An experimental comparative analyse between three low capacity PLCs

G N Popa, C M Diniş, A Iagăr and I Popa
Politehnica University of Timisoara, Department of Electrical Engineering and Industrial Informatics, 5 Revolution Street, Hunedoara, 331128, Romania

E-mail: gabriel.popa@fih.upt.ro

Abstract. PLCs are used increasingly more often in industrial applications and beyond. PLCs from different companies involved different programming languages. Some of the PLCs can be programmed in all languages, while others do not. For three types of PLCs was made an application for a traffic light with priority. PLCs usually have a general and a special set of instructions that only some of these are found from a PLC to another. Programs were made with specific instruction set of respectively PLC. The programs were carried out in three languages. For a case study, it makes a comparative analysis between three programming languages.

1. Introduction

Control process aims to increase the competitiveness of a product which can be done directly, through the cost and quality, and or indirectly by improving working conditions [1-3].

Programmable logic controllers (PLCs) are the main devices of today's industrial automation. They are more and more often included in different automation. PLCs used industrial microcontroller systems (in more recent times it meet microprocessors instead of microcontrollers), where hardware and software are designed to industrial environment. Basically, PLCs are simple microcomputers are used to solve problems of sequential logic. The goal of PLCs is replacing classical electrical diagrams conducted in wired logic with logic circuits and relays. PLCs can be used successfully in most industrial applications of small or medium complexity [4-8].

The first application of PLCs was done in the automotive. The goal was to replace the traditional control electrical diagrams made with logic circuits and relays, with cheaper equipment, more reliable and more flexible. In this way, the number of electrical connections between electrical equipment was significantly reduced [1], [5].

The first language was used for programming PLCs was Ladder Diagram (LAD), because it has the advantage that it can be used by people who have no training in informatics. Over time, there were hundreds of companies producing PLCs, each has developed its own variant of LAD, different to some extent from each other. International Electrotechnical Commission (IEC) conducted a standard IEC 1131 (later IEC 61131) for standard of instructions and blocks used at PLCs languages [9-12].

They were performed and other programming languages besides LAD, graphical languages for beginner users, and textual languages for the advanced users. So, have appeared PLCs that have general functions, that are found at the most basic instructions used in all PLCs, and special functions, which enabled the introduction of special instructions that may be different from one producer to
another [2], [3]. However, working with a PLC must begin with knowledge of instructions and blocks used in programming [13], [14].

2. PLCs operation and software

Programs for PLCs are made cyclic (Figure 1) [1]. When the program starts, it checks the status of the PLC inputs. PLC’s inputs are connected to sensors (for measurement of electrical and non-electrical quantities), which can be digital (on a bit), numerically (on the more bits, eg. 8) or analogue (eg. continuous voltage). Sensors in automation, depending on the type of construction, are connected to the inputs of the PLC. Status of inputs is written in image input table (which may be register or a file).

Thereafter, the program itself is performed to scan and solve the program lines depending on the state of inputs. The results are written in the output image table (register or file). After implementing the program, results from output image table are transferred to the output where actuators are connected. Actuators can be ordered digital, numerically or analogue. Then check the internal memory, speed of execution and operation. Also in this phase is made communication with other devices (eg. an operator panel). After that, the process is repeated. Depending on the type of PLC’s, some stages of the operating cycle, may be missing or may be repeated (eg. in the same cycle of realization of the program can scan multiple times the inputs).

The scan time of PLC depending on the speed of the microcontroller (or microprocessor), the length of the program, the type of instructions executed, conditions inside the program. The PLC computes the scan time each time from the first instruction to the END instruction. Typical scan time data include the maximum scan time and the last scan time.

![Figure 1. Typical program scan cycle for a PLC](image)

PLCs can be programmed with specific programming languages (Figure 2). There are textual and graphical languages languages. Basically, graphical languages are easier than textual languages. Instruction List language (IL) has similar language with assembler language using specific mnemonics of PLC. Structured Text language (ST) is high level language, and is similar with C or Pascal languages.

Graphic languages are easier to use, especially for non-specialists in informatics. Ladder Diagram (LAD or LD) is similar with control circuit diagrams of various installations, but are rotated with 90°. Programming in this language lends to those who worked in electrical installations (electricians). LAD used the so-called networks. Function Block Diagram (FBD) closely resembles the layout of the control electronic circuits and them interconnection. It is easy to use by those who have worked with electronic devices (electronics).
Perhaps, most simple graphic language used in PLCs is Sequential Function Chart (SFC) which is similar to a flow chart that can be written by anyone. An SFC is composed with steps-actions and transitions-receptivity (Figure 3) [1]. Not all PLCs can be programmed in all languages; some may accept only one programming language (eg. IL by PS3 PLC from Klöckner Moeller).

3. Comparative analyses between three PLCs. A case study
At comparative study were used three small capacity PLCs.

The first PLC from Klöckner-Moeller is PS3. It is an old PLC (constructed in three variants) that allows programming in IL on programmable console (PRG 3S). There use specific mnemonics for programming. It has 16 digital inputs, 8 digital outputs - relays (other variant has 16 digital outputs - transistors), 4 analogue inputs and 1 analogue outputs. At programming can be used 32 on-delay timers, 32 counters, clocks, comparators, etc. It does not support extension modules, but they can be connected into network with multiple PLCs.

The second PLC is S7 214 Siemens (there are designed four variants). There are two families: CPU210 (oldest) and CPU 220 (newer). S7 214 can program the PLC in three different languages (IL, LAD or FBD, with the possibility of automatic conversion from one to another) using a RS232 to RS
485 converter. The PLC supports general functions and special functions (on-delay timers, off-delay timers, with retention timers, comparators, up-down counter, comparators, clocks, etc.). It has 10 digital inputs and 8 digital outputs. It does not have analogue inputs/outputs (which can be a major drawback in some applications). It supports extension modules (with digital or analogue inputs and outputs, with specialized modules to connect the temperature sensor module, for controlling stepper motors, communications, etc.). S7 214 can connect to industrial networks through specialized modules.

The third type of PLC is LOGO! Siemens (designed in several variants). General supports functions and special functions (on-delay timers, off-delay timers, pulses, comparators, trigger, RS flip-flop, counters, etc.). It has 12 digital inputs and 8 digital outputs. It does not have analogue inputs/outputs, but can connect with various extension modules. Programming can be done on PC or with built console (in FBD).

To make comparative analysis is considered an application with traffic lights, with the primary road and secondary road, with priority for pedestrians crossing the main path.

An intersection has two paths, a primary road and a secondary road (Figure 4). Traffic lights at the corners of the intersection have three lights: red, yellow, and green. Traffic lights are synchronized, so that if the primary road is the green light, there is red light on secondary road and vice versa. The transition from green to red is both ways via the yellow warning light. The transition from red to green is without passing through intermediate yellow light. Since the primary road passing many cars, light green on the primary road stays on for longer time than the green light on the secondary road. To eliminate the disadvantage that pedestrians wishing to cross the main road staying too long, was installed at every traffic lights a switch (normally open contact) that allows those who wish to cross the main road, to request, after some time, passing red light on the main road. Pressing the switch (Sp) has the effect of shortening the waiting time to a value that is equal to the secondary road. The four switches generate a single signal, connected to one input of PLC, which is 1 if the switch is pressed and 0 otherwise.

![Traffic lights](image)

**Figure 4.** Driving traffic lights at an intersection with priority
Table 1. The acronyms for traffic lights on primary and secondary roads

| Acronym | Description            |
|---------|------------------------|
| RLP     | Red Light Primary      |
| RLS     | Red Light Secondary    |
| YLP     | Yellow Light Primary   |
| YLS     | Yellow Light Secondary |
| GLP     | Green Light Primary    |
| GLS     | Green Light Secondary  |

Figure 5. Timing diagrams for traffic lights

In Figure 5 are timing diagrams for this application. The significance of the output is shown in Table 1. It was considered: 0-\( t_1 \)=3 min.; \( t_1 \)’-\( t_1 \)=10 s; \( t_2 \)\( t_2 \)=10 s; \( t_2 \)\( t_3 \)=1 min.; \( t_3 \)\( t_4 \)=10 s. The application starts at 0. \( t_1 \)’ is the time when push the \( S_p \) switch on traffic lights by pedestrians; \( t_1 \)’\[0-\( t_1 \)].

Figure 6. Graph automation for application
The graph of Figure 6 is implemented in the control unit. The left branch is typically traveled for traffic lights. If a pedestrian push the switch from one of the traffic lights, when green light is on the primary road, the program no longer made on the left branch and the right branch will go. The application has two inputs (one input for turning traffic lights with normally open contact with mechanical locking; at another input of PLC is connected four normally open contact switches, parallel connections) and six outputs (corresponding red, yellow, green lights for the primary and secondary roads). Each PLC's output has a normally open contact switch that is connected in series with the lamps of the traffic lights.

3.1. *Using of PLC PS3 from Klöckner-Moeller*

The input I0.0 is used to turning on the traffic lights. I0.1 connects the four switches on the four traffic lights. Table 2 shows output variables used for PS3 PLC.

**Table 2. The output variables for PS3 PLC**

|   | LP  | Q0.10 | RLS | Q0.15 |
|---|-----|-------|-----|-------|
|   | YLP | Q0.9  | YLS | Q0.14 |
| GLP | Q0.8 | GLS | Q0.13 |

|   |   |   |   |   |
|---|---|---|---|---|
| 000 | L | NH0.0 | 015 | L | M1.0 |
| 001 | A | T0.0 | 016 | S | Q0.13 |
| 002 | S | ML.0 | 017 | S | Q0.10 |
| 003 | L | T0.1 | 018 | R | Q0.14 |
| 004 | S | ML.2 | 019 | R | Q0.15 |
| 005 | L | ML.2 | 020 | R | Q0.16 |
| 006 | JC | A | 021 | R | Q0.9 |
| 007 | TR1 | 022 | L | M1.4 |
|     | S | : | ML.6 | 023 | S | Q0.14 |
|     | STOP | : | 024 | S | Q0.10 |
|     | I | W | EW1800 | 025 | R | Q0.13 |
|     | EQ | : | ML.1 | 026 | R | Q0.15 |
|     | 077 | R | Q0.8 |
| 008 | TR2 | 028 | R | Q0.9 |
|     | S | : | ML.2 | 029 | L | M1.5 |
|     | STOP | : | 030 | S | Q0.13 |
|     | I | W | EW100 | 031 | S | Q0.8 |
|     | EQ | : | ML.3 | 032 | R | Q0.13 |
|     | 033 | R | Q0.14 |
| 009 | L | ML.1 | 034 | R | Q0.9 |
| 010 | O | ML.3 | 035 | R | Q0.10 |
| 011 | = | ML.4 | 036 | R | M1.6 |
| 012 | TR3 | 037 | S | Q0.15 |
|     | S | : | ML.4 | 038 | S | Q0.9 |
|     | STOP | : | 039 | R | Q0.13 |
|     | I | W | EW100 | 040 | R | Q0.14 |
|     | EQ | : | ML.6 | 041 | R | Q0.8 |
|     | 042 | R | Q0.10 |
| 013 | TR4 | 043 | R | M1.2 |
|     | S | : | ML.5 | 044 | L | M10.0 |
|     | STOP | : | 045 | R | Q0.13 |
|     | I | W | EW500 | 046 | R | Q0.10 |
|     | EQ | : | ML.6 | 047 | L | M1.7 |
|     | 048 | = | M0.0 |

**Figure 7. IL language for PS3 PLC**
Program listing is given in Figure 7. Implementing programs in IL requires dexterity and experience. PLC programming is done the hardest in this language. Programming was done by a programmable console (PRG3S), which made it difficult to troubleshoot program. Significant limitations on program PLC in IL, arise of using one type of timer (on-delay timer, with time base of 100 ms). It used intermediate memory M1.0, M1.4, M1.5, M1.6, and M1.7 for setting two groups of lamps for traffic lights (via primary and secondary roads) according to the timing diagrams from Figure 5.

For timers were used: TR1 for $0-t_1=3 \text{ min.}$; TR2 for $t_1-t_2=10 \text{ s}$; TR3 for $t_1-t_2=10 \text{ s}$; TR4 for $t_2-t_3=1 \text{ min.}$; TR5 for $t_3-t_4=10 \text{ s}$. Timers are set by a bit, and the end of time is set other bit.

3.2. Using PLC S7 214 Siemens

The inputs I0.0 and I0.1 have the same meaning as the previous application. The output variables are presented in Table 3. The program was carried out in the LAD. Program listing is given in Figure 8. Programming was done more easily than when using PLC PS3. PLC programming on PC’s, and then transfer the program through cable greatly relieved debugging the program. Network programming is intuitive, allowing the sharing of more complex issue in simple problems to be solved. Although it can use several types of timers (on-delay timer, on-delay timer with retention, off-delay timer) were embedded in the application only on-delay timer with time base of 100 ms (there are others timers with resolution of 10 ms and 1 ms).
The program was divided into program areas, each with a well-established operation (Figure 5). It has used the technique of activation, deactivation of steps (Figure 6). For timers were used: T38 for $0-t_1=3$ min.; T39 for $t_1'-t_1=10$ s; T40 for $t_1-t_2=10$ s; T41 for $t_2-t_3=1$ min.; T42 for $t_3-t_4=10$ s.

Table 3. The output variables for S7 214 PLC

|     | RLP | Q0.0 | RLS | Q0.3 |
|-----|-----|------|-----|------|
| YLP | Q0.1| YLS  | 2.  | Q0.4 |
| GLP | Q0.2| GLS  | 3.  | Q0.5 |

3.3. Using PLC LOGO! Siemens

The input I1 has significance of I0.0, I2 has significance of I0.1 in the previous application. Output variables are shown in Table 4. The program was carried out in the FBD. Program listing is given in Figure 9. The easiest programming (and most intuitive) is performed in FBD. The plurality of blocks, especially special ones, enables easy program. It used AND logic blocks with detection on rising and falling edge for inputs and memories, even when moving from one steps to another. At the same time, it was used memory (M1-M5) for setting and resetting the steps. Programming was done relatively easily and is intuitive in view of the timing diagram from Figure 5. It uses timers that perform off-delay time. For timers were used: B01 for $0-t_1=3$ min.; B05 for $t_1'-t_1=10$ s; B07 for $t_1-t_2=10$ s; B13 for $t_2-t_3=1$ min.; B15 for $t_3-t_4=10$ s.
Figure 9. FBD language for LOGO! Siemens

Table 4. The output variables for LOGO! Siemens

| RLP | Q3 | RLS | Q6 |
|-----|----|-----|----|
| YLP | Q2 | YLS | Q5 |
| GLP | Q1 | GLS | Q4 |
4. Conclusion

From comparative analysis of the three types of PLCs (PS3 from Klöckner-Moeller, S7 214 Siemens
and LOGO! Siemens) results:
- PLCs are flexible in development of applications;
- It is easier to program in the language that was used more;
- Programming is made easier in graphic languages (LAD/FBD);
- Are useful PLCs that can be programmed in any language (eg. S7-200);
- PC programming (eg. for S7-200 and LOGO!) is more elegant than the one on programmable
  console (eg. PS 3).

References
[1] ***2011, Basic of PLC Programming, The McGraw-Hill Companies, Inc.
[2] Collins K 2010 PLC Programming for Industrial Automation, CRC Press
[3] Matik N 2008 Introduction to PLC Controllers, MikroElectronika
[4] Costano M 1997 Programmable Logic Controllers, Edward Arnold
[5] Mărgineanu I 2005 PLCs, Albastră Publishing House, Cluj-Napoca, Romania
[6] Cunțan C D, Baciu I and Osaci M 2015 Operational Study of a Frequency Converter with a
  Control Sequence, Utilizing Xilinx Software, Acta Polytehnica Hungarica 12(6) 201-212
[7] Baciu I and Cunțan C D 2012 Operation Analysis of a Frequency Converter with Control
  Realized in LabView, Proceedings of IEEE ICIT 2012, Athens, Greece, March 19-21, pp
  432-437
[8] Schütz C et al. 2013 Development of PLC-Based Software for Increasing the Dependability
  of Production Automation Systems, IEEE Transactions on Industrial Informatics 9(4) 2397-
  2406
[9] Bolton W 2010 Ladder and Functional Block Programming, www.newnespress.com 453-481
[10] Heuser B V et al. 2013 Evaluation of a UML-Based Versus an IEC 61131-3-Based Software
  Engineering Approach for Teaching PLC Programming, IEEE Transactions on Education
  56(3) 329-335
[11] Adiego B F et al. 2015 Applying Model Checking to Industrial-Sized PLC Programs, IEEE
  Transactions on Industrial Informatics 11(6) 1400-1410
[12] Isik M F et al. 2014 Monitoring and Control of PLC Based Motion, Control Systems via Device-
  Net, 16th International Power Electronics and Motion Control Conference and Exposition,
  Antalya, Turkey, September 21-24, pp 963-966
[13] Frey G and Litz L 2000 Formal Methods in PLC Programming, Proceedings of the IEEE
  Conference on Systems Man and Cybernetics SMC 2000, Nashville, USA, October 8-11, pp
  2431-2436
[14] Berger H 2011 Automatisierung mit STEP 7 in AWL und SCL, Siemens