A User-Friendly Software Tool for Teaching of Industrial Automation Circuits Design and Simulation

PANAGIOTIS MICHAEL[1], STAMATIS MANESIS[1], DIONISIS KANDRIS[2]
[1]Department of Electrical and Computer Engineering,
University of Patras,
GR-26500, Patras,
GREECE
[2]Department of Electronics, School of Technological Applications,
Technological Educational Institution (T.E.I.) of Athens,
GR-12210, Athens,
GREECE

Abstract: - This paper describes a computer-based tool developed to design, simulate and verify Relay Ladder Circuits applied in industrial automation applications. By using a computer, a student can create Ladder circuits and extensively simulate, test and modify them. The visualized test of operation of a circuit can be performed either in real time or in a step-by-step procedure. In order to establish the proper operation of a sequential control system, the software developed includes a verification option by which possible errors can be identified and control logic can be investigated for various scenarios of inputs. The software package is suitable for both educational and industrial practice purposes.

Key-Words: - Industrial Automation, Relay Ladder Circuits, Computer Aided Design, Simulation

1 Introduction

In the majority of industrial applications, systems with hundreds or thousands of inputs and outputs are controlled by using programmable logic controllers (PLCs), which are the most suitable and widely employed control technology in factory automation. In such cases a logic controller must handle not only the normal operation sequence and synchronization of systems, but also the operator interface, error handling and recovery routines. The control program is usually written in a low-level language called Relay Ladder Logic and can be very complex for large systems. As PLC technology evolves into a family of control functions of many types at many different levels of complexity, there is an increasing need for software tools which enable the systematic development of real industrial applications.

Usually, an industrial automation system includes various levels of control and monitoring from the field controllers up to the supervising layers. There is an intensive research on theoretical subjects of the higher levels of this hierarchical scheme and consequently various software tools have been developed for modelling, simulation, execution and translation purposes. For example, a rule-based method has been presented to derive a Ladder-logic program from a high-level system model [1]. At lower levels of the hierarchical scheme mentioned above, there is a need to develop user friendly tools. These tools should have graphical interfaces with analytical mapping, and high computational efficiency in order to allow industry to adopt real applications of discrete event system theory. Many researchers work on the improvement of industrial logic creation and design. The problem that many try to solve is the perceived inefficiency of the current methods, which are very time-consuming and make use of primitive, low-level design languages [2]. Companies such as Rockwell and Siemens have already started to develop software tools for this purpose, e.g. VALID software from Siemens [3]. Various software tools have been developed for the modelling and verification of PLC-based systems such as UPPAAL2k, KRONOS, Supremica and HyTech mainly for programs written in statement list language called also Boolean [4], [5], [6], [7]. The GRAFCET formalism that is often used in the implementation of PLCs is closely related to Petri net representations. Grafchart is a software tool for supervisor control system based on Grafcet, Petri nets and object oriented programming. It has

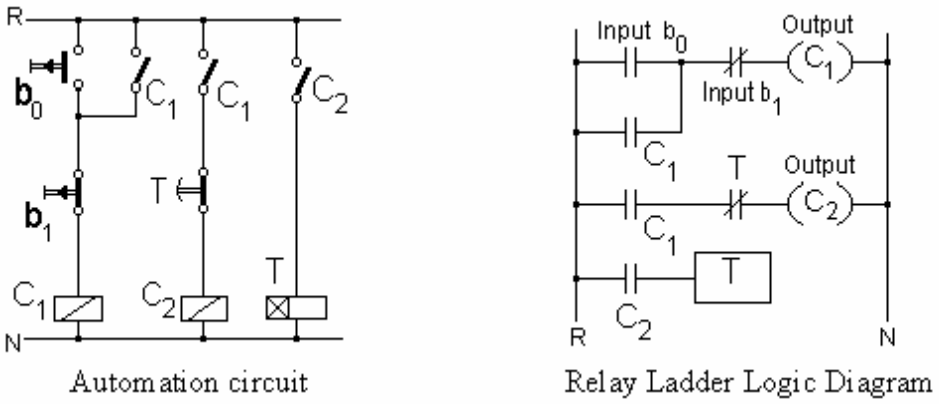

Fig.1. Two forms of the same Boolean logic, the conventional (wiring) and the programming one (software).

been converted to a Java platform under the name JGrafchart [8].Also, a graphical editor for state charts has been developed by ABB in Java language [9]. Some other tools initially developed for educational purposes, were further enhanced by their developers for general use like DESCO at Chalmers University, UMDES at the University of Michigan and CTCT at the University of Toronto [3].

Programmable logic controllers are extensively used in factory automation. The first step an engineer has to do in order to program a PLC is to make a Relay - Ladder diagram. At this point, it is essential to make clear the difference between a relay-ladder program and a relay-ladder diagram. Actually, a relay-ladder program is a Boolean-based PLC program, while the relay ladder-diagram or automation circuit is the initial schematic featuring the very first form of a relay logic sequential control system. An example that illustrates the difference between these two forms is shown in Fig.1.

Usually, the design of a sequential control system starts with its automation circuit design. Next, the automation circuit operation is implemented by the development of a corresponding PLC program. Various programming languages are available for PLCs such as Boolean, Relay-Ladder Logic, Function Chart and Grafcet [10]. Therefore, there is an apparent need for the development of a software tool for ladder diagrams efficient design and off-line simulation. The software package presented in this paper, called ACS (Automatic Circuit Simulation), was developed in order to meet this specific need. It was originally designed for educational purposes in order to assist undergraduate and postgraduate students to design and simulate relay ladder diagrams of industrial automation systems. This is very important, given that systems of such a kind are by default difficult or even impossible to be available in an educational institution. However, as it is shown later on, ACS is a simple yet effective tool even for use in an industrial environment. One of its main features is its verification option. It enables the user to identify any malfunctions may occur during the real operation and investigate all alternative scenarios of operation. In this way, it becomes viable to apply several verification methods which have been developed for PLCs to test the safety and reliability of control systems [11], [12].

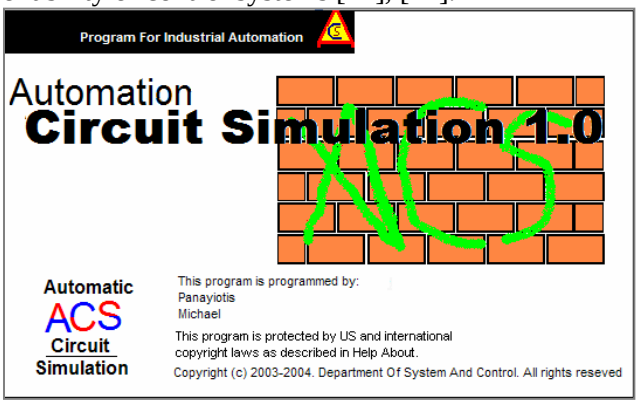

Fig. 2. Starting mask of ACS software tool.

2 Software Tool Description

ACS software tool, the starting mask of which is shown in Fig.2, requires an Intel P3 based or compatible personal computer with at least 256 MB RAM running Windows XP operating system. It requires a VGA graphics card with 128 MB memory.

ACS generates Ladder electric circuits as the first output form of the solution to an automation problem. Then, the Ladder electric circuit can be easily translated into a Ladder logic program according to IEC 61131 standard, which remains the most popular programming language for PLCs [13]. The program was developed by using Visual Basic v6.0 permitting some routines to handle dynamic lists in order to achieve maximum performance. A Graphics User Interface (GUI) has been developed

with pull-down menus where the main functions are included. On the top there is an extensive library, as shown in Fig.3, with all the symbols needed to design the diagram (relays, time relays, buttons, contacts, limit switches etc.). The user can pick the appropriate symbols and place them in the desirable positions. The program can accept up to 100 pages, 1200 relays and 8400 contact-type elements. The only limitation in the number of usable symbols is set by the memory available in the computer system used. Usually, 256 MB of memory is considered to be enough even for very large circuits.

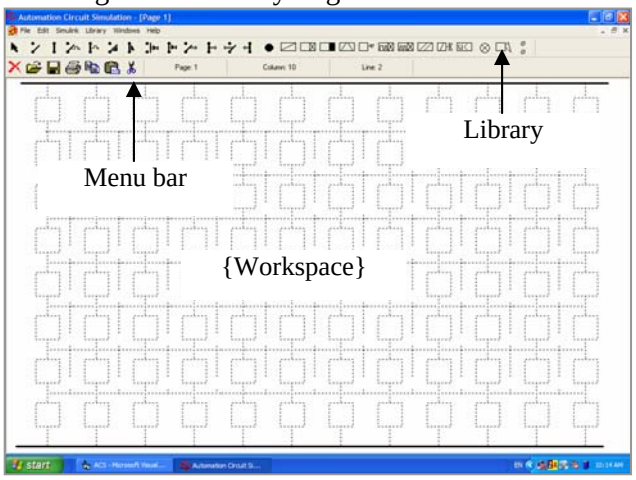

Fig. 3. User interface and workspace environment

The user can delete symbols, modify parameters (if any), save the work, print it and file it. The edit menu offers various operations like *cut*, *copy* and *paste* of an entire page or of a part of it. Complicated circuits can be designed on successive pages, while each node can have as many connections as needed. The workspace scrolls so that any part of the design can easily be accessed. The file menu offers the typical options of *open save* and *print* of an entire program. The library of the program is user open and can accept circuits as elements or sub-libraries. There is also a well organized menu for on-line help.

The simulation menu offers two types of simulation. The first one is the real time simulation while the second one is the step-by-step simulation. In the real time simulation the program tests the circuit operation as it would be performed if it was implemented. In the step-by-step mode, the simulation is paused or triggered by the user so that any possible malfunctions can be detected and localised more easily than in real time simulation. In the simulation menu there is also the verification option by which it is possible to perform a more thorough and extensive test of a circuit operation. This can be done by setting limitations, virtual damages or undesired combinations and making the computer check and detect if these scenarios are possible to happen. This feature is very important because in large circuits this can't be done manually or if it can be done it will take much more time and with a great probability of mistake.

During the drawing procedure of an automation circuit the components are coloured red. This means that no current flows through them. When the user starts the simulation the parts through which current flows are drawn green as it is shown in Fig.4. Thereby, the user can visualize the operation through the computer monitor. Using the menu the user has the ability to create virtual damages (i.e. a cable cut). In this way, it can be examined how the automation circuit responds under specific undesired situations. The verification option enables the examination of the circuit operation under various scenarios that the user can define. Usually the verification is based on manual test and the designer's experience. But as the complexity increases, the more difficult becomes for a human to test all possible scenarios. This can be done by ACS in a much more reliable and faster way. The user can inform the system about specific combinations of inputs and outputs that are undesirable to be activated at the same time and the program can check if the current design allows such combinations.

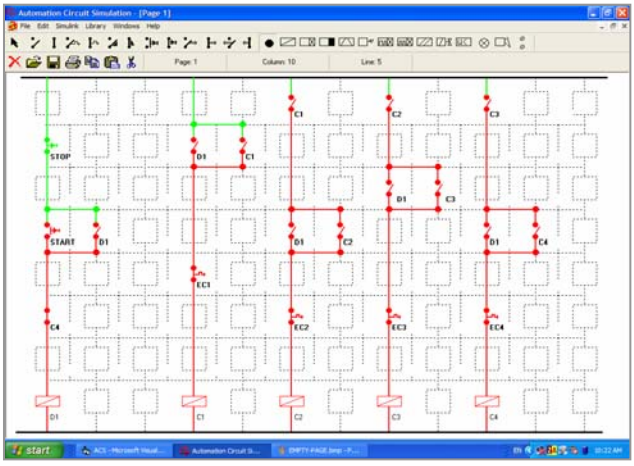

Fig. 4. Automation Circuit under simulation

3 Computation procedure

Fig. 5 shows the main flowchart of ACS software tool. First, the user input is transformed into a circuit file which consists of a number of basic tables. After the circuit creation, the compilation phase follows. The run or simulation procedure can be performed only if the compilation has already been completed. During compilation, the program scans exhaustively the circuit column-by-column and element-by-element, while in parallel creates a set of basic tables containing the characteristics of the various elements as shown in Fig.6. For each type of elements, a

corresponding table is created with a definite sequence which the address counter follows during the simulation phase. The address counter and an instruction register are responsible for the Boolean logic satisfaction and can jump from table to table.

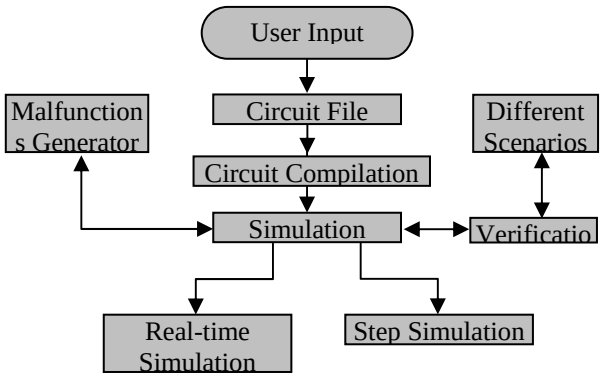

Figure 5. ACS main flowchart.

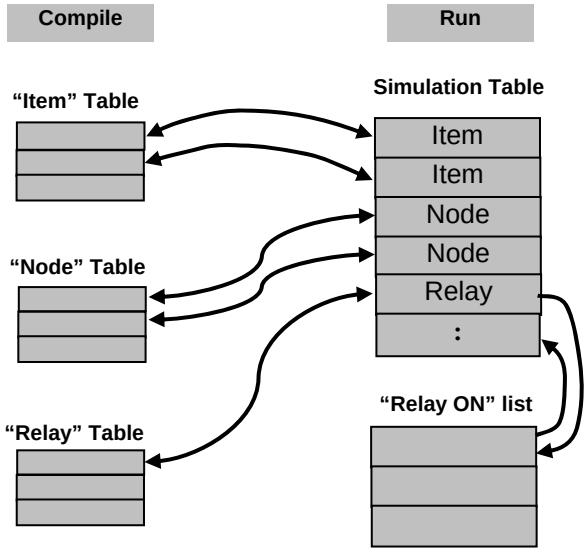

Fig. 6. Diagram of actions during simulation.

4 Conclusion

This paper presented ACS, a computer aided software tool developed for designing, simulating and verifying Relay-Ladder diagrams under alternative scenarios through a verification process.

ACS software has definite advantages, which namely are: user friendly Graphics User Interface, visualization environment with analytical mapping, ability for accommodating complex circuits and verifying all possible scenarios of operation, option for virtual creation of malfunctions, choice of real time and step by step simulation, full detailed help menu, ease of installation and use, low hardware requirements. Due to these advantages, it is considered to be ideal for educational purposes while at the same time it is suitable for industrial use too.

Acknowledgements:
This work and its dissemination efforts have been funded by the Greek Operational Programme for Education and Initial Vocational Training (O.P. Education) in the context of action 2.2.2 entitled "Reformation of Undergraduate Studies Programs".

References:

[1] M. A. Jafari and T. O. Boucher, A rule-based system for generating a Ladder logic control program from a high-level systems model, *Journal of Intelligent Manufacturing*, 1994; vol.5, pp.103-120.
[2] M. R. Lucas and D. M. Tilbury, *The Practice of Industrial Logic Design. Proceedings of the American Control Conference ACC 2004*, pp.1350-1355, Boston, Massachusetts, USA, 2004.
[3] R. Boel, Unity in Diversity, Diversity in Unity: Retrospective and Prospective Views on Control of Discrete Event Systems, Report on panel discussion of the 5th Workshop on Discrete Event Systems WODES2000, *Discrete Event Dynamic Systems: Theory and Applications*, Kluwer 2002; vol.12, pp.253-264.
[4] L. Bengtsson and L. Harju, *Modeling and verification of PLC control in buses*, Master Thesis report in Automation, EX034, Chalmers Univ. of Technology, Göteborg, Sweden, 2002.
[5] W. Yi and K. G. Larsen, UPPAAL2k: An integrated tool environment for modeling, simulation and verification of real-time systems, *www.docs.uu.se/docs/rtmv/ uppaal*; 1999.
[6] T. A. Henzinger, P. H. Ho and H.Wong-Toi, HyTech: A model checker for hybrid systems, *International Journal of software tools for technology transfer*, 1997; vol.1, pp.110-122.
[7] S. Yovine, Kronos: A verification tool for real-time systems, *International Journal of software tools for technology transfer*, 1997; vol.1, pp.123-133.
[8] K. E. Årzen, JGrafchart, Grafchart home page, *www.control.lth.se/~grafchart/*; 2002.
[9] G. O. Carlsson, *Statecharts in ABB Control*^{IT}, Master Thesis, Lund Institute of Technology, 2001.
[10] F. D. Petruzella,. *Programmable Logic Controllers*, MacMillan/MacGraw-Hill, 1991.
[11] I. Moon, Modeling programmable logic controllers for Logic verification, *IEEE control systems magazine*, Vol.14, No.2, Apr 1994,pp.53-59.
[12] A. Falcione and B.H Krogh, Design Recovery for relay ladder logic, *IEEE Control Systems Magazine*, Vol. 13, No. 2, Apr 1993, pp.90 -98.
[13] J. R. Wright, The Debate over which PLC programming language is the state-of-the-art, *Journal of Industrial Technology*, Vol.15, No.4, 1999, pp.2-5.