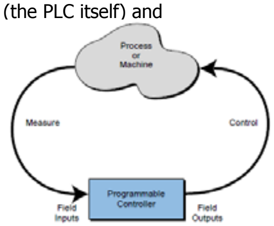
PLC: introduction

- **Programmable logic controllers**, also called *programmable controllers* or *PLCs*, are **solid-state** members of the computer family, using integrated circuits instead of electromechanical devices to implement control functions.
- They are capable of storing instructions, such as sequencing, timing, counting, arithmetic, data manipulation, and communication, to control industrial machines and processes.
- Figure 9.1 illustrates a conceptual diagram of a PLC application.
- PLCs can be thought of in simple terms as industrial computers with specially designed architecture in both their central units (the PLC itself) and their interfacing circuitry to field devices (input/output connections to the real world).
- PLCs are mature industrial controllers with their design roots based on the principles of simplicity and practical application.



Fugure 9.1 PLC conceptual application diagram.

PLC: introduction

- The Hydramatic Division of the General Motors Corporation specified the design criteria for the first programmable controller in 1968.
- A programmable controller, as illustrated in Figure 9.2, consists of two basic sections:
  - the central processing unit
  - the input/output interface system

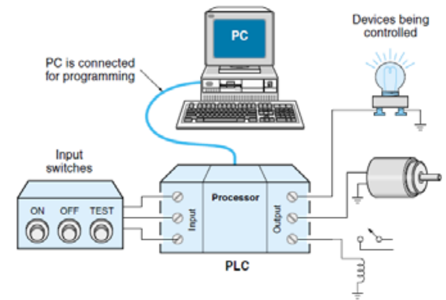


Figure 9.2 A PLC and its related components.

PLC: application

- Since its inception, the PLC has been successfully applied in virtually every segment of industry, including
  - steel mills,
  - paper plants,
  - food-processing plants,
  - chemical plants, and
  - power plants.
- PLCs perform a great variety of control tasks, from repetitive ON/OFF control of simple machines to sophisticated manufacturing and process control.
- Because the applications of programmable controllers are extensive, it is impossible to list them all.
- Let us consider a small sample of how PLCs are being used in an automotive industry:
  - **1. Internal Combustion Engine Monitoring**
  - A PLC acquires data recorded from sensors located at the internal combustion engine.
  - Measurements taken include water temperature, oil temperature, RPMs, torque, exhaust temperature, oil pressure, manifold pressure, and timing.

PLC: application

2. Carburetor Production Testing

- PLCs provide on-line analysis of automotive carburetors in a production assembly line.
- The systems significantly reduce the test time, while providing greater yield and better quality carburetors.
- Pressure, vacuum, and fuel and air flow are some of the variables tested.

3. Monitoring Automotive Production Machines

- The system monitors total parts, rejected parts, parts produced, machine cycle time, and machine efficiency.
- Statistical data is available to the operator anytime or after each shift.

4. Power Steering Valve Assembly and Testing

- The PLC system controls a machine to ensure proper balance of the valves and to maximize left and right turning ratios.

PLC: components

- Figure 9.3 graphically illustrates programmable controller product ranges.
- This chart is not definitive, but for practical purposes, it is valid.
- The PLC market can be segmented into five groups:
  1. micro PLCs
  2. small PLCs
  3. medium PLCs
  4. large PLCs
  5. very large PLCs

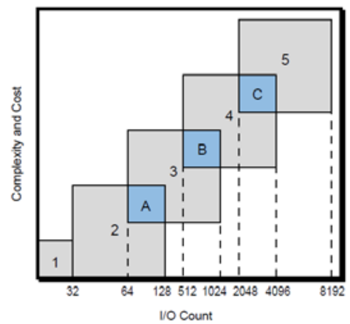
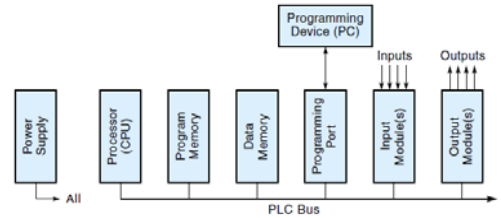


Figure 9.3 PLC product ranges..

PLC: components

Fugure 9.4
PLC conceptual application diagram.



Power Supply

- PLCs are usually powered directly from 120 or 240 Vac.
- The power supply converts the AC into DC voltages for the internal microprocessor components.
- It may also provide the user with a source of reduced voltage to drive switches, small relays, indicator lamps, and the like.

PLC: components

Processor

- The **processor** is a microprocessor-based CPU and is the part of the PLC that is capable of reading and executing the program instructions, one-by-one.
- During its operation, the CPU completes three processes:
  (1) it **reads**, or accepts, the input data from the field devices via the input interfaces,
  (2) it **executes**, or performs, the control program stored in the memory system, and
  (3) it **writes**, or updates, the output devices via the output interfaces.
- This process of sequentially reading the inputs, executing the program in memory, and updating the outputs is known as **scanning.**
- Figure 9.5 illustrates a graphic representation of a scan.

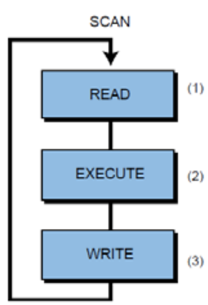


Figure 9.5. Illustration of a scan.

PLC: components

Memory

- The **memory** system is the area in the PLC's CPU where all of the sequences of instructions, or *programs*, are stored and executed by the processor to provide the desired control of field devices.
- The total memory system in a PLC is actually composed of two different memories (see Figure 5-1):
  - the executive memory (Program memory)
  - the application memory (Data memory)
- The **executive memory** is a collection of permanently stored programs that are considered part of the PLC itself.
  - These supervisory programs direct all system activities, such as execution of the control program and communication with peripheral devices.
  - The executive section is the part of the PLC's memory where the system's available instruction software is stored (i.e., relay instructions, block transfer functions, math instructions, etc.).
  - This area of memory is not accessible to the user.

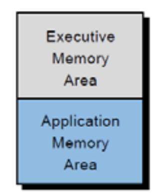


Figure 9.6 Simplified block diagram of the total PLC memory system.

PLC: Memory

- The **application memory** provides a storage area for the user-programmed instructions that form the application program.
- The application memory area is composed of several areas, each having a specific function and usage.

Memory types

- The following discussion describes six types of memory
- **Read-only memory (ROM)** is designed to permanently store a fixed program that is not alterable under ordinary circumstances.
  - It gets its name from the fact that its contents can be examined, or *read*, but not altered once information has been stored.
  - ROMs are generally immune to alteration due to electrical noise or loss of power.
  - Executive programs are often stored in ROM.
  - PLCs rarely use read-only memory for their application memory.
- **Random-access memory (RAM),** often referred to as *read/write memory (R/W)*, is designed so that information can be written into or read from the memory storage area.
  - does not retain its contents if power is lost; therefore, it is a volatile type of memory.
  - normally uses a battery backup to sustain its contents in the event of a power outage.

PLC: Memory

- **Programmable read-only memory (PROM)** is a special type of ROM because it can be programmed.
  - Very few of today's programmable controllers use PROM for application memory.
  - Although a PROM is programmable and, like any other ROM, has the advantage of non-volatility, it has the disadvantage of requiring special programming equipment.
  - Also, once programmed, it cannot be easily erased or altered; any program change requires a new set of PROM chips.
  - A PROM memory is suitable for storing a program that has been thoroughly checked while residing in RAM and will not require further changes or on-line data entry.
- **Erasable programmable read-only memory (EPROM)** is a specially designed PROM that can be reprogrammed after being entirely erased by an ultraviolet (UV) light source.
  - Complete erasure of the contents of the chip requires that the window of the chip be exposed to a UV light source for approximately twenty minutes.
  - EPROM can be considered a semi-permanent storage device, because it permanently stores a program until it is ready to be altered.
  - many controllers offer EPROM application memory as an optional backup to battery-supported RAM.
  - EPROM, with its permanent storage capability, combined with RAM, which is easily altered, makes a suitable memory system for many applications.

PLC: Memory

- **Electrically alterable read-only memory (EAROM)** is similar to EPROM, but instead of requiring an ultraviolet light source to erase it, an erasing voltage on the proper pin of an EAROM chip can wipe the chip clean.
  - Very few controllers use EAROM as application memory, but like EPROM, it provides a nonvolatile means of program storage and can be used as a backup to RAM-type memories.
- **Electrically erasable programmable read-only memory (EEPROM)** is an integrated circuit memory storage device that was developed in the mid- 1970s.
  - Like ROMs and EPROMs, it is a nonvolatile memory, yet it offers the same programming flexibility as RAM does.
  - Several of today's small and medium-sized controllers use EEPROM as the only memory within the system.
  - It provides permanent storage for the program and can be easily changed with the use of a programming device (e.g., a PC) or a manual programming unit.

PLC: Memory calculation

Determine the memory requirements for an application with the following specifications:

- 70 outputs, with each output driven by logic composed of 10 contact elements
- 11 timers and 3 counters, each having 8 and 5 elements, respectively
- 20 instructions that include addition, subtraction, and comparison, each driven by 5 contact elements

- Table 9.1 provides information about the application's memory utilization requirements.

Instruction	Words of Memory Required
Examine ON or OFF (contacts)	1
Output coil	1
Add/subtract/compare	1
Timer/counter	3

Table 9.1 Memory utilization requirements.

- Using the given information, a preliminary estimation of memory is:

(a) Control logic = 10 contact elements/output rung, Number of output rungs = 70
(b) Control logic = 8 contact elements/timer, Number of timers = 11
(c) Control logic = 5 contact elements/counter, Number of counters = 3
(d) Control logic = 5 contact elements/math and compare, Number of math and compare = 20

PLC: Memory calculation

- Based on the memory utilization information from Table 9.1, the total number of words is:

(a) Total contact elements	(70 x 10)	700
Total outputs	(70 x 1)	70
Total words		770
(b) Total contact elements	(11 x 8)	88
Total timers	(11 x 3)	33
Total words		121
(c) Total contact elements	(3 x 5)	15
Total counters	(3 x 3)	9
Total words		24
(d) Total contact elements	(20 x 5)	100
Total math and compare	(20 x 1)	20
Total words		120

- Thus, the total words of memory required for the storage of the instructions, outputs, timers, and counters is 1035 words (770 + 121 + 24 + 120), or just over 1K of memory.

PLC: Discrete input module

- connect real-world switches to the PLC and are available for either AC or DC voltages (typically, 240 Vac, 120 Vac, 24 Vdc, and 5 Vdc).
- the module converts the switched voltage into a logic voltage for the processor.
- An AC/DC input circuit has two primary parts:
  - the power section
  - the logic section

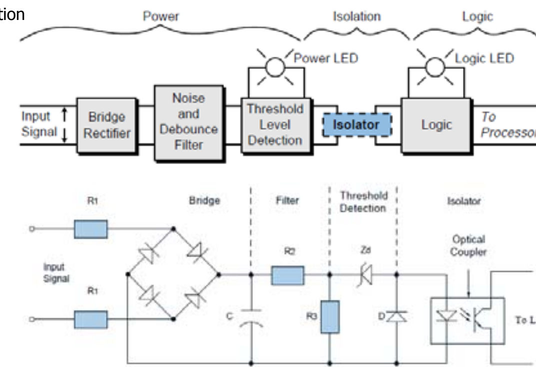


Figure 9.7
Typical AC/DC input circuit.

PLC: Discrete input module

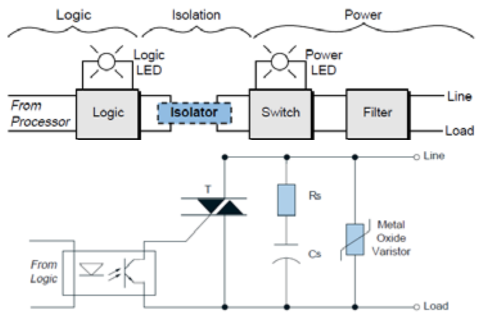
- These sections are normally, but not always, coupled through a circuit that electrically separates them, providing isolation.
- The power section of an AC/DC input interface converts the incoming AC voltage from an input-sensing device to a DC, logic-level signal that the processor can use during the read input section of its scan.
- During this process, the bridge rectifier circuit of the interface's power section converts the incoming AC signal to a DC-level signal.
- It then passes the signal through a filter circuit, which protects the signal against bouncing and electrical noise on the input power line. This filter causes a signal delay of typically 9–25 msec.
- The power section's threshold circuit detects whether the signal has reached the proper voltage level for the specified input rating. If the input signal exceeds and remains above the threshold voltage for a duration equal to the filter delay, the signal is recognized as a valid input.
- After the interface detects a valid signal, it passes the signal through an isolation circuit, which completes the electrically isolated transition from an AC signal to a DC, logic-level signal.

PLC: Discrete input module

- The logic circuit then makes the DC signal available to the processor through the rack's back plane data bus, a pathway along which data moves.
- The signal is electrically isolated so that there is no electrical connection between the field device (power) and the controller (logic).
  - This electrical separation helps prevent large voltage spikes from damaging either the logic side of the interface or the PLC.
  - An optical coupler or a pulse transformer provides the coupling between the power and logic sections.
- A typical discrete input module would have 4, 8, or 16 inputs.

PLC: Discrete output module

- provide on-off signals to drive lamps, relays, small motors, motor starters, and other devices.
- several types of output ports are available:
    1. Triac outputs control AC devices,
    2. Transistor switches control DC devices, and
    3. Relays control AC or DC devices.



Fugure 9.8 AC output circuit block diagram.

PLC: Discrete output module

Triac outputs control AC devices

- The switching circuit in the power section of an AC output module uses either a triac or a silicon controlled rectifier (SCR) to switch power.
- The AC switch is normally protected by an RC snubber and/or a metal oxide varistor (MOV), which limits the peak voltage to some value below the maximum rating.
- Snubber and MOV circuits also prevent electrical noise from affecting the circuit operation.
- Furthermore, an AC output circuit may contain a fuse that prevents excessive current from damaging the switch. If the circuit does not contain a fuse, the user should install one that complies with the manufacturer's specifications.

PLC: Discrete output module

DC output interfaces

- control discrete DC loads by switching them ON and OFF.
- employs a power transistor to switch the load.
- Like triacs, transistors are also susceptible to excessive applied voltages and large surge currents, which can cause over dissipation and short-circuit conditions.
- To prevent these conditions, a power transistor is usually protected by a freewheeling diode placed across the load (field output device).
- DC outputs may also incorporate a fuse to protect the transistor during moderate overloads. These fuses are capable of opening, or breaking continuity, quickly before excessive heat due to over currents occurs.

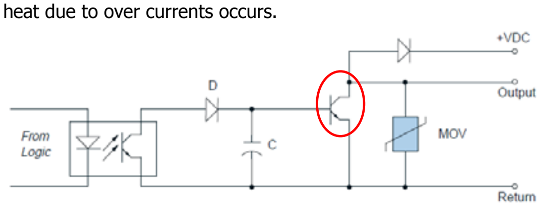


Figure 9.9 Typical sourcing DC output circuit.

PLC: Discrete output module

Contact output interfaces

- allow output devices to be switched by normally open or normally closed relay contacts.
- provide electrical isolation between the power output signal and the logic signal through separation between contacts and between the coil and contacts.
- These outputs also include filtering, suppression, and fuses.
- When the processor sends status data (1 or 0) to the module during the output update, the state of the contacts changes.
    - If the processor sends a 1 to the module, normally open contacts close and normally closed contacts open.
    - If the processor sends a 0, no change occurs to the normal state of the contacts.

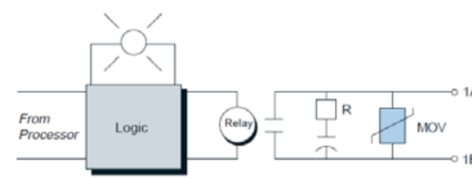


Figure 9.10 Contact output circuit.

PLC: Analogue I/O module

- allow the PLC to handle analog signals.
- An *analog input module* has one or more ADCs (analog-to-digital converters), allowing analog sensors, such as temperature, to be connected directly to the PLC.
- Depending on the module, the analog voltage or current is converted into an 8-, 12-, or 16-bit digital word.
- An *analog output module* contains one or more DACs (digital to- analog converters), allowing the PLC to provide an analog output
  - for example, to drive a DC motor at various voltage levels.
- Specialized modules that perform particular functions are available for many PLCs.
- Examples include:
  - Thermocouple module: Interfaces a thermocouple to the PLC.
  - Motion-control module: Runs independently to control muti-axis motion in a device such as a robot (covered later in this chapter).
  - Communication module: Connects the PLC to a network.
  - High-speed counter module: Counts the number of input pulses for a fixed period of time.
  - PID module: An independently running PID self-contained controller.

PLC: Analogue I/O module

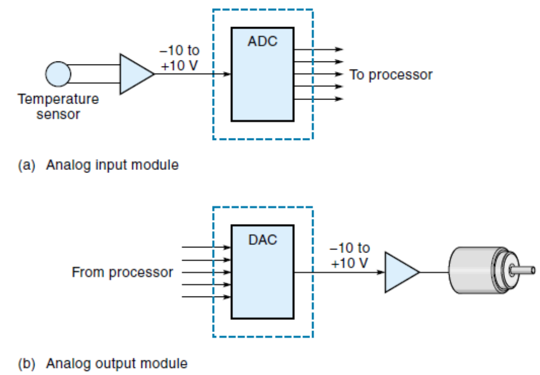


Figure 9.11 Analog I/O modules

PLC: Analogue I/O module

- An input module, which is connected to a temperature transducer, has an A/D with a 12-bit resolution (see Figure 9.12). When the temperature transducer receives a valid signal from the process (100 to 600°C), it provides, via a transmitter, a +1 to +5 VDC signal compatible with the analog input module.

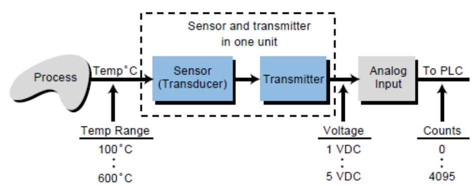


Figure 9.12 An A/D and an analog input module connected to a temperature sensing device.

(a) Find the equivalent voltage change for each count change (the voltage change per degree Celsius change) and the equivalent number of counts per degree Celsius, assuming that the input module transforms the data into a linear 0 to 4095 counts, and

(b) find the same values for a module with a 10-bit resolution.

PLC: Analogue I/O module

(a) The relationship between temperature, voltage signal, and module counts is:

Temperature	Voltage Signal	Input Counts
100°C	1 VDC	0
.	.	.
.	.	.
.	.	.
600°C	5 VDC	4095

The changes (Δ) in temperature, voltage, and input counts are 500°C, 4 VDC, and 4095 counts. Therefore, the voltage change for a 1°C temperature change is:

$$\Delta 500\,^\circ C = \Delta 4 \text{ VDC}$$
$$1^\circ C = \frac{4 \text{ VDC}}{500} = 8.0 \text{ mVDC}$$

The change in voltage for each input count is:

$$\Delta 4095 \text{ counts} = \Delta 4 \text{ VDC}$$
$$1 \text{ count} = \frac{4 \text{ VDC}}{4095} = 0.9768 \text{ mVDC}$$

Therefore, the corresponding number of counts per degree Celsius is:

$$\Delta 500^\circ C = \Delta 4095 \text{ counts}$$
$$1^\circ C = \frac{4095 \text{ counts}}{500} = 8.19 \text{ counts}$$

PLC:

PLC Bus

- the wires in the *backplane* of a PLC modular system,
  - contains the data bus, address bus, and control signals.
  - The processor uses the bus to communicate with the modules.

Programming PLC

- The three types of programming languages used in PLCs are:
  - ladder
  - Boolean
  - Grafcet
- The ladder and Boolean languages essentially implement operations in the same way, but they differ in the way their instructions are represented and how they are entered into the PLC.
- The Grafcet language implements control instructions in a different manner, based on steps and actions in a graphic oriented program.

Ladder language

- The PLC accomplishes its job of implementing the ladder logic control program in typical computer fashion—one step at a time.
- let's examine a simple PLC appli-cation of turning lamps on and off.
- The PLC has one input module and one output module.
- Two external switches (SW-0 and SW-1) are connected to the PLC via terminals *IN-0* and *IN-1* of the input module.
- Two terminals of the output module (*OUT-0* and *OUT-1*) drive two indicator lamps (LAMP-0 and LAMP-1).
- The ladder diagram for this application has only two rungs.

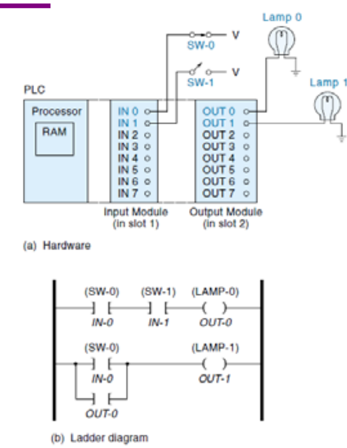


Figure 9.13 A simple PLC application and ladder diagram.

Ladder language

- The top rung will light LAMP-0 if *both* SW-0 and SW-1 are closed.
- The bottom rung will light LAMP-1 if *either* SW-0 or *OUT-0* are closed.
- (The *OUT-0* contacts can be thought of as NO relay contacts of coil *OUT-0* in rung 1.)

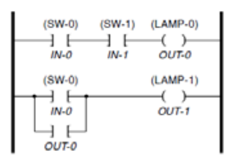


Figure 9.13 ladder diagram. (repetition)

Note

- PLC ladder diagram is basically a logic diagram consisting of symbols (called input and output instructions), and a rung becomes continuous when there is a logical TRUE path from rail to rail.

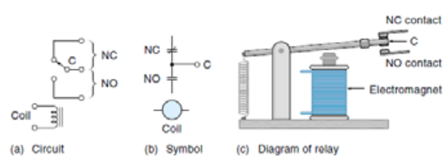


Figure 9.14 A relay: showing normally closed (NC) and normally open (NO) contacts.

Programming the PLC: Ladder Diagram Programming

- The most common way to program the PLC is to design the desired control circuit in the form of a relay logic ladder diagram and then enter this ladder diagram into a programming terminal
  - (e.g. PC, a dedicated PLC programming terminal, or a special handheld switch box).
- The programming terminal is capable of converting the ladder diagram into digital codes and then sending this program to the PLC where it is stored in memory.
- The PLC deals with two kinds of data that are maintained in memory:
  - a **program file** in RAM or EEPROM, which contains the ladder logic program
  - **data files** in RAM, which hold the changing data from the control application, such as switch settings.

Programming the PLC: Ladder Diagram Programming

- Table below lists some of the more common types of data files.
- Each data file occupies a portion of RAM memory and consists of any number of 8- or 16-bit words.

Types of Data Files

	Data file	Description
I	Input Data	The current status of externally connected switches
O	Output Data	The status of those devices specified as outputs
B	Bit Data	The "scratch pad" to store individual bits
T	Timer Data	The data associated with timers
C	Counter Data	The data associated with counters
R	Control Data	The data associated with sequencers and other devices
N	Integer Data	Numbers, such as temperatures (in binary form)

- Writing a PLC program involves placing instructions in rungs so as to obtain the desired result.
- Basic program components: switches, relays, timers, counters, and sequencers.

Ladder Diagram Programming: Bit Instructions

- Switches and relays are referred to as **bit instructions** because it takes only 1 bit to describe if a switch is open or closed.
- There are two kinds of switch instructions:
  - One represents a NO switch and the other a NC switch.
- The relay coils usually represented by an output symbol
- The operation of the three bit instructions are summarized as

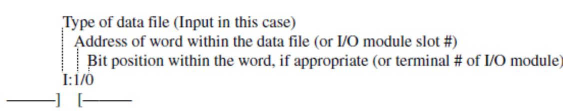
Summary of Bit Instructions

If the data file bit is	NO instruction —] [—	NC instruction —]/[—	OUTPUT —( )—
Logic 0	FALSE	TRUE	FALSE
Logic 1	TRUE	FALSE	TRUE

- If there is a continuous TRUE path through the rung, then the OUTPUT will go TRUE

Ladder Diagram Programming: Bit Instructions

- In a PLC ladder diagram, each symbol is accompanied by its RAM address because this address tells the PLC where to find the present logical state of the symbol.
- The address of an individual bit may be specified by three quantities:
  - the file type, word number, and position of the bit within the word.



- For example, in the top rung of Figure 9.15, the address of *Flow sw* is given as I : 1/0.
- The I stands for Input Data file, the 1 means that the input module is plugged into slot 1 of the PLC chassis, and the 0 means that the switch is wired to terminal 0 of the module

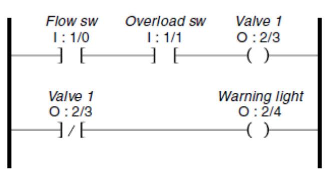


Figure 9.15
A simple PLC program.

Ladder Diagram Programming: Bit Instructions

- The second NO instruction *Overload sw* on the first rung (Figure 9.15) represents a switch wired to terminal 1 of the same input module used by *Flow sw.*
- The OUTPUT instruction activates terminal 3 of the output module plugged into slot 2.
- The operation of the top rung is as follows:
  - The *Valve 1* output will go TRUE when there is continuity through the rung.
  - Continuity will be established when both NO instructions are TRUE, i.e., when the contacts of both *Flow sw* and *Overload sw* are closed.
- Operation of the second rung illustrates two important concepts.
  - First, notice that the address of the NC instruction is the same as the OUTPUT instruction in the top rung, meaning that the NC instruction will be controlled by the *Valve 1* bit (located at address 0:2/3).
  - Second, because it is an NC instruction, it will go TRUE only when *Valve 1* is FALSE. So *Warning light* will come on when the valve goes off.

- The Timer instruction provides a time delay, performing the function of a time-delay relay.
- Examples: controlling the time for a mixing operation or the duration of a warning beep.
- The length of time delay is determined by specifying a preset value.
- The timer is enabled when the rung conditions become TRUE.
- Once enabled, it automatically counts up until it reaches the Preset value and then goes TRUE (and stays TRUE).

T : 1 The timer address (actually the first of three addresses in RAM) where the first address holds the status bits EN, TT, and DN the second address holds the preset value; and the third address holds the accumulator value.

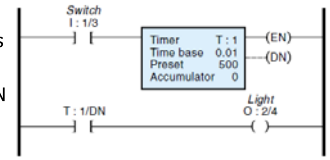


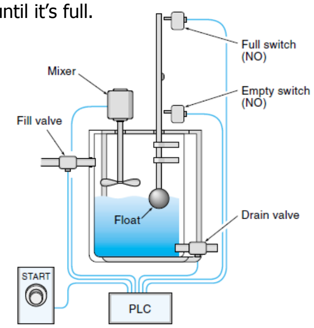
Figure 9.16 The timer instruction.

Time base	The value of 0.01 means that each count corresponds to 0.01 seconds
Preset	The value of 500 means that the delay will last 500 counts, which in this case is 5 s (0.01s × 500 = 5 s).
Accumulator	Holds the value of the current count.
EN	A bit that is TRUE as long as the timer rung is TRUE.
TT	A bit that is TRUE as long as the timer is counting, that is, is TRUE for the time delay.
DN	A bit that goes TRUE when the timer is done, i.e., when the count gets to the preset value.

A batch process—which involves filling a vat with a liquid, mixing the liquid, and draining the vat—is automated with a PLC. Figure 9.17(a) shows the hardware. The specific sequence of events is as follows: When the start button near the process is pushed:

1. A fill valve opens and lets a liquid into a vat until it's full.
2. The liquid in the vat is mixed for 3 min.
3. A drain valve opens and drains the tank.

Draw the ladder diagram for the PLC program.



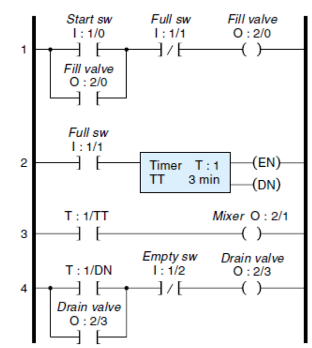
(a) Hardware

Figure 9.17
Using a PLC timer instruction to control a batch-mixing operation.

1. Rung 1 activates the fill valve and will go TRUE (and seal) when the local control start switch is momentarily activated and the tank is less than full.

- The physical float-activated NO switch (Full switch) drives the *Full sw* NC instruction in rung 1; therefore, the NC instruction will be TRUE as long as the vat is not full (it changes to FALSE when the vat is full).
- As long as the entire rung is TRUE, the OUTPUT instruction (*Fill valve*) will be TRUE, which turns on the actual fill valve. (During the fill period, none of the other rungs are TRUE.)

2. Rung 2 activates the 3-min mixing timer. When the vat is full, the physical full switch closes (goes to on), so the NO instruction (*Full sw*) goes TRUE, which starts Timer T : 1 (which has been programmed to provide the 3-min mixing time).



(b) Ladder diagram

Ladder Diagram Programming: Timers

3. Rung 3 controls the mixer motor.

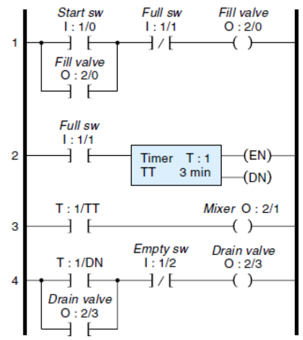
–As long as the TT bit of the timer is TRUE, the OUTPUT instruction *Mixer* will be TRUE, which activates the mixer motor.

4. Rung 4 activates the drain valve.

–After 3 min, the timer "times out," and the DN bit (T : 1/DN) goes TRUE, making the first instruction in rung 4 go TRUE.

–The NC instruction (*Empty sw*) will also be TRUE because the vat is still full (that is, the Empty switch remains unactivated).

–Therefore, rung 4 will be continuous, causing the OUTPUT instruction *Drain valve* to go TRUE, which opens the drain valve, allowing the liquid to leave.



(b) Ladder diagram

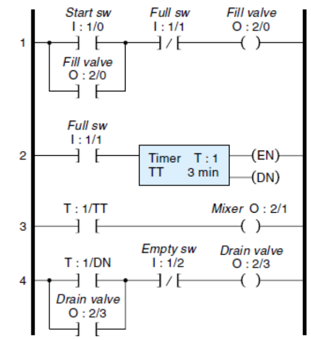
Ladder Diagram Programming: Timers

–Note: As soon as the liquid starts to drain, the full switch will deactivate, making rung 2 go FALSE, which causes the timer's DN bit to go FALSE.

–This latter action will cause the first instruction in rung 4 to go FALSE.

–So, to keep rung 4 TRUE until the vat has completely drained, the T : 1/DN instruction has a parallel branch instruction, which is activated by *Drain valve* itself.

–In effect, rung 4 is latched on until *Empty sw* activates and breaks the seal.



(b) Ladder diagram

Ladder Diagram Programming: Counters

- A Counter instruction keeps track of the number of times some event occurs.
- The count could represent the number of parts to be loaded into a box or the number of times some operation is done in a day.
- Counters may be either count-up or count-down types.
- The Counter instruction is placed in a rung and will increment (or decrement) every time the rung makes a FALSE-to-TRUE transition.
- The count is retained until a RESET instruction is enabled.
- The Counter has a Preset value associated with it.
- When the count gets up to the Preset value, the output goes TRUE. This allows the program to initiate some action based on a certain count.
- For example, after 50 items are loaded in a box, a new box is moved into place.

Ladder Diagram Programming: Counters

C : 1 The counter address (actually the first of three addresses) where the first address holds such bits as CU and DN; the second address holds the preset value; and the third address holds the accumulated value.

Preset When the counter gets to the Preset value, it makes the DN bit go TRUE and keeps on counting.

Accumulator Holds the value of the current count.

EN Goes TRUE when the counter rung is TRUE.

DN Stands for done—goes TRUE when the count meets or exceeds the PRESET value.

RESET A separate instruction with the same counter address; when this bit goes TRUE, the Accumulator resets to zero (resets the counter).

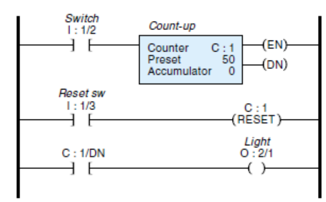


Figure 9.18 The Counter instruction.

– The above ladder diagram is a simple circuit that counts the number of times that *Switch* (address I : 1/2) is opened and closed.
– The Counter can be reset at any time by closing *Reset sw* because this action activates the RESET instruction.
– The third rung uses the DN bit from the Counter to turn on a light when the count gets to 50 or higher.

Ladder Diagram Programming: Sequencers

- The Sequencer instruction is used when a repeating sequence of outputs is required.
- Traditionally, electromechanical sequencers (Figure 9.19) were used in this type of application (where a drum rotates slowly, and cams on the drum activate switches).
- The Sequencer instruction allows the PLC to implement this common control strategy.
- Operation of PLC Sequencer is as follows:
  - The desired output-bit patterns for each step are stored (sequentially) as words in memory—as usual, each bit in the word corresponds to a specific terminal of the output module.
  - Every time the Sequencer instruction steps, it connects the next output pattern in memory to the designated output module.

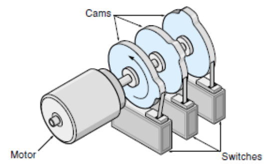


Figure 9.19 An electro-mechanical sequencer.

Ladder Diagram Programming: Sequencers

– The address of the first pattern (word) in the sequence is given in the Sequencer instruction as the Sequencer File Adr (which is address B : 1).

– The Sequencer steps when the rung makes a FALSE-to-TRUE transition.

Sequencer File Adr The first address of the Sequence file, which stores the sets of outputs.
Destination Adr The output address (or slot) to which the Sequence file words are transferred with each step.
Length The length of the Sequence file, i.e., the number of steps in the sequence.
EN Goes TRUE when the Sequencer rung is TRUE.
DN Stands for done—goes TRUE when it has operated on the last word in the Sequence file.
Control Adr Address that stores the control bits (EN, DN) and words (length) of the sequencer.

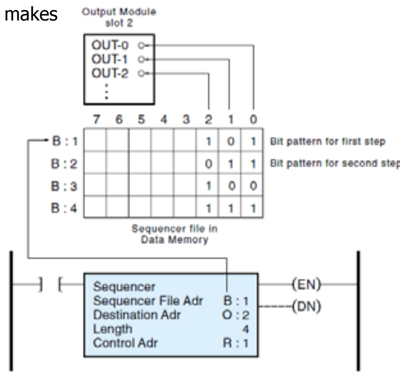


Figure 9.20 The Sequencer instruction.

Sequencers: example

A process for washing parts requires the following sequence:

1. Spray water and detergent for 2 min (*Wash cycle*).
2. Rinse with water spray only for 1 min (*Rinse cycle*).
3. Water off, air blow dry for 3 min (*Drying cycle*).

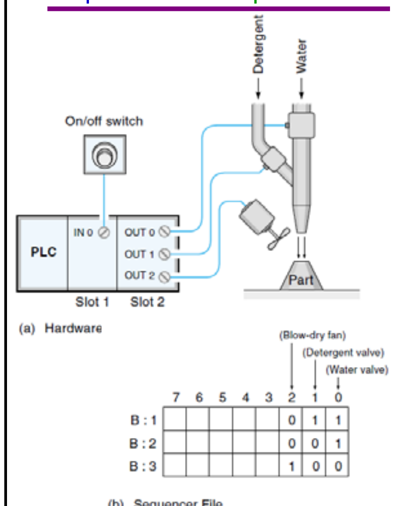
The sequence is started with a toggle switch.

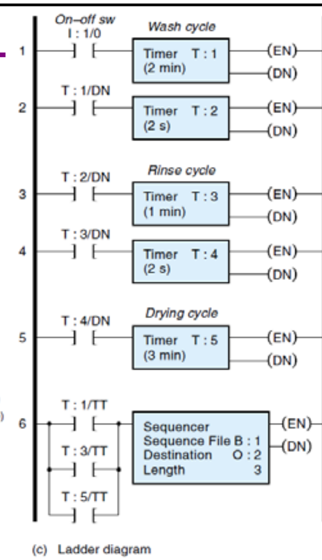
Draw the ladder diagram for this process (using the Sequencer instruction).

Solution

- The completed ladder diagram consists of six rungs.
- Rungs 1 and 2 generate the 2-min time delay for *Wash cycle*.
- Rungs 3 and 4 generate the 1-min time delay for *Rinse cycle*, and
- rung 5 generates the 3-min time delay for *Drying cycle*.
- Rung 6 has the Sequencer instruction.
- The Sequencer instruction specifies that the Sequence file starts at address B : 1, that the file is three words long, and that the output module is in slot 2.
- Looking at the Sequence file, we see that bit 0 controls the water valve, bit 1 controls the detergent valve, and bit 2 controls the blow-dry fan.

Sequencers: example





Sequencers: example

Operation Overview

- The program consists of five timer rungs and a sequencer rung.
- The timers are activated, one after the other, down the program—that is, each timer, when finished, starts the timer next in line.
- The outputs of Timers T : 1, T : 3, and T : 5 are used to step the Sequencer instruction.
- The Sequencer instruction presents three output-bit patterns to module 2.

Detailed Operation

Operation of the ladder program is as follows:

- The action starts on rung 1 when the operator switches the on-off toggle switch to on. This provides a FALSE-to-TRUE transition that starts Timer T : 1 (*Wash cycle*), causing its timing bit (TT) to go TRUE for 2 min.
- This same timing bit (T : 1/TT), makes rung 6 go TRUE, stepping the sequencer to its first position (connecting the word at address B : 1 to output module 0 : 2). Notice that bit 0 and bit 1 of word B : 1 are 1s, which cause terminals OUT-0 and OUT-1 to go on, turning on the water and detergent valves.

Sequencers: example

- After 2 min, the DN bit of Timer T : 1 (T : 1/DN) goes TRUE, which starts the next Timer (T : 2 in rung 2).
  - This timer is only 2 s long, and its purpose is to create a short gap between the steps (to allow the sequencer rung to reset).
- When Timer T : 2 is done, its DN bit (T : 2/DN) starts the 1-min Timer T : 3 (*Rinse cycle*).
  - Timer T : 3 timing bit (T : 3/TT) causes the Sequencer instruction to advance to the second step (B : 2).
  - Notice that in word B : 2, only bit 0 is a 1, causing OUT-0 to remain on (leaving the water on, but turning off the detergent).
- Following in the same manner, Timer T : 4 provides a short gap, and then Timer T : 5 advances the sequencer to the third and last position (B : 3) for the 3-min *Drying cycle*.
  - Notice that bit 2 of word B : 3 becomes OUT-2, and turns on the blow-dry fan.

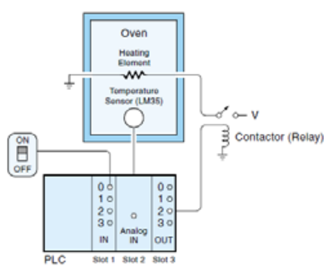
Using a PLC as a Two-Point Controller: example

The temperature in an electric oven is to be maintained by a 16-bit PLC at approximately 100°C, using two-point control (actual range: 98-102°).

Figure 12.25(a) shows the system hardware: an oven with an electric heating element driven by a *contactor* (high-current relay), an LM35 temperature sensor produces 10 mV/°C), an operator on-off switch, and the PLC. The PLC has a processor and three I/O modules: a discrete input module (slot 1), a 16-bit analog input module (slot 2), and a discrete output module (slot 3).

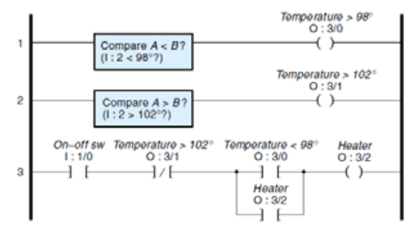
Draw the ladder diagram for this system.

– When the oven temperature falls below 98°, the heating element will turn on and stay on until the temperature reaches 102°. Thus, the PLC will be giving a discrete output (to the heater), based on an analog input (temperature).



Using a PLC as a Two-Point Controller: example

1. Rung 1 determines if the temperature is below 98°C and will go TRUE when the conditions of the Compare instruction are met.
   - This particular Compare instruction compares two 16-bit words and goes TRUE if the value of word A is less than word B.
   - In this case, word A is the actual temperature (as converted by the ADC and stored at address I : 2), and word B is the binary equivalent of the lower-limit temperature of 98°.
   - Therefore, the rung will go TRUE if the oven temperature is less than 98°. The value of this rung is stored in bit 0 : 3/0.



(b) Ladder diagram

2. Rung 2 determines if the temperature is above 102°C.
   - This rung has another Compare instruction, but this one goes TRUE if word A is greater than word B.

Using a PLC as a Two-Point Controller: example

- Notice that word A is again the actual temperature (from the ADC), and word B is the upper-limit temperature of 102°.
- Therefore, the rung will go TRUE if the oven temperature is greater than 102°.
- The value of this rung is stored in bit 0 : 3/1.

3. Rung 3 activates the heating element via the control logic that turns the heating element on and off—that is, the OUTPUT instruction *Heater* (0 : 3/2) directly controls the heating element.

– The rung will go TRUE if the on-off switch is on, *and* the temperature is not over 102° (notice this is an NC instruction), *and* the temperature is less than 98°.

– Once *Heater* is on, it is sealed on with the parallel branch 0 : 3/2 so that, even when the oven temperature rises above 98°, the rung will stay TRUE.

– With the heating element on, the oven temperature will eventually rise to above 102°, and so the NC instruction (*Temperature > 102°*) will go FALSE, breaking the seal and turning *Heater* off.

– The rung will stay FALSE until the temperature drops below 98°, and then the cycle starts over.