Extending the Use of PLC Simulator Software in Student Laboratory Works

Cristian-Gyozo HABA
"Gheorghe Asachi" Technical University of Iasi, Romania
cghaba@ee.tuiasi.ro

***Abstract*—This paper presents a system that can extend the use of PLC simulator software in real world experiments. The system is composed of a software and a hardware part. The software is designed to monitor the simulator user interface, to provide user-like inputs to this interface and to exchange data with the hardware part. The hardware is used to interface the simulator running on a PC with external input and output devices. Input signals are converted into inputs for the simulated design; outputs from the simulated design are converted into output signals that control output devices connected to the hardware part. The system can work with various types of simulators and can be used in university labs to design and test digital control systems with low to moderate complexity.**

***Index Terms*—engineering education, PLC simulation software, hardware interface**

I. Introduction

The advantages of simulation tools in the flow of a product design are well known. The main advantage is that they can be used for testing purposes in different stages of the design flow without the need to involve prototyping hardware, thus reducing the total costs. Simulation tools have become widely used also as complementary teaching tools in different courses. Simulators are good educational tools because they can be used to teach the steps of a design flow and how to test the functionality of the design without involving any hardware parts that can be damaged by careless or incorrect use [1, 2]. Whether they replace all or part of the real experiments, the lab works result to be cheaper and safer.

The simulators available on the market differ in functionality, user interface, friendliness, cost and computer resource demanding. They range from simple freeware simulators issued from student projects to high cost, dedicated simulators written by highly skilled development teams. Choosing the right simulator depends on the studied system and on the available financial resources.

It is well known that simulators have limitations in covering all test and verification processes of a product design. In the same way, they have limitation in the educational process. Simulators can be used to teach some concepts but they cannot fully replace the real experiment and the amount of information and knowledge students can gain from it. Some studies have been carried on to bring together simulators and real experiments by combining running software on the PC with pieces of hardware for interfacing with external signals and events [3,4]. For example in [4], the controlled process is simulated on the PC, whereas the control is provided by an external hardware device (PLC). The interface between the two is implemented into a dedicated piece of hardware.

In our approach, the control part is implemented in a design that is simulated on the PC using a PLC simulator while the controlled process is external. The two parts are connected using a hardware interface. The special feature of our system is that we can interface a large set of PLC simulators without involving reverse engineering or intrusion into the simulator software. A direct application, which we discuss in this paper, is to use our system in an educational environment. Our system can thus transform the simulation-only based labs into more practical ones, in order to provide students with an experience closer to real world applications.

II. Description Of The System

There are a lot of initiatives to develop and use simulation software for courses in the area of digital systems [6-8]. Depending on the implementation, there are two kinds of simulators:

- generic simulators – they simulate digital designs described using logic diagrams, finite state machines, ladder logic or sequential functional charts;
- dedicated simulators – they are tailored to simulate a specific proprietary hardware device (smart relay or programmable logic controller - PLC).

Generic simulators are used only for simulation. They are normally used for teaching purposes and they are not related to some hardware. They are stand alone programs which allow user interaction through a text or graphical based user interface.

Dedicated simulators are usually integrated with the programming software developed for a specific PLC family. Developing a PLC application implies writing the application program using the available programming language (the standard are instruction list, ladder diagram, function block, structured text and sequential function chart), simulating the program, transferring the program to the PLC and running the application. If the PLC programming software also includes a monitor, the operation of the PLC can be observed on the computer display and the user can interact with the PLC at a certain degree (Figure 1).

In order to cover the entire flow of PLC application development, we need the programming software (including simulator and monitor), the PLC and the input and output devices that can be used to test the application. But providing enough equipment for the PLC lab can be quite expensive. In order to reduce equipment costs, we propose our system.

Digital Object Identifier 10.4316/AECE.2010.01015

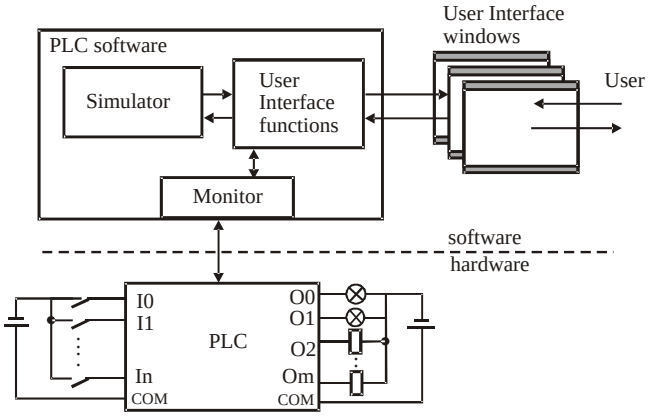


Figure 1. PLC monitored using associated software.

From working with different PLC simulation software, we have observed that they have some common features:

- can simulate most part of the associated hardware functionality;
- allow changing status of inputs using the keyboard or mouse clicks;
- display input and output On and Off states with different symbols and/or colors.

Taking into consideration these features, we have developed our system called *Monitor and commander* (M&C) consisting of a software part and a hardware part. The system interacts with the PLC simulator user interface by monitoring graphical representations of design outputs within the simulator window and issuing keyboard presses and mouse clicks in areas associated with design inputs. Graphical symbols for outputs are monitored in one point only (it must be a relevant point, i.e. that changes color if output changes status). The information regarding the modification of input/output status of the simulated application is exchanged with a hardware part, which implements the interface between the simulator and the real world (Figure 2).

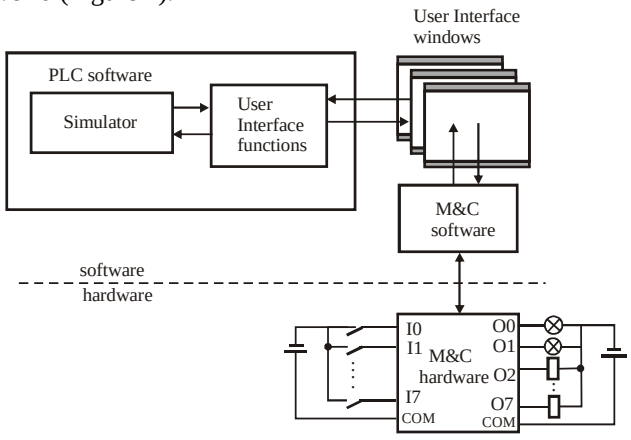


Figure 2. M&C proposed system composed of a hardware and a software component.

The developed software is used in conjunction with the simulator software follows the steps listed below:

1. Start the simulator program (and its user interface);
2. Start the M&C software. The program starts in configuration mode. In the configuration mode, the user sets the points on the screen to be monitored and the points on the screen that will be clicked on.
3. Identify the text or graphic elements of the simulator user interface that show the status of the design outputs. Identify for each output element one point that will change color when the corresponding design output changes its status. These points will be set in M&C as output points to be monitored.
4. Identify the text or graphic elements of the simulator user interface that are used to change the status of design inputs. Identify for each input element one point (in the input sensitive area) that, when left clicked, will change the status of the corresponding design input. These points will be set in M&C as the input points to be commanded.
5. Start the monitoring and command mode of the M&C program. In the monitoring and command mode the program will cycle through the following operations:
    a. monitor the status of the output points (a change of color will indicate a status change for the corresponding output). The change in the output will be reflected in an output register (used to update the outputs of the hardware).
    b. monitor the status of the input register (update based on the hardware part inputs). A change in the input register will indicate a change in the state of the corresponding input of the hardware part. A change in the input state will issue a click on the point corresponding to the changed input.

Input register mirrors the status of the external inputs whether the output register mirrors the status of the simulated design outputs. Information in these registers is exchanged between M&C software and M&C hardware using the computer parallel port.

III. System Implementation

A. Hardware component

The hardware component is used to interface the M&C software (and thus the simulator) with external input and output devices. The main functions implemented in the hardware component are:

- read input data from input devices (push buttons, switches, on-off sensors, limit switchers, etc.) and send it to the simulator using the parallel port. The actual version of the hardware component supports maximum 8 input devices;
- control On-Off outputs (relays, LEDs, valves) connected to the hardware component based on the data received from M&C software (and thus from the simulator), using the parallel port. The actual version of the hardware component supports maximum 8 outputs.

Depending on specific applications, additional functions can be added to the hardware component:

- LEDs for status display of inputs and outputs;
- output protection for inductive loads;
- provision of necessary current to drive output devices.

In order to keep the hardware simple and compatible with older versions of parallel ports, we have chosen to read and write data in nibble mode.

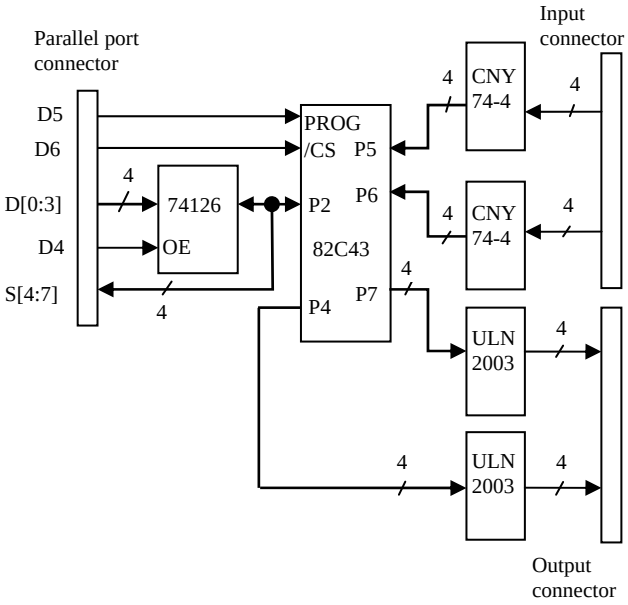


Figure 3. Block diagram of the hardware component of the proposed system.

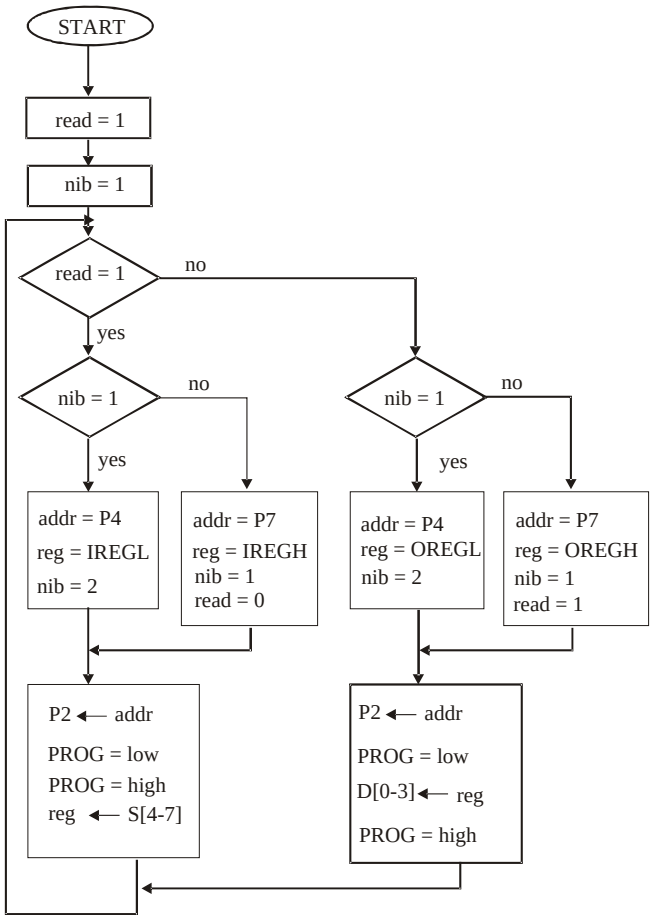


Figure 4. Logic diagram for exchanging data between the software and hardware components.

We have implemented a prototype for the hardware component based on the 8243 I/O expander. Two of 8243 circuit ports are used as input ports (*P4*, *P7*) and two are used as output ports (*P5*, *P6*). In order to allow read/write operation from port P2 using the parallel port of the computer, a 74LS126 three-state buffer is used. When the buffer is enabled (*D4* high), data from *D0-D3* lines of the parallel port are transferred to port *P2*. These four lines of the data register of the parallel port are used to send, in nibble mode, the status of the 8 bit output register to the hardware component. The information will be converted into signals that will activate the external output devices. When *D4* is low, outputs of the buffer are three-stated and data can be read from port *P2*. To achieve this, four status lines of the parallel port (*S4-S7*) are used to input data. The two nibbles are used to update the 8 bit input register. In order to transfer data between *P2* and the other ports, the 8243 circuit must be programmed. Data register lines *D5* and *D6* are connected to *PROG* and */CS* pins and they are used to control the operation of 8243 circuit.

Inputs of the hardware component are optically decoupled using two CNY74-4 multi-channel optocouplers with phototransistor outputs. For the outputs, two ULN2003 Darlington arrays are used to allow higher currents at the outputs.

The block diagram of the hardware component is depicted in Figure 3.

The logic diagram in Figure 4 depicts the process of reading data from the hardware component into the input register (IREG) and writing data from the output register (OREG) to the hardware component, both operations in nibble mode. IREGL and IREGH represent the low and high nibbles of the IREG register. In the same way, OREGL and OREGH represent the low and high nibbles of the OREG register.

B. Software component

The software part of the system was written in Visual Basic version 6 using the DriverLINX Port I/O Driver. It consists of a set of functions that perform the following:

- setup of input and output points,
- monitor and control output and input points respectively,
- display monitoring and control information in the interface window,
- exchange data with the hardware part using the PC parallel port.

Setup of input and output points is based on functions that get the coordinates and displayed color at the current mouse position and store them in appropriated variables.

Monitoring of the output points is done by periodically reading the displayed color at the saved output point positions and comparing them with the initial values. If the color has changed at any of the output points, the logic value corresponding to the output point in the output register is set to 1. Otherwise the logic value is set to 0.

The command of the input points is done by periodically checking the modification of the input register. Any value modification in the input register results in a virtual mouse click at the corresponding input point. In a similar way, a virtual key press can be issued for keyboard controlled simulators.

Data exchange function controls data flow from computer to hardware component and vice-versa by performing periodical data readings and writings from/to the parallel port registers.

C. M&C User Interface

The M&C program interface has three sections: input section, output section and interface setup and control section (Figure 5).

The output section lists information on the monitored points (corresponding to the monitored outputs). The left hand side of the section lists the position (in pixels) and color (at selection time) for the points to be monitored. The color which a point has at selection time is considered to be the color of the *Off* status of the output. This usually corresponds to the situation when the simulator is disabled or has not yet been started: there are no active inputs or outputs. The right hand side of the output section lists the actual color of the monitored points and the status of the outputs based on this color, also displayed in the window. A text label for each output indicates whether the output changed or remained in the same status. We can see in Figure 1 that points corresponding to outputs 1, 3, 5 and 8 have the same color as at setup time, therefore their status is *Off* (logic 0). At the same time, points corresponding to outputs 2, 4, 6 and 7 have changed their colors, therefore it is considered that they are in the *On* state (logic 1).

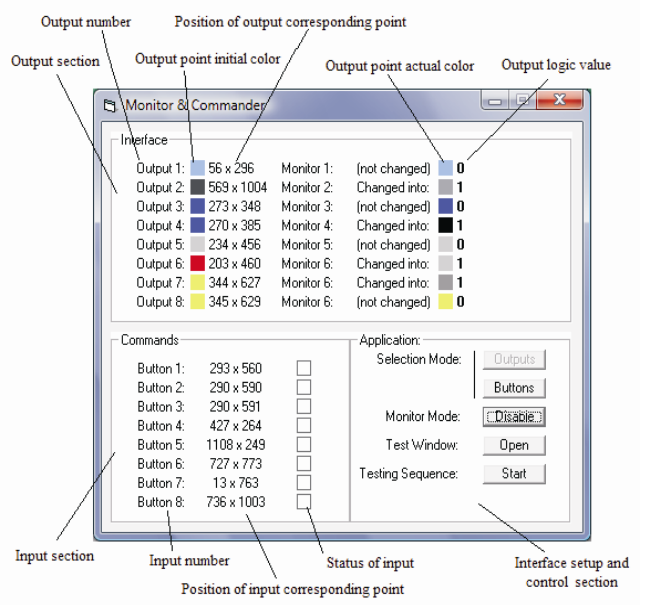


Figure 5. M&C user interface window with the input, output and control sections.

The input region lists information of the input points, i.e. their position (in pixels) and their status: normally *Off* and on mouse left click, *On*.

Setting the position of the input points is done by clicking the *Buttons* button. A process of tracking mouse position is initiated. When left mouse button is clicked, the current position of mouse cursor is associated to the first input. Immediately mouse is tracked to establish the position for the second input point. Process repeats until position for all input points is selected with mouse clicks.

Pushing *Outputs* button starts the process of setting the positions of the output points using the same track mouse and pick position on left click procedure. Beside the mouse position, in the output case the color of the point is recorded.

Setting the input and output points can be re-done by pushing again the corresponding buttons. In the actual version of the software we cannot set the position for single input or output points. Once we have started the setting process, either for input or output points, the position for all points must be set.

After input and output points are selected, the monitor and command process can be started by pushing *Enable* button (M&C enters the running mode). This process can be stopped by pushing the same button, which in running mode is labeled as *Disable*.

The *Open* button is used for debugging purposes. When pushing this button, an additional window appears containing a set of push buttons that can be used to test if the input command part of the software works.

IV. Simulator Examples

A lot of PLC manufacturers offer software for programming their PLCs. Simulation is an important step in the development of the application code when designers eliminate a large part of initial design bugs. Therefore, it is not unusual for the PLC programming software to include also a simulator. These simulators can also be useful during student labs as they provide a safe environment for the initial stage of teaching how to develop a PLC application.

PLCs are offered in different sizes and hardware configurations but usually members of the same family are based on the same processor and therefore use the same application development software.

It would be ideal to have in the lab enough PLCs so that each student could work on its own PLC. It would be useful to have the possibility to update PLC hardware equipment each time a new family is introduced on the market. Unfortunately, excepting the smaller members of the simpler families, PLCs have prices that are not affordable in large quantities.

Teaching how to program a particular PLC becomes easier by using our proposed system. The condition is that the programming software should include a simulator that doesn't need the hardware to be present.

We give some examples of PLC simulators we have tested with our system.

Mitsubishi Alpha devices are small PLCs that can be acquired in two family versions Alpha and Alpha XL [10]. They can be programmed using a Function Block (FB) language but which is not conforming to the IEC 1131-3 standard [11]. Nevertheless it has a large set of graphical symbols for inputs and outputs that can make the FB diagrams easy to understand and which can suggest the real application that can be controlled with. The programming software has a simulation mode that shows the diagram operation by changing the shape and/or color of the graphical symbols. Activation of interconnection wires is also visible as they change colors depending on their status. In Figure 6 it is depicted the use of M&C software in conjunction with the ALVLS software, which runs in Simulation mode.

We can see that the control diagram has four outputs that drive four lamps. The graphical symbol of the output lamp is colored black when not activated and yellow when activated (to suggest lamp lighting). Setting the output points in the region that changes color from black to yellow allows our program to monitor output changes in the simulator. Accordingly, we can see that outputs 1 and 2 are off, which is reflected by the fact that initial and actual color of corresponding output points are both black. Outputs 2 and 4 are on, therefore the actual color of corresponding output points are, as we expect, yellow.

We have tested our system with different PLC

programming software. These were Easy Soft Demo for Moeller Easy family PLCs [12], GLOFA PLC simulator [13] for the GLOFA PLC family from LGIS and TRiLOGI [14] for T100MD+ and T100MX+ PLCs from Triangle Research. The user interfaces for these programs are given in Figures 7, 8 and 9.

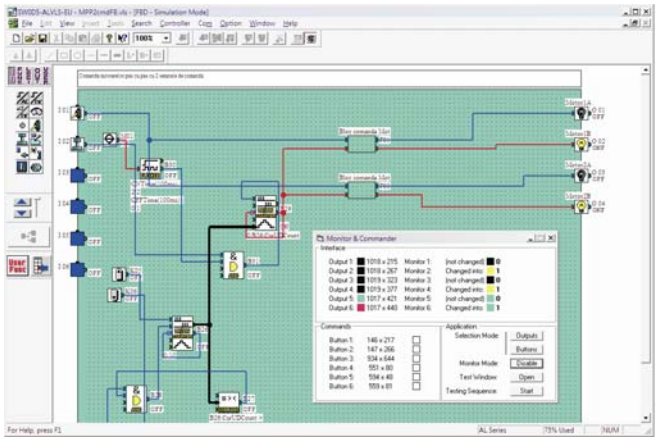


Figure 6. A function block diagram simulated in ALVLS software with M&C software monitoring the outputs and controlling two inputs.

We can see that the simulators have different graphical interfaces and, for each one, we shall present in the following some of their specific characteristics.

Moeller Easy PLCs belong to the same class as the Mitsubishi Alpha PLCs (smart relays) but, as we can see by comparing the two corresponding pictures, Easy PLCs are programmed using a variant of Ladder Diagram language (Figure 7). Secondly, we must say that, in fact, Easy Soft Demo program is based on a Flash animation. The simulator provides graphical elements (push buttons for inputs and LEDs for displaying output status);

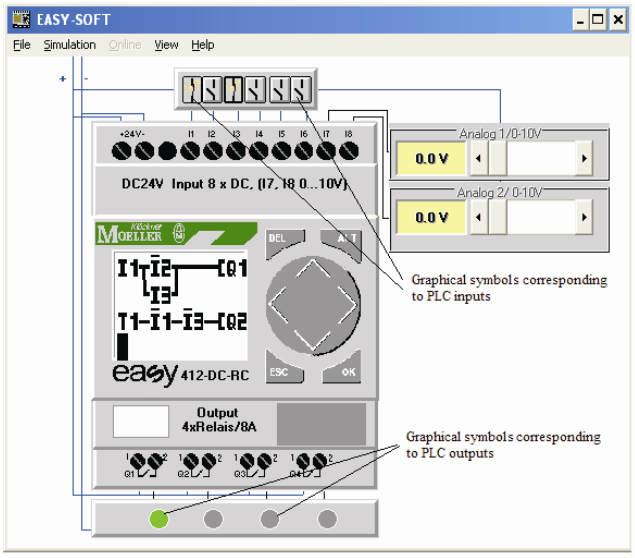


Figure 7. Moeller EASY smart relay simulator in run mode.

GLOFA PLCs are medium size PLCs that can have up to 60 I/Os. The GMWIN programming software provide international standard (IEC1131-3) programming languages: IL (Instruction List), LD (Ladder Diagram), SFC (Sequential Function Chart). The GLOFA PLC Simulator presents an user interface that parallels the view of a modular system where extra I/O modules can be added if necessary (Figure 8). Status of the inputs and outputs are represented by LED elements. The status of an input can be changed by clicking on its corresponding LED element.

We have chosen TRiLOGI software because it has both DOS and Windows versions of the simulator. The DOS version has a text based user interface. We have also tested the DOS version of the simulator which demonstrated that our system also works with simulators that have text-based user interface.

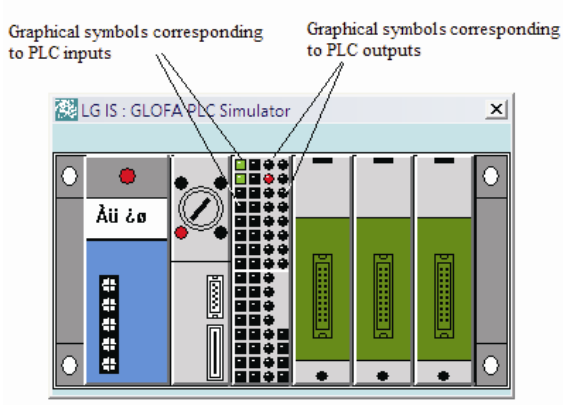
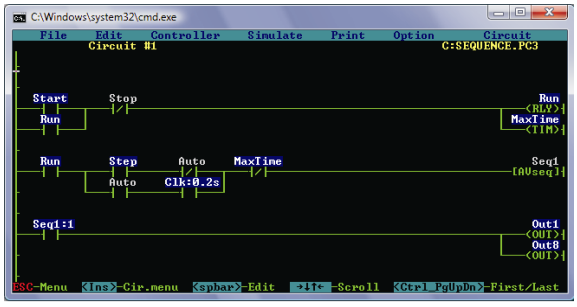
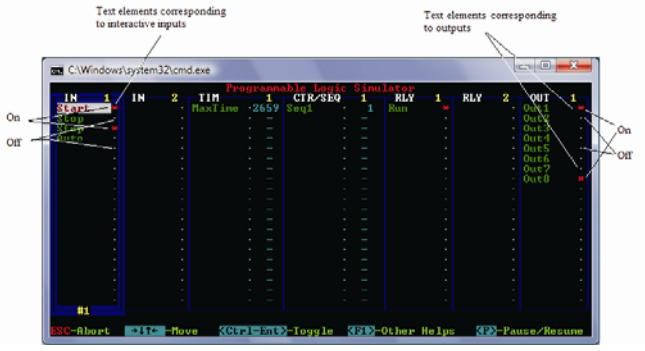


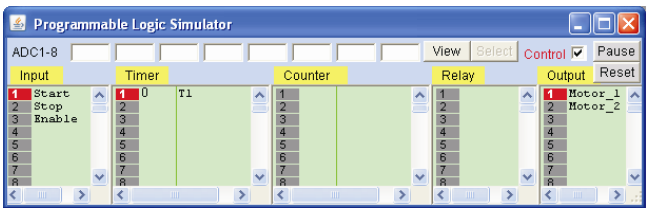
Figure 8. GLOFA PLC simulator in run mode.



(a)



(b)



(c)

Figure 9. TRiLOGI simulator in run mode: (a) DOS version ladder diagram, (b) DOS version simulator interface, (c) Windows version simulator interface.

V. Discussion

Using the M&C system with its two components (hardware and software), any PLC simulator software can be transformed into a PLC, which students can use to develop PLC applications. They can implement control systems that

can receive digital inputs from input devices connected to the hardware part inputs, process this information and, on the basis of the simulated results, issue the answer represented by digital signals at the output of the hardware part. These signals can be used to command On-Off output devices like lamps, relays, valves, motors etc. In this way, students can not only simulate the functionality of their design but also test it without possessing the original hardware.

The advantages of the presented system are the following:

- work with a large set of PLC simulators including also generic simulators (not bounded to a specific PLC device);
- receive digital inputs signals and can issue digital output signals;
- use the parallel port and a simple protocol to interface the simulator with input and output devices;
- the software part can be modified so that other computer port be used to do the communication between the PC and the hardware interface (ex. serial RS232 or USB);
- provide students equal experience with simulation and real experiments.
- Nevertheless, some drawbacks exist:
- overall system performance and accuracy is limited by the PLC simulator performances;
- overall system speed depends on several factors such as: simulator speed, PC characteristics and PC load during simulation;
- the combination simulator - M&C system cannot implement original hardware functionality;
- safety issues can arise during simulation – M&C system operation (PC freeze or reset, uncontrolled pop-up windows, wrong keyboard or mouse commands), therefore this system should not be used in critical experiments or applications;
- some functional features of the original hardware are impossible to be reproduced (ex. analog or high-speed inputs).

VI. Conclusions

In this paper, we have presented a system consisting of a software part and a hardware part that can be used in labs to extend the simulator functionality so that they can be used to do experiments in the real world. We have presented four simulator examples that have been used to test our system. However, other PLC simulator software can be used as easily.

The use of the same M&C system allows students to test the functionality of different PLCs and to become familiar with the development of applications for these systems without the need to buy the expensive equipment.

Our system can be further improved to offer students a better experience during lab works. Some of the possible improvements are:

- configuration of the number of inputs and outputs;
- possibility to select the port type for the communication between PC and hardware interface;
- independent selection of output or input points;
- separate monitor part from the command part in order to extend system application.

Acknowledgment

Part of the research in this paper was supported within the research grant CNCSIS code 223, contract GR/11.06.2008.

References

[1] Z. Kurmas, “Improving Student Performance Using Automated Testing of Simulated Digital Logic Circuits”, Proceedings of ITiCSE’08, June 30-July 2 2008, Madrid, Spain, pp. 265-270.

[2] D. A. Poplawski, “A pedagogically targeted logic design and simulation tool”, In WCAE ’07: Proceedings of the 2007 Workshop on Computer Architecture Education, 2007, pp. 1–7.

[3] F. W. Bruns, Lernen in Mixed Reality, ABWF (Ed.): Kompetenzentwicklung 2003, Waxmann, Berlin, pp. 71-112, 2003

[4] EasyVeep – Handbuch/Manual, Festo Didactic GmbH & Co. KG, February, 2003.

[5] C. Burch, “Logisim: a graphical system for logic circuit design and simulation”, J. Educ. Resour. Comput. 2(2002), 5–16.

[6] H. Dierks and J. Tapken, “Modelling and Verifying of 'Cash-Point Service' Using Moby/PLC”, Formal Aspects of Computing, 12 (2000), 222-221.

[7] D. E. Clough, “The Missing Link in Process Control Education – Incorporating PLC’s Into the ChE’s Control Course”, In Proceedings of the 2002 American Society for Engineering Education Annual Conference & Exposition, Montreal, Quebec, June 16-19 2002.

[8] C. A. Chung, “A cost-effective approach for the development of an integrated PC-PLC-robot system for industrial engineering education”, IEEE Transaction on Education, 41 (1998), 306-310.

[9] Alpha Simple Application Controller. Software Manual, Mitsubishi Electric Corporation, July 2002.

[10] IEC 1131-3 Programmable Controllers - Part 3: Programming Languages, International Electrotechnical Commission, 1993.

[11] Easy800 Steuerrelais, Bedienungshandbuch, Moeller GmbH, August, 2004.

[12] User Manual of GMWIN version 4, LG Industrial Systems. January 2000.

[13] Internet TRiLOGI v.53. Programmer’s Reference, Triangle Research International, Inc., 2003.