Programmable Logic Controllers (PLCs): Workhorse of Industrial Automation

Article · March 2010

2 authors, including:

Sajid Iqbal
Harbin Institute of Technology

40 PUBLICATIONS **101** CITATIONS

SEE PROFILE

"New Horizons"

Journal of
The Institution of
Electrical & Electronics
Engineers Pakistan

VOL # 68-69 April -Sept 2010

Published by
Engr. Muhammad Anwar Khalid
President
for
The Institution of
Electrical and Electronics
Engineers Pakistan
4-Lawrence Road, Lahore
Phone: (042) 3630 5289
Fax: 042 36360287
Email:info@ieeep.org.pk
Website.www.ieep.org.pk
ieeep.org

Disseminate Technical Knowledge

Conserve Electricity

CONTENTS

Programmable Logic Controllers (PLCs): Workhorse Of Industrial Automation

Sajid Iqbal[a] and Ahsan Wasim[b]
[a]Deptt of Elect. Engg. University of Gujrat.
[b]Deptt. of Elect. Eng. UET, Lahore.

Abstract

Programmable Logic Controller (PLC) also referred to as Programmable Controller (PC) is a solid-state, digital, and industrial computer used for automation of electro-mechanical processes, such as control of machinery on factory assembly lines. Earlier electro-mechanical relays were used to perform logic functions. Unlike personal computers (PCs), the PLCs are designed for multiple inputs and outputs, extended temperature ranges, immunity to electrical noise, and resistance to vibration and impact. Higher-level languages ease the programming task for large systems. PLCs and their unique language, ladder logic, are the workhorses of factory automation.

Keywords: PLC, Ladder Logic, Function block diagram (FBD), Structured text (ST), Instruction list (IL), Sequential function chart (SFC)

I. INTRODUCTION

The National Electrical Manufacturers Association (NEMA) defines a programmable logic controller (PLC) as: a programmable controller is a digitally operating electronic apparatus which uses a programmable memory for the internal storage of instructions for implementing specific functions, such as logic, sequencing, timing, counting and arithmetic, to control through digital or analog input/output (I/O), various types of machines or process [1].

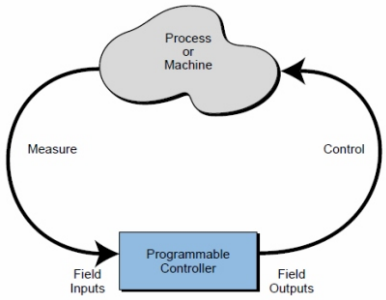

Fig. 1 PLC Conceptual Application Diagram

Fig. 1 shows a conceptual diagram of a PLC application [2]. Often, a single PLC can be programmed to replace thousands of relays. Software to control machines is stored in a battery-powered or non-volatile memory. A PLC is an example of a real-time system since output results must be produced in response to input conditions within a bounded time, otherwise unintended operation will result. These are used for control and operation of manufacturing process, equipment and machinery. It monitors inputs, make decisions based on its program and controls outputs to automate a process or machine [2,4].

II. HISTORY OF PLC

In the 1960s, a typical automated assembly or other manufacturing line had a cabinet of relays wired to control the operation. Before the PLC, control, sequencing, and safety interlock logic for manufacturing automobiles was accomplished using relays, cam timers, and drum sequencers and dedicated closed-loop controllers. The process for updating such facilities for the yearly model change-over was time consuming and expensive, as electricians needed to individually rewire each and every relay. Troubling-shooting (debugging) relay failures was also tedious when so many relays were involved [3,4].

PLCs were developed to replace hard-wired relay logic (*Relay logic* is a method of controlling industrial electronic circuits by using relays and contacts). In 1968, the Hydramatic Division of General Motors Corporation (GM) specified design criteria for the first PLC. Bedford Associates of Bedford won the proposal. The first PLC was designated as *the 084* because it was Bedford Associates' eighty-fourth project. Dick Morley is considered to be the *father of the PLC* as he was one of the people who worked on that project. They Bedford Associates started a new company for dedicated to developing, manufacturing, selling, and servicing this new product: Modicon (MOdular DIgital CONtroller). The *Modicon 084* brought the world's first PLC into commercial use. It is still a popular brand of PLC today, now owned by Schneider Electric. Other well-known PLC brands are Allen-Bradley, ABB, IDEC, Mitsubishi, Siemens, Omron, Rockwell Automation and General Electric [3-4].

The first PLCs offered relay functionality, thus replacing the original hardwired relay logic, which used electrically operated devices to mechanically switch electrical circuits. They met the requirements of modularity, expandability, programmability, and ease of use in an industrial environment. These controllers were easily installed, used less space, and were reusable.

The programming, although a little tedious, had a recognizable plant standard, the ladder diagram format.

Initially most PLCs utilized Ladder Logic Diagram Programming, a model which emulated electromechanical control panel devices (such as the contact and coils of relays) which PLCs replaced [2-3]. The automotive industry is still one of the largest users of PLCs. In a short period, PLCs spread to other industries. By 1971, PLCs were being used to provide relay replacement as the first steps toward control automation in other industries, such as food and beverage, metals, manufacturing, and pulp and paper [2].

III. BASIC PLC OPERATION

PLCs consist of input modules, a central processing unit (CPU) and output modules. Input modules accept a variety of signals (digital or analog) from various sensors and convert them into logic signals that can be used by the CPU. The CPU is the *brain* of the system. It reads input data, executes the stored user program and sends appropriate commands to control devices (output modules - actuators) [4].

The input switch contacts like push-buttons, limit switches, or pressure or temperature sensors have negligible resistance. The output element could be any resistive load e.g., a relay coil, lamp, motor or any other device that can be electrically actuated. The basic operations of a PLC correspond to the combinational control of a logic circuit. In addition, a PLC can carry out other operations such as counting, the processing of signal delays, and a wait for defined time intervals [4]. A PLC must operate in real time. The input and processing of external signals can take place in two ways in a PLC, by polling (repeated requests) or via interrupt signals.

IV. PLC ARCHITECTURE

The architecture of a general PLC is shown in Fig. 2. The main parts of a PLC are its processor, power supply, and input/output (I/O) modules [4].

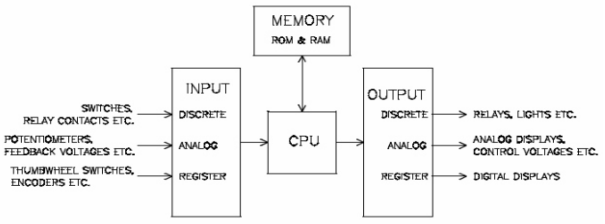

Fig. 2 PLC Block Diagram

The *processor unit* or *central processor unit (CPU)* contains the microprocessor which interprets the program commands retrieved from memory and acts on those commands. The *power supply* unit is needed to convert the AC line voltage to the low DC voltage. The *input module* has terminals into which outside process electrical signals generated by sensors or transducers are entered. The input signals can come from a wide variety of devices, i.e., push--buttons, rotary switches, limit switches etc. The *output module* has terminals to which output signals are sent to activate relays, solenoids, motors and displays. Output modules are generally of two types, *discrete* and *analog*. *Discrete output modules* fall into two classifications, *solid-state switching* and *relay output switching* [3-5].

ADVANTAGES OF PLC

The major advantage of a PLC is that a single circuit with a compact construction can replace a hundred of relays. Secondly, a PLC is programmable and not hardwired so that its operation can be changed with limited effort. On the other hand, PLCs can be slower than hardwired- relay logic [10].

The architecture of the PLC is basically the same as a PC. The main difference from other computers is that PLCs are armored for severe conditions (such as dust, moisture, heat, cold) and have the facility for extensive input/output (I/O) arrangements. These connect the PLC to sensors and actuators. PLCs read limit switches, analog process variables (such as temperature and pressure), and the positions of complex positioning systems. On the actuator side, PLCs operate electric motors, pneumatic or hydraulic cylinders, magnetic relays, solenoids, or analog outputs [7]-[10].

The PLC hardware is built to fit a typical industrial environment, especially regarding heat, humidity, electrical noise, electromagnetic interference (EMI), unreliable power supplies and mechanical shocks and vibrations [5,6,7].

They are easy to use by plant technicians. Hardware interfaces are easily connected. Modular and self-diagnosing interface circuits pinpoint malfunctions and are easily replaced. They are programmed using ladder logic, which is easy to learn. The PLC executes a single program in an orderly and sequential fashion. However, large PLCs have instructions that allow subroutine calling, interrupt routines, and the bypass of certain instructions [5,6,7].

VI. PLC PROGRAMMING LANGUAGES

PLC programming consists of mainly defining control sequences. PLC programs are typically written in a special application on a PC, and then downloaded by a direct-connection cable or over a network to the PLC. The program is stored in the PLC either in a battery-backed-up RAM or some other non-volatile flash memory.

The International Electro-technical Commission (IEC) is a non-profit, non-governmental international standards organization that prepares and publishes International Standards for all electrical, electronic and related technologies collectively known as *electro-technology* [7]. Developed with the input of vendors, end-users and academics, IEC 1131 consists of five parts:

1. General information
2. Equipment and test requirements

3. PLC programming languages
4. User guidelines
5. Communications

IEC 61131-3 is the third part of the open international standard IEC 61131, and was first published in December 1993 by the IEC. The current (second) edition was published in 2003. It specifies the syntax, semantics and display for the following suite of PLC programming languages:

1. Ladder diagram (LD), graphical
2. Function block diagram (FBD), graphical
3. Structured text (ST), textual, similar to the Pascal programming language
4. Instruction list (IL), textual, similar to Assembly language
5. Sequential function chart (SFC), has elements to organize programs for sequential and parallel control processing.

All of the languages share IEC61131 common elements. The variables and function calls are defined by the common elements, so different languages can be used in the same program. One of the primary benefits of the standard is that it allows multiple languages to be used within the same programmable controller. This allows the program developer to select the language best suited to each particular task. While the fundamental concepts of PLC programming are common to all manufacturers, differences in I/O addressing, memory organization and instruction sets mean that PLC programs are never perfectly interchangeable between different makers. Even within the same product line of a single manufacturer, different models may not be directly compatible [7-9].

A. Ladder Logic (LD)

The automotive industry was a major early adopter of PLCs. They wanted a programming method that could be easily understood by their existing controls engineers and technicians. The result of this desire was a programming language called *Relay Ladder Logic* or *ladder logic*.

Ladder logic is a programming language that represents a program by a graphical diagram based on the circuit diagrams of relay-based logic hardware. The name *ladder diagrams* is based on the observation that programs in this language resemble ladders, with two vertical rails and a series of horizontal rungs between them i.e., these programs look like the rungs on a ladder [2-3].

The layout of Ladder Logic is very similar to reading the diagrams for hard wired relay controls. Ladder Logic is still one of the most popular languages for programming PLCs, but many others have developed over the years. With ladder logic, an imaginary relay network is described. If such network were in fact real, the desired control would take place. A ladder logic diagram must be read as symbols not as switch contacts [2-4].

The ladder logic symbology was developed from the relay ladder logic wiring diagram. Consider the simple problem of turning on a lamp when both switches A and B are closed, as shown in Fig. 3(a). Also note that the output for the rung occurs on the extreme right side of the rung and power is assumed to flow from left to right. One would interpret the rung symbology as: When input (switch) A is ON and input (switch) B is ON then the lamp is ON [3].

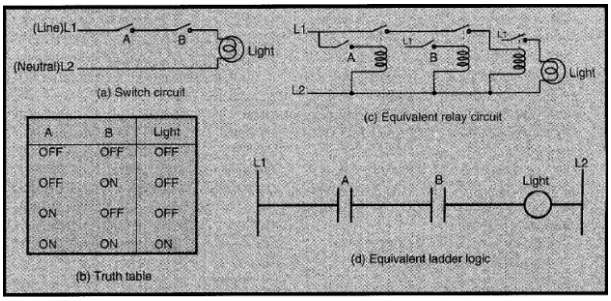

Fig 3 Series Circuit a) Switch circuit, b) Truth table, c) Relay logic, d) Ladder logic

Each rung has a connection to the left (power) rail and a connection to the right (neutral) rail. The rungs of a ladder diagram are *executed* simultaneously in a wiring diagram. Each *rung of* the LD must contain at least one output element; otherwise, a short circuit between power supply and ground will take place. In reality, the ladder logic diagram is only a symbolic representation of the computer program. So, power does not really flow through any actual contacts; however, the concept of power flowing through contacts is useful when explaining the program operation [3].

The three basic ladder logic symbols are: Normally open (NO) contact, Normally closed (NC) contact and Output (relay coil). The *normal state* is the one in which the coil is energized. The output is energized whenever any left-to-right path of input contacts is closed [3]. PLC evaluated the rungs sequentially, usually from top to bottom and from left to right.

One aspect of ladder logic that is often confusing is the use of the NC contact. The contact symbol in the ladder does not necessarily correspond to the actual switch type used in the field. After all, the PLC does not know how the switch is wired in the field, only whether the switch is open (off) or closed (on). So, a NO switch does not require a -I I- in the ladder logic and a NC switch does not require a -I I- in the ladder logic. Regardless of the type of switch in the field, when one wants *action* (something to be logically true, or on) when the switch is closed (on), use the -I I- symbol.

When one wants *action* (something to be logically true, or on) when the switch is open (off), use the -I I- symbol. One must eventually learn to read a ladder logic diagram as symbols and not as relay contacts [3].

There are three classes of ladder logic instructions: *input/contact instructions* and *output/coil instructions*. Input instructions are the contact instructions replace a contact instruction. These instructions are the conditions to turn on the output. In contrast, an output instruction always occurs on the extreme right side of the rung [3-4].

Not all instructions are contacts or coils. All other types of instructions are often called *function blocks* or *box instructions* because that is how they appear in the symbology. Timers, counters, comparison, and computation sequencers, shift registers, and data move instructions are all the most common box instructions.

B. Function Block Diagram (FBD)

is a graphical language for a PLC or a Distributed Control System (DCS). A FBD, as shown in fig. 4, is a diagram that describes a function between input variables and output variables. A function is described as a set of elementary blocks. Input and output variables are connected to blocks by connection lines. An output of a block may also be connected to an input of another block.

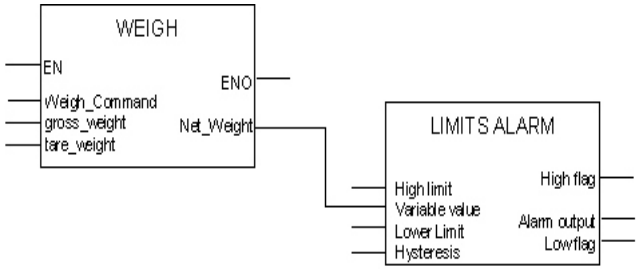

Fig. 4 Function Block Diagram Language

The primary concept behind a FBD is data flow. In these types of programs the values flow from the inputs to the outputs, through function blocks. FBDs use logic gates (AND/OR) for digital signals and numeric function blocks (arithmetic, filters, controllers for numeric signals [2,8,9].

C. Structured Text (ST)

is a structured high level language and syntactically resembles Pascal, BASIC and C. It allows structured programming, meaning that many complex tasks can be broken down into smaller ones i.e., it uses subroutines to perform different parts of the control function and passes parameters and values between the different sections of the program. A simple example is shown in Fig 5 [2,8,9].

```
PROGRAM main
Var
i:INT;
END_VAR
i:=0;
REPEAT
i:=i+1;
UNTIL i>=10;
END_REPEAT;
END_PROGRAM
```

Fig. 5 A Structured Text Example Program

ST also supports iterations, such as WHILE-DO and REPEAT-UNTIL, as well as other conditional executions, such as IF-THEN-ELSE. Problem-oriented ST programming is particularly suited to applications involving data handling, computational sorting, and intensive mathematical applications utilizing floating-point values. It supports Boolean operations (AND/OR). It is also the best language for implementing artificial intelligence (AI) computations, fuzzy logic, and decision making [2,8,9].

D. Instruction List (IL)

Is a low level language like Assembly language. It contains simple mnemonics such as LD, AN, ADD, etc. It is the most fundamental level of programming language - all other programming languages can be converted to IL programs. This type of Assembler-like IL language is useful for small applications, as well as applications that require speed optimization of the program or a specific routine in the program. It is useful in cases where small functions are repeated often and is used to create custom function blocks. Although it is powerful, it is considered to be difficult to learn. Fig 6 shows an IL [2,8,9].

```
LD A
AND  B
STO   LIGHT
```

Fig 6. An Instruction List for Series Circuit of Fig.2

E. equential Function Chart (SFC)

is a graphical technique for writing concurrent control programs. It is defined in IEC 848, *Preparation of function charts for control systems*, and was based on Grafcet (Graphe Fonctionnel de Commande Étape Transition). The three main components of an SFC are steps, actions and transitions. *Steps* are merely chunks of logic, i.e., a unit of programming logic that accomplishes a particular control task. *Actions* are the individual aspects of that task. *Transitions* are the mechanisms used to move from one task

to another. Control logic for each Step, Action and Transition is programmed in one of the other languages such as Ladder Diagram or Structured Text. SFCs are suited to processes with parallel operations [2,8,9].

Grafcet is a symbolic, graphic language, which originated in France that represents the control program as steps or stages in the machine or process. In fact, the English translation of Grafcet means *step transition function charts*. It is the foundation for the IEC 1131 standard's SFCs, which allow several PLC languages to be used in one control program. SFCs have been developed to accommodate the programming of more advanced systems. These are similar to flowcharts, but much more powerful. Fig. 7 shows a selection branch, in which only one branch is executed depending on which transition is active [2,8,9].

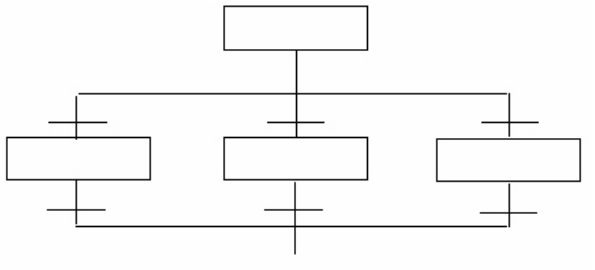

Fig 7 SFC Selection Branch

Basically, Grafcet employs only written action statements whereas SFC implements actions in a number of ways using LD, IL, ST, and FBD or a combination of these languages, including custom function blocks.

VII. CONCLUSION

PLCs are at the forefront of manufacturing automation. Many factories use PLCs to cut production costs and or increase quality of products. IEC 61131-3 deals with programming languages and defines two graphical and two textual PLC programming language standards. When Ladder Logic was seen unable to meet the automation needs of the 21st century, high-level languages emerged. Although other languages are used, ladder logic presently remains the dominant language of automation.

An engineer working in a manufacturing environment will at least encounter PLCs and ladder logic, if not use them on a regular basis. Electrical engineering students should have some know how of PLCs because of widespread use of PLCs in domestic and industrial applications.

REFERENCES

[1] National Electrical Manufacturers Association. http://www.nema.org/

[2] L. A. Bryan & E. A. Bryan, *Programmable Controllers Theory and Implementation.* Industrial Text Company. 2nd Edition. 1997.

[3] Erickson, K.T., "Programmable Logic Controllers", IEEE Potentials. Vol. 15, Issue 1. 1996. pp: 14 17.

[4] John R. Hackworth & Frederick D. Hackworth, Jr. *Programmable Logic Controllers: Programming Methods and Applications.* Pearson Education, Inc. 2004.

[5] John W. Webb & Ronald A. Reis, *Programmable Logic Controllers: Principles and Applications.* Prentice Hall. 5th Edition. 2002.

[6] Gary Dunning, *Introduction to Programmable Logic Controllers.* Thomson learning Inc. 1st Edition. 1998.

[7] International Electrotechnical Commission. http://www.iec.ch/

[8] E.A. Parr, *Programmable Controllers,* Reed Elsevier Plc Group. 3rd edition. 2003.

[9] Hugh Jack. *Automating Manufacturing Systems with PLCs. EBook.* Version 5.0. 2007. *http://claymore.engineer.gvsu.edu/~jackh/books.html*

[10] Handbook of Networked and Embedded Control Systems. Hristu-Varsakelis, Dimitrios; Levine, William S. (Eds.). 1st ed.. A Birkhäuser Boston book. 2005

****