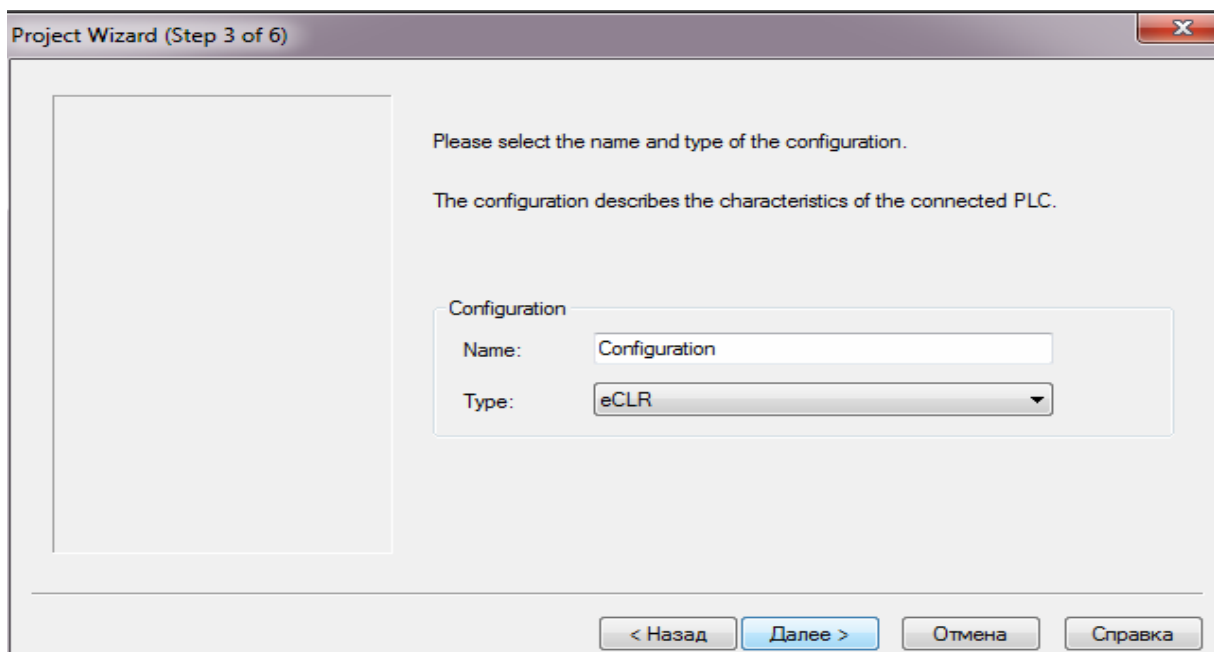


Fig. 3.9. The window for selecting the name of the configuration and the family of processors

In the fourth stage select the model of the controller you are using from the drop-down list (see Fig. 3.10).

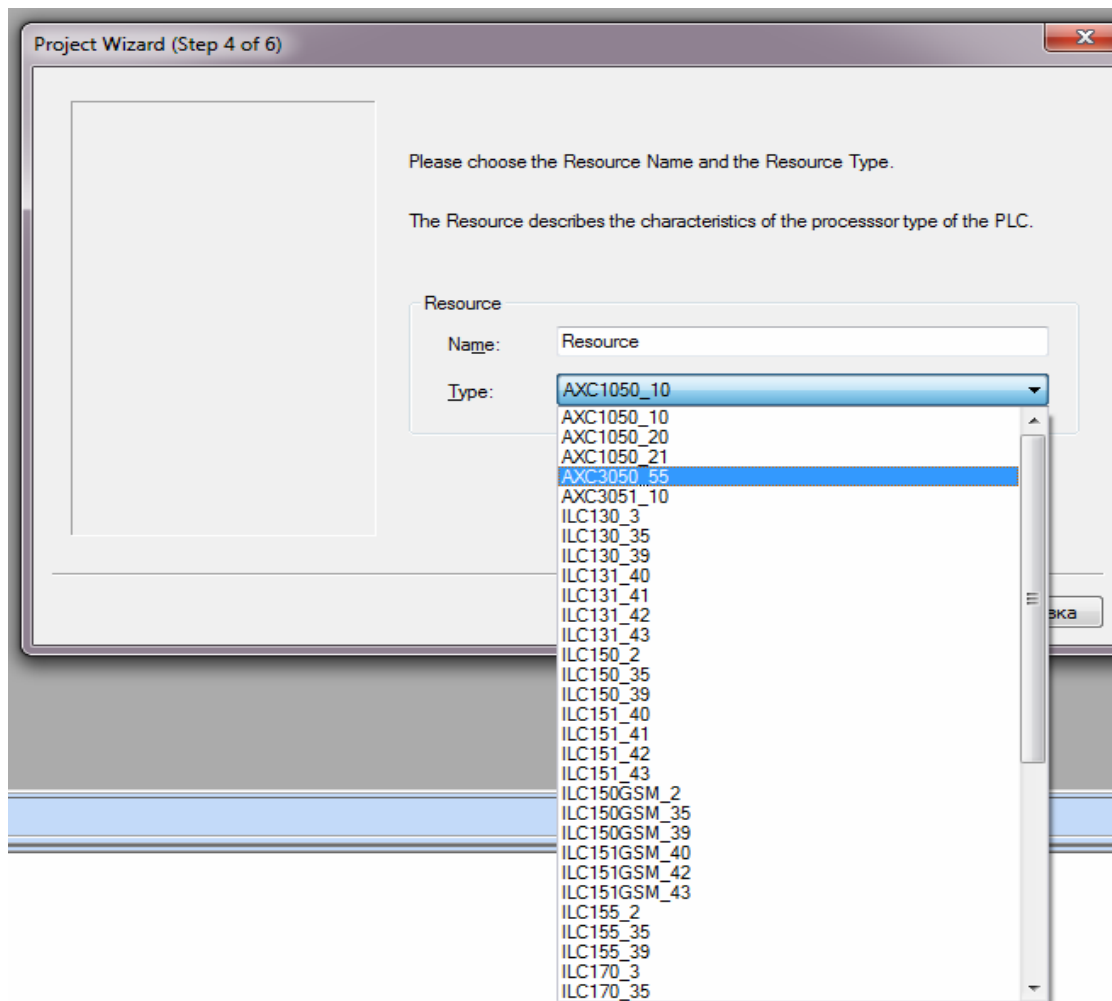


Fig. 3.10. Controller model selection window

In the fifth stage (Fig. 3.11) enter the name of the task and select its type from the drop-down list.

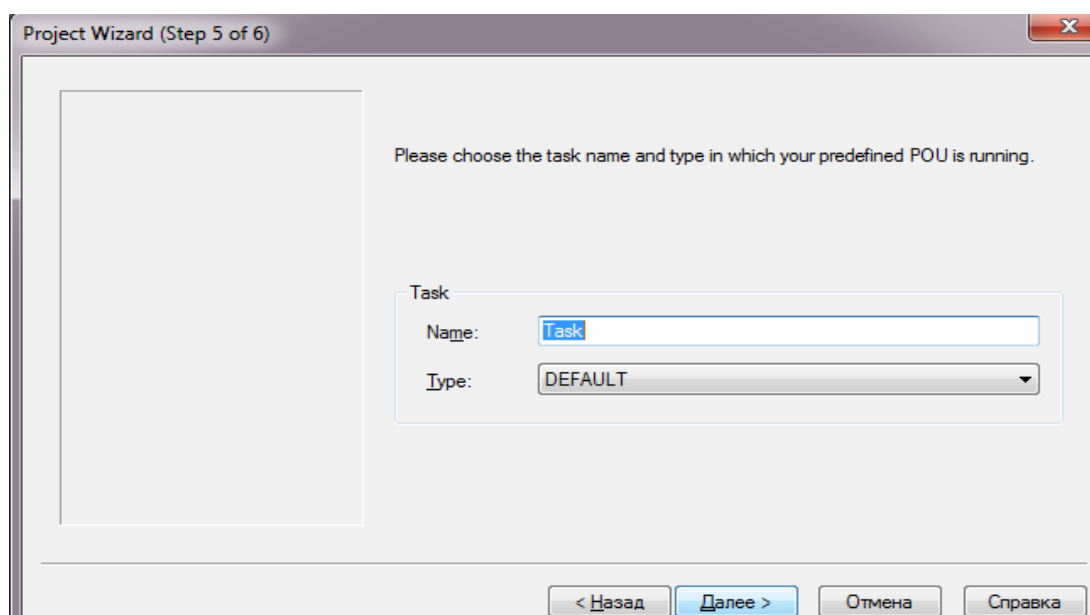


Fig. 3.11. The window for entering the name of the task and selecting its type

In the sixth stage a window appears on the screen (Fig. 3.12) which lists the previously selected properties and characteristics of the project.

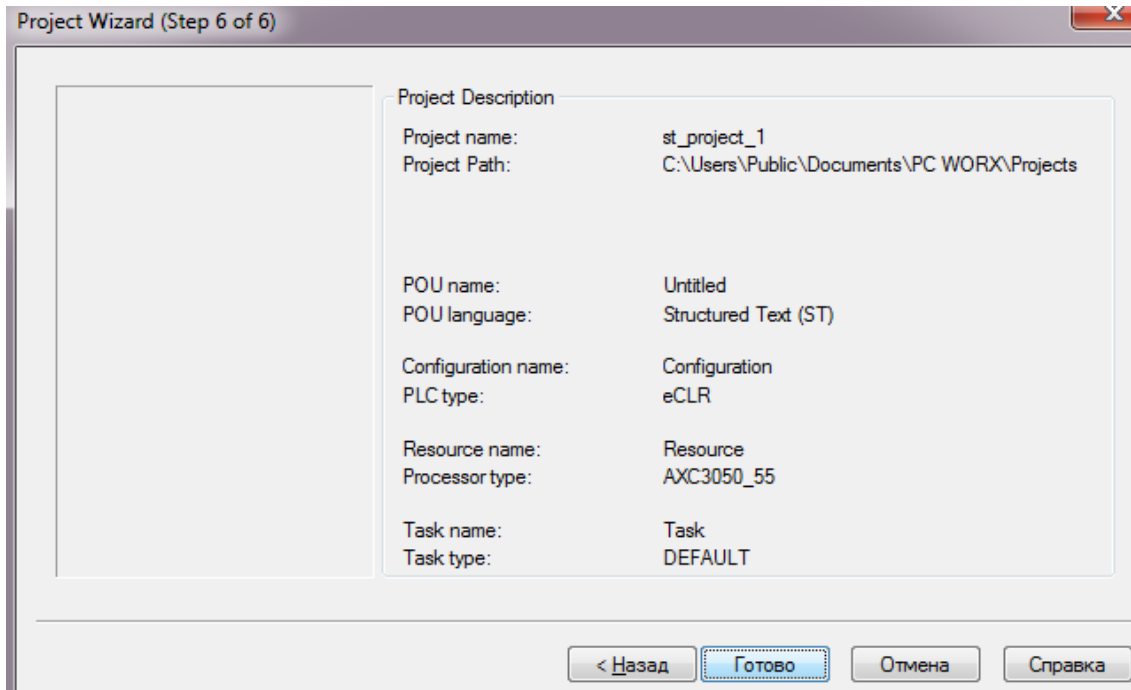


Fig. 3.12. Window containing the properties and characteristics of the project

If necessary, by pressing the "Back" button, it's possible to return to the previous stages and make the necessary changes.

After clicking the "Finish" button and loading PC Worx, the Project Tree window appears (Fig. 3.13), which lists all the elements of the project.

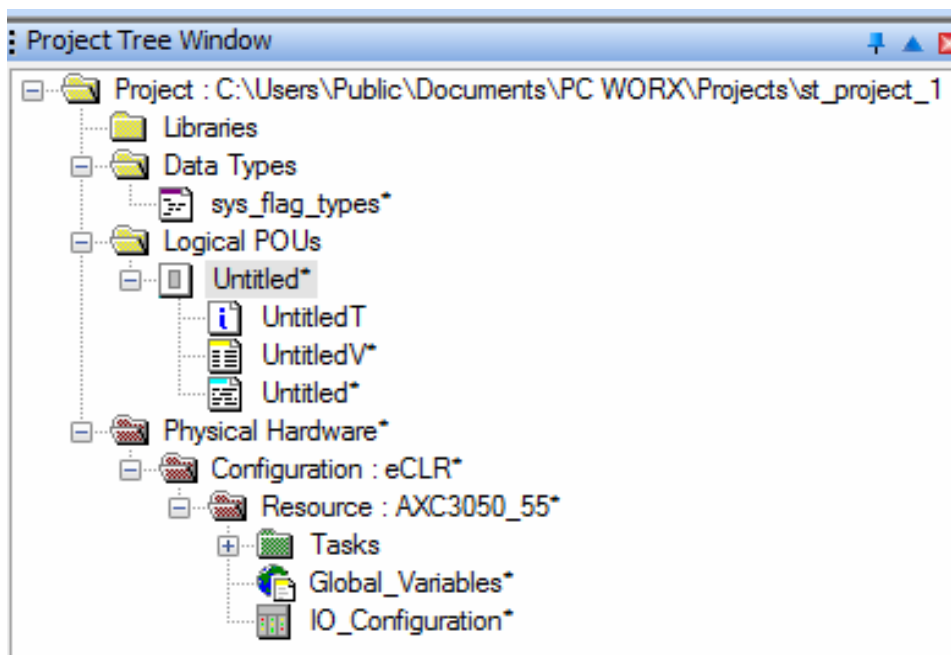


Fig. 3.13. Project Tree window with the listed project elements

The created program contains three parts: Untitled – the program itself in the selected programming language; UntitledV - description of variables; UntitledT – text comments.

PC Worx contains standard functions and function blocks that the automation system developer can use to create his projects. So, the PC Worx version 6.30.1202, released at the end of 2015, contains 247 functions, logically divided into 11 groups, and 113 functional blocks logically divided into 21 groups. In addition to standard functions and function blocks, it's possible to use additional free and paid libraries. The latter allow adding very complex devices to the created projects. For example, the Resy+ library can be used for automation of water, gas and energy facilities, Powerworx library – for automation of networks with low and medium voltage.

Also, any previously created project can be used as a library when creating a new project.

You can add a library to the existing project in two ways.

1. Select the Project menu item, then click Add library, and in the dropped submenu (Fig. 3.14) select either User library or Firmware library.

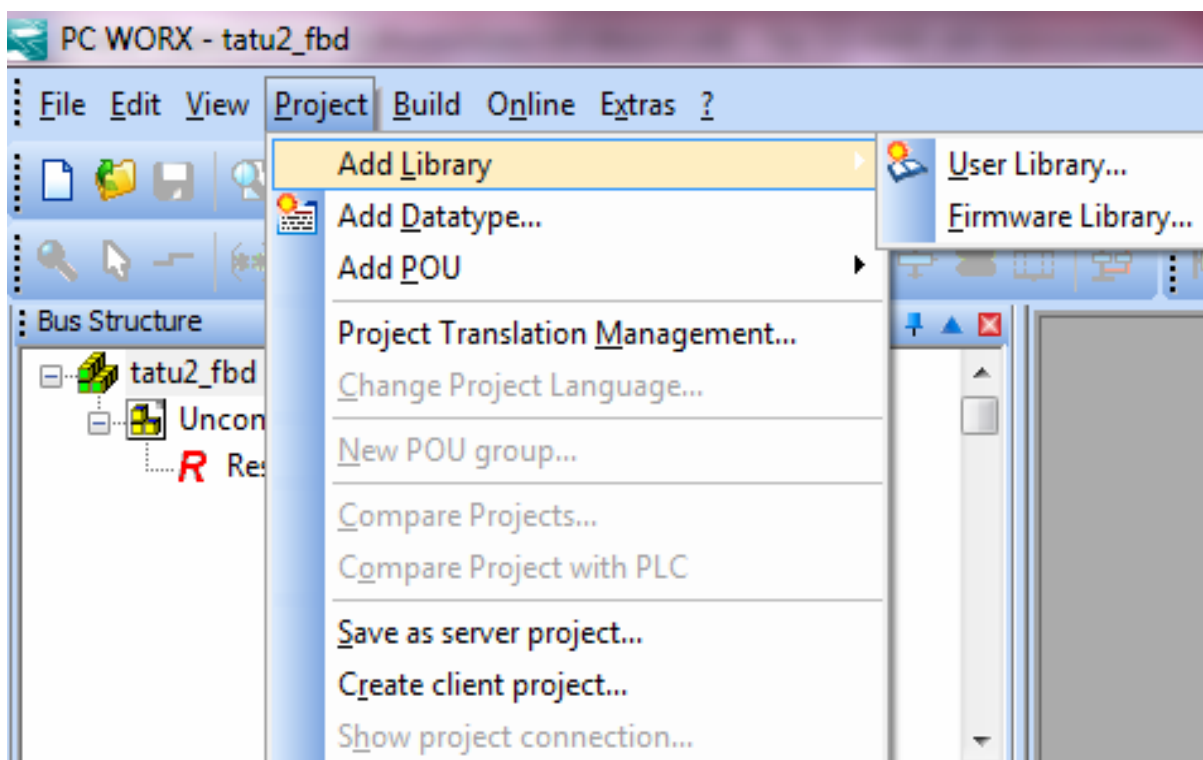


Fig. 3.14. Adding a library using the main menu

2. In the Project Tree Window (Fig. 3.15) place the cursor over the "Libraries" line, right-click and select Insert in the appearing context menu. Then select either the User library or Firmware library.

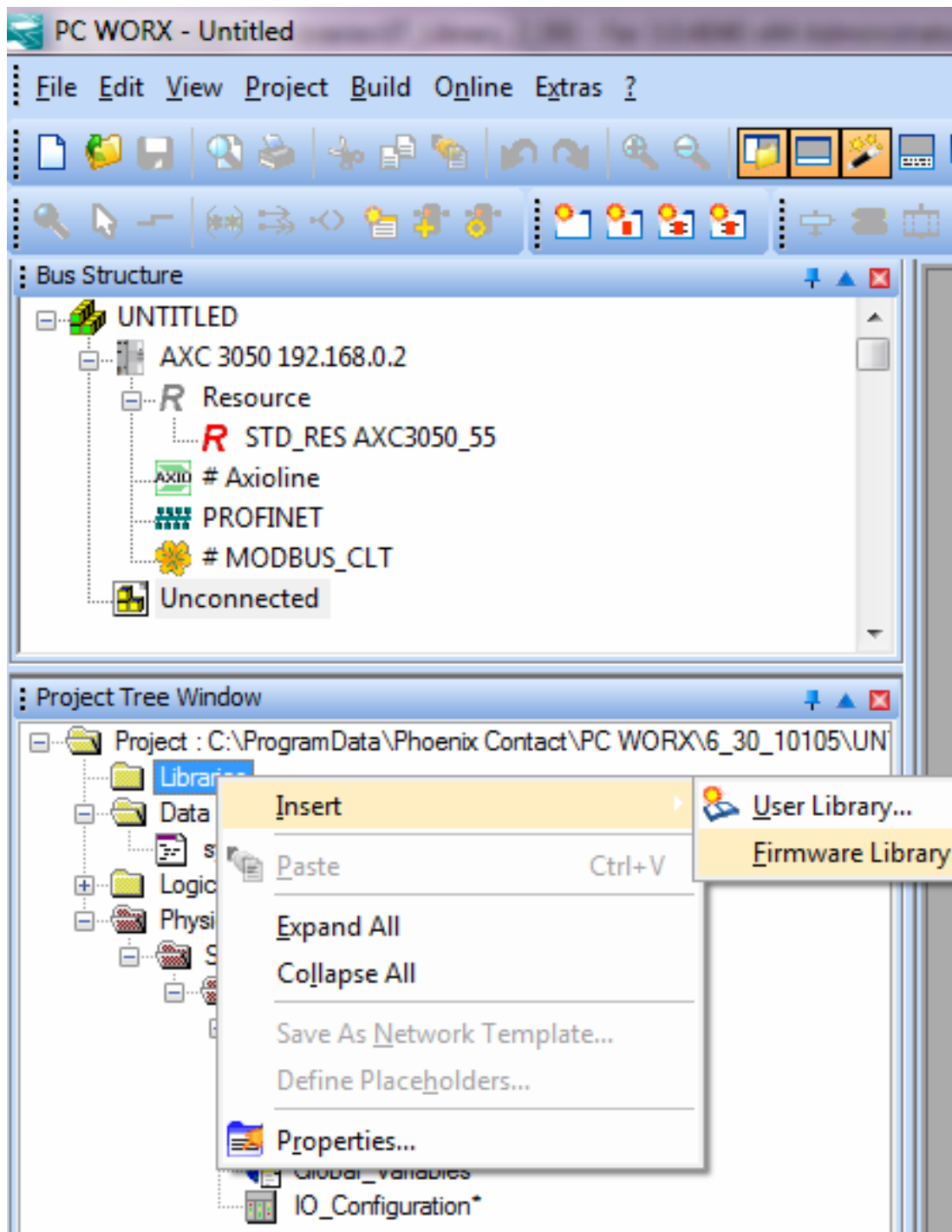


Fig. 3.15. Adding a library using the context menu

If you select Firmware library, specify the path to the file with the extension FWL (FirmWare Library), which contains a description of the specialized original library (Fig. 3.16).

The possible path to the corresponding folder is shown in Fig. 3.17.

Free and paid libraries are stored in files with mwt and mwe extensions. For example, in Fig. 3.18 the file WirelessTechnology_V1_06.mwt contains a free library for expanding the possibilities of working with wireless data transmission technologies.

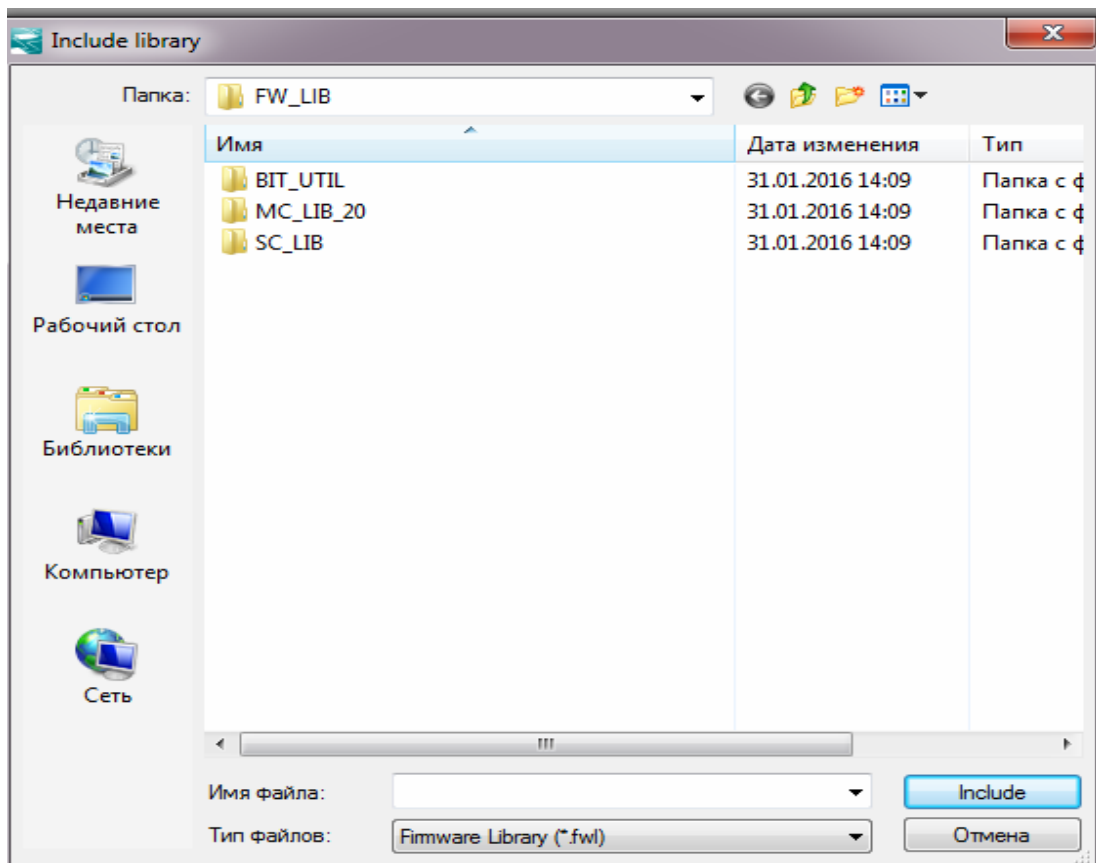


Fig. 3.16. Selection of a specialized original library

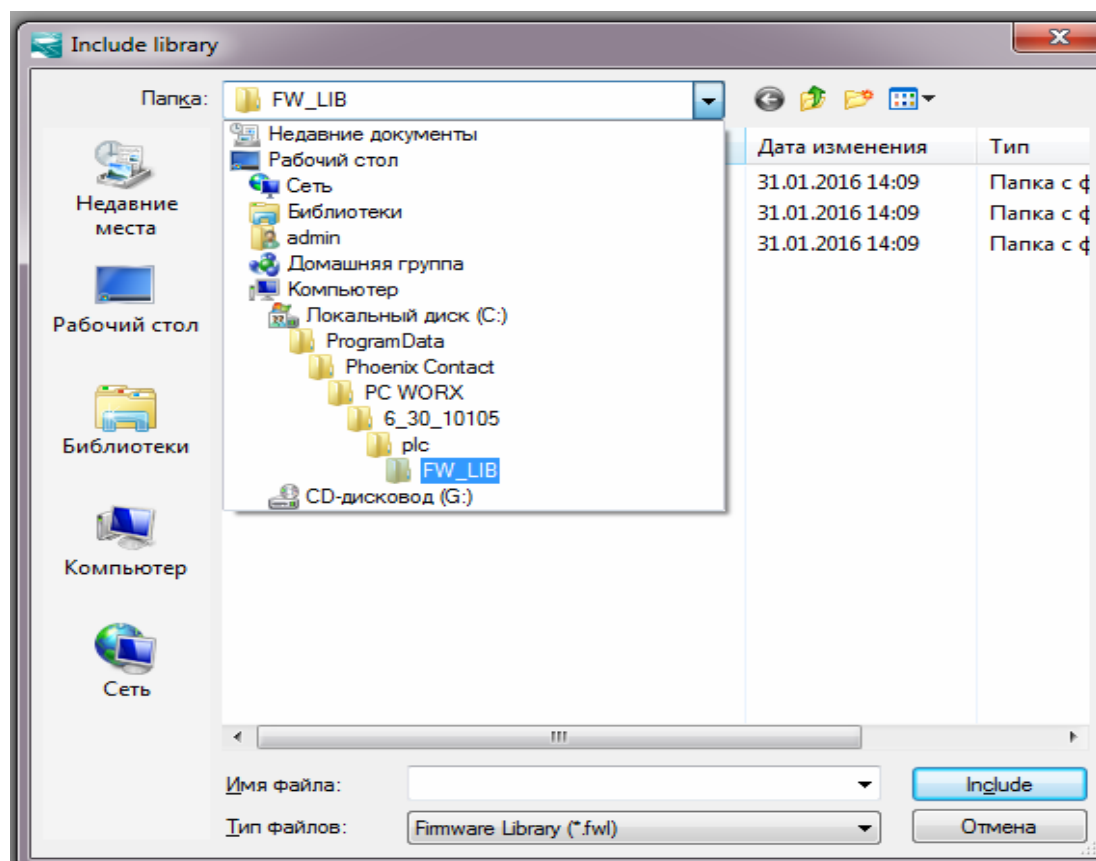


Fig. 3.17. The path to the file containing the specialized original library

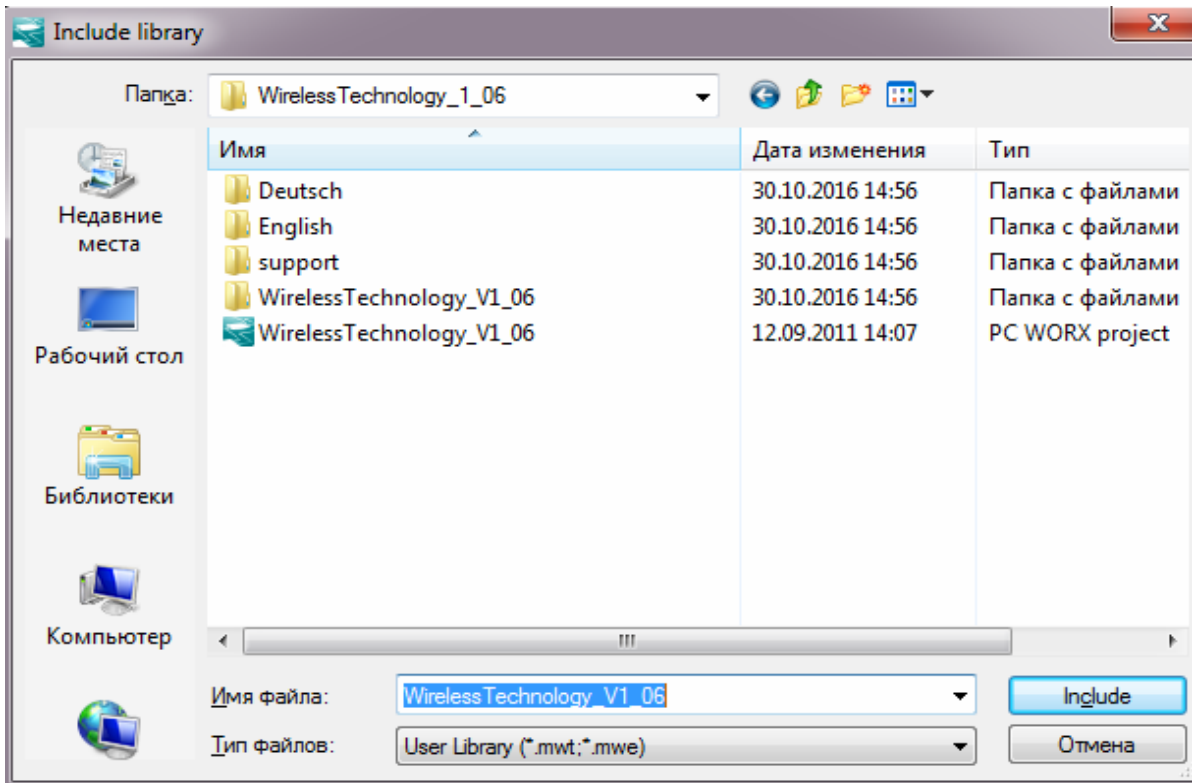


Fig. 3.18. Selecting a library file

If you select the library that is run in the same version of PC Worx as the project itself (or there are slight differences between the versions), after clicking the Include button the project will be completed with the selected library. If the library is executed in an older version of PC Worx (the combination V1_06 in the file name means its version number is 1.06), after clicking the Include button the window shown in Fig. 3.19 will appear. This window allows converting the library to a newer format.

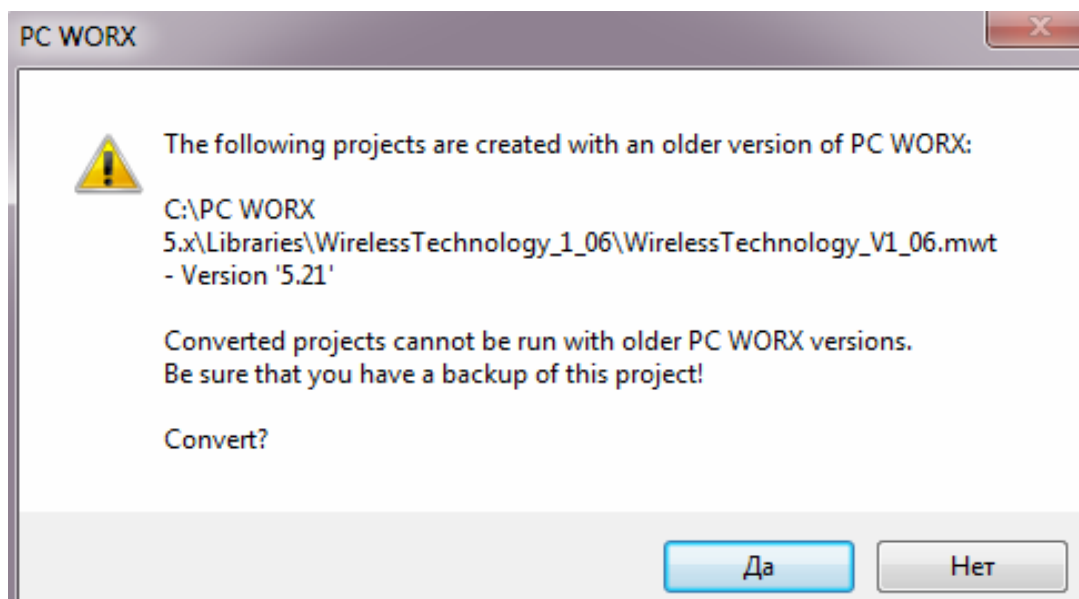


Fig. 3.19. Notification window for converting a library to a newer format

After clicking the "Yes" button a warning window appears indicating that after conversion the library cannot be run with previous versions of PC Worx (Fig. 3.20).

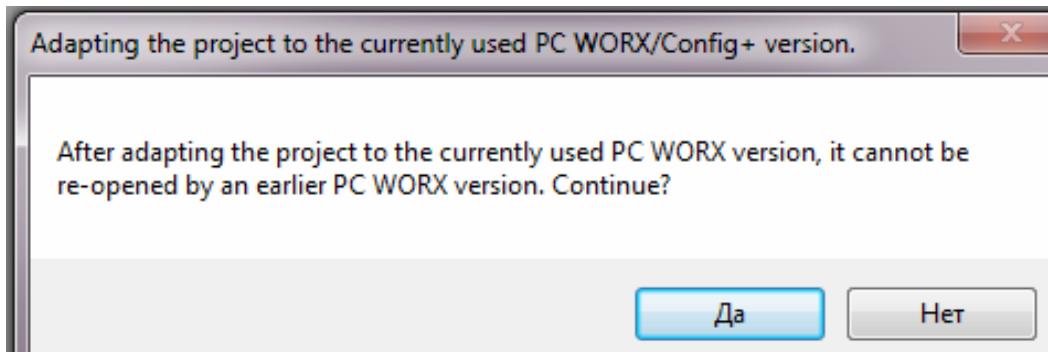


Fig. 3.20. Warning about the impossibility of further use of the library in previous versions of PC Worx

If you confirm the conversion by clicking the "Yes" button, the Libraries tab will be added with the name of the library added to the project (Fig. 3.21).

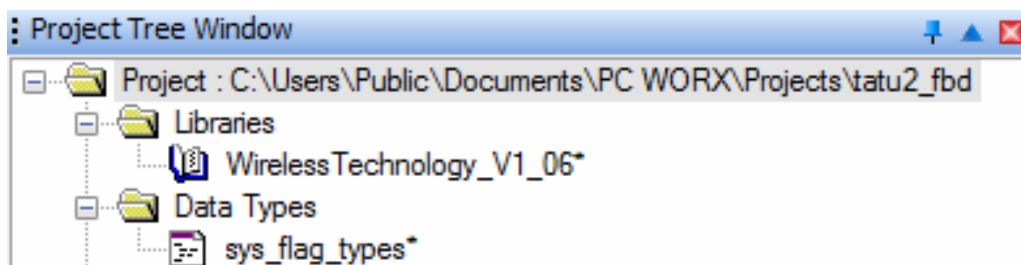


Fig. 3.21. Library added to the automation project

In order to get the list of function blocks and functions available in the added library, select the <WirelessTechnology_V1_06> line in the Edit Wizard window (Fig. 3.22). The specified window is located by default in the upper right part of the screen.

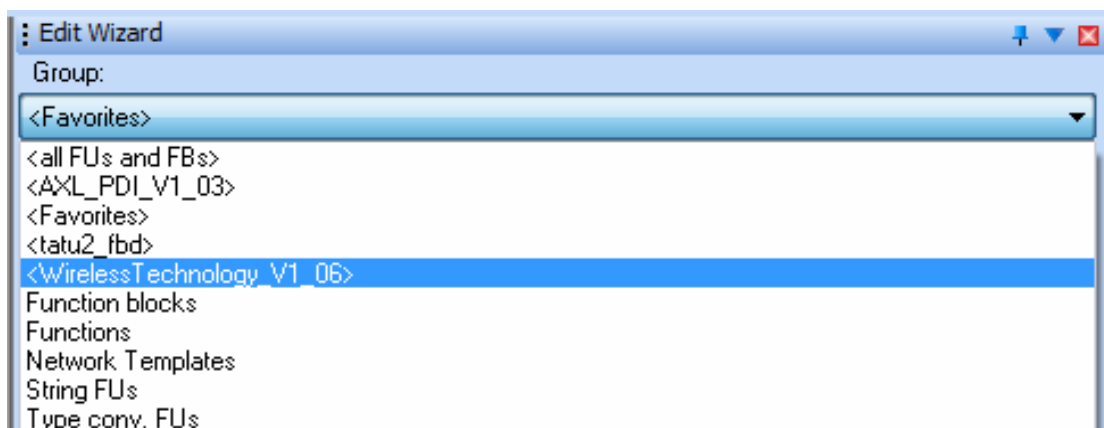


Fig. 3.22. Selecting the connected libraries window

The list of elements (functions and function blocks) available in this library will appear on the screen (Fig. 3.23).

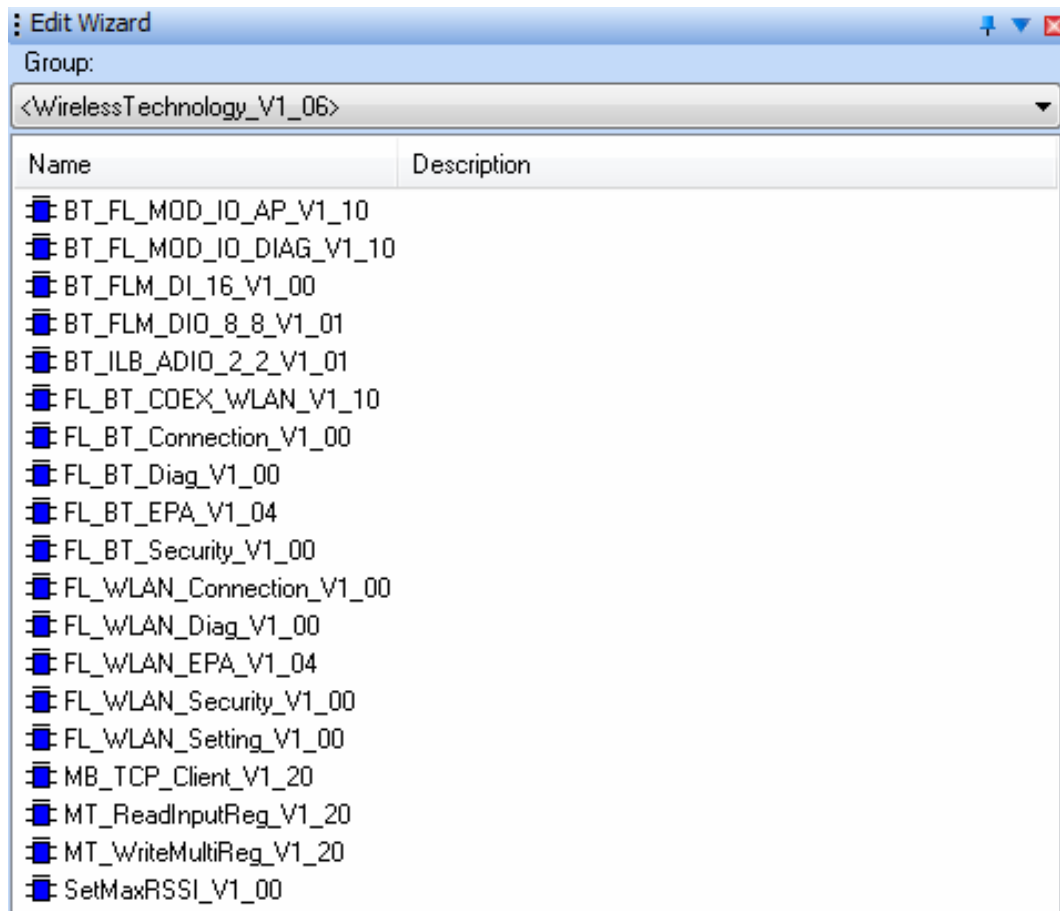


Fig. 3.23. List of selected library elements

Test questions

1. How can the Project Wizard be started and what are the benefits of using it?
2. How can the folder for saving the files of the created project be specified?
3. How can the base programming language be selected in the project?
4. How can the family of processors used in the controller be selected?
5. How can the controller family and model be selected?
6. How can the characteristics of the created project before the completion of the work of the Wizard be selected?
7. What elements appear in the created project after the completion of the work of the Wizard?
8. What do the letters "T" and "V" mean in the names of the elements of the created project?
9. How can a library created in another project be added to the project?

3.3. Configuring PC Worx when working with ILC 151 GSM/GPRS controller

Before configuring PC Worx the computer and the controller must be physically connected using a patch cord. On the computer side the cable should be connected to the network interface card, and from the ILC 151 GSM/GPRS controller to the RJ45 interface on the front panel (Fig. 3.24).

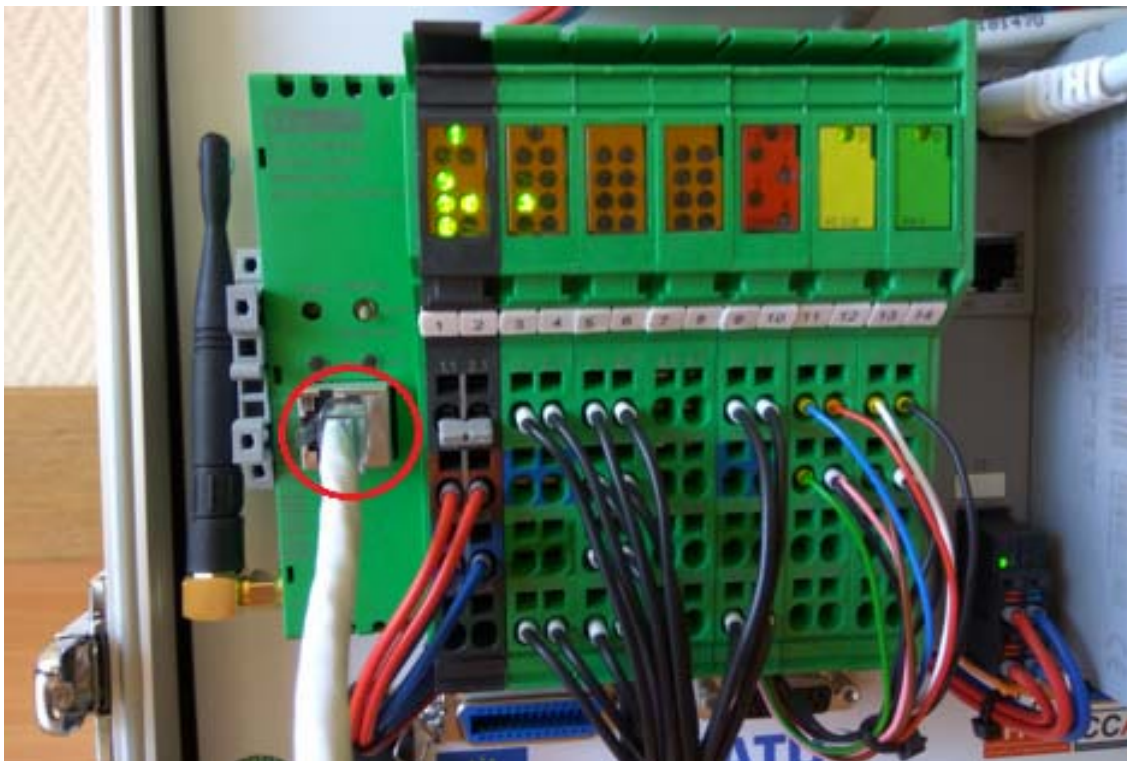


Fig. 3.24. Connecting the ILC 151 controller

In order to run PC Worx select the path: Start => All Programs => Phoenix Contact => AUTOMATIONWORX Software Suite 2015 1.82 => PC WORX 6.30.1202. If the information window appears (Fig. 3.25) with the warning that the program will work in demo mode, you either have to agree to this mode or contact the distributor to purchase the license.

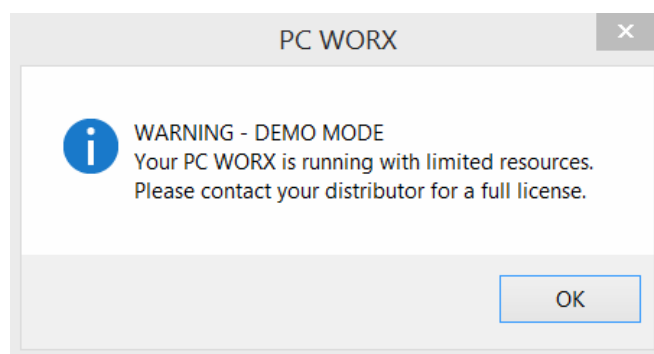


Fig. 3.25. Demo mode warning

In order to create a new project select File => New project, as described in 3.2 (Fig. 3.26).

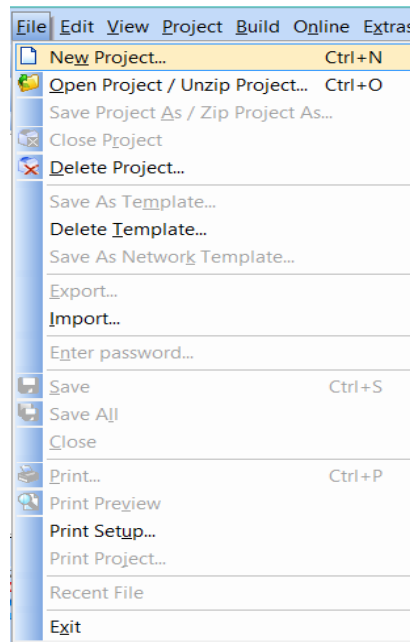


Fig. 3.26. New project creation tab

After that select the tab with the required ILC 1xx controller class (Fig. 3.27), highlight ILC 151 GSM/GPRS Rev. > 00/4.10 in the controller list and press the OK button.

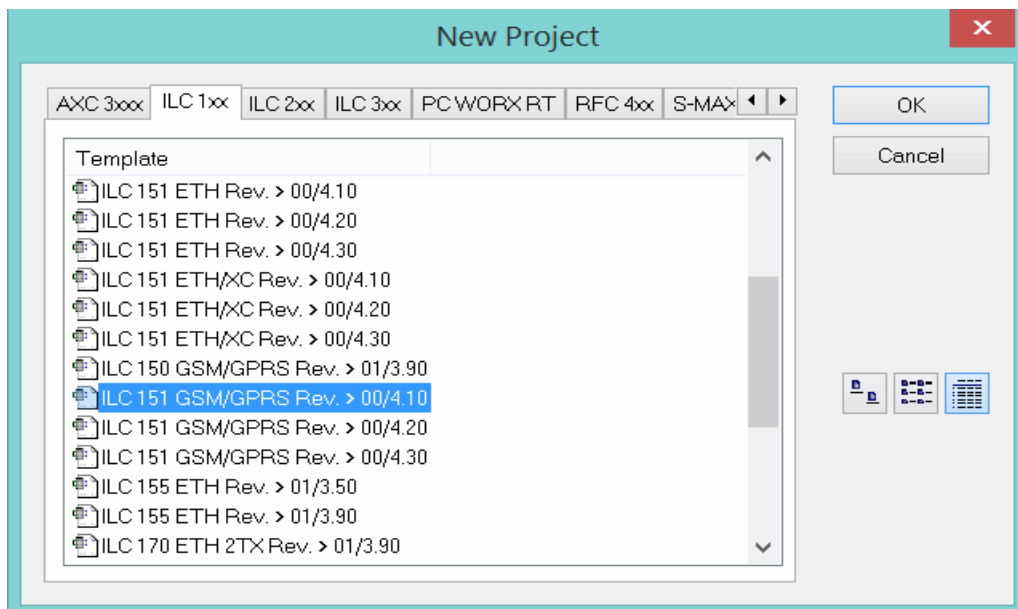


Fig. 3.27. Controller selection window

For some time the program performs internal configuration of the selected controller. Then View => Bus Configuration (in earlier versions of Interbus Connections) should be selected in the top menu.

After switching to this mode the interface shown in Fig. 3.28 appears. The Device Details window in the Project tab included at the bottom of the window shows general information about the project.

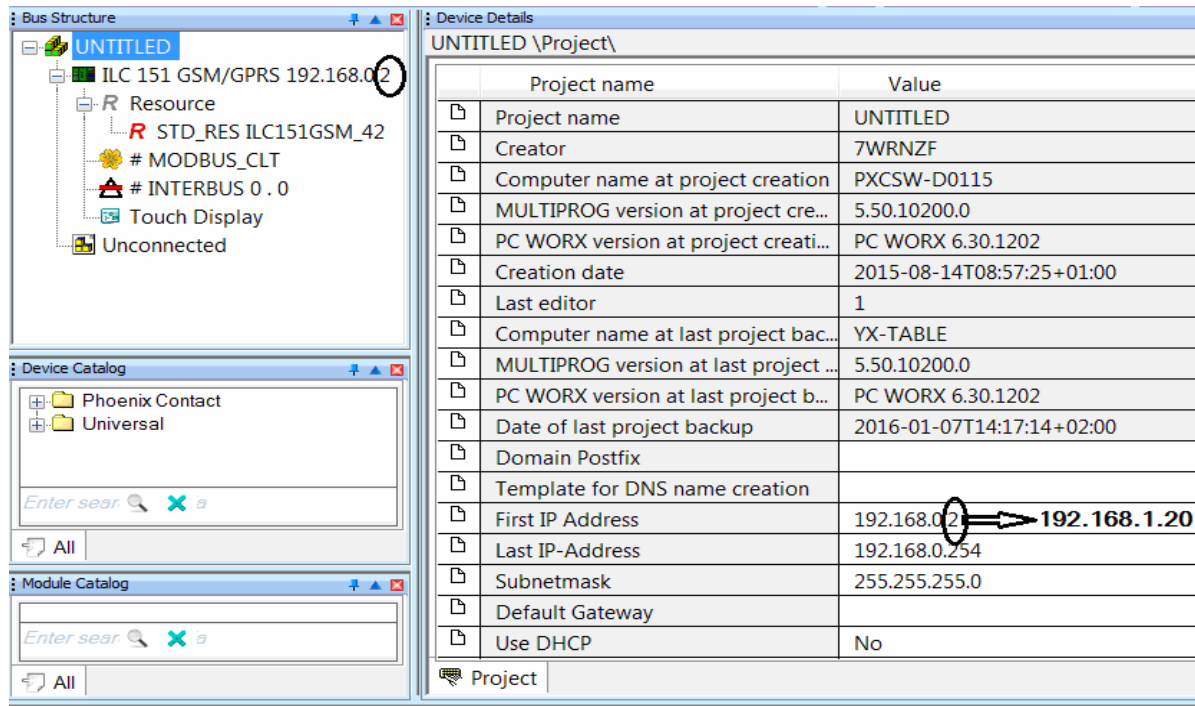


Fig. 3.28. Project Properties

By default, the IP Address of the project is set to the same as that of the controller – 192.168.0.2. It should be changed: 192.168.0.20.

In order to configure the controller, the ILC 151 GSM/GPRS controller having a tree structure should be highlighted in the Bus Structure window, as shown in Fig. 3.29.

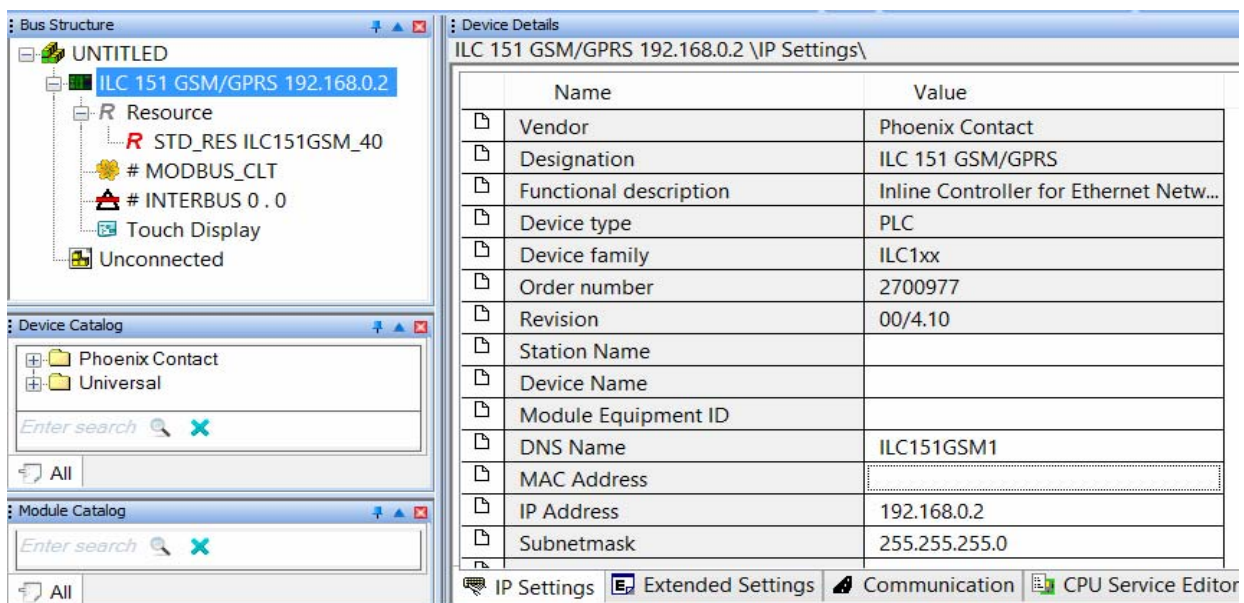


Fig. 3.29. ILC 151 GSM/GPRS features

The Project tab in the Device Details window will be replaced with several tabs. The first tab (IP Settings) informs about the configured controller should be entered. MAC Address written on the front panel of the controller (00-A0-45-8D-27-C3), should be entered as shown in Fig. 3.30. Then IP Address should be entered in accordance with the documentation – 192.168.1.20, as shown in Fig. 3.31.

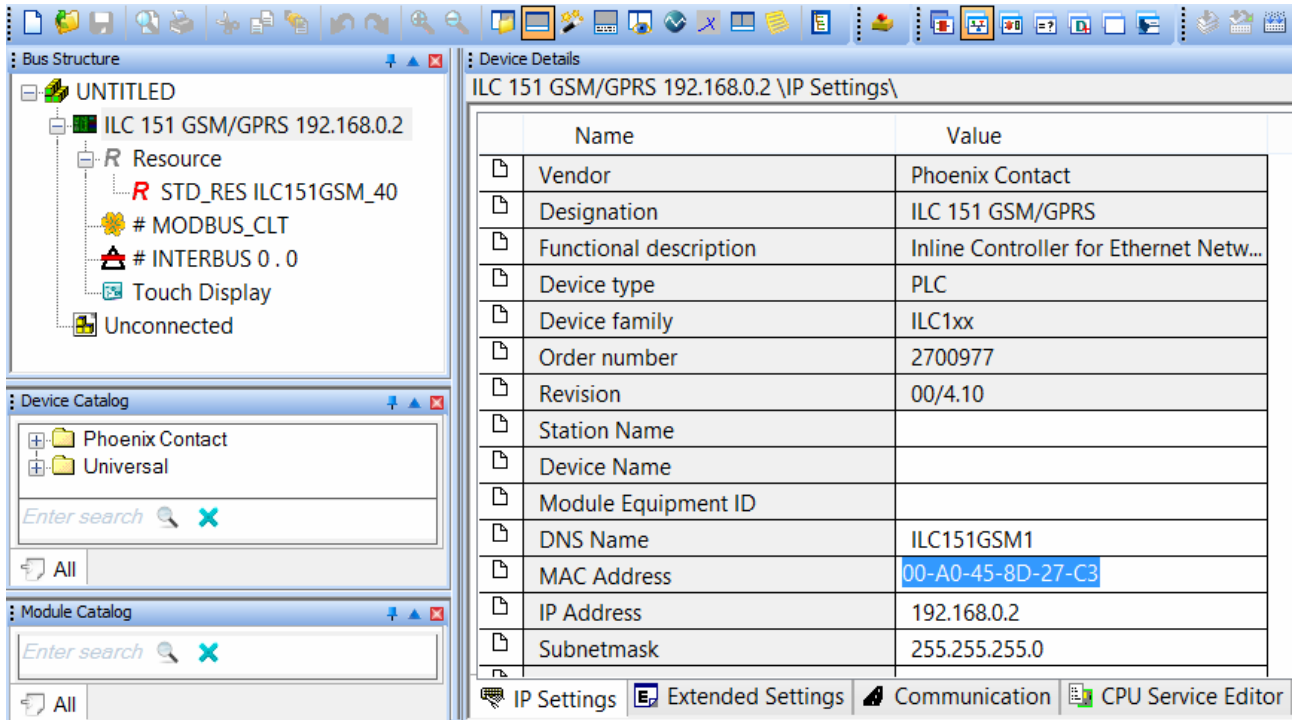


Fig. 3.30. Entering MAC Address

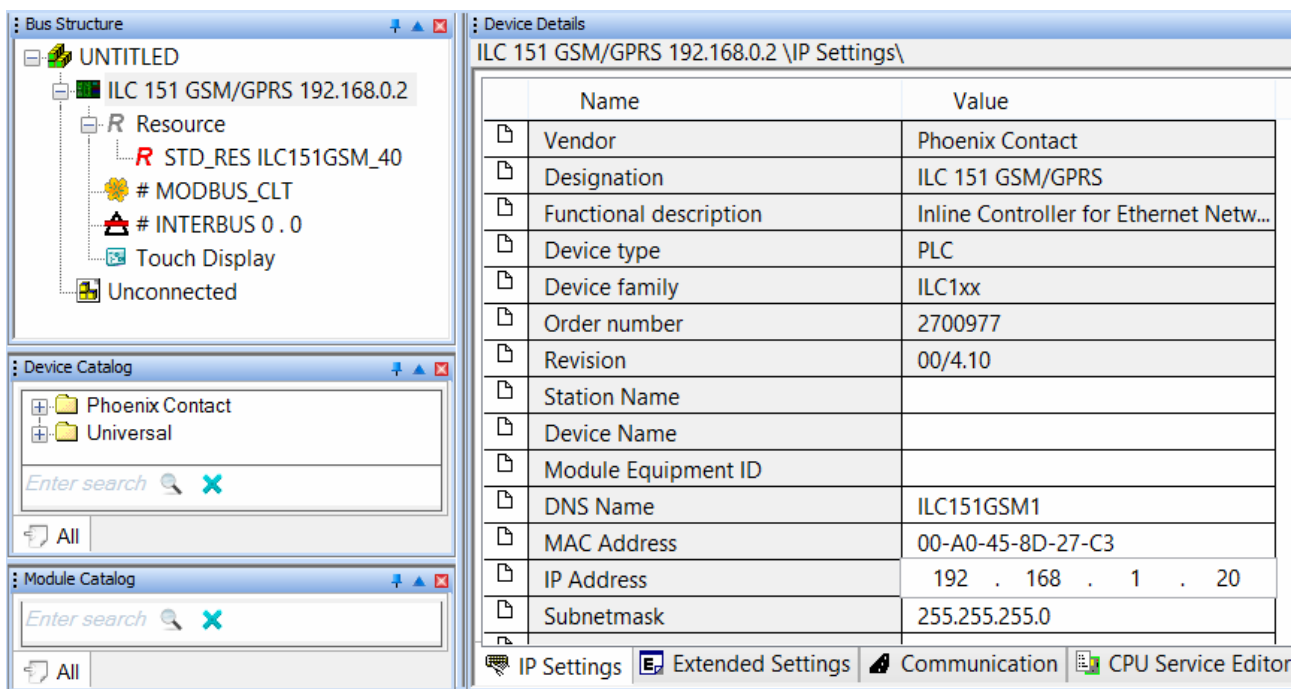


Fig. 3.31. Entering IP Address

After that it is recommended to go to the Communication tab of the Device Details window (as shown in Fig. 3.32) to check the connection between the personal computer and the controller.

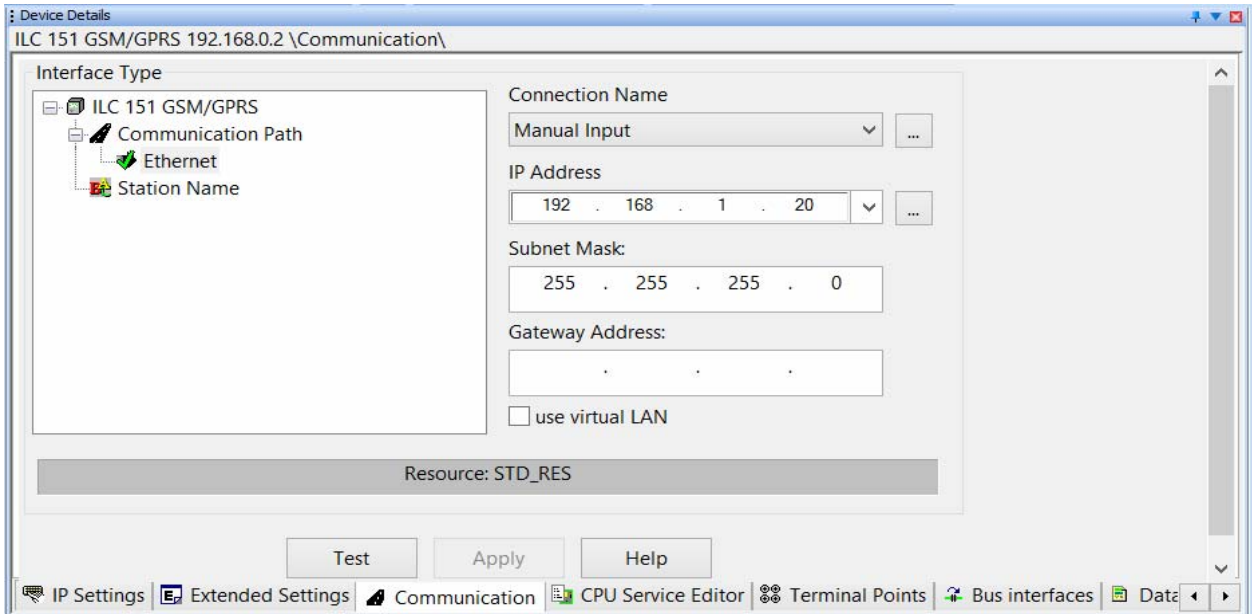


Fig. 3.32. Communication check window

Here in the Interface Type section the list of interfaces used by the controller works is listed. The right part of the window shows the IP Address entered in the IP Settings tab.

In order to check the connection with the controller, press the Test button. The result is reported on the Resource: STD_RES progress bar.

If the response is positive, the progress bar will turn green as shown in Fig. 3.33. Otherwise, the progress bar will be red.

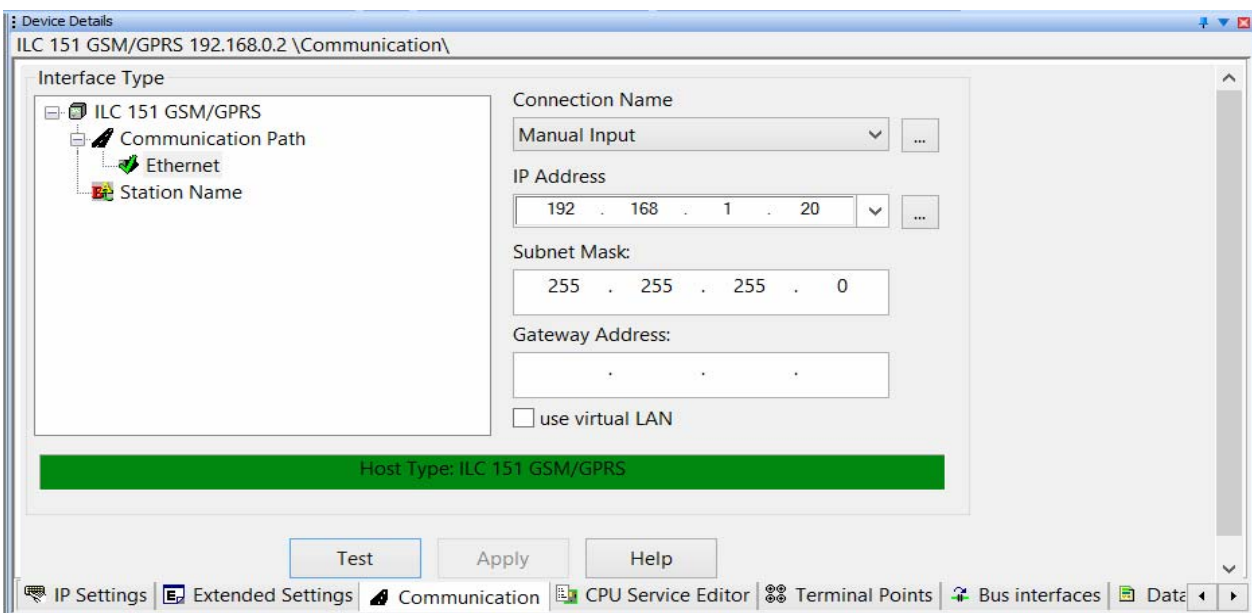


Fig. 3.33. Successful completion of communication test

The next stage of configuring is the connection of additional modules. In order to do this, select the Connected INTERBUS item in the View top menu (Fig. 3.34).

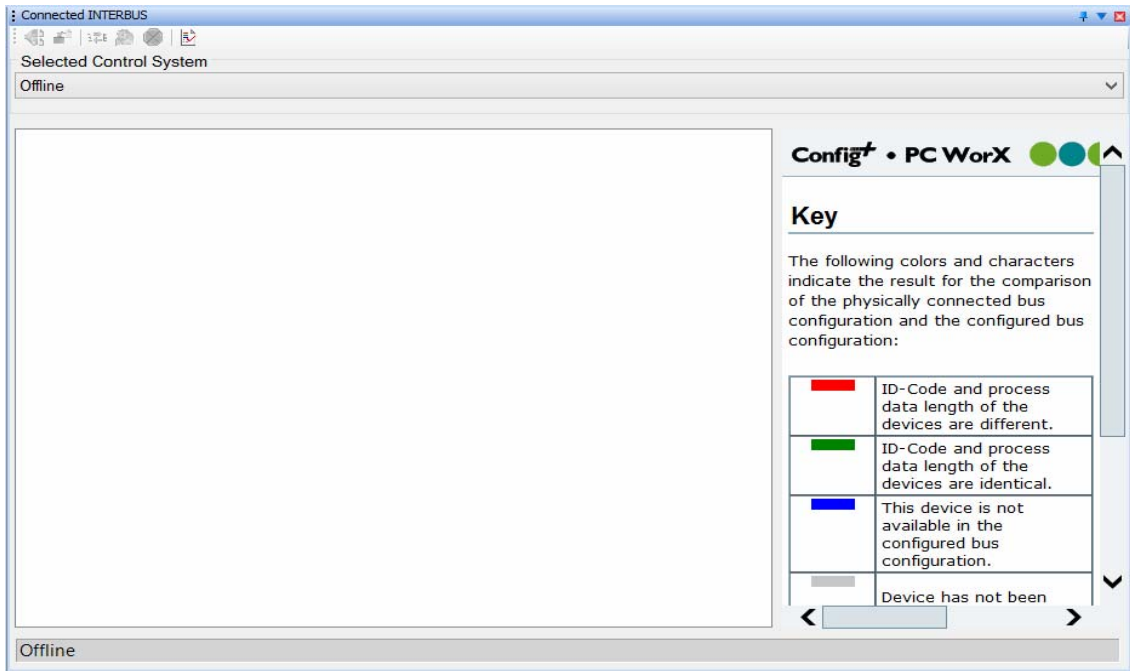


Fig. 3.34. Connected INTERBUS window

In the Selected Control System section the drop-down menu is opened as shown in Fig. 3.35. There it's possible to select the connected controller ILC 151 GSM/GPRS (192.168.1.20).

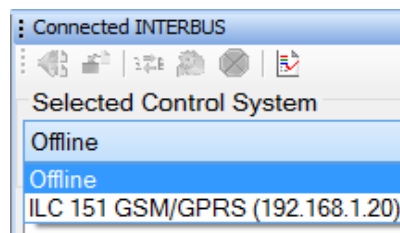


Fig. 3.35. Selecting the ILC 151 GSM/GPRS controller

The system will detect the presence of three modules (Fig. 3.36).

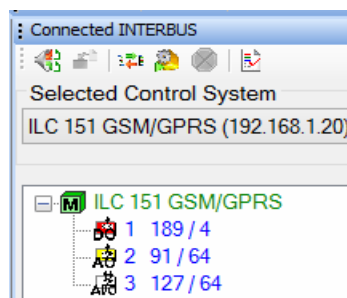


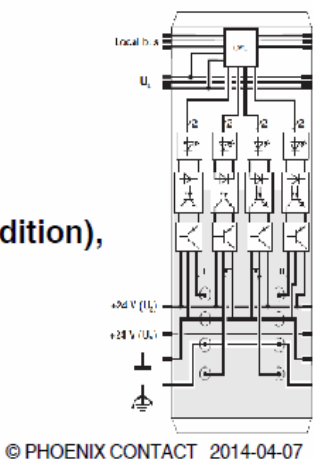
Fig. 3.36. Detecting of plug-in modules

The names of additional modules can be found on the front panel of each of them. They are presented in Fig. 3.37.

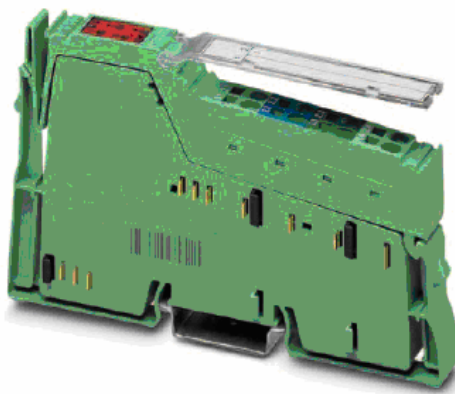
IB IL 24 DO 4-ME

Inline digital output terminal,
Inline ME versions (Machine Edition),
4 outputs, 24 V DC, 500 mA

Data sheet
7036_en_01



© PHOENIX CONTACT 2014-04-07



a

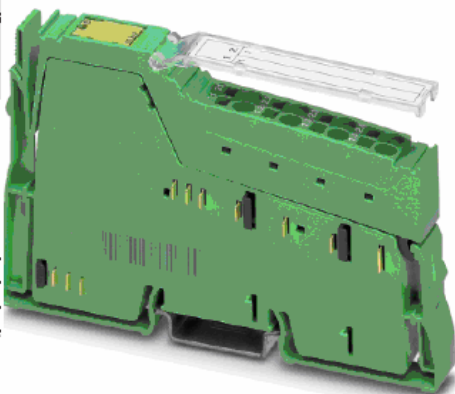
IB IL AO 2/UI-PAC

Inline analog output terminal, 2 outputs for
connection of current and voltage signals

Data sheet
8195_en_03



© PHOENIX CONTACT 2014-12-05

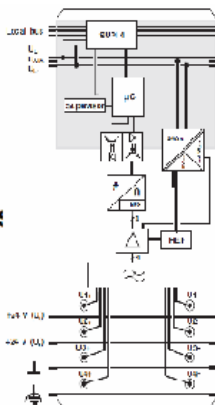


b

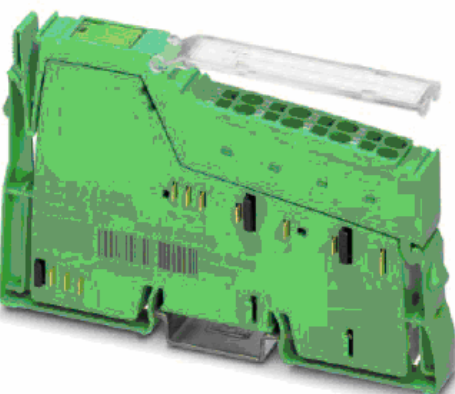
IB IL AI 4/U-PAC

Inline analog input terminal,
4 inputs for connecting voltage signals

Data sheet
8081_en_01_C01



© PHOENIX CONTACT 2015-04-14



c

Fig. 3.37. Appearance of the modules: *a* – IB IL 24 DO 4-ME;
b – IB IL AO 2/UI-PAC; *c* – IB IL AI 4/U-PAC

The blue color of the displayed modules shows that they have not yet been imported into the project. By right-clicking the mouse it's possible to activate the drop-down menu (Fig. 3.38), where it's possible to select the Import to Project item.

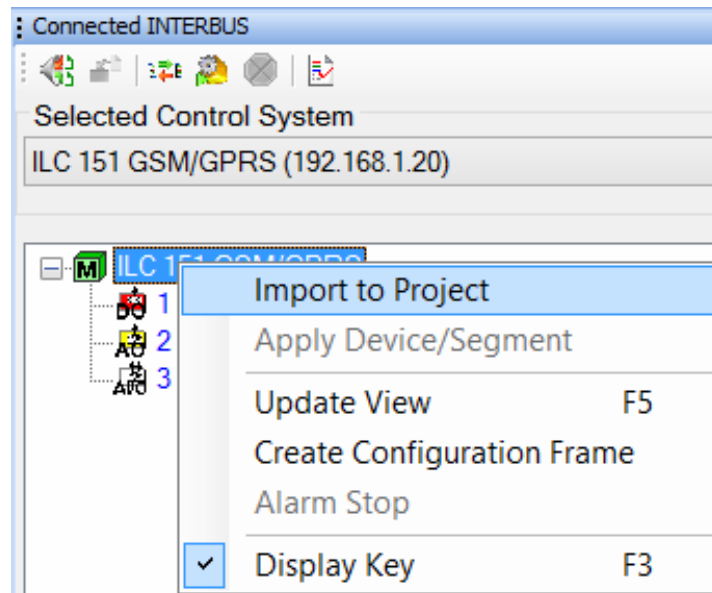


Fig. 3.38. Importing additional modules into the project

After that the wizard is enabled. By answering the questions when selecting additional modules, the latter are colored in green. The final window of the wizard is shown in Fig. 3.39.

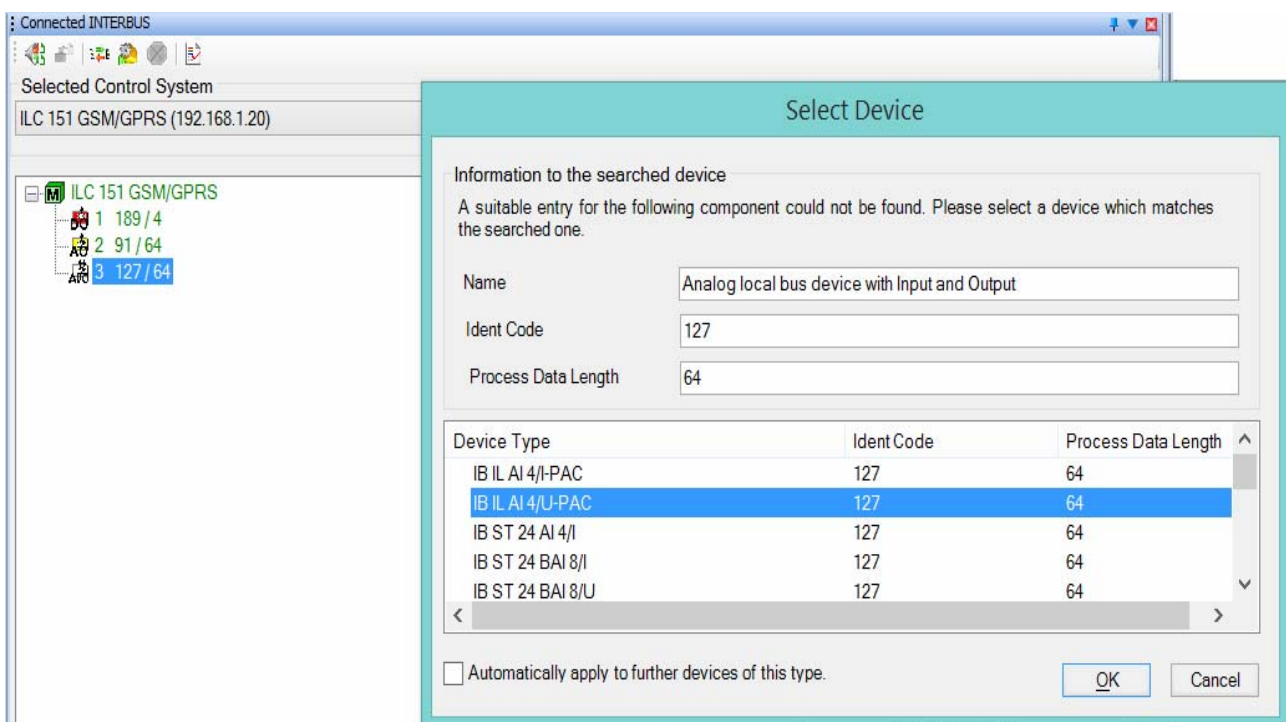


Fig. 3.39. Window for importing the additional modules

ILC 151 GSM/GPRS controller with the additional modules has been configured. In order to start programming you need to switch to this mode. Select the "IEC Programming" item in the View top menu and the interface takes the appearance shown in Fig. 3.40.

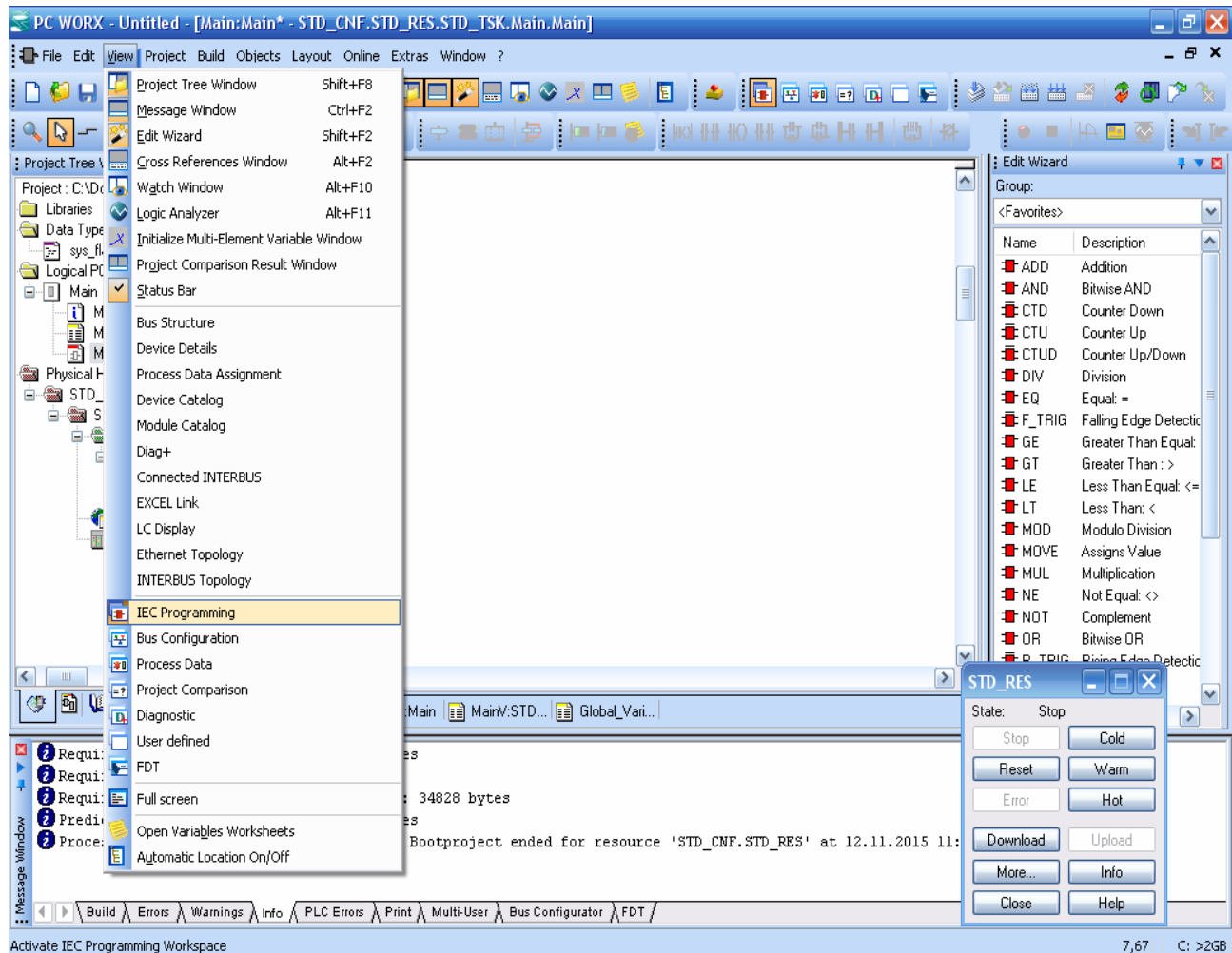


Fig. 3.40. PC Worx interface in the IEC Programming mode

Test questions

1. How can a new project be created?
2. How can the controller configuration mode be entered?
3. Where is the information on the controller MAC address displayed?
4. How can the connection between the personal computer and the programmable controller be checked?
5. How can the additional modules be added when configuring the controller?

3.4. Configuring PC Worx when working with AXC 3050 controller

In order to work with the AXC 3050 controller in PC Worx a new project should be created: after starting PC Worx in the File menu select New Project or click the corresponding icon in the toolbar. In the appearing dialog box (Fig. 3.41) select the AXC 3xxx tab, and in the appearing list select the line "AXC 3050 Rev. > 01/5.50"(or a newer version of the controller software).

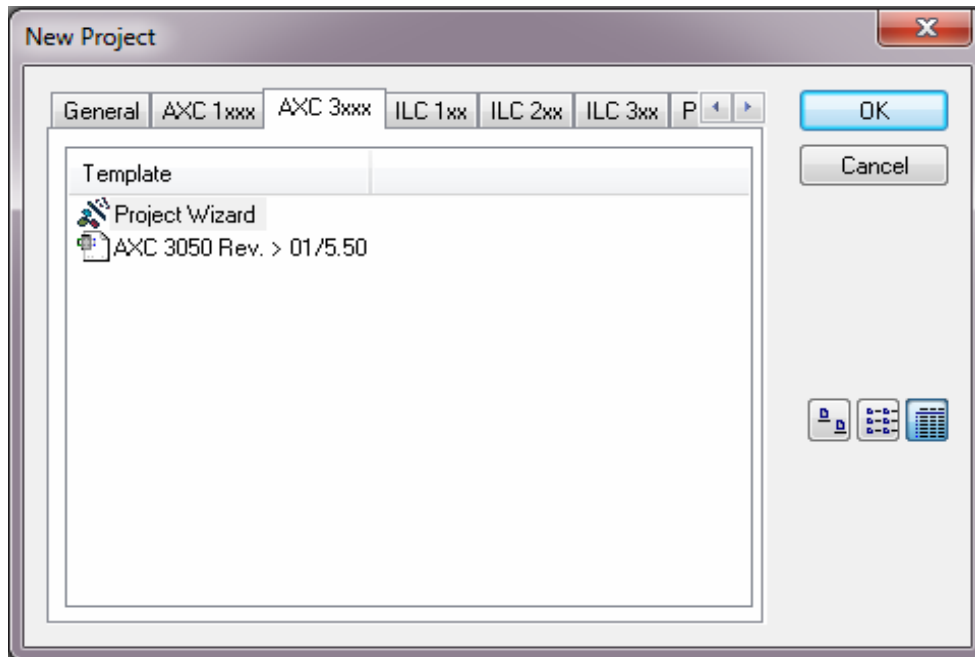


Fig. 3.41. Selection window of the type of controller and its software version

The initial controller default settings are displayed in the Device Details window (Fig. 3.42). This information about the controller is displayed if the line with the project name (if it has already been saved) or the UNTITLED line (if the project has not been saved) is selected in the Bus Structure window (Fig. 3.43).

The AXC 3050 controller has three MAC addresses: the base one, indicated in the factory label of the, and two additional ones. Each of them can be assigned to one of the three Ethernet interfaces (X1, X2 and X3). Due to the three independent interfaces the AXC 3050 controller can be used in Ethernet/PROFINET networks for managing branched and complex automation systems. Assignment is performed according to the scheme given in Table 3.1.

Table 3.1

Distribution of MAC addresses of the AXC3050 controller

Interface	Address	Example
X1	00.A0.45.XX.XX.XX	00.A0.45.B0.B3.97
X2	00.A0.45.XX.XX.XX + 02hex	00.A0.45.B0.B3.99
X3	00.A0.45.XX.XX.XX + 04hex	00.A0.45.B0.B3.9B

Device Details

UNTITLED \Project\

	Project name	Value
📄	Project name	UNTITLED
📄	Creator	pyx129
📄	Computer name at project creation	PYRAE0087
📄	MULTIPROG version at project creation	5.48.592.6
📄	PC WORX version at project creation	PC WORX 6.30.767
📄	Creation date	2014-02-10T10:48:08+01:00
📄	Last editor	pyx129
📄	Computer name at last project backup	PYRAE0087
📄	MULTIPROG version at last project backup	5.48.592.6
📄	PC WORX version at last project backup	PC WORX 6.30.767
📄	Date of last project backup	2014-02-10T10:48:31+01:00
📄	Domain Postfix	
📄	Template for DNS name creation	
📄	First IP Address	192.168.0.2
📄	Last IP-Address	192.168.0.254
📄	Subnetmask	255.255.255.0
📄	Default Gateway	
📄	Use DHCP	No
📄	Certificate information	
📄	Organization	PHOENIX CONTACT GmbH _Co. KG
📄	Organizational Unit	
📄	Locality	Blomberg
📄	State or Province	Nordrhein-Westfalen
📄	Country	DE

Project

Fig. 3.42. Standard settings of the AXC3050 controller

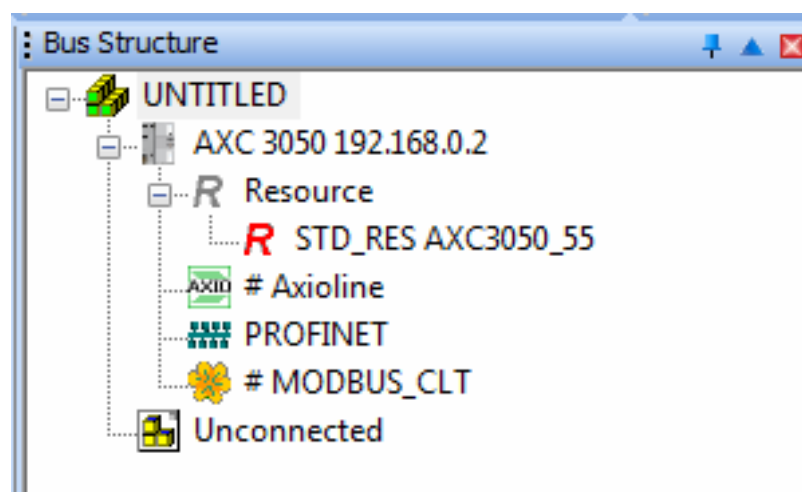


Fig. 3.43. Bus Structure window

00.A0.45 is the first part of MAC address, which is the same for all devices of the same manufacturer; it consists of six hexadecimal characters or 24 bits; XX.XX.XX is the second part of MAC address, which is unique to the particular device of the specified manufacturer and also consists of six hexadecimal characters or 24 bits.

You can assign the IP address that depends on the project to the corresponding Ethernet interface (X1, X2 or X3) depending on the MAC address. This is done in PC Worx by the BootP server.

Operation with the AXC 3050 controller depends on which of the three Ethernet interfaces is used for connecting the remaining devices of the network. Each of the three interfaces must have its own unique for the local network IP address. In this case the IP addresses of interfaces X1, X2 and X3 must be in different subnets as shown in Table 3.2 (the first two bytes of the address should match, and the third must be different).

To operate the AXC 3050 controller in the PROFINET controller (master) mode, the IP address must be assigned to the X3 interface.

To operate the AXC 3050 controller in the PROFINET device (slave) mode, the IP address must be assigned to any of the X1, X2 or X3 interfaces. The operating mode of the AXC 3050 controller as a PROFINET device must be enabled in the PC Worx (by default it is disabled).

Computer with PC Worx installed is connected to the AXC 3050 controller using the X3 interface.

Table 3.2

Assigning IP addresses to controller Interfaces

Interface	IP address	Subnet mask
X1	192.168.1.2	255.255.255.0
X2	192.168.2.2	255.255.255.0
X3	192.168.0.2	255.255.255.0

X1, X2, and X3 interfaces of the AXC 3050 controller can be assigned IP addresses when connecting the computer via Ethernet or USB interfaces.

The advantage of assigning IP addresses via the USB interface is that you do not need to perform any network settings of the personal computer. You need to connect via the PROG (X4) USB interface to the computer running PC Worx (the USB driver for the controller must be installed on the computer). The computer will automatically detect the presence of a programmable controller. Then in PC Worx switch to the Bus Configuration window using the View menu item, select the AXC 3050 controller in the device list, and then select the Extended Settings tab in the Device Details window (Fig. 3.44).

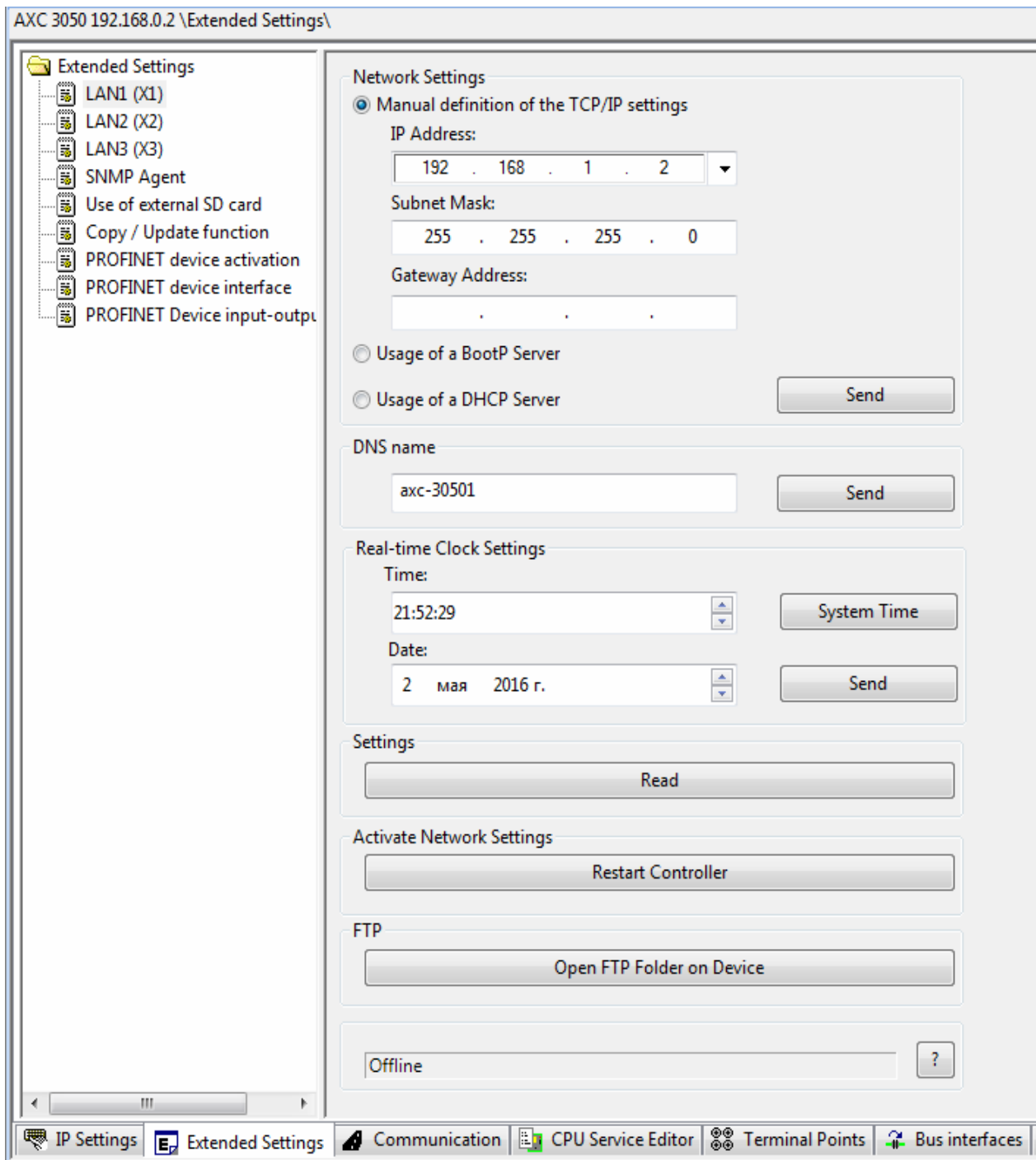


Fig. 3.44. Controller settings when connected to computer via USB

After that, select the interface (for example, X1) in the left part of the window, and in the right part of the window manually assign the IP address and subnet mask, then repeat these actions for the X2 and X3 interfaces (see Fig. 3.44). Then press the upper button Send (see Fig. 3.44) to save the settings in the controller memory. In the appearing dialog box (Fig. 3.45) select the USB interface. After the message "Service executed successfully" appearing on the green background, click the Restart Controller button to activate the network settings.

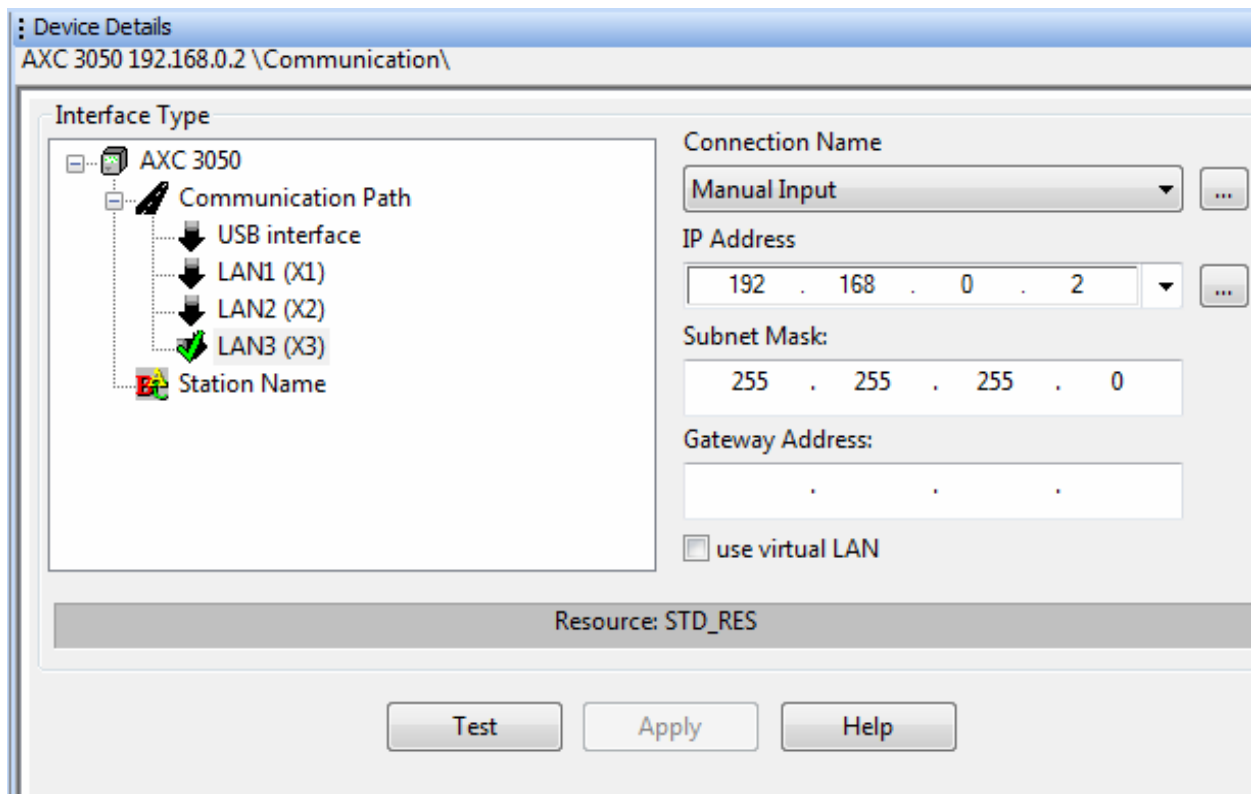


Fig. 3.45. Functional check of the controller settings

On the Communication tab it's possible to see a list of interfaces that allow exchanging data between the computer and the controller. After selecting the desired interface and pressing the Test button (the settings should be correct) the message "Service executed successfully" appears on the green background.

IP addresses can also be assigned using a BootP server in the Ethernet networks (using the Bootstrap protocol). The AXC 3050 controller, which acts as a BootP client, sends a Boot_Request broadcast request over the network via one of the Ethernet interfaces (X1, X2 or X3). The MAC address of the sending device is sent along with the Boot_Request request to uniquely identify the device. If the BootP server is activated in PC Worx, PC Worx sends a Boot_Reply response on the network to inform the AXC 3050 controller about its IP address and subnet mask. In this case, the BootP server must "know" the MAC address sent by the BootP client. For this MAC address in PC Worx a corresponding IP address and subnet mask must be assigned.

After the IP data is successfully transferred to the AXC 3050 controller, PC Worx sends the appropriate confirmation message.

By default, the BootP protocol is enabled in the controller. The IP addresses that are assigned to the controller are also sent over TCP/IP as the corresponding IP addresses that can be selected in PC Worx. After assigning the IP address PC Worx automatically establishes the connection with the controller via the TCP/IP protocol.

In order to set IP addresses in PC Worx do the following:
 switch to the Bus configuration window;
 select the AXC 3050 controller;
 select the IP Settings tab in the Device Details window;
 enter MAC address of the controller (Fig. 3.46).

Name	Value
Vendor	Phoenix Contact
Designation	AXC 3050
Functional description	Axiocontrol: Controller for AxioLine F IO Sys...
Device type	PLC
Device family	AXC3xxx
Order number	2700989
Revision	01/5.50
Station Name	
Device Name	
Module Equipment ID	
LAN1 (X1)	
DNS Name	axc-30501
MAC Address	00-A0-45-B0-B3-97
IP Address	192.168.1.2
Subnetmask	255.255.255.0
Default Gateway	
LAN2 (X2)	
DNS Name	axc-30502
MAC Address	00-A0-45-B0-B3-99
IP Address	192.168.2.2
Subnetmask	255.255.255.0
Default Gateway	
LAN3 (X3)	
DNS Name	axc-30503
MAC Address	00-A0-45-B0-B3-9B
IP Address	192.168.0.2
Subnetmask	255.255.255.0
Default Gateway	

IP Settings | Extended Settings | Communication | CPU Service Editor | Terminal

Fig. 3.46. Entering MAC address of the AXC 3050 controller

MAC address printed on the controller itself and starting with the hexadecimal characters "00.A0.45." must be assigned to the Ethernet interface X1.

In order to perform the configuration using the BootP server, select Extras from the PC Worx main menu, and then BootP/SNMP/TFTP-Configuration ... (Fig. 3.47).

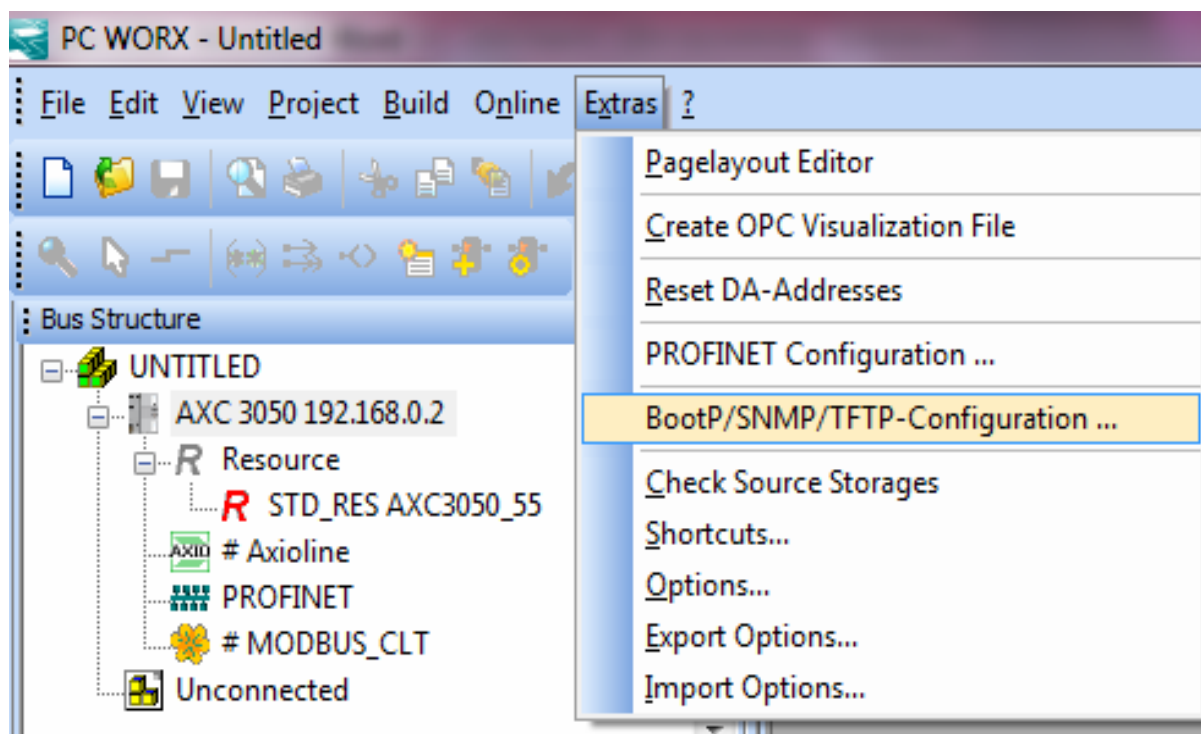


Fig. 3.47. Configuring the controller using the BootP server

The BootP server is activated by pressing the corresponding button (Fig. 3.48).

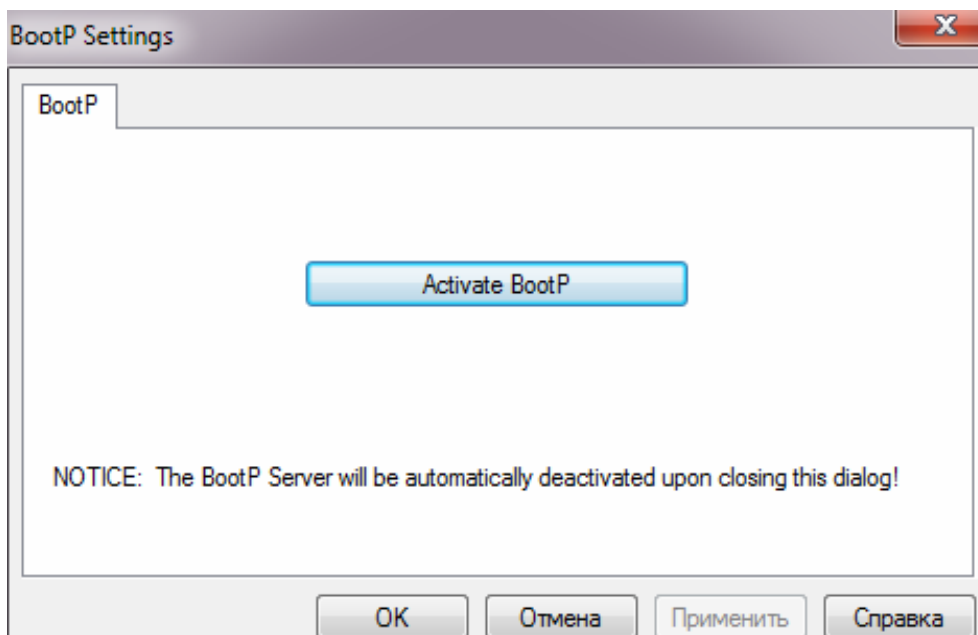


Fig. 3.48. Activating the BootP server

Then perform a "cold" restart of the controller: turn off the power and turn it on again in two seconds.

The controller will receive the IP address assigned for it in the project. In the "Bus Configurator" tab of the message window at the bottom of the screen the message will appear shown in Fig. 3.49.

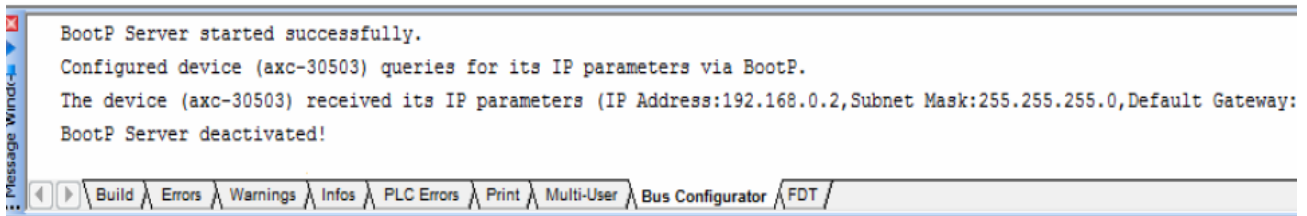


Fig. 3.49. Message confirming the successful launch of the BootP server

Now the IP address is permanently stored in the external flash memory of the controller. The described process should be repeated for all Ethernet interfaces if they require assigning IP addresses.

Next, deactivate the BootP server by clicking the appropriate button (Fig. 3.50).

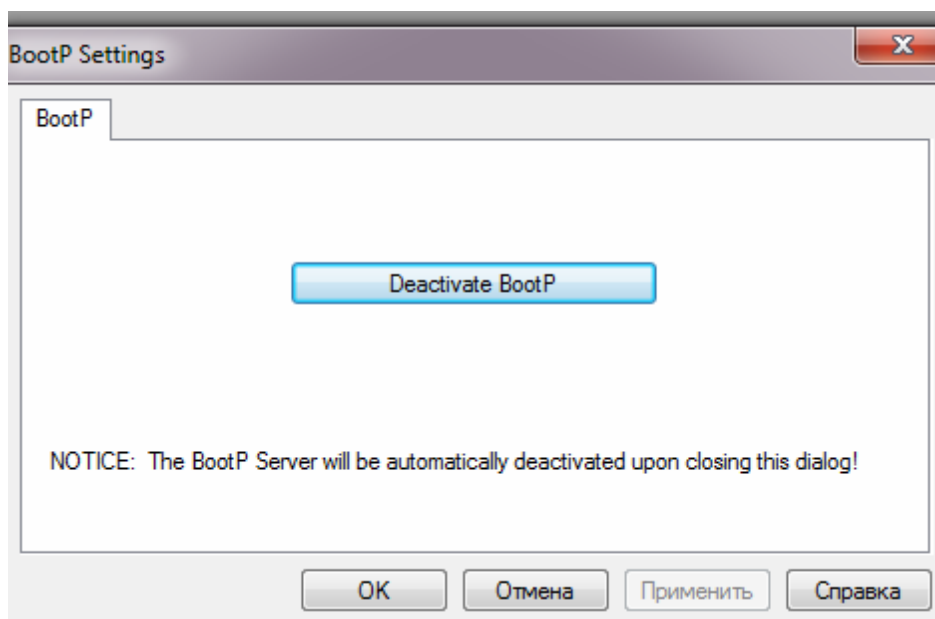


Fig. 3.50. Deactivating the BootP server

PC Worx allows the use of modules connected to the controller via the Axioline bus (a new version of the I/O bus used in all AXC family controllers) to work in the project. In order to do this, select the line "Axioline" in the "Bus Structure" window and by right-clicking on it select "Read Axioline" in the appearing context menu (Fig. 3.51).

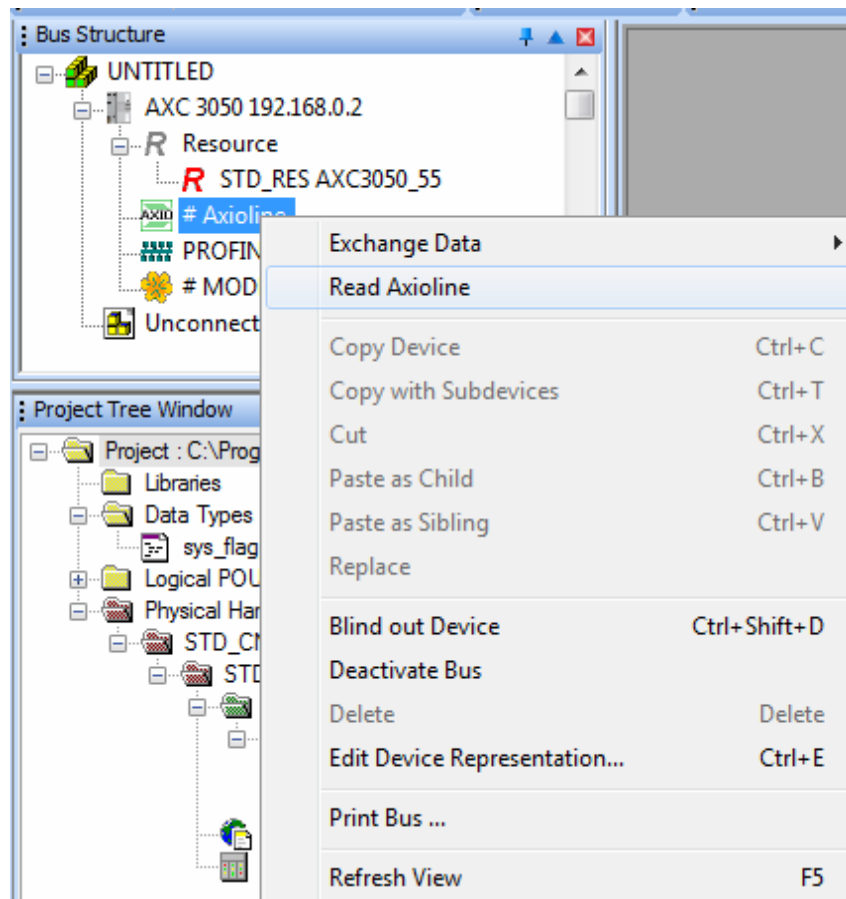


Fig. 3.51. Preparing for reading the parameters of the modules connected to the controller

The Axioline local bus supports any data transfer protocols based on the Ethernet family networks. It can be used for sequential installation of various modules (devices) one closely to another, allows connecting up to 63 devices, and has a typical cycle time of about 10 ms.

The list of all modules connected to the controller via the Axioline bus is displayed in the opening "Read Axioline" window (Fig. 3.52). Black color indicates the parameters of the modules that were not read by the PC Worx project. Green stands for the modules that are already been used in the project (their data has been read before).

If the "Re-insert all modules" option is enabled, the modules which data has already been read and entered into the project will be entered into the project again by re-reading. This data will be written "over" the old one.

If you answer positively to the question "Read Axioline ~ Do you want to delete all configured" Axioline modules and accept the connected modules?" the parameters of the devices connected to the Axioline bus will be read and added to the PC Worx project.

The identified devices connected to the controller via the Axioline bus with their default configuration are displayed in the "Bus Structure" window. Detailed information on each device is displayed on the right side of the window (Fig. 3.53).

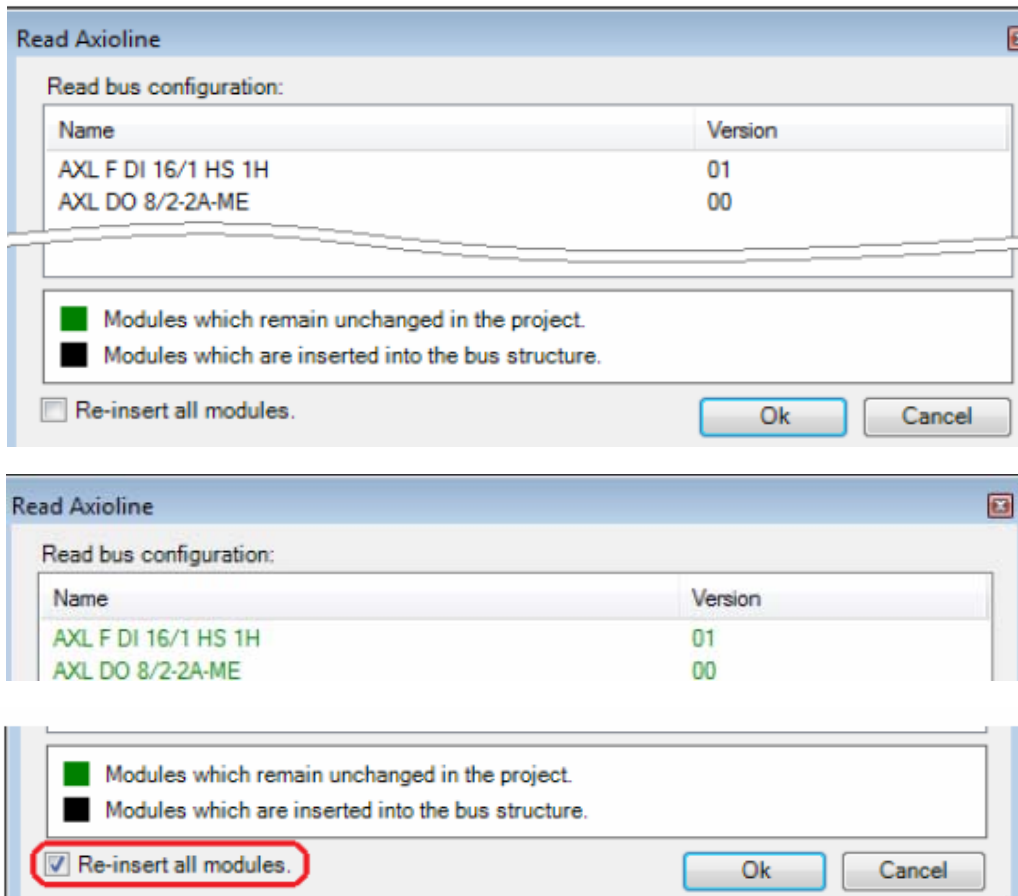


Fig. 3.52. Results of reading the parameters of the modules connected to the controller

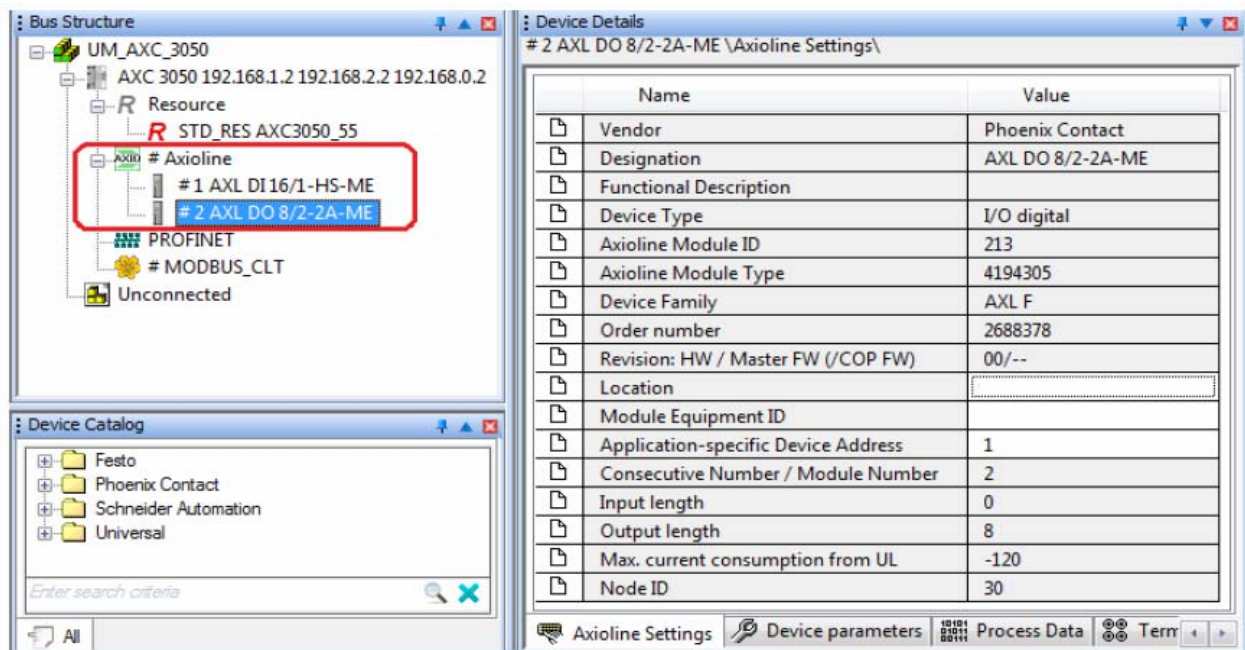


Fig. 3.53. Adding Axioline devices connected to the bus to the PC Worx project

Let us consider configuring of the AXC 3050 controller, which is used as a PROFINET controller with the connected PROFINET BK bus coupler.

In order to use the AXC 3050 controller in the PROFINET network, it must be connected via the X3 Ethernet interface and be enabled in Worx mode as a PROFINET device.

To recognize devices connected to the PROFINET network in PC Worx, select the "Extras – PROFINET Configuration" menu item (Fig. 3.54). Then select the network interface card that is used for accessing the PROFINET network devices (Fig. 3.55).

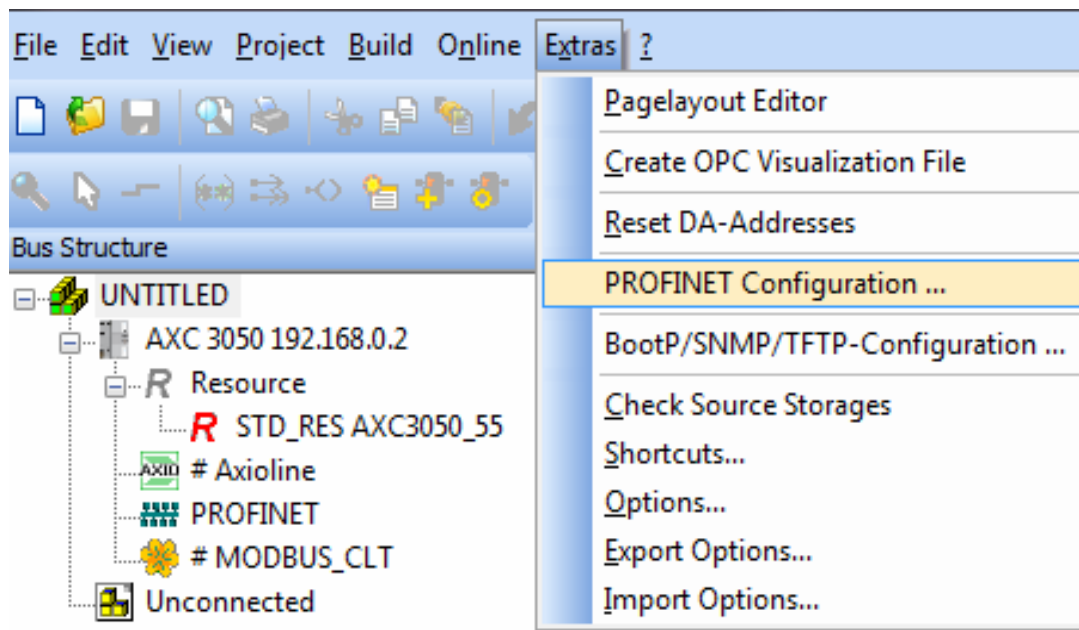


Fig. 3.54. Selecting the PROFINET network setup option

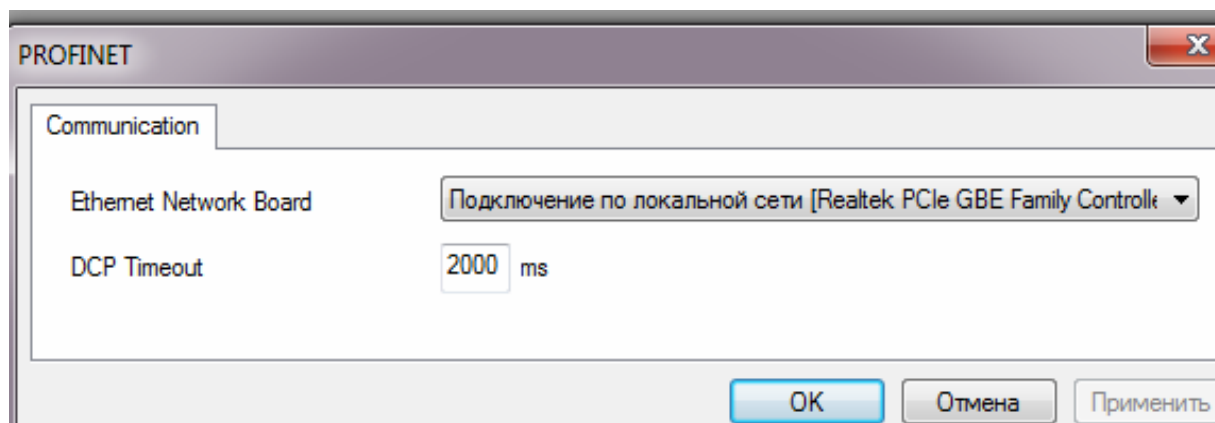


Fig. 3.55. Selecting the network interface card that provides connection to the PROFINET network

Devices can also be selected from the device catalog if you are sure that they are connected to the network.

To read data about devices connected to the PROFINET network, in the context menu appearing after right-clicking on the PROFINET line in the "Bus Structure" window select the "Read PROFINET ..." item (Fig. 3.56).

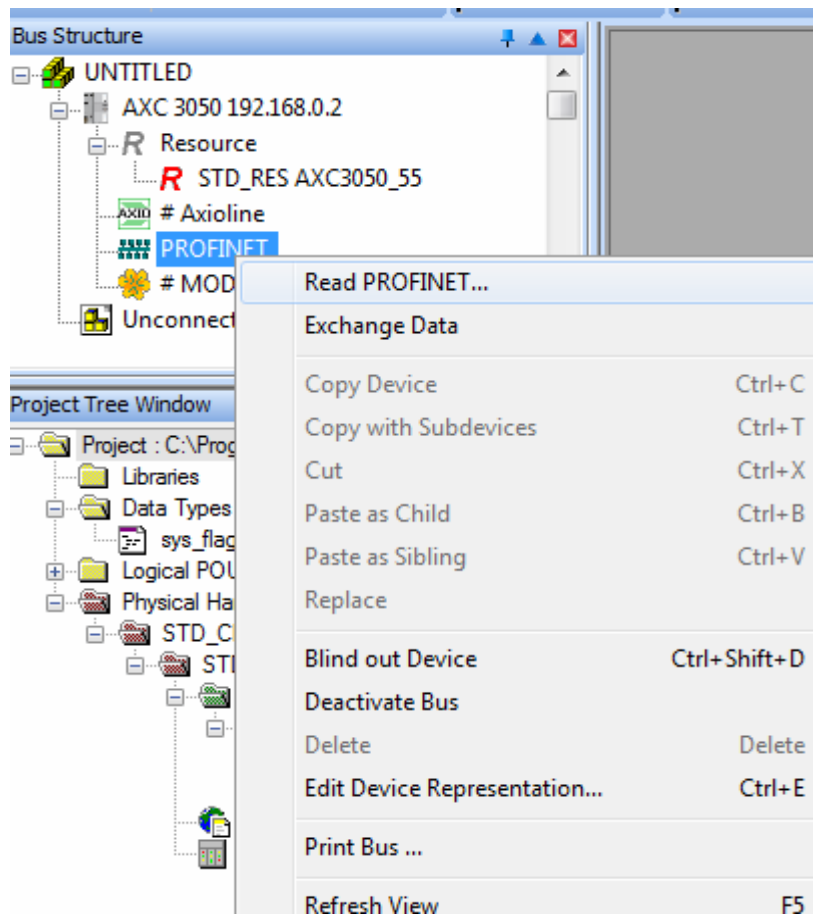


Fig. 3.56. Selecting the option of reading the PROFINET network device data

In a few seconds in the appearing window (Fig. 3.57) the list of devices that were found in the PROFINET network is displayed.

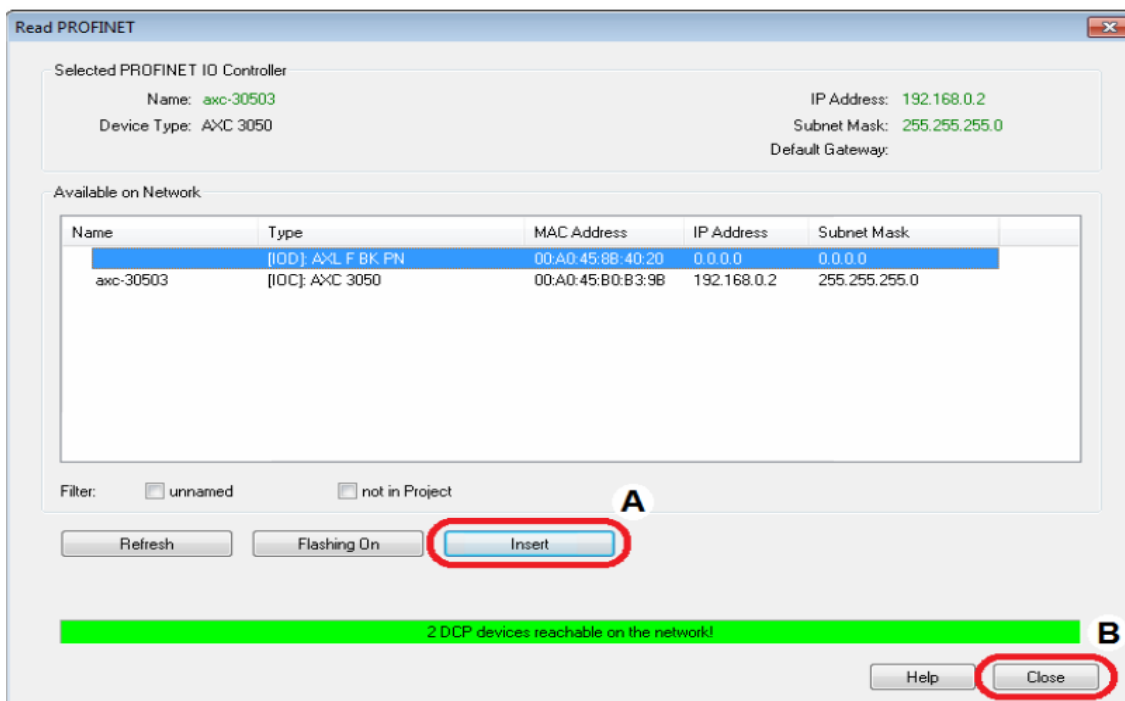


Fig. 3.57. List of PROFINET devices found in the network

Select the Axioline PROFINET AXL F BK PN BK module and add it to the project as a PROFINET device by clicking on the "Insert" button.

When the "Select PROFINET device description" window appears, to select from the list the device which name and firmware version correspond to the device actually installed (Fig. 3.58).

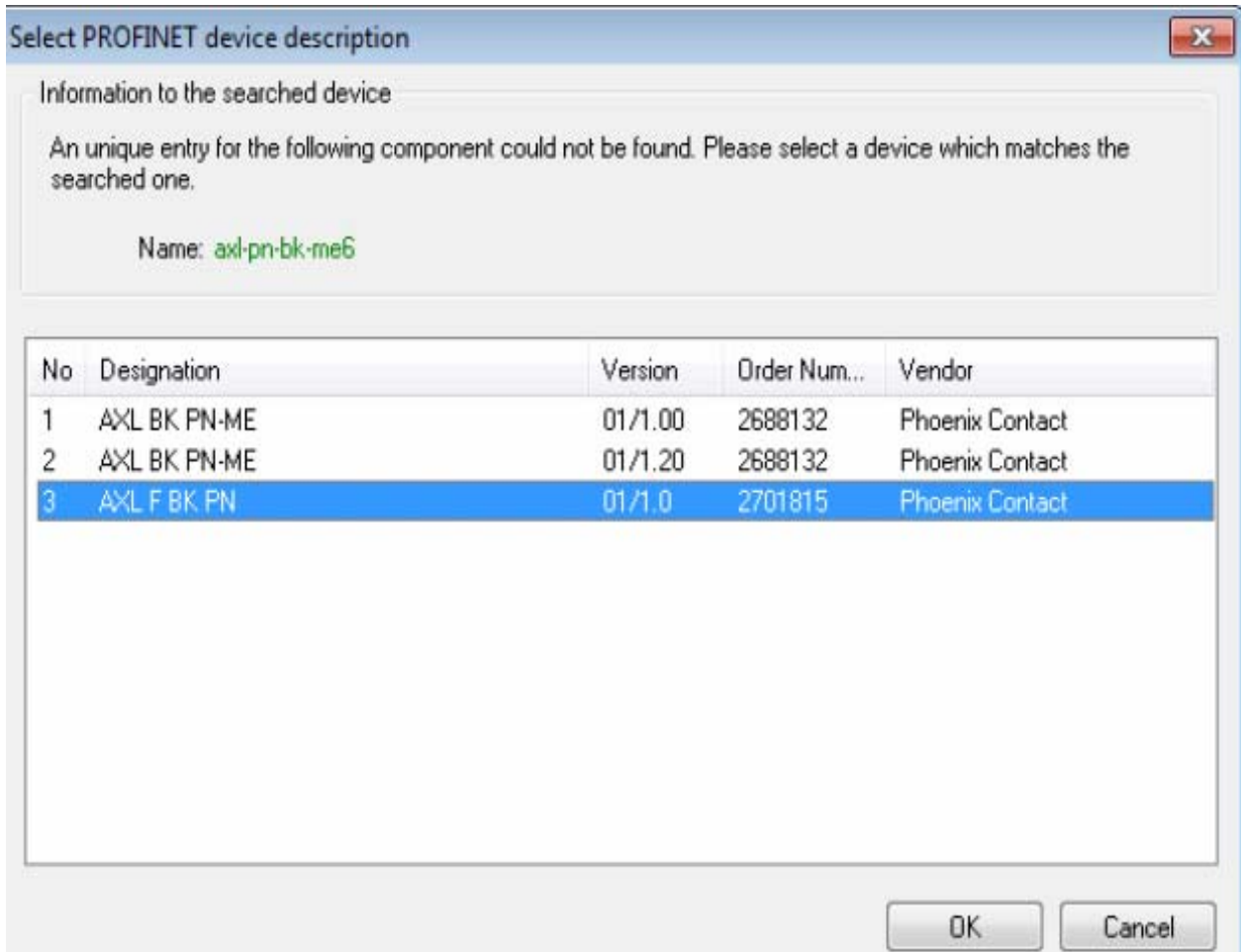


Fig. 3.58. Selecting the device with the corresponding firmware

If the name of the PROFINET device to be added has not yet been given, PC Worx will give out a corresponding request.

Similarly, by pressing the Insert button add all devices from the list to the project (see Fig. 3.57) and then click the "Close" button.

The PROFINET devices added to the project are displayed in the "Bus Structure" window (Fig. 3.59). Their characteristics can be seen by selecting the "PROFINET Settings" tab. For added devices, PC Worx automatically sets IP addresses. They can also be set by selecting the "PROFINET Stationnames" tab (see Fig. 3.59).

The parameters of the PROFINET devices can be seen by selecting the "Device parameters" tab (Fig. 3.60).

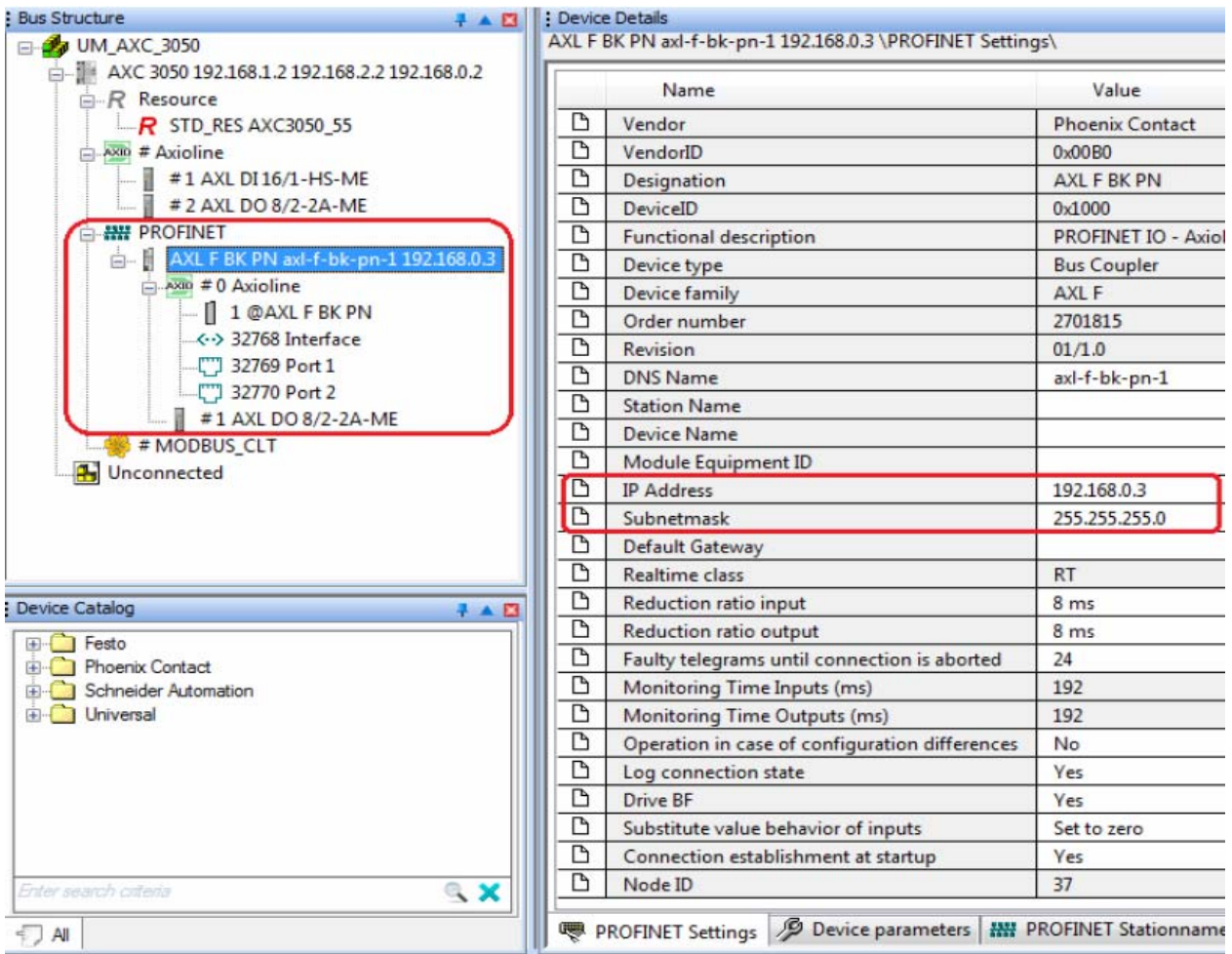


Fig. 3.59. Components and settings of the bus coupler connected to the PROFINET network

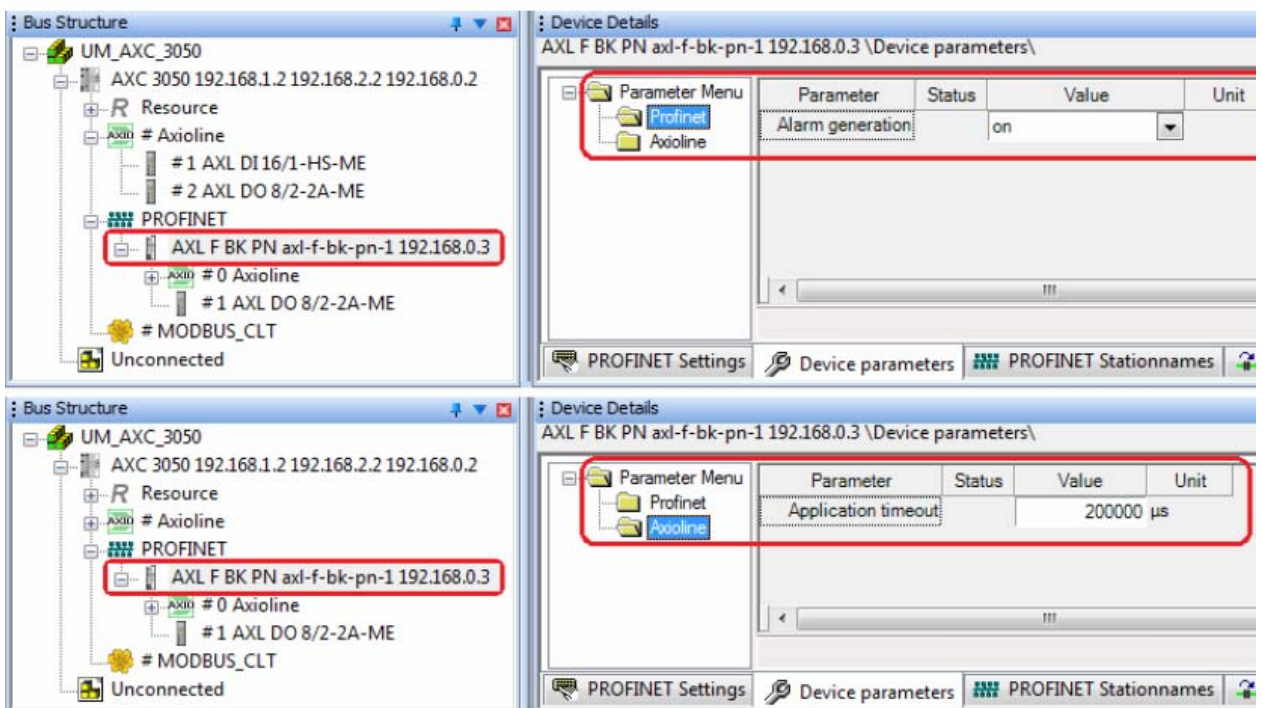
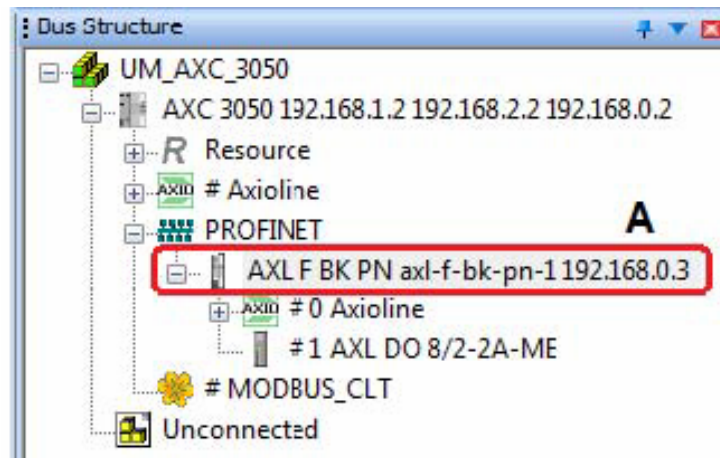
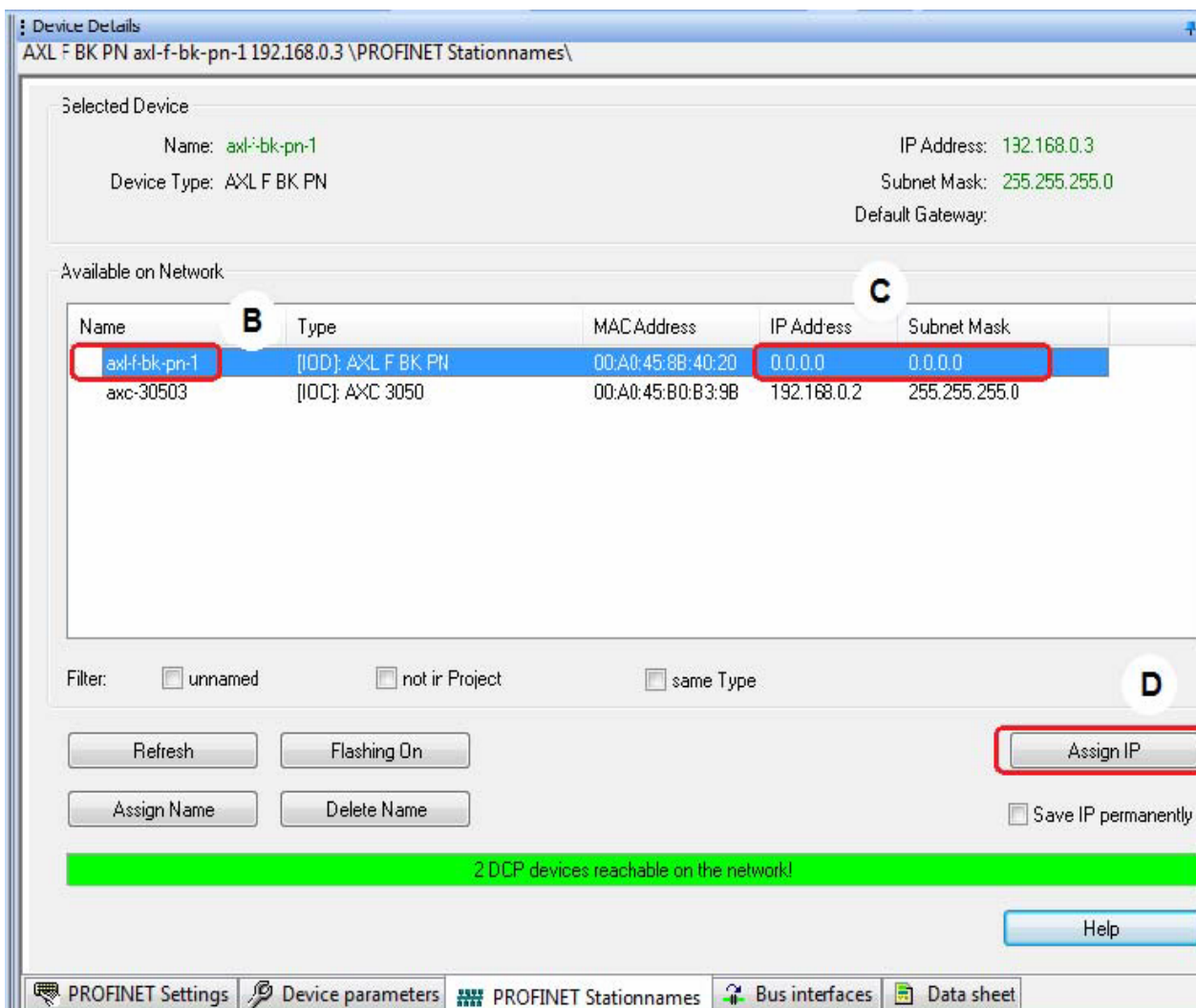


Fig. 3.60. Characteristics of the bus coupler connected to the PROFINET network

By selecting the "PROFINET Stationnames" tab (see Fig. 3.60), it's possible to assign the name and IP address to the PROFINET device that was selected in the Bus Structure window (Fig. 3.61, *b*, tags B and C).



a



b

Fig. 3.61. Changing the name and addresses of the bus coupler

In this case the AXL F BK PN module (tag A in Fig. 3.61, a) belonging to the PROFINET BK device already has the name "axl-f-bk-pn-1", but does not have the IP address. In order to assign it to the device, selected in the "Bus structure" window, press the "Assign IP" button (tag D in Fig. 3.61, b). The result of this procedure is displayed in a separate window of PC Worx (Fig. 3.62).

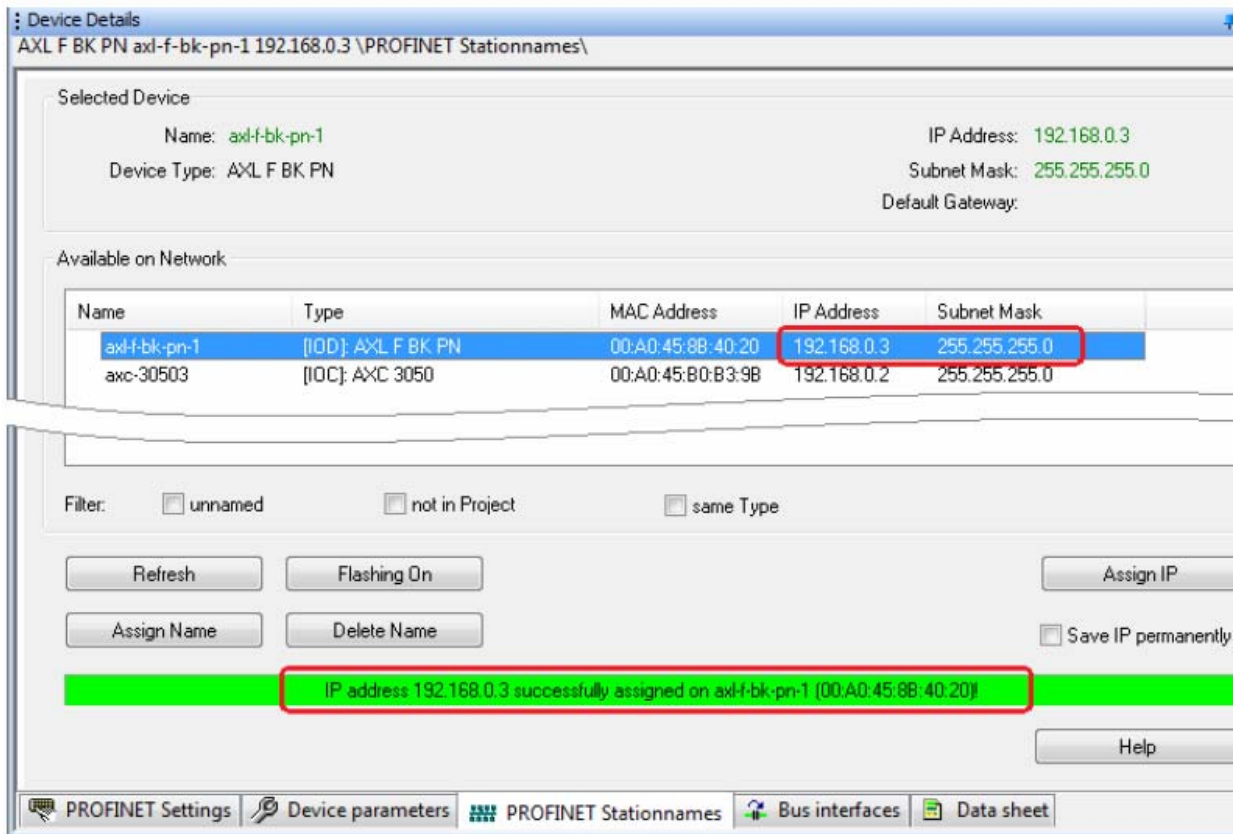


Fig. 3.62. Assigning IP Address to the bus coupler

The AXC 3050 controller can function as a PROFINET device, but this function is disabled by default. It can be turned on/off in PC Worx in one of the three Ethernet interfaces (X1, X2, X3) regardless of whether the AXC 3050 is running as a PROFINET controller.

In order to enable this function, you need to establish a USB or Ethernet connection between the AXC 3050 and the computer running PC Worx. Below it's possible to see how to enable the function by using a USB connection.

In the left part of the "Device Details" window select "PROFINET Device" in the "Extended Settings" (Fig. 3.63), and on the right side – "LAN1 (X1)" interface.

Then click the "Send" button in the right part of the window, and in the "Settings Communication Path" dialog (Fig. 3.64) confirm the choice of the USB interface by clicking "OK".

After successful completion of the operation the status window shown in Fig. 3.65 will appear.

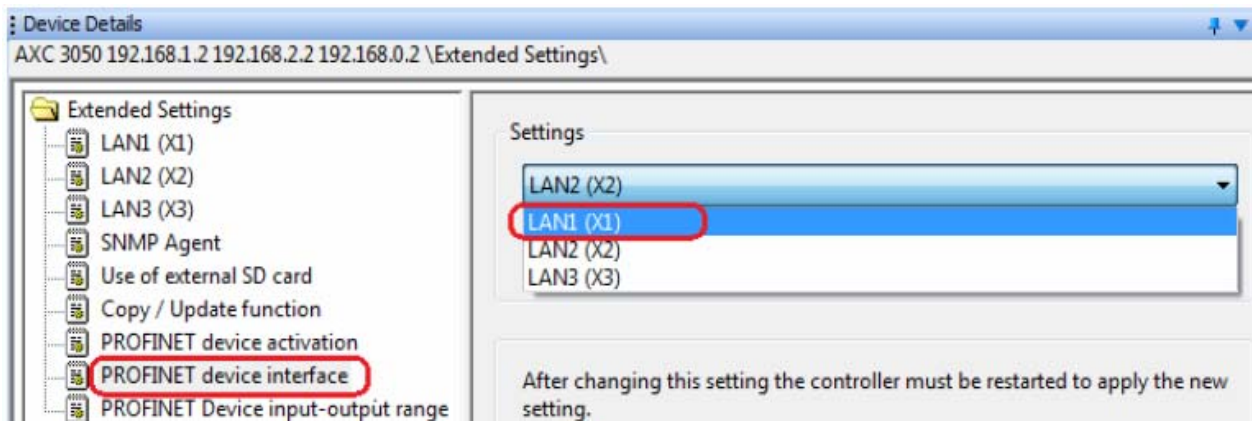


Fig. 3.63. The first step of the procedure for selecting the connection interface between personal computer and controller

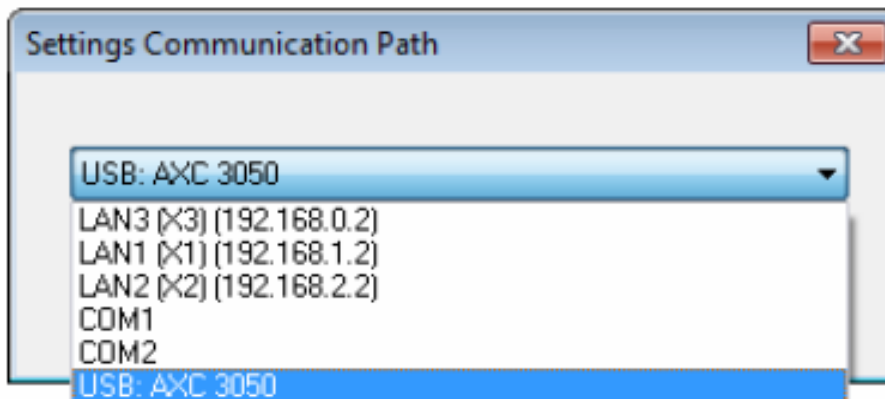


Fig. 3.64. Selecting USB interface for data exchange between personal computer and controller

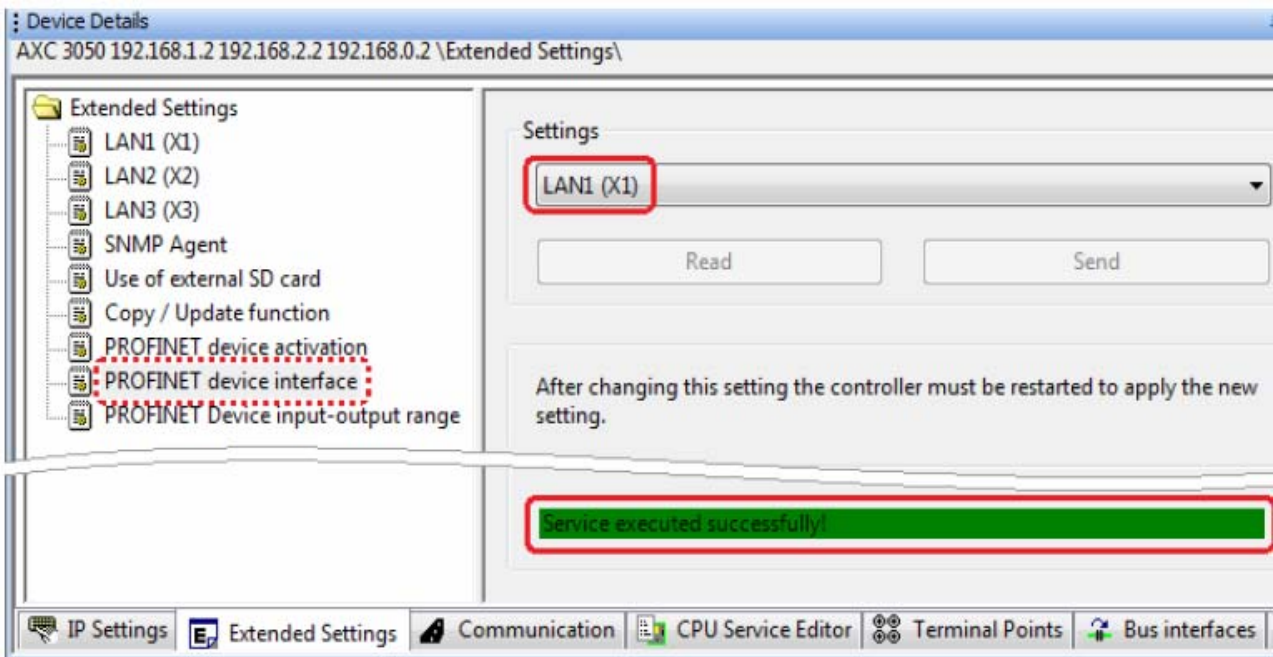


Fig. 3.65. Status window showing the successful completion of the operation

In the left part of the "Device Details" window select the "PROFINET device activation" item in the "Extended Settings" (Fig. 3.66) and in the right part of the "Settings" window – "PROFINET device activated".

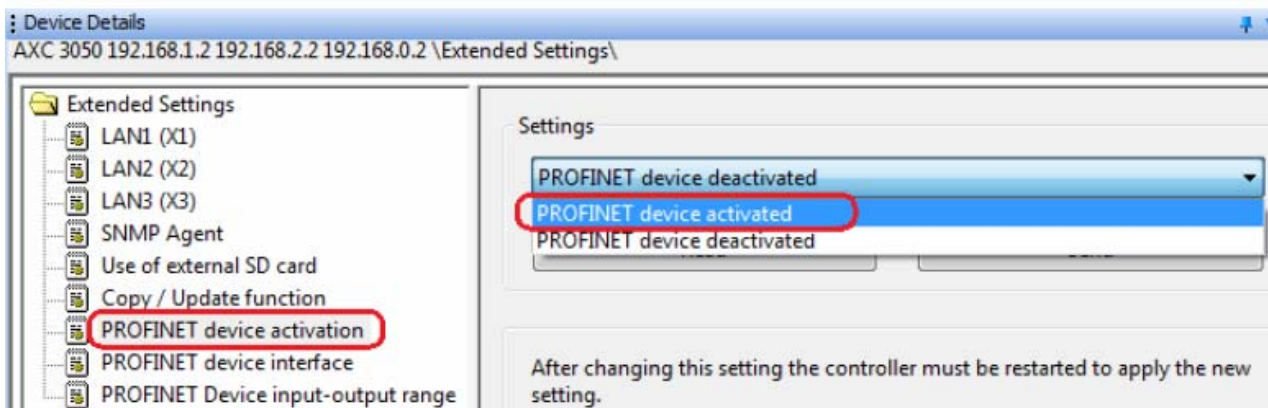


Fig. 3.66. Activating PROFINET device

In order to establish the connection, click the "Send" button on the right side of the "Settings" window. In the "Settings Communication Path" dialog box confirm the USB connection by clicking the "OK" button (see Fig. 3.64). The successful establishment of connection will be shown in the status window (see Fig. 3.65).

In order to activate the settings made, the AXC 3050 controller should be restarted. Click the "Restart Controller" button in the "Activate Network Settings" window and confirm the request to restart the controller with the "Yes" button.

In the "Settings Communication Path" dialog box confirm the selected USB connection or other connection by pressing the "OK" button (see Fig. 3.64). Confirmation of successful connection start is displayed in the status window (see Fig. 3.65).

The BF (Bus Fail) LED flashes during successful operation, indicating the availability of the established communication and the absence of connection to the PROFINET IO controller.

Now it's possible to work with the AXC 3050 controller as with the PROFINET device in the PC Worx project.

In order to read the data from the PROFINET network in PC Worx, the Ethernet network interface card through which it's possible to access PROFINET devices in the network must be selected in the "Extras, PROFINET Configuration" menu. In addition, the IP address must be assigned at least to the X1 interface (as it was described above).

Let us consider the example of adding the AXC 3050 controller to the project as a PROFINET device.

In order to perform the example, the following settings should be used.

For the upper level controller RFC 470 PN 3TX: IP address of LAN 1.1/1.2 – 192.168.1.100; subnet mask 255.255.255.0; name of the PROFINET device rfc-470-pn-1e-9d-e0.

For the AXC 3050 controller operating as a PROFINET device: IP address 192.168.1.2; subnet mask 255.255.255.0; name of the PROFINET device is axc-30501.

In PC Worx the data in the AXC 3050 controller as a PROFINET device can be entered under the following conditions:

in PC Worx the AXC 3050 controller function as a PROFINET device should be activated;

"PROFINET Device" interface of the AXC 3050 controller should be selected;

controller and PROFINET devices should be installed (controller AXC 3050 and other PROFINET devices in accordance with the application/project);

PROFINET controller should be configured in accordance with the application/project.

The sequence of actions is as follows.

In the "Bus Structure" window (Fig. 3.67) in the PROFINET context menu which is called by the right mouse button click select "Read PROFINET".

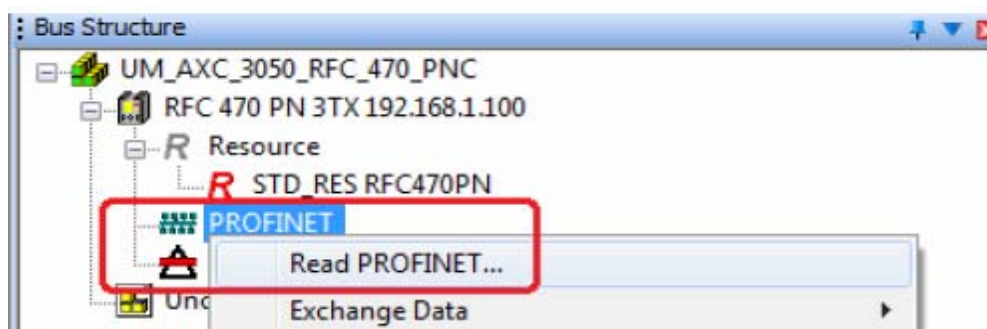


Fig. 3.67. Start the process of reading the data on PROFINET devices

The "Read PROFINET" dialog box displays the PROFINET devices detected in the network (Fig. 3.68).

In the PROFINET device list select the AXC 3050 controller and add it to the project as a PROFINET device by pressing the "Insert" button (see Fig. 3.68).

If the "Select PROFINET device description" dialog box appears, select the device description that corresponds to the AXC 3050 controller. If the PROFINET device to be added to the project has not yet been assigned a name, it is offered to assign it at this stage. If the name has already been assigned to the PROFINET device, repeat the addition to the project of each additional PROFINET device that was installed and detected in the network. After clicking the "Close" button (see Fig. 3.68) the "Read PROFINET" window will be closed and the PROFINET devices added to the project earlier will be displayed in the "Bus Structure" window (Fig. 3.69, left part).

The characteristics of the PROFINET device will be displayed in the "Process Data" tab of the "Device Details" window (Fig. 3.70). The AXC 3050 controller is now available as a PROFINET device in the PC Worx project.

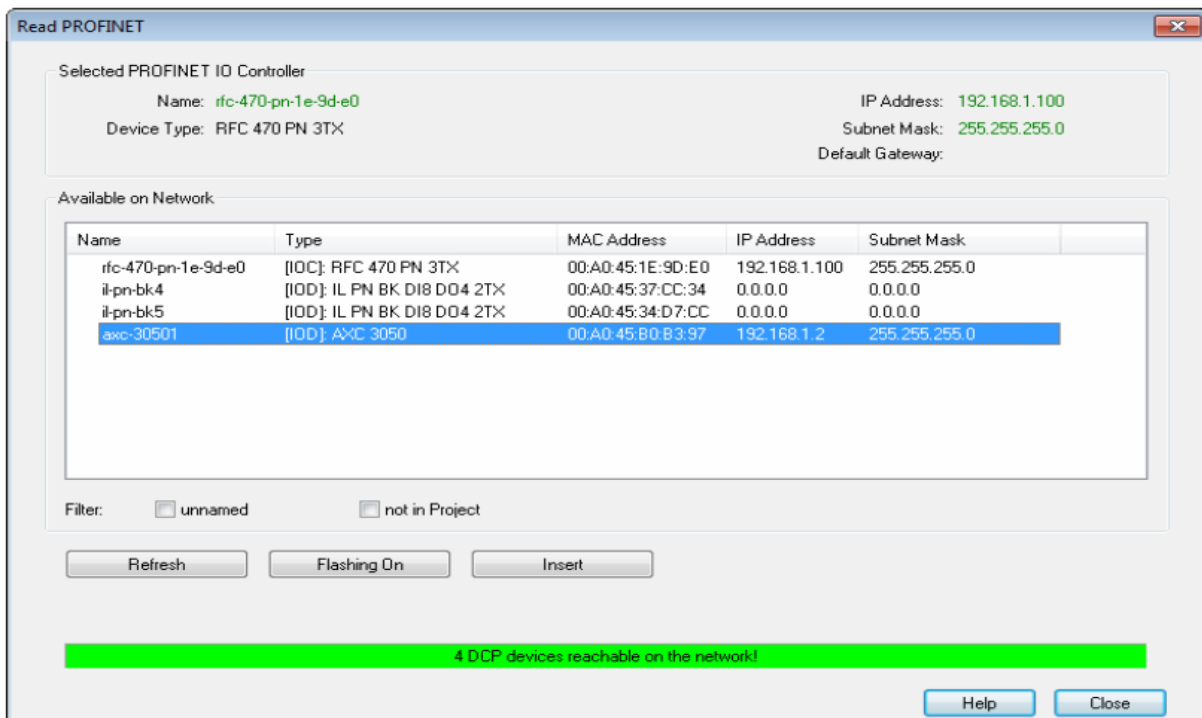


Fig. 3.68. List of PROFINET devices found in the network

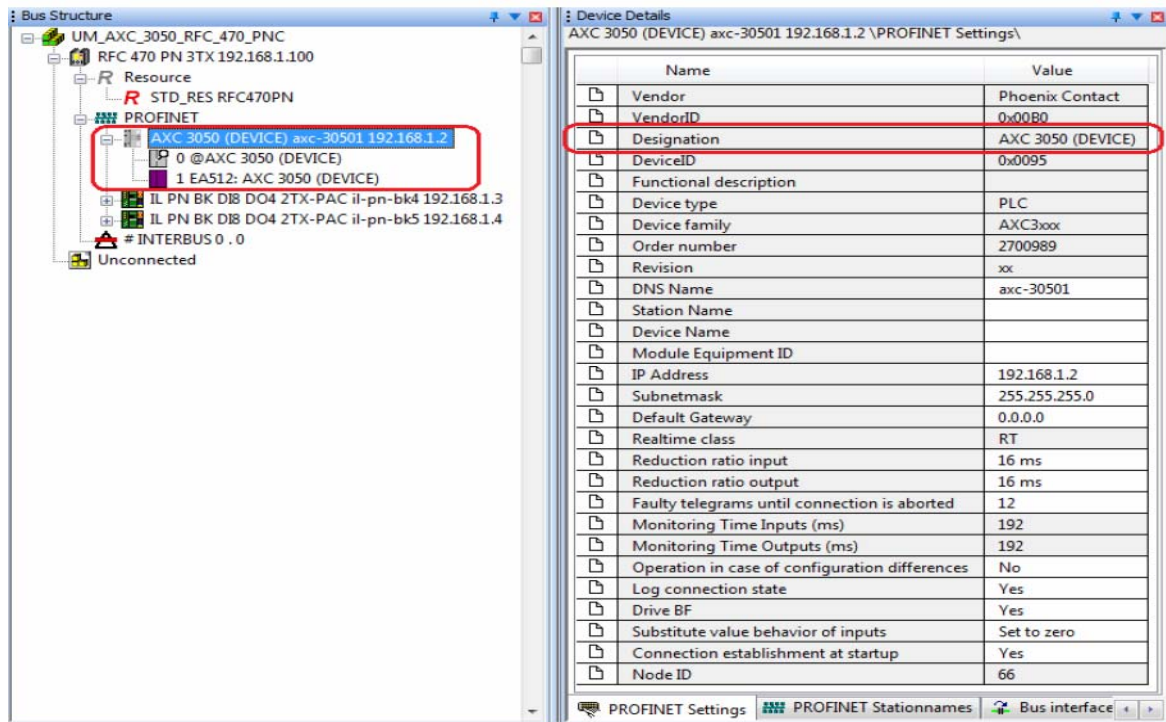


Fig. 3.69. List of PROFINET devices added to the project

Similarly, other devices having different purposes and configurations can be added.

Since the AXC 3050 controller has three Ethernet interfaces, it can be used for building complex branch automation systems based on Ethernet and PROFINET. Using the AXC 3050 controller as a PROFINET controller in a two-level network is shown in Fig. 3.71, where PROFINET BK bus coupler operates in PROFINET device mode with Axioline I/O modules connected.

Device Details								
1 EA512: AXC 3050 (DEVICE) \Process Data\								
	Process Data	Data Type	Byte.Bit	I/Q	Function Text	plug	terminal point	Terminal equipment id
📄	Q512		0.0	I				
📄	I512		0.0	Q				

Module Settings 10101 Process Data Data sheet
 01011
 00111

Fig. 3.70. Characteristics of the PROFINET device

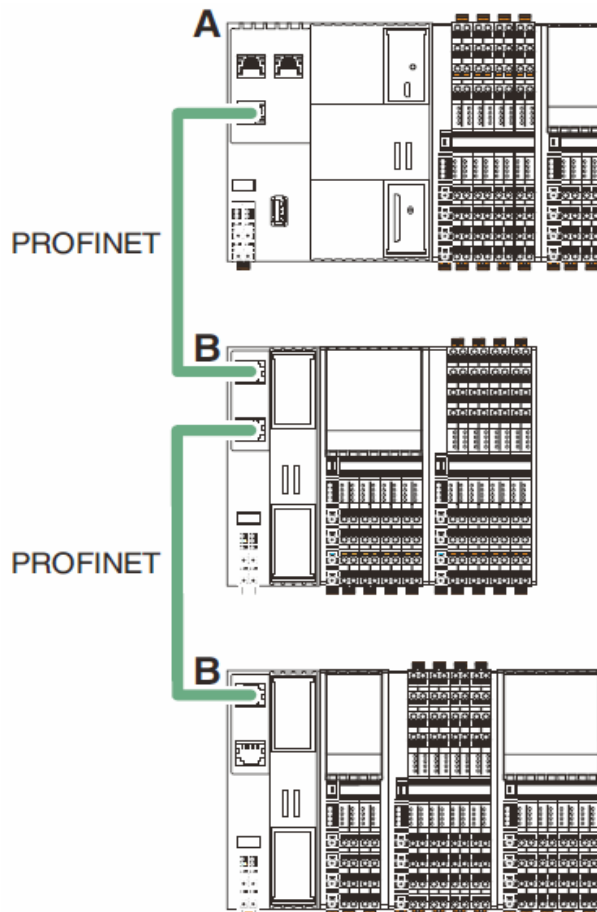


Fig. 3.71. Using the AXC 3050 controller in branch automation systems: **A** – controller operates in the PROFINET controller mode; **B** – PROFINET BK bus coupler works in PROFINET mode

When implementing remote management all devices used in the PC Worx project should have unique IP addresses allocated by Internet service provider and system administrator.

4. PROGRAMMING IN PC WORX INTEGRATED DEVELOPMENT ENVIRONMENT

4.1. FBD (Function Block Diagram)

The FBD language is the graphical programming language of the IEC 61131-3 standard. Library case sets and its own blocks are used for programming, also written in FBD or other languages of the IEC 61131-3 standard. Programming includes connecting ready-made components. The result is the graphical representation of the program.

When choosing a *variable name* the following rules should be observed:

variable names must begin with a letter;

letters of only the Latin alphabet must be used in variable names;

variable names should consist only of letters, digits and the underscore "_";

variable names cannot repeat reserved words;

variable names must be unique within their scope;

variable names are undemanding to the register.

When creating variables it is recommended to make names as informative as possible.

The main *data types* used in PC Worx are listed in Table 4.1. TIME variable can be specified by days (d), hours (h), minutes (m), seconds (s), milliseconds (ms) and their combinations.

When creating variables it is necessary to choose the optimal type of data in accordance with the tasks to be solved, and adhere to the following recommendations:

the range of values should lie within the required limits if the value is constant and will not change. If during the execution of the program the variable changes its values, it is required to provide such a type of data, the range of which will not go beyond the necessary limits;

memory size allocated to each variable must be minimal;

it should be remembered that operations with integer values are performed faster than with real values.

When using a graphical editor, it's possible to define variables in one of the following ways:

insert variables that have been declared in advance;

insert new variables in the worksheet code and declare them using the "Variable Properties" dialog box.

In the FBD language variables are inserted unconnected during the declaration, or already connected to a function or function block. In the first case, the variable appears with two connecting points on the sheet. Then it can be connected to other objects by "dragging" (using "Drag & Drop" function) or by using the connection mode.

Table 4.1

Data types

Data type	Description	Size, bytes	Range of values	Assigned initial value
BOOL	Logical	1	0 or 1	0
SINT	Short integer	8	from -128 to 127	0
INT	Integer	16	from -32 768 to 32 767	0
DINT	Long integer	32	from -2 147 483 648 to 2 147 483 647	0
USINT	Unsigned short integer	8	from 0 to 255	0
UINT	Unsigned integer	16	from 0 to 65 535	0
UDINT	Unsigned long integer	32	from 0 to 4 294 967 295	0
REAL	Real numbers	32	from -3,402823466 E+38 to -1,175494351 E-38 and from +1,175494351 E-38 to +3,402823466 E+38	0.0
LREAL	Long real numbers	64	from ~ -1,798 E+308 to ~ -2,225 E-308 and from ~ +2,225 E-308 to ~ +1,798 E+308	0.0
TIME	Time variable	32	from 0 ... to 4 294 967 295 ms	0
BYTE	8-bit hexadecimal number	8	from 0 to 255 (from 16#00 to 16#FF)	0
WORD	16-bit hexadecimal number	16	from 0 to 65 535 (from 16#00 to 16#FFFF)	0
DWORD	32-bit hexadecimal number	32	from 0 to 4 294 967 295 (from 16#00 to 16#FFFFFFFF)	0

In the FBD language variables are inserted unconnected during the declaration, or already connected to a function or function block. In the first case, the variable appears with two connecting points on the sheet. Then it can be connected to other objects by "dragging" (using "Drag & Drop" function) or by using the connection mode.

In order to create a variable click the left mouse button on the worksheet. In this case, the section will appear in the upper Object menu, where you should select the Variable item (Fig. 4.1).

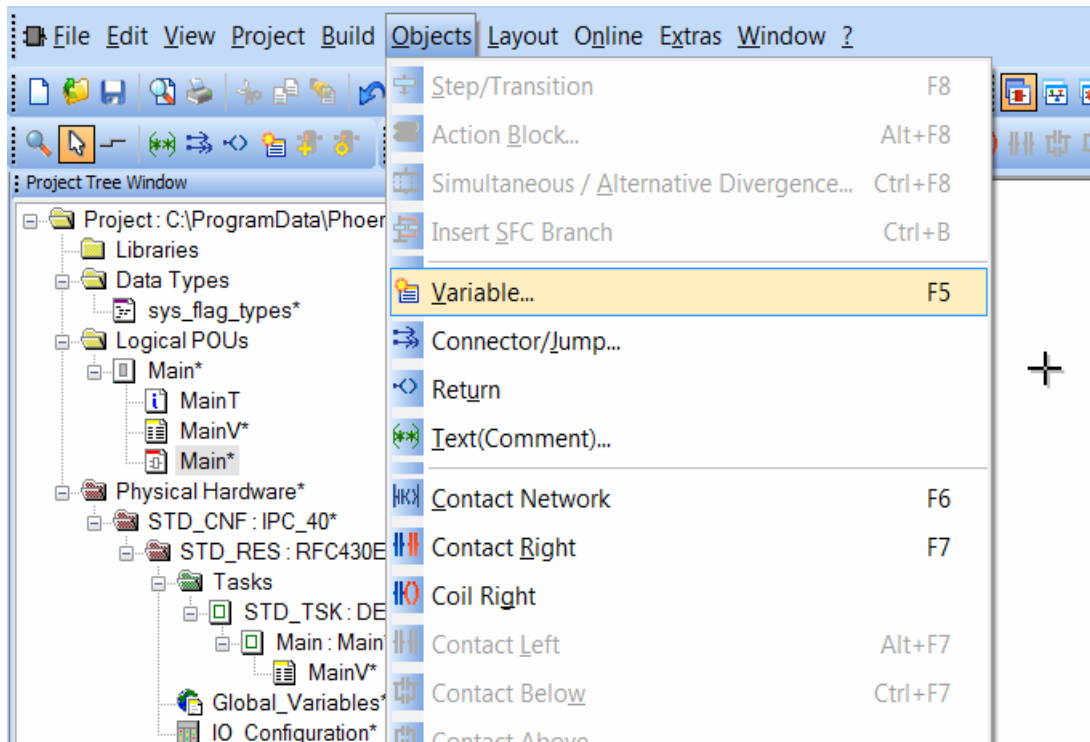


Fig. 4.1. Creating a new variable

After that, the window with the properties of the created variable will appear on the screen (Fig. 4.2).

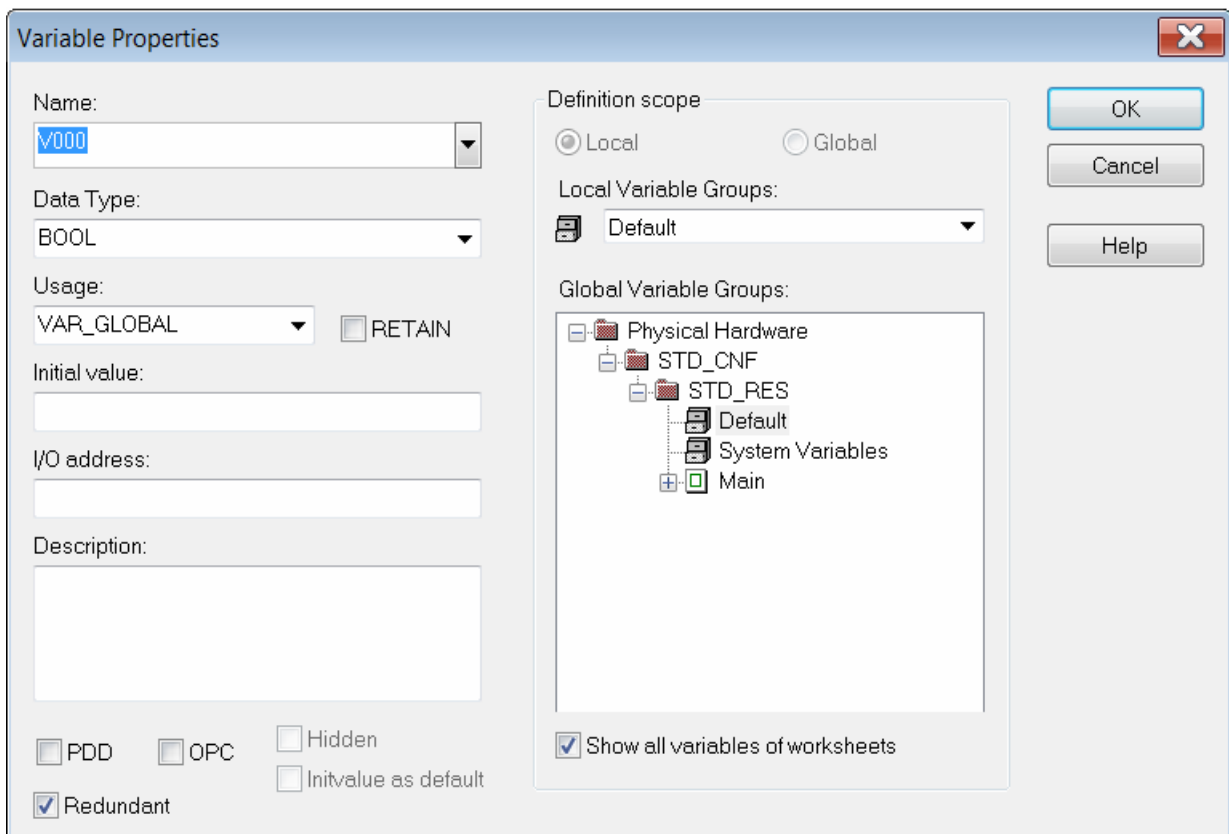


Fig. 4.2. Variable properties window

The purpose of the fields in this window is as follows.

Name is the name of the variable. It is possible to introduce a new variable name or select a variable already existing in the system – select the variable from the list in the drop-down menu to the right of the name.

Data Type is the data type of the variable. Type can be entered from the keyboard or selected from the drop-down list on the right.

Usage allows selecting the type of variable usage: locally (only within one POU section; VAR, VAR_INPUT or VAR_OUTPUT is declared) or globally (in the entire project area; VAR_GLOBAL or VAR_EXTERNAL is declared in each POU where it is used).

Initial value – allows entering the initial value of the variable, i.e. the value that will be assigned to the variable upon initial initialization.

I/O address is the address assigned to the variable.

Description is the description of the variable that is displayed as a tooltip when you hover the mouse cursor over a variable (Fig. 4.3).

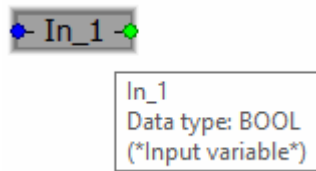


Fig. 4.3. Using description function to describe the variable

Using the Definition scope section of the variable properties window, it's possible to select variables from different sections of the system.

As an example, create the program consisting of the input and output variables (Fig. 4.4).

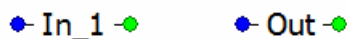


Fig. 4.4. Creating two variables

Each variable has two connections – on the right and on the left. The In_1 variable will simulate a switch or a sensor. It will send a signal to the circuit, so the connection of the variable will be on the right. The Out variable will simulate the actuator. The program will control this variable, so the variable connection will be on the left. Connect them as shown in Fig. 4.5 by pressing the left mouse button.



Fig. 4.5. Connected variables

The program has been created. In order to compile the program select the Make item in the top Build menu (Fig. 4.6) or press the F9 hotkey.

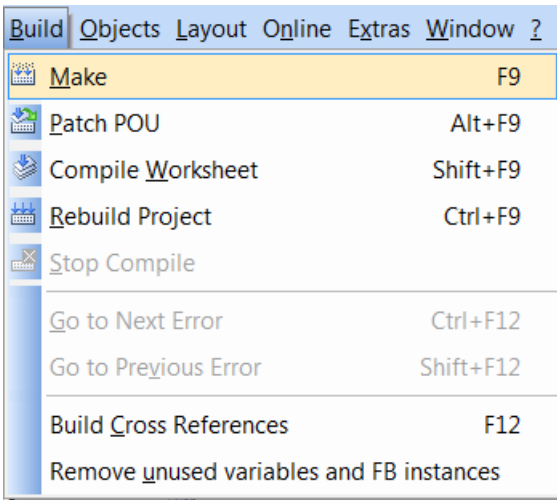


Fig. 4.6. Selecting the program compilation path

After that in the information window of the Message Window it's possible to observe the process of compiling the program (Fig. 4.7). At the end of the process it's possible to see the number of Errors and Warnings.

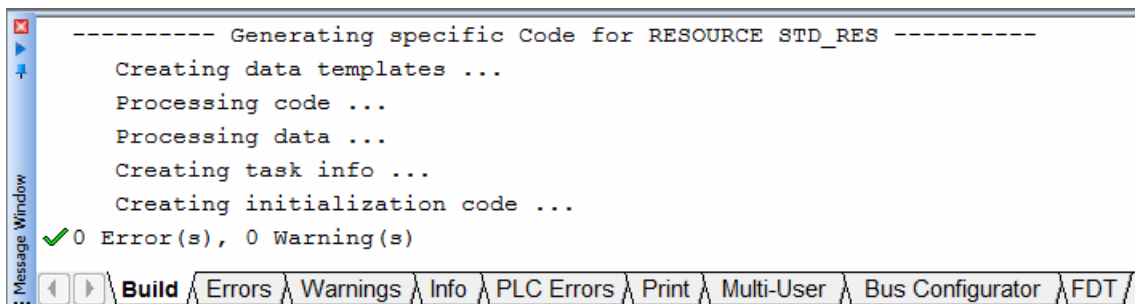


Fig. 4.7. Message Window

If the number of Errors in the program is different from zero, such a program cannot be loaded into the controller. It is necessary to fix all the errors and compile the program again, having achieved zero errors.

A certain number of Warnings is allowed. In this case the program can be loaded into the controller, but the performance will be reduced.

The next step is to load the program into the controller. Choose Project Control in the top Online menu. The STD_RES window will appear on the screen (Fig. 4.8).



Fig. 4.8. STD_RES window

In the STD_RES window click the Download button. The download window will appear. In the Project section set the Include Bootproject check box, as shown in Fig. 4.9, and click the Download button.

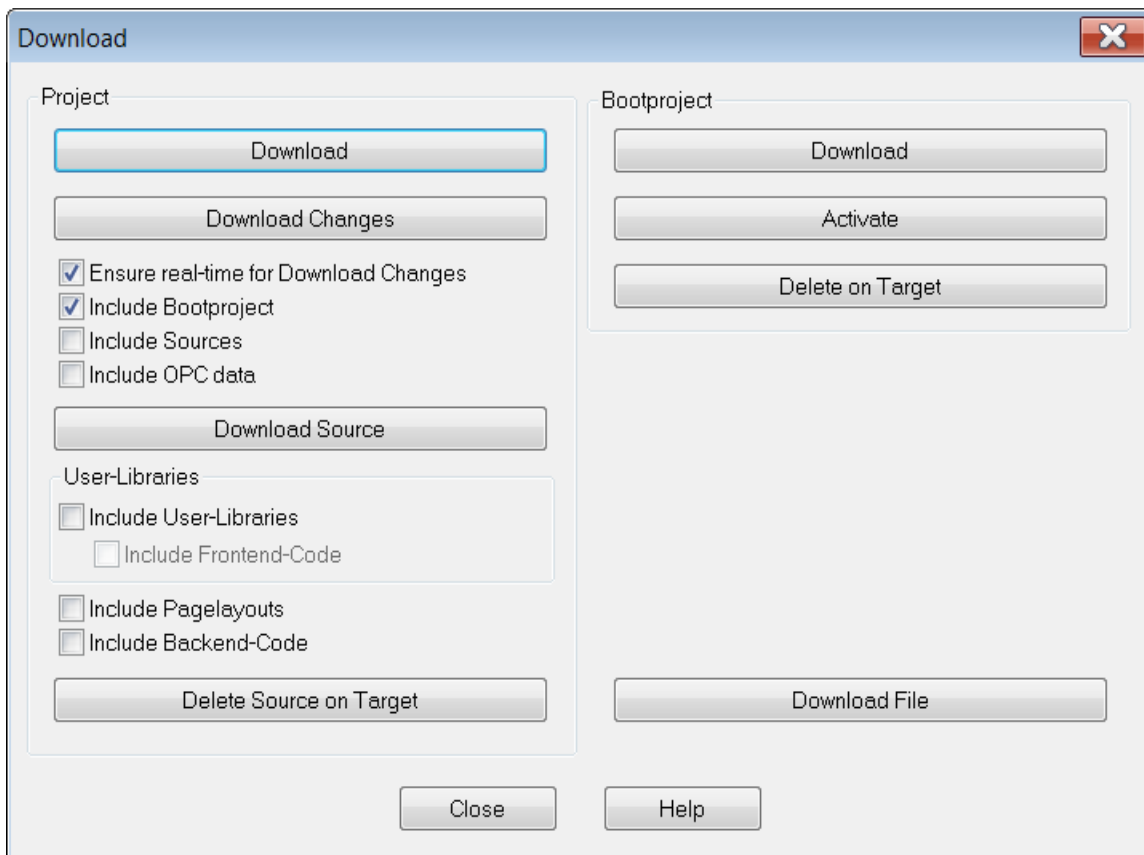


Fig. 4.9. Download window

At the bottom of the program it's possible to see the progress of the project loading into the controller (Fig. 4.10).



Fig. 4.10. Downloading of the program

The blue and green colors of the LED indicate a successful download. The red color indicates an error.

After downloading the program it should be started. Press the Cold button in the STD_RES window (see Fig. 4.8) and then select the Debug item in the top Online menu. The running program is shown in Fig. 4.11.

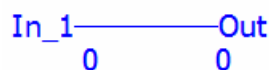


Fig. 4.11. Running program

The variables are colored blue and numeric values are displayed below them. In this case the signal of the In_1 sensor is zero and the actuator is turned off. To change the state of the sensor, double click on the input variable In_1. A window appears showing the state of the variable (Fig. 4.12).

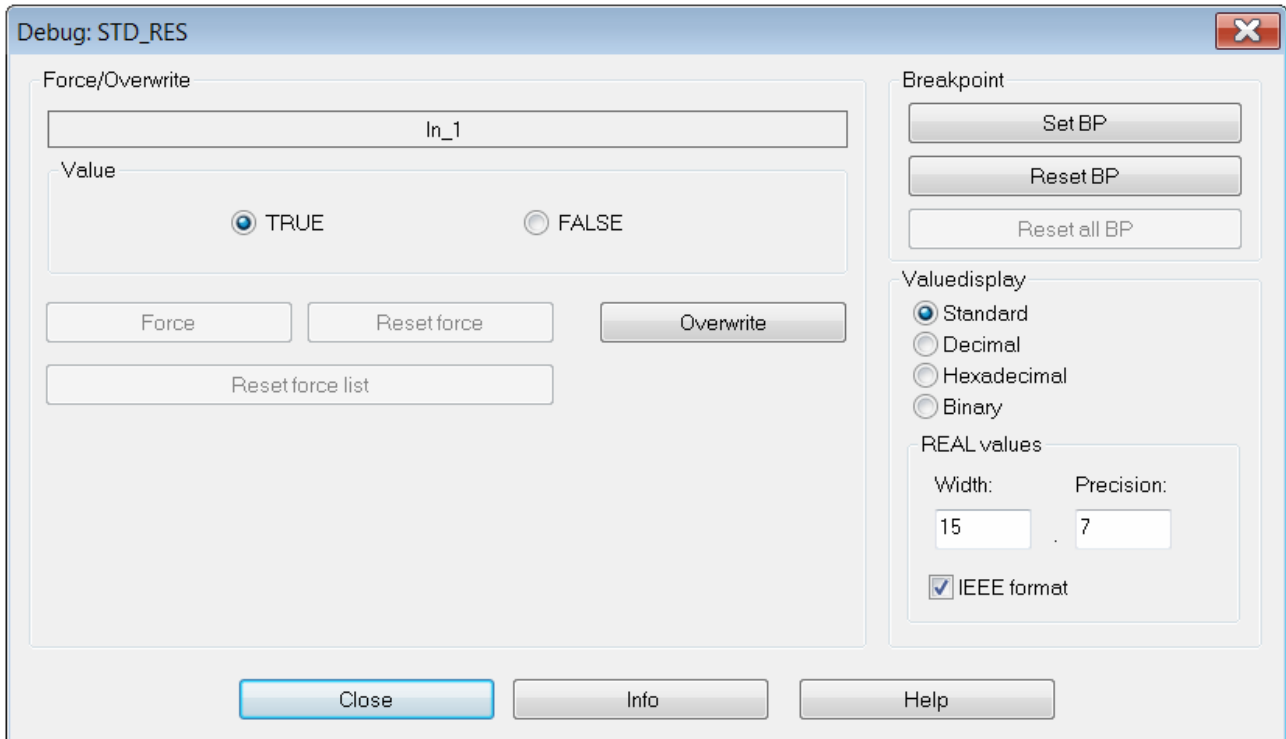


Fig. 4.12. Variable state window

To change the state of this variable, press the Overwrite button. The In_1 variable will change its state. If the type of the variable is Bool, the current state, which is equal to zero, will change to the opposite one, and the program will take the form shown in Fig. 4.13.

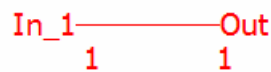


Fig. 4.13. Changing the state of the input variable

It's possible to see from Fig. 4.13 that the change of the state of the input variable In_1 affected the output variable Out and the control device turned on.

As an example, consider an electrical diagram – the analog of the work of this program (Fig. 4.14). Here the In_1 variable is represented as a switch with normally open contacts. The Out variable is represented by the load. In the initial enclosed, current starts to flow in the circuit and the load turns on.

To stop the program, press the Stop button in the STD_RES window, and then select the Debug item in the top Online menu.

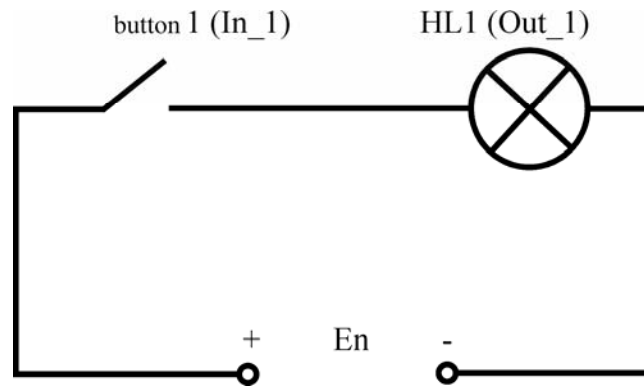


Fig. 4.14. Electrical diagram – the analogue of the completed program

Logical operations of the FBD language can be found in the Edit Wizard window (Fig. 4.15).

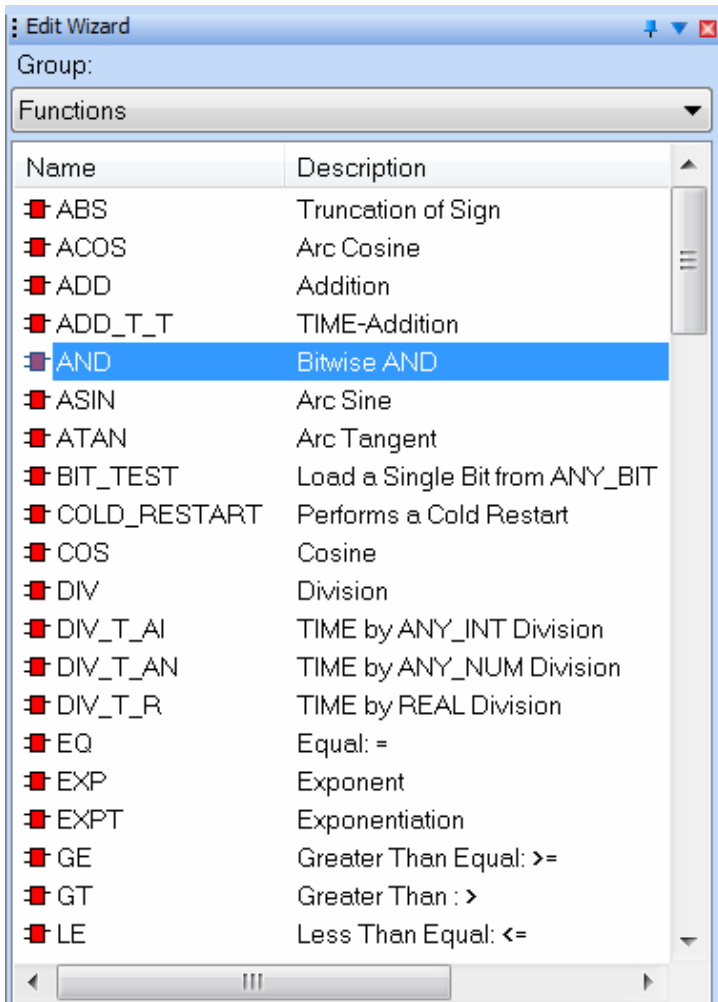



Fig. 4.15. Edit Wizard window

This window is located on the right side of the screen. If otherwise, it's possible to turn it on by selecting the Edit Wizard item from the top View menu or by selecting the icon  on the control panel. At the top of this window it's possible to use the drop-down menu (Fig. 4.16), which allows selecting the functions grouped by sections or full list.

- <all FUs and FBs>
- <Favorites>
- <UNTITLED>
- Function blocks
- Functions
- Network Templates
- String FUs
- Type conv. FUs

Fig. 4.16. Function groups of the Edit Wizard window

Examine the basic logic functions.

Select the AND function, as shown in Fig. 4.15, and transfer it to the working area. The function will be obtained as shown in Fig. 4.17.

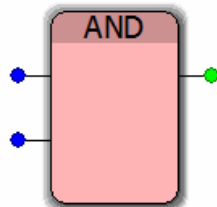


Fig. 4.17. The AND function

In the basic version, this function has two inputs on the left and one output on the right. Create two input variables and one output variable (Fig. 4.18).

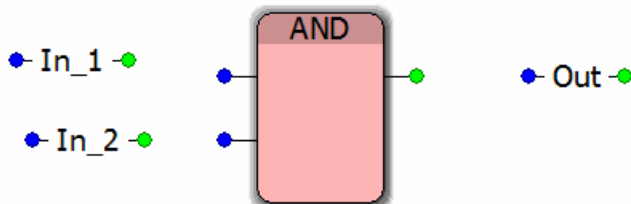


Fig. 4.18. The AND function with unconnected variables

Connect the variables to the function and compile, load and run the program (Fig. 4.19).

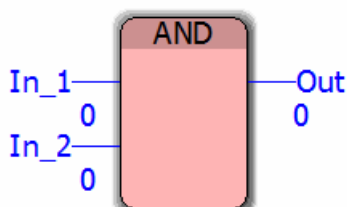


Fig. 4.19. Operation of the AND function

After starting the program fill out the Table 4.2 by changing the values of the In_1 and In_2 input variables. Display the numeric values in the Out column corresponding to the combination of the input parameters.

Table 4.2

Truth table for the AND function

In_1	In_2	Out
0	0	0
0	1	0
1	0	0
1	1	1

Next take the OR function and repeat the previous steps – create the program (Fig. 4.20), run it and fill in the Table 4.3.

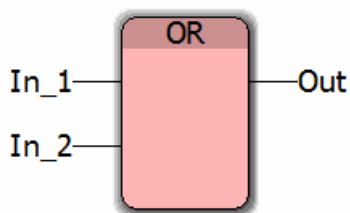


Fig. 4.20. Program using the OR function

Table 4.3

Truth table for the OR function

In_1	In_2	Out
0	0	0
0	1	1
1	0	1
1	1	1

Then create the program using the NOT function (Fig. 4.21) and fill in the corresponding truth table (Table 4.4).

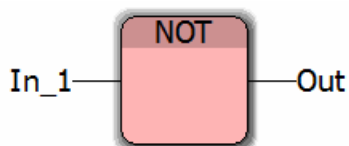


Fig. 4.21. Program using the NOT function

Table 4.4

Truth table for the NOT function

In_1	Out
0	1
1	0

The next step is to create the program of the "NAND" operator by combining the two functions. In order to do this, send a signal from the output of the "AND" function to the input of the "NOT" function, as shown in Fig. 4.22. Fill in the Table 4.5.

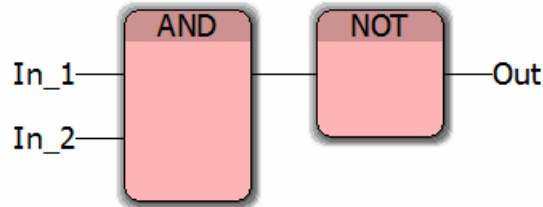


Fig. 4.22. Combination of two blocks

Table 4.5

Truth table for the NAND function

In_1	In_2	Out
0	0	1
0	1	1
1	0	1
1	1	0

In the FBD language it is not necessary to use the individual NOT block for output inversion. Let us create the program according to Fig. 4.23.

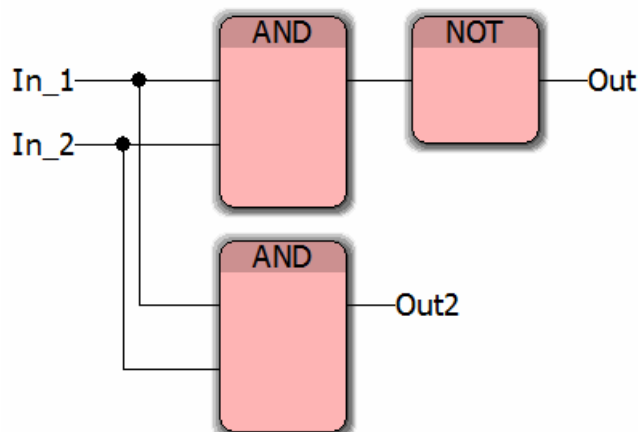


Fig. 4.23. Modified program

Open the properties of the bottom AND function by double clicking on it. At the bottom in the Format Parameters section the table will be displayed. Mark the Out output parameter in the Negated column in this table, as shown in Fig. 4.24.

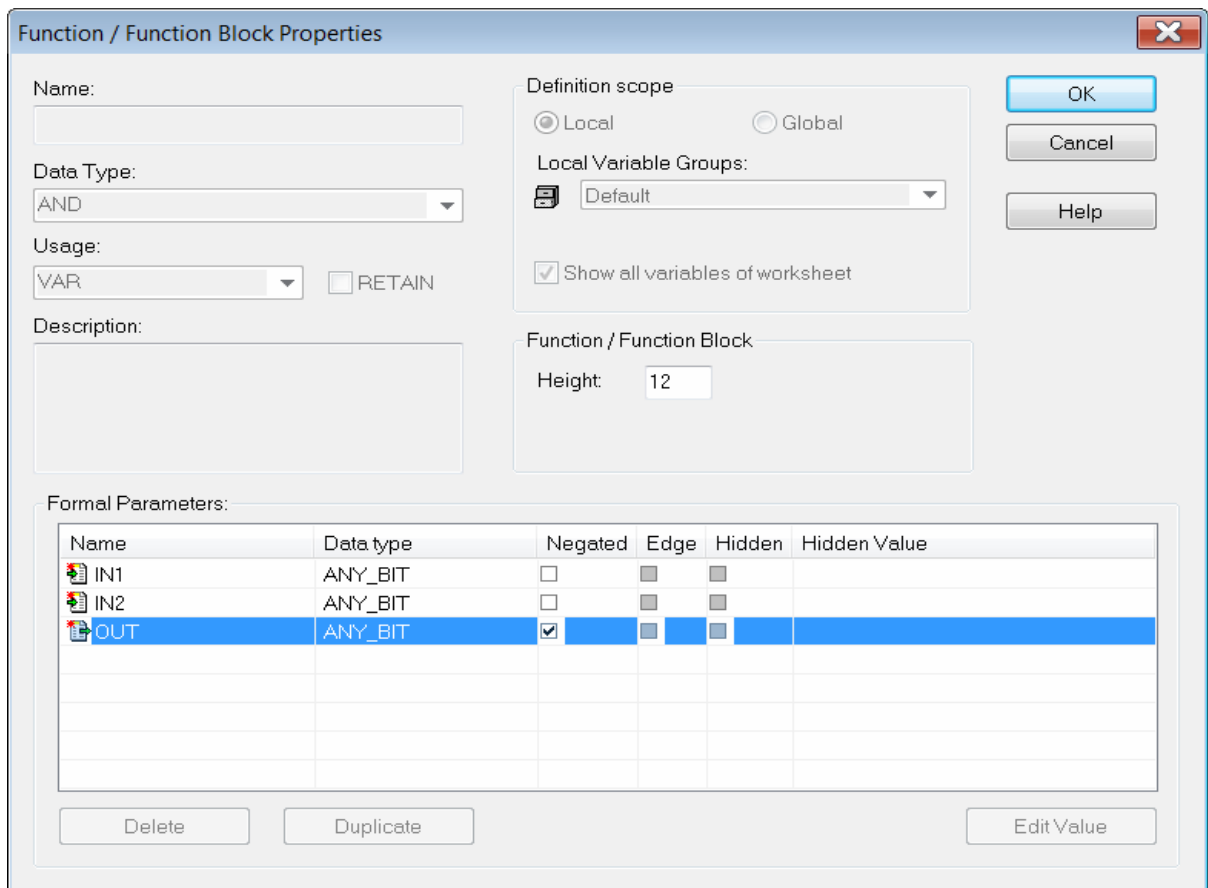


Fig. 4.24. Properties of the AND function

Press the OK button and the program will appear as shown in Fig. 4.25. The lower AND function at the Out2 output will display the inversion icon. When you start the program, both functions will work the same.

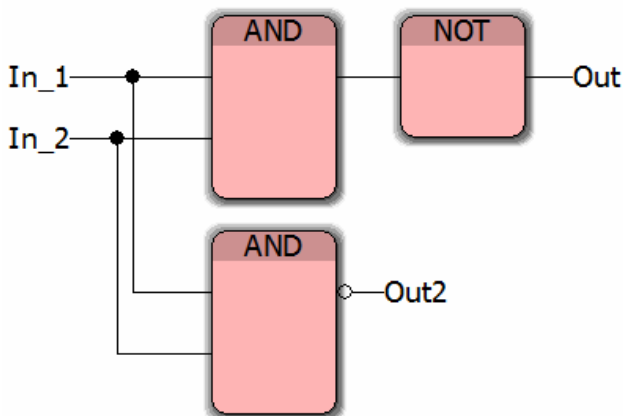


Fig. 4.25. Using inversion

In addition to inversion of the output parameter, it's possible to invert the input in the same way. Operators on the diagram shown in Fig. 4.26 will work in the same way.

Also, if necessary, it is possible to add the required number of inputs (Fig. 4.27). In order to do this, select the input IN2 in the function properties of the Format Parameters section and press the Duplicate button the required number of times, as shown in Fig. 4.28.

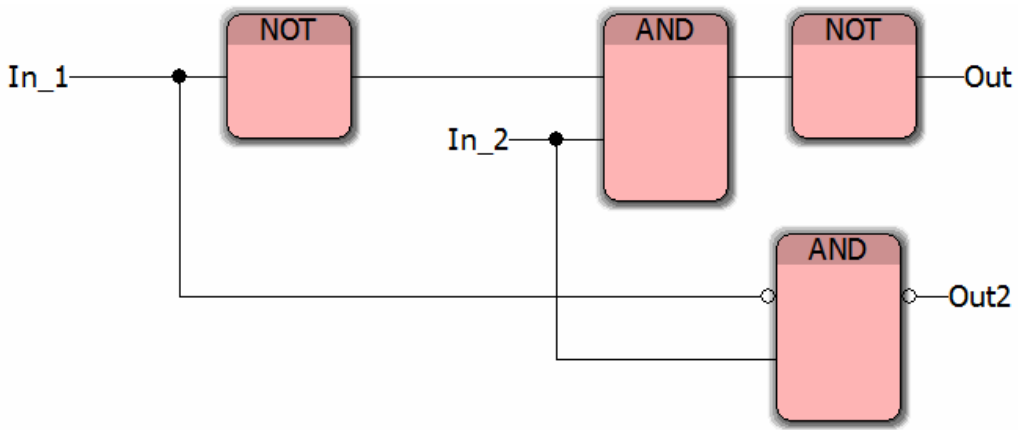


Fig. 4.26. Using inversion at the input and output

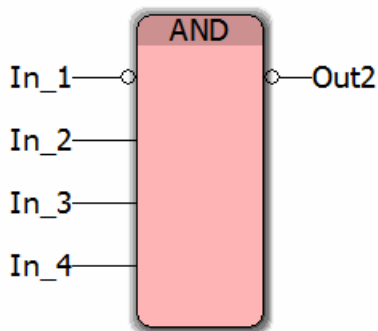


Fig. 4.27. Using the four-way AND function

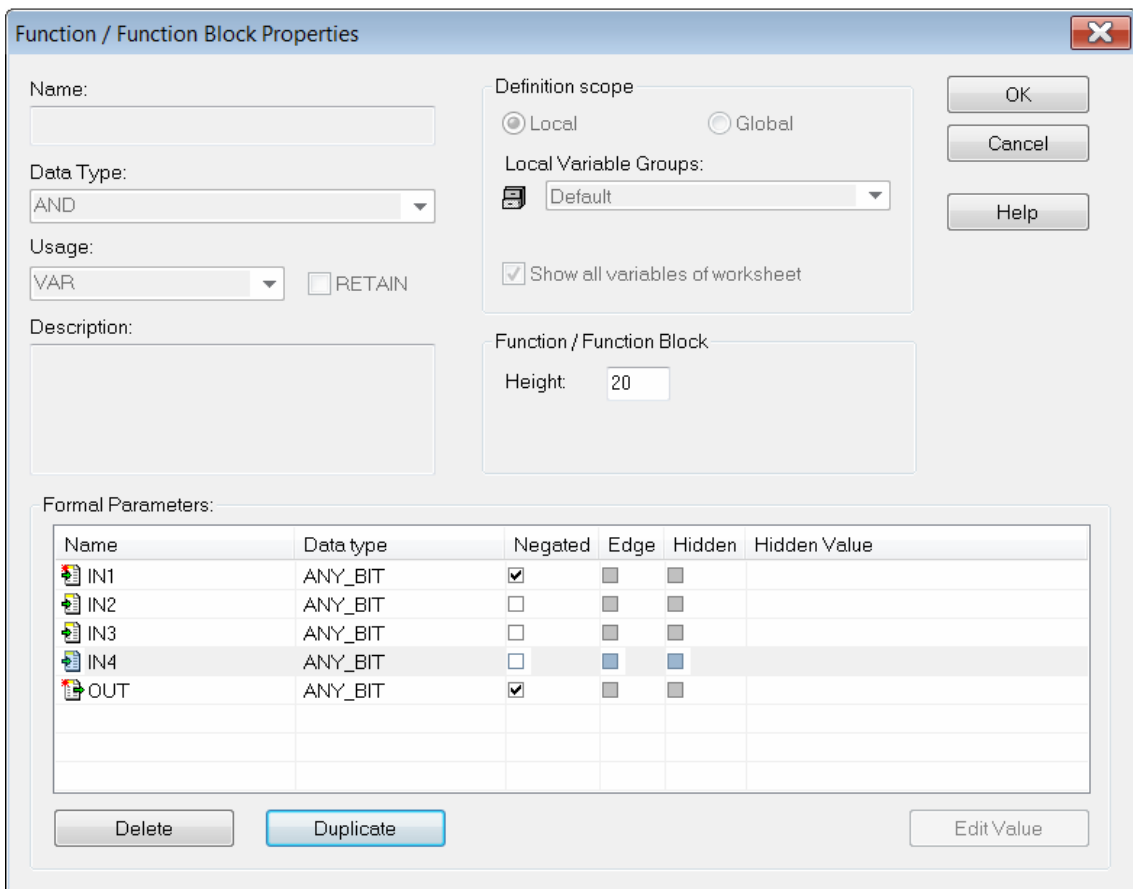


Fig. 4.28. The AND function with four inputs

The order of execution (priority) of logical operations in a complex logical expression is as follows.

1. Logical negation (inversion).
2. Logical multiplication (conjunction).
3. Logical addition (disjunction).
4. Logical consequence (implication).
5. Logical equivalence.

Parentheses are used for changing the order of operations. Also, they are recommended for using in complex expressions to reduce the likelihood of subjective programming errors.

Assuming that the first and third hardware modules are connected as specified in 1.5, consider the correspondence of control elements of the third hardware module to PC Worx variables: switch to the IEC Programming mode and go to the global variables section in the Project tree Window (Fig. 4.29). The input global variables are highlighted in red.

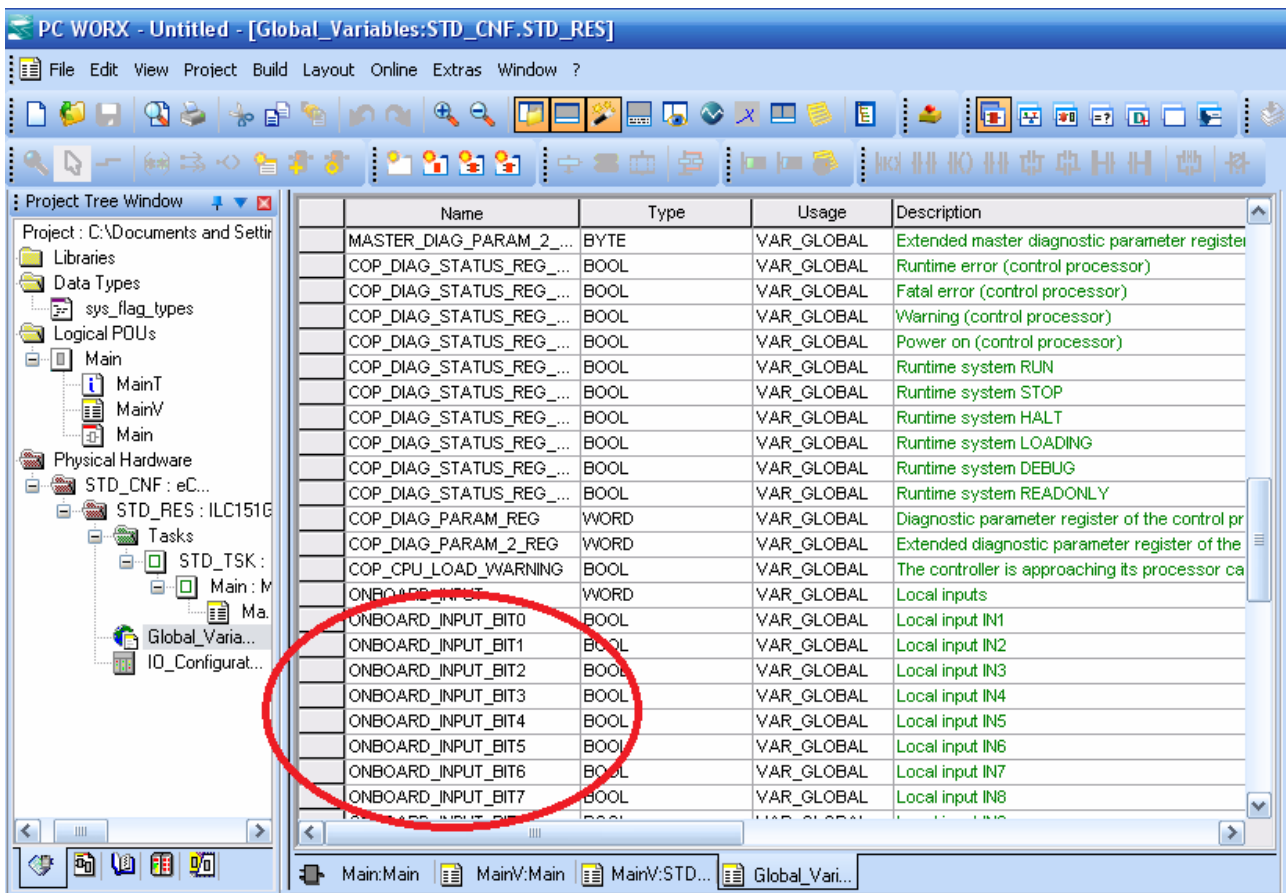


Fig. 4.29. Input global variables

These variables correspond to the block of eight switches selected in Fig. 4.30.

In this section there are also output global variables (Fig. 4.31), which correspond to the LEDs marked in red in Fig. 4.32.

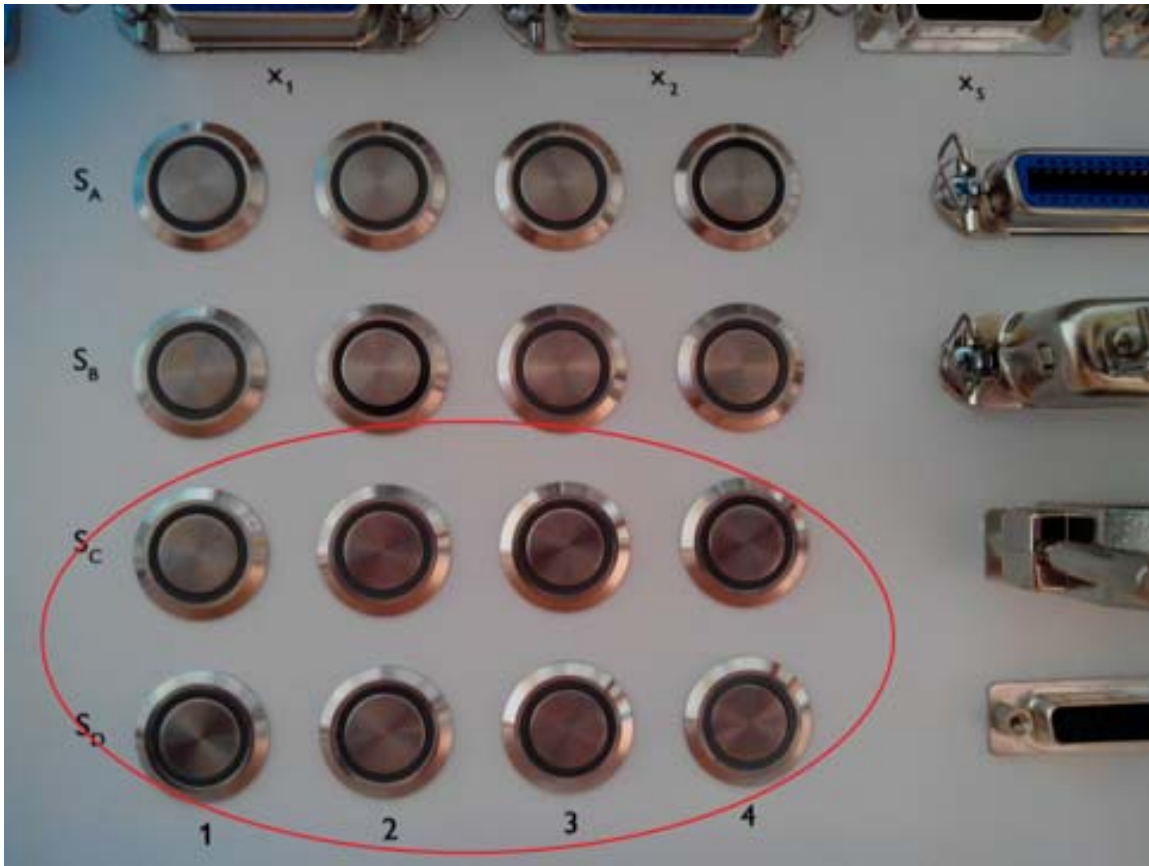


Fig. 4.30. Block of switches of input global variables in the third hardware module

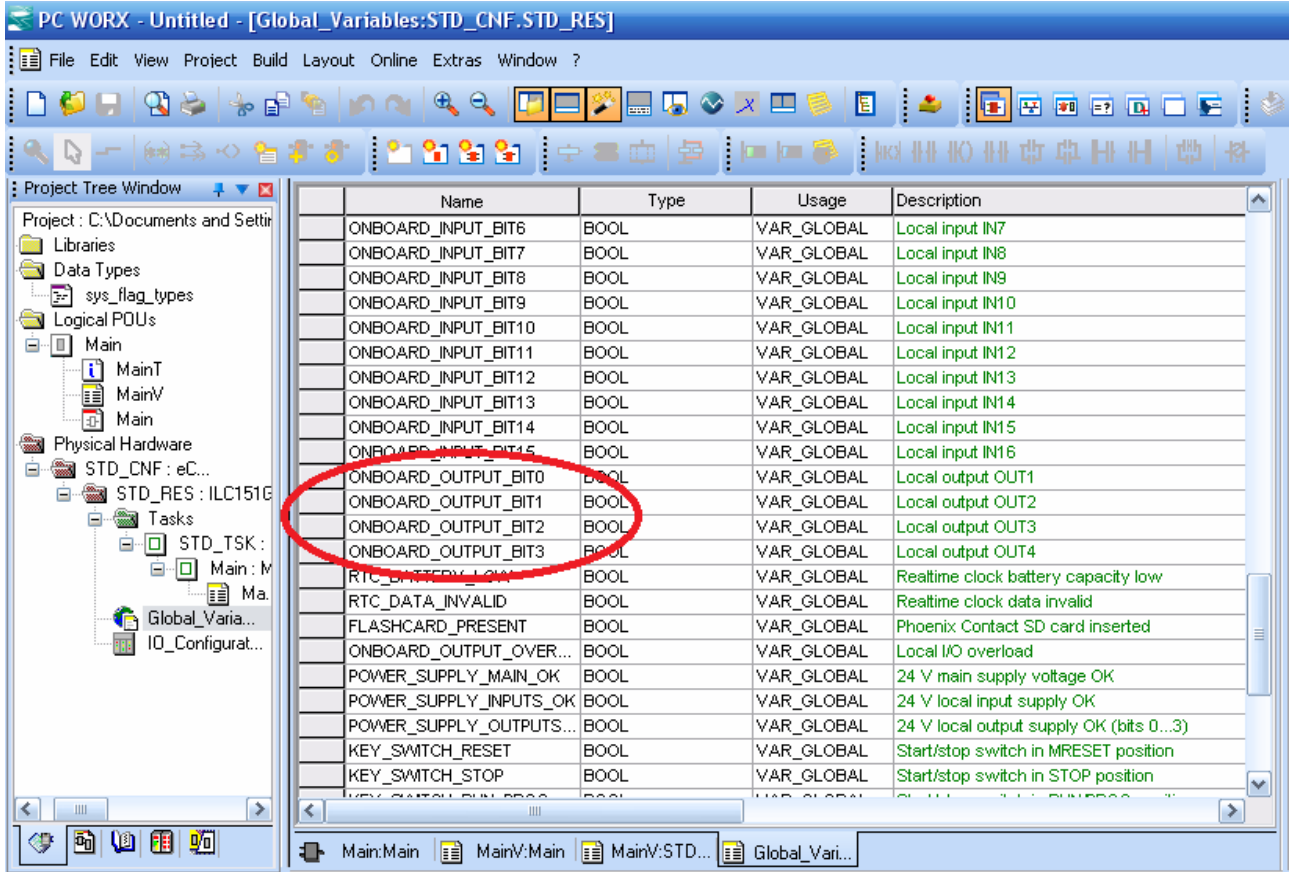


Fig. 4.31. Output global variables

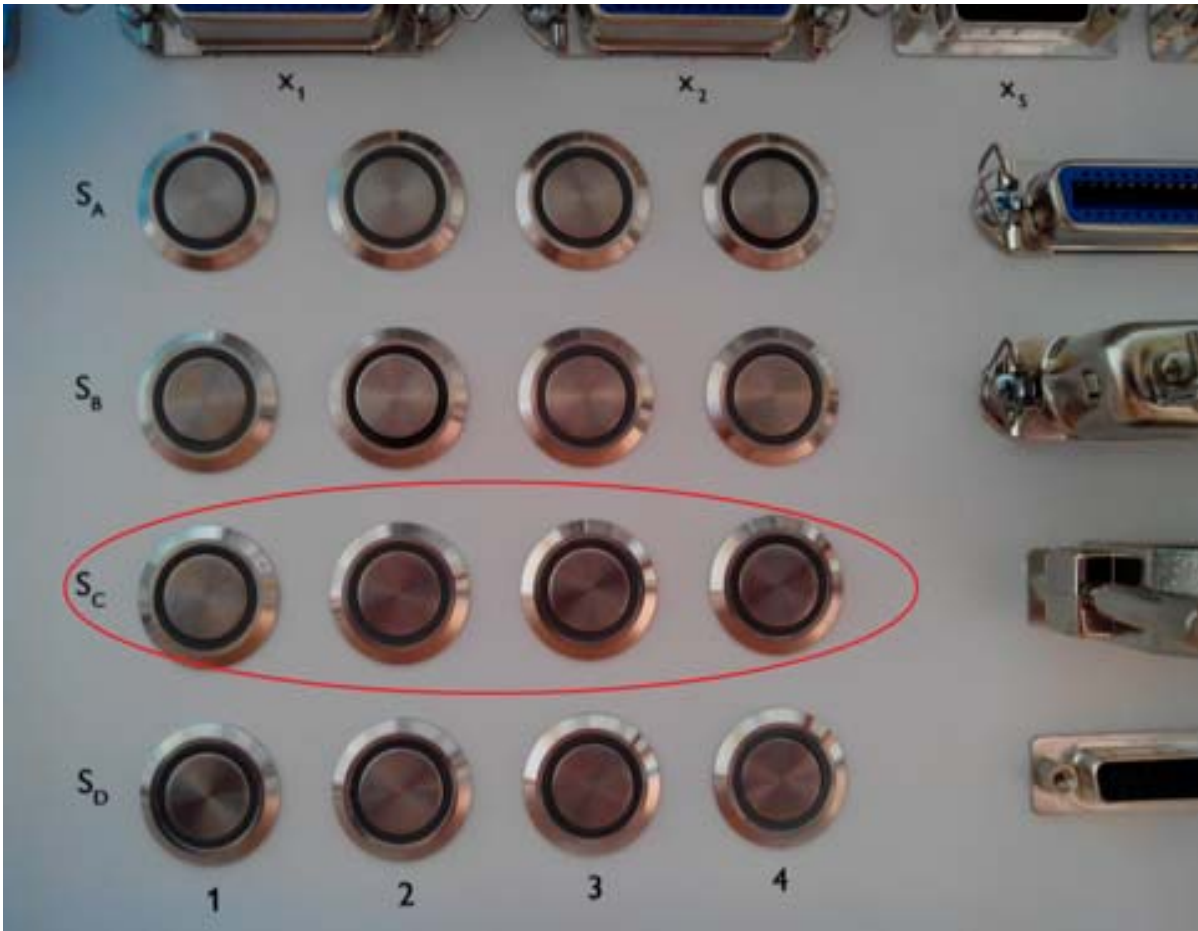


Fig. 4.32. Block of LEDs of output global variables in the third hardware module

Create the program consisting of input and output variables (Fig. 4.33, *a*).

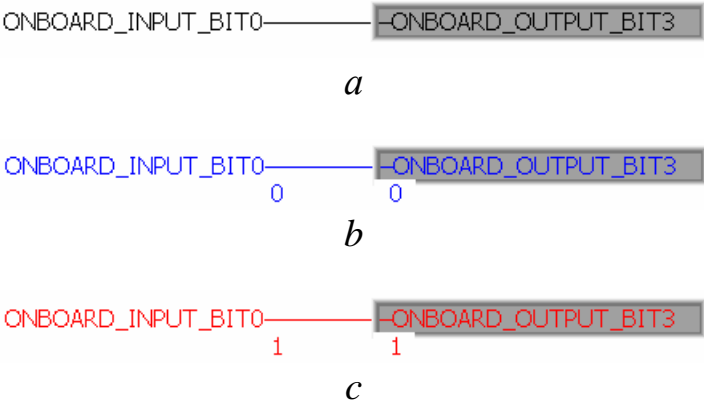


Fig. 4.33. Program for working with bit variables: *a* – source program; *b* – input element in the disabled state; *c* – input element in enabled state

After startup the program will read the state of the input switch and transmit the information to the output LED (Fig. 4.33, *b*). When the switch is pressed the state of the input variable will change, which will be displayed on the output LED, as shown in Fig. 4.34.



Fig. 4.34. The result of the program (LED turned on)

In order to use analog controls in the program, first connect them in the PC Worx shell. Switch to the Process Data mode and select the connected block responsible for input/output of analog information IB IL AI 4/U-PAC 0.3. In the lower part the variables included in this block will be represented. By selecting the necessary variables with the mouse and dragging them, they are set in the system (Fig. 4.35). The variables are initialized by selecting the Build/Make item.

Symbol/Variable	Data Type	Process Data Item	Description	Device	Process Data Item	I/Q	Data Type	Byte.Bit	Address	Symbol
I_0_3_AI_1_VOLTAGE	WORD	# 3 IB IL AI 4/U-P...		# 2 I...	Channel 1: Parameters	I	WORD	0.0		
I_0_2_CHANNEL_1_OUTP...	WORD	# 2 IB IL AO 2/UI-...		# 2 I...	Channel 2: Parameters	I	WORD	2.0		
Q_0_2_CHANNEL_1_OUT...	WORD	# 2 IB IL AO 2/UI-...		# 2 I...	Channel 1: Output value	I	WORD	4.0		
Q_0_2_CHANNEL_2_OUT...	WORD	# 2 IB IL AO 2/UI-...		# 2 I...	Channel 2: Output value	I	WORD	6.0		
				# 2 I...	Channel 1: Parameters	Q	WORD	0.0		
				# 2 I...	Channel 2: Parameters	Q	WORD	2.0		
				# 2 I...	Channel 1: Output value	Q	WORD	4.0		STD_C...
				# 2 I...	Channel 2: Output value	Q	WORD	6.0		STD_C...
				# 2 I...	~AI 64	I	LWORD	0.0		
				# 2 I...	~AO 64	Q	LWORD	0.0		

Fig. 4.35. Connecting analog variables to the project

Create the program where the connected analog variables will be used (Fig. 4.36).

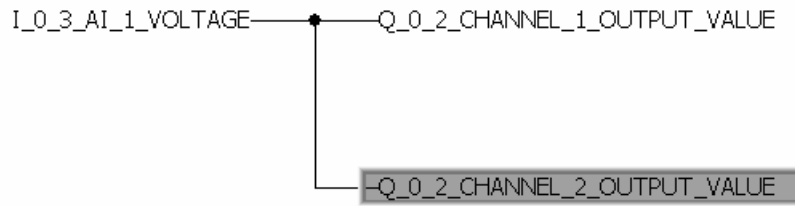


Fig. 4.36. Analog information display program

After the program starts the numerical values of the variables (Fig. 4.37), which are read from the controller and transferred to the indicators will appear (Fig. 4.38).

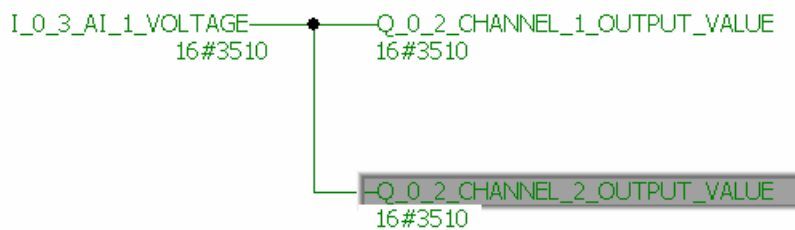


Fig. 4.37. Example of the running program

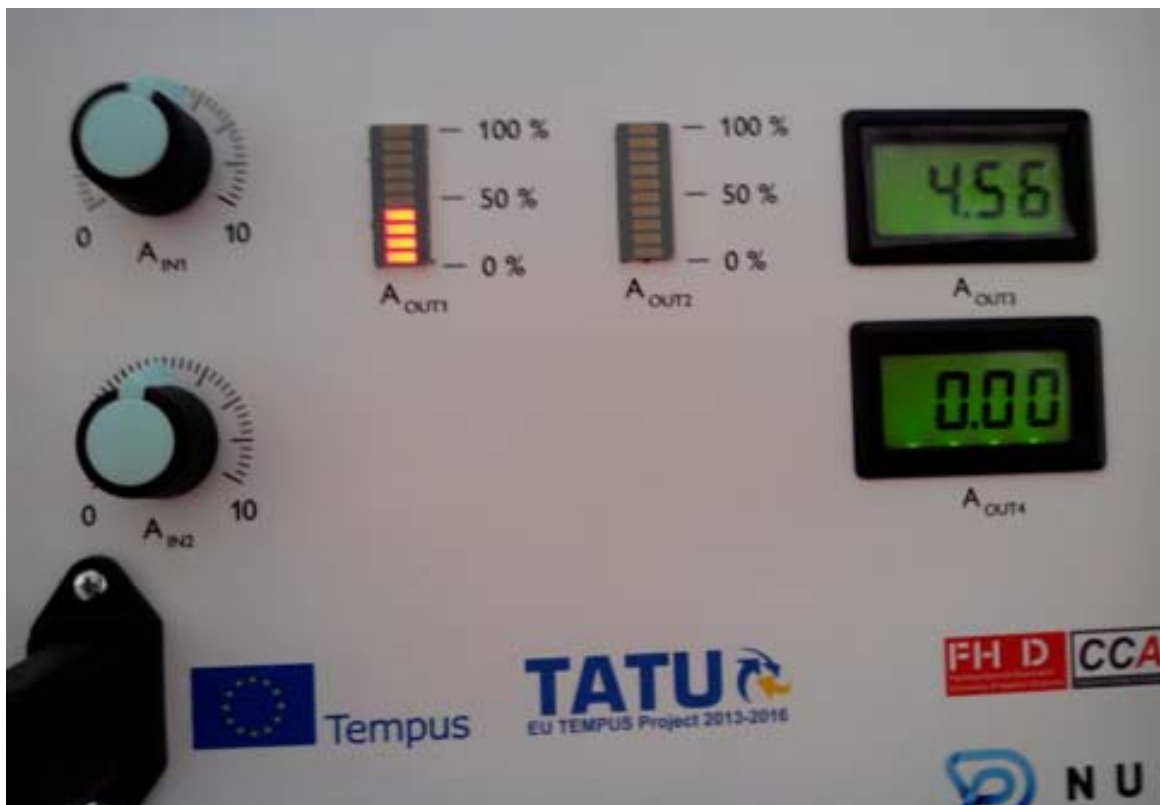


Fig. 4.38. The signal transmitted from the potentiometer to the indicators

If for example you want to connect a proportional–integral–derivative (PID) controller to the project, you should use a standard library containing the function block that has the PID name.

In order to add it to the project, select the line "Function blocks" in the Edit Wizard window, which by default is in the upper right corner of the screen (Fig. 4.39).

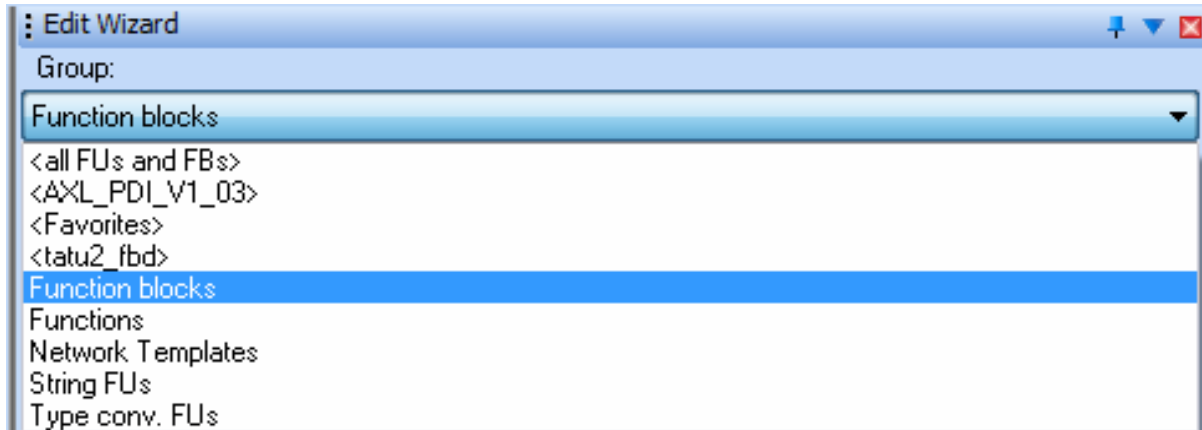


Fig. 4.39. Selecting the standard function block library

In the appearing list of function blocks (Fig. 4.40) select the line that contains the name of the PID function block with its PID Control description and drag the selected function block to the project working area.

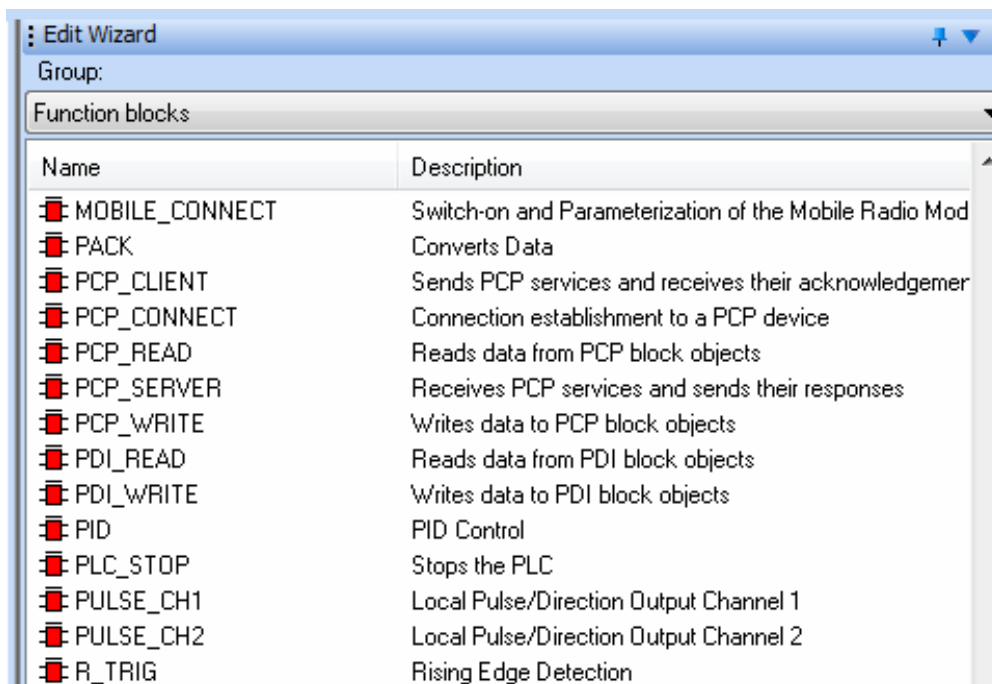


Fig. 4.40. Selection of the PID function block

The window will appear shown in Fig. 4.41.

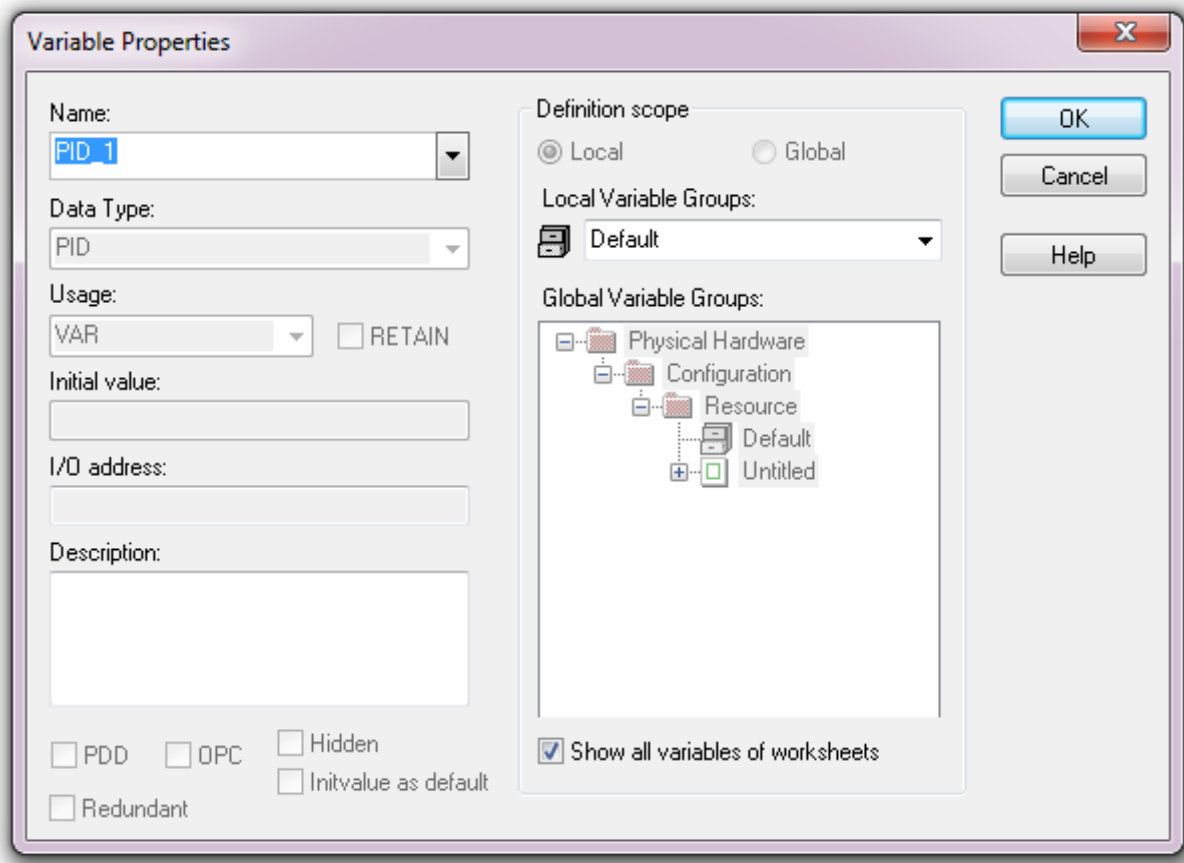


Fig. 4.41. The window for entering the initial parameters of the PID controller function block added to the project

To create the PID controller in the project, agree with the proposed characteristics and click OK. The PID controller added to the project is shown in Fig. 4.42.

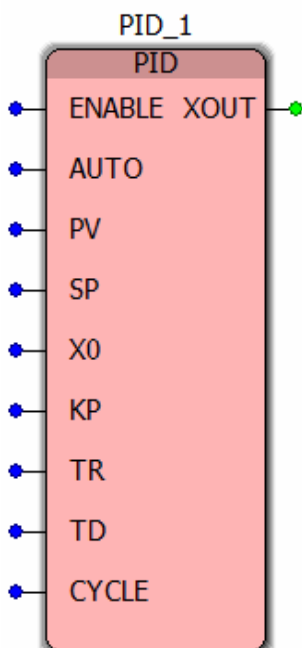


Fig. 4.42. The PID controller in the PC Worx project: ENABLE – starts the controller (BOOL); AUTO – determines the mode of operation – automatic (TRUE) or manual (FALSE); PV – input signal (REAL); SP – set point (REAL); X0 – offset of the controller output signal (REAL); KP – proportional gain (REAL); TR – integration time (REAL); TD – derivative time (REAL); CYCLE – block recalculation time (TIME); XOUT – inverted output of the controller (REAL)

To view and change the parameters of the PID controller, double click on the function block with the left mouse button. The opening window is shown in Fig. 4.43.

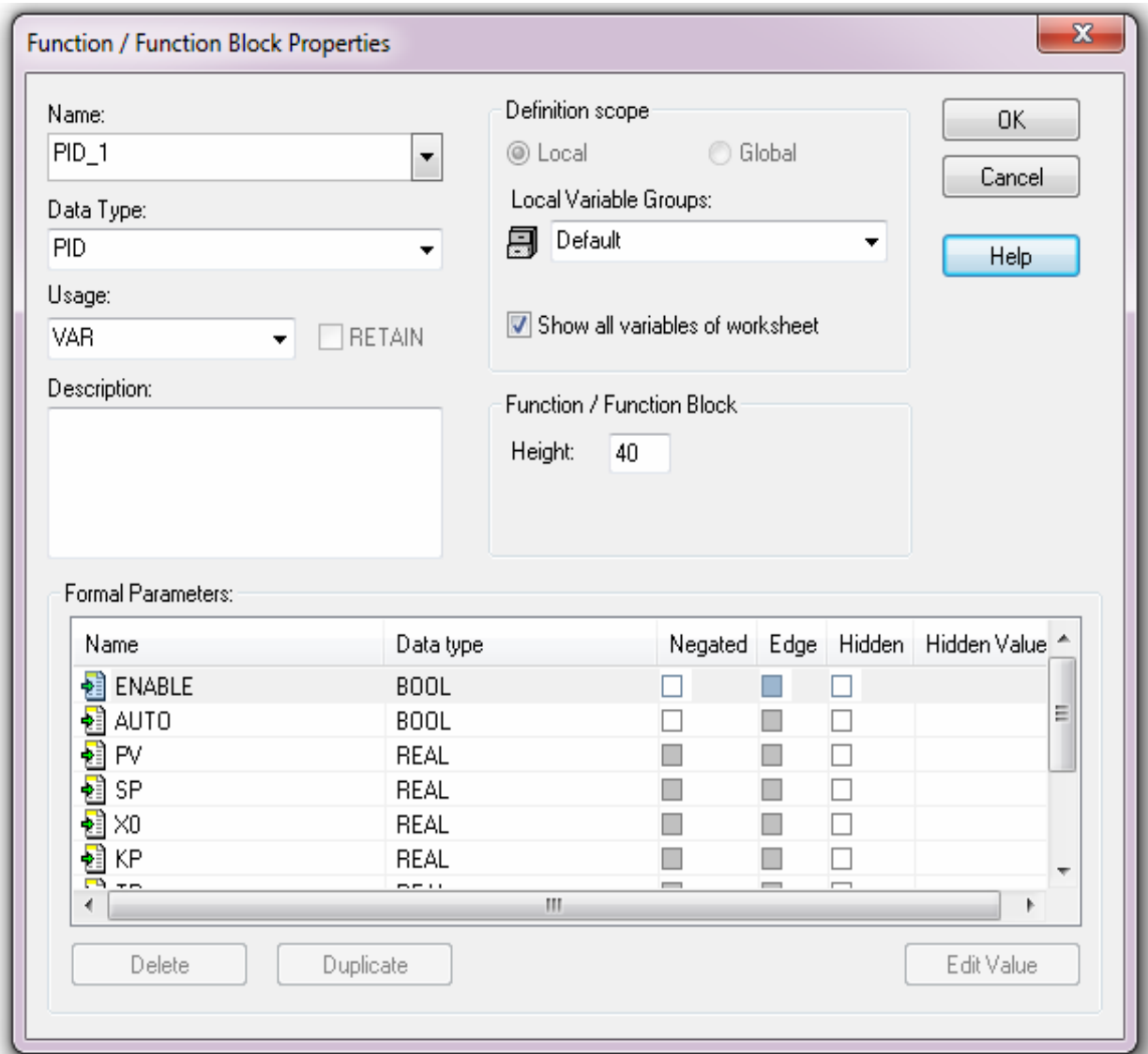


Fig. 4.43. PID-controller settings window

Individual work

Create a program in FBD (Functional Block Diagram) to calculate the following logical expression (do not simplify the expression):

$$Y = (X3 + \overline{X2}) \cdot X1.$$

Fill in the truth table (Table 4.6) and check the values obtained.

Use the switches of the third hardware module as input signals $X1$, $X2$, $X3$.

Use the LED of the third hardware module as the output signal Y .

Truth table

No.	Input signals			Output signal Y	Check
	X1	X2	X2		
1	0	0	0		
2	0	0	1		
3	0	1	0		
4	0	1	1		
5	1	0	0		
6	1	0	1		
7	1	1	0		
8	1	1	1		

Test questions

1. List the main data types that are used in PC Worx.
2. What information is stored by the INT data type?
3. How can a new variable be created?
4. What is the Initial value parameter responsible for?
5. How can a program be downloaded to the controller?
6. How can the downloaded program be started?
7. How can the state of a variable when the program is running be changed?
8. How can the analog variables from IB IL AI 4/U-PAC 0.3 be connected to the project?

4.2. Ladder Diagram (LD)

Ladder diagram is a graphical language using standardized graphic symbols for relay contact circuits. It is suitable for building logical switches, with which chains of any complexity can be created.

The LD program is clear and represents logical operations as an electrical circuit with closed and open contacts. The flow or absence of current in this circuit corresponds to the result of a logical operation (unit – if current flows, zero – if current does not flow).

There are normally closed and normally open contact elements in electrical circuits:

—| |— normally open contact;

—| / |— normally closed contact;

—()— actuator, which is called a coil;

—(/)— actuator simulating the connection to the normally closed contacts of the relay winding.

Contacts and coils can have linked variables. The contacts do not change the value of the bound variable, only the link state on the right is changed depending on the linked variable.

The coil copies the state of the link to the left to the link on the right without changing and stores the corresponding state function or the left link transition in the corresponding linked variable.

A complete list of the elements of Ladder Diagram is given in the Annex.

If the diagram contains several chains, they are executed in each cycle from bottom to top. This means that the upper chain will receive the values of the variables from the lower chain only in the next cycle. However, the order of the circuits can be changed using labels and transitions.

The program is created in the following sequence.

1. After configuring the controller select the working window in PC Worx.
2. In the Project Tree Window select Logical POU's/Main* (Fig. 4.44).
3. Place the cursor in the Worksheet area.
4. In the main menu select the Object tab, which contains the LD operators (see Fig. 4.44).

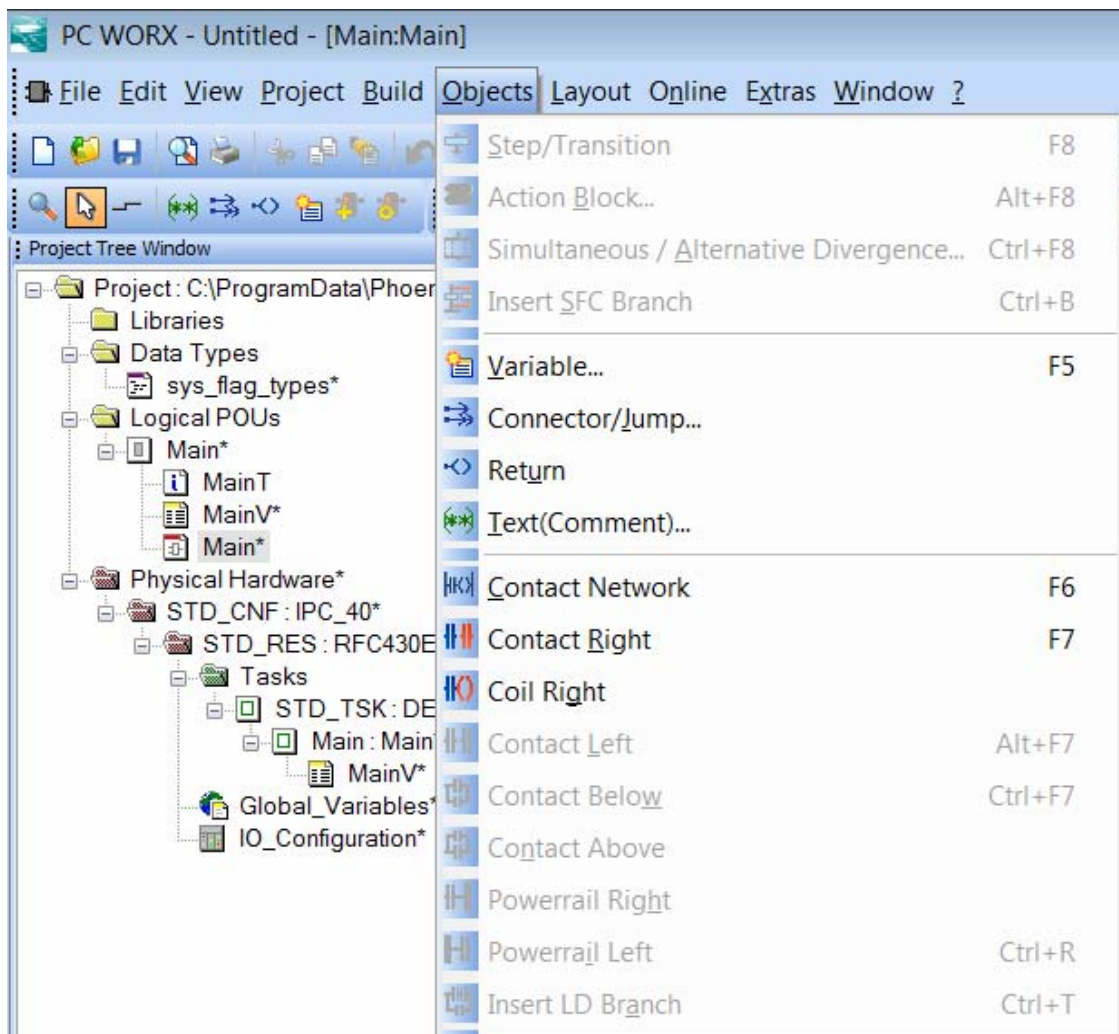


Fig. 4.44. Window to start working with LD

The program should be started from Contact Network (Fig. 4.45).



Fig. 4.45. Basic components of Ladder Diagram

At the initial stage the program consists of only three elements – poles 001, contact C000 and actuator C001.

If you try to compile this Build/Make program, error messages will appear in the Message Window (see Fig. 4.46).

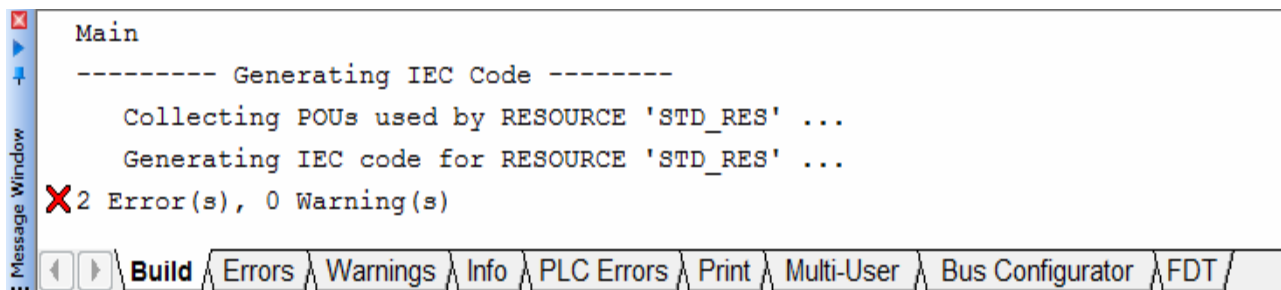


Fig. 4.46. Error messages

In order to detail the errors received, switch to the Errors tab in the Message Window (see Fig. 4.47).

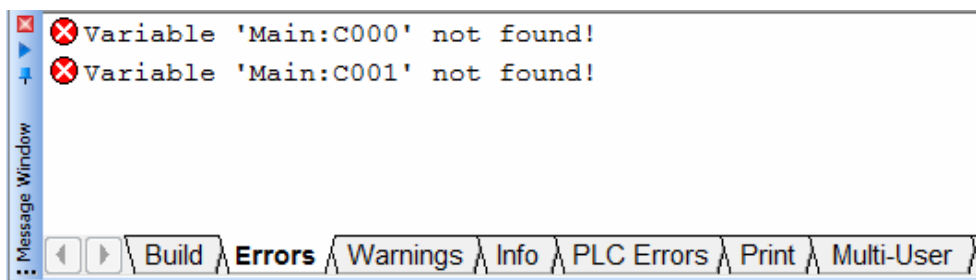


Fig. 4.47. Opening the Errors tab

Messages mean that variables C000 and C001 are not found. To test it, go to the variables section and make sure that the variables are only on the screen, not in the system. To initialize the variables, double-click on the C000 contact to open the variable input interface. Enter the name of the contact here, for example, In_1. Select the data type BOOL (Fig. 4.48) and click OK. Similarly, the actuator C001 named Out will be renamed.

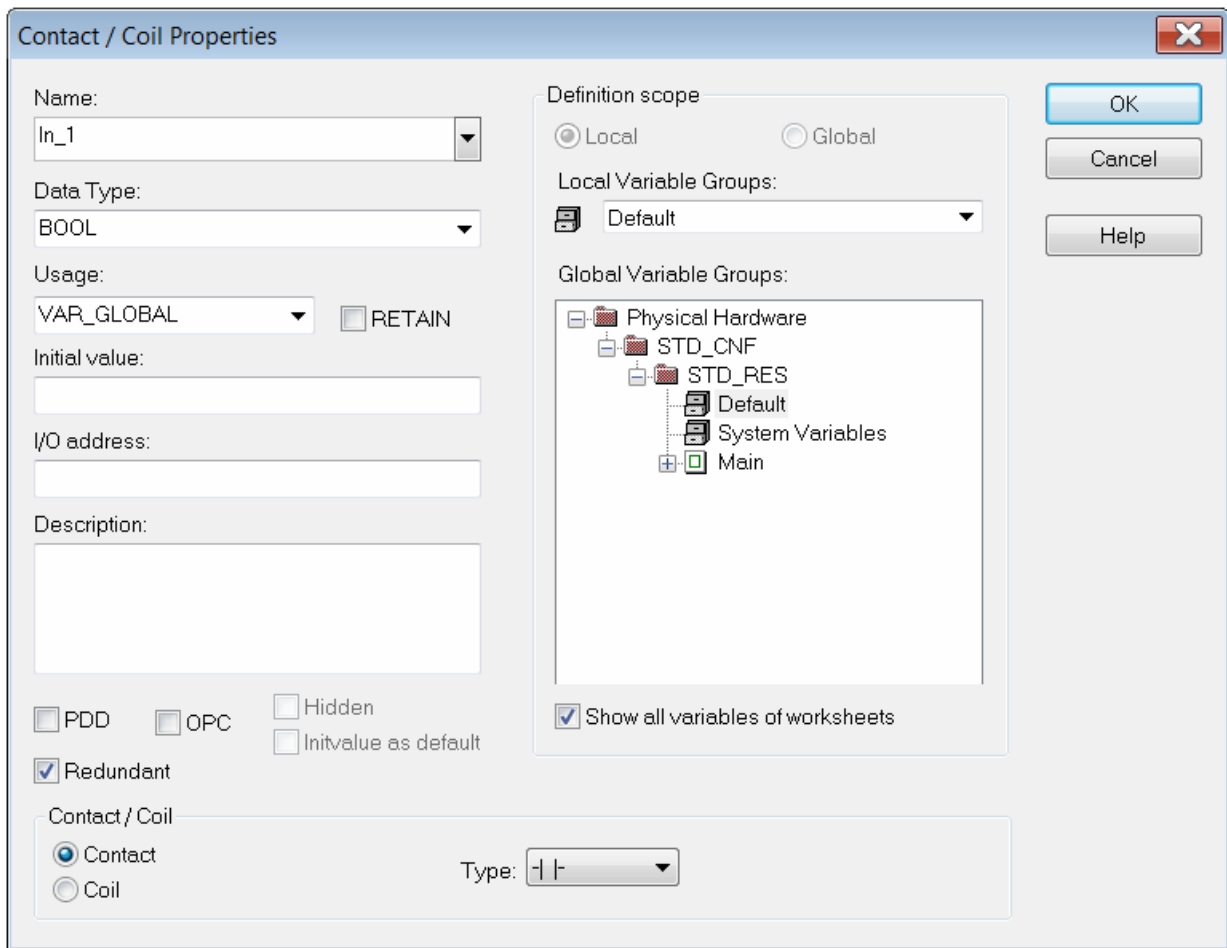


Fig. 4.48. Variable input interface

To check the newly entered variables, go to the variables section. To do this, the Logical POU's section opens in the Project Tree Window. The entered variables will be present in the list (Fig. 4.49).

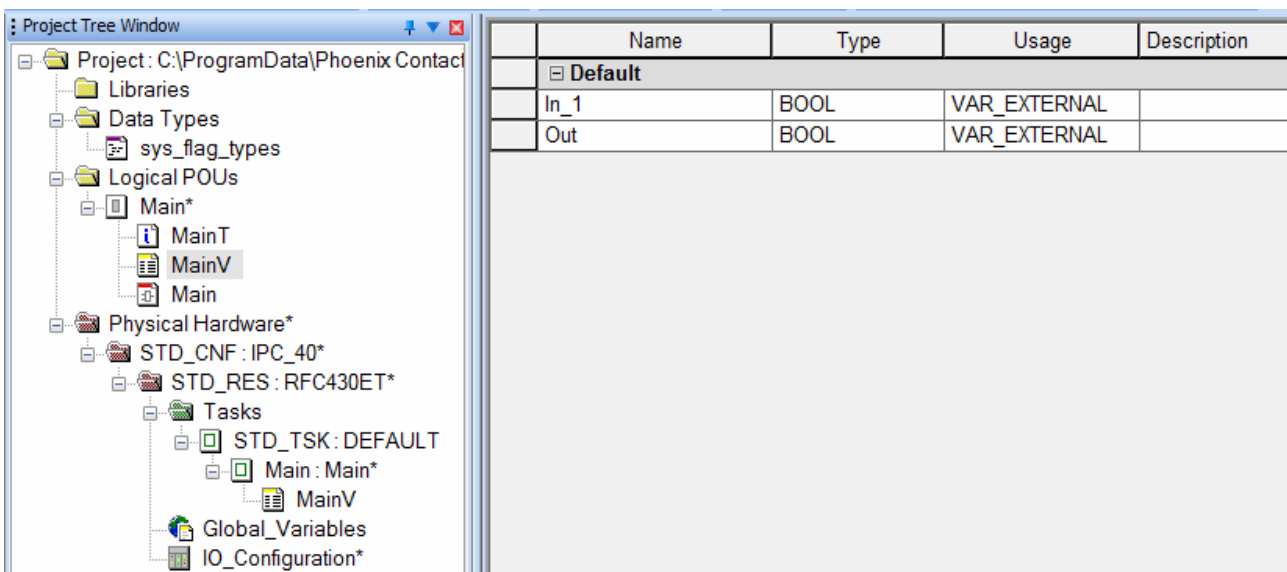


Fig. 4.49. Window with the variables list

After that the program will be compiled without errors and after running the program will work as shown in Fig. 4.50.



Fig. 4.50. Program operation at the initial stage

The blue color of the contact indicates its status. It corresponds to the "low level" (0) and the open contact status. At that, the current along the circuit between poles 001 does not flow and the actuator is switched off (also of blue color). The next step is to change the status of the contact. To do this, double-click on the contact (when the program is running) opening the status window (Fig. 4.51).

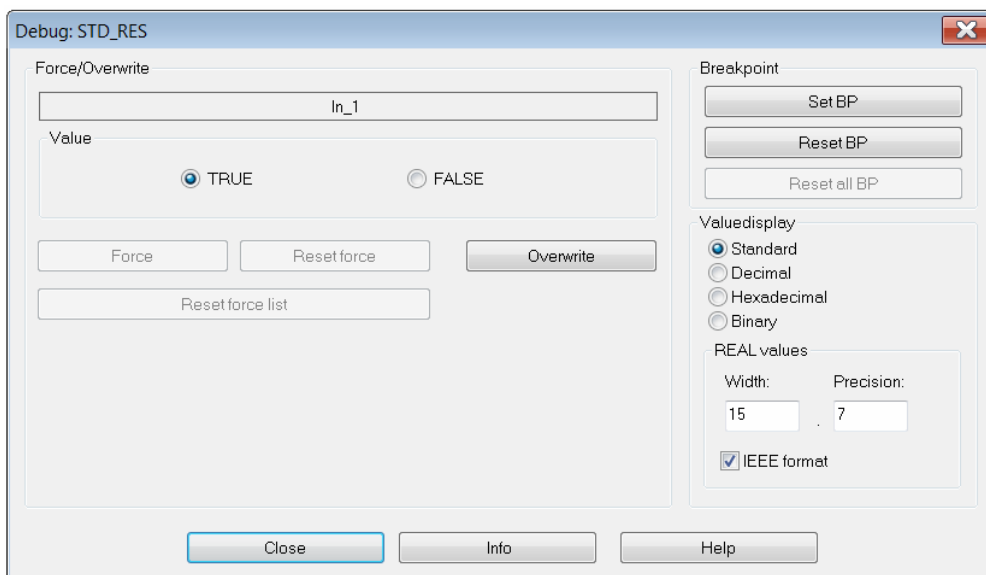


Fig. 4.51. Variable status window

In order to change the status, press the Overwrite button. The program will change the mode of operation: the contact and the coil are colored red (Fig. 4.52). This corresponds to logical "1" and means that the contact is closed. Therefore, the current between the poles 001 flows through the closed contact In_1 and turns on the actuator Out.



Fig. 4.52. Changing the operating mode of the program

This example illustrates the operation of the basic scheme.

Consider examples of adding supplementary elements using the example of parallel and serial connection of several elements.

Example 1. For parallel inclusion select the element with respect to which the commutation will be carried out by the mouse. Select In_1, open the top Object menu and select Contact Below (Fig. 4.53).

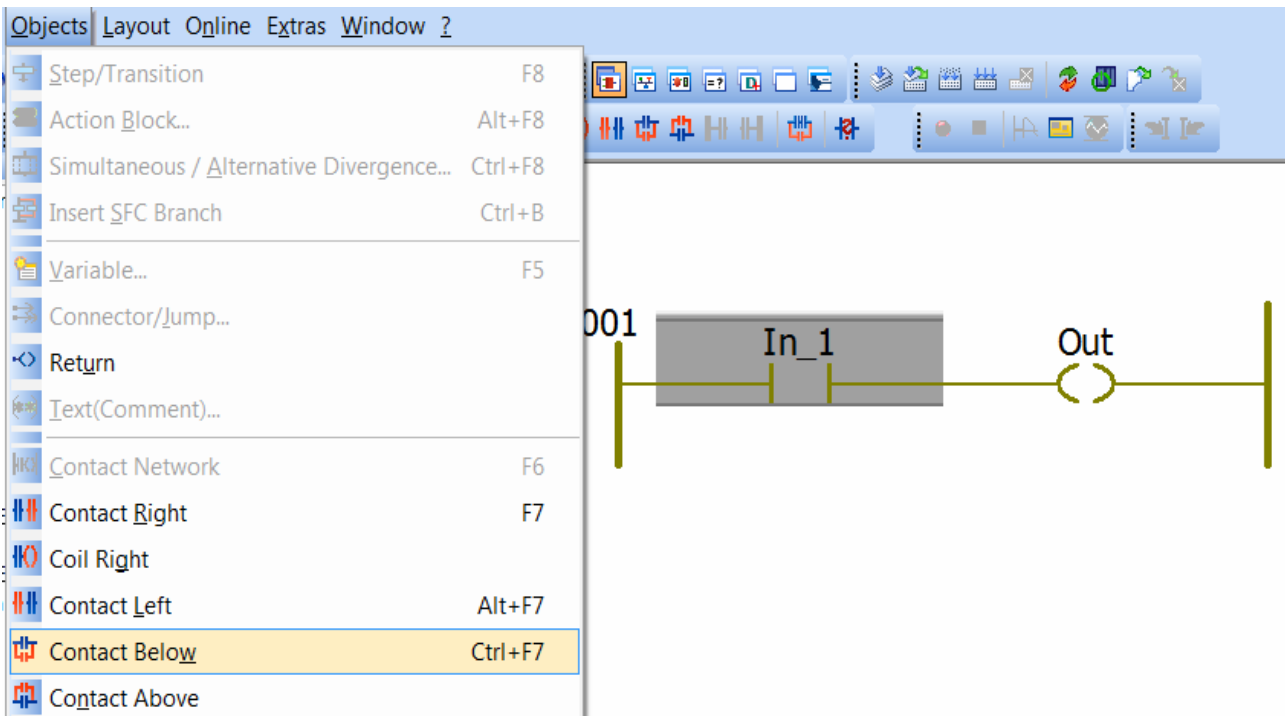


Fig. 4.53. Adding a contact to the scheme

The scheme will change, as shown in Fig. 4.54.

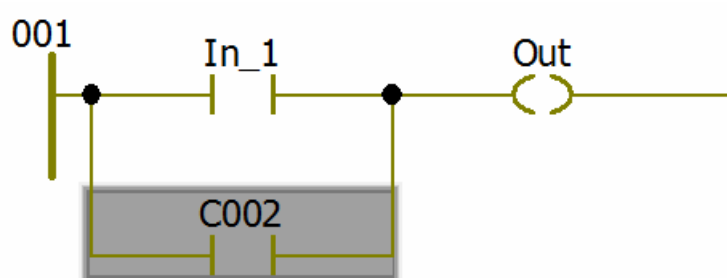


Fig. 4.54. Added new contact in the scheme

It can be seen that the newly appeared element is named C002 and is not included in the variable table. Double click to open the interface of the variable and rename this contact, assigning the name In_2. Compile and run the program. Then, changing the values of variables In_1 and In_2, fill in the truth table (Table 4.7).

Table 4.7

Truth table for parallel connection of contacts

In_1	In_2	Out
0	0	0
0	1	1
1	0	1
1	1	1

Example 2. Conduct circuit switching and connect the contacts in series, as shown in Fig. 4.55.

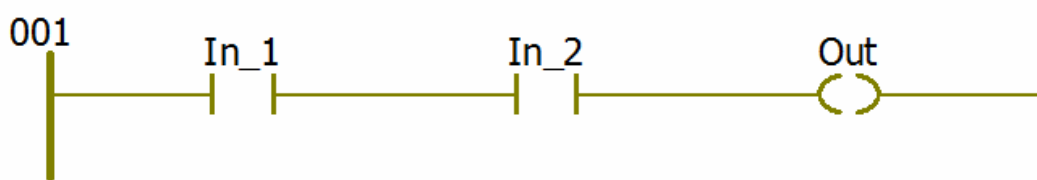


Fig. 4.55. Sequential connection of contacts in the scheme

Run the program and fill in the truth table (Table 4.8).

Table 4.8

Truth table for serial connection of contacts

In_1	In_2	Out
0	0	0
0	1	0
1	0	0
1	1	1

Example 3. Consider inverting operations. To do this, modify the circuit and return to basic scheme (Fig. 4.56).

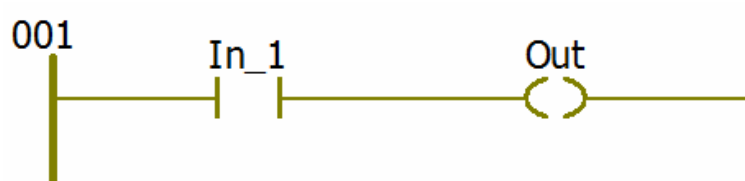


Fig. 4.56. Basic scheme of the program

Double-clicking on the In_1 contact opens the properties window. Change the type of the Type $\text{---}|\text{---}|\text{---}$ element at the bottom of the Contact/Coil section, as shown in Fig. 4.57. After this, the circuit will change and take the form shown in Fig. 4.58.

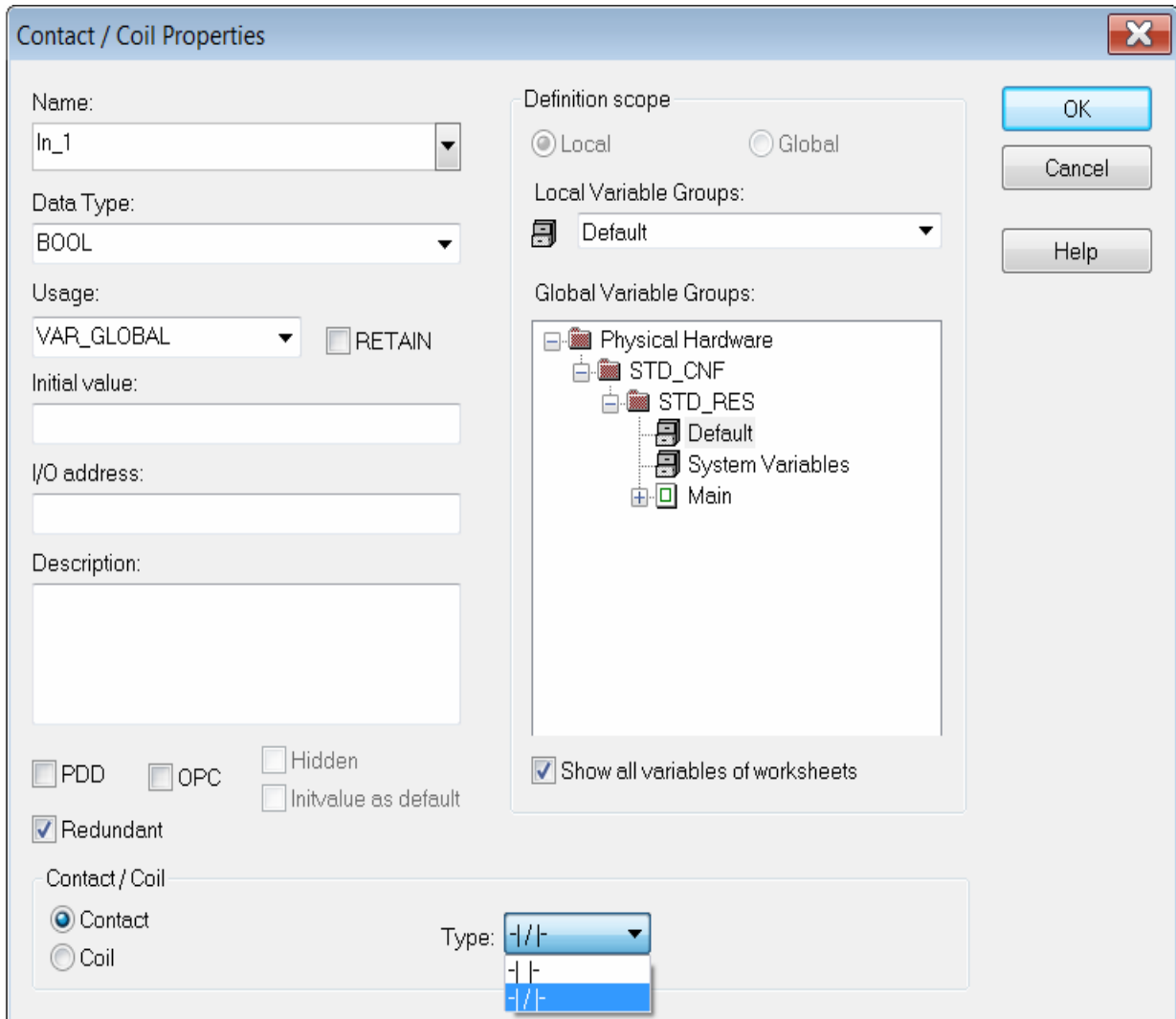


Fig. 4.57. Changing the contact properties



Fig. 4.58. Changing the contact type

Run the program and fill in the truth table with a normally closed contact (Table 4.9).

Table 4.9

Truth table for normally closed contact

In_1	Out
1	1
0	0

Example 4. In the next step make changes to the scheme and remove the inversion from the contact "In_1", and then set the inversion to the actuator Out. The resulting scheme is shown in Fig. 4.59.



Fig. 4.59. Changing the type of actuator

Run the program and fill the truth table at the inverting of the actuator (Table 4.10).

Table 4.10

Truth table for inversion of the actuator

In_1	Out
0	1
1	0

Having analyzed the work of the resulting schemes with inversion of the contact and the coil and based on the filled truth tables, it can be concluded that the operation of the entire scheme as a whole is affected by inversion of various elements.

Individual work

Create a program in Ladder logic to calculate the next logical expression with the input signals $X1$, $X2$, $X3$ and the output signal Y :

$$Y = \overline{X1} + X2 \cdot X3.$$

The expression should not be simplified. Fill in the truth table (Table 4.11). Check the received values.

Truth table

No.	Input signals			Output signal Y	Check
	X1	X2	X2		
1	0	0	0		
2	0	0	1		
3	0	1	0		
4	0	1	1		
5	1	0	0		
6	1	0	1		
7	1	1	0		
8	1	1	1		

Test questions

1. When is it advisable to use the Ladder Diagram?
2. List the elements used in the Ladder Diagram.
3. How can be a new LD element set on the work area?
4. What state corresponds to the blue color of the element?
5. How is the inversion of the Ladder Diagram element carried out?

4.3. Structured Text (ST)

Structured Text is a high-level programming language, which is syntactically close to the classical Pascal language. It can be used in basic programs, in the body of functions or function blocks (FBDs), for describing actions within program elements written in SFC and Flow Chart languages. Structured Text is intended for programming complex algorithms.

Many ST keywords (for example, AND, OR, NOT, IF, THEN, ELSE, WHILE, etc.) found in other programming languages are obvious and familiar, so the program code has good readability and allows quick understanding of the programs written by other developers.

When you enter keywords, delimiters and comments, a syntax check is performed. When a keyword, delimiter or comment is found, they are highlighted in color. When you enter unauthorized keywords (syntax errors), they are also highlighted in color. According to IEC 61131-3, keywords must be entered with uppercase characters. Spaces and tabs do not affect the syntax.

Expressions in the ST consist of operands and operators. An operand can be a constant, a variable, a structured variable, a function call, an output of a function block, or a direct address.

The data types that variables can have are described in par. 4.1 and Table 4.1.

Expressions of the ST programming language must end with semicolons. One line can contain several expressions, which are separated from each other by semicolons.

The data types of the operands processed in the expression must be the same. When working with data of various types, data types are first converted. For example, in expression

$$r3 := r4 + \text{SIN}(\text{INT_TO_REAL}(i1)) ;$$

the integer variable *i1* (data type INT) is first converted to a real variable (data type REAL), and then added to the real (REAL) variable *r4* (":"=" is the assignment operator).

TIME data type is the exception if variables are used with arithmetic operators "*" (multiplication) and "/" (division). The value of the TIME data type can be processed together with the value of the ANY_NUM data type (the generic designation of any numeric data types). The result of such an expression will have the TIME data type. For example, in expression

$$t1 := t2 * i4 ;$$

variable *t2* (TIME data type) is multiplied by the integer variable *i4*, and the result is assigned to the variable *t1* with the TIME data type.

An expression using different operators is evaluated in the order specified by the rule of precedence of operators. The operator with the highest precedence in the expression will be executed first, then the operator with the next higher precedence will be executed, etc., until the computation is completed. Operators with equal precedence are executed from left to right, as written in the expression. This order can be changed with parentheses.

ST operators are listed in Table 4.12.

During exponentiation the value of the first operand (base) is raised to the power of the value of the second operand (exponent), for example:

$$\text{OUT} := \text{IN1} ** \text{IN2} ;$$

With the help of the NOT operator the bitwise inversion of the operand is performed, for example:

$$\text{OUT} := \text{NOT IN1} ;$$

If $\text{IN1} = 1100110011$, then after performing the inversion the variable *OUT* will be 0011001100 .

Table 4.12

ST programming language operators

Operator	Operation	Operand	Example	Priority
()	Parentheses	Expression	(A+B)/C	11 (highest)
FUNCNAME (list of actual parameters)	Calling a function	Expression, constant, variable, direct address of data type ANY	MAX(X, Y)	10
^	Dereference	Variable address indicator (operator is used to get the value of the variable)	R^	9
-	Negation	Expression, constant, variable, direct address of data type ANY_NUM	-A, - A	8
NOT	Complement	Expression, constant, variable, direct address of data type ANY_BIT	NOT C	8
**	Exponentiation	Expression, constant, variable, direct address of data type REAL (base), ANY_NUM (exponent)	A**B	7
*	Multiply	Expression, constant, variable, direct address of data type ANY_NUM or TIME	A*B, A * B	6
/	Divide	Expression, constant, variable, direct address of data type ANY_NUM	A/B, A/B	6
MOD	Modulo	Expression, constant, variable, direct address of data type ANY_NUM	A MOD B	6
+	Add	Expression, constant, variable, direct address of data type ANY_NUM or TIME	A+B, A + B	5
-	Subtract	Expression, constant, variable, direct address of data type ANY_NUM or TIME	A-B, A - B	5
<, >, <=, >=, =, <>	Comparison	Expression, constant, variable, direct address of data type ANY_ELEM	A<B, A > B	4
&, AND	Boolean AND	Expression, constant, variable, direct address of data type ANY_BIT	A&B, A & B	3
XOR	Exclusive OR	Expression, constant, variable, direct address of data type ANY_BIT	A XOR B	2
OR	Boolean OR	Expression, constant, variable, direct address of data type ANY_BIT	A OR B	1 (lowest)

In the MOD operator the value of the first operand is divided by the value of the second operand, and the remainder of the division is returned as a result. For example, in expression

$$\text{OUT} := \text{IN1 MOD IN2} ;$$

the OUT variable will be equal to 2 when IN1 = 12 and IN2 = 5.

When using the comparison operator "more than" the value of the first operand is compared with the value of the second operand. If the first operand is greater than the second, the result is logical unit. If the first operand is less than or equal to the second operand, the result is logical zero. For example, in expression

$$\text{OUT} := \text{IN1} > 10 ;$$

the OUT variable will be equal to 1 if IN1 > 10, otherwise it is equal to 0.

Similarly, relational operators "less", "greater than or equal to", "less than or equal to", "equal", "not equal" are performed.

When using the AND operator (logical AND, logical multiplication operation) the "AND" logical operation is performed between the operands. For example, using the expression

$$\text{OUT} := \text{IN1 AND IN2 AND IN3} ;$$

or

$$\text{OUT} := \text{IN1} \& \text{IN2} \& \text{IN3} ;$$

the OUT variable is equal to 1 if all variables IN1, IN2 and IN3 are 1, otherwise OUT is equal to 0.

When using the OR operator (logical "OR", logical addition operation), the "OR" logical operation is performed between the operands. For example, in expression

$$\text{OUT} := \text{IN1 OR IN2 OR IN3} ;$$

the OUT variable is equal to 1 if at least one of the variables IN1, IN2, IN3 is equal to 1.

With variables that have BYTE and WORD data types operations are performed bitwise.

When using the XOR operator (eXclusive "OR"), the "exclusive OR" operation is performed between the operands. For example, in expression

$$\text{OUT} := \text{IN1 XOR IN2} ;$$

the OUT variable is equal to 1 if the variables IN1, IN2 are not equal. If the variables IN1 and IN2 have the same state (both are equal to 0 or 1), the OUT variable will be equal to 0.

When using three or more operands, the XOR operator works as follows: if the number of variables having the value "1" is odd, the result of the operation is 1, otherwise the result is 0. For example, if

```
OUT := IN1 XOR IN2 XOR IN3 XOR IN4 XOR IN5;
```

the OUT variable will be equal to 1 if the odd number of operand variables (1, 3 or all 5) is equal to 1.

The following expressions are available in ST:

assignment;

declaration of variables VAR ... END_VAR;

declaration of function blocks;

commands IF ... THEN ... END_IF;

ELSE;

ELSIF ... THEN;

CASE ... OF ... END_CASE;

FOR ... TO ... BY ... DO ... END_FOR;

WHILE ... DO ... END_WHILE;

REPEAT;

EXIT;

RETURN;

In the assignment operator A: = B; both variables should have the same data type.

The VAR ... END_VAR construct is used for declaring function blocks and direct addresses if they are not applied to the default data types.

The keyword VAR is entered once at the beginning of the section of the project code. All function blocks and direct addresses that differ from the default data types must be declared in this section. The keyword END_VAR completes the declaration of variables.

Declaration of function blocks and direct addresses applies only to the current section. If the same type of function block or the same address is used in another section, the function block type or address must be declared again.

In the function block declaration each block name is assigned the name of the block instance necessary to identify the function block in the project. The instance name must be unique for the entire project. When specifying the names of instances, the letter case does not matter (lowercase is equated to capital letters). The instance name must satisfy the formal requirements for the name otherwise an error message will appear.

After entering the instance name, enter the type of function block, for example CTD_DINT, ROL_WORD, SIN_REAL.

With generalized types of function blocks (for example, MUX, SEL), the data type is not specified. It will be determined by the data type of the actual parameters. If the actual parameters consist of constants, the INT data type will be accepted for the function block.

Any number of instance names can be declared for function blocks. The example of the declaration of function blocks is given below.

```
VAR
    RAMP_UP, RAMP_DOWN, RAMP_X: TON ;
    COUNT: CTU_DINT ;
    CLOCK: SYSCLOCK ;
    PULSE: TON ;
END_VAR
```

In this example, RAMP_UP, RAMP_DOWN, RAMP_X, PULSE are instances of TON function blocks; COUNT and CLOCK are input signals.

In the direct address declaration each used direct address, that has data type other than the default, is assigned the selected data type. For example:

```
VAR
    AT %QW1: WORD ;
    AT %IW15: UINT ;
    AT %ID45: DINT ;
    AT %QD4: TIME ;
END_VAR ;
```

The construct IF ... THEN ... END_IF allows organizing the conditional check. Statement block is executed if the condition is met (or the statement specified in the condition results in logical 1), which is specified after the IF keyword. The keyword THEN is between the condition and the statement being executed. The keyword END_IF is placed at the end of the construct. For example:

```
IF A>B THEN
    C := SIN(A) * COS(B) ;
    B := C - A ;
END_IF ;
```

If the FLAG variable in the following expression is equal to logical 1, this segment of the program will also be executed.

```
IF FLAG THEN
    C := SIN(A) * COS(B) ;
    B := C - A ;
END_IF ;
```

In the IF NOT ... THEN ... END_IF construct the NOT keyword is used to invert the condition. For example:

```
IF NOT FLAG THEN
    C := SIN(A) * COS(B) ;
    B := C - A ;
END_IF ;
```

This section of code will be executed if the FLAG variable has the value equal to logical 0.

In order to test complex conditions, any IF ... THEN ... END_IF constructs can be embedded into each other.

The keyword ELSE always follows the keywords IF ... THEN, ELSIF ... THEN or CASE.

If the keyword ELSE follows the keyword THEN or ELSIF, the statement after the ELSE keyword (or statement block) will be executed if all previous conditions that are checked after the IF or ELSIF keywords are not executed (or the result is logical 0). For example:

```
IF A>B THEN
    C := SIN(A) * COS_REAL(B) ;
    B := C - A ;
ELSE
    C := A + B ;
    B := C * A ;
END_IF ;
```

With the help of the keyword ELSIF the ELSIF ... THEN construct allows organizing a more complex test. If the condition after the IF keyword is not executed or its test result is logical 0, and the condition specified after the ELSIF is executed (or its test result is logical 1), then the statement block specified after the keywords ELSEIF and THEN will be executed. For example:

```
IF A>B THEN
    C := SIN(A) * COS(B) ;
    B := SUB(C, A) ;
ELSIF A=B THEN
    C := ADD(A, B) ;
    B := MUL(C, A) ;
END_IF ;
```

In order to test complex conditions, it's possible to embed any number of statements IF ... THEN ... ELSIF ... THEN ... END_IF into each other. For example:

```

IF A>B THEN
    IF B=C THEN
        C := SIN(A) * COS(B) ;
    ELSE
        B := SUB(C, A) ;
    END_IF ;
ELSIF A=B THEN
    C := ADD(A,B) ;
    B := MUL(C,A) ;
ELSE
    C := DIV(A,B) ;
END_IF ;

```

The CASE ... OF ... END_CASE construct allows selecting the statement or statement block from the list of statements, depending on the value of the selection variable that should have INT data type. Each statement or group has a label that contains integer, several integer numbers separated by commas (ANY_INT data type) or integer values range (the lower and upper limits of the range must be specified). The statement (statement block) will be executed only when the label name matches the value of the selection variable. The OF keyword opens the beginning of the list of labels.

The statement (statement block) after the keyword ELSE will be executed if the selection variable does not match any of the above labels and was not found in any range of labels.

The keyword END_CASE means the end of this construct. For example:

```

CASE SELECT OF
    1, 5:      C := SIN(A)*COS(B) ;
    2:        B := C-A ;
    6..10:    C := A+B ;
ELSE        B := C*A ;
           C := A/B ;
END_CASE ;

```

The FOR ... TO ... BY ... DO ... END_FOR construct is used for organizing loops with a known number of repetitions. The keyword FOR is in the header of the loop and starts the execution of the loop body. The keyword END_FOR is in the last line of the construct and completes the loop. The number of repetitions is determined by the start and end values of the cycle parameters and the control variable. The start and end values and the control variable must have the same data type (DINT or INT) and cannot be changed while the loop is running.

The FOR keyword changes the value of the control variable from the initial value specified in the header of the loop to the final value. The default value of the control variable increment is 1. At the beginning of each loop repetition it is checked whether the upper limit of the control variable change is reached. If the upper limit is exceeded, the loop is automatically terminated otherwise the loop body is executed again.

The DO keyword indicates the end of the loop header and the beginning of the loop body. Terminating the loop can be performed ahead of time using the keyword EXIT. The keyword END_FOR completes the loop construct. For example, this cycle will be executed 50 times:

```
FOR I := 1 TO 50 DO  
  C := C * COS(B) ;  
END_FOR ;
```

When I = 51, the loop will be terminated.

If you want to use an increment that is not equal to 1, it's possible to specify it with the BY keyword. The increment value, the start and end value of the control variable must have the same data type (DINT or INT). If the increment is positive, the control variable will change from a lower value to a larger value (the lower limit should be less than the upper limit). If the increment is negative, the control variable will change from a larger value to a lower value (the lower limit should be greater than the upper limit). For example:

```
FOR I := 1 TO 10 BY 2 DO  
  C := C * COS(B) ;  
END_FOR ;
```

In this case, the loop will be executed 5 times, and the control variable will take the values 1, 3, 5, 7, 9. If I = 11, the loop will be terminated.

Loop

```
FOR I := 10 TO 1 BY -1 DO  
  C := C * COS(B) ;  
END_FOR ;
```

will be executed 10 times, and the control variable will take the values 10, 9, 8, 7, 6, 5, 4, 3, 2, 1. If I = 0, the loop will be terminated.

In the loop

```
FOR I := 10 TO 1 BY J DO  
  C := C * COS(B) ;  
END_FOR ;
```

the variable J must be defined before the start of the cycle. If it is greater than 0, the cycle will be executed only once. The number of times for the loop to be executed $J < 0$, depends on the value of the variable J. For example, if $J = -4$, the loop will be executed at $I = 10, I = 6, I = 2$, i.e. 3 times. If $J = -2.5$, the loop will be executed at $I = 10, I = 7.5, I = 5, I = 2.5$, i. e. 4 times, and for $I = 0$ the loop will be terminated.

If in the example

```
FOR I := 1 TO 10 BY J DO
  C := C * COS(B) ;
END_FOR ;
```

the value of the increment is negative ($J < 0$), the cycle will be executed only once (for $I = 1$).

An eternal loop may occur if the increment is 0. In these cases the appropriate error message is generated. For example:

```
FOR I := 1 TO 10 BY 0 DO
  C := C * COS(B) ;
END_FOR ;
```

```
FOR I := 1 TO 10 BY J DO
  C := C * COS(B) ;
END_FOR ;
```

In the second example the error occurs when $J = 0$.

The WHILE ... DO ... END_WHILE construct allows arranging a loop with a previously unknown number of repetitions. As long as the condition after the WHILE keyword is correct (true), the expressions that are the body of the loop are satisfied. If the condition is initially violated (false), the loop is not executed at all. As soon as the condition ceases to be true, the loop is terminated.

The early termination of the loop can be performed using the keyword EXIT. The END_WHILE keyword terminates the loop. For example:

```
WHILE var <= 20 DO
  var := var + 4;
END_WHILE ;
```

If the variable var in this example, which must be specified before the start of the loop, is negative or greater than 20, the loop will not be executed at all. So, with the initial value $var = 5$, the loop will execute when $var = 5, 9, 13, 17$. With $var = 21$, the loop will be terminated.

The REPEAT ... UNTIL ... END_REPEAT construct also allows organizing a cycle with a previously unknown number of repetitions. The cycle will be executed at least once. The condition for repeating the cycle is indicated after the keyword UNTIL. The cycle is repeated until the condition specified after the keyword UNTIL is violated.

The cycle can be terminated in advance with the use of the keyword EXIT. The construction ends with the keyword END_REPEAT. For example:

```
var := -1
REPEAT
    var := var + 2
    UNTIL var >= 101
END_REPEAT ;
```

The EXIT keyword is used for organizing the early termination from the FOR, WHILE and REPEAT loops.

If the EXIT keyword is inside the embedded loop, then an early termination occurs exactly from the cycle in which the keyword EXIT is located. After the early termination of the loop using the EXIT keyword, the first statement which is outside the left loop is executed, that is, after the keywords END_FOR, END_WHILE or END_REPEAT. For example:

```
SUM := 0 ;
FOR I := 1 TO 3 DO
    FOR J := 1 TO 2 DO
        IF FLAG=1 THEN EXIT;
        END_IF ;
        SUM := SUM + J ;
    END_FOR ;
SUM := SUM + I ;
END_FOR ;
```

If the FLAG variable is 0, the SUM variable will be 15 (1+ (1+2) + 2 + (1+2) + 3 + (1+2)) after the end of the loop. If the FLAG variable is set to 1, the SUM variable will be 6 (1 + 2 + 3) after the end of the loops (an early termination will be performed from the inner loop and the J variable will not be added to the accumulated amount).

By using the RETURN keyword an early exit from function, function block or program is performed, similar to using the EXIT keyword.

When processing and storing large amounts of data, it is advisable to use arrays. In order to create them, use the keyword ARRAY, and for declaration – the keywords TYPE ... END_TYPE. Below is an example of declaring one-dimensional and two-dimensional arrays.

TYPE

AR_1_4_W : ARRAY [1..4] OF WORD;

AR_1_8_1_4_W : ARRAY [1..8] OF AR_1_4_W;

END_TYPE

The one-dimensional array AR_1_4_W consists of four elements having the WORD data type. The two-dimensional array AR_1_8_1_4_W consists of 8 elements, each of which is the previously declared one-dimensional array AR_1_4_W, consisting of four elements. It follows that this two-dimensional array can be represented in the form of 8 rows of 4 elements (columns) in each, that is, a total of 32 elements having the WORD data type.

When accessing the ARRAY variable arrays, pointers (line and column numbers) can contain constants, integer variables of type ANY_INT and statements formed with their joint use, for example:

```
var1[i] := 8 ;
```

```
var2.tree[4] := var3 ;
```

```
var4[1+i+j*5] := 4;
```

Function blocks are called by the statement that consists of the name of the block instance and the list of values in parentheses (actual parameters) that will be substituted for the formal parameters (the names of the input signals). If the formal parameter is not assigned a value, then when the function block instance is called, the initial value of the variable or input signal, defined in the PC Worx Variable Editor, is used. If the initial value of the variable or input signal is not specified, the default value (0) is used.

Each instance of a function block can be called only once. If the function block has no inputs or the inputs are not parameterized, the function block is called before its outputs are used in other operations.

Before calling a function block, it must be declared using the VAR ... END_VAR construct.

The example of the function block call is given below.

```
CLOCK ( ) ;
```

```
COUNT (CU:=CLOCK.CLK3, R:=RESET, PV:=100 + VALUE) ;
```

```
PULSE (IN:=COUNT.Q, PT:=T#1S);
```

Here CLOCK () – is the name of the function block instance, CLOCK.CLK3 and RESET – are formal parameters, COUNT.Q and PT – are real parameters.

The outputs of function blocks can be used in conjunction with variables intended only for reading data from them. For example:

```
OUT := COUNT.Q ;
```

```
ACT := COUNT.CV;
```

Here COUNT – is the name of the function block instance, Q and CV – are formal parameters, OUT and ACT – are real parameters.

Comments are allowed in the Structured Text. They start with the symbols "(" (opening parenthesis and asterisk) and end with the symbols ")" (asterisk and closing parenthesis). Between these two pairs of characters an arbitrary comment in any language can be entered. Comments can be placed in any program position and are highlighted in green (Fig. 4.60). When you add a function block, a template is automatically created, in which the variable names and data types are displayed as comments.

```

1
2  (* Result as ANY_BIT *) := (* IN1 as ANY_BIT *) & (* IN2 as ANY_BIT *);
3
4  (* Result as ANY_BIT *) := (* IN1 as ANY_BIT *) OR (* IN2 as ANY_BIT *);
5

```

Fig. 4.60. Function block templates

Fig. 4.61 shows an example of the program in the FBD language, and Fig. 4.62 shows its equivalent in the ST language.

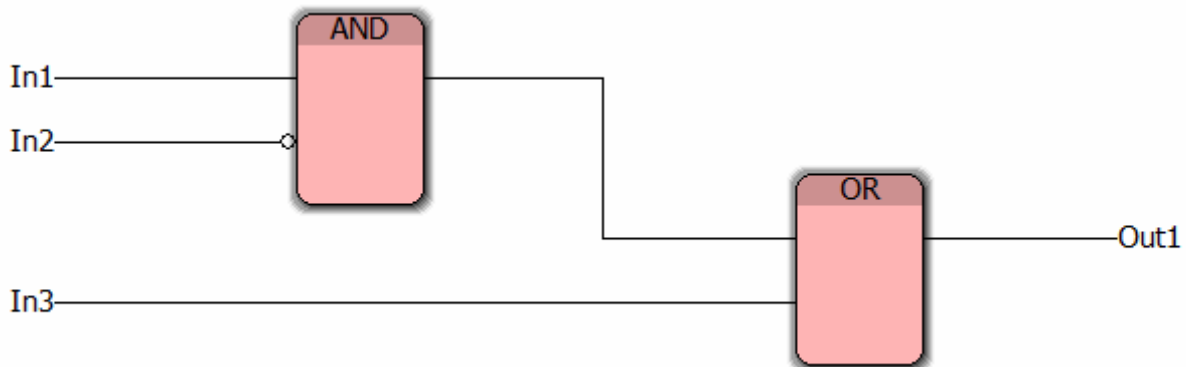


Fig. 4.61. The scheme (program) of the project in the FBD language

```

16
17 Out1 := In1 & not In2 or In3;
18

```

Fig. 4.62. The program in ST

Fig. 4.63 shows the functional block of the electronic low-pressure switch that is triggered (logical unit appears at the output of RND), when the current pressure P_s exceeds the specified upper limit value RND_Hi. If the current pressure is below the specified minimum value, the relay does not work (logic zero at the output of RND).

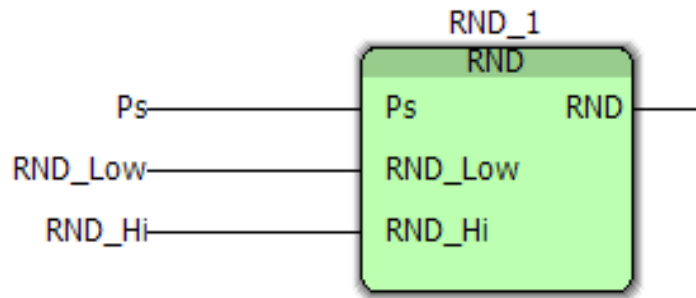


Fig. 4.63. Function block that implements the electronic low-pressure switch

Fig. 4.64 shows the corresponding ST program.

```

IF Ps > RND_Hi THEN RND := TRUE;
ELSE IF Ps < RND_Low THEN RND := FALSE;
END_IF;
END_IF;

```

Fig. 4.64. The program that implements the electronic low-pressure switch

In order to add the ST program to an open project, do the following.

1. Right-click on the Logical POU's line (Fig. 4.65).
2. In the appearing context menu select the Insert item, and then in the expanded window select the Program item.

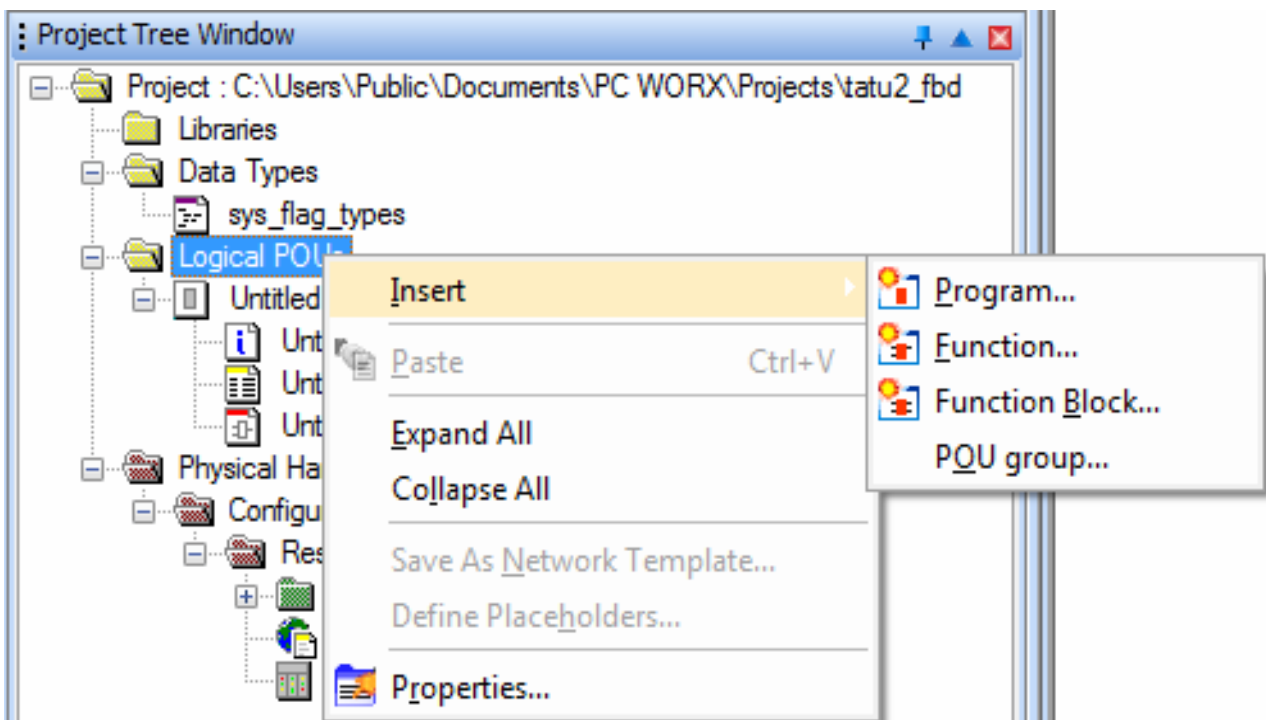


Fig. 4.65. Adding the ST program to an open project

The window will appear shown in Fig. 4.66.

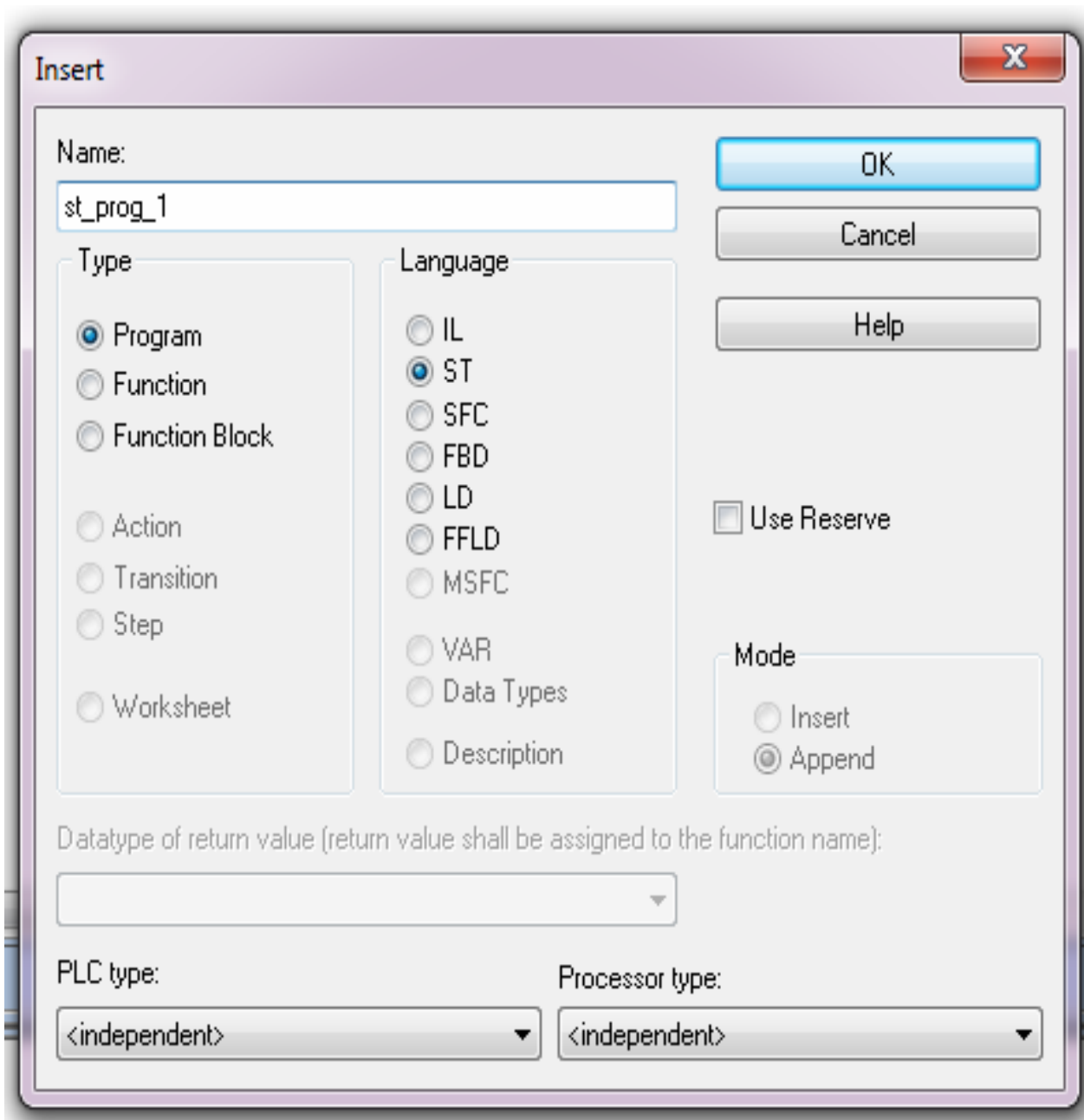


Fig. 4.66. The window for selecting the name of the program and the programming language

In this window select the ST language, type the name of the created program in the Name field (here st_prog_1) and click OK. The newly created st_prog_1 will appear in the list of programs used in the project (Fig. 4.67). The created program contains 3 parts: st_prog_1 – the program itself in the ST language; St_prog_1V – description of variables; St_prog_1T – text comments.

Similarly, it's possible to add an arbitrary number of programs in different programming languages to the project, which are allowed to be used in PC Worx.

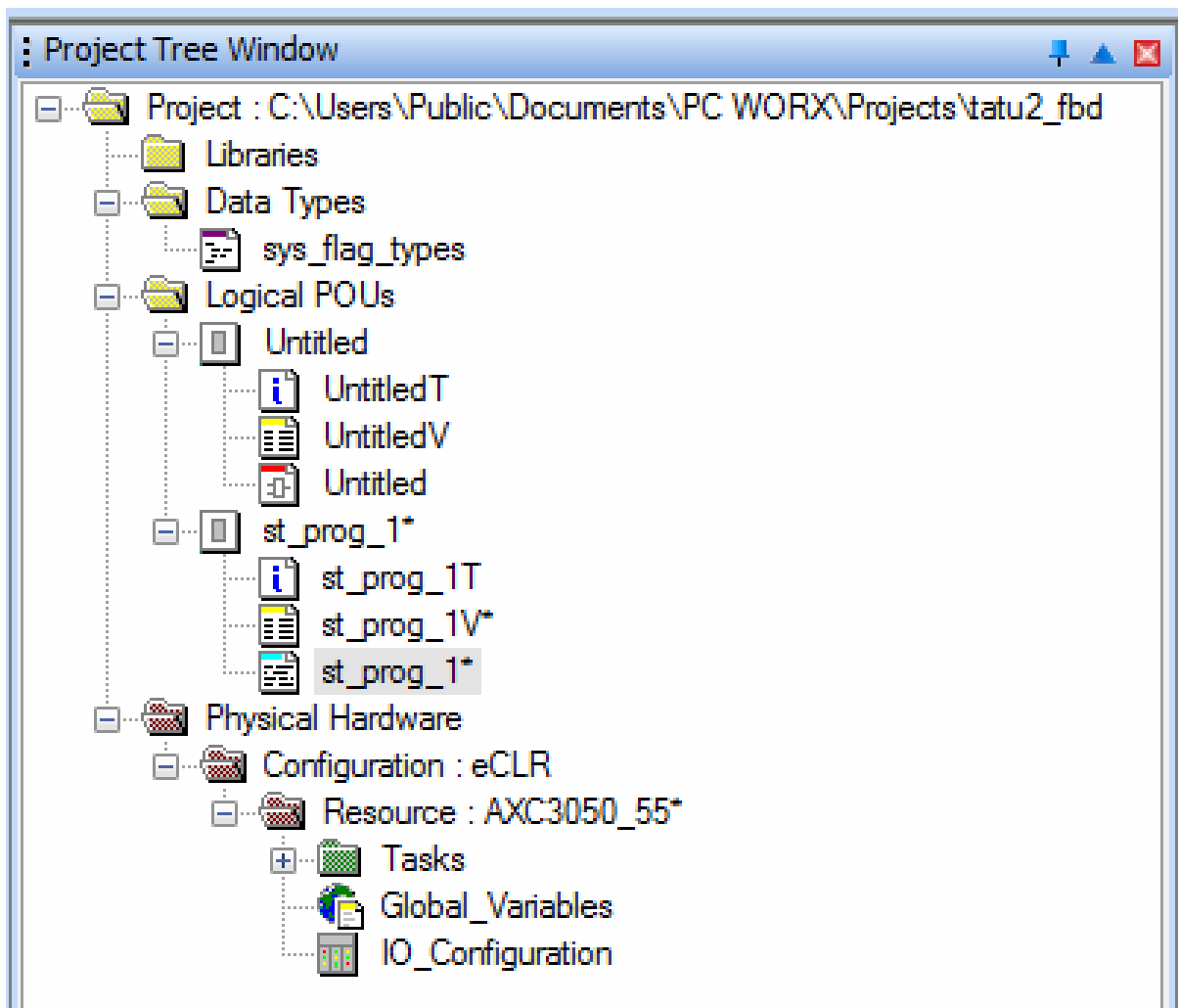


Fig. 4.67. The Project Tree window containing the previous and added items

Test questions

1. What are the advantages of Structured Text for programming controllers?
2. What software constructs (operators, keywords) allow checking the conditions in Structured Text; what are the differences between them?
3. What are the fundamentally different types of loops that can be used in Structured Text?
4. How does the FOR loop work?
5. How does the WHILE loop work?
6. How does the REPEAT loop work?
7. How is the function block called in Structured Text?
8. How is the function called in Structured Text?
9. How are one-dimensional and two-dimensional arrays declared in Structured Text?

5. EXAMPLES OF CONTROL SYSTEMS PROJECTS

5.1. Simulation model of ship auxiliary boiler control system

The control system model is designed for a ship auxiliary boiler equipped with a fuel injection system of "Monarch" type. The model was created using the Phoenix Contact ILC 130 ETH programmable controller. A schematic diagram of control system is shown in Fig. 5.1. The program of the ILC 130 ETH controller is written in the FBD language (Fig. 5.2).

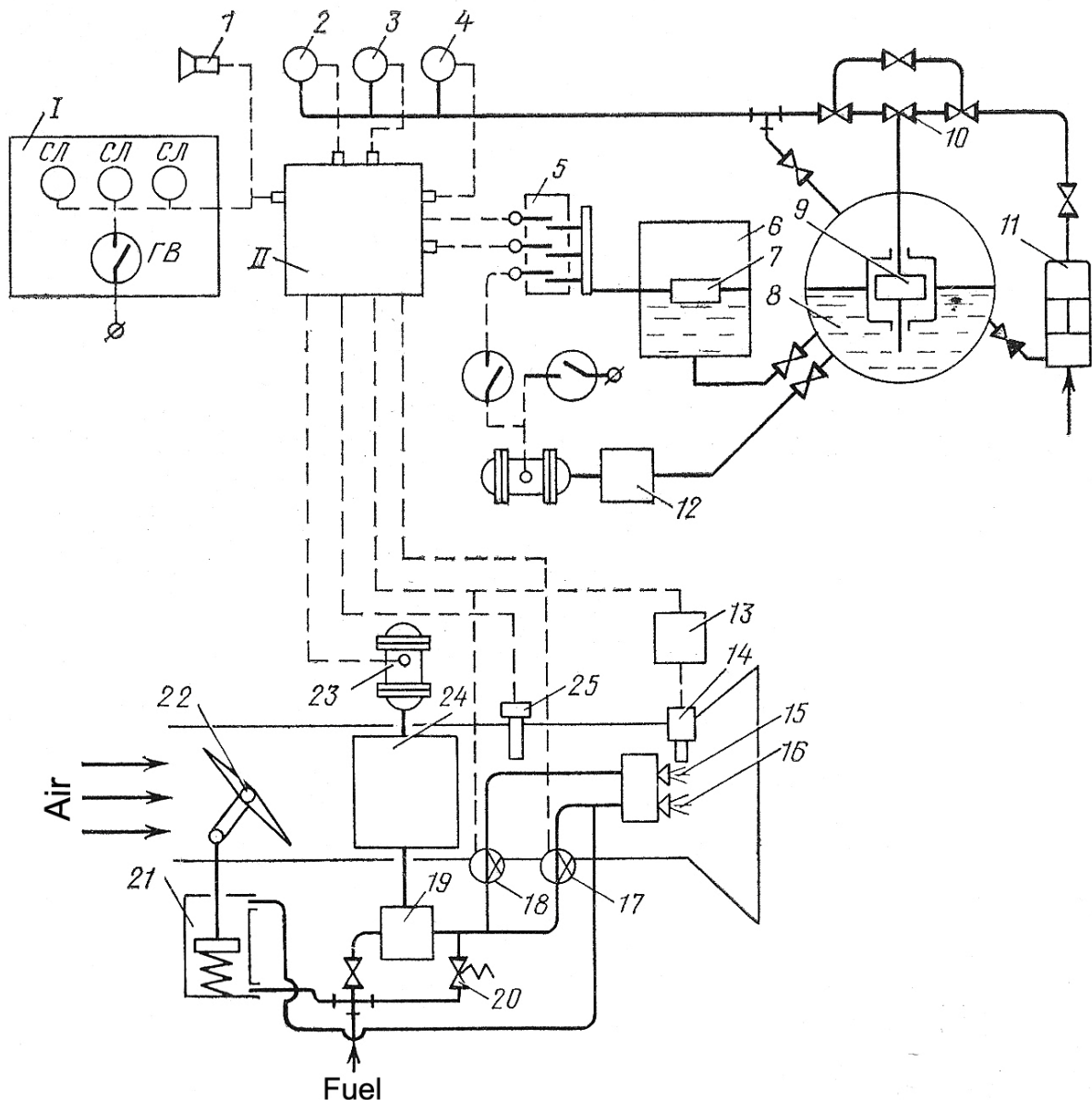


Fig. 5.1. Schematic diagram of the automatic position control system of auxiliary boiler: I – starting unit; II – programmed control unit; 1 – alarm; 2, 3, 4 – pressure relay; 5 – terminal box; 6, 7 – floating level sensors; 8, 9 – feeding pumps; 10 – ignition transformer; 11 – fuel injectors; 12, 13 – fuel valves; 14 – fuel supply pump; 15 – bypass valve; 16 – fan; 17 – photo sensor; 18 – damper servo piston

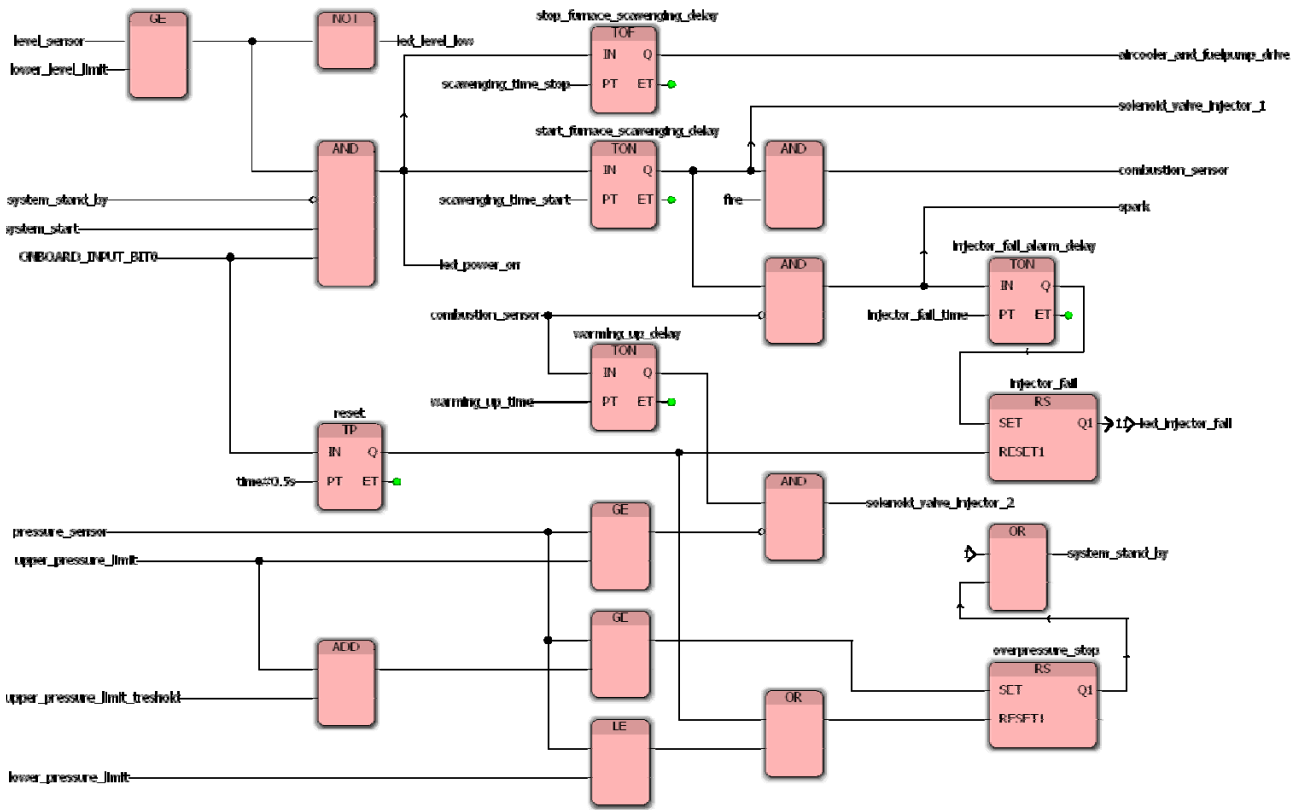


Fig. 5.2. Controller program in the FBD language

Visualization of the technological process is carried out in the form of a website in the WebVisit editor (Fig. 5.3).

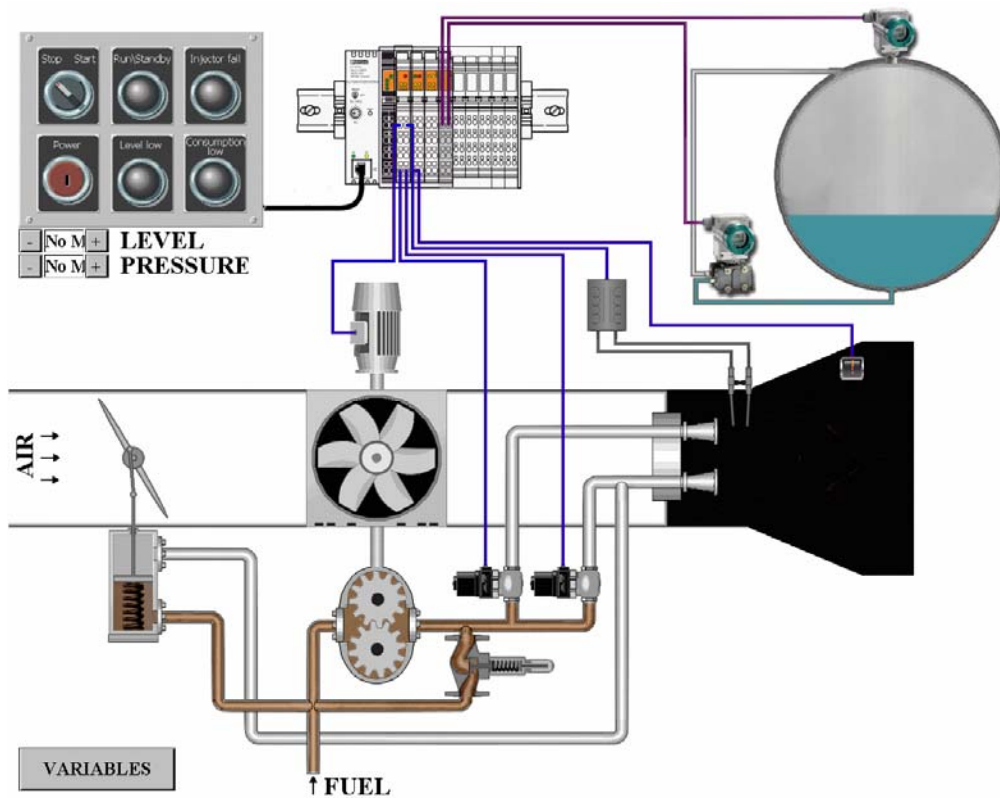


Fig. 5.3. Process visualization window

The software component of the computer system is a web server built into the control device. Depending on certain conditions, visualization of variables in the web browser is performed using a Java applet that is downloaded from the control device server and executed in the client browser that supports Java.

Fig. 5.4 shows the block diagram of computer-integrated management by Phoenix Contact tools.

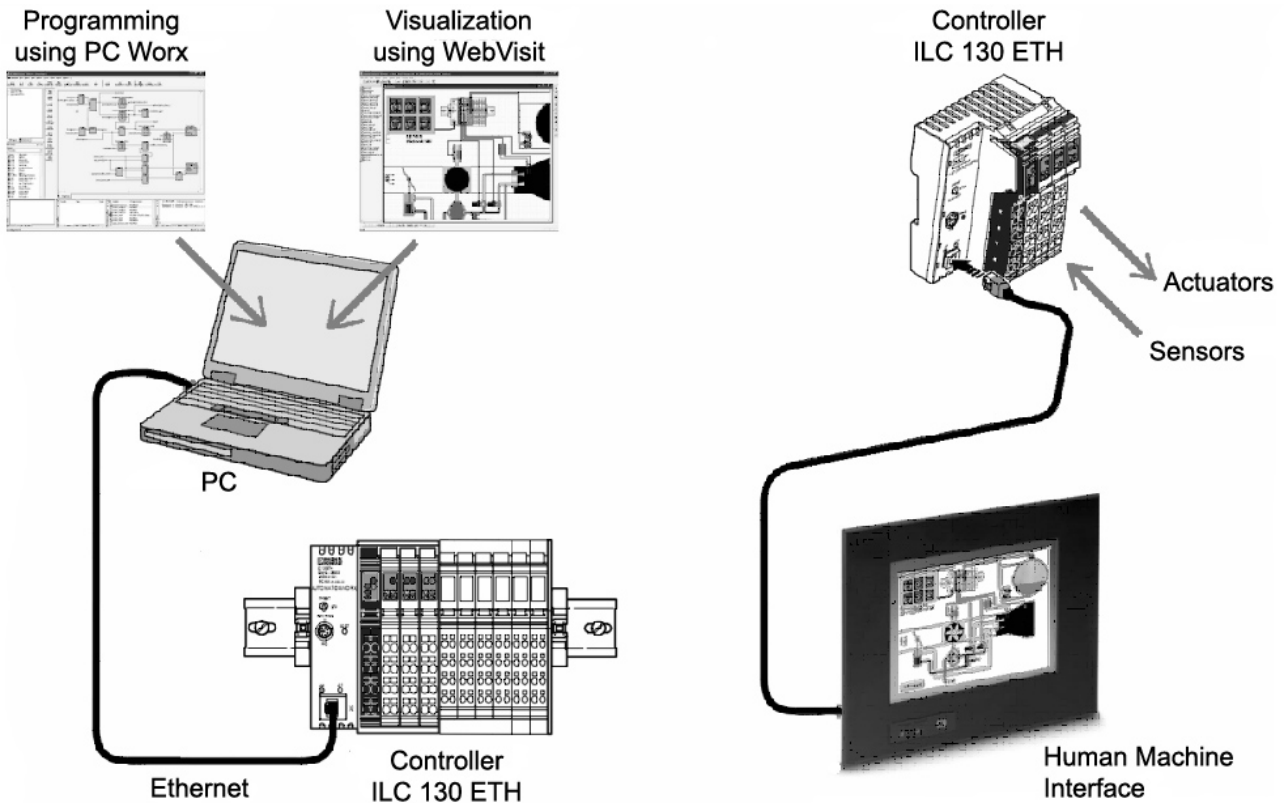


Fig. 5.4. The structure of computer-integrated management by Phoenix Contact tools

5.2. Control system for reciprocating compressor unit

The MAK-FV6 marine refrigeration system is used for cooling the provision storerooms by the air cooling system when the refrigerant evaporates in the air coolers, being a closed and hermetically sealed system (Fig. 5.5).

The operational monitoring system for current assessment and subsequent technical diagnosis is implemented on the basis of a personal computer. Such a system operates in "soft" real time, which leads to loss of information about the state of the refrigeration system. To provide operation in "hard" real time, a monitoring system based on the Inline ILC 130 ETH programmable controller is offered.

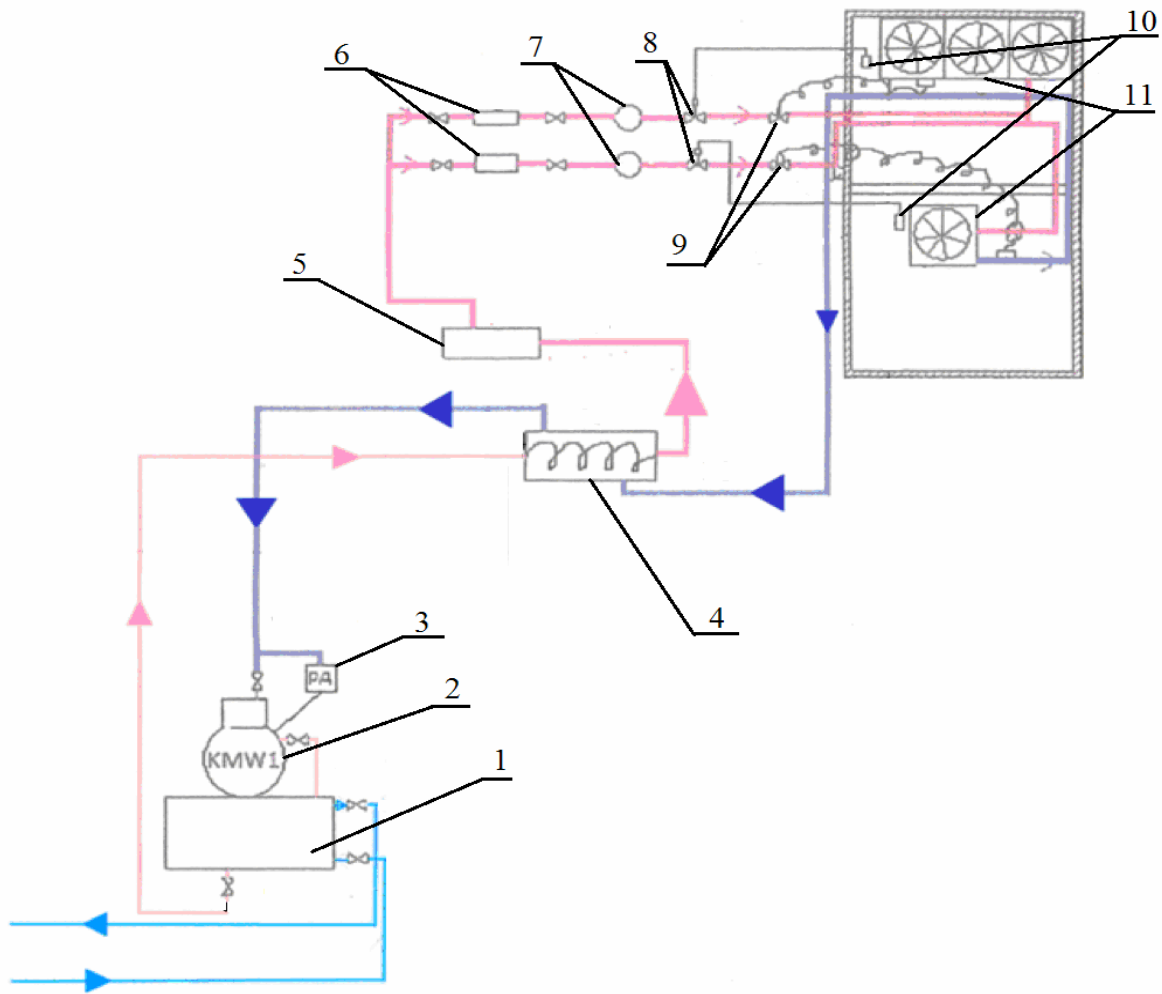


Fig. 5.5. Block diagram of refrigeration system:

1 – condenser; 2 – compressor; 3 – pressure switch; 4 – recuperative heat exchanger; 5 – filter-drier; 6 – filters; 7 – humidity detectors; 8 – solenoid valves; 9 – thermostatic valves; 10 – temperature switch; 11 – air coolers

Compressor specifications: grade – FV6; type – single-stage piston return flow vertical stuffing; number of cylinders – 2; cylinder diameter: 67.5 mm; piston stroke 50 mm; cooling capacity 6,000 kcal/h; refrigeration agent – Freon R-134a; boiling point 15 °C, condensation temperature 30 °C (under standard conditions). Freon unit is designed to cool two provision storerooms and allows maintaining the different temperature in them.

Two pressure sensors EWPA030 by Eliwell Controls srl (Fig. 5.6, a) are used for continuous pressure measurement on the suction side and the compressor discharge. The main characteristics of the sensors: measuring range 0 – 30 bar; output signal 4 – 20 mA; supply voltage of the sensor 8 – 28 V; wiring diagram – two-wire; operating temperature –20 ... +80 °C; pressure measurement error $\pm 1\%$ at 0 +50 °C and $\pm 2,5\%$ at temperatures below 0 and above +50 °C; protection degree IP65.

To obtain information about the temperature in the cooled rooms, as well as in the characteristic points of the refrigerator cycle, NTC thermistors by Semitec were used (Fig. 5.6, *b*). This model belongs to the category of high-precision temperature sensors with the following parameters: thermistor resistance at a temperature of 25 °C is kOhm; measuring range of the temperature is –50 ... +150 °C.



Fig. 5.6. Sensors used in the system: *a* – pressure EWPA030; *b* – NTC thermistor

A module made up of eight analog inputs IB IL AI 8/SF-PAC by Phoenix Contact was used as a communication device with objects for receiving information from sensors.

To obtain information on the state of refrigeration unit, the WP 04T operator panel by Phoenix Contact was used, the main characteristics of which are summarized in Table 5.1.

Table 5.1

WP 04T operator panel specifications

Screen	8.9 sm/3.5"-TFT
Screen resolution	320 x 240 Pixel (QVGA)
Backlight	LED
Number of colors	65,536
CPU	RISC ARM9™ CPU; 200 MHz
RAM	64 MByte SDRAM
Memory	32 MByte flash memory
Interfaces	2x USB Host 2.0
Network	1x Ethernet (10/100 MBit/s), RJ45
Operating system	Windows CE 5.0

The functional diagram of the monitoring system for the refrigeration system is shown in Fig. 5.7. It additionally provides for the possibility of control.

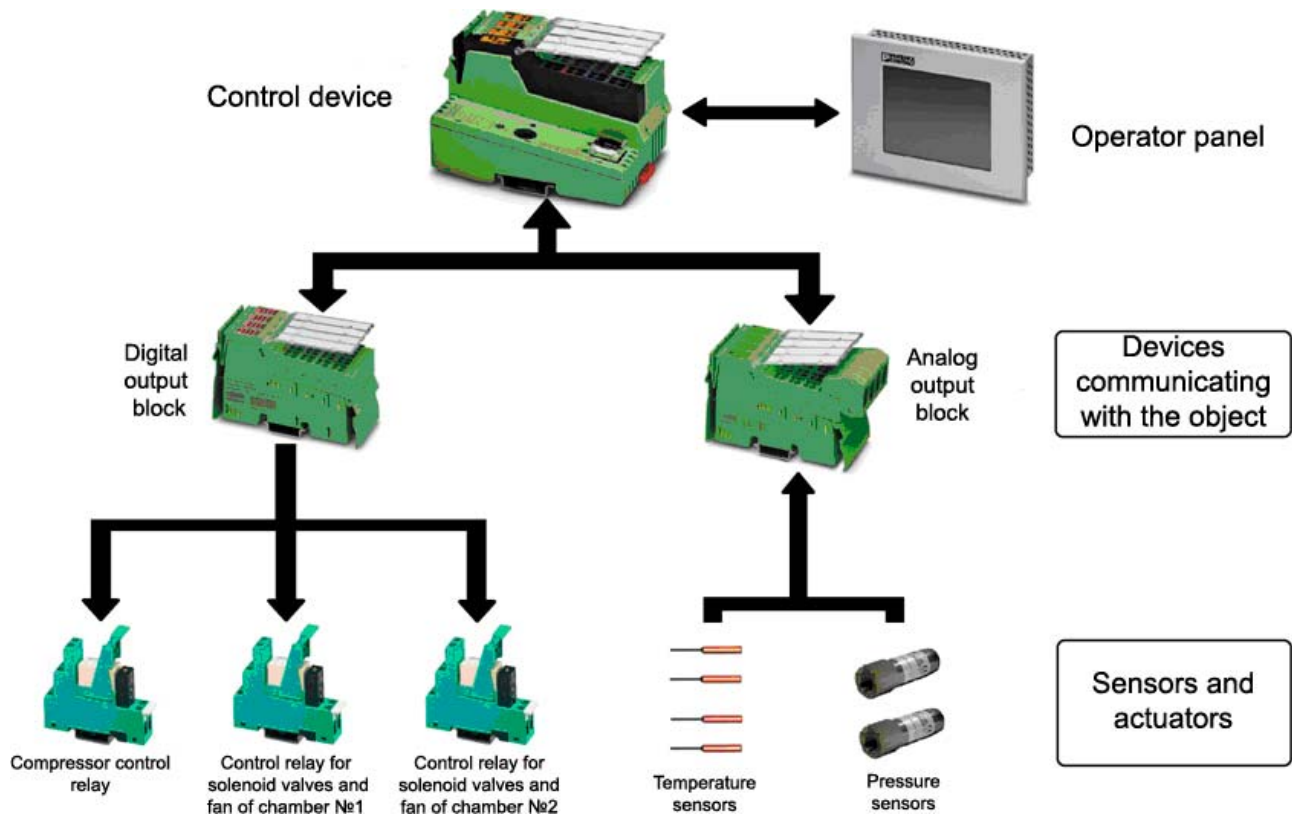


Fig. 5.7. Functional diagram of control system

Below are the steps (segments) of the program developed in PC Worx.

To simulate the operation of the low-pressure switch (LPS) a function block was constructed in the ST language that performs the same functions as its physical analog. Fig. 5.8 shows the appearance of the block and its program code.

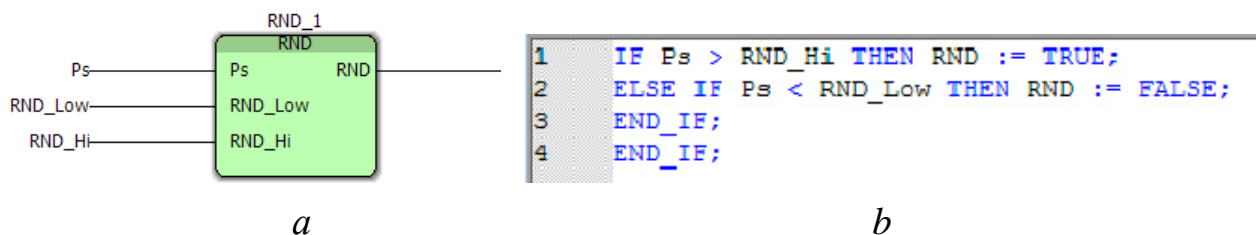


Fig. 5.8. Low-pressure switch: *a* – LPS function block; *b* – program code of the LPS block

The block has two configuration options: relay actuation pressure (RND_Hi) and cut-off pressure (RND_Low).

The input signal Ps for the block is the suction pressure of compressor, and the output signal is generated depending on the set points.

The block simulating the operation of the high-pressure switch (HPS) is formed similarly to the LPS block (Fig. 5.9).

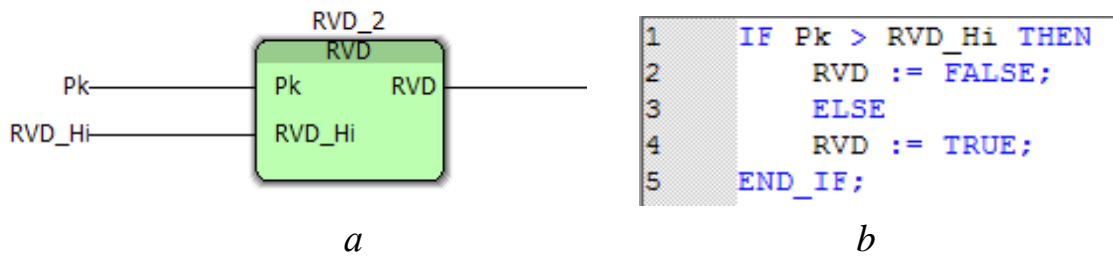


Fig. 5.9. High-pressure switch: *a* – HPS function block;
b – program code of the HPS block

The input signal P_k for the block is the discharge pressure of compressor. The block has one configuration options – relay actuation pressure (RVD_Hi). The block generates the output signal if the discharge pressure value exceeds the set point value.

The operation of the high temperature relay (HTR) unit is similar to that of the LPS unit. The only difference is that the temperature value on the discharge side of the compressor is used as the input signal T_k . The HTR block also has two configuration options, namely the actuation temperature T_Hi and the cut-off temperature T_Low .

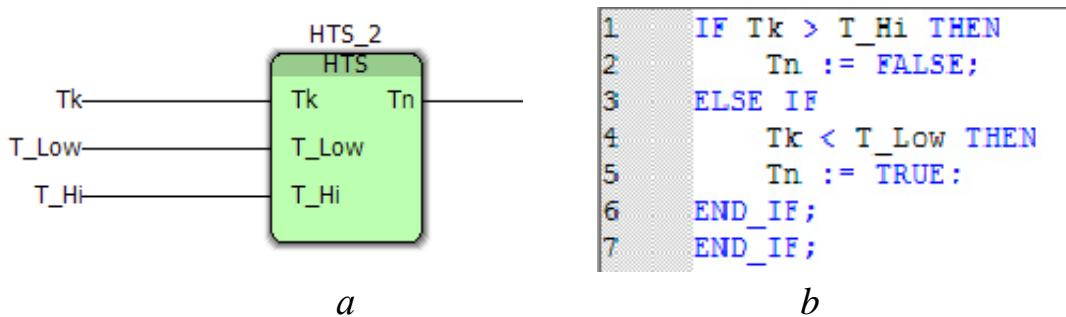


Fig. 5.10. High temperature relay: *a* – HTR function block;
b – program code of the HTR unit

Thermostat is a temperature relay with a thermal bulb and a capillary tube made of stainless steel or copper. The temperature that the thermostat maintains must correspond to the set average temperature (set point). The thermostat has an unregulated differential.

In order to simulate the operation of thermostat, a functional unit was constructed (Fig. 5.11, *a*) for maintaining the set temperature in the cooled rooms by acting on the solenoid valves on the fluid line of the compressor before the

thermal expansion valve, and starting and stopping the evaporator fans. The thermostat is able to maintain the set temperature in two provision storerooms. The fans are switched on and off together with the operation of the solenoid valves (Fig. 5.11, *b*).

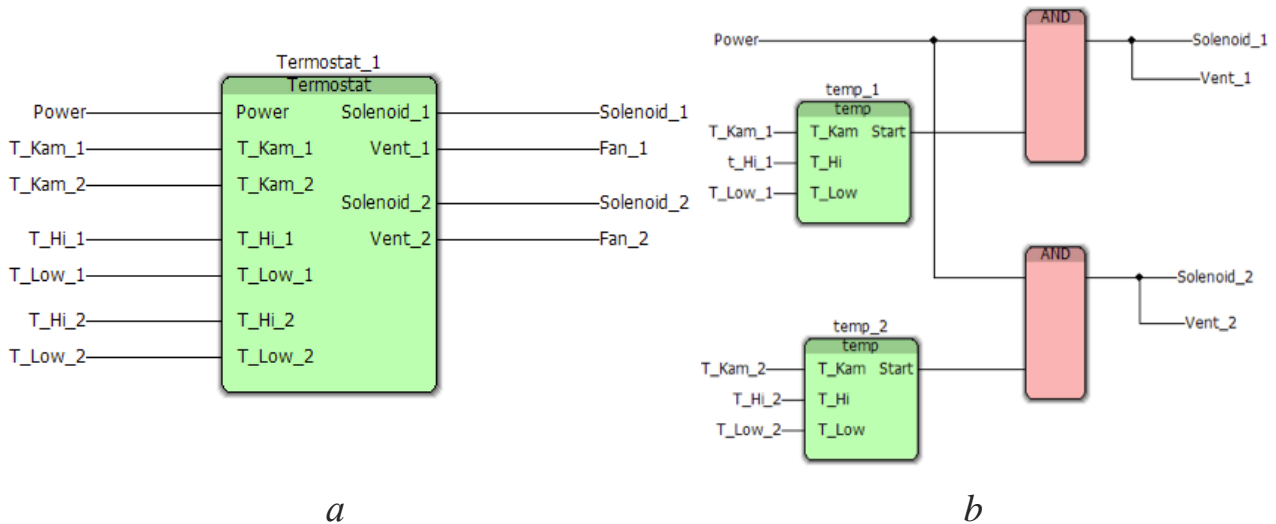


Fig. 5.11. Thermostat: *a* – thermostat function block; *b* – block program code in FBD language

The input signals for the "Thermostat" block are the air temperature in the chambers T_Kam_1 and T_Kam_2, as well as the power supply signal to the power circuit Power.

The output signals are the signals for opening the solenoid of the first chamber Solenoid_1 and the solenoid of the second chamber Solenoid_2. Simultaneously with the opening of the solenoids, commands are issued for starting the fans of the first and second chambers (Fan_1 and Fan_2, respectively).

The values of the actuation temperature T_Hi_1 and T_Hi_2, as well as the cut-off temperature T_Low_1 and T_Low_2 (respectively for chamber No. 1 and chamber No. 2) are used as the thermostat settings.

The algorithm of the block operation is the following: when the power is supplied to the power circuit, the thermostat will continuously compare the current temperature in the chamber with the settings. If the temperature in the chamber is higher than the set point T_Hi, the control signal for opening the solenoid valve and starting the fans in the corresponding provision storeroom are generated by the unit. When the temperature in the chamber falls below the set point T_Low, the thermostat closes the solenoid valve and stops the fans.

According to the FBD operation algorithm of refrigeration unit, a functional unit is created that starts and stops the compressor in manual and automatic modes (Fig. 5.12).

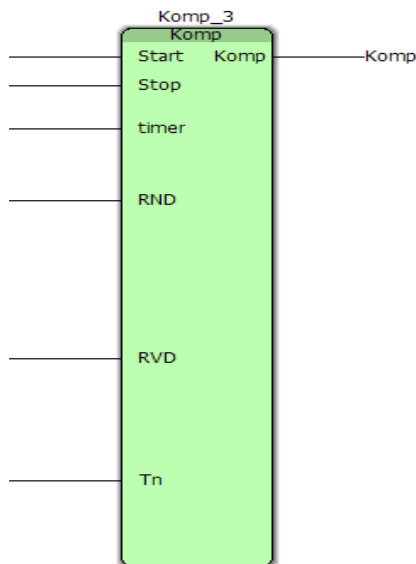


Fig. 5.12. Compressor start-stop block

The program code of this block is shown in Fig. 5.13.

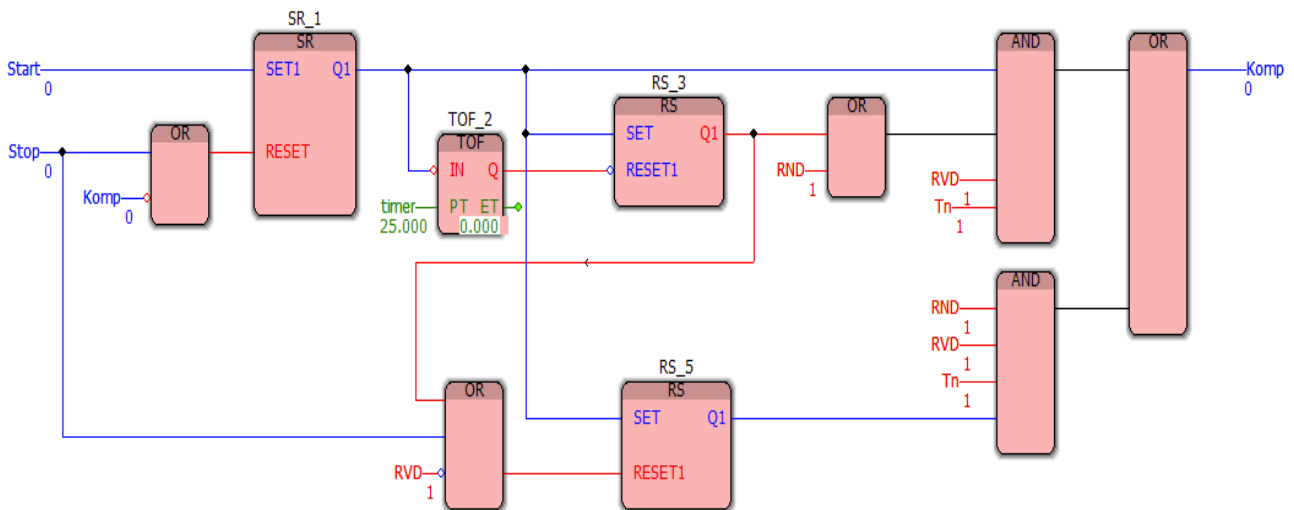


Fig. 5.13. The program code of the compressor start-stop unit

Input data for the operation of the unit are Start and Stop signals of the compressor, timer setting for the temporary shutoff in case of low suction pressure, as well as protecting signal in case of low RND suction pressure, high RVD discharge pressure and high refrigerant vapor pressure on the compressor discharge side Tn. At the output of the block a signal is generated to start the compressor Komp.

The program of the unit performs the following algorithm: after receiving the command to start the compressor (by pressing the "Start" button), the low-pressure switch delay timer is started for 25 s and the prelaunch check of the compressor protection from high temperature and high discharge pressure is carried out. Provided that the protection parameters are within the permissible limits, the

compressor starting signal is sent to the Komp output, which closes the contacts of the compressor drive starter. 25 seconds after start-up the compressor low-pressure protection is automatically activated.

Also, this program implements the start-stop algorithm of the compressor by the LPS signal. When the suction pressure drops below the cut-off setting RND_Low, the LPS will turn off the compressor. If the suction pressure rises to the RND_Hi setting, the LPS will activate the compressor again.

The LPS delay timer is only activated if the compressor is started manually.

The compressor protection circuit is an integral part of the compressor start-stop circuit (Fig. 5.14).

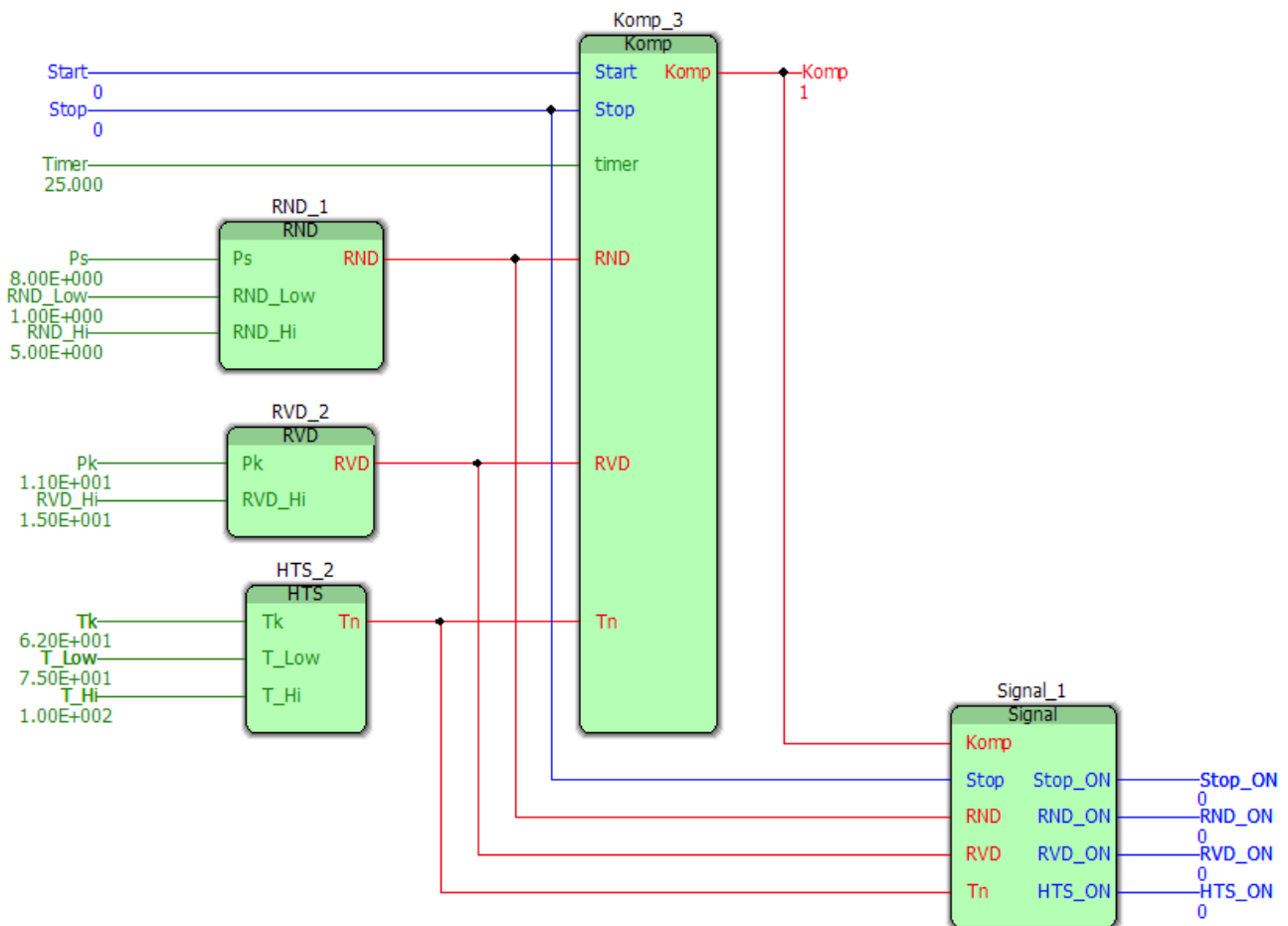


Fig. 5.14. The program of the compressor start-stop circuit in conjunction with the protection and signaling circuit: RND_1 – LPS; RVD_2 – HPS; HTS_2 – HTR; Komp_3 – compressor start-stop block; Signal_1 – indication block of compressor shutdown cause

The protection system includes LPS, HPS and HTR. The main task of these relays is to prevent the compressor from operating at hazardous conditions and to prevent an emergency situation. The task of the relay is to immediately stop the compressor unit when these conditions are reached.

The start-stop program of the compressor (Fig. 5.14) is designed in such a way that when high-pressure protection is activated, the compressor stops and its subsequent start-up will only be possible in manual mode.

When low-pressure protection is triggered, the compressor stops and automatically starts up when the suction pressure rises above the set value.

High discharge temperature protection stops the compressor when the discharge temperature is above the temperature allowable for normal operation of the compressor T_Hi. When the temperature drops to the normal value T_Low, the compressor automatically starts.

The signaling for the cause of the compressor stop is carried out by the "Signal" function block (Fig. 5.15). The block provides an indication of the reason for stopping the compressor when the following conditions occur and then displays this information on the operator panel and turns on the alarm:

- the "Stop" button was pressed;
- LPS was triggered;
- HPS was triggered;
- HTR was triggered.

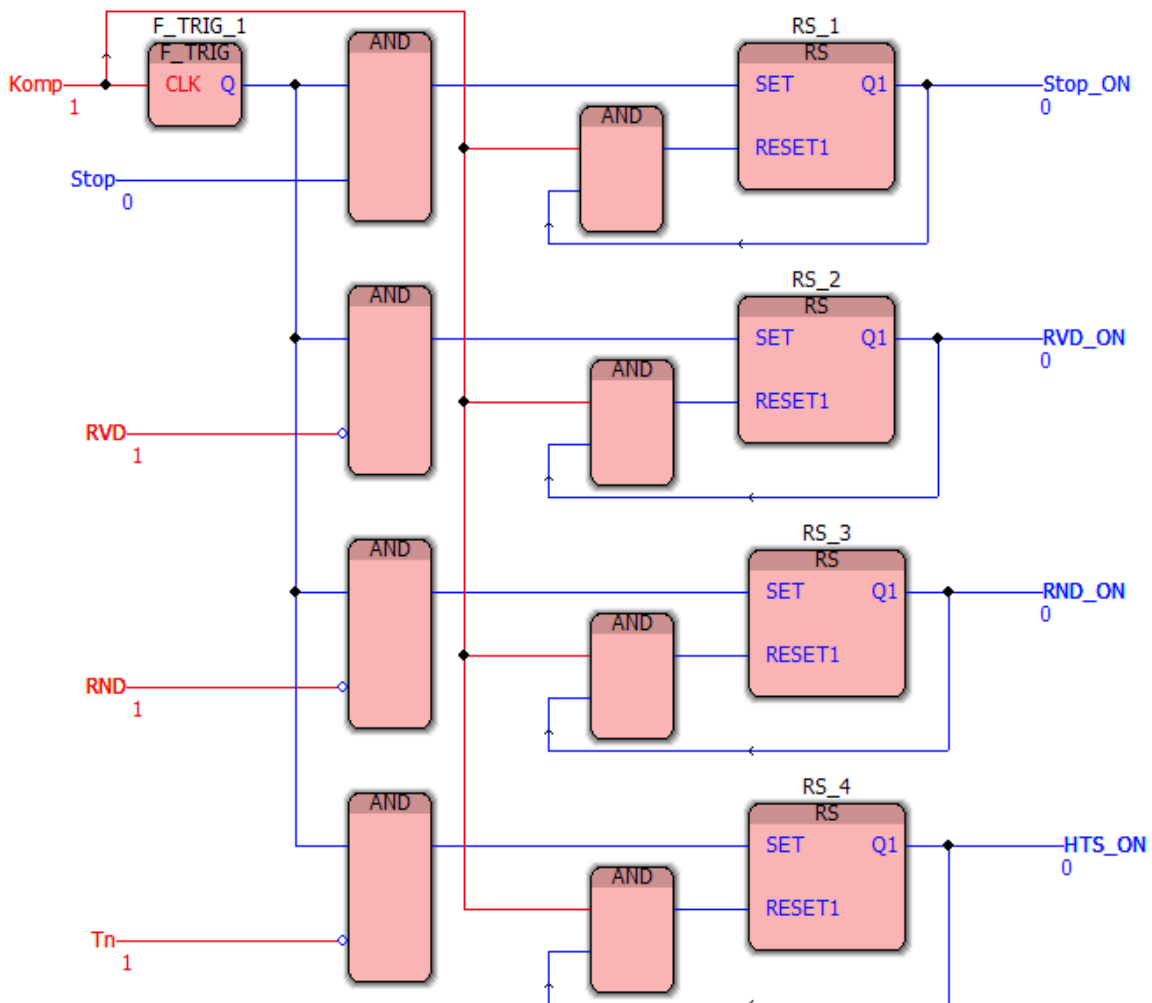


Fig. 5.15. Program code of the "Signal" block

Visu+ has a graphical interface for intuitive and simple control of the refrigeration unit. It provides for information exchange between the controller and the operator.

At first, the operator is given the opportunity to enter all the necessary settings (Fig. 5.16) for the proper functioning of protection system. The list of these settings includes: LPS actuation pressure; LPS cut-off pressure; HPS actuation pressure; HPR actuation temperature; HTR cut-off temperature.

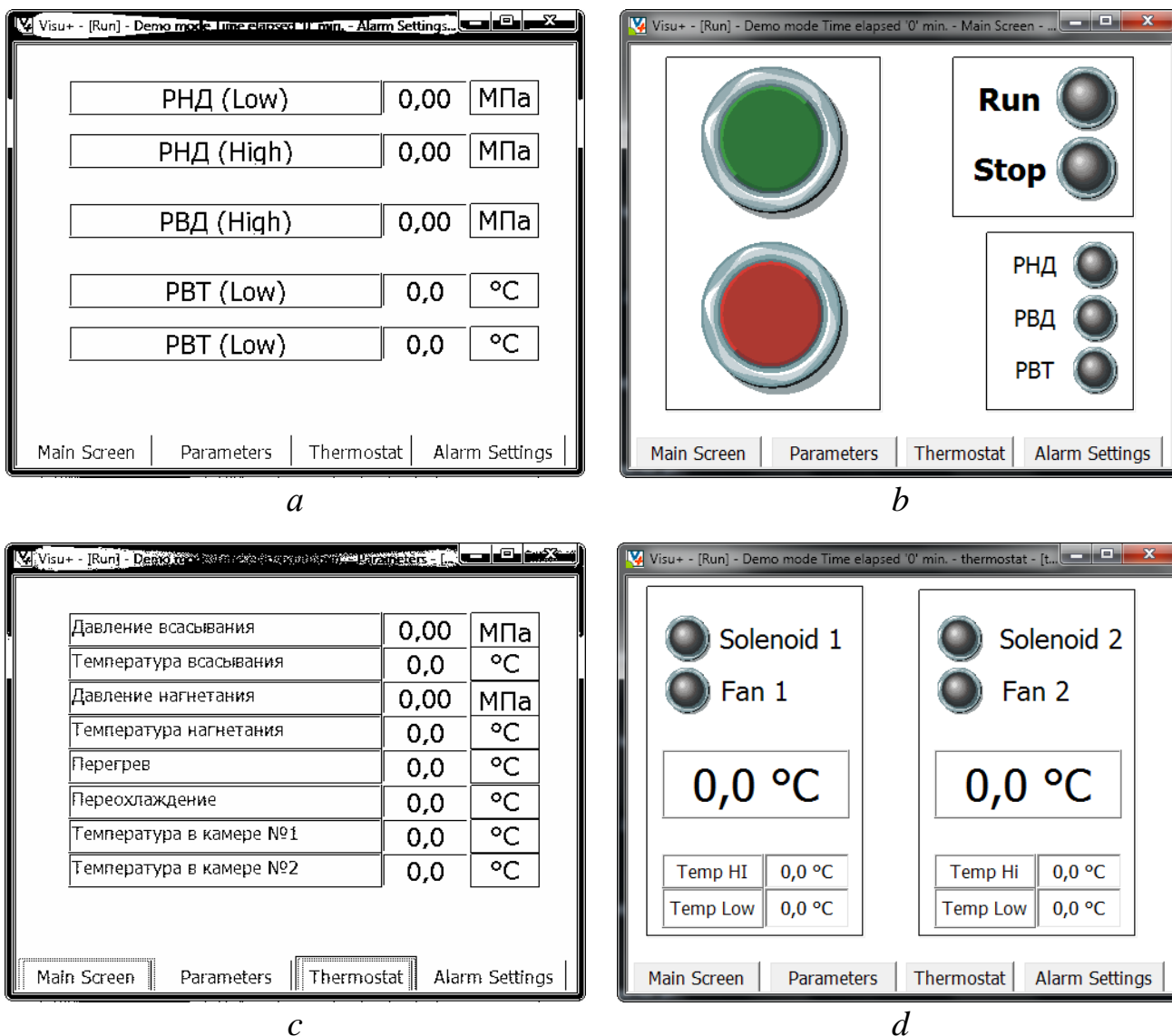


Fig. 5.16. Graphical interface: *a* – window for entering the settings of protection system; *b* – main window of graphical interface; *c* – window for indicating the parameters of refrigeration system; *d* – thermostat operation window

The page for entering protection settings has limited access in order to protect it from unauthorized changes.

In addition to entering the required settings, the operator is able to start and stop the compressor in manual mode, using the graphical interface (Fig. 5.16, *b*). In the same window information is displayed on the current state of the compressor operation, as well as the reason for stopping it (manually by the operator or when the protection is triggered).

The display of the suction pressure and temperature, the compressor discharge pressure and temperature, the superheat and supercooling of the refrigerant is carried out in real time (Fig. 5.16, *c*).

In addition, the current temperature parameters in the provision storerooms, information on the position of the solenoid valves and the operation of the fans of the air coolers (Fig. 5.16, *d*) are displayed on the screen. Also, the operator can set the temperature values that must be maintained in each provision storerooms.

As a result, using the obtained processing data with the help of the CoolPack software package, in the coordinates i -lg*P* the loops of the refrigerating unit are built (Fig. 5.17). The analysis of the graphs proved the advantage of using a programmable controller, since the results of its operation are the closest to the calculated (ideal) values.

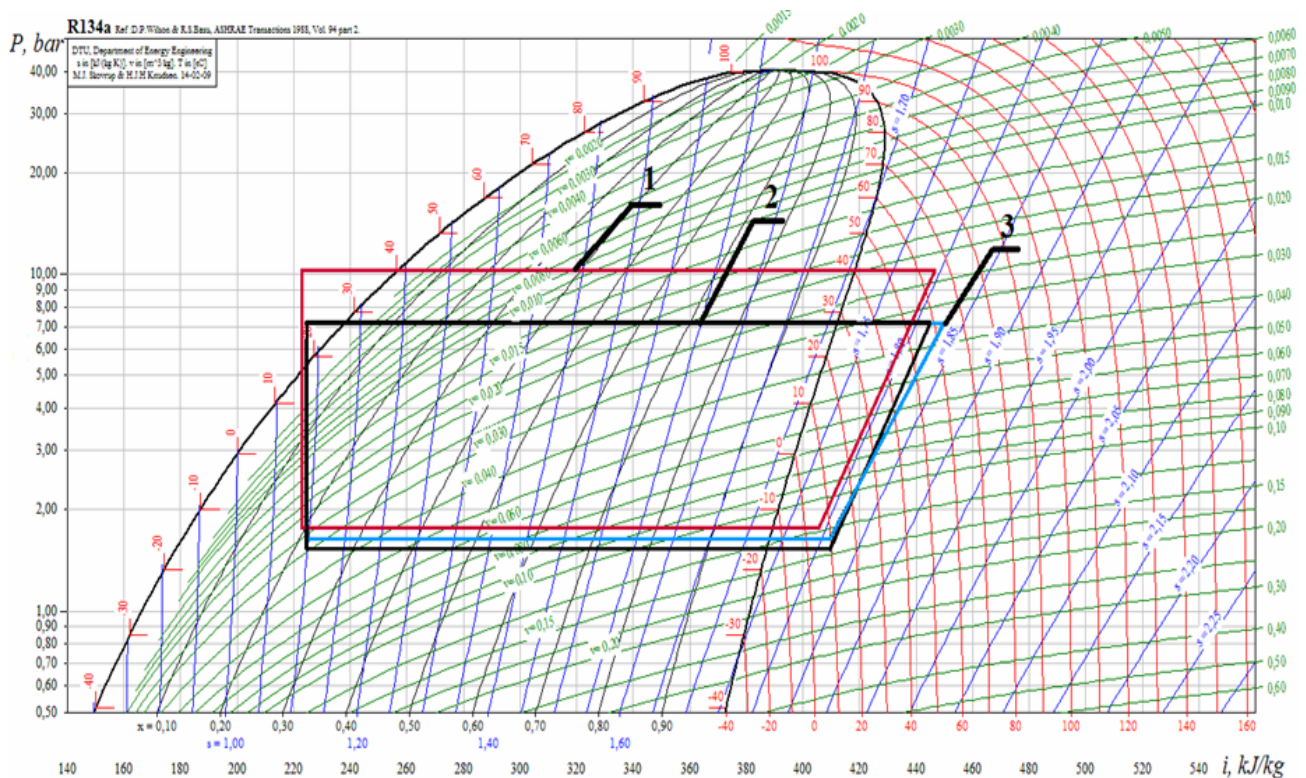


Fig. 5.17. Refrigeration unit loops: 1 – loop calculated by the system on the basis of a personal computer; 2 – theoretical loop; 3 – loop calculated by the system based on a programmable controller

5.3. Scarecrow for airports and gardens

The project was designed taking into account the dangers posed by birds in airports and in the fields. Fig. 5.18 shows the block diagram of the scarecrow control system and Fig. 5.19 is the schematic model including linear motion motors, signal tower, base platform with motion sensors, ground drive and limit switches.

Fig. 5.20 shows the connection diagram of limit switches to the programmable controller.

For signaling about the triggered sector and the current status of the Scarecrow, the visualization panel WP 04T is used. Program segment is shown in Fig. 5.21.

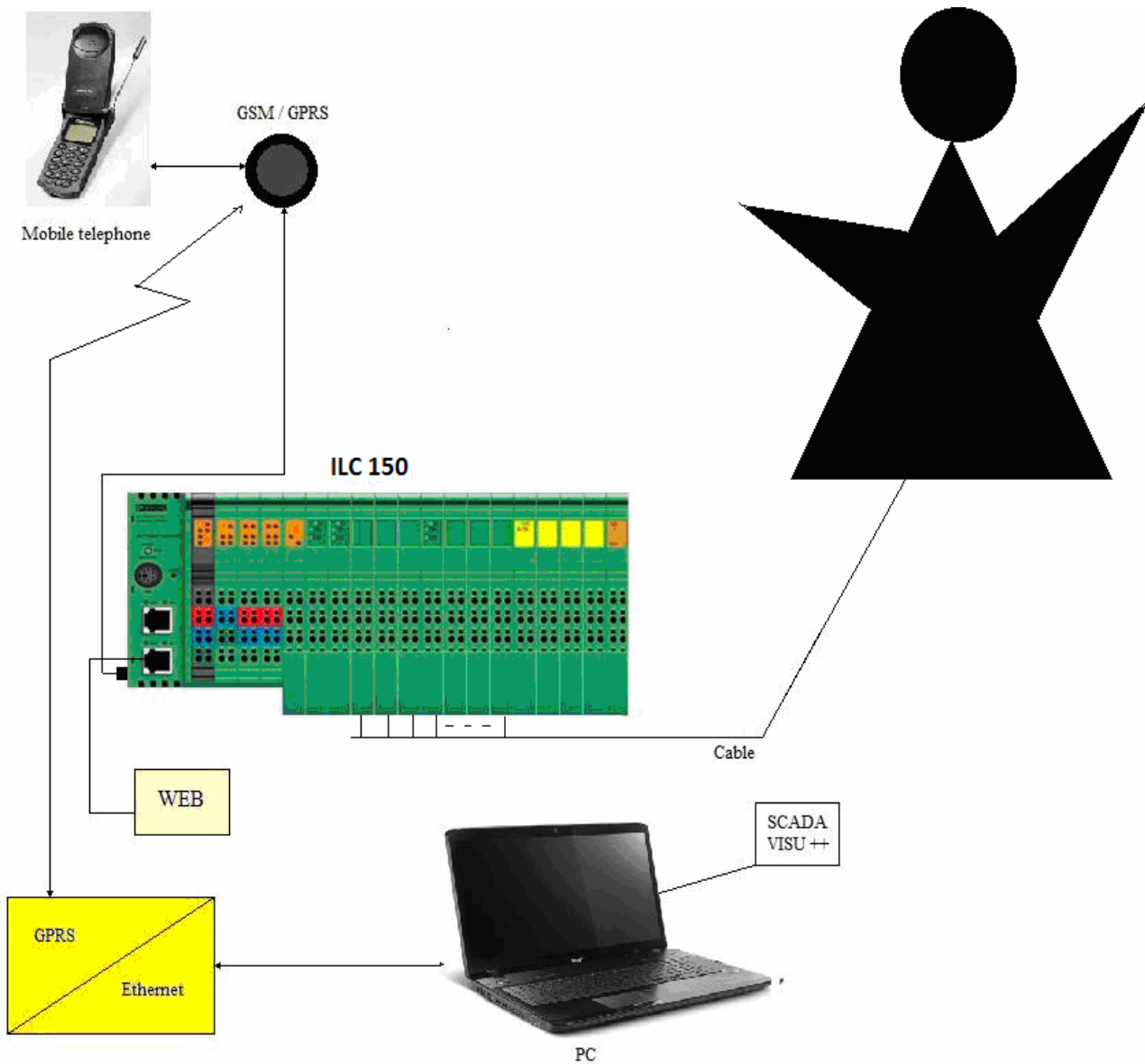


Fig. 5.18. Schematic block diagram of the scarecrow control system

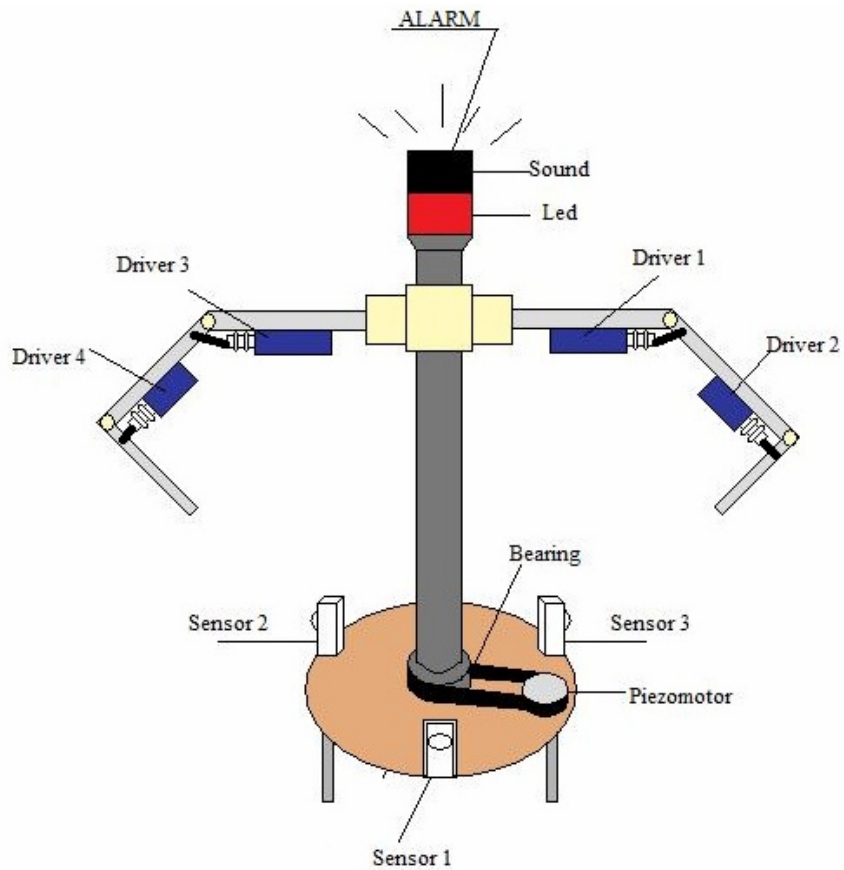


Fig. 5.19. Schematic model of scarecrow

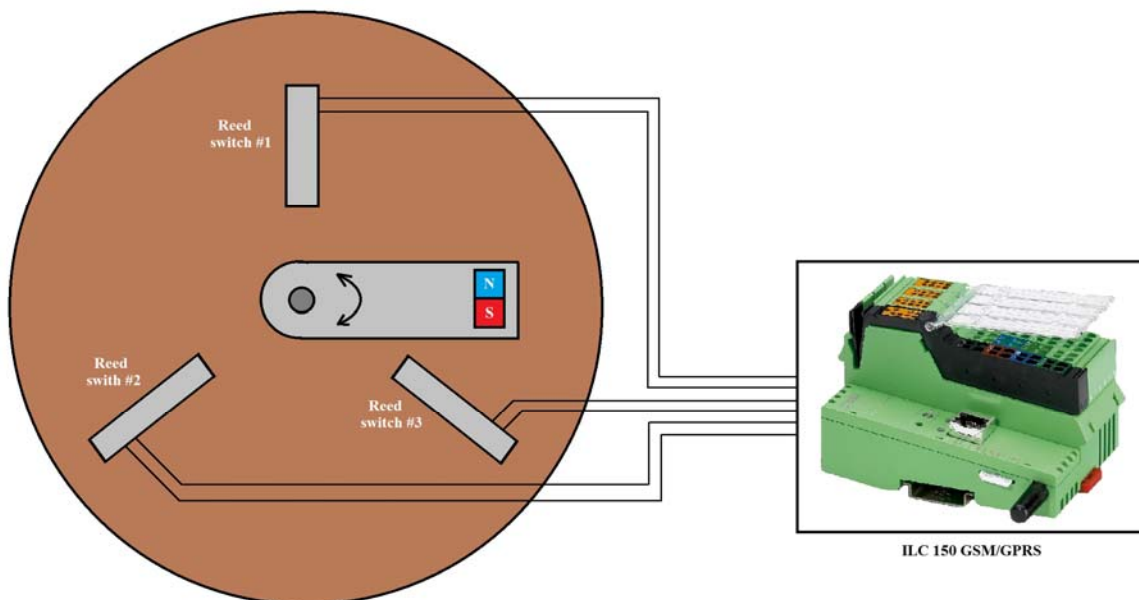


Fig. 5.20. Connection diagram of limit switches

Fig. 5.22 shows the operating program of the "Scarecrow". Since ILC 150 GSM/GPRS controller was used, this object was controlled using a mobile phone via SMS messages. For this purpose, Mobile_Connect and SMS_Receive blocks were used in the program.

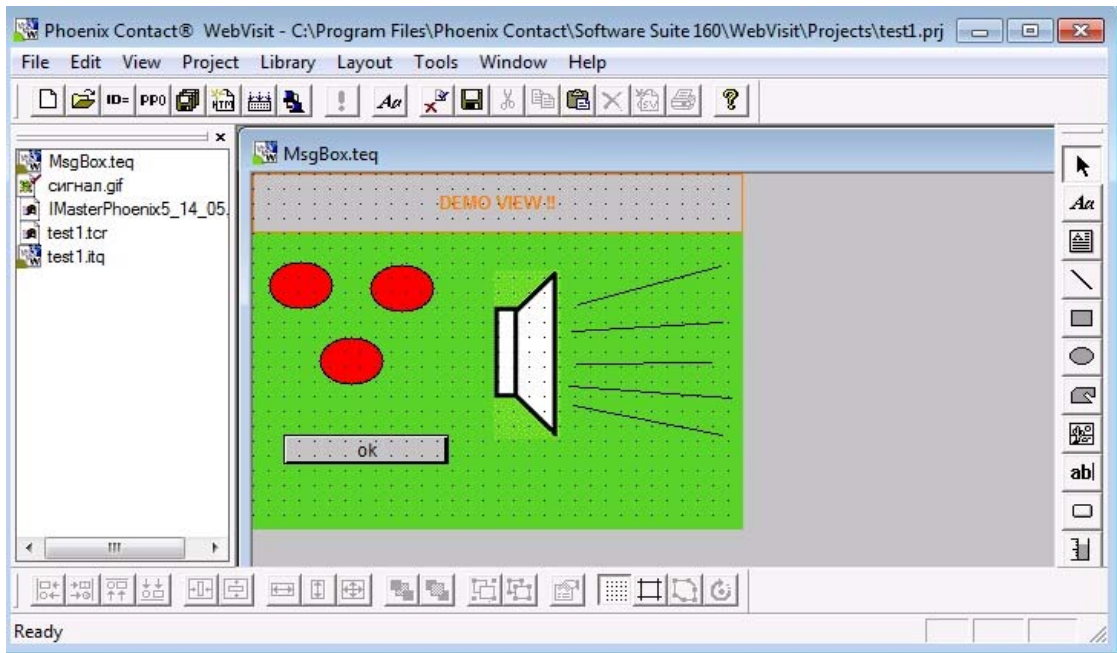


Fig. 5.21. Alarm program in the WP 04T panel created in WebVisit

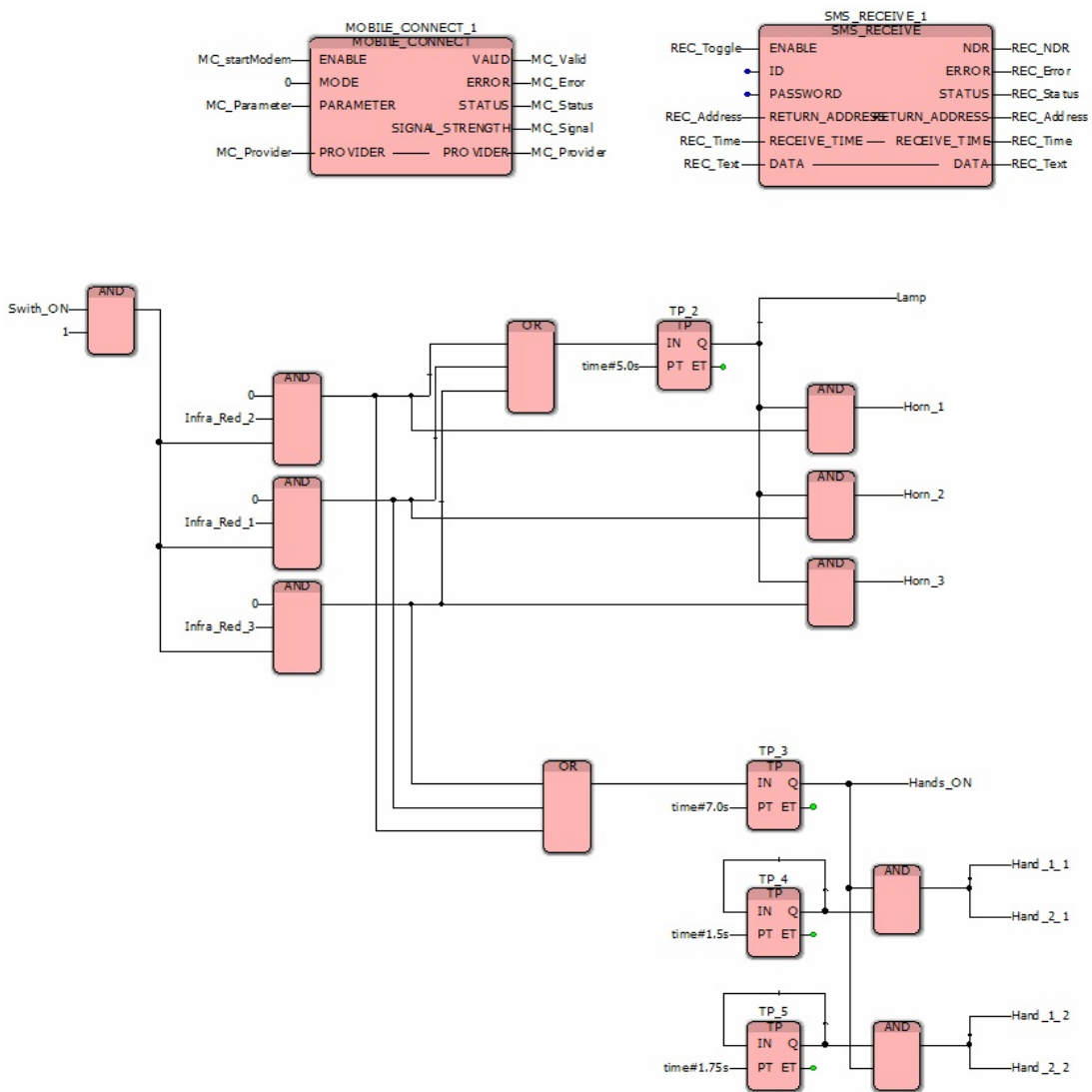


Fig. 5.23. Program created with FBD language in PC Worx

CONCLUSION



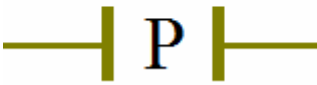

The material presented in the training manual cannot fully cover all the issues of programming controllers, as the development tools and hardware components are dynamically developing. Some models are used for less than a year. Therefore, the main focus of the training manual is on conceptual provisions that should allow further independent learning of programming.

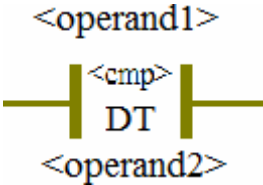
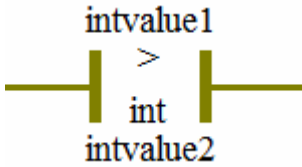
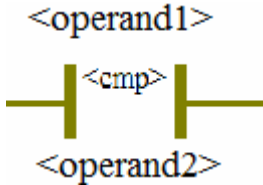
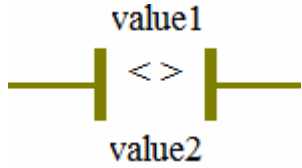
Along with the examples of control system projects given in Section 5 it is recommended to get acquainted with the software solutions described in papers [2 – 4], including the implementation of the transport delay link, fan speed control and alarm system construction.







It should also be noted that the emerging trend of unification of project development tools facilitates their learning, and therefore the transition from one tool to another should not cause great difficulties.

ANNEX

Elements of the LD language according to IEC 61131-3

Description and Symbol	Explanation
<i>Static contacts</i>	
Normally open contact <div style="text-align: center;"> ***  </div>	The state of the left link is copied to the right link if the state of the associated Boolean variable (indicated by "****") is ON. Otherwise, the state of the right link is OFF
Normally closed contact <div style="text-align: center;"> ***  </div>	The state of the left link is copied to the right link if the state of the associated Boolean variable is OFF. Otherwise, the state of the right link is OFF
<i>Transition-sensing contacts</i>	
Positive transition-sensing contact <div style="text-align: center;"> ***  </div>	The state of the right link is ON from one evaluation of this element to the next when a transition of the associated variable from OFF to ON is sensed at the same time that the state of the left link is ON. The state of the right link shall be OFF at all other times
Negative transition-sensing contact <div style="text-align: center;"> ***  </div>	The state of the right link is ON from one evaluation of this element to the next when a transition of the associated variable from ON to OFF is sensed at the same time that the state of the left link is ON. The state of the right link shall be OFF at all other times

Description and Symbol	Explanation
<p>Compare contact (typed)</p> 	<p>The state of the right link is ON from one evaluation of this element to the next when the left link is ON and the <cmp> result of the operands 1 and 2 is true. The state of the right link shall be OFF otherwise. < cmp> may be substituted by one of the compare functions that are valid for the given data type. DT is the data type of both given operands.</p> <p>Example:</p>  <p>If the left link is ON and (intvalue1 > intvalue2) the right link switches to ON. Both intvalue1 and intvalue2 are of the data type INT</p>
<p>Compare contact, (overloaded)</p> 	<p>The state of the right link is ON from one evaluation of this element to the next when the left link is ON and the <cmp> result of the operands 1 and 2 is true. The state of the right link shall be OFF otherwise. <cmp> may be substituted by one of the compare functions that are valid for the operands data type. The rules defined in 6.6.1.7 shall apply.</p> <p>Example:</p>  <p>If the left link is ON and (value1 <> value2) the right link switches to ON</p>

Description and Symbol	Explanation
<i>Momentary coils</i>	
Coil 	The state of the left link is copied to the associated Boolean variable and to the right link
Negated coil 	The state of the left link is copied to the right link. The inverse of the state of the left link is copied to the associated Boolean variable, that is, if the state of the left link is OFF, then the state of the associated variable is ON, and vice versa
<i>Latched coils</i>	
Set (latch) coil 	The associated Boolean variable is set to the ON state when the left link is in the ON state, and remains set until reset by a RESET coil
Reset (unlatch) coil 	The associated Boolean variable is reset to the OFF state when the left link is in the ON state, and remains reset until set by a SET coil
<i>Transition-sensing coils</i>	
Positive transition-sensing coil 	The state of the associated Boolean variable is ON from one evaluation of this element to the next when a transition of the left link from OFF to ON is sensed. The state of the left link is always copied to the right link
Negative transition-sensing coil 	The state of the associated Boolean variable is ON from one evaluation of this element to the next when a transition of the left link from ON to OFF is sensed. The state of the left link is always copied to the right link

List of references

1. PC WORX 6. IEC 61131-Programming. – Blomberg: Phoenix Contact GmbH & Co. KG, 2010. – 442 p.

2. Бурцев А.Г. Программное обеспечение систем управления. Выполнение семестровой (контрольной) работы: методические указания. – 15 с. // Сборник "Методические указания". – Вып. 3. – Волжский: ВПИ (филиал) ВолгГТУ, 2016; [Электронный ресурс] CD-ROM.

3. Бурцев А.Г. Программное обеспечение систем управления. Лабораторный практикум: методические указания. – Часть 2. – 29 с. // Сборник "Методические указания". – Вып. 3. – Волжский: ВПИ (филиал) ВолгГТУ, 2016; [Электронный ресурс] CD-ROM.

4. Бурцев А.Г., Севастьянов Б.Г. Программная реализация технологической сигнализации на промышленных контроллерах Phoenix Contact: методические указания. – 12 с. // Сборник "Методические указания". – Вып. 3. – Волжский: ВПИ (филиал) ВолгГТУ, 2016; [Электронный ресурс] CD-ROM.

5. Деменков Н.П. Языки программирования промышленных контроллеров: учебное пособие /Под ред. К.А. Пупкова. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2004. – 172 с.

6. Никольский В.В., Очеретяный Ю.А., Танасийчук М.И. Мониторинг судовой холодильной техники с использованием программируемых логических контроллеров (ПЛК) // Судовые энергетические установки: научн.-техн. сб. – 2014. – Вып. 34. – Одесса: ОНМА. – С. 41 – 52.

7. Петров И.В. Программируемые контроллеры. Стандартные языки и приёмы прикладного программирования /Под ред. проф. В.П. Дьяконова. – М.: СОЛОН-Пресс, 2004. – 256 с.

УДК 001.8 (075.8)

Г67 Горб С. І., Нікольський В. В., Шапо В. Ф., Хнюнін С. Г.
Програмування контролерів в інструментальному середовищі: навчальний посібник. – Одеса: НУ "ОМА", 2017. – 164 с. Англ. мовою.

Програмування контролерів розглянуто з використанням лабораторних стендів, які розроблені групою університетів у рамках проекту TEMPUS 544010-TEMPUS-1-2013-1-DE-TEMPUS-JPHES – "Trainings in Automation Technologies for Ukraine" (TATU). Ці стенди відрізняються повним набором перспективних технологій, що використовуються в системах автоматизації, модульністю побудови, мінімальним набором технічних засобів для організації комплексного навчання використанню контролерів, адаптацією обладнання для організації навчального процесу.

У якості інструментального середовища обрано середовище розробки проектів PC Work. Наведені приклади рішення задач автоматизації технологічних процесів різної складності.

Призначений для підготовки бакалаврів, фахівців і магістрів галузей знань "інформаційні технології", "механічна інженерія", "електрична інженерія", "автоматизація та приладобудування", "транспорт", а також післядипломної освіти інженерів, що забезпечують проектування і експлуатацію систем автоматизації (у тому числі суднових).

Gorb S.I., Nikolskyi V.V., Shapo V.F., Khniunin S.H. Programming controllers in the integrated development environment: training manual. Practice

Published in the author's edition within the framework of the project TEMPUS 544010-TEMPUS-1-2013-1-DE-TEMPUS-JPHES – "Trainings in Automation Technologies for Ukraine" (TATU)

Cover Design Trubitsyn A.A.

Підписано до друку 11.05.2017. Друк цифровий. Папір офсетний. Гарнітура Times New Roman. Формат 60x84/16. Ум. друк. арк. 9,53. Тираж 300. Замовлення № И17-06-13

Національний університет "Одеська морська академія". Свідоцтво ДК №1292 від 20.03.2003. 65029, м. Одеса, вул. Дідріхсона, 8
Тел./факс: +38 (0482) 34-14-12
publish-r@onma.edu.ua