NUCET NUCLEAR ENERGY AND TECHNOLOGY

Nuclear Energy and Technology 4(4): 223–228
DOI 10.3897/nucet.4.31868

Research Article

Verification on application program generation and loading for safety systems of nuclear power plants based on the reverse engineering method*

Mikhail A. Belonosov[1], Vladimir L. Kishkin[1,2], Sergey A. Korolev[2]

1 *FSUE VNIIA 22 Sushchevskaya str., Moscow, 127055 Russia*

2 *NRNU MEPhI, 31 Kashirskoe shosse, Moscow, 115409 Russia*

Corresponding author: *Mikhail A. Belonosov* (mbelonosov@vniia.ru)

Academic editor: *Georgy Tikhomirov* ♦ **Received** 15 August 2018 ♦ **Accepted** 17 November 2018 ♦ **Published** 13 December 2018

Citation: Belonosov MA, Kishkin VL, Korolev SA (2018) Verification on application program generation and loading for safety systems of nuclear power plants based on the reverse engineering method. Nuclear Energy and Technology 4(4): 223–228. https://doi.org/10.3897/nucet.4.31868

Abstract

The article describes an automated verification method used for application software of control safety systems based on the TPTS-SB equipment. Verification is performed by comparing two mathematical models (oriented graphs): one obtained by processing the original design data, i.e., graphical functional diagrams, and the other formed by reversing the program code loaded from the controller. The vertices in both graphs are functional blocks of mathematical and logical operations; the edges are connections between them. The constructed mathematical models undergo a comparison, covering the vertices and edges of the graphs as well as the memory cells and values of constants. The equivalence of mathematical models proves the correspondence between the program code and the initial set of design functional diagrams.

The proposed automated verification method makes it possible to prove that no distortion is introduced into the program during the process of converting graphical functional diagrams into the program code with its subsequent translation and loading into the controller. It is postulated that any distortions will be detected during the verification procedure, which is performed every time after loading the code into the controller.

The solution provides an acceptable speed when large volumes of vector graphics stored in a relational database are processed, and makes it possible to visualize the verification results. The proposed method is implemented in the GET-R1 instrumentation tools for TPTS-SB and is used in designing and verifying the application software of the safety systems at the Belarusian NPP.

Keywords

Verification; reverse engineering; code generation; safety systems; controller; mathematical model; instrumentation tools

Introduction

Specialists of the FSUE VNIIA have developed a technological software/hardware complex (TPTS-SB) for digital control safety systems (CSS) of NPP instrumentation and control (I&C) systems. These systems are assigned the most important task of ensuring nuclear safety at power units during beyond-design-basis accidents; therefo-

* Russian text published: Izvestiya vuzov. Yadernaya Energetika (ISSN 0204-3327), 2018, n. 2, pp. 146–156.

re, they are subject to the most stringent requirements for software diversity, reliability and correctness. The TPTS-SB software/hardware products were developed to meet all modern requirements for such systems.

The application programs of TPTS-SB-based control safety systems are created as graphical functional diagrams by means of the GET-R1 tool environment (Belonosov et al. 2015) in the problem-oriented language (Standard IEC61131 2003, Zyubin 2005). Man-made graphical control algorithms are tested for compliance with the design requirements for application programs for the TPTS-SB equipment; then the program code is automatically generated in the problem-oriented language and translated into a binary representation (byte-code). The translated byte-code is loaded into a controller.

The article describes a standard procedure for verification of an application program after it is loaded into the controller, including reading from the controller and reverse engineering from the byte-code into a graphical representation of the algorithm.

It is postulated that, in the process of converting graphical algorithms into the program code, translating and loading the code into the controller, there are no hidden distortions in the program: any distortion will be detected by means of the proposed verification procedure.

TPTS-SB system architecture and programming principles

TPTS-SB is a software/hardware system with built-in hardware/software variety designed to build NPP digital control safety systems. The built-in variety is achieved due to division by two independent different implementations: Diversities A and B, which duplicate each other, are functionally equivalent, but differ in hardware and software. The arrangement and architecture of TPTS-SB-based software/hardware systems are described in detail in (Timohin et al. 2015, Naritz et al. 2015).

Fig. 1 shows the structure of an integrated TPTS-SB-based CSS.

Logical processing in Diversities A/B is performed by programmable automation processor modules (APM). These controller modules read cyclically the data from input modules, execute a user program using a special interpreter, and supply signals to output modules and priority control modules.

To program the APMs, the STEP-S programming language is used, which includes logical and arithmetic instructions as well as complex technological ones, such as integration, signal limiting, voting, etc. A program written in the STEP-S programming language is a strictly linear sequence of instructions that does not contain cycles and transitions. Each instruction contains a strictly defined set of arguments and presents an operator of the form

CMD opd1 ... opdN … value1 … valueN

where CMD is the alphanumeric sequence indicating the instruction; opd (operand or marker) is the symbolic address of the APM memory cell; value is the numeric or symbolic constant.

To be loaded to the APM, a STEP-S program is translated into a binary representation (byte-code) while the program structure remains unchanged.

Initial process of developing and loading application programs

STEP-S programs are the result of processing a large number of control algorithms that are developed graphically in the language of functional diagrams (Standard IEC61131 2003, Zyubin 2005). These diagrams consist of functional blocks denoting arithmetic, logical, or complex technological operations. The inputs and outputs of the functional blocks are interconnected; thus, the functional diagram is a graphically depicted sequence of calculations. Each function block is preassigned with a patterned sequence of STEP-S instructions containing at least one instruction.

As an example, Fig. 2 shows a functional diagram of processing signals from seismic sensors by voting, taking into account the reliability of the signals and issuing an emergency protection signal. Table 1 shows a fragment of the resulting program – the generated sequence of instructions in the STEP-S language corresponding to the functional diagram.

Each functional block shown in the diagram is converted into a sequence of one or more instructions. The code is generated in four stages.

1. APM memory cells (markers) necessary for calculations and storing the results are estimated and assigned.
2. The order of calculations is determined. The place where the instructions corresponding to a specific functional block enter the resulting program is determined by this block sequence number which is automatically calculated by topological sorting of an acyclic oriented graph – a mathematical model corresponding to the APM functional diagrams (Tarjan 1971). The numbers of functional blocks in Fig. 3 and the numbers of STEP-S instructions in Table 1 fit together.
3. The design data integrity and correctness as well as compliance with the formal design rules for the TPTS-SB platform are automatically checked. In case of errors, generation is terminated.
4. The sequence of functional blocks is processed as determined at Stage 2. Template instructions corresponding to each functional block are inserted into the resulting program, and previously calculated memory cells and values are entered.

All the generation stages are performed automatically. The resulting program undergoes the translation stage, and then is loaded into the APM of TPTS-SB in the form of byte-code.

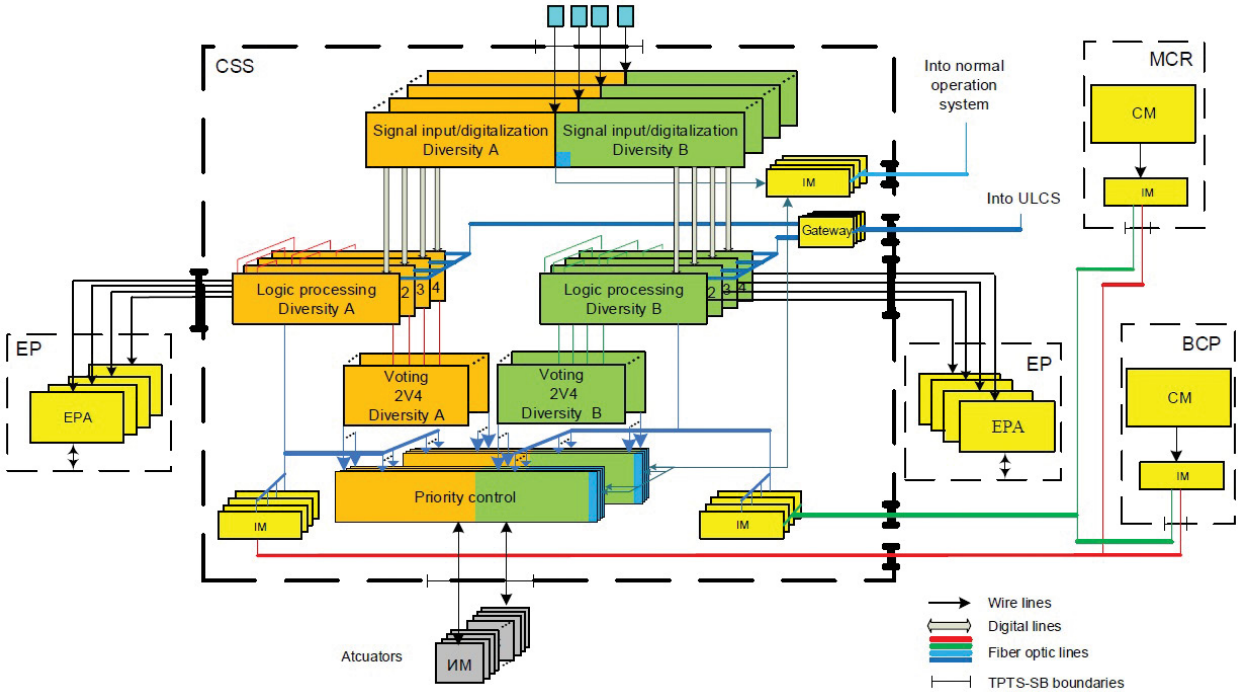

Figure 1. Integrated TPTS-SB-based CSS: EP – emergency protection; MCR – main control room; BCP – backup control panel; ULCS – upper level control system; CSS – control safety system; EPA – emergency protection automatics; IM – interface module; CM – communication module.

Reverse engineering of a program into graphical or tabular representation

When designing NPP control safety systems, it is necessary to prove the correctness of all stages of code generation, translation and loading as well as the absence of distortions in the resulting program as compared to the graphics algorithm. To do this directly is extremely difficult for the following reasons:

- rigorous proof of the correctness of all programs operating at different code generation stages is a labor-intensive process;
- there is no guarantee that the generation process will not fail, which will affect the data integrity. A failure can also be caused by external factors, such as data transmission errors, and internal factors, such as hidden program errors, data read/write errors, etc.

Therefore, the only acceptable way to prove the correctness of the program is to develop a procedure for reverse engineering of the generated and loaded program into a graphical or tabular representation and to compare the restored data with the original project. At the same time, a graphical representation is a general functional diagram, which includes all functional blocks from which the APM program is generated and connections between them; a tabular representation is a list of all functional blocks and their parameters in a tabulated form. However, this proof is also associated with certain difficulties. The program generated in the STEP-S language does not have the following data:

- information on whether the STEP-S instructions belong to a specific control algorithm;
- graphic information necessary for dividing the graphics into diagrams and determining the coordinates of functional blocks;
- designations of algorithms, inscriptions and decoding necessary for users to understand the algorithm;
- information about the graphic implementation of functional blocks (several icons can correspond to the same functional unit).

Table 1. Fragment of the STEP-S application program.

No.	STEP-S instruction
1	NOT M,133 M,141
2	2/4-FS ET,4,9 ET,2,10 ET,5,11 M,140 M,145 M,146
3	NOT M,133 M,142
4	AND-3 M,150 M,141 M,135 M,152
5	2/4-FS2 M,138 ET,4,9 ET,2,10 ET,5,11 M,153 M,154
6	NOT M,135 M,143
7	AND-3 M,133 M,153 M,143 M,155
8	2/4 M,138 ET,4,9 ET,2,10 ET,5,11 M,147
9	B-DFLT M,147 M,148 M,149 1 1
10	NOT M,135 M,144
11	AND-3 M,142 M,156 M,144 M,158
12	B-LADK M,143 0
13	2/4-FS M,138 ET,4,9 ET,2,10 ET,5,11 M,156 M,157
14	OR-4 M,158 M,155 M,152 M,146 M,159
15	B-DFLT M,147 M,150 M,151 1 1

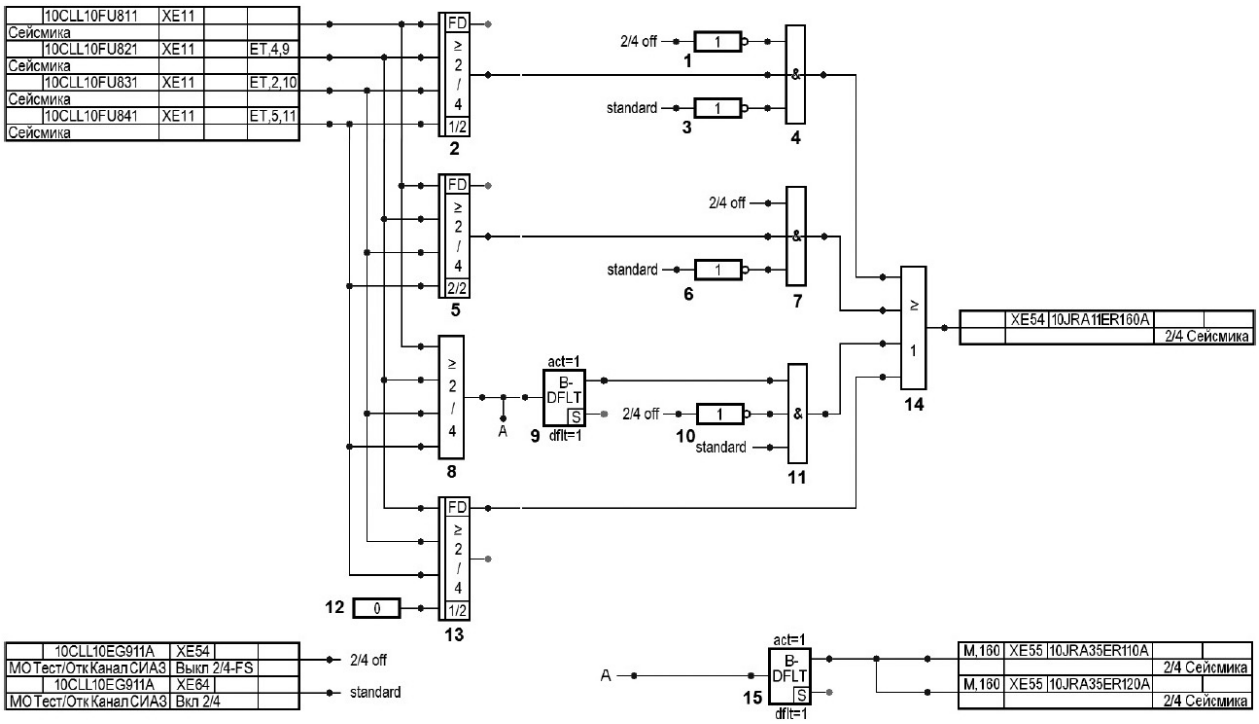

Figure 2. Functional diagram of processing signals from seismic sensors.

Therefore, it is impossible to completely restore the original graphical representation of the control algorithms using the program code without involving the project information.

However, to prove the correctness of the code generation and loading procedures, it is not required to completely restore the graphics. Moreover, with large project volumes, these procedures will take an unreasonably long time. Instead of full restoration, it is proposed to construct mathematical models of the project in the form of graphs oriented to different source data: one graph is constructed according to the project; the other is constructed according to the program code. The vertices in both graphs are functional blocks of mathematical and logical operations, while the edges are connections between them. The constructed mathematical models undergo a comparison, covering the vertices and edges of the graphs as well as the memory cells and values of constants. The equivalence of mathematical models is the proof of the correspondence between the program code and the initial set of project functional diagrams. Fig. 3 shows a mathematical model for the above functional diagram.

The reverse engineering procedure is implemented independently of the code loading, translation, and generation programs. Restoration is performed in six stages.

1. The byte-code is read from the APM and back-translated into a string representation in the STEP-S language.
2. A list of STEP-S instructions is created for the functional blocks from the library and this list is sorted by the number of instructions in each block.
3. A search is carried out for template sequences of the functional blocks in the program code and the program is split into corresponding fragments. Each fragment of the STEP-S instructions is replaced by the corresponding functional block.
4. The memory cells (markers) and constants from the code are inserted into the inputs and outputs of the functional blocks.
5. The connections between the functional blocks are determined by the correspondence of the memory cells (if the same marker is assigned to the input of one block and the output of the other, then these input and output are connected). At the end of this stage, there is already a full-fledged oriented graph constructed by the program code.
6. An identical oriented graph is constructed (Filatova 2012) according to the project (database). At the same time, the sequence of the functional blocks and connections between them is already stored in the database. The remaining project information is ignored.

The result of the comparison of these two oriented graphs can be visualized in a tabular form or in the form of a general functional diagram, which includes all the APM algorithms. When the comparison results are visualized in the form of a general functional diagram, it is considered that different graphical representations of the same function, for example, the vertical and horizontal image of the AND2 function (conjunction of two boolean values), are invariant with each other. Therefore, in comparison, it is reasonable to use a graphical version of the blocks, which is the most suitable from a visual point of view. The functional diagrams generated by the code and the project of the same versions will be identical.

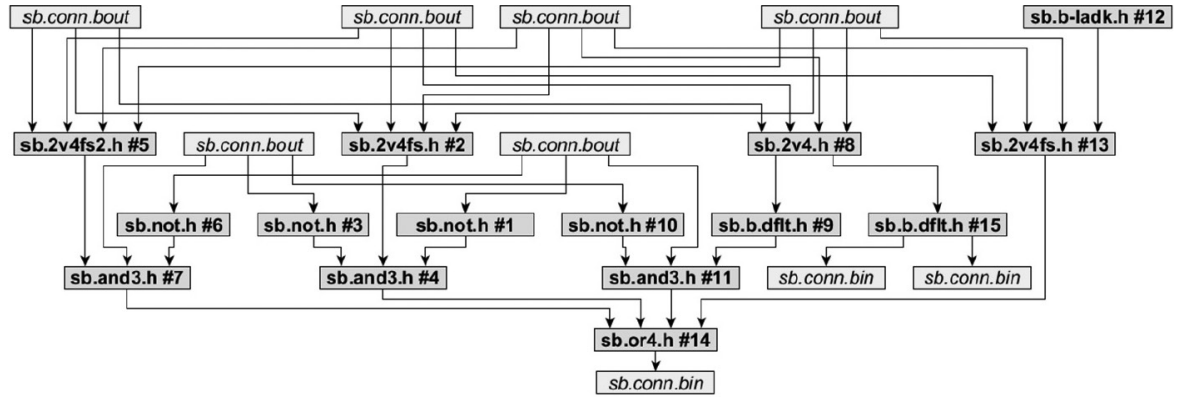

Figure 3. Mathematical model graph corresponding to the functional diagram in Fig. 2 The vertices *sb.conn.bout* correspond to the input signals of the functional diagram, the vertices *sb.conn.bin* correspond to the output signals. The remaining vertices correspond to the functional blocks and have a sequence number.

Advantages in comparison with known solutions

There is a known procedure for verifying the code generator of the SPACE tools (Miedl 1996) for the TELEPERM XS (AREVA) software/hardware package. The program that performs verification is called Retrans and was developed by the staff of the Institut fur Sicherheitstechnologie (ISTec). Retrans is an independent procedure for a project reverse conversion from the C code to a software representation with a subsequent comparison. In the restoration process, Retrans uses both the program code itself and additional project information contained in the generated source code as comments.

The solution applied to TELEPERM XS does not verify the code translation and loading stages where errors may occur. Verification of the generated C code proves only the correctness of the generation process, but cannot serve as proof of the identity of the project and the compiled program loaded into the controller.

The solution proposed for the TPTS-SB equipment is more complete, since the entire processing chain is verified (including generation, translation, and loading). In this case, the project mathematical model is restored completely independently without any project information involved. In addition, the use of graph models makes it possible to clearly visualize the comparison results using the hierarchical embedding algorithm of graphs (Baburin 2018, Sponemann et al. 2009). The probability that two independent errors will occur in the code generator and in the reverse conversion procedure at the same time, with one of which hiding the other, is negligible. Therefore, this solution can be considered as strictly proving the correctness of automated conversion of control algorithms into the object code of an application program provided that the reverse conversion procedure is performed each time after the code is generated and loaded.

Conclusion

The authors propose a method for complete verification of controller programs through reverse engineering. The method can be used for verifying application programs of any controller programmed by means of graphical languages of the IEC 61131-3 standard.

The GET-R1 tools for TPTS include a software component that implements this method. Within these tools, the proposed verification method is used in developing the application software for the TPTS-SB-based control safety systems of the Belarusian NPP-2.

To completely guarantee the code correctness, the TPTS-SB equipment allows for the code loading with a delayed start, which makes it possible to first load the program, then perform the reverse conversion procedure and only after that start a new program.

The main design solutions for I&C systems of modern Russian NPPs are presented in (Zverkov 2017, Bozhenkov 2009, Zverkov 2015, Dunaev and Korolev 2011, Zverkov 2014, Yastrebenetsky 2011); the verification methods for software/hardware complexes of NPP TPTS-based I&C systems are described in (Korolev et al. 2017, Korolev et al. 2016); the modern international safety requirements for software/hardware complexes of NPP are contained in the IAEA/IEC documents (IAEA SSR-2/1 2012, IEC 61513-2002 2002, IAEA NS-G-1.1 2000)

References

- Baburin DE (2018) Hierarchical approach for automatic allocation of acyclic graphs. Available at: http://www.iis.nsk.su/files/articles/sbor_kas_09_baburin.pdf [Accessed Feb 2, 2018] [In Russian]
- Belonosov MA, Galitsyn YS, Krayushkin UV, Zhukov IM, Gritsenko SY (2015) The end-to-end engineering tools for instrumentation

and control systems for nuclear power plants. Reports of BSUIR, 2(88): 47–51. [In Russian]

- Bozhenkov OL (2009) System engineering of the automated process control system of NPPs. Yadernye izmeritelno-informacionnye tehnologii [Nuclear Engineering and Information Technology], 2: 27–30. [In Russian]
- Yastrebenetsky MA (2011) Control systems and protection of nuclear reactors. Ser. Safety of Nuclear Power Plants. (Ed.) Kiev. Osnova-Print Publ., 770 pp. [In Russian]
- Dunaev VG, Korolev SA (2011) PCS of power units of nuclear power plants with VVER. In: Nuclear Power. Problems. Solutions. Part 1. (Ed.) Strikhanov MN. Moscow: TsSPiM Publ.: 315–356. [In Russian]
- Filatova NN (2012) Structural synthesis of automation schemes in conditions of incomplete requirements for technical implementation. Izvestiya VolGTU [Bulletin of VolSTU], 4(13): 17–22. [In Russian]
- IAEA NS-G-1.1 (2000) The software of control systems, important for safety, executed on the basis of computer equipment. Safety Guide. Vienna. IAEA.
- IAEA SSR-2/1 (2012) Safety of nuclear power plants: design. Specific safety requirements. Vienna. IAEA.
- IEC 61513-2002 (2002) Nuclear power plants. Monitoring and control systems important for safety. General requirements.
- Korolev S, Tolokonsky A, Rogov V (2017) The optimal approach for the processes of verification and validation of NPP software and hardware complexes. Journal of Physics: Conference Series, 781(1): 82–89. https://doi.org/10.1088/1742-6596/781/1/012048
- Korolev SA, Tolokonsky AO, Rogov VV (2016) Modern methods of verification of software and hardware complexes of automated process control systems of nuclear power plants based on TPTS. Elektricheskie stantsii [Electric Power Plants], 8: 9–15. [In Russian]
- Miedl H (1996) Retrans – a tool to verify the functional equivalence of automatically generated source code with its specification. Probabilistic Safety Assessment and Management (PSAM-III). Crete, Greece: 137–147.
- Naritz AD, Moiseev MI, Novikov AN, Karpov PS, Borzenko AA (2015) The complex of automation system TPTS-SB. Reports of BSUIR, 2(88): 38–42. [In Russian]
- Sponemann M, Hanxleden R, Fuhrmann DI (2009) On the automatic layout of data flow diagrams. Arbeit.
- Standard IEC61131 (2003) International Electrotechnical Comission, 3, 226 pp.
- Tarjan R (1971) Depth-first search and linear graph algorithms. 12th Annu. Symp. Switch. Autom. Theory (SWAT 1971), 1 (2): 146–160. https://doi.org/10.1109/SWAT.1971.10 [In Russian]
- Timohin DS, Gritsenko SYu, Artemyev KP (2015) The structure of the automated process control system of the Belarusian NPP in terms of safety. Reports of BSUIR, 2 (88): 28–32. [In Russian]
- Zverkov VV (2014) Automated control system for technological processes of NPPs. Moscow. MEPhI Publ., 558 pp. [In Russian]
- Zverkov VV (2015) Analysis of approaches to the construction of automated process control systems of NPPs. Elektricheskie stantsii [Electric Power Plants], 8: 2–6. [In Russian]
- Zverkov VV (2017) Program-Technical Complexes of Control Systems for Safety of Nuclear Power Plants. Elektricheskie stantsii, 1: 2–10. [In Russian]
- Zyubin VE (2005) PLC Programming: IEC 61131-3 languages and possible alternatives. Promyshlennye ASU i kontrollery [Industrial ACS and Controllers], 11: 31–35. [In Russian]