

Journal of

INDUSTRIAL TECHNOLOGY

Volume 15, Number 4 - August 1999 to October 1999

The Debate Over Which PLC Programming Language is the State-of-the-Art

By Dr. John R. Wright, Jr.

KEYWORD SEARCH

***Computer Programming
Manufacturing
Robotics***

Reviewed Article

The Official Electronic Publication of the National Association of Industrial Technology • www.nait.org

© 1999



The Debate Over Which PLC Programming Language is the State-of-the-Art

By Dr. John R. Wright, Jr.

Dr. Wright is an Assistant Professor of Automation and Electronics in the Department of Industry and Technology at Millersville University. He is a Certified Senior Industrial Technologist and has recently concluded a post-doctoral research fellowship at the General Robotics, Automation, Sensing and Perception (GRASP) Laboratory at the University of Pennsylvania. At GRASP, he acted as the Project Manager for the Tactical Mobile Robotics Program that is currently funded by DARPA (\$475K/yr). His technical interests include robotics, programmable logic controllers, and electric/alternative vehicle design.

Abstract

This paper explores whether relay ladder logic (RLL) is still the state-of-the-art programming language for programmable logic controllers. State Logic, is presented as a possible alternative to the widely employed RLL. The debate primarily focuses upon the human-computer interactive element in programming PLCs. The implementation of a natural program language processing (NPLP) technology, utilized in the GE Fanuc State Logic product, is a worthy and much debated competitor for the state-of-the-art PLC programming title. This paper seeks to highlight the debate and discussion concerning which programming technology is ideal for programming PLCs, the most commonly encountered automated device in the manufacturing environment.

Introduction

Programmable Logic Controllers (PLCs) were first introduced into manufacturing in 1969, and are now the most widely employed industrial process control technology used today (Lauzon, Mills, & Benhabib, 1997, p. 91). Typically, PLCs are used to collect and disseminate information, to sequence robots and other automated devices, and to control conveyor systems in the industrial environment. “As with most microprocessor-based

technology, PLCs have evolved from awkward systems with rather limited capabilities and memories to state-of-the-art systems that sometimes rival distributed digital control systems in both flexibility and capability” (Dartt, 1984, p. 54). According to Hee (1995):

The National Electrical Manufacturers Association (NEMA) defines a PLC as a digitally operating electronic apparatus which uses a programmable memory for internal storage of instructions by implementing specific functions, such as logic, sequencing, timing, counting, and arithmetic to control through digital or analog I/O [Input/Output] modules various types of machines or processes. (p. 20)

“Basically, the PLC is a device that can manipulate, execute, and/or monitor data from a process or communication system” (Wright, 1998, p.1). Robert Hee, an Electronic/Electrical Engineer in private practice in Virginia Beach, Virginia, stated that PLCs have gained a substantial hold in the manufacturing environment (Hee, 1995, p. 20). A block diagram of a PLC and how it interacts with its environment is illustrated in Figure 1.

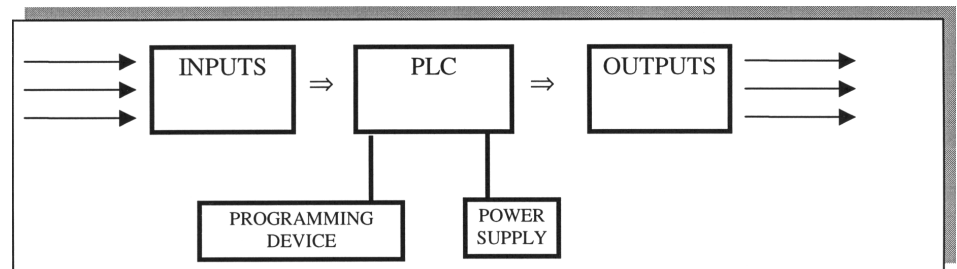
“As the PLC receives input conditions (i.e. voltage or no voltage), it examines them against the programmed code within the PLC and then executes the proper outputs associated or specified within the programmed code” (Wright, 1998, p. 1).

Relay Ladder Logic

“Typically, PLCs are programmed using a language known as Relay Ladder Logic (RLL). This language is a mixed modality language which has been around since the inception of the Programmable Logic Controller. A mixed modality language is one which uses both graphics and short syntactical statements to prompt the programmer on the use of programming functions (Wright, 1998, p. 2).

Almost any modern day graphical user interface (GUI) possesses this type of language classification. “In 1969, RLL consisted merely of input and output symbols with their respective short syntactical descriptors, and performed tasks which only required relay operations” (Wright, 1998, p. 2). By 1998, RLL had been modified to include more functions such as timers, counters, mathematical operands, and

Figure 1. PLC block diagram



communication. Figure 2 shows some typical RLL mixed modality representations which a programmer uses to select certain desired functions in a program.

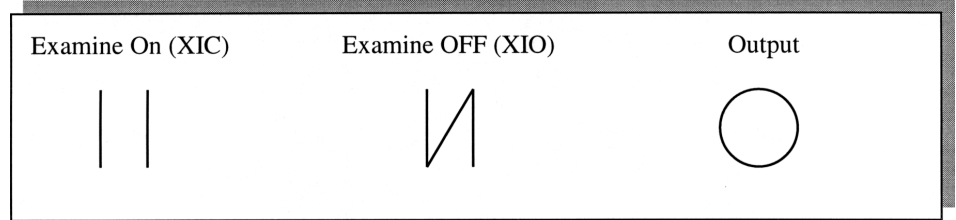
The correct mixed modality representation, for any given application, depends upon the hardware to be utilized (i.e., normal open or normal closed microswitches) and the desired action to be enacted by another device or operator. When picking the graphic symbols during RLL programming, the programmer is confronted by menus which contain both a symbol and a respective short syntactical descriptor. Therefore, RLL is considered a mixed modality language because it uses both symbols and syntactical representations to communicate a desired function to the programmer (Wright, 1998, p. 2).

State Logic

One alternative to RLL is known as State Logic. State Logic differs from RLL in two ways. It is a language which forces the user to build his/her programs with finite states. State Logic also utilizes a natural language, known as ECLiPS or English Control Language Programming Software, that allows the programmer to have the freedom of writing commands in his/her own natural words. The natural language technology differs from syntactical languages because syntactical languages require the programmer to memorize specific strings of characters to input into the programming environment. Natural languages utilize a technology known as Natural Program Language Processing (NPLP) or Natural Language Processing (NLP) to allow its users the freedom of programming a device in his/her own natural form of communication or language (Wright, 1998, p. 3). "NPLP is a branch of artificial intelligence that allows the user to interact with the system in his or her own natural language" (Fountain, 1994, p. 85).

Programming this type of language consists of writing text statements, in the programmer's natural language (i.e., English) within the software program. The programmer then executes a menu choice which, in turn, initiates a check through the words

Figure 2. Examples of RLL's mixed modality representations



used. Similar to the way a word processor performs a spell check, any words that are not known to the ECLiPS software are flagged and identified to the user. The system then asks the user to define any word that it does not recognize by selecting a synonym from its existing bank of terminology. For instance, the word "power" may not be instantly recognized by the ECLiPS software. The computer would then ask the programmer to identify a similar word which means the same thing. The programmer might select the term "execute" from the existing knowledge base and it will now understand the meaning of the word "power." The term "power" is then added to its large bank of terminology and will be recognized in all future programming (Wright, 1998, pp. 3-4). Natural languages, such as the

State Logic/ECLiPS product, may learn new words, even if they are misspelled or misrepresented. State Logic may also be programmed in French, German, and Spanish. It is even rumored that the creators of the language, engineers at Adatek Inc., even programmed a PLC running the State Logic system in Klingon (Star Trek). Several companies which have implemented this PLC programming technology are listed along with their targeted applications in Table 1 (State-logic workshop, p. 3).

The Debate

The method of programming which was originally introduced with the PLC is known as Relay Ladder Logic (RLL). The advantage of using RLL is described by Crater (1997):

Table 1. Where the state logic system is being used?

Company	Application
Baxter Labs	FDA Validated Pharmaceutical Application
Monsanto	Fiber Processing
Hewlett-Packard	High Volume Manufacturing
Goodyear	Press Control
John Deere	Facilities Control
3M	Environmental Control
Johnson & Johnson	Production Line Control
General Mills	Packaging Machinery Control
Ford	Automated Test Stands
Bio Engineering Firm	Biotech Fermentation
Leviton	Light Switch Test Stand
Stanley Tools	Machine Control
Phillips Lighting	Vapor Tube Manufacturing
AMOCO	System Diagnostics
Consolidated Hydro	Power Generation
Fiskars	Scissors Manufacturing
AT&T Micro Electric	Product Line Control
IBM	Discrete Manufacturing
Wilson Technology	Compressed Natural Gas Delivery
Inner Tite	Discrete Manufacturing

The principal advantage offered by combinational languages is the existence of trained technicians. There is a long tradition of expressing machine control programs in combinational terms, using ladder logic or Boolean expressions. These languages use a limited range of symbols, and perform simple associations using them, which makes modest combinatorial programs easy to understand for a broad range of individuals. (p. 91)

Although Crater (1997) describes RLL as a symbolic type of language, it does utilize short syntactical prompts in combination with symbols. Guastello et al. (1989) would describe this as a mixed modality type of classification. According to Pollard's (1994) article "Ladder Logic Remains the PLC Language of Choice," there are ten reasons why RLL has been around for a long time and has not out-lived its usefulness. These reasons include:

1. RLL is symbolic and picture-like
2. it is very simple to interpret
3. control engineers are familiar with it
4. maintenance personnel can understand it
5. it executes quickly
6. it is productive for design and troubleshooting disciplines
7. it has vast third-party software and vendor support
8. it allows for on-line programming with real-time compilation
9. each of its instructions is an object, allowing for future developments and integration
10. it easily allows for user extensions. (p. 77)

"Approximately half of the above reasons stated by Pollard (1994) focus upon the human-interface element in using the programming software" (Wright, 1998, p. 32). Where programming times are concerned, RLL enthusiasts insist that symbols are easier to use than text. According to Dick Hollbeck, President of Tele-Denken Resources, "You can't get more symbolic than ladder logic. There is no real reason to abandon it" (Pollard, 1994, p. 78).

"Today, relay ladders are long gone, but RLL is still with us in most of the PLCs currently on the market. It has become the de facto standard like the English system of measures—outdated yet too widely used to replace entirely" (VanDoren, 1996, p. 110). Over the past 30 years, RLL has been modified to keep up with the increasing demands of industry's control needs. RLL was originally designed for easy use and understanding of its users. At the time these users consisted primarily of electricians (Wright, 1998, p. 34).

VanDoren (1996) stated the following: *Unfortunately, that plan backfired. RLL did prove to be familiar to the electricians who had formally been responsible for creating relay ladder circuits, but RLL programs longer than a few rungs quickly became incomprehensible to anyone else. Programming a PLC with RLL turned out to be much like writing a doctoral dissertation with hieroglyphics—the results could be absolutely brilliant, but no one else would ever be able to truly understand it, much less improve upon it. (p. 110)*

"While ladder logic is still the industry standard programming language, the trend is toward state logic, sequential function charts, graphics, and versions that are programmed in Basic, C, or other high-level languages" (Plcs speak, 1995, p. 158). Brookings (1991) stated the following:

Relay ladder logic (RLL) is great for the jobs that it fits. Even in those jobs it doesn't quite fit, a skilled and clever user with plenty of time can often make it work. However, if you started with a clean piece of paper to design the software tool to solve most automatic control problems most effectively, it probably would not be RLL. (p. 36)

State Logic was created for today's modern control needs. With the use of State Logic, "program development and modification time is decreased substantially compared to traditional control methodologies because the program is a direct description of the

desired system" (Brookings, 1991, p. 38). The added ability of artificial intelligence, specifically NPLP, allows increased flexibility to syntactical-based programming languages (Wright, 1998, p. 34). Walker (1992) states the value of considering a natural language for computer programming:

Standardized interfaces will soon be needed to permit greater expansion of computers throughout society, no matter what the task. Natural language interfaces represent one possible solution to the impediments to communication created by the many extremely varied computer interfaces we now have. By a natural language interface, we mean one that communicates like a human being, and has a two-way interaction with the user in full English sentences. (p. 294)

According to (Wright, 1998) the use of a natural language, specifically the ECLiPS software, produced a savings of 71.5% in mean program development times for novice PLC programmers. This comparison was done against another GE Fanuc product, Logimaster RLL, which represented the mixed modality language classification. The study did not account for the finite state capability of the State Logic product, as this capability was intentionally factored out of the comparison (small programs were developed which were limited to one rung of RLL or one state of State Logic) as to isolate the "true" contribution of the NLP technology versus a mixed modality interface.

Summary

Although mixed modality languages yield many advantages over languages which use syntactical statements or graphics alone, they still require memorization and recognition of their associated functions. Natural languages do not require this association process. In 1989, Coury & Pietras stated the need for examining task presentation in a dynamic process plant environment:

... there remains a need to provide an answer to the fundamental question facing system designers:

What is the appropriate representation for system information? Although there are a number of guidelines for designing control room displays ([Frey & Sides, 1984; Rouse & Hunt, 1984]), there remains a paucity of research evidence to indicate the appropriateness of a particular type of display format (or format combination) for presenting process plant data under specific conditions. The question still remains whether the formats currently in use or proposed for control systems are appropriate for the information needs of the operator. (p. 1373)

The use of a natural language on the factory floor might provide a simpler platform for the systems operators. Information communicated in the operator's own natural language may reduce the learning curves associated with programming, thereby increasing an industry's productivity. The PLC is the most widely employed automated control device on the factory floor. As industrial technologists, we strive to continually stay abreast of the new technology and recommend those which "best" fit the environment to which they will be implemented. Due to the much varied and argued debate presented in this paper, and the nature of the PLC technology that is so widely employed, a call for more applied research in the form of statistical studies is needed if we are to resolve which technology is truly the state-of-the-art.

References

- Brookings, K. R. (1991, March). State logic: a paradigm for integrated control in the 90s. Control Engineering 38(5), 36-39.
- Coury, B. G., & Pietras, C. M. (1989, November). Alphanumeric and graphic displays for dynamic process monitoring and control. Ergonomics 32(11), 1373-1389.
- Crater, K. (1997, June). Plc programming: state language is the way to go. Instrumentation & Control Systems 70(6), 91-92.
- Dartt, S. R. (1984, February). Programmable logic controller survey: comparison of top systems. Pulp & Paper 58(2), 54-58.
- Fountain, W. (1994, February). State logic speeds plc programming. Instrumentation & Control Systems 67(2), 83-85.
- Frey, P. R. & Sides, Jr., W. H. (1984). Computer generated display system guideline, vol. 1: display design. The Electric Power Research Institute, Palo Alto, CA. (EPRI Report No. NP-3701)
- Guastello, S. J., Traut, M., & Korienek, G. (1989). Verbal versus pictorial representations of objects in a human-computer interface. International Journal of Man-Machine Studies 31, 99-120.
- Hee, R. B. (1995, October). Knowing the basics of plcs—part 1. Electrical Construction and Maintenance 94(10), 20-28.
- Lauzon, S. C., Mills, J. K., & Benhabib, B. (1997). An implementation methodology for the supervisory control of flexible manufacturing workcells. Journal of Manufacturing Systems 16(2), 91-101.
- Plcs speak in many tongues. (1995, April 20). Machine Design, 67(8), 158.
- Pollard, J. R. (1994, April). Ladder logic remains the plc language of choice. Control Engineering 41(5), 77-79.
- Rouse, W. B., and Hunt, R. M. (1984). Computer generated display system guideline, vol. 2: developing an evaluation plan. The Electric Power Research Institute, Palo Alto, CA. (EPRI Report No. NP-3701)
- State-logic workshop roll the rock. GE Fanuc/Crescent Electric Supply Company.
- VanDoren, V. (1996, June). Designing plc-based control without ladder logic. Control Engineering 43(8), 110.
- Walker, S. (1992, October). The value of a natural language capability in the computer. Behavioral Science 37(4), 294-309.
- Wright, J. R. (1998). Natural language and mixed modality task presentations in the human-computer interaction using programmable logic controllers. Ph.D. Thesis, Iowa State University.