European Scholar Journal (ESJ)
Available Online at: https://www.scholarzest.com
Vol. 2 No. 5, MAY 2021,
ISSN: 2660-5562

DEVELOPMENT OF SOFTPLC MODULE TESTING METHODOLOGY

Urinov Nasillo Fayzilloevich
Candidate of technical sciences, associate professor
Yuldoshev Mirzobek Nurilloevich
Probationer teacher
Bakaeva Shakhnoza Nasriddin kizi
Undergraduate
Bukhara Engineering-Technological Institute

Article history:	
Received:	17th April 2021
Accepted:	30th April 2021
Published:	31th May 2021

Abstract:
The article covers methodology for testing SoftPLC module included in "AxiOMA Control" CNC system, implemented by creating a package of test programs in the FBD language, and test program algorithm has been developed.

Keywords: CNC system, SoftPLC controller, "AxiOMA Control", control program, user block, PLC program, programmable logic controller

INTRODUCTION

Software-implemented controller SoftPLC can function both as autonomous solution and built in the CNC system. In the CNC system, "AxiOMA Control" is implemented as part of general software-mathematical support of system, and solves the logical control problem. Software provision of the conditional software-implemented controller functions in the same operator area with the software provision of the control concept, because of which the maximum favorable result is achieved from its use. The features of controller are: hardware and platform independence, lack of a commercial license, and presence of debugging capabilities for control projects. The controller has a full set of functionality for solving automation tasks of varying complexity. [1]

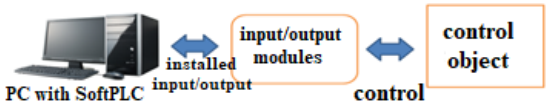

Fig. 1. Object control scheme using a software-implemented controller (SoftPLC).

Operation of control program and all mathematical calculations are performed in the software core of the CNC system. The core also controls hardware input/output of various types (analog, discrete), which, in turn, control the hardware. It should be noted that the controller could work in "offline" mode, i.e. without the use of hardware.
Applied toolkit of the controller includes a software package containing three main tools: development and debugging environment for control programs (FBEditor); device configurator; client for interaction with the controller core. [2]

MAIN PART

The field of research, as well as the debugging of managing projects FBEditor is considered a multipurpose device for the formation, visualization and debugging of PLC projects, and is also adapted for the purpose of working with hardware components of different manufacturers. The device configurator is used to set up the configuration of the connected equipment for correct work with it. Together with the editor, the client is automatically launched to connect to the computational core of the controller (CNC system). This is necessary to run the programmable logistics controller PLC programs and the ability to debug them. In the open dialog box, one must specify the type of connection to the controller core: remote or local. To connect to the remote core, you need to specify the IP address of the computer on which it is running and select the item opposite the lines for entering the IP address. The "Save" button allows remembering the specified IP address and specify it again on subsequent launches of the editor. To connect to the local core, i.e. for a core application running on the same computer, select "Local core", or specify the address "127.0.0.1" in the IP address column (selected by default). [3]

The first task that should be performed is to connect to the core of the CNC system to implement the ability to run and debug the developed control programs. We need to select the point of connecting to the core using the address 127.0.0.1, to connect the client to the local core. By clicking on the "Connect to the core" button, the interaction client is activated, the status of which is displayed in the information field on the right. After some time (5-15 seconds), the connection result is displayed in the color bar. If there is no connection, the status line is highlighted in red with a corresponding message. If the client was able to connect to the core, then the status line is highlighted

in green with a corresponding message, and the client is closed.

After finishing work with the connection client, the editor opens a dialog box for creating / loading a program file, which can be of two types:

1) project - control program; 2) library - creating custom blocks.

In the first stage, we will develop simple control programs without developing custom libraries. We will talk about the development of custom libraries a little later. After that, the main PLC programming environment, or FB editor, opens, which has many opportunities for developing, documenting and debugging control programs, as well as their verification. The FB editor is considered the main practical module for purpose of forming also the debugging of controlling projects. It is considered a controlling projects with a PLC target (FB editor), created using NET, XML technologies. The interface of the research environment for commanding projects allows carrying out research of commanding projects, setting the characteristics of multifunctional structures, the presence of a support panel of options, but also debugging projects.

The interface contains the following main components: the main menu; the menu for working with control files of the program; the objects panel - a list with the objects added at the moment, with which you can quickly find the necessary objects and make their selection; the object settings panel - serves to configure the parameters of functional blocks, as well as input/output objects; operation area managing projects - the main part of the interface, including visualization of PLC projects; multifunctional blocks menu - a panel including keys that correspond to different multifunctional blocks, which allows making the presence of forming managing projects; editor status bar - informative panel with the aim of reflecting the current capital connection to the core of the concept, but also the status of the activities of projects. [4]. The function block is added to the upper left corner of the editor screen by default. To perform any operation with a block (moving, deleting, setting parameters), it must be selected. This is done by single-clicking left mouse button on the block, while its color should change. To select several blocks, you need successively clicking on the blocks while holding down the "Ctrl" key on the keyboard. [5] For clarity of presentation of the program in the editor, it is possible to move the block to the foreground/background relative to other objects of the program. To do this, we will implement it by selecting the appropriate items in the context menu of the block. All function blocks have parameters that can be changed using the function block settings panel located at the bottom right of the screen. Most function blocks have the same parameter set and include basic parameters, graphic parameters and block parameters. Only "object name" and "object description", which are in the main parameters, are editable. The object name can contain a string of up to 15 characters, which is displayed above the block and allows the operator to more conveniently navigate the program. The object description is a more detailed description of the block, displayed only in the parameters panel. All this allows naming and describing function blocks so that the program is understandable to users [6]. For convenience, many function blocks allow changing the number of inputs (arithmetic, logical blocks). To do this, right-click on the function block, and select "Change inputs". A dialog box will appear in which you can add (+), delete (-), or move the inputs of the function block.

PLC programming comes down to creating a program based on the existing set of function blocks, their parameterization and connecting their contacts with links to transfer values between them. The connection of function blocks by links is realized by hovering the mouse pointer over the required block contact (input or output), from which it is necessary to create a link , and maintaining it with the left key pressed until the next contact (output or input, respectively), where you want to transfer the value. Links are created between the output and the input of blocks, and only one link is always given to the input to avoid ambiguity of the program. Any number of outgoing links can be created from one output.

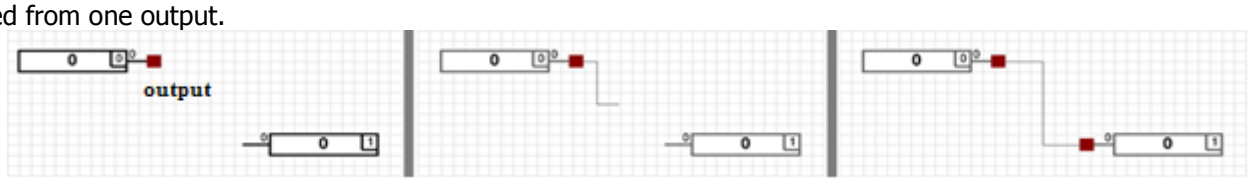

Fig. 2. Stages of creating a link between two blocks

When created, a link has two tie points to the contacts and two breakpoints by default. In the editor, we have the ability to prepare for printing kinks in the directions of the relationship, which allows avoiding overlapping directions of the relationship in multifunctional structures, and also make the installation most obvious and comfortable. For the purpose of this, it is necessary to select the interconnection with the line of a single group with the mouse in each section of the necessary interconnection, and also wind because of one of the break points up or down. In this case, add a break point in the center of the horizontal part of the connection, from which we make it "pull" with the mouse. Thus, we can create an unlimited number of kinks in the connection for their visual display in the program and to avoid intersections with other elements of the program. To remove any of the break points of the connection (except for the two basic ones), you need to either double-click on the break point, which must be removed, or manually align one break point with another.

To delete a link, you must select it and press the "Del" key on the keyboard, or select the "Delete" menu item in the "Edit" menu or the context menu.

Let us analyze in more detail the multifunctional constructions that the user is capable of doing in the presence of work. Let's take a brief look at the functionality of any of the blocks. Input/output constructions: Design information is provided for the purpose of the task of constant meanings, as well as for communication with the

hardware, by reading data from input devices (buttons, sensors), or issuing signals to the outputs of input/output devices. The "Input" block is the basic element of the program (initial), the "Output" block is the end point of the control program. The functionality of the FB editor provides the ability to adapt the corresponding properties of the inputs / outputs: the location of the port or memory, the type of input / output (BusCoupler, CommonPlcMemory, Port), length, importance, bit exponent, slot ratio, equipment ratio, type of insured significance, (Bit, Bool, Analog, Digital).

Types of possible values of input/output objects (In/Out): Bit - possible values - 0, 1; Bool - possible values - True, False; Analog - digital values - whole and fractional numbers; Digital - digital values - only whole numbers

In the programming environment of SoftPLC controllers there are two types of functional objects, which the client is capable of performing if he has a study managing plans: characteristic, in addition, custom ones. Simple multifunctional blocks are: input/output objects, objects of specific, logical operations, timers, counters, etc. User functional systems - systems based on standard elements, linked into a single resource with a user-specified number of inputs and outputs. This type of parts is specialized in order to significantly reduce the size of managing projects, but also with the aim of repeatedly using a set of structures that implement specific logic (Fig. 3)

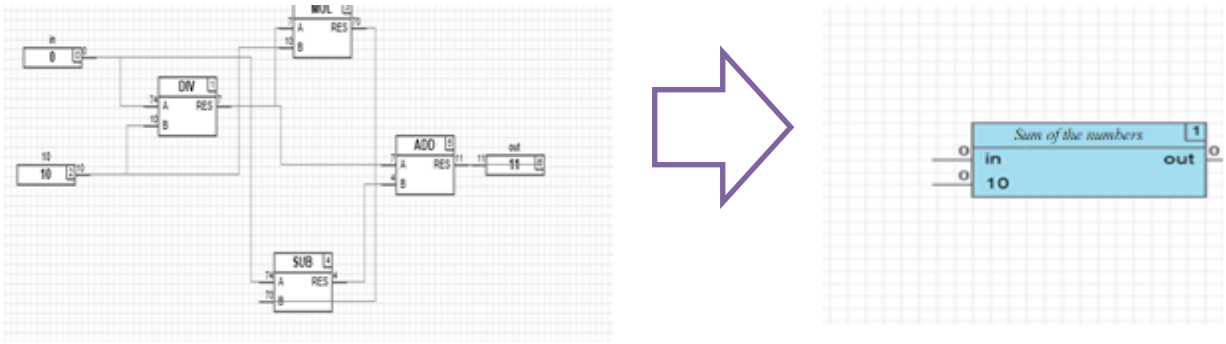

Figure 3. Representing some logic as a single block

We create and use special blocks for developing PLC control software. Connect to the core of the CNC system through the interaction client, and start the FB editor. When creating a file, select the "Library" file type. Let us open the same project window as when creating a regular program. The principle of creating a control program remains the same, the only change is that this project cannot be sent to the core and launched, but can only be saved for further use in simple programs.

Consider an example: creating a simple library to implement logic: Develop a custom library that converts an input integer (0-9) into a set of seven logic signals corresponding to segments of a digital LED. When submitting a number, the enabled ("1") outputs must be a combination of segments corresponding to the displayed number. The names and locations of the outputs in the library must match the illustration. Create a control program using the developed library that feeds a number from 0 to 9 to the library input with a frequency of 3 seconds. We use the following functional blocks: timer (2nd version), counter, as well as comparison blocks. Thereby, the input data will be two integers. Let us add two inputs to the circuit and set the data type "Digital". To change the data type, we go to the section object settings panel and the type of stored value.

The type of the stored value is of the following types: Digital, Bit, Analog, None. In the future, these inputs will be the inputs of the user library. You need to add an output with the "Bit" type to the circuit 8. In addition, we will add logical blocks "OR" from the section. The output will be the result - eight output, integer type. Thus, we need to add to the circuit eight outputs with the "Bit" type. Then, we need to implement the direct logic of the objects. The addition function is implemented using the ADD function block, and multiplication, as you know, is the MUL function block. Thereby, the diagram shown in Fig.4 should be obtained.

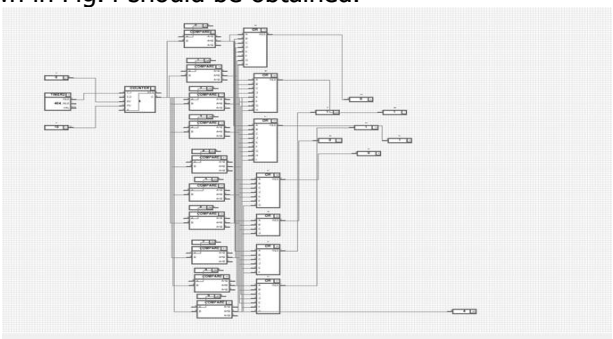

Fig. 4. Project of the control program using a user block.

In order for the future user block to be able to determine what its input/output of the object are intended for, we need to rename the input / output blocks to understandable names. The next step is to link the input/output objects to the future contacts of the custom block. That is, we need to specify which I / O objects will correspond to

which input/output (pins) of the custom block, and in what order. To do this, we need to call the input/output binding window located in the Library -> input/output binding menu. When this dialog is called, a window for linking inputs/outputs and basic settings of the user library opens. Block Display Name - The name of the custom function block that will be displayed on it ("NUMBER2").

Block Description - a detailed description of how a custom block works. Block inputs/outputs - panels with the ability to bind input / output objects to future contacts of the block. These panels have keys "+" and "-", respectively, for adding a contact with an associated input / output, or deleting it. For our purpose, we need to bind three inputs and one output. To do this, click on the "+" key in the "Block Inputs" panel, specify the first input object named "Inputs 1", and click "OK". Let us perform the same operation with the rest of the inputs / outputs, and also set the name and description of the custom block, then press the "OK" key.

Thereby, the creation of a library is realized, which can be reused in the future in simple control programs for PLCs. We save the file in our folder. Let us create a simple control program project. To use the created custom block in the program, it must be imported into the project. To do this, run the File-Import user library command. Find the saved library file and click "Open". If the import is successful, we should receive a message about this action. In addition, in the menu of function blocks, we should have a key corresponding to the name of the new user block. Now this user block is a full-fledged functional object that we can reuse in any programs

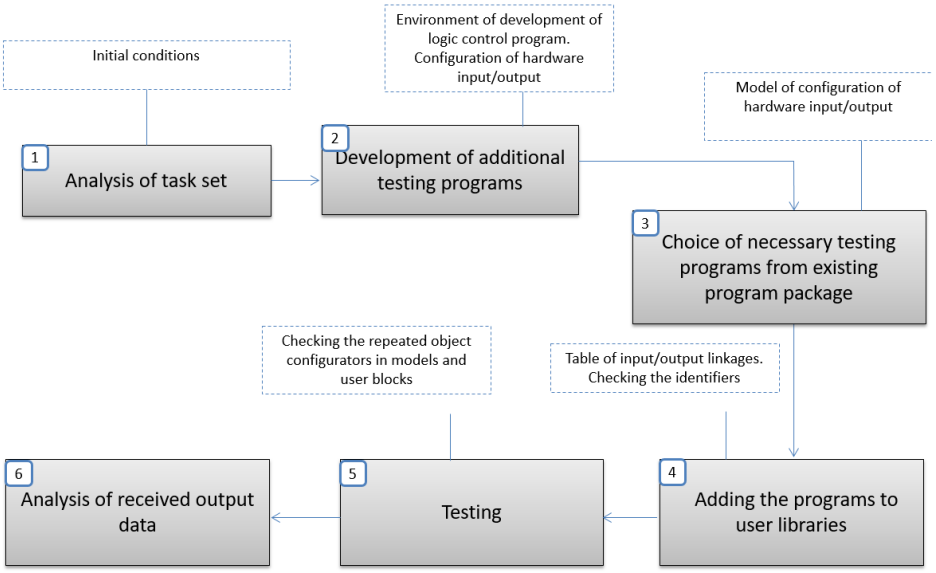

Fig.5 Development of a structure for testing FBEditor as part of "AxiOMA Control".

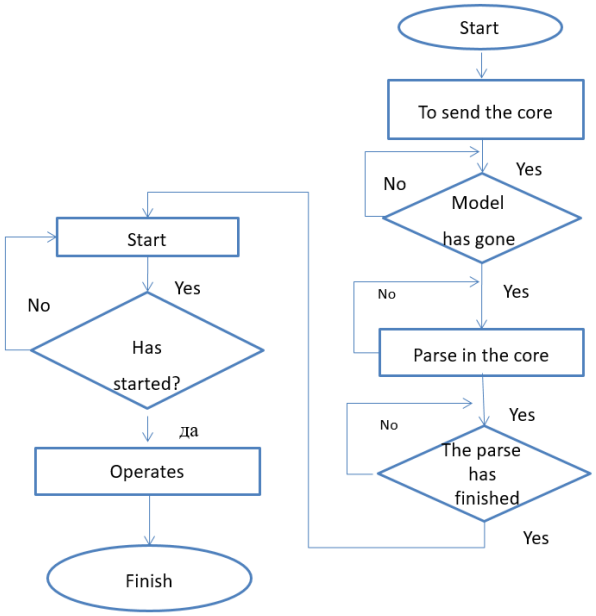

Figure 6. Development of a control algorithm for SoftPLC

CONCLUSION

In process of research, a set of testing methods was considered for developing the SoftPLC module and testing program algorithms. The AxiOMA Control SoftPLC module was tested based on the developed methodology. The work with connections between functional blocks and the creation of libraries of components in the PLC programming environment FBEditor is considered. The project of the control program using a custom block is presented.

LITERATURE.

1. Sosonkin V.L., Martinov G.M., Numerical control systems, Moscow, Logos, 2005
2. Sosonkin V.L., Martinov G.M. The concept of numerical control of mechatronic systems: the implementation of a logical task, Mechatronics. 2001. No. 2, p. 3-5
3. Martinov G.M. Sosonkin V.L. The concept of numerical programmed control of mechatronic systems: the problem of real time // Mechatronics. 2000. No. 3, p. 37-4014. http://www.osp.ru 15. www.vxworks.ru
4. Sosonkin V.L., Martinov G.M., Perepelkina M.M. The concept of electro-automatic control of CNC machines as virtual SoftPLC controllers.
5. Nezhmetdinov R.A., "Application of software-implemented controller of Soft PLC type for control of electro-automatic machines with PCNC CNC". Scientific-technical conference "MTI-2008", vol 2.
6. Shemelin V.K., Nezhmetdinov R.A., "Software implementation of logical problem of numerical control (CNC) based on a Soft PLC controller". United Scientific Journal, No. 10 (216) 6 Moscow, 2008, p. 46-48 (ISSN 1729-3707)