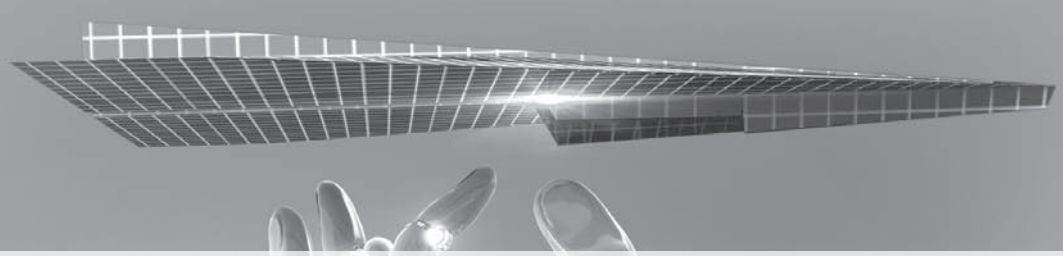


Chapter fourteen

Programmable logic controllers

Objectives

The objectives of this chapter are that, after studying it, the reader should be able to:

- **Describe the basic structure of PLCs and their operation.**
- **Develop ladder programs for a PLC involving logic functions, latching, internal relays and sequencing.**
- **Develop programs involving timers, counters, shift registers, master relays, jumps and data handling.**

14.1 Programmable logic controller

A **programmable logic controller** (PLC) is a digital electronic device that uses a programmable memory to store instructions and to implement functions such as logic, sequencing, timing, counting and arithmetic in order to control machines and processes and has been specifically designed to make programming easy. The term logic is used because the programming is primarily concerned with implementing logic and switching operations. Input devices, e.g. switches, and output devices, e.g. motors, being controlled are connected to the PLC and then the controller monitors the inputs and outputs according to the program stored in the PLC by the operator and so controls the machine or process. Originally PLCs were designed as a replacement for hard-wired relay (e.g. Figure 9.2) and timer logic control systems. PLCs have the great advantage that it is possible to modify a control system without having to rewire the connections to the input and output devices, the only requirement being that an operator has to key in a different set of instructions. Also they are much faster than relay-operated systems. The result is a flexible system which can be used to control systems which vary quite widely in their nature and complexity. Such systems are widely used for the implementation of logic control functions because they are easy to use and program.

PLCs are similar to computers but have certain features which are specific to their use as controllers. These are:

1 they are rugged and designed to withstand vibrations, temperature, humidity and noise;
2 the interfacing for inputs and outputs is inside the controller;
3 they are easily programmed.

14.2 Basic PLC structure

Figure 14.1 shows the basic internal structure of a PLC. It consists essentially of a central processing unit (CPU), memory and input/output interfaces. The CPU controls and processes all the operations within the PLC. It is

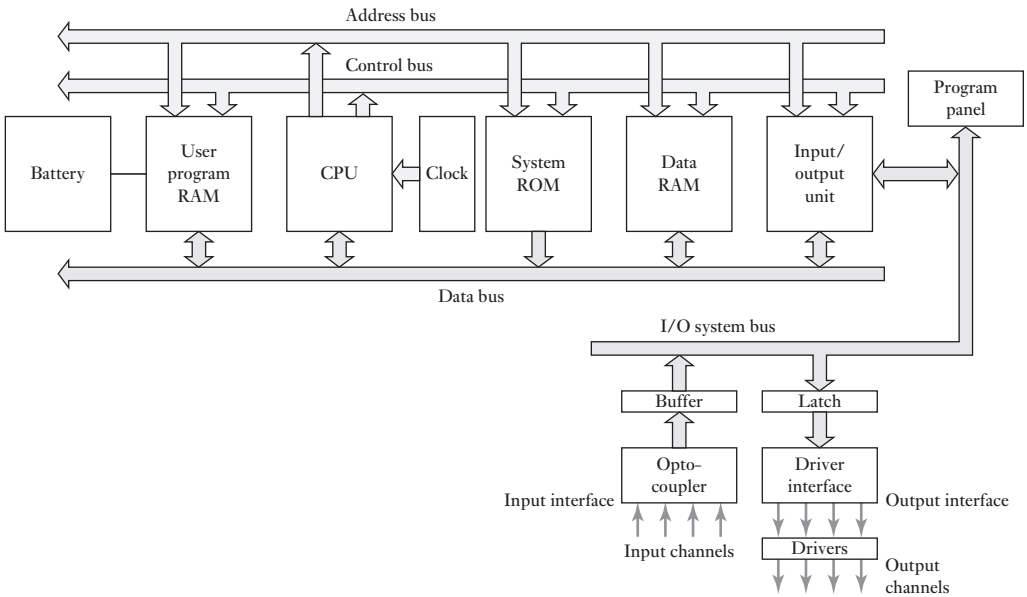


Figure 14.1 Architecture of a PLC.

supplied with a clock with a frequency of typically between 1 and 8 MHz. This frequency determines the operating speed of the PLC and provides the timing and synchronisation for all elements in the system. A bus system carries information and data to and from the CPU, memory and input/output units. There are several memory elements: a system ROM to give permanent storage for the operating system and fixed data, RAM for the user's program and temporary buffer stores for the input/output channels.

14.2.1 Input/output

The input and output units provide the interface between the system and the outside world and are where the processor receives information from external devices and communicates information to external devices. The input/output interfaces provide isolation and signal conditioning functions so that sensors and actuators can often be directly connected to them without the need for other circuitry. Inputs might be from limit switches which are activated when some event occurs, or other sensors such as temperature sensors, or flow sensors. The outputs might be to motor starter coils, solenoid valves, etc. Electrical isolation from the external world is usually by means of optoisolators (see Section 3.3).

Figure 14.2 shows the basic form of an input channel. The digital signal that is generally compatible with the microprocessor in the PLC is 5 V d.c. However, signal conditioning in the input channel, with isolation, enables a wide range of input signals to be supplied to it. Thus, with a larger PLC we might have possible input voltages of 5 V, 24 V, 110 V and 240 V. A small PLC is likely to have just one form of input, e.g. 24 V.

The output to the output unit will be digital with a level of 5 V. Outputs are specified as being of relay type, transistor type or triac type. With the relay type, the signal from the PLC output is used internally to operate a relay and so is able to switch currents of the order of a few amperes in an external

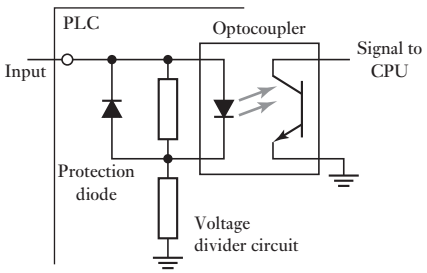


Figure 14.2 Input channel.

circuit. The relay isolates the PLC from the external circuit and can be used for both d.c. and a.c. switching. Relays are, however, relatively slow to operate. The transistor type of output uses a transistor to switch current through the external circuit. This gives a faster switching action. Optoisolators are used with transistor switches to provide isolation between the external circuit and the PLC. The transistor output is only for d.c. switching. Triac outputs can be used to control external loads which are connected to the a.c. power supply. Optoisolators are again used to provide isolation. Thus we can have outputs from the output channel which might be a 24 V, 100 mA switching signal, a d.c. voltage of 110 V, 1 A or perhaps 240 V, 1 A a.c., or 240 V, 2 A a.c., from a triac output channel. With a small PLC, all the outputs might be of one type, e.g. 240 V a.c., 1 A. With modular PLCs, however, a range of outputs can be accommodated by selection of the modules to be used.

The terms **sourcing** and **sinking** are used to describe the way in which d.c. devices are connected to a PLC. With sourcing, using the conventional current flow direction as from positive to negative, an input device receives current from the input module (Figure 14.3(a)). If the current flows from the output module to an output load then the output module is referred to as sourcing (Figure 14.3(b)). With sinking, an input device supplies current to the input module (Figure 14.3(c)). If the current flows to the output module from an output load then the output module is referred to as sinking (Figure 14.3(d)).

The input/output unit provides the interface between the system and the outside world, allowing for connections to be made through input/output channels to input devices such as sensors and output devices such as motors and solenoids. It is also through the input/output unit that programs are entered from a program panel. Every input/output point has a unique address which

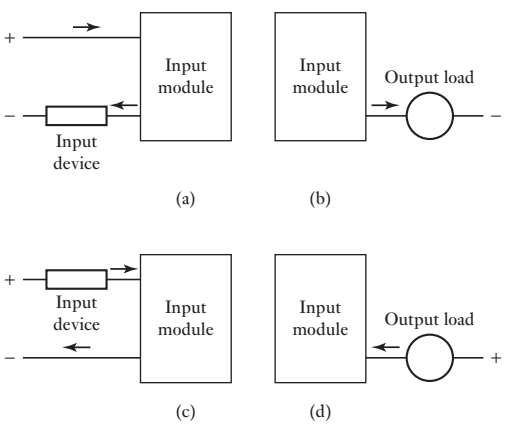


Figure 14.3 (a), (b) Sourcing, (c), (d) sinking.

can be used by the CPU. It is like a row of houses along a road: number 10 might be the 'house' to be used for an input from a particular sensor while number '45' might be the 'house' to be used for the output to a particular motor.

14.2.2 Inputting programs

Programs are entered into the input/output unit from small hand-held programming devices, desktop consoles with a visual display unit (VDU), keyboard and screen display, or by means of a link to a personal computer (PC) which is loaded with an appropriate software package. Only when the program has been designed on the programming device and is ready is it transferred to the memory unit of the PLC.

The programs in RAM can be changed by the user. However, to prevent the loss of these programs when the power supply is switched off, a battery is likely to be used in the PLC to maintain the RAM contents for a period of time. After a program has been developed in RAM it may be loaded into an EPROM chip and so made permanent. Specifications for small PLCs often specify the program memory size in terms of the number of program steps that can be stored. A program step is an instruction for some event to occur. A program task might consist of a number of steps and could be, for example: examine the state of switch A, examine the state of switch B, if A and B are closed then energise solenoid P which then might result in the operation of some actuator. When this happens another task might then be started. Typically the number of steps that can be handled by a small PLC is of the order of 300 to 1000, which is generally adequate for most control situations.

14.2.3 Forms of PLCs

PLCs were first conceived in 1968. They are now widely used and extend from small self-contained units, i.e. single boxes, for use with perhaps 20 digital input/outputs to rack-mounted systems which can be used for large numbers of inputs/outputs, handle digital or analogue inputs/outputs, and also carry out proportional plus integral plus derivative (PID) control modes. The single-box type is commonly used for small programmable controllers and is supplied as an integral compact package complete with power supply, processor, memory and input/output units. Typically such a PLC might have 6, 8, 12 or 24 inputs and 4, 8 or 16 outputs and a memory which can store some 300 to 1000 instructions. For example, the MELSEC FX3U has models which can have 6, 8, 12 or 24 inputs and 4, 8 or 16 relay outputs and a memory which can store some 300 to 1000 instructions. Some systems are able to be extended to cope with more inputs and outputs by linking input/output boxes to them.

Systems with larger numbers of inputs and outputs are likely to be modular and designed to fit in racks. These consist of separate modules for power supply, processor, input/output, etc., and are mounted on rails within a metal cabinet. The rack type can be used for all sizes of programmable controllers and has the various functional units packaged in individual modules which can be plugged into sockets in a base rack. The mix of modules required for a particular purpose is decided by the user and the appropriate ones then plugged into the rack. So the number of input/output connections can be increased by just adding more input/output modules. For example, the SIMATIC S7-300/400 PLC is rack mounted with components for the power supply, the

CPU, input/output interface modules, signal modules which can be used to provide signal conditioning for inputs or outputs and communication modules which can be used to connect PLCs to each other or to other systems.

Another example of a modular system is that provided by the Allen-Bradley SLC-500 programmable logic controller system. This is a small, chassis-based, modular family of programmable controllers having multiple processor choices, numerous power supply options and extensive input/output capacity. The SLC 500 allows the creation of a system specifically designed for an application. PLC blocks are mounted in a rack, with interconnections between the blocks being via a backplane bus. The PLC power supply is the end box in a rack with the next box containing the microprocessor. The backplane bus has copper conductors and provides the means by which the blocks slotted into the rack receive electrical power and for exchanging data between the modules and the processor. The modules slide into the rack and engage connectors on the backplane. SLC 500 series PLC racks are available to take 4, 7, 10 or 13 modules. Modules are available providing 8, 16 or current sinking d.c. inputs, 8, 16 or 32 current sourcing d.c. outputs, 8, 16 or 32 current sourcing d.c. outputs, 4, 8 or 16 relay a.c./d.c. outputs, communication module to enable additional communications with other computers or PLCs. Software is available to allow for programming from a Windows environment.

14.3 Input/output processing

A PLC is continuously running through its program and updating it as a result of the input signals. Each such loop is termed a **cycle**. There are two methods that can be used for input/output processing: continuous updating and mass input/output copying.

14.3.1 Continuous updating

Continuous updating involves the CPU scanning the input channels as they occur in the program instructions. Each input point is examined individually and its effect on the program determined. There will be a built-in delay, typically about 3 ms, when each input is examined in order to ensure that only valid input signals are read by the microprocessor. This delay enables the microprocessor to avoid counting an input signal twice, or, more frequently, if there is contact bounce at a switch. A number of inputs may have to be scanned, each with a 3 ms delay, before the program has the instruction for a logic operation to be executed and an output to occur. The outputs are latched so that they retain their status until the next updating.

14.3.2 Mass input/output copying

Because, with continuous updating, there has to be a 3 ms delay on each input, the time taken to examine several hundred input/output points can become comparatively long. To allow a more rapid execution of a program, a specific area of RAM is used as a buffer store between the control logic and the input/output unit. Each input/output has an address in this memory. At the start of each program cycle the CPU scans all the inputs and copies their status into the input/output addresses in RAM. As the program is executed the stored, input data is read, as required, from RAM and the logic operations carried out. The resulting output signals are stored in the reserved input/output section of RAM. At the end of each program cycle all the outputs are

transferred from RAM to the output channels. The outputs are latched so that they retain their status until the next updating. The sequence is:

1 scan all the inputs and copy into RAM;
2 fetch and decode and execute all program instructions in sequence, copying output instructions to RAM;
3 update all outputs;
4 repeat the sequence.

A PLC takes time to complete a cycle of scanning inputs and updating outputs according to the program instructions and so the inputs are not watched all the time but only examined periodically. A typical PLC cycle time is of the order of 10 to 50 ms and so the inputs and outputs are updated every 10 to 50 ms. This means that if a very brief input appears at the wrong moment in the cycle, it could be missed. Thus for a PLC with a cycle time of 40 ms, the maximum frequency of digital impulses that can be detected will be if one pulse occurs every 40 ms. The Mitsubishi compact PLC, MELSEC FX3U, has a quoted program cycle time of 0.065 μs per logical instruction and so the more complex the program, the longer the cycle time.

14.3.3 Input/output addresses

The PLC has to be able to identify each particular input and output and it does this by assigning addresses to each, rather like houses in a town have addresses to enable post to be delivered to the right families. With a small PLC the addresses are likely to be just a number preceded by a letter to indicate whether it is an input or output. For example, Mitsubishi and Toshiba have inputs identified as X400, X401, X402, etc., and outputs as Y430, Y431, etc. With larger PLCs having several racks of input and output channels and a number of modules in each rack, the racks and modules are numbered and so an input or output is identified by its rack number followed by the number of the module in that rack and then a number to show its terminal number in the module. For example, the Allen-Bradley PCL-5 has I:012/03 to indicate an input in rack 01 at module 2 and terminal 03.

14.4 Ladder programming

The form of programming commonly used with PLCs is **ladder programming**. This involves each program task being specified as though a rung of a ladder. Thus such a rung could specify that the state of switches A and B, the inputs, be examined and if A and B are both closed then a solenoid, the output, is energised. Figure 14.4 illustrates this idea by comparing it with an electric circuit.

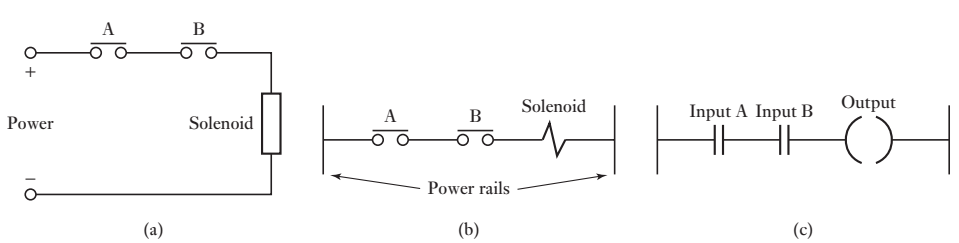


Figure 14.4 (a), (b) Alternative ways of drawing an electric circuit, (c) comparable rung in a ladder program.

The sequence followed by a PLC when carrying out a program can be summarised as follows.

1 Scan the inputs associated with one rung of the ladder program.
2 Solve the logic operation involving those inputs.
3 Set/reset the outputs for that rung.
4 Move on to the next rung and repeat operations 1, 2, 3.
5 Move on to the next rung and repeat operations 1, 2, 3.
6 Move on to the next rung and repeat operations 1, 2, 3.
7 And so on until the end of the program with each rung of the ladder program scanned in turn. The PLC then goes back to the beginning of the program and starts again.

PLC programming based on the use of **ladder diagrams** involves writing a program in a similar manner to drawing a switching circuit. The ladder diagram consists of two vertical lines representing the power rails. Circuits are connected as horizontal lines, i.e. the rungs of the ladder, between these two verticals. Figure 14.5 shows the basic standard symbols that are used and an example of rungs in a ladder diagram. In drawing the circuit line for a rung, inputs must always precede outputs and there must be at least one output on each line. Each rung must start with an input or a series of inputs and end with an output.

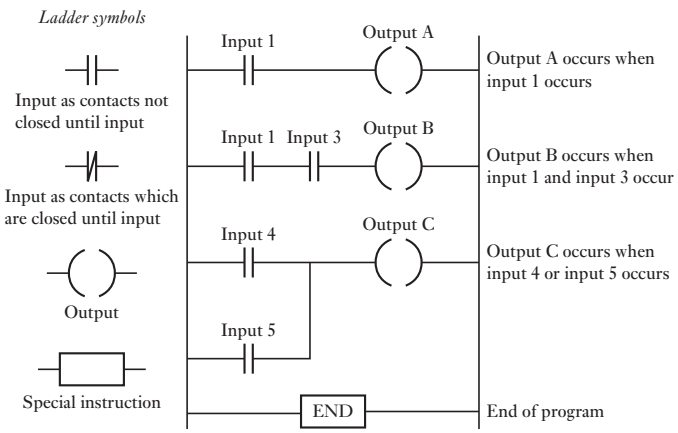


Figure 14.5 Ladder program.

To illustrate the drawing of a ladder diagram, consider a situation where the output from the PLC is to energise a solenoid when a normally open start switch connected to the input is activated by being closed (Figure 14.6(a)). The program required is shown in Figure 14.6(b)). Starting with the input, we have the normally open symbol | |. This might have an input address X400. The line terminates with the output, the solenoid, with the symbol ( ). This might have the output address Y430. To indicate the end of the program, the end rung is marked. When the switch is closed the solenoid is activated. This might, for example, be a solenoid valve which opens to allow water to enter a vessel.

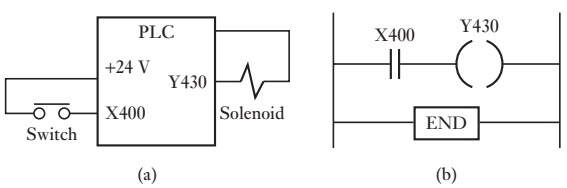


Figure 14.6 Switch controlling a solenoid.

Another example might be an on/off temperature control (Figure 14.7(a)) in which the input goes from low to high when the temperature sensor reaches the set temperature. The output is then to go from on to off. The temperature sensor shown in the figure is a thermistor connected in a bridge arrangement with output to an operational amplifier connected as a comparator (see Section 3.2.7). The program (Figure 14.7(b)) shows the input as a normally closed pair of contacts, so giving the on signal and hence an output. When the contacts are opened to give the off signal then the output is switched off.

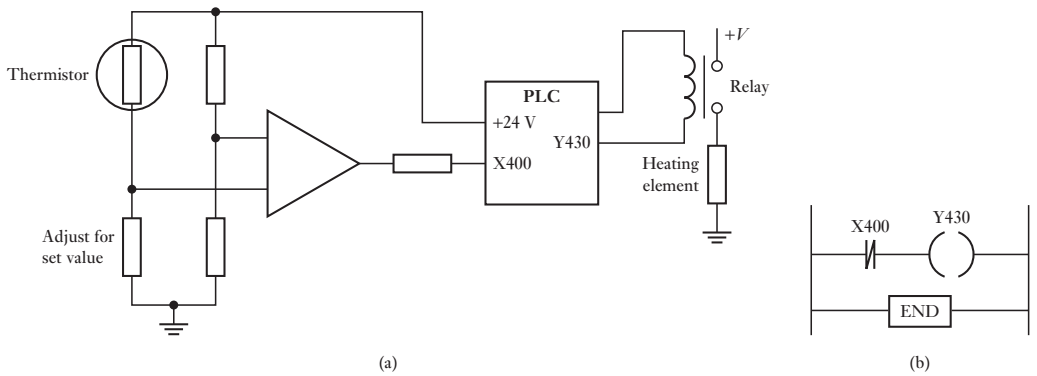


Figure 14.7 Temperature control system.

14.4.1 Logic functions

The logic functions can be obtained by combinations of switches (see Section 5.2) and the following shows how we can write ladder programs for such combinations (Figure 14.8).

1 AND
Figure 14.8(a) shows a situation where a coil is not energised unless two, normally open, switches are both closed. Switch A and switch B have both to be closed, which thus gives an AND logic situation. The equivalent ladder diagram starts with | |, labelled Input 1, to represent switch A and in series with it | |, labelled Input 2, to represent switch B. The line then terminates with ( ) to represent the output.

2 OR
Figure 14.8(b) shows a situation where a coil is not energised until either, normally open, switch A or B is closed. The situation is an OR logic gate. The equivalent ladder diagram starts with | |, labelled Input 1, to represent switch A and in parallel with it | |, labelled Input 2, to represent switch B. The line then terminates with ( ) to represent the output.

3 NOR
Figure 14.8(c) shows how we can represent the ladder program line for a NOR gate. Since there has to be an output when neither A nor B have an input, and when there is an input to A or B the output ceases, the ladder program shows Input 1 in parallel with Input 2, with both being represented by normally closed contacts.

4 NAND
Figure 14.8(d) shows a NAND gate. There is no output when both A and B have an input. Thus for the ladder program line to obtain an output we require no inputs to Input 1 and to Input 2.

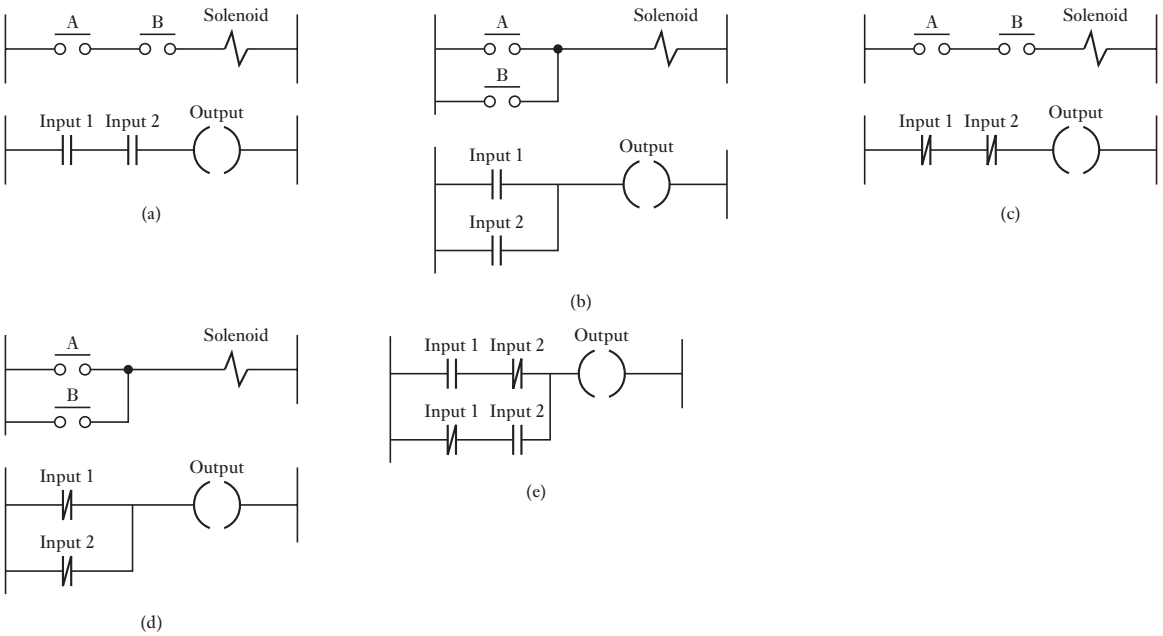


Figure 14.8 (a) AND, (b) OR, (c) NOR, (d) NAND, (e) XOR.

5 EXCLUSIVE-OR (XOR)
Figure 14.8(e) shows how we can draw the ladder program line for an XOR gate, there being no output when there is no input to Input 1 and Input 2 and when there is an input to both Input 1 and Input 2. Note that we have represented each input by two sets of contacts, one normally open and one normally closed.

Consider a situation where a normally open switch A must be activated and either of two other, normally open, switches B and C must be activated for a coil to be energised. We can represent this arrangement of switches as switch A in series with two parallel switches B and C (Figure 14.9(a)). For the coil to be energised we require A to be closed and either B or C to be closed. Switch A when considered with the parallel switches gives an AND logic situation. The two parallel switches give an OR logic situation. We thus have a combination of two gates. The truth table is:

Inputs			
A	**B**	**C**	**Output**
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

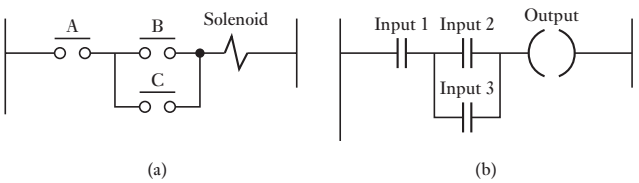


Figure 14.9 Switches controlling a solenoid.

For the ladder diagram, we start with | |, labelled Input 1, to represent switch A. This is in series with two | | in parallel, labelled Input 1 and Input 2, for switches B and C. The line then terminates with ( ) to represent the output, the coil. Figure 14.9(b) shows the line.

As a simple example of a program using logic gates, consider the requirement for there to be an output to the solenoid controlling the valve that will open a shop door when the shopkeeper has closed a switch to open the shop and a customer approaches the door and is detected by a sensor which then gives a high signal. The truth table for this system is thus:

Shop open switch	Customer approaching sensor	Solenoid output
Off	Off	Off
Off	On	Off
On	Off	Off
On	On	On

This truth table is that of an AND gate and thus the program for a PLC controlling the door is as shown in Figure 14.10.

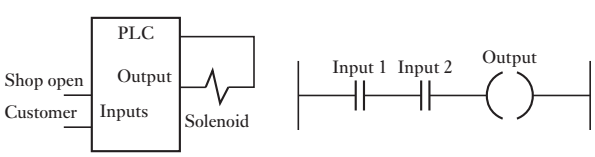


Figure 14.10 Shop door system.

14.5 Instruction lists

Each horizontal rung on the ladder in a ladder program represents a line in the program and the entire ladder gives the complete program in 'ladder language'. The programmer can enter the program into the PLC using a keyboard with the graphic symbols for the ladder elements, or using a computer screen and a mouse to select symbols, and the program panel or computer then translates these symbols into machine language that can be stored in the PLC memory. There is an alternative way of entering a program and that is to translate the ladder program into an **instruction list** and then enter this into the programming panel or computer.

Instruction lists consist of a series of instructions with each instruction being on a separate line. An instruction consists of an operator followed by one or more operands, i.e. the subjects of the operator. In terms of ladder programs, each operator in a program may be regarded as a ladder element. Thus we might have for the equivalent of an input to a ladder program:

```
LD A (*Load input A*)
```

Table 14.1 Instruction code mnemonics.

IEC 1131-3	Mitsubishi	OMRON	Siemens	Operation	Ladder diagram
LD	LD	LD	A	Load operand into result register	Start a rung with open contacts
LDN	LDI	LD NOT	AN	Load negative operand into result register	Start a rung with closed contacts
AND	AND	AND	A	Boolean AND	A series element with open contacts
ANDN	ANI	AND NOT	AN	Boolean AND with negative operand	A series element with closed contacts
OR	OR	OR	O	Boolean OR	A parallel element with open contacts
ORN	ORI	OR NOT	ON	Boolean OR with negative operand	A parallel element with closed contacts
ST	OUT	OUT	=	Store result register into operand	An output from a rung

The operator is LD for loading, the operand A as the subject being loaded and the words preceded by and concluded by * in brackets are comments explaining what the operation is and are not part of the program operation instructions to the PLC, but to aid a reader in comprehending what the program is about.

The mnemonic codes used by different PLC manufacturers differ but an international standard (IEC 1131-3) has been proposed and is widely used. Table 14.1 shows common core mnemonics. In examples discussed in the rest of this chapter, where general descriptions are not used, the Mitsubishi mnemonics will be used. However, those used by other manufacturers do not differ widely from these and the principles involved in their use are the same.

14.5.1 Instruction lists and logic functions

The following shows how individual rungs on a ladder are entered using the Mitsubishi mnemonics where logic functions are involved (Figure 14.11).

14.5.2 Instruction lists and branching

The EXCLUSIVE-OR (XOR) gate shown in Figure 14.12 has two parallel arms with an AND situation in each arm. In such a situation Mitsubishi (Figure 14.12(a)) uses an ORB instruction to indicate 'OR together parallel branches'. The first instruction is for a normally open pair of contacts X400, the next instruction for a set of normally closed contacts X401, hence ANI X401. The third instruction describes a new line, its being recognised as a new line because it starts with LDI, all new lines starting with LD or LDI. Because the first line has not been ended by an output, the PLC recognises that a parallel line is involved for the second line and reads together the listed

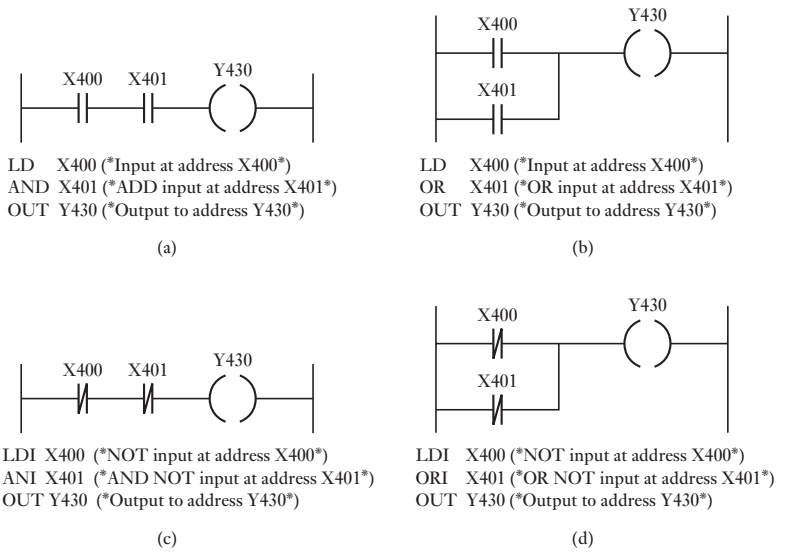


Figure 14.11 (a) AND, (b) OR, (c) NOR, (d) NAND.

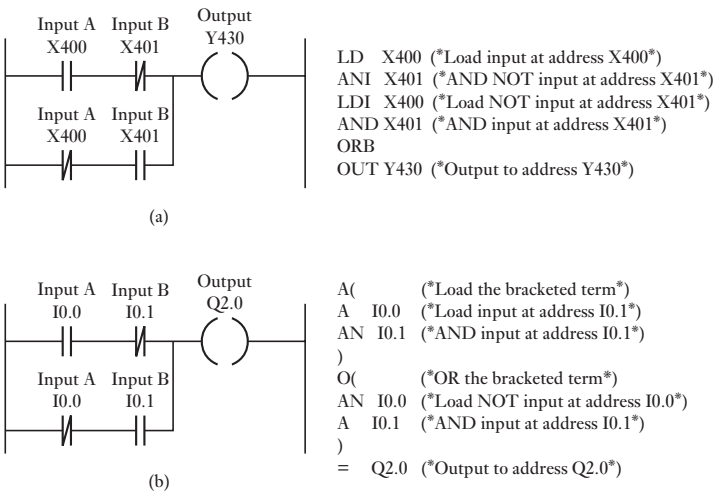


Figure 14.12 XOR.

elements until the ORB instruction is reached. ORB indicates to the PLC that it should OR the results of the first and second instructions with that of the new branch with the third and fourth instructions. The list concludes with the output OUT Y430. Figure 14.12(b) shows the Siemens version of XOR gate. Brackets are used to indicate that certain instructions are to be carried out as a block and are used in the same way as brackets in any mathematical equation. For example, $(1 + 2)/4$ means that the 1 and 2 must be added before dividing by 4. Thus with the Siemens instruction list the A(means that the load instruction A is only applied after the bracketed steps have been completed and) is reached. The IEC 1131-3 standard for such programming is to use brackets in the way used in the above Siemens example.

14.6 Latching and internal relays

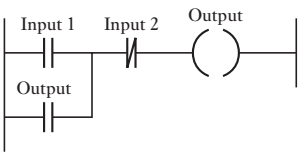
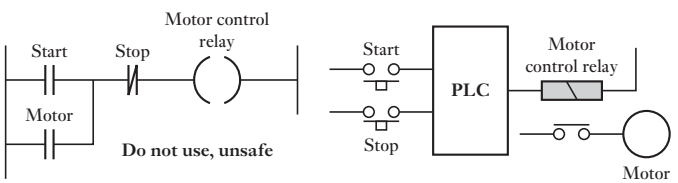


Figure 14.13 A latch circuit.

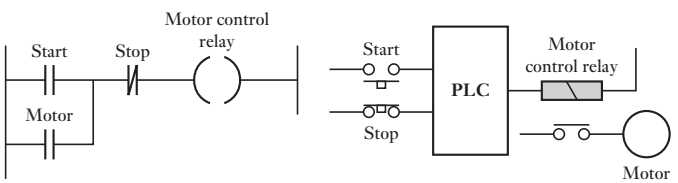
There are often situations where it is necessary to hold a coil energised, even when the input which energised it ceases. The term **latch circuit** is used for the circuit which carries out such an operation. It is a self-maintaining circuit in that, after being energised, it maintains that state until another input is received. It remembers its last state. An example of a latch circuit is shown in Figure 14.13. When Input 1 is energised and closes, there is an output. However, when there is an output, a set of contacts associated with the output is energised and closes. These contacts OR the Input 1 contacts. Thus, even if Input 1 contacts open, the circuit will still maintain the output energised. The only way to release the output is by operating the normally closed contact Input 2.

As an example of the use of a latching circuit, consider the requirement for a PLC to control a motor so that when the start signal button is momentarily pressed the motor starts and when the stop switch is used the motor switches off. Safety must be a priority in the design of a PLC system, so stop buttons must be hard-wired and not depend on the PLC software for implementation so that if there is a failure of the stop switch or PLC, the system is automatically safe. With a PLC system, a stop signal can be provided by a switch as shown in Figure 14.14(a). To start we momentarily close the press-button start switch and the motor internal control relay latches this closure and the output remains on. To stop we momentarily open the stop switch and this unlatches the start switch. However, if the stop switch cannot be operated then we cannot stop the system. Thus this system must *not* be used as it is unsafe, because if there is a fault and the switch cannot be operated, then no stop signal can be provided. What we require is a system that will still stop if a failure occurs in the stop switch. Figure 14.14(b) shows such a system. The program now has the stop switch as open contacts. However, because the hard-wired stop switch has normally closed contacts, then the program receives the signal to close the program contacts. Pressing the stop switch then opens the program contacts and stops the system.

Figure 14.14 Stop system: (a) unsafe, (b) safe.



(a)



(b)

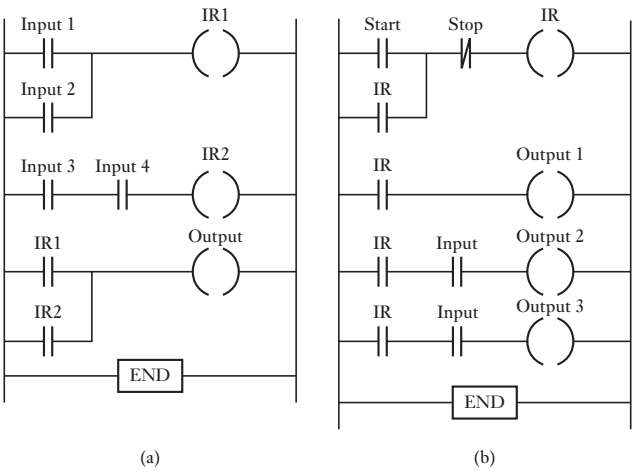


Figure 14.15 (a) An output controlled by two input arrangements, (b) starting of multiple outputs.

14.6.1 Internal relays

The term **internal relay, auxiliary relay** or **marker** is used for what can be considered as an internal relay in the PLC. These behave like relays with their associated contacts, but in reality are not actual relays but simulations by the software of the PLC. Some have battery back-up so that they can be used in circuits to ensure a safe shut-down of plant in the event of a power failure. Internal relays can be very useful aids in the implementation of switching sequences.

Internal relays are often used when there are programs with multiple input conditions. Consider the situation where the excitation of an output depends on two different input arrangements. Figure 14.15(a) shows how we can draw a ladder diagram using internal relays. The first rung shows one input arrangement being used to control the coil of internal relay IR1. The second rung shows the other input arrangement controlling the coil of internal relay IR2. The contacts of the two relays are then put in an OR situation to control the output.

Another use of internal relays is for the starting of multiple outputs. Figure 14.15(b) shows such a ladder program. When the start contacts are closed, the internal relay is activated and latches the input. It also starts Output 1 and makes it possible for Outputs 2 and 3 to be activated.

Another example of the use of internal relays is resetting a latch. Figure 14.16 shows the ladder diagram. When the contacts of Input 1 are momentarily pressed, the output is energised. The contacts of the output are then closed and so latch the output, i.e. keep it on even when the contacts of the input are no longer closed. The output can be unlatched by the internal relay contacts opening. This will occur if Input 2 is closed and energises the coil of the internal relay.

An example of the use of a battery-backed internal relay is shown in Figure 14.17. When the contacts of Input 1 close, the coil of the battery-backed internal relay is energised. This closes the internal relay contacts and so even if contacts of the input open as a result of power failure, the internal relay contacts remain closed. This means that the output controlled by the internal relay remains energised, even when there is a power failure.

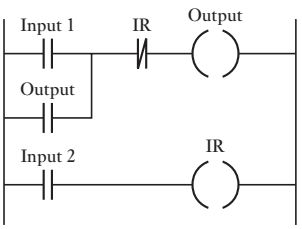


Figure 14.16 Resetting a latch.

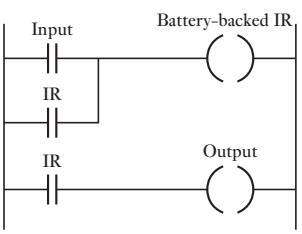


Figure 14.17 Use of a battery-backed internal relay.

14.7 Sequencing

There are often control situations where sequences of outputs are required, with the switch from one output to another being controlled by sensors. Consider the requirement for a ladder program for a pneumatic system (Figure 14.18) with double-solenoid valves controlling two double-acting cylinders A and B if limit switches a−, a+, b−, b+ are used to detect the limits of the piston rod movements in the cylinders and the cylinder activation sequence A+, B+, A−, B− is required. A possible program is shown in the figure. A start switch input has been included in the first rung. Thus cylinder extension for A, i.e. the solenoid A+ energised, only occurs when the start switch is closed and the b− switch is closed, this switch indicating that the B cylinder is retracted. When cylinder A is extended, the switch a+, which indicates the extension of A, is activated. This then leads to an output to solenoid B+ which results in B extending. This closes the switch indicating the extension of B, i.e. the b+ switch, and leads to the output to solenoid A− and the retraction of cylinder A. This retraction closes limit switch a− and so gives the output to solenoid B− which results in B retracting. This concludes the program cycle and leads to the first rung again, which awaits the closure of the start switch before being repeated.

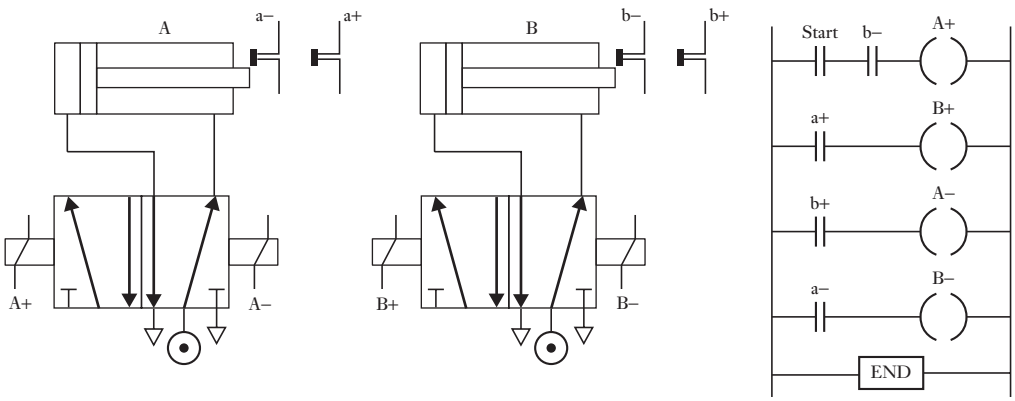


Figure 14.18 Cylinder sequencing.

As a further illustration, consider the problem of devising a ladder program to control a pneumatic system with double-solenoid-controlled valves and two cylinders A and B if limit switches a−, a+, b− and b+ are used to detect the limits of movement of the piston rod movements in the cylinders and the sequence required is for the piston rod in A to extend, followed by the piston rod in B extending, then the piston in B retracting and finally the cycle is completed by the piston in A retracting. An internal relay can be used to switch between groups of outputs to give the form of control for pneumatic cylinders, which is termed **cascade control** (see Section 7.5). Figure 14.19 shows a possible program. When the start switch is closed, the internal relay is activated. This energises solenoid A+ with the result that the piston in cylinder A extends. When extended it closes limit switch a+ and the piston in cylinder B extends. When this is extended it closes the limit switch b+. This activates the relay. As a result the B− solenoid is energised and the piston in B retracts. When this closes limit switch b−, solenoid A− is energised and the piston in cylinder A retracts.

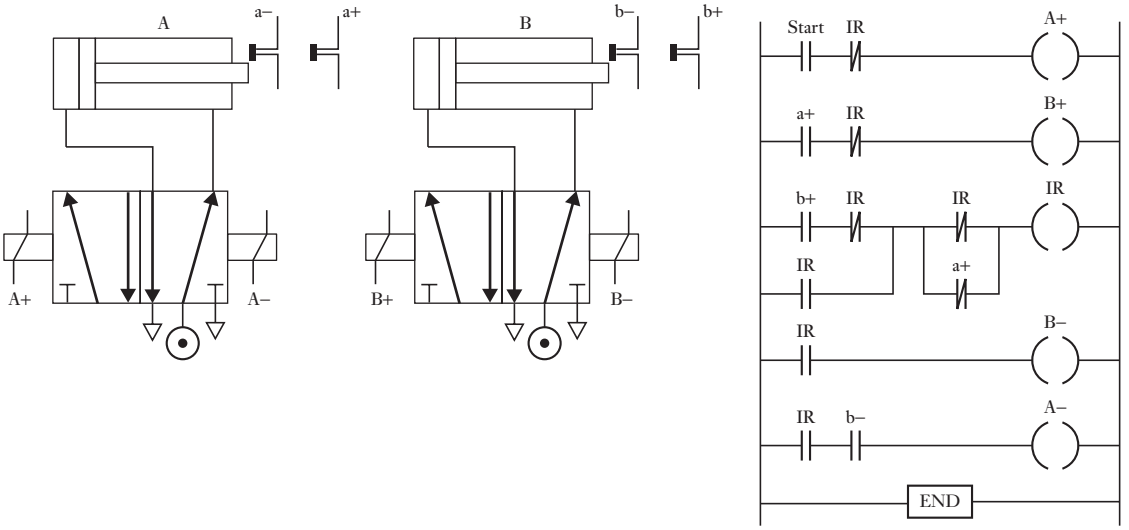


Figure 14.19 Cylinder sequencing.

14.8 Timers and counters

The previous sections in this chapter have been concerned with tasks requiring the series and parallel connections of input contacts. However, there are tasks which can involve time delays and event counting. These requirements can be met by the timers and counters which are supplied as a feature of PLCs. They can be controlled by logic instructions and represented on ladder diagrams.

14.8.1 Timers

A common approach used by PLC manufacturers is to consider timers to behave like relays with coils which when energised result in the closure or opening of contacts after some preset time. The timer is thus treated as an output for a rung with control being exercised over pairs of contacts elsewhere (Figure 14.20(a)). Others consider a timer as a delay block in a rung which delays signals in that rung reaching the output (Figure 14.20(b)).

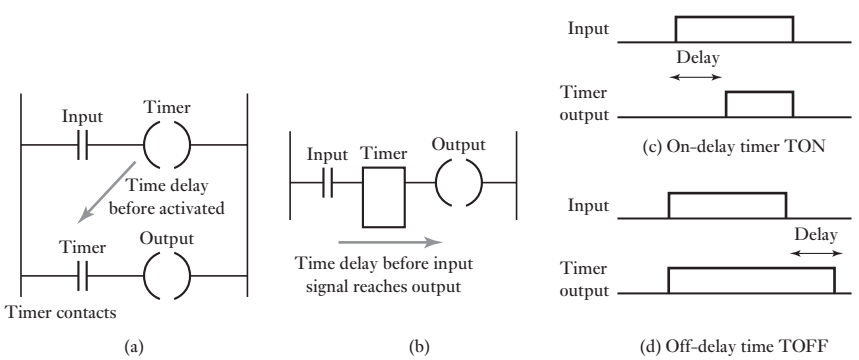


Figure 14.20 (a), (b) Delay-on timer, (c) timing with on-delay, (d) timing with off-delay.

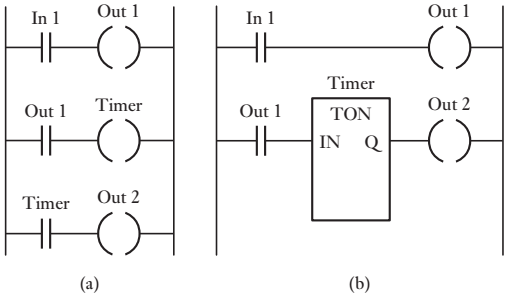


Figure 14.21 Timed sequence.

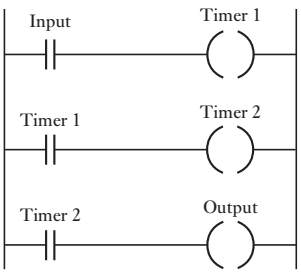


Figure 14.22 Cascaded timers.

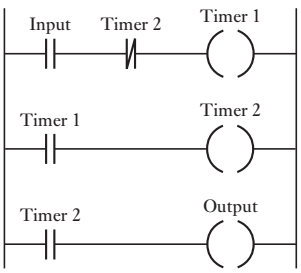


Figure 14.23 On/off cyclic timer.

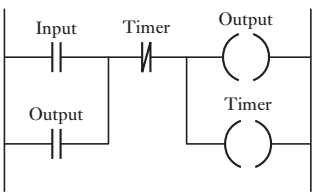


Figure 14.24 Delay-off timer.

PLCs are generally provided with only a delay-on timer (TON), small PLCs possibly having only this type. Such a timer waits for a fixed delay period before turning on (Figure 14.20(c)), e.g. a period which can be set between 0.1 and 999 s in steps of 0.1 s. Other time-delay ranges and steps are possible.

As an illustration of the use of a timer for sequencing, consider the ladder diagram shown in Figure 14.21(a) or (b). When the input In 1 is on, the output Out 1 is switched on. The contacts associated with this output then start the timer. The contacts of the timer will close after the preset time delay. When this happens, output Out 2 is switched on.

Timers can be linked together, or **cascaded**, to give larger delay times than is possible with just one timer. Figure 14.22 shows such an arrangement. When the input contacts close, Timer 1 is started. After its time delay, its contacts close and Timer 2 is started. After its time delay, its contacts close and there is an output.

Figure 14.23 shows a program that can be used to cause an output to go on for 0.5 s, then off for 0.5 s, then on for 0.5 s, then off for 0.5 s, and so on. When the input contacts close, Timer 1 is started and comes on after 0.5 s, this being the time for which it was preset. After this time the Timer 1 contacts close and start Timer 2. It comes on after 0.5 s, its preset time, and opens its contacts. This results in Timer 1 being switched off. This results in its contact opening and switching off Timer 2. This then closes its contact and so starts the entire cycle again. The result is that timer contacts for Timer 1 are switched on for 0.5 s, then off for 0.5 s, on for 0.5 s, and so on. Thus the output is switched on for 0.5 s, then off for 0.5 s, on for 0.5 s, and so on.

Figure 14.24 shows how a delay-off timer, i.e. a timer which switches off an output after a time delay from being energised, can be devised. When the input contacts are momentarily closed the output is energised (Figure 14.20(d)) and the timer started. The output contacts latch the input and keep the output on. After the preset time of the timer, the timer comes on and breaks the latch circuit, so switching the output off.

14.8.2 Counters

Counters are used when there is a need to count a specified number of contact operations, e.g. where items pass along a conveyor into boxes, and when the specified number of items has passed into a box, the next item is diverted into another box. Counter circuits are supplied as an internal

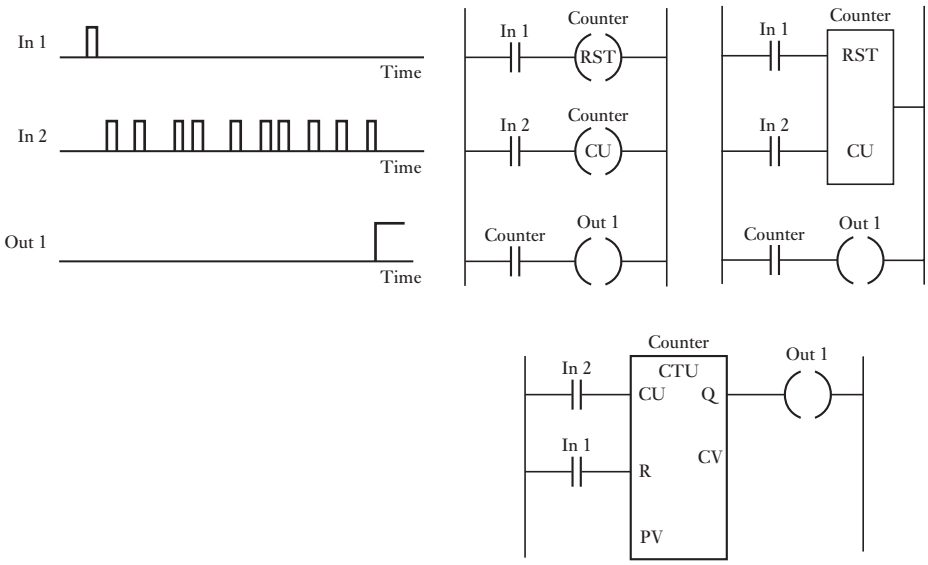


Figure 14.25 The inputs and output for a counter and various ways of representing the same program.

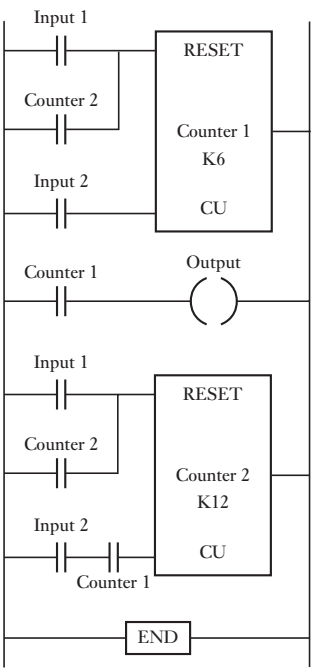


Figure 14.26 Counter.

feature of PLCs. In most cases the counter operates as a **down-counter**. This means that the counter counts down from the present value to zero, i.e. events are subtracted from the set value. When zero is reached the counter's contact changes state. An **up-counter** would count up to the preset value, i.e. events are added until the number reaches the set value. When the set value is reached the counter's contact changes state.

Different PLC manufacturers deal with counters in different ways. Some consider the counter to consist of two basic elements: one output coil to count input pulses and one to reset the counter, the associated contacts of the counter being used in other rungs, e.g. Mitsubishi and Allen-Bradley. Others treat the counter as an intermediate block in a rung from which signals emanate when the count is attained, e.g. Siemens. As an illustration, Figure 14.25 shows a basic counting circuit. When there is a pulse input to In 1, the counter is reset. When there is an input to In 2, the counter starts counting. If the counter is set for, say, 10 pulses, then when 10 pulse inputs have been received at In 2, the counter's contacts will close and there will be an output from Out 1. If at any time during the counting there is an input to In 1, the counter will be reset and start all over again and count for 10 pulses.

As an illustration of the use of a counter, consider the problem of the control for a machine which is required to direct 6 items along one path for packaging in a box, and then 12 items along another path for packaging in another box. Figure 14.26 shows the program that could be used. It involves two counters, one preset to count 6 and the other to count 12. Input 1 momentarily closes its contacts to start the counting cycle, resetting both counters. Input 2 contacts could be activated by a microswitch, which is activated every time an item passes up to the junction in the paths. Counter 1

counts 6 items and then closes its contact. This activates the output, which might be a solenoid used to activate a flap which closes one path and opens another. Counter 1 also has contacts which close and enables Counter 2 to start counting. When Counter 2 has counted 12 items it resets both the counters and opens the Counter 1 contacts, which then results in the output becoming deactivated and items no longer directed towards the box to contain 12 items.

14.9 Shift registers

A number of internal relays can be grouped together to form a register which can provide a storage area for a series sequence of individual bits. A 4-bit register would be formed by using four internal relays, an 8-bit using eight. The term **shift register** is used because the bits can be shifted along by 1 bit when there is a suitable input to the register. For example, with an 8-bit register we might initially have:

1	0	1	1	0	1	0	1

Then there is an input of a 0 shift pulse:

$0 \rightarrow$ | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | $\rightarrow 1$

with the result that all the bits shift along one place and the last bit overflows.

The grouping together of a number of auxiliary registers to form a shift register is done automatically by a PLC when the shift register function is selected at the control panel. With the Mitsubishi PLC, this is done by using the programming function SFT (shift) against the auxiliary relay number that is to be the first in the register array. This then causes the block of relays, starting from that initial number, to be reserved for the shift register. Thus, if we select M140 to be the first relay then the shift register will consist of M140, M141, M142, M143, M144, M145, M146 and M147.

Shift registers have three inputs: one to load data into the first element of the register (OUT), one as the shift command (SFT) and one for resetting (RST). With OUT, a logic level 0 or 1 is loaded into the first element of the shift register. With SFT, a pulse moves the contents of the register along 1 bit at a time, the final bit overflowing and being lost. With RST, a pulse of the closure of a contact resets the register contents to all zeros.

Figure 14.27 gives a ladder diagram involving a shift register when Mitsubishi notation is used; the principle is, however, the same with other manufacturers. M140 has been designated as the first relay of the register. When X400 is switched on, a logic 1 is loaded into the first element of the shift register, i.e. M140. We thus have the register with 10000000. The circuit shows that each element of the shift register has been connected as a contact in the circuit. Thus M140 contact closes and Y430 is switched on. When contact X401 is closed, then the bits in the register are shifted along the register by one place to give 11000000, a 1 being shifted into the register because X400 is still on. Contact M141 thus closes and Y430 is switched on. As each bit is shifted along, the outputs are energised in turn. Shift registers can thus be used to sequence events.

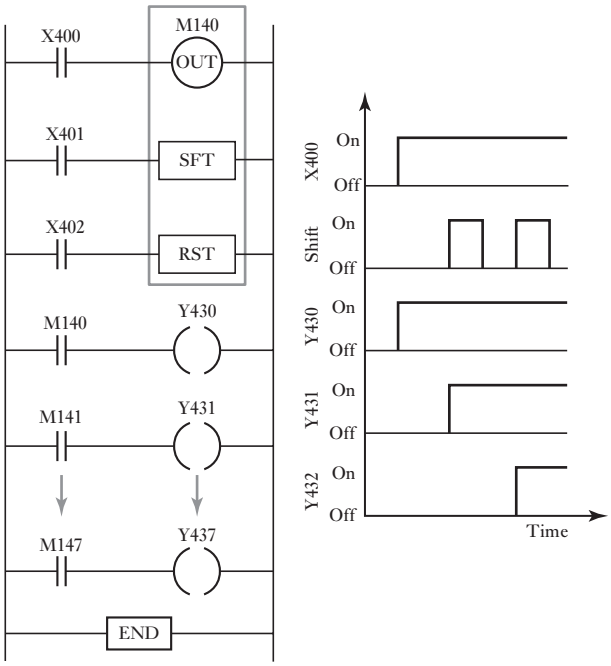


Figure 14.27 Shift register.

14.10 Master and jump controls

A whole block of outputs can be simultaneously turned off or on by using the same internal relay contacts in each output rung so that switching it on or off affects every one of the rungs. An alternative way of programming to achieve the same effect is to use a **master relay**. Figure 14.28 illustrates its use. We can think of it as controlling the power to a length of the vertical rails of the ladder. When there is an input to close Input 1 contacts, master relay MC1 is activated and then the block of program rungs controlled by that relay follows. The end of a master-relay-controlled section is indicated by the reset MCR. It is thus a branching program in that if there is Input 1 then branch to follow the MC1 controlled path; if not, follow the rest of the program and ignore the branch.

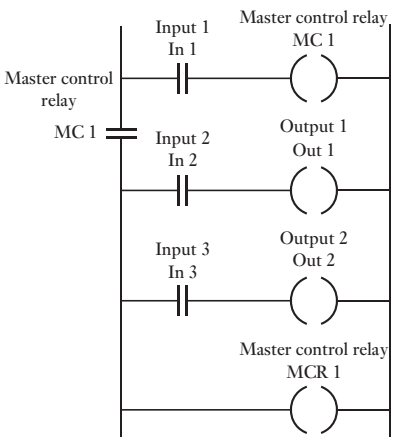


Figure 14.28 Master control relay.

With a Mitsubishi PLC, an internal relay can be designated as a master control relay by programming it accordingly. Thus to program an internal relay M100 as a master control relay, the program instruction is:

```
MC M100
```

To indicate the end of the section controlled by a master control relay, the program instruction is:

```
MCR M100
```

14.10.1 Jumps

A function which is often provided with PLCs is the **conditional jump** function. Such a function enables programs to be designed so that if a certain condition exists then a section of the program is jumped. Figure 14.29 illustrates this on a flow diagram and with a section of ladder program. Following a section of program, A, the program rung is encountered with Input 1 and the conditional jump relay CJP. If Input 1 occurs then the program jumps to the rung with the end of jump relay coil EJP and so continues with that section of the program labelled as C, otherwise it continues with the program rungs labelled as program B.

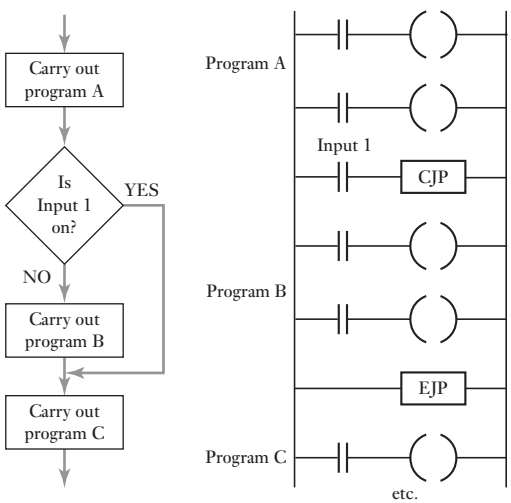


Figure 14.29 Jump.

14.11 Data handling

With the exception of the shift register, the previous parts of this chapter have been concerned with the handling of individual bits of information, e.g. a switch being closed or not. There are, however, some control tasks where it is useful to deal with related groups of bits, e.g. a block of eight inputs, and so operate on them as a data word. Such a situation can arise when a sensor supplies an analogue signal which is converted to, say, an 8-bit word before becoming an input to a PLC.

The operations that may be carried out with a PLC on data words normally include:

1 moving data;
2 comparison of magnitudes of data, i.e. greater than, equal to, or less than;
3 arithmetic operations such as addition and subtraction;
4 conversions between binary-coded decimal (BCD), binary and octal.

As discussed earlier, individual bits have been stored in memory locations specified by unique addresses. For example, for the Mitsubishi PLC, input memory addresses have been preceded by an A, outputs by a Y, timers by a T, auxiliary relays by an M, etc. Data instructions also require memory addresses and the locations in the PLC memory allocated for data are termed **data registers**. Each data register can store a binary word of, usually, 8 or 16 bits and is given an address such as D0, D1, D2, etc. An 8-bit word means that a quantity is specified to a precision of 1 in 256, a 16-bit word a precision of 1 in 65536.

Each instruction has to specify the form of the operation, the source of the data used in terms of its data register and the destination data register of the data.

14.11.1 Data movement

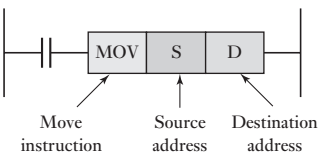


Figure 14.30 Move data.

For data movement the instruction will contain the move data instruction, the source address of the data and the destination address of the data. Thus the ladder rung could be of the form shown in Figure 14.30.

Such data transfers might be to move a constant into a data register, a time or count value to a data register, data from a data register to a timer or counter, data from a data register to an output, input data to a data register, etc.

14.11.2 Data comparison

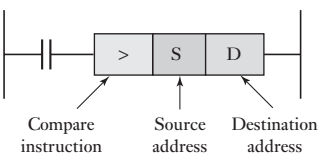


Figure 14.31 Compare data.

PLCs can generally make the data comparisons of *less than* (usually denoted by $<$ or LES), *equal to* ($=$ or EQU), *less than or equal to* ($\leq$ or $<=$ or LEQ), *greater than* ($>$ or GRT), *greater than or equal to* ($\geq$, $>=$ or GEQ) and *not equal to* ($\neq$ or $<>$ or NEQ). To compare data, the program instruction will contain the comparison instruction, the source address of the data and the destination address. Thus to compare the data in data register D1 to see if it is greater than data in data register D2, the ladder program rung would be of the form shown in Figure 14.31.

Such a comparison might be used when the signals from two sensors are to be compared by the PLC before action is taken. For example, an alarm might be required to be sounded if a sensor indicates a temperature above 80°C and remain sounding until the temperature falls below 70°C. Figure 14.32 shows the ladder program that could be used. The input temperature data is inputted to the source address and the destination address contains the set value. When the temperature rises to 80°C, or higher, the data value in the source address becomes $\geq$ the destination address value and there is an output to the alarm which latches the input. When the temperature falls to 70°C, or lower, the data value in the source address becomes $\leq$ the destination address value and there is an output to the relay which then opens its contacts and so switches the alarm off.

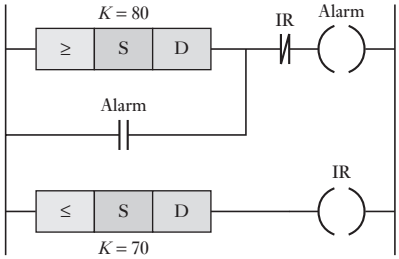


Figure 14.32 Temperature alarm.

14.11.3 Arithmetic operations

Some PLCs can carry out just the arithmetic operations of addition and subtraction, others have even more arithmetic functions. The instruction to add or subtract generally states the instruction, the register containing the address of the value to be added or subtracted, the address of the value to which the addition or from which the subtraction is to be made and the register where the result is to be stored. Figure 14.33 shows the form used for the ladder symbol for addition with OMRON.

Addition or subtraction might be used to alter the value of some sensor input value, perhaps a correction or offset term, or alter the preset values of timers or counters.

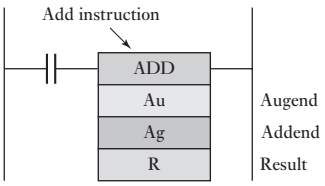


Figure 14.33 Add data.

14.11.4 Code conversions

All the internal operations in the CPU of a PLC are carried out using binary numbers. Thus, when the input is a signal which is decimal, conversion to BCD is used. Likewise, where a decimal output is required, conversion to decimal is required. Such conversions are provided with most PLCs. For example, with Mitsubishi, the ladder rung to convert BCD to binary is of the form shown in Figure 14.34. The data at the source address is in BCD and converted to binary and placed at the destination address.

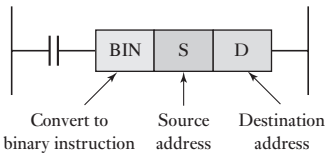


Figure 14.34 BCD to binary.

14.12 Analogue input/output

Many sensors generate analogue signals and many actuators require analogue signals. Thus, some PLCs may have an analogue-to-digital converter (ADC) module fitted to input channels and a digital-to-analogue converter (DAC) module fitted to output channels. An example of where such an item might be used is for the control of the speed of a motor so that its speed moves up to its steady value at a steady rate (Figure 14.35). The input is an on/off switch to start the operation. This opens the contacts for the data register and so it stores zero. Thus the output from the controller is zero and the analogue signal from the DAC is zero and hence motor speed is zero. The closing of the start contacts gives outputs to the DAC and the data register. Each time the program cycles through these rungs on the program, the data register is incremented by 1 and so the analogue signal is increased and hence the motor speed. Full speed is realised when the output from the data register is the word 11111111. The timer function of a PLC can be used to incorporate a delay between each of the output bit signals.

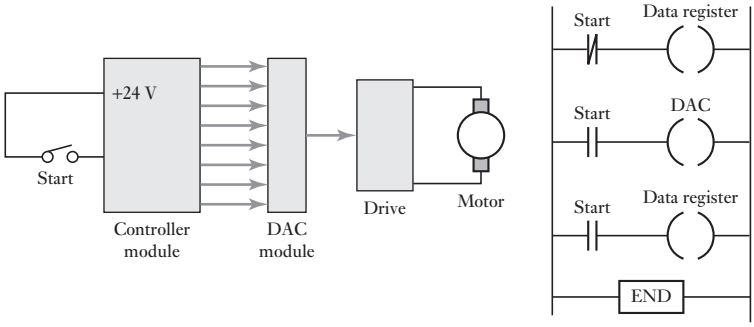


Figure 14.35 Ramping the speed of a motor.

A PLC equipped with analogue input channels can be used to carry out a continuous control function, i.e. PID control (see Section 22.7). Thus, for example, to carry out proportional control on an analogue input the following set of operations can be used:

1 Convert the sensor output to a digital signal.
2 Compare the converted actual sensor output with the required sensor value, i.e. the set point, and obtain the difference. This difference is the error.
3 Multiply the error by the proportional constant K_P.
4 Move this result to the DAC output and use the result as the correction signal to the actuator.

An example of where such a control action might be used is with a temperature controller. Figure 14.36 shows a possibility. The input could be from a thermocouple, which after amplification is fed through an ADC into the PLC. The PLC is programmed to give an output proportional to the

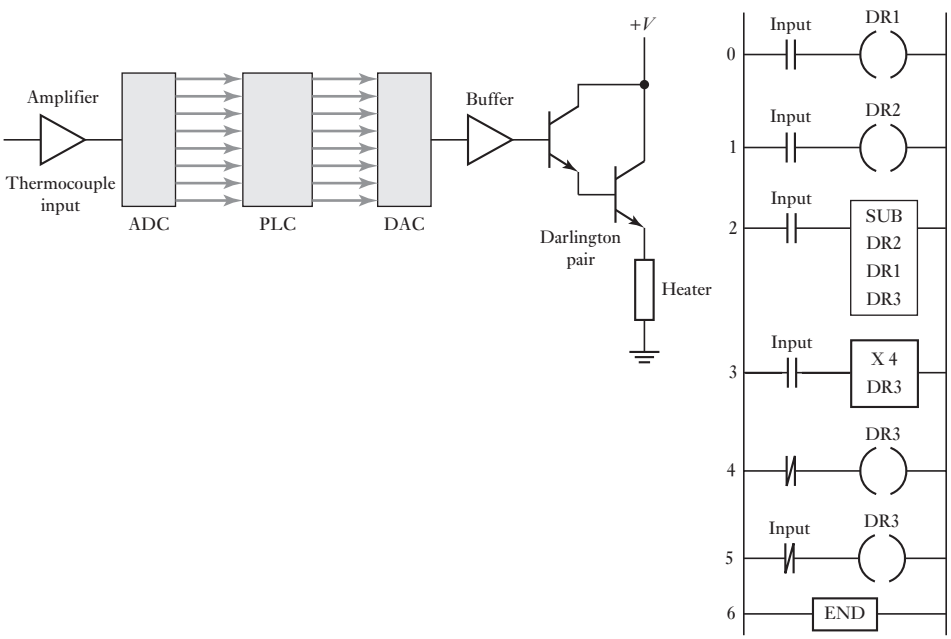


Figure 14.36 Proportional control of temperature.

error between the input from the sensor and the required temperature. The output word is then fed through a DAC to the actuator, a heater, in order to reduce the error.

With the ladder program shown, rung 0 reads the ADC and stores the temperature value in data register DR1. With rung 1, the data register DR2 is used to store the set point temperature. Rung 2 uses the subtract function to subtract the values held in data registers DR1 and DR2 and store the result in data register DR3, i.e. this data register holds the error value. With rung 3 a multiply function is used, in this case to multiply the value in data register DR3 by the proportional gain of 4. Rung 4 uses an internal relay which can be programmed to switch off DR3 if it takes a negative value. With rung 5 the data register DR3 is reset to zero when the input is switched off. Some PLCs have add-on modules which more easily enable PLC control to be used, without the need to write lists of instructions in the way outlined above.

Summary

A **programmable logic controller** (PLC) is a digital electronic device that uses a programmable memory to store instructions and to implement functions such as logic, sequencing, timing, counting and arithmetic in order to control machines and processes and has been specifically designed to make programming easy.

A PLC is continuously running through its program and updating it as a result of the input signals. Each such loop is termed a **cycle**. The form of programming commonly used with PLCs is **ladder programming**. This involves each program task being specified as though a rung of a ladder. There is an alternative way of entering a program and that is to translate the ladder program into an **instruction list**. Instruction lists consist of a series of instructions with each instruction being on a separate line. An instruction consists of an operator followed by one or more operands, i.e. the subjects of the operator.

A **latch circuit** is a circuit that, after being energised, maintains that state until another input is received. The term **internal relay**, **auxiliary relay** or **marker** is used for what can be considered as an internal relay in the PLC, these behaving like relays with their associated contacts. **Timers** can be considered to behave like relays with coils which when energised result in the closure or opening of contacts after some preset time or as a delay block in a rung which delays signals in that rung reaching the output. **Counters** are used to count a specified number of contact operations, being considered to be an output coil to count input pulses with a coil to reset the counter and the associated contacts of the counter being used in other rungs or as an intermediate block in a rung from which signals emanate when the count is attained. A **shift register** is a number of internal relays which have been grouped together to form a register for a series sequence of individual bits. A **master relay** enables a whole block of outputs to be simultaneously turned off or on. The **conditional jump** function enables a section of the program to be jumped if a certain condition exists. Operations that may be carried out with **data words** include moving data, comparison of magnitudes of data, arithmetic operations and conversions between binary-coded decimal (BCD), binary and octal.

Problems

14.1 What are the logic functions used for switches (a) in series, (b) in parallel?

14.2 Draw the ladder rungs to represent:

(a) two switches are normally open and both have to be closed for a motor to operate;

(b) either of two, normally open, switches have to be closed for a coil to be energised and operate an actuator;

(c) a motor is switched on by pressing a spring-return push-button start switch, and the motor remains on until another spring-return push-button stop switch is pressed.

14.3 Write the program instructions corresponding to the latch program shown in Figure 14.37.

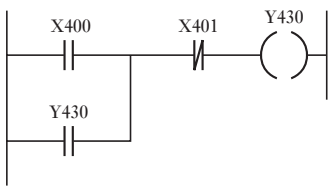


Figure 14.37 Problem 14.3.

14.4 Write the program instructions for the program in Figure 14.38 and state how the output varies with time.

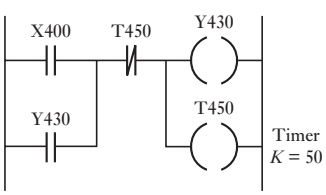


Figure 14.38 Problem 14.4.

14.5 Write the program instructions corresponding to the program in Figure 14.39 and state the results of inputs to the PLC.

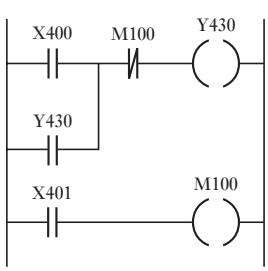


Figure 14.39 Problem 14.5.

14.6 Devise a timing circuit that will switch on an output for 1 s then off for 20 s, then on for 1 s, then off for 20 s, and so on.

14.7 Devise a timing circuit that will switch on an output for 10 s then switch it off.

14.8 Devise a circuit that can be used to start a motor and then after a delay of 100 s start a pump. When the motor is switched off there should be a delay of 10 s before the pump is switched off.

14.9 Devise a circuit that could be used with a domestic washing machine to switch on a pump to pump water for 100 s into the machine, then switch off and switch on a heater for 50 s to heat the water. The heater is then switched off and another pump is to empty the water from the machine for 100 s.

14.10 Devise a circuit that could be used with a conveyor belt which is used to move an item to a work station. The presence of the item at the work station is detected by means of breaking a contact activated by a beam of light to a photosensor. There the item stops for 100 s for an operation to be carried out before moving on and off the conveyor. The motor for the belt is started by a normally open start switch and stopped by a normally closed switch.

14.11 How would the timing pattern for the shift register in Figure 14.27 change if the data input X400 was of the form shown in Figure 14.40?

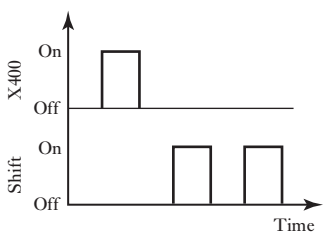


Figure 14.40 Problem 14.11.

14.12 Explain how a PLC can be used to handle an analogue input.

14.13 Devise a system, using a PLC, which can be used to control the movement of a piston in a cylinder so that when a switch is momentarily pressed, the piston moves in one direction and when a second switch is momentarily pressed, the piston moves in the other direction. Hint: you might consider using a 4/2 solenoid-controlled valve.

14.14 Devise a system, using a PLC, which can be used to control the movement of a piston in a cylinder using a 4/2 solenoid-operated pilot valve. The piston is to move in one direction when a proximity sensor at one end of the stroke closes contacts and in the other direction when a proximity sensor at the other end of the stroke indicates its arrival there.