Programmable logic controllers: A course in programming and implementation

Miguel A. Ortiz
Graduation candidate

Abstract

The purpose of this paper is to provide basic information related to programmable logic controllers (PLCs), their applications and programming. It covers the historical background and an overview of the programmable controller plus brief discussions of its input/output system and peripheral devices and programming languages. The document was designed to serve as an instructor's guide for a fifth year engineering course and continuing education at a graduate level.

Sinopsis

Este artículo provee información básica sobre los controladores de lógica programable, sus aplicaciones y distintos métodos de programación. En el documento se presenta el trasfondo histórico y una descripción de los controladores de lógica programable, sus sistemas de entrada/salida y equipo periferal y los lenguajes de programación. El artículo se redactó como guía para un curso de quinto año de ingeniería y como base para educación continuada a nivel graduado.

Historical background

The National Electrical Manufacturers Association (NEMA) defines a PLC as a "digital electronic device with a programmable memory for storing instructions to implement specific functions such as logic, sequencing, timing, counting and arithmetic to control machines and processes." Even though a personal computer (PC) can be used for control functions, it cannot stand the harsher environments where a PLC operates. In fact, the PLC is less flexible but more rugged than the PC.

In 1968, the Hydramatic Division of General Motors Corporation requested, by definition, an electronic machine to replace the multiple relays used in the mass manufacture of automobiles and their parts. According to the simplified specifications by General Motors, the electronic machine must:

1. be easily programmed and reprogrammed with a minimum of down time

2. be easily maintained while rugged enough to operate in an industrial environment

3. consume less power and require less cabinet and floor space than a relay control system

4. be competitive in cost and have expandable memory

5. communicate with the data collection systems and accept 120 volt AC signals.

During the 1970s, the developments in microprocessor technology provided a more powerful and flexible PLC. Mathematics, data manipulation and communications were improved. Memory capacity was increased and communications with host computers was introduced.

Overview of the programmable controller

A programmable controller is composed of two basic sections: the central processing unit (CPU) and the input/output (I/O) interface system. The CPU has three major elements: the power supply, the microprocessor and the memory.

The power supply provides power for the CPU and, normally, to the peripherals. It is designed to operate in areas of high electromagnetic interference (EMI). The output of the power supply is filter-regulated and short-circuit-protected.

The microprocessor, as used in a PLC, is of the same type as those used for a personal computer (PC). There are several microprocessors available, among them the Z80, 8080, 8086, 6800, 80286, 68000 and the 9900 families. The microprocessor controls all activities of the PLC. The basic differences between a PC and a PLC are in the nature of the power supply, noise protection circuitry, programming language and nature of input/output circuitry.

Microprocessors are generally categorized by word size. A greater word length allows for faster manipulation of information, as more data are handled in one operation. It receives and manipulates external information from the input modules and then updates the outputs. This sequence is known as a scan. The scan time is measured in milliseconds per kilobyte. The input signals should not be quicker than the scan time of the microprocessor for the input data to be processed.

The implementation of specific functions in a PLC is provided by the use of a programmable memory. The executive memory provides the operating system assigned by the manufacturer and normally resides in a read-only-memory (ROM). The executive memory needs a temporary storage of data that is provided by the random-access memory (RAM), which is not accessible to the user. The I/O status memory is in a portion of a RAM that is assigned for the storage of the current I/O status and is part of the executive or operating system. The application memory is used to fulfill the control function and it is subdivided into data memory and user memory. The data memory, generally contains preset values for counters and timers, and instructions for data manipulation of mathematical functions. Since this is a changing situation, the memory used is a RAM. On the other hand, the user memory is the most accessible to the programmer and provides the programmable feature of the PLC. The microprocessor scans the user memory by the use of the address bus, which provides a particular location, and the data bus, which makes available the data stored in the memory. Because of the continuously changing nature of the data input, this memory is also a RAM.

The input/output system of the PLC

A. The input modules

The I/O system provides the interface between the PLC and the outside world, either from the inputs or to the outputs. The most common interface is the digital or discrete type. The input signals are either Open (O) or Closed (1). Some discrete input devices are limit switches, push-buttons, photoelectric devices, level switches, relay contacts, reed switches and selector switches. These devices receive their operating voltage from the PLC system and it may vary from 24 volts AC/DC depending on the specific design. Another type of discrete input devices is the transistor-transistor-logic (TTL) level or a contact closure.

The AC/DC input interface converts the external signal, which normally carries undesired electrical components, into an acceptable signal to be used by the processor. A typical AC/DC input interface is composed of two primary parts: the power section and the logic section. The power section performs the function of converting the relatively high voltage to a DC value by using a bridge rectifier. A logic circuit determines whether the signal has reached the threshold level to accept it as valid. Then an optical isolator transmits the signal to the processor (fig. 1). A variation to this type of interface is the isolated AC/DC input module, which has a separate return line for each input terminal instead of a common terminal for a group of inputs, which is the case for the standard AC/DC input module.

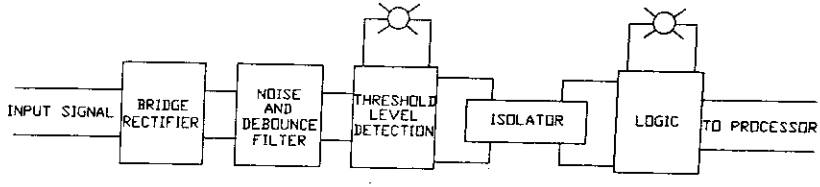

Figure 1. Block diagram of the AC/DC input circuit

On the other hand, the DC input module does not contain a bridge circuit, because conversion to DC is not necessary. Different from AC/DC input modules, the DC module can interface in the current sinking or sourcing mode. A device that provides current is the sourcing element whereas the one that receives current is the sinking element.

Another type of input module to the PCL is the TTL input module, which provides the interface between the controller and TTL compatible devices. Normally, TTL devices are faster than the standard AC/DC input. This module can be used successfully with binary coded decimal (BCD) thumbwheel switches which operate at TTL levels.

An input interface can also be provided by the numerical data interface module. This module handles multiple bits, such as the BCD, which is represented by four bits of data. An example of a multiple bit input would be the representation in a thumbwheel of the number 5 as 0101.

The input interface could also be the non-voltage input module. This module does not require that the field device provide power when energized. A "dry contact" is a typical representation. Solid state relays and instrumentation outputs with open collectors are also part of this classification.

Finally, the analog input interface module is part of the modern trend of PLCs input types. Signals from thermocouples, pressure transducers, resistance temperature detectors (RTDs) and strain gauges are representative analog inputs. These signals are converted from analog to digital and then fed to the processor in order to obtain the desired output.

B. The output modules

The discrete output module is the most common type of output interface. This module interconnects between the controller and the output field devices. The controlled devices will exhibit a digital or discrete operation (ON/OFF). Some of these devices are alarms, solenoids, relays, valves, lights and fans. The voltage characteristics are similar to those described for the input modules, although they can be different in a system. For example, the input can be 24 volts DC, while the outputs can be 120 volts AC. The voltage characteristics can also be mixed in the input or output modules, meaning that a device or group of devices can operate with one voltage characteristic while other groups can operate with different characteristics, within the same system.

Figure 2 shows another type of output interface. The diagram presents a general description of the AC output module, whose configuration varies between manufacturers. This circuit consists of the logic and power sections, coupled by an isolator circuit. It operates as a switch through which power is provided to control the output device. When an ON signal is fed into the output module, it is coupled through the optical isolator, which in turn will switch the voltage through the power section to the field device. The switching section uses a triac, a silicon controlled rectifier (SCR), or a relay, protected by an R-C snubber or by a metal oxide varistor (MOV).

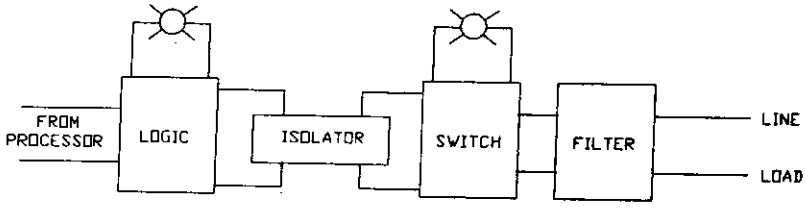

Figure 2. Block diagram of the AC output circuit

Another type of output interface is the DC output module. This module has the same characteristics of the DC input module. It also operates similar to the AC output except for the fact that its output element generally employs a transistor to switch the load. For protection of the transistor, a "free-wheeling" diode should be connected at the load. A fuse is also recommended to protect the module from moderate overloads.

The TTL output module is an output interface used to directly drive TTL output devices such as seven segment LED displays and integrated circuits. The numerical data interface provides parallel communication between the processor and the output device. This output may have multiplexing capabilities, where several groups of outputs may be controlled with one interface.

Lastly, the contact output is simply a switch which can turn ON or OFF its load. The contacts are magnetically coupled, so isolation from the load to controller is obtained. This output type can be used for switching AC or DC loads of moderate current requirements.

Peripheral interfaces

The most common peripheral interfacing devices communicate in serial form at speeds ranging from 110 to 19200 bits per second. The most popular of this group is the EIA RS-232C, followed by the EIA RS-422. Other interface methods are the IEEE 488 instrument bus, the PDP-11 Unibus and the 20 mA current loop.

Programming languages

Programming languages have evolved since the inception of the PLC in the early 1960s. Nevertheless, some of the basic languages have maintained their original standings with some necessary changes and additions. The most popular of these languages is the relay ladder logic format (RLL). As more sophisticated equipment has been introduced, more versatile instructions have also evolved. Many of these instructions are enhancements to the original ladder format. Other instructions deal with the transferring of data to and from the I/O modules.

The functional descriptions of the various instructions presented in this article will provide an understanding of their operation. The instructions, although covered in a generic nature, are of the same shape, form and function in most programmable controllers.

Types of instructions

The basic requirements in the development of the programmable controller were the effective representation of the program logic needed to control a machine or process and the ease of programming. Based on these concepts, the RLL was the first one implemented. Evolution of this language provided more powerful instruction sets.

Most industrial processes require the completion of several operations to produce the required output. Machining manufacturing, finishing, assembling and transporting of products require the precise coordination of several tasks for an economical system to function. Electronic controls have provided the necessary coordination and monitoring of these industrial processes in the past, and with more sophistication today. These controls can be divided into two general categories: sequential and combinatorial. Sequential controls are required for processes which demand that certain operations be performed in a particular order. Combinatorial processes do not require this order.

Basic ladder logic instructions

Earlier sequence controls used electromechanical multicontact relays in switching circuits. Logic functions were performed by state assignment of open or closed relay contacts. The closed contact was usually the true or 1 state. The open contact was the false or O state. Figure 3 shows some elementary relay contact connections and their equivalent logic functions.

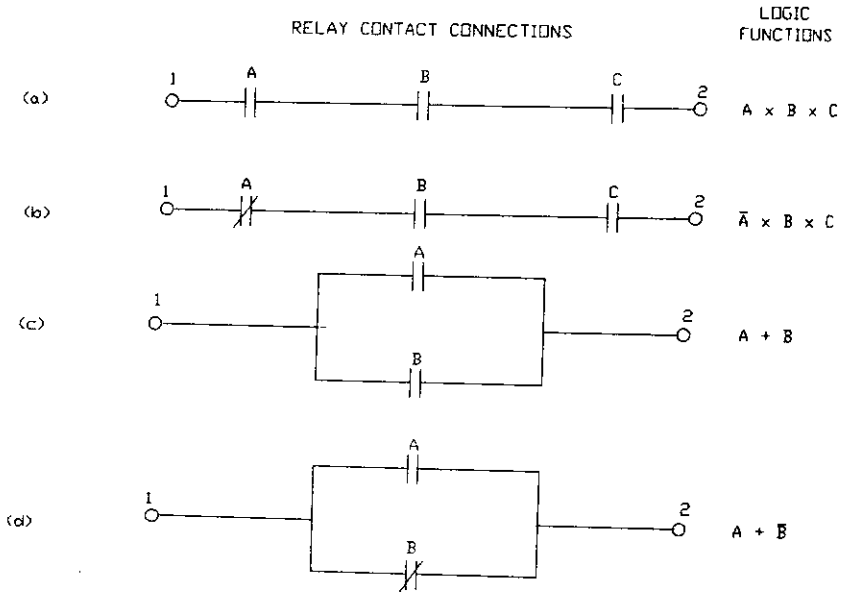

Figure 3. Relationship between relay contact connections and their logic functions

It is very important to observe the interpretation of the symbols. They are basic to the understanding of the RLL. In figure 3 (a) continuity between points 1 and 2 is granted if relays A and B and C are energized. The symbology of figure 3 (b) means that there will be continuity between points 1 and 2 if relay coil A is not energized and relay coils B and C are energized. Contacts A, B and C correspond to a relay coil of the same name. Figure 3 (c) shows that there will be continuity between points 1 and 2 if either relay coil A or B is energized. On the other hand, according to figure 3 (d), continuity between points 1 and 2 is granted if relay coil A is energized or relay coil B is not energized. The entire operating sequence of a system can be determined from the circuit diagram showing the interconnection of all the relay coils, relay contacts and actuating devices. The diagram is usually shown in ladder form (fig. 4).

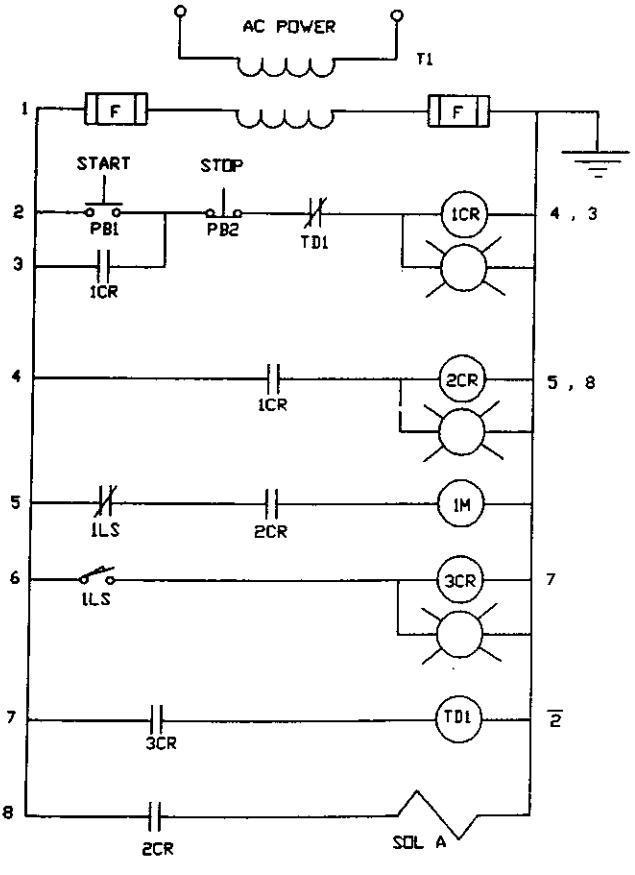

Figure 4. Ladder diagram of a circuit

The secondary terminals of the control transformer extend vertically downward to form the boundary lines for the diagram. Field input devices attach from left to right, while output devices attach to the right leg. This right leg, designated as the common line, is normally wired to ground. Each horizontal line is numbered at the left side in sequence for easy referencing. The numbers next to each relay coil indicate the lines on which the relay contacts are used. If the contact of reference is a normally closed contact, a line is marked either above or underlining the line number depicting the location. The operating sequence can be listed by following the ladder

diagram line by line. The sequence shown in figure 4 is described below:

Line 2 Start push-button is actuated; 1CR is energized if 3CR is unenergized and the stop push-button is not actuated.

Line 3 Contact 1CR closes, latching relay coil 1CR even after the start push-button is released.

Line 4 Contact 1CR closes, energizing relay coil 2CR if limit switch 1 is not actuated.

Line 5 Contact 2CR closes, energizing motor control 1M if limit switch 1 is not actuated.

Line 8 Contact 2CR closes, energizing solenoid A which might control air or hydraulic fluid.

Line 6 Limit switch 1 is actuated, energizing relay coil 3CR.

Line 5 Contact 1LS opens de-energizing relay coil 1M, shutting off the motor.

Line 7 Contact 3CR closes, energizing the time delay coil, TD1.

Line 2 After a predetermined time delay, contact 1TD opens, de-energizing relay coil 1CR.

Line 4 Relay contact 1CR opens, de-energizing relay coil 2CR.

Line 8 Contact 2CR opens, de-energizing solenoid coil A to end the sequence.

Because sequential systems require that a specific order of operations be performed if a desired outcome is to be realized, the simple techniques used to design combinatorial systems are no longer adequate, inasmuch as the time element must now be considered. Nevertheless, most requests for electronic control systems come in the form of word statements, specifications or manufacturing process statements. The first step is converting these statements into the language of digital logic. Figure 5 shows the basic symbols used for ladder logic programming.

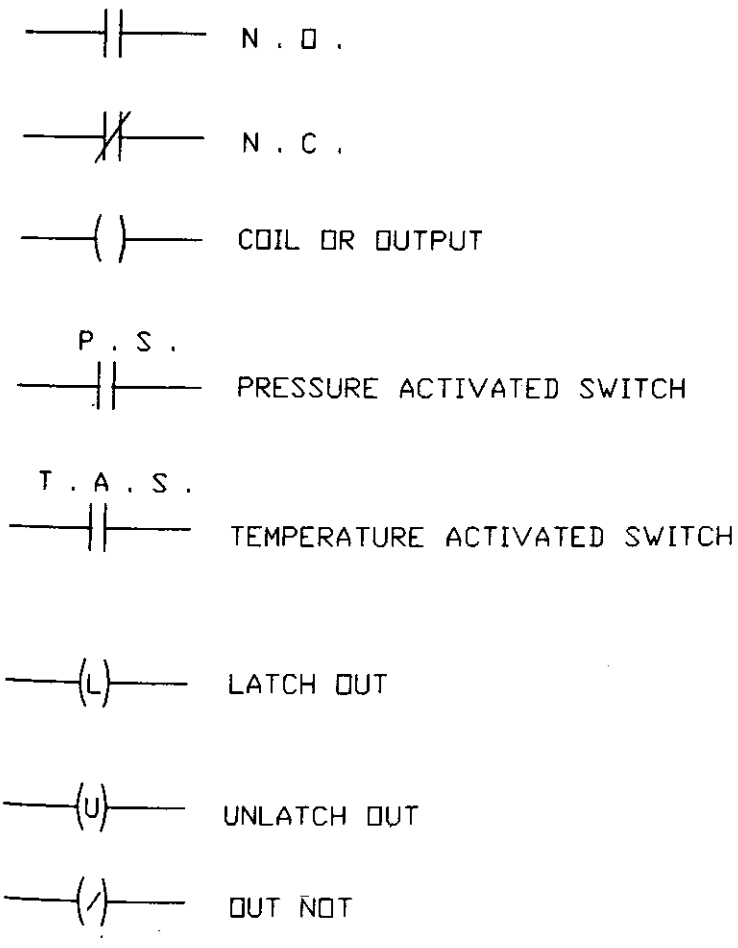

Figure 5. Basic symbols for ladder logic programming

The first three symbols in figure 5 are the normally open, normally closed and output symbols. The last three symbols , latch out, unlatch out, and out not are commonly known as relay-type instructions. The two middle symbols represent specific device contacts. Recalling that the PLC is in part designed to replace relays, the first three symbols need to be presented differently.

The N.O. contact indicates that a specific signal is required to close the contact and complete the current flow path through the rung of the ladder. The symbol is not a representation of a real contact. It represents a no input signal status. The N.C. contact represents a normal flow of current through the rung when there is no input signal. (The word normally refers to the status of a contact when it is neither electrically nor mechanically activated). The output symbol represents a coil, but it might as well represent other output devices such as motors, lamps or solenoids.

The out and out not relay-type instructions are the normally open and normally closed contact counterparts. The out is on when the corresponding input signal is on. The out not is on when there is no corresponding input signal. The inverse holds true in both cases. The latch out and unlatch out simulate a latching relay which consists of two coils; the latching and the unlatching coils, respectively. When the latching coil is activated, the corresponding N.O. contacts close and the N.C. contacts open. The contacts remain in that condition even after deactivating the latch coil. The contacts return to their normal condition when power is applied to the unlatch coil. A short pulse is all that is required to toggle each of the coils.

Programming symbols vary slightly from manufacturer to manufacturer but the differences are usually minor. The output symbols for example, have been shown up to this point as a circle or as a set of parentheses. The latter is used at computer terminals, whereas design drawings are normally shown with circles.

Boolean mnemonic language

Boolean mnemonic language is the second most popular programming language behind the ladder logic system. The popularity behind this method of programming is derived from the fact that it is interchangeable with the ladder logic method. This equivalency can be confirmed by carefully studying figure 6.

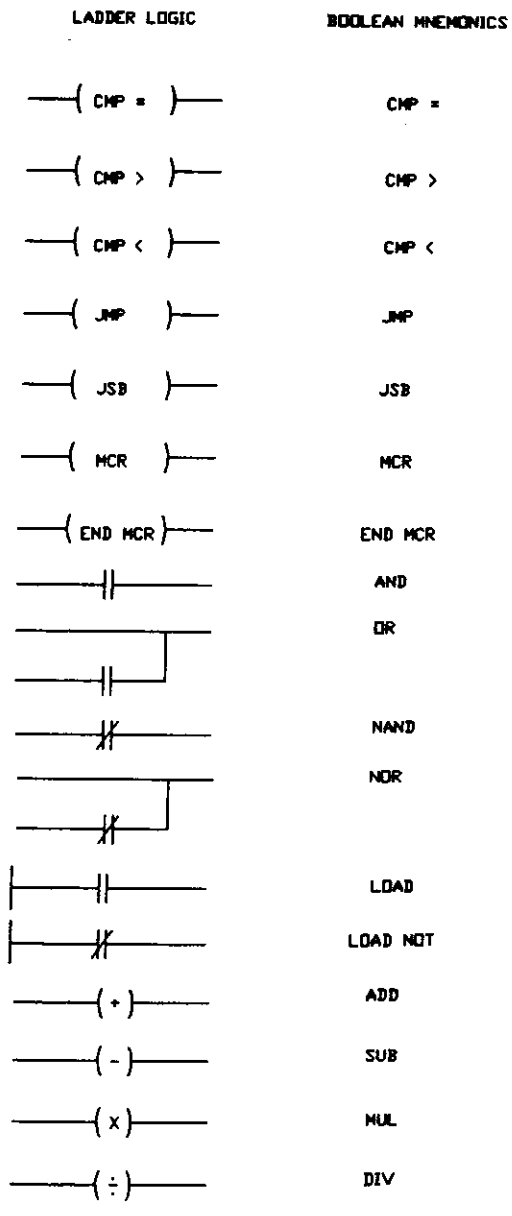

Figure 6. Equivalency between ladder logic and Boolean mnemonics

In many cases the Boolean characters are the same as those in the ladder format, except for some symbols that are not used in the Boolean system. Observe the compare, jump, jump-to-subroutine, and master control relay instructions.

Some symbols that deserve special attention are the OR, NOR, LOAD and LOAD NOT instructions. The OR instruction is a normally open contact in parallel with a rung, thereby providing continuity to the output through alternate paths. The NOR is connected in the same format and performs in opposite form to the OR as it employs a normally closed contact. The LOAD and LOAD NOT instructions are used to start programming from a rung with a normally open and normally closed contacts, respectively. Some manufacturers used the word START instead of LOAD. It is a good practice to study the manufacturers instruction manuals in order to become familiar with their particularities.

The four arithmetical symbols shown in figure 6 are addition, subtraction, multiplication and division. These operations can be performed on memory data, on constants or on both.

Enhanced programs

Microprocessor technology improvements which have appeared subsequently in the market at a reasonable cost have also provided modern programmers with subroutine blocks which can be introduced into the simple ladder logic diagram. These blocks contain more complicated built-in instructions. The blocks can represent the GET and PUT instructions, timers, counters, compare, jump, shift registers and more complicated functions like proportional, integrals and derivative loops (PID).

The GET instruction (fig. 7) includes a number above the contact symbol. This number indicates the memory address reference, where data are stored.

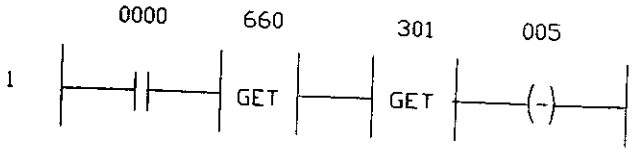

Figure 7. Example of a subroutine block

Transducers like the temperature, flow, pressure or level types can feed data to these addresses to be operated upon according to the program in memory. The GET instruction is then used to retrieve the stored information.

Going back to figure 7, the program flow for subtraction of data between two registers can be followed. (The memory locations for the temporary storage of data, instruction, or information are often called registers). When the control contact at the left is closed, datum at register 660, and the difference is stored at register 005.

Figure 8 illustrates operations involving data and constants. When contact 0000 closes, the contents of register 660 is multiplied by the constant 10; the product is stored in register 522. When the control contract 0001 is closed, the product previously stored at register 522 is divided by the contents of register 220 and the quotient is then stored at register 330.

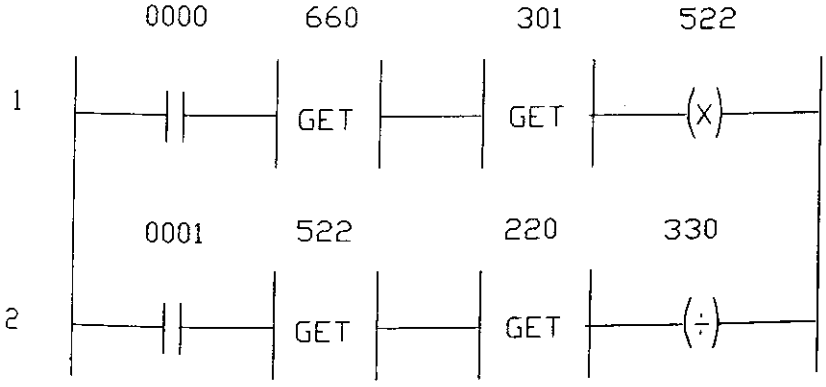

Figure 8. Operations involving data and constants in a subroutine block

Companion to the GET instruction is the PUT instruction. This instruction is shown in figure 9 as an output with the corresponding register location on top. The put instruction is used to store outputs into a particular register or to move data from one particular register to another.

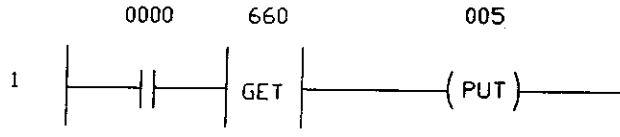

Figure 9. Illustration of the use of the GET and PUT instructions

High-level languages

Block symbol language

Block diagrams can be used to form high-level languages to allow the programmer to implement complex routines still using the simple ladder diagram approach. The block diagram symbols are introduced into the ladder diagrams as if they were simple output elements. The symbols are placed in the corresponding rung of the ladder diagram, where a permissive signal is required to let the operation be carried out.

Figure 10 shows the most frequent block diagram symbols in use today. The first block corresponds to the timer symbol. The control input is programmed in series with the corresponding control contacts to permit starting of operation upon contact closure. The timer will operate until the preset time is reached and then provides an output by closing all its programmed contacts. This particular configuration shows the timer operating internally at 1 second intervals up to the preset value of 10 seconds. At that elapsed period of time, its output will then be activated. Upon application of a signal to the reset contact, the timer status will return to the original condition; it is reset to 0 (zero) and disabled until the next control signal is received. The reset input is not provided in most of the configurations sold today, because the timer resets at the time the control signal is removed.

The counter shown in the second block of figure 10 includes a direction. In other words, it will count down from the preset value of 30 in single decrements per input signal pulse until the output is obtained at the value of 0 (zero). When the reset input is present, the counter returns to the preset value of 30. To change the counting direction, the word down is replaced by the word up.

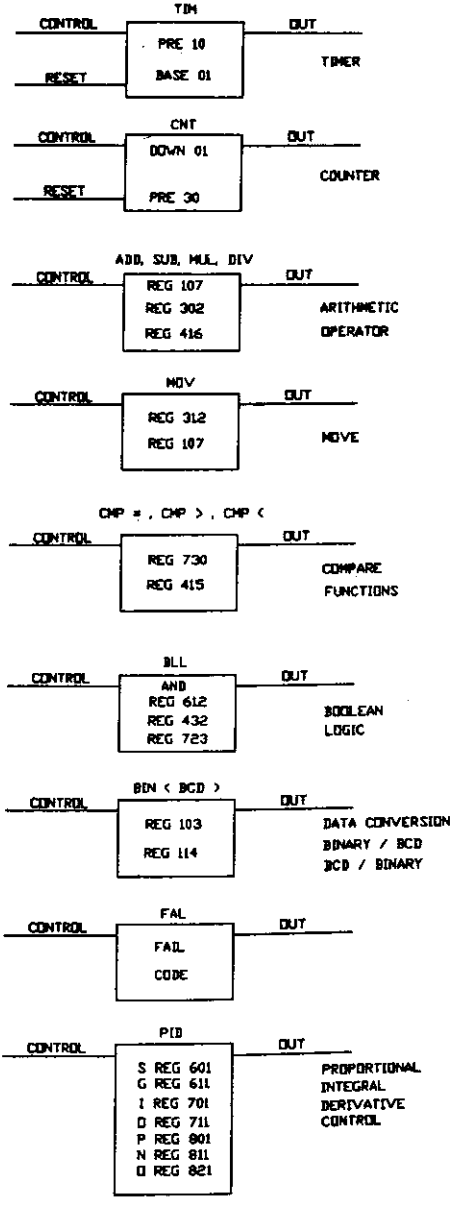

Figure 10. Block diagram symbols

The third block corresponds to the four arithmetical operations. When the control signal is present, the information contained in registers 107 and 302 (data or constants), is used to perform the indicated arithmetical operation, and the result is placed in register 416. The output line is used for overflow indication. Higher level mathematics is used in modern equipment in similar fashion.

The fourth block, MOVE, performs the same operation as the previously presented combination of GET and PUT in figure 9. When the control signal is present, the stored information at register 312 is relocated to register 107. Many instruction manuals provide other variations for data transfer instructions. Some of the more frequently used are shift register, word shift, move not, arithmetic shift left/right and block move.

The fifth block in figure 10 represents the compare functions. The information in the first two registers are compared and if the selected condition is satisfied, the output is energized.

The sixth block represents the Boolean block symbol and it performs the AND operation. It makes use of registers 612 and 432. When the control signal is on, the operation will be completed by storing the result at register 723 and energizing the output. Some manufacturers do not assign the third register to store the results. In this case the second register is used to store the result at the expense of destroying the previously stored information, by overwriting.

The last block in figure 10 refers to the proportional-integral-derivative (PID) module. Table 1 shows the module connection points and their corresponding functional description.

Table 1. Module connection points

Connection	Function	Register
S	process set point	601
G	desired gain value	611
I	input from process	701
O	output control signal	711
P	proportional constant	801
N	integral constant	811
D	derivative constant	821

The manufacturing sector in Puerto Rico is using this concept through PLCs instead of using discrete instrumentation. This way it can reduce cost, space and down-time in order to improve troubleshooting efficiency and providing a basis for management-oriented information and data collection systems.

Computer-type languages (CTL)

Computer-type languages make use of the English language and are similar to Basic. They are operator-oriented and therefore user-friendly. The variations between this language and the Basic language must be verified from the particular PLC manufacturer's literature.

The program can be read from a ladder diagram and transferred directly into CTL. If we were to program the ladder rung below (fig. 11) using CTL, the program would look like this:

Figure 11. Ladder diagram for CTL program below

```
10 READ X
20 IF X=0  YHRN Y=0
30 IF X=1  THEN Y=1
```

Although simple, this program illustrates how easy it is to transfer from one language to the other, using the same source of information. Mathematical instructions like SQR, LOG, SIN, and COS are used for Real time computation.

Two principal types of languages used to program PLCs have been discussed: low-level and high-level languages. The ladder-type language, upon which others in the same category as well as those known as high-level are built, was found to be the most popular. This language was developed in

order to provide a smooth transition between the original relay ladder diagrams and the then new development of the PLC in 1968. Boolean mnemonic language is the second most popular language. Nevertheless, they both are limited in flexibility on the basis of user demands.

Block language and computer-type-languages (CTLs) form the high-level group discussed in this presentation. Although this paper is not intended as a substitute for a formal textbook, it covers the general composition of PLC programming languages.

Proposed syllabus for a course in programmable controllers

I. General information

Course title	:	Programmable controllers Theory and applications
Codification	:	EE-XXX
Credits	:	3
Timetable	:	4 hours per week
Pre-requisites	:	EE-435 (Automatic control systems) EE-536 (Microprocessors II)

II. Course description

The course will cover the development and evolution of the programmable controller (PLC); programming instructions used by the different manufacturers; conversion from hardware to software logic; computer-like capabilities of the PLC; high level languages; programming devices and peripheral; selection, applications and installation; industrial analog devices and actual programming throughout the course with real PLCs.

III. Course objectives

Upon completion of this course, the student will have valuable skills and knowledge to be able to:

1. Name and understand the major applications, advantages and benefits of a typical programmable logic controller.

2. Name the major components necessary for the implementation in a typical industrial installation.

3. Read, understand and design ladder diagrams and other high level languages.

4. Provide a solution, from analysis to design, of a real case.

IV. Course outline

A. Brief review of microprocessors and number systems.

B. History of the PLC

1. Origins and development
2. Contrasts and similarities with the personal computer
3. The PLC industry

C. Applications

1. Selection of a system
2. Control schemes
3. Reliability

D. Hardware

1. Input/output (I/O) structure
2. Input/output (I/O) configuration
3. The CPU
4. Power supplies

E. Programming

1. Relay ladder logic
2. Extended relay instructions
3. Timers and counters
4. Mathematics and number conversion

F. Peripherals and accessories

1. Peripherals and support
2. Programming devices
3. Communications and accessories

G. Selection and specifications

H. Assembly, installation and maintenance

I. Selected programs

V. Textbook

Programmable Controller Handbook
Robert E. Wilhelm, Jr.
Hayden Book Company, 1985

Note: This course will be complemented with 20 hours of in-class training sessions in programming of a commercial PLC.

A field trip to an industrial installation is advisable.