Modeling and Programming of Elements of Integrated Systems in Industrial Controller Languages

Mykhailo Poliakov[1], Bohdan Zhurakovskyi[2], Oleksii Poliakov[2] and Ivan Shyshkin[1]

[1] *National University "Zaporizhzhia Polytechnic", Zaporizhzhia, Ukraine.*
[2] *National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute" Kyiv, Ukraine*

Abstract

Context. Building integrated systems based on industrial controllers is a promising direction for creating adaptive control systems. The problems in this area are the limited functionality of operating and control automata models, the lack of a methodology for their software implementation as part of typical elements of an integrated system (TEIS).

Objective. The goal of the work is to expand the functionality of TEIS, to create a methodology for their programming in the languages of the IEC 61131-3 standard. The scientific novelty of the work lies in the fact that new types of controls for the structure and parameters of operational and control automata of an integrated system are proposed, such as control of the initial state, blocking of specified states, control of the output parameters of the control automaton and the parameters of operational automata.

Method. The possibilities of changing the parameters of elements of set-theoretic models of TEIS control and operational automata by external control from the overlying control automaton of the hierarchical system are analyzed. A methodology for detailing the control structure to the Program Organization Unit level in the languages of the IEC 61131-3 standard is described.

Experiments. An example of TEIS programming in languages according to the IEC 61131-3 standard in the OpenPLC application environment with subsequent loading and execution of the code on the Arduino microcontroller board is given.

Results. An experiment with a logical controller and a physical board confirmed the functionality of the structural and parametric adaptation of the control algorithm inherent in the TEIS.

Conclusions. The current problem of modeling adaptive behavior in an integrated hierarchical system and programming such behavior using the languages of the IEC 61131-3 standard has been solved. The proposed methods for controlling the parameters of operating and control automata as part of the TEIS expand the functionality of adaptive behavior in the system. The practical value of the proposed methodology for implementing adaptive behavior lies in the possibility of constructing TEIS in the programming languages of industrial controllers.

Keywords

Integrated control system, programming languages for industrial controllers, controlled operating and control machines.

1. Introduction

Programmable logic controllers (PLC) are the main elements of complex industrial automation systems [1, 2]. PLC programming is performed in languages according to the IEC 61131-3 standard [3, 4]. The ability to program in these languages is implemented in the applications OpenPLC [5], ISaGRAF, [6], CoDeSys [7] and others. The program that is loaded into the PLC is a software implementation of the system object control algorithm.

One of the classes of control systems are integrated systems. At least two subsystems in such a system have a common element that performs different functions in these systems [8]. For example, in one system – the function of a control unit, and in the second – a control object. This interaction of subsystems makes it possible to create systems with a control hierarchy, with the help of which the structure and parameters of the system are adapted to changes in the goals of their functioning and the parameters of external influences on the system [9].

2. Problem statement

An element of interaction between subsystems within the control hierarchy is the complex of controlled and control automata. The lack of models and methods for programming such a complex in programming languages according to the IEC 61131-3 standard is an urgent unsolved problem.

Object of research: the process of modeling the functional structure and programming of integrated PLC-based control systems in an application environment that supports languages according to the IEC 61131-3 standard.

Purpose of the work: to develop a methodology for designing standard elements of integrated systems with software implementation of control algorithms in the languages of the IEC 61131 - 3 standard.

3. Review of the literature

The starting point for writing a control program for a control system is the model of the system's control unit. When creating such a model for discrete and hybrid control systems, the formalism of control and operational automata is widely used [2], [10–13].

In the model of a classical control finite automaton there are three sets (inputs, outputs and states), an element of the set of states "initial state" and two functions (transitions and outputs). The elements of these sets are binary. Each element of these sets can take the values "active" or "passive". At each moment of time, only one input, one output and one state are active. The activity of the output coincides in time with the activity of the state. The action in the active state is independent of the input whose activity caused the state to activate. Such a model is simple but modeling complex system control algorithms with its help leads to an increase in the dimension of this model. In addition, the classical model does not provide for structural adaptation of the machine during its operation.

The works [14], [19] describe a model of an automaton with an additional set of controls and a set of functions of the automaton in its states. An element of a set of controls determines the possibility of the automaton exiting the current state under various control options, which makes it possible to change the trajectory of state changes during the operation of the system and thereby implement the structural adaptation of the automaton. An element of the set of functions of an automaton in its state describes the activation function of the state, output functions and structure. However, such an automaton does not form an integrated system.

Models of operating machines are classified by location in the system in relation to the control machine (input and output) and by the type of operation performed [10]. The input operational automaton forms the input (element of the set of inputs) of the control automaton. The logic of the operation performed in this case determines the meaning of the input activity. For example, the activity of the "Temperature above normal" input requires different control actions (outputs) at different standards. Thus, by changing the "norm" parameter of the operating machine, we influence the control logic. The output operational automaton generates an impact on the control object based on the activity of the output (an element of the set of outputs) of the control automaton. The parameters of this influence can serve as an object of control by a higher-level control machine.

The work [8] describes a model of an element of an integrated system, consisting of two interconnected automata CA1 and CA2 in combination with input operational automata IOA1, IOA2 and output operational automata OOA.

The element represents a two-level integrated system. In this system, the automaton CA1 in the lower-level subsystem is the control automaton, and in the upper-level subsystem it is the object of control by the automaton CA2. The automaton CA1 receives events X_1 from the input operational automaton IOA1 and the control option C_1 from the superior automaton CA2. In turn, the automaton CA1 generates control signals Y_{01} for the output operational automaton OOA and information signals Y_{12} for the automaton CA2. Automaton CA2 receives events X_2 from automaton IOA2 and control option C_2 from a higher level of the hierarchical system. In addition, the machine CA2 generates control signals C_1 and information signals Y_{23}. At the same time, work [8] does not reflect the capabilities of controlling operational machines of an element of an integrated system and the issues of implementing this element of an integrated system in the environment of programming applications in languages according to the IEC 61131-3 standard. These languages are Ladder Diagram (LD), Function Block Diagram (FBD), Sequential Function Chart (SFC), Structured Text (ST) and Instruction List (IL) [3].

In this work, the OpenPLC application [5] was used to implement the system control algorithm in software. This is a freely distributed software product that allows programming in all languages according to the IEC 61131-3 standard. The application contains OpenPLC Editor and OpenPLC Runtime components. The OpenPLC Editor component is a programming environment and contains a logic controller with which you can simulate the process of executing a user program. The OpenPLC Runtime component allows you to transfer a program to a PLC, which can be either an industrial controller of the Controllino family or popular microcontroller boards of the Arduino, Raspberry, ESP32, STM32 and others [15] – [18]. To perform the transfer, the PLC must be connected to the computer via a USB port.

4. Materials and methods

By a controlled automaton we mean an automaton CA1, which, with the help of another automaton CA2, can change at least one element of its tuple. For example, block some inputs [8], change the initial state and others. It is possible to change several or even all elements of a tuple at the same time. Such changes, as a rule, will lead to changes in the results of the machine's operation. The automaton CA2, which changes the elements of the tuple of the automaton CA1, will be called the manager. In general, a controlled automaton can receive control from more than one control automaton. Conversely, a control automaton can control more than one automaton.

The control signal generated by the control automaton CA2 can initiate changes in the input operating automata. For example, changes in the thresholds of signals that form events (inputs) of the machine. The purpose of such changes may be the parametric adaptation of the system control device.

The set of machine inputs characterizes the observability of processes in the control object. The point of blocking individual inputs of an automaton in a certain state is to change the number of control cycles performed in the automaton (possible paths in the graph of the automaton), and therefore the control functionality. Changing the initial state of the machine can reduce the time the system reaches a rational control cycle.

The set of outputs of the machine characterizes the controllability of the object. For a lower-level automaton this is a system object, and for other automata it is the underlying operating and control automaton. In binary control automata, control of outputs comes down to blocking/unblocking them, in non-binary ones, also to changing the parameters of influences (level, nomenclature, time interval).

Blocking the activity of the state of the machine is equivalent to blocking all inputs that lead to a transition to this state and blocking actions in it.

Thus, controlling the elements of sets of an automaton tuple changes its functions within wide limits. The subject of such control is the CA2 automaton (Fig. 1). The outputs of this control automaton in its active state determine the control vector of the automaton CA1, including the initial state of CA1, activity in the active state and paths of transition from this state to others. Elements of the control vector can be discrete (allow/block an element of the tuple set of a controlled automaton) or analog types. In the latter case, the value of the control element can be considered, for example, as a threshold for the formation of an input event by an input operational automaton of a controlled automaton, or as a coefficient by which the amplitude of the output signal of the output operational automaton is multiplied, or as the duration of output activity, or as a binary code, in which single outputs serve resolution of individual outputs of a subset of outputs of the controlled automaton in a given state.

The structure of the control automaton CA2 specifies the trajectory for changing the controls of the automaton CA1 within a certain cycle of object control. These trajectories, in turn, can be the object of control at the next level of the control hierarchy in the system. Typical control change trajectories provide expansion/narrowing of functionality, decreasing/increasing the number of stages in control cycles, decreasing/increasing the number of unlocked inputs of the lower-level machine.

The block diagram of the modernized typical element of the integrated system is shown in Fig. 1.

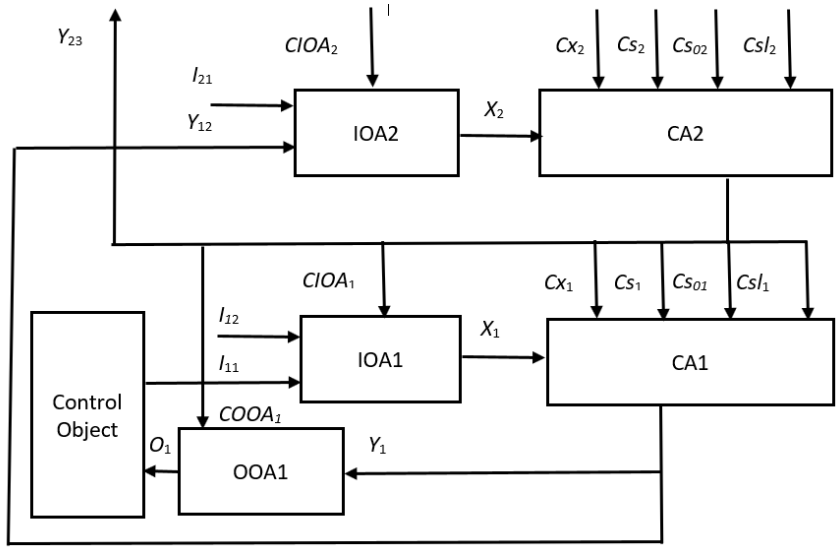


Figure 1. Block diagram of a typical element of an integrated system

The diagram shows control automata of the first (CA1) and second (CA2) levels, input operational automata of the first (IOA1) and second (IOA2) levels, output operational automaton of the first level (OOA1) and a system control object. The system control object responds to influences O_1 by changing parameters I_{11}. The IOA1 automaton uses the parameters of the object I_{11} and the parameters of the external environment I_{12} to form the X_1 inputs of the CA1 automaton. In turn, the automaton, under the influence of inputs X_1, changes its state and generates impact outputs Y_1 for the automaton OOA1 and information outputs Y_{12} for informing the automaton CA2. The *CIOA*1 input controls the parameters of the IOA1 automaton, the *COOA*1 input controls the parameters of the OOA1 automaton. And the inputs C_{x1}, C_{s1}, C_{s01}, C_{sl1} control the structure of the CA1 automaton. All of the listed control signals are generated as outputs of the CA2 machine, which also has similar control from the overlying machine.

The choice of the program organization unit (POU) for implementing TEIS elements in industrial controller programming languages was made based on the following considerations:

1. The IOA inputs receive the results of measurements of object parameters. The IOA machine, as a rule, performs the operation of data conversion, calculation of mathematical expressions, and checking the satisfiability of logical conditions. In this case, the IOA can be implemented as a function or function block in the ST or FBD programming languages. And the control signal *CIOA* will have the meaning of a scaling factor or a logical condition element. For example, for the IOA machine, which generates input x_1 "Temperature is normal", control consists of changing the parameters of this norm.

2. Control automata in the control program are implemented as functional blocks in the SFC language: the state of the automaton corresponds to the steps of the program; machine inputs – transition conditions; Several options for exiting the state correspond to alternative branches.

3. The input control logic C_{x1} is implemented at the "program" POU level. In this case, each input of the control device is supplied to the corresponding input of the machine only if there is permission from the C_{x1} logic, which is implemented using the POU of the functional block.

4. State blocking logic C_{s1} boils down to blocking inputs for all possible transitions to this state.

5. To implement the logic C_{s01} of changing the initial state, additional inputs are added to the control machine to transition from state s_0 to another initial state, which is specified by control C_{s01}.

6. The logic of actions in a state may depend on the input through which the state of the machine was activated. And this logic can be controlled by specifying control of C_{sl1} from the higher-level machine. But this significantly complicates the management program and requires separate consideration.

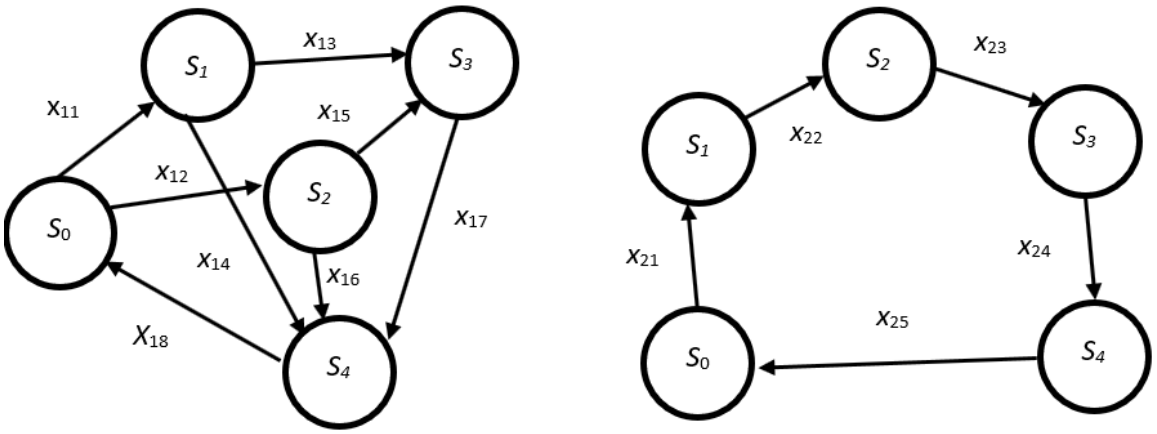
7. The OOA automaton is implemented as a set of actions in each step of the corresponding functional block of the control automaton. These actions in program steps correspond to the outputs of the machine. For example, such as the result of assigning certain numerical values to output variables, which can be interpreted as: scaling factor of the output amplitude for a given state; N – bit binary code that controls the activation (one in the code) of N digital outputs of the control device; action qualifier parameter in this step.

Thus, TEIS elements can be mapped to a set of POUs of the type program, function and functional block and implemented in programming languages of the IEC 61131-3 standard.

5. Experiments

Let's consider the implementation of an element of the integrated system (Fig. 1) using the example of automata CA1 and CA2, specified by the graphs shown in Fig. 2. The automaton CA1 has sets of states' $S = (s_0, s_1, s_2, s_3, s_4)$, inputs $X = (x_{11}, x_{12}, x_{13}, x_{14}, x_{15}, x_{16}, x_{17}, x_{18})$ and outputs $Y = (y_0, y_1, y_2, y_3, y_4)$. Moreover, output y_0 is active in state s_0, output y_1 is active in state s_1, and so on. As can be seen from Fig. 2a, the paths of four cycles of system object control pass through state s_0 in the automaton CA1. This is path 1: s_0-s_1-s_3-s_4-s_0; path 2: s_0-s_1-s_4-s_0; path 3: s_0-s_2-s_3-s_4-s_0 and path 4: s_0-s_2-s_4-s_0.

The control cycle changes when activity passes through state $s0$. The control strategy is to move the activity of the machine in the sequence: path 1 - path 2 - path 3 - path 4. Then the sequence is repeated again. The control strategy is formed by the automaton CA2, in which there is only one path s_0-s_1-s_2-s_3-s_4-s_0. The state change on this path occurs at the moment of completion of the next cycle of the automaton CA1. That is $x_{21} = x_{22} = x_{23} = x_{24} = x_{25} = Change$.



a b

Figure 2 — Graph of controlled CA1 (a) and control CA2 (b) automaton

The task of controlling the automaton CA1 from the side of the automaton CA2 is to limit the inputs of the automaton CA1 so that it implements only one given path in the current control cycle. A graphical FBD representation of an integrated system element program in the OpenPLC application environment is shown in Fig. 3.

The program contains standard functional blocks EQ, R_TRIGO, AND, user functional blocks of types CA1, CA2 and contr. Graphical representations of which are shown in Fig. 4-6.

At the initial moment, both automata are set to state S_0 (variables *stateCA1*=0 and *stateCA2*=0), a pulse arrives at the Change input of automaton CA2, which sets automaton CA2 to state S_1. As a result, the count block generates permissions/prohibitions at inputs $e_1 – e_8$, which correspond to strategy “1”. Allowed events from the set $x_1 – x_8$ arrive at the inputs of the automaton CA1 and cause transitions of this automaton to other states in the control cycle of the system object. At the moment of completion of this cycle, a new impulse is formed at the *Change* input and the described cycle is repeated for strategies “2”, “3”, “4”, “0”, “1”, “2”...

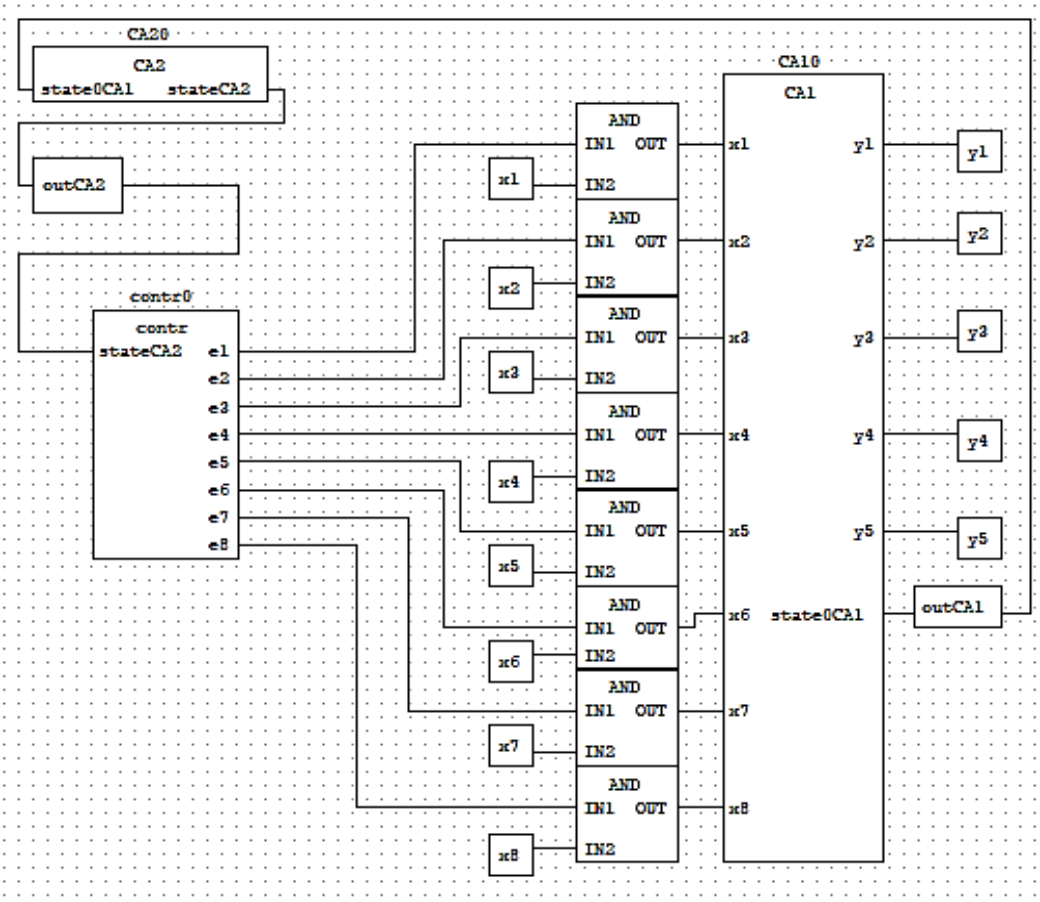


Figure 3 – Graphical FBD representation of the integrated system element program.

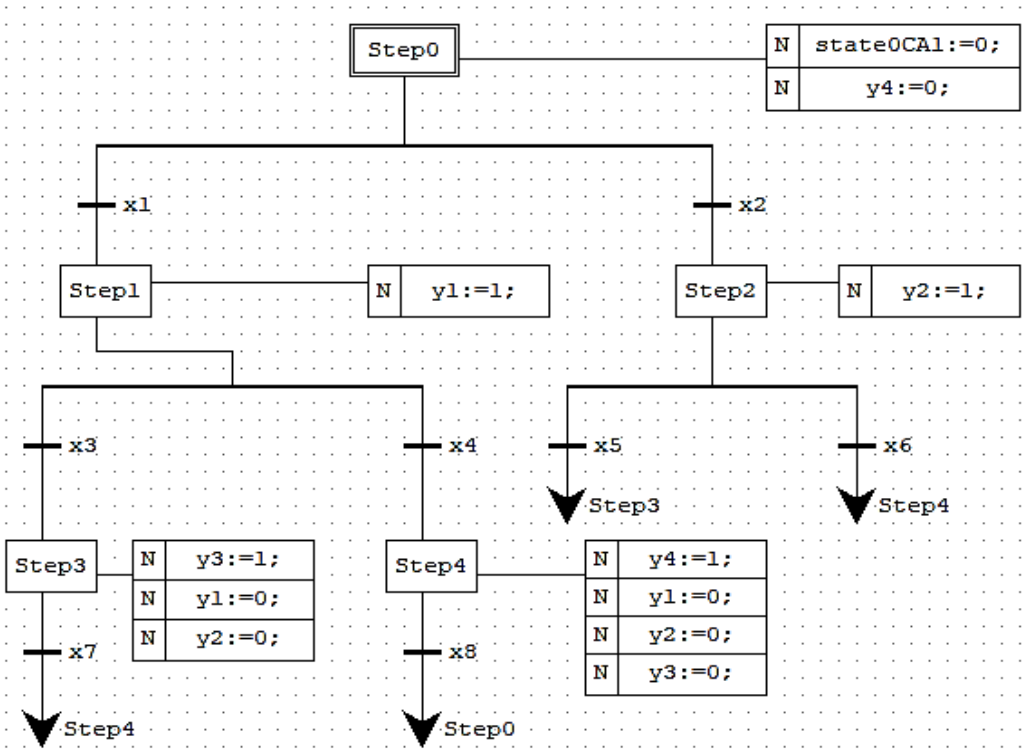


Figure 4 – SFC functional block of the CA1 machine

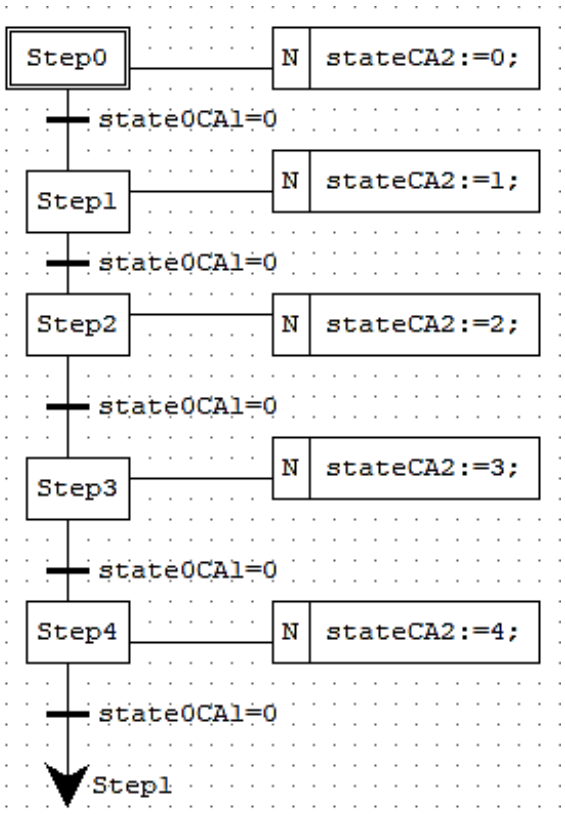


Figure 5 – SFC functional block of the CA2 machine

```
CASE stateCA2 OF
  0: e1:=0; e2:=0; e3:=0; e4:=0; e5:=0; e6:=0; e7:=0; e8:=0;
  1: e1:=1; e2:=0; e3:=1; e4:=0; e5:=0; e6:=0; e7:=1; e8:=1;
  2: e1:=1; e2:=0; e3:=0; e4:=1; e5:=0; e6:=0; e7:=0; e8:=1;
  3: e1:=0; e2:=1; e3:=0; e4:=0; e5:=1; e6:=0; e7:=1; e8:=1;
  4: e1:=0; e2:=1; e3:=0; e4:=0; e5:=0; e6:=1; e7:=0; e8:=1;
  ELSE
    e1:=0; e2:=0; e3:=0; e4:=0; e5:=0; e6:=0; e7:=0; e8:=0;
    END CASE;
```

Figure 6 – ST functional block contr for controlling the input permissions of the CA1 machine

Shown in Fig. 4 SFC functional block of the automaton CA1 describes the structure of transitions and actions in states according to the graph of the automaton in Fig. 2a.

Similar problems are solved by the SFC functional block of the CA2 machine shown in Fig. 5. In this example, the automaton CA2 is uncontrollable, but if another control level appears in the system, then this automaton can also be an object of control from the upper level.

ST functional block contr (Fig. 6) controls the permissions of the inputs of the CA1 machine in the control strategies described above. With the addition of a new strategy, a new element of the CASE statement must be added. For example, if all inputs are enabled, then control can be performed according to any of the cycles described above, depending on the sequence of occurrence of events at inputs x_1 - x_8.

In Fig. Figure 7 shows an example of SFC - a model of a controlled automaton CA2 with two control strategies of the CA2 level.

In the first strategy of level CA2, the change of strategies for controlling the automaton CA1 is carried out in the order: path 1 - path 2 - path 3 - path 4. And in the second strategy of level CA2 - in the reverse order. The first strategy uses the Change input permission, and the second one uses Change1. These signals are generated at the outputs of the CA3 machine of the third level in the system control hierarchy.

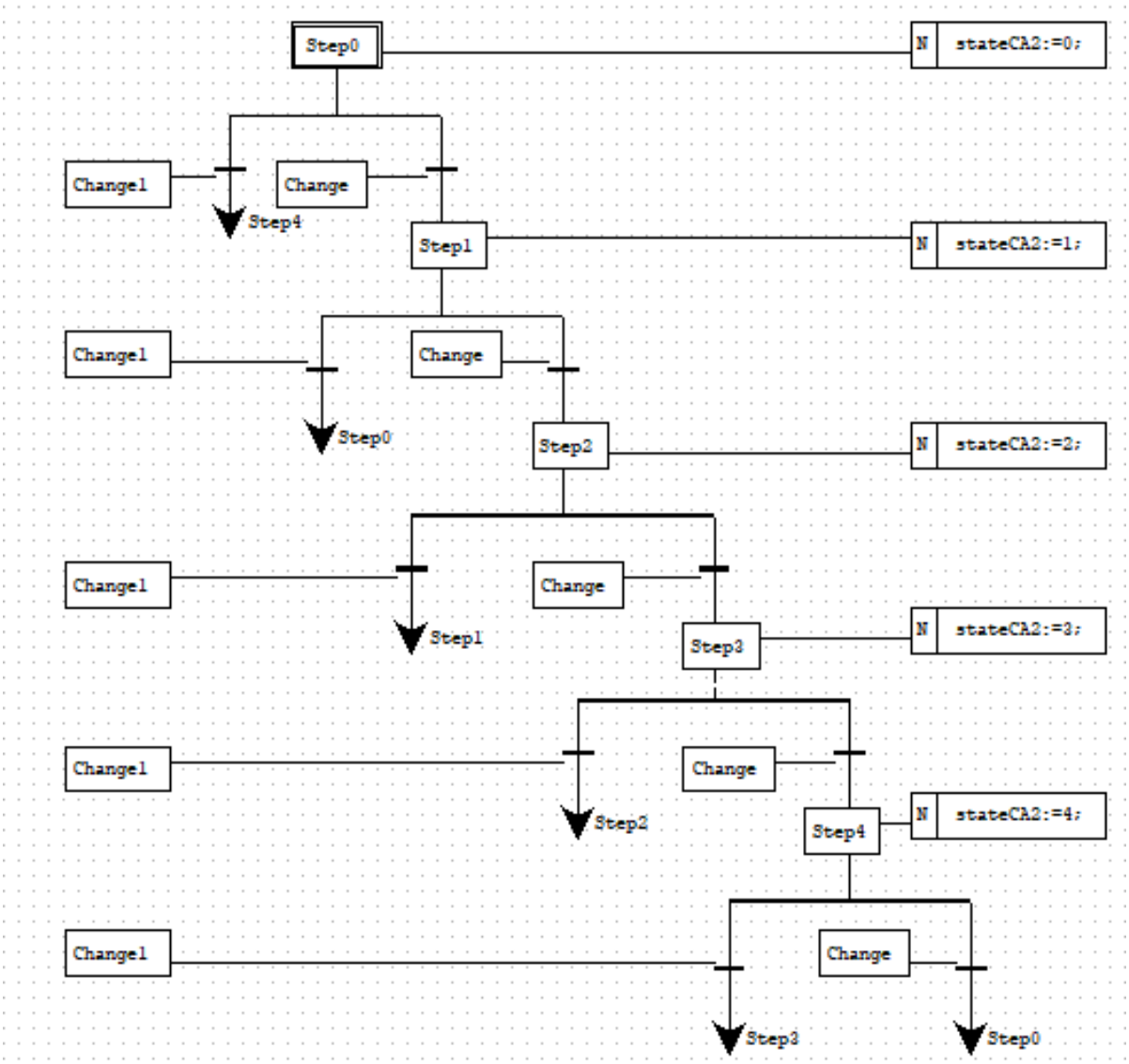


Figure 7 – SFC model of controlled automaton CA2 with two control strategies

6. Results

Testing of the developed project for a typical element of an integrated system was carried out both using the built-in debugger of the OpenPLC Editor application, and by transferring the project code to the Arduino board. In the latter case, the simulation of the inputs x1 - x8 of the CA1 machine was carried out using buttons, and the outputs - using light-emitting diodes. The test results confirmed the functionality of the integrated system declared in the project, such as managing the cycle of actions in the system and managing the typical sequence of changing control cycles.

Thus, the typical structure of an element of a hierarchical integrated system is implemented in programming languages according to the IEC 61131-3 standard in the OpenPLC application environment. This structure can be useful, for example, when forming a sequence of diagnostic procedures that is adapted to the results of previous measurements of the parameters of a system object.

7. Discussion

Modeling adaptive behavior reflects the relationship between two adjacent levels of control in a hierarchical system. The object of adaptation can be almost any element of the tuple of control automata and the parameter of the operation performed by the operating automata of the system.

The model uses expressive means of graphical programming languages for industrial controllers. The POUs of this model have two levels. TEIS operating and control machines are presented at the

lower level in the form of functions or functional blocks. The second level, the program level, displays the logic of the relationship between these elements. Example experiments showed the practical value of the proposed methodology for modeling adaptive control in an integrated hierarchical system.

The graphical representation of the model in the OpenPLC application describes the control logic in the system at the functional level, at the same time allowing translation into executable code and transfer of this code to a wide range of popular microcontroller boards.

Modeling of integrated systems in the OpenPLC application as part of the master's training process allows you to study the programming of industrial controllers using free software and inexpensive popular microcontroller boards.

8. Conclusions

The current problem of modeling adaptive behavior in an integrated hierarchical system and programming such behavior using the languages of the IEC 61131-3 standard has been solved.

The proposed methods for controlling the parameters of operating and control automata as part of the TEIS expand the functionality of adaptive behavior in the system.

The practical value of the proposed methodology for implementing adaptive behavior lies in the possibility of using TEIS implemented in the programming languages of industrial controllers.

The conducted research is limited to modeling the behavior of discrete systems based on binary finite state machines. Prospects for further research include studying the possibility of using the proposed models and methods to build integrated hybrid and cognitive systems using both binary and non-binary automata [20].

9. Acknowledgments

The work was supported by the state budget research work of the National University “Zaporozhzhia Polytechnic” “Intelligent information technologies for data processing” (state registration number 0118U100063).

10.References

[1] Parr, E. A. Programmable Controllers. An engineer’s guide / E. A. Parr. 3rd ed. – Oxford: Newnes, 2003. – 429 p.
[2] Hooper, J. F. Introduction toPLCs. Second Edition. Published By: Carolina Academic Press. 2006. – 120 p.
[3] IEC 61131-3, Revision 3.0, February 2013 - Programmable controllers – Part 3: Programming languages. Published By: International Electrotechnical Commission (IEC). – 468 p.
[4] Karl-Heinz John –IEC 61131-3: Programming Industrial Automation Systems: Concepts and Programming Languages, Requirements for Programming Systems, Aids to Decision & Making Tools. 1995, Springer Verlag.
[5] OpenPLC Overview – Autonomy (autonomylogic.com) [Electronic resource]– Access mode: https://autonomylogic.com/docs/openplc-overview/
[6] ISaGRAF – [Electronic resource]– Access mode: http://www.isagraf.com;
[7] CoDeSys – [Electronic resource]– Access mode:http://www.3s-software.com;
[8] Poliakov, M. O. Interoperability of Integrated Hierarchial Systems / M. O. Poliakov, S. O. Subbotin, O. M. Poliakov // System technologies: regional interuniversity collection of scientific works. – Dnipro, 2021. – No. 2 (133). – pp. 68–78. DOI 10.34185/1562-9945-2-133-2021-08
[9] Poliakov, M. Behavior classification of control unit of systems//Radio Electronics, Computer Science, Control. 2022. № 3, pp. 183-195 doi 10.15588/1607-3274-2022-3-17.
[10] Glushkov, V. M. Sintez tsifrovykh avtomatov. Moscow, Fiz-matizdat, 1962, 476 p.
[11] Hopcroft, J. E. Introduction to automata theory, languages, and computation / by John E. Hopcroft, Rajeev Motwani, Jeffrey D. Ullman. -- 3rd ed. //Addison Wesley, 2006. – 550 p.
[12] Solov’ev V. V., Klimowicz A. S., Structural models of finite-state machines for their implementation on programmable logic devices and systems on chip, Journal of Computer and Systems Sciences International, 2015, Vol. 54, № 2, pp. 230–242. doi.org/10.1134/s1064230715010074

[13] Stéphane Lafortune. Discrete Event Systems: Modeling, Observation, and Control, Annual Review of Control, Robotics, and Autonomous Systems, 2019, No. 2:1, pp. 141– 159. doi.org/10.1146/annurev-control-053018-023659
[14] Polyakov, M. A. Finite automata with non-binary elements of sets / M. A. Polyakov, I. A. Andrias // System technologies: regional inter-university collection of scientific works. – Dnipropetrovsk, 2019. – № 2 (121). – P. 85–94.
[15] Arduino - Home [Electronic resource]– Access mode: https://www.arduino.cc/
[16] Raspberry Pi for industry [Electronic resource]– Access mode: https://www.raspberrypi.com/for-industry/
[17] The Internet of Things with ESP32 [Electronic resource]– Access mode http://esp32.net/
[18] STM32 32-bit Arm Cortex MCUs. Overview. [Electronic resource]– Access mode: https://www.st.com/en/microcontrollers-microprocessors/stm32-32-bit-arm-cortex-mcus.html
[19] Poliakov M., Subbotin S., Poliakov O. Performance indicators of models of non-binary control automates, Experience of Designing and Application of CAD Systems: IEEE 16th International Conference. Lviv, 22–26 Feb. 2021, proceedings, pp. 38–42. http:doi.org/10.1109/CADSM52681.2021.9385220
[20] Poliakov, M., Subbotin S., Poliakov O. Set-theoretical FSM models activity subsystem for Cognitive Control Systems. // In Proceeding of the 15th International Conference ”The Experience of Designing and Application of CAD Systems” (CADSM), (26 February - 2 March, 2019, Polyana-Svalyava (Zakarpattya), Ukraine). P. 1/1 – 1/4.