

Documentation on

PROGRAMMABLE LOGIC CONTROLLER (PLC)

Prepared by

Dr. M.CHAKRAVARTHY

Professor and HOD-EEE



DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING

VASAVI COLLEGE OF ENGINEERING (AUTONOMOUS)

LIST OF CONTENTS

1. INTRODUCTION OF PLC

2. TYPES OF INPUT AND OUTPUT DEVICES

3. PROGRAMMING OF MILLENIUM PLC

4. FUNCTIONAL BLOCK DIAGRAM (FBD)

- 4.1 Study of basic control function
- 4.2 Implementation of logic gates and Boolean functions
- 4.3 Study of PLC timer functions
- 4.4 Study of PLC counters functions
- 4.5 Study of PLC Arithmetic functions
- 4.6 Study of Number Comparison functions
- 4.7 Study of sequencer
- 4.8 Applications
 - 4.8.1 Motor control using PLC
 - 4.8.2 Sequential lighting of bulbs
 - 4.8.3 Automatic Traffic control
 - 4.8.4 Industrial applications

5. LADDER PROGRAMMING (LAD)

- 5.1 Study of basic control function
- 5.2 Implementation of logic gates and Boolean functions
- 5.3 Study of PLC timer functions
- 5.4 Study of PLC counters functions
- 5.5 Study of Number Comparison functions
- 5.6 Applications
 - 5.6.1 Motor control using PLC
 - 5.6.2 Sequential lighting of bulbs

1. INTRODUCTION OF PLC

A PLC is a user-friendly, microprocessor based specialized computer that carries out control functions of many types and level of complexity. Its purpose is to monitor crucial process parameters and adjust process operations accordingly. It can be programmed, controlled and operated by a person. Essentially a PLC operator draws the lines and devices of ladder diagram and functional block diagram with a keyboard onto the display screen. The resulting drawing is converted into computer machine language and run as a user program.

1.1 Programmable Logic Controller

A programmable logic controller (PLC) is a special form of microprocessor-based controller that uses programmable memory to store instructions and to implement functions such as logic, sequencing, timing, counting, and arithmetic in order to control machines and processes. It is designed to be operated by engineers with perhaps a limited knowledge of computers and computing languages. They are not designed so that only computer programmers can set up or change the programs. Thus, the designers of the PLC have pre programmed it so that the control program can be entered using a simple, rather intuitive form of language. The term logic is used because programming is primarily concerned with implementing logic and switching operations; for example, if A or B occurs, switch on C; if A and B occurs, switch on D. Input devices (that is, sensors such as switches) and output devices (motors, valves, etc.) in the system being controlled are connected to the PLC. The operator then enters a sequence of instructions, a program, into the memory of the PLC. The controller then monitors the inputs and outputs according to this program and carries out the control rules for which it has been programmed. PLCs have the great advantage that the same basic controller can be used with a wide range of control systems..

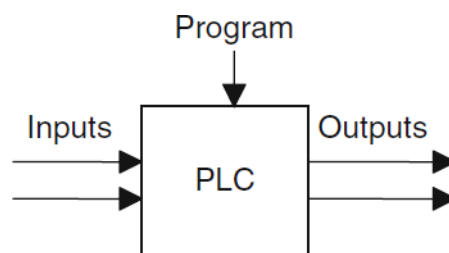


Fig 1. A Programmable Logic Controller

Programmable Logic Controller

To modify a control system and the rules that are to be used, all that is necessary is for an operator to key in a different set of instructions. There is no need to rewire. The result is a flexible, cost-effective system that can be used with control systems, which vary quite widely in their nature and complexity. PLCs are similar to computers, but whereas computers are optimized for calculation and display tasks, PLCs are optimized for control tasks and the industrial environment. Thus PLCs:

- Are rugged and designed to withstand vibrations, temperature, humidity, and noise
- Have interfacing for inputs and outputs already inside the controller
- Are easily programmed and have an easily understood programming language that is primarily concerned with logic and switching operations

The first PLC was developed in 1969. PLCs are now widely used and extend from small, self-contained units for use with perhaps 20 digital inputs/outputs to modular systems that can be used for large numbers of inputs/outputs, handle digital or analog inputs/outputs, and carry out proportional-integral-derivative control modes.

Application Areas

Programmable Logic Controllers are suitable for a variety of automation tasks. They provide a simple and economic solution to many automation tasks such as

1. Logic/Sequence control
2. PID control and computing
3. Coordination and communication
4. Operator control and monitoring
5. Plant start-up, shut-down

Any manufacturing application that involves controlling repetitive, discrete operations is a potential candidate for PLC usage, e.g. machine tools, automatic assembly equipment, molding and extrusion machinery, textile machinery and automatic test equipment. Some typical industrial areas that widely deploy PLC controls are as follows.

1. Chemical/ Petrochemical Metals
 - a) Batch process
 - b) Pipeline Control
 - c) Weighing, Mixing
 - d) Finished Product Handling

Programmable Logic Controller

2. Metal

- a) Blast Furnace
- b) Continuous Casting
- c) Rolling Mills
- d) Soaking Pit
- e) Steel Melting Shop

3. Manufacturing/Machining

- a) Material Conveyors, Cranes
- b) Assembly
- c) Milling, Grinding, Boring
- d) Plating, Welding, Painting
- e) Molding/ casting/forming

Hardware

Typically a PLC system has the basic functional components of processor unit, memory, power supply unit, input/output interface section, communications interface, and the programming device in the basic arrangement.

- The processor unit or central processing unit (CPU) is the unit containing the microprocessor. This unit interprets the input signals and carries out the control actions according to the program stored in its memory, communicating the decisions as action signals to the outputs.

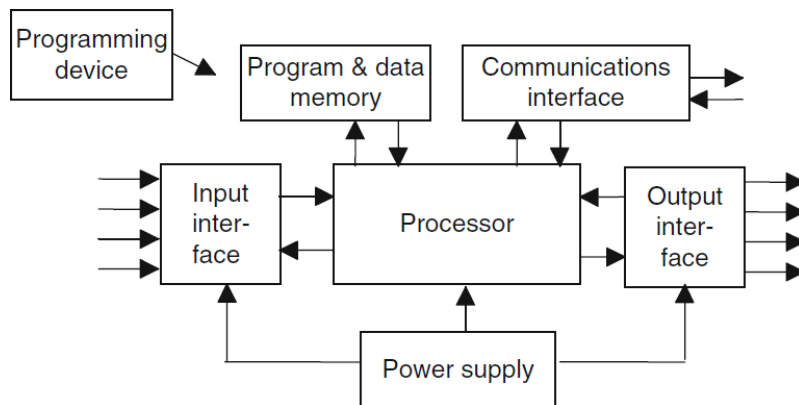


Fig 2. The PLC System

Programmable Logic Controller

- The power supply unit is needed to convert the mains AC voltage to the low DC voltage (5V) necessary for the processor and the circuits in the input and output interface modules.
- The programming device is used to enter the required program into the memory of the processor. The program is developed in the device and then transferred to the memory unit of the PLC.



Fig 3. Types of signals: Discrete, Digital and Analog.

- The memory unit is where the program containing the control actions to be exercised by the microprocessor is stored and where the data is stored from the input for processing and for the output.
- The input and output sections are where the processor receives information from external devices and communicates information to external devices. The inputs might thus be from switches, as illustrated in Figure with the automatic drill, or other sensors such as photoelectric cells, as in the counter mechanism in Figure, temperature sensors, flow sensors, or the like. The outputs might be to motor starter coils, solenoid valves, or similar things. Input and output devices can be classified as giving signals that are discrete, digital or analog. Devices giving discrete or digital signals are ones where the signals are either off or on. Thus a switch is a device giving a discrete signal, either no voltage or a voltage. Digital devices can be considered essentially as discrete devices that give a sequence of on/off signals. Analog devices give signals of which the size is proportional to the size of the variable being monitored. For example, a temperature sensor may give a voltage proportional to the temperature.
- The communications interface is used to receive and transmit data on communication networks from or to other remote PLCs. It is concerned with such actions as device verification, data acquisition, synchronization between user applications, and connection management.

Programmable Logic Controller

Internal Architecture

Figure shows the basic internal architecture of a PLC. It consists of a central processing unit (CPU) containing the system microprocessor, memory, and input/output circuitry. The CPU controls and processes all the operations within the PLC. It is supplied with a clock.

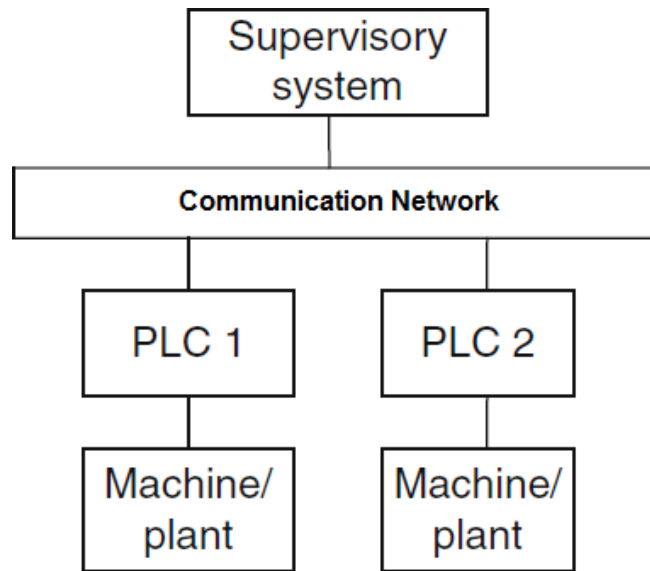


Fig 4. Basic Communication Model

that has a frequency of typically between 1 and 8 MHz. This frequency determines the operating speed of the PLC and provides the timing and synchronization for all elements in the system. The information within the PLC is carried by means of digital signals. The internal paths along which digital signals flow are called buses. In the physical sense, a bus is just a number of conductors along which electrical signals can flow. It might be tracks on a printed circuit board or wires in a ribbon cable.

The CPU uses the data bus for sending data between the constituent elements, the address bus to send the addresses of locations for accessing stored data, and the control bus for signals relating to internal control actions. The system bus is used for communications between the input/output ports and the input/output unit.

Programmable Logic Controller

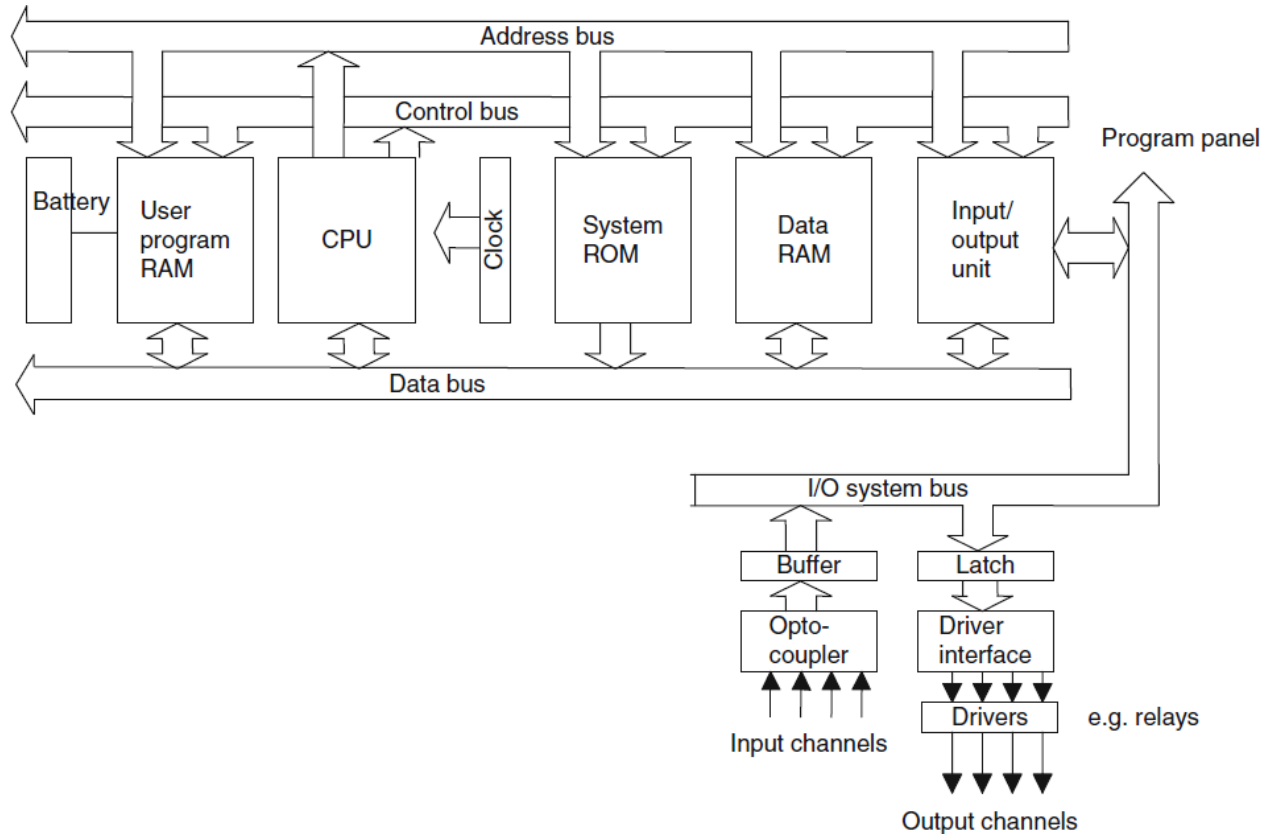


Fig 5. Architecture of PLC

CPU

The internal structure of the CPU depends on the microprocessor concerned. In general, CPUs have the following:

- An arithmetic and logic unit (ALU) that is responsible for data manipulation and carrying out arithmetic operations of addition and subtraction and logic operations of AND, OR, NOT, and EXCLUSIVE-OR.
- Memory, termed registers, located within the microprocessor and used to store information involved in program execution.
- A control unit that is used to control the timing of operations.

Programmable Logic Controller

Buses

The buses are the paths used for communication within the PLC. The information is transmitted in binary form, that is, as a group of bits, with a bit being a binary digit of 1 or 0, indicating on/off states. The term word is used for the group of bits constituting some information. Thus an 8-bit word might be the binary number 00100110. Each of the bits is communicated simultaneously along its own parallel wire. The system has four buses:

- The data bus carries the data used in the processing done by the CPU. A microprocessor termed as being 8-bit has an internal data bus that can handle 8-bit numbers. It can thus perform operations between 8-bit numbers and deliver results as 8-bit values.
- The address bus is used to carry the addresses of memory locations. So that each word can be located in memory, every memory location is given a unique address. Just like houses in a town are each given a distinct address so that they can be located, so each word location is given an address so that data stored at a particular location can be accessed by the CPU, either to read data located there or put, that is, write, data there. It is the address bus that carries the information indicating which address is to be accessed. If the address bus consists of eight lines, the number of 8-bit words, and hence number of distinct addresses, is $2^8 = 256$. With 16 address lines, 65,536 addresses are possible.
- The control bus carries the signals used by the CPU for control, such as to inform memory devices whether they are to receive data from an input or output data and to carry timing signals used to synchronize actions.
- The system bus is used for communications between the input/output ports and the input/output unit.

Memory

To operate the PLC system there is a need for it to access the data to be processed and instructions, that is, the program, which informs it how the data is to be processed. Both are stored in the PLC memory for access during processing. There are several memory elements in a PLC system:

- System Read-Only-Memory (ROM) gives permanent storage for the operating system and fixed data used by the CPU.

Programmable Logic Controller

- Random-Access Memory (RAM) is used for the user's program.
- Random-access memory (RAM) is used for data. This is where information is stored on the status of input and output devices and the values of timers and counters and other internal devices. The data RAM is sometimes referred to as a data table or register table. Part of this memory, that is, a block of addresses, will be set aside for input and output addresses and the states of those inputs and outputs. Part will be set aside for preset data and part for storing counter values, timer values, and the like.
- Possibly, as a bolt-on extra module, Erasable and Programmable Read-only-memory (EPROM) is used to store programs permanently. The programs and data in RAM can be changed by the user. All PLCs will have some amount of RAM to store programs that have been developed by the user and program data. However, to prevent the loss of programs when the power supply is switched off, a battery is 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 memory chip, often a bolt-on module to the PLC, and so made permanent. In addition, there are temporary buffer stores for the input/output channels.

Input/Output Unit

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 that can be used by the CPU. It is like a row of houses along a road; number 10 might be the "house" used for an input from a particular sensor, whereas number 45 might be the "house" used for the output to a particular motor.

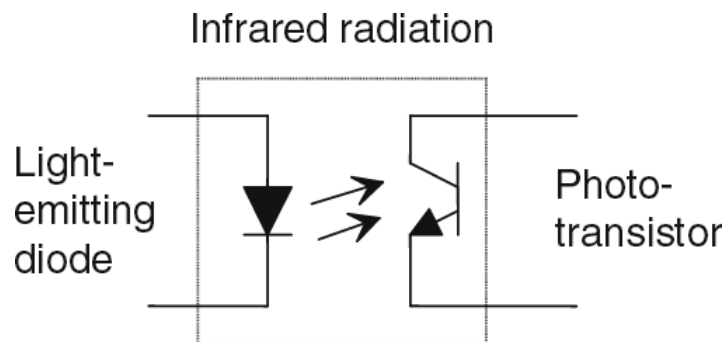


Fig 6. An Opto-isolator

Programmable Logic Controller

The input/output channels provide isolation and signal conditioning functions so that sensors and actuators can often be directly connected to them without the need for other circuitry. Electrical isolation from the external world is usually by means of opto-isolators (the term opto-coupler is also often used). Figure shows the principle of an opto-isolator. When a digital pulse passes through the light-emitting diode, a pulse of infrared radiation is produced. This pulse is detected by the phototransistor and gives rise to a voltage in that circuit. The gap between the light-emitting diode and the phototransistor gives electrical isolation, but the arrangement still allows for a digital pulse in one circuit to give rise to a digital pulse in another circuit.

The digital signal that is generally compatible with the microprocessor in the PLC is 5 V DC. However, signal conditioning in the input channel, with isolation, enables a wide range of input signals to be supplied to it.

A range of inputs might be available with a larger PLC, such as 5 V, 24 V, 110 V, and 240 V digital/discrete, that is, on/ off, signals. A small PLC is likely to have just one form of input, such as 24 V.

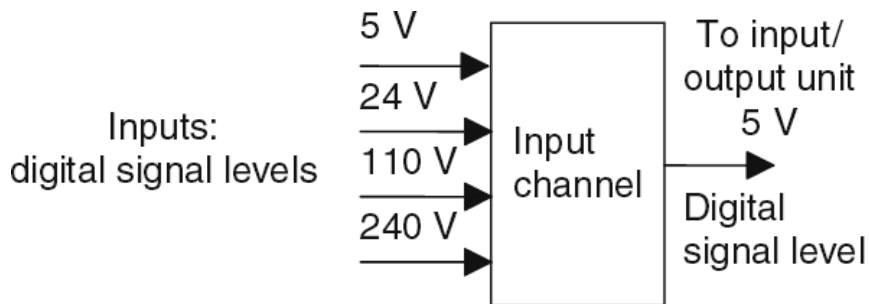


Fig 7. Input Levels

The output from the input/output unit will be digital with a level of 5 V. However, after signal conditioning with relays, transistors, or TRIACs, the output from the output channel might be a 24 V, 100 mA switching signal; a DC voltage of 110 V, 1 A; or perhaps 240 V, 1A AC or 240 V, 2 A AC, from a triac output channel. With a small PLC, all the outputs might be of one type, such as 240 V, 1 A AC. With modular PLCs, however, a range of outputs can be accommodated by selection of the modules to be used. Outputs are specified as being of relay type, transistor type, or triac type.

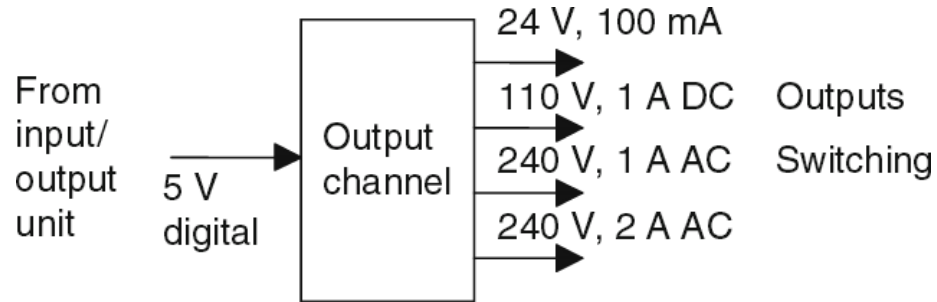


Fig 8. Output Levels

- With the relay type, the signal from the PLC output is used to operate a relay and is able to switch currents of the order of a few amperes in an external circuit. The relay not only allows small currents to switch much larger currents but also isolates the PLC from the external circuit. Relays are, however, relatively slow to operate. Relay outputs are suitable for AC and DC switching. They can withstand high surge currents and voltage transients.
- The transistor type of output uses a transistor to switch current through the external circuit. This gives a considerably faster switching action. It is, however, strictly for DC switching and is destroyed by over-current and high reverse voltage. For protection, either a fuse or built-in electronic protection is used. Opto-isolators are used to provide isolation.
- Triac outputs, with opto-isolators for isolation, can be used to control external loads that are connected to the AC power supply. It is strictly for AC operation and is very easily destroyed by over-current. Fuses are virtually always included to protect such outputs.

Sourcing and Sinking

The terms sourcing and sinking are used to describe the way in which DC 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, that is, the input module is the source of the current. With sinking, using the conventional current flow direction, an input device supplies current to the input module, that is, the input module is the sink for the current.

Programmable Logic Controller

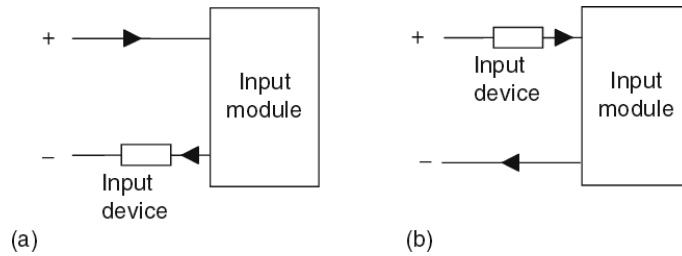


Fig 9. Inputs: a) Sourcing b) Sinking

If the current flows from the output module to an output load, the output module is referred to as sourcing. If the current flows to the output module from an output load, the output module is referred to as sinking.

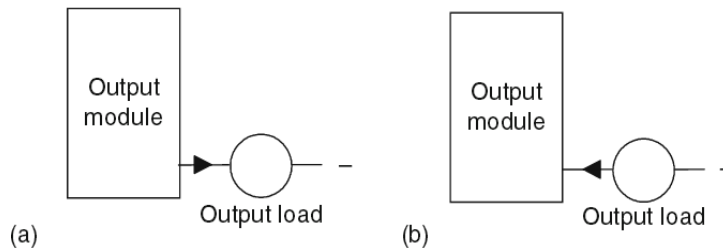


Fig 10. Outputs: a) Sourcing b) Sinking

It is important to know the type of input or output concerned so that it can be correctly connected to the PLC. Thus, sensors with sourcing outputs should be connected to sinking PLC inputs and sensors with sinking outputs should be connected to sourcing PLC inputs. The interface with the PLC will not function and damage may occur if this guideline is not followed.

Now we can summarize that programmable logic controller (PLC) is a special form of microprocessor-based controller that uses a programmable memory to store instructions and to implement functions such as logic, sequencing, timing, counting, and arithmetic to control machines and processes and is designed to be operated by engineers with perhaps a limited knowledge of computers and computing languages. Typically, a PLC system has the basic functional components of processor unit, memory, power supply unit, input/output interface section, communications interface, and programming device. To operate the PLC system there is a need for it to access the data to be processed and the instructions, that is, the program, that informs it how the data is to be processed. Both are stored in the PLC memory for access during processing. The input/output channels provide isolation and signal conditioning functions so that sensors and actuators can often be directly connected to them without the need for other

Programmable Logic Controller

circuitry. Outputs are specified as being of relay type, transistor type, or triac type. The communications interface is used to receive and transmit data on communications networks from or to other remote PLCs. There are two common types of mechanical design for PLC systems—a single box and the modular/rack types.

The IEC 61131 defined the standards for PLCs, with 61131-3 defining the programming languages: ladder diagrams (LAD), instruction list (IL), sequential function charts (SFC), structured text (ST), and function block diagrams (FBD).

2. TYPES OF INPUT AND OUTPUT DEVICES

I/O Processing

The input/output (I/O) unit provides the interface between the PLC controller and the outside world and must therefore provide the necessary signal conditioning to get the signal to the required level and also to isolate it from possible electrical hazards such as high voltages. It includes the forms of typical input/output modules and, in an installation where sensors are some distance from the PLC processing unit, and their communication links to the PLC.

Input/Output Units

Input signals from sensors and outputs required for actuating devices can be:

- **Analog:** A signal for which the size is related to the size of the quantity being sensed.
- **Discrete:** Essentially just an on/off signal.
- **Digital:** A sequence of pulses.

The CPU, however, must have an input of digital signals of a particular size, normally 0 to 5 V. The output from the CPU is digital, normally 0 to 5 V. Thus there is generally a need to manipulate input and output signals so that they are in the required form. The input/output (I/O) units of PLCs are designed so that a range of input signals can be changed into 5 V digital signals and so that a range of outputs are available to drive external devices. It is this built-in facility to enable a range of inputs and outputs to be handled that makes PLCs so easy to use..

Input Units

The terms sourcing and sinking refer to the manner in which DC devices are interfaced with the PLC. For a PLC input unit with sourcing, it is the source of the current supply for the input device connected to it (Figure 1 a). With sinking, the input device provides the current to the input unit (Figure 1 b).

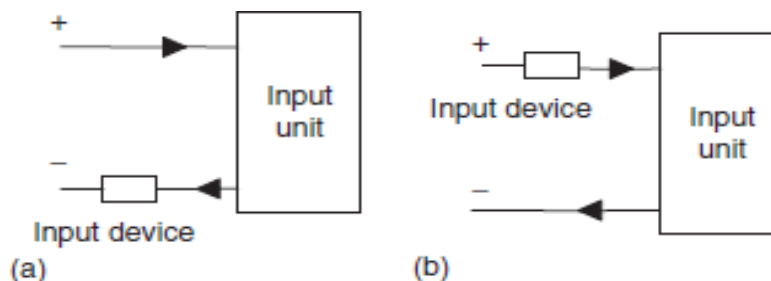


Figure 1: Input unit: (a) sourcing and (b) sinking.

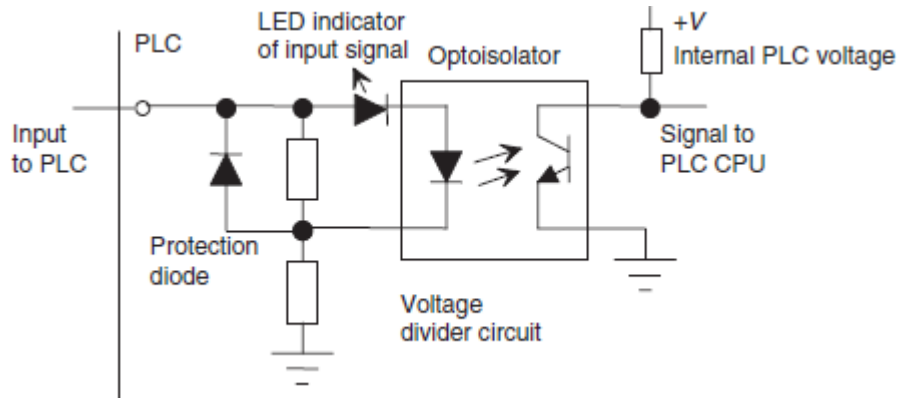


Figure 2: DC input unit.

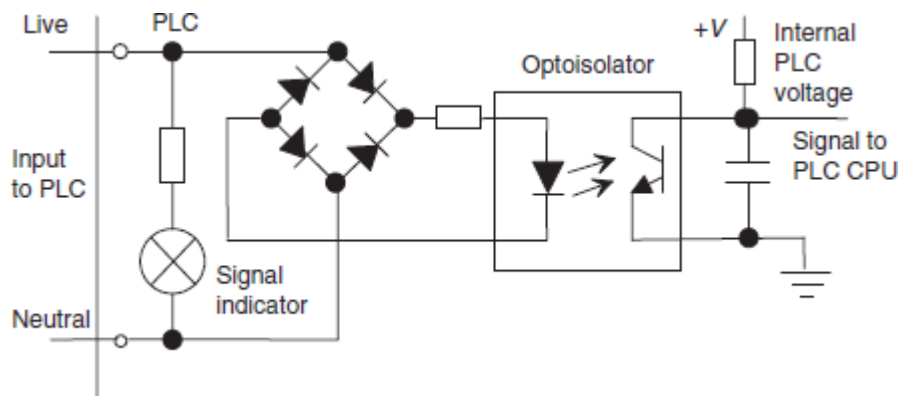


Figure 3: AC input unit.

Figures 2 and 3 show the basic input unit circuits for DC and AC inputs. Opto-isolators are used to provide protection. With the AC input unit, a rectifier bridge network is used to rectify the AC so that the resulting DC signal can provide the signal for use by the opto-isolator to give the input signals to the CPU of the PLC. Individual status lights are provided for each input to indicate when the input device is providing a signal.

Programmable Logic Controller

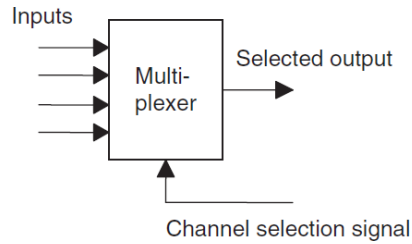


Figure 4: Multiplexer.

When analog signals are inputted to a PLC, the input channel needs to convert the signal to a digital signal using an analog-to-digital converter. With a rack-mounted system this may be achieved by mounting a suitable analog input card in the rack. So that one analog card is not required for each analog input, multiplexing is generally used (Figure 4). This involves more than one analog input being connected to the card and then electronic switches used to each input in turn. Cards are typically available containing 4, 8, or 16 analog inputs. Figure 5a illustrates the function of an analog-to-digital converter (ADC). A single analog input signal gives rise to on/off output signals along perhaps eight separate wires. The eight signals then constitute the so-called digital word corresponding to the analog input signal level. With such an 8-bit converter there are $2^8 = 256$ different digital values possible; these are 0000 0000 to 1111 1111, that is, 0 to 255. The digital output goes up in steps (Figure 5b) and the analog voltages required to produce each digital output are termed quantization levels.

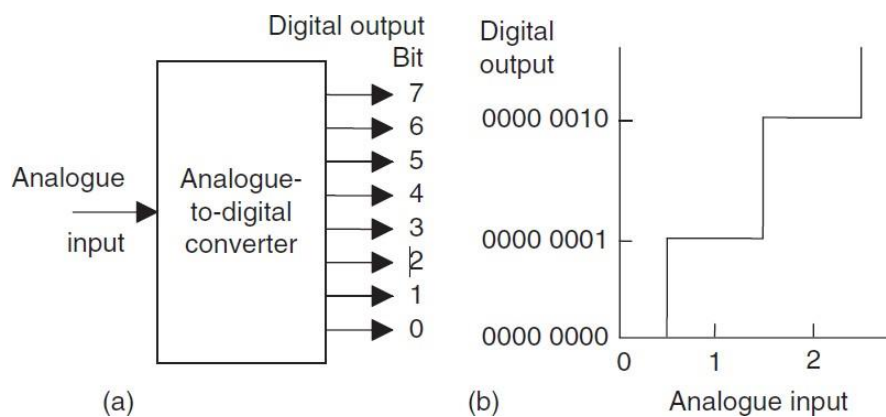


Figure 5: (a) Function of an analog-to-digital converter, and (b) an illustration of the relationship between the analog input and the digital output.

TABLE

Programmable Logic Controller

Analog Input (V)	Digital Output (V)
0.00	0000 0000
0.04	0000 0001
0.08	0000 0010
0.12	0000 0011
0.16	0000 0100
0.20	0000 0101
0.24	0000 0110
0.28	0000 0111
0.32	0000 1000
etc.	

Input/Output Devices:

The input devices considered include digital and analog devices such as mechanical switches for position detection, proximity switches, photoelectric switches, encoders, temperature and Pressure switches, potentiometers, linear variable differential transformers, strain gauges, thermistors, thermo transistors, and thermocouples. Output devices considered include relays, contactors, solenoid valves, and motors.

Input Devices:

The term sensor is used for an input device that provides a usable output in response to a specified physical input. For example, a thermocouple is a sensor that converts a temperature difference into an electrical output. The term transducer is generally used to refer to a device that converts a signal from one form to a different physical form. Thus sensors are often transducers, but also other devices can be transducers, such as a motor that converts an electrical input into rotation. Sensors that give digital or discrete, that is, on/off, outputs can be easily connected to the input ports of PLCs. An analog sensor gives an output proportional to the measured variable. Such analog signals have to be converted to digital signals before they can be input to PLC ports.

Mechanical Switches:

A mechanical switch generates an on/off signal or signals as a result of some mechanical input causing the switch to open or close. Such a switch might be used to indicate the presence of a work piece on a machining table, the work piece pressing against the switch and so closing it.



Mechanical switch

The absence of the work piece is indicated by the switch being open and its presence by it being closed. Thus, with the arrangement the input signals to a single input channel of the PLC are thus the logic levels:

Work piece not present: 0 ,Work piece present: 1

The 1 level might correspond to a 24 V DC input, the 0 to a 0 V input.when the switch is open the supply voltage is applied to the PLC input; when the switch is closed the input voltage drops to a low value.The logic levels are thus:

Workpiece not present: 1 ,Workpiece present: 0

Switches are available with normally open (NO) or normally closed (NC) contacts or can be configured as either by choice of the relevant contacts. An NO switch has its contacts open in the absence of a mechanical input and the mechanical input is used to close the switch. An NC switch has its contacts closed in the absence of a mechanical input and the mechanical input is used to open the switch. Mechanical switches are specified in terms of number of poles, that is, the number of separate circuits that can be completed by the same switching action, and number of throws, that is, the number of individual contacts for each pole.

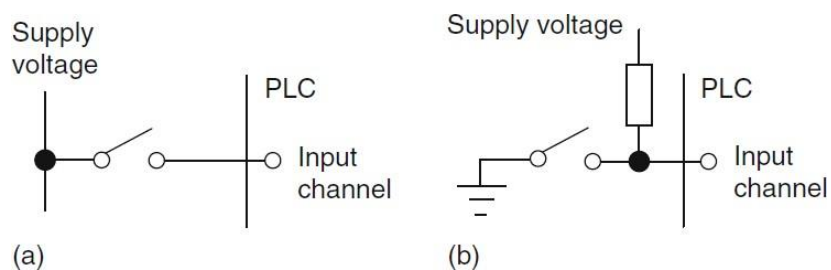


Figure 2.3: Switch sensors.

Programmable Logic Controller

Push Button: A push-button or simply button is a simple switch mechanism for controlling some aspects of machine or a process. A push button is a momentary or non-latching switch which causes a temporary change in the state of an electrical circuit only while the switch is physically actuated. An automatic mechanism (i.e. a spring) returns the switch to its default position immediately afterwards, restoring the initial circuit condition.



Push button

Buttons are typically made out of hard material, usually plastic or metal. The surface is usually flat or shaped to accommodate the human finger or hand, so as to be easily depressed or pushed. Buttons are most often biased switches, though even many un-biased buttons require a spring to return to their un-pushed state. Different people use different terms for the "pushing" of the button, such as press, depress, mash, hit, and punch.

Selector switch: A manually operated multi-position switch, which is usually adjusted by a knob or handle, and may have detents to hold in a given position. Used for instance, in devices or instruments with multiple functions, ranges, or modes of operation. Such a switch is usually rotary also called selector.



Selector switch

Programmable Logic Controller

Selector Switch works on a general principle, they contain a simple selector switch on the front of the panel, and a broad range of potential contact combinations (via the contact blocks), on the inside of the enclosure. The major difference between the selector switch and the pushbutton is that, while a pushbutton has a plate that pushes down both contact plungers at the same time, a selector switch has a rotating cam with ridges and flats, allowing to actuate the plunger independently. Selector switches are available in 2, 3, or 4-position versions, and are often used when more than one control option is needed. In general, the center position of the selector switch is the starting cam position. Left position presses the left plunger in the selector switch. Turning the selector switch to the right presses down the right plunger.

Contactor: A contactor is an electrically controlled switch used for switching an electrical power circuit, similar to a relay except with higher current ratings. A contactor is controlled by a circuit which has a much lower power level than the switched circuit.



Contactor

Contactors come in many forms with varying capacities and features. Unlike a circuit breaker control electric motors, lighting, heating, capacitor banks, a contactor is not intended to interrupt a short circuit current. Contactors range from those having a breaking current of several amperes to thousands of amperes and 24 V DC to many kilovolts. The physical size of contactors ranges from a device small enough to pick up with one hand, to large devices approximately a meter (yard) on a side. Contactors are used to, thermal evaporators, and other electrical loads.

Programmable Logic Controller

Illuminated push-button: A push-button or simply button is a simple switch mechanism for controlling some aspect of a machine or a process. Buttons are typically made out of hard material, usually plastic or metal. The surface is usually flat or shaped to accommodate the human finger or hand, so as to be easily depressed or pushed. Buttons are most often biased switches, though even many un-biased buttons (due to their physical nature) require a spring to return to their un-pushed state.



Illuminated push-button

Proximity Switches

Proximity switches are used to detect the presence of an item without making contact with it. There are a number of forms of such switches, some being suitable only for metallic objects.



Proximity switch

The eddy current type of proximity switch has a coil that is energized by a constant alternating current and produces a constant alternating magnetic field. When a metallic object is close to it, eddy currents are induced in it. The magnetic field due to these eddy currents induces an EMF back in the coil with the result that the voltage amplitude needed to maintain the constant coil current changes. The voltage amplitude is thus a measure of the proximity of metallic objects. The voltage can be used to activate an electronic switch circuit, basically a transistor that has its

Programmable Logic Controller

output switched from low to high by the voltage change, creating an on/off device. The range over which such objects can be detected is typically about 0.5 to 20 mm.

2.1.3 Photoelectric Sensors and Switches: A photoelectric sensor, or photo eye, is an equipment used to discover the distance, absence, or presence of an object by using a light transmitter, often infrared, and a photoelectric receiver. They are largely used in industrial manufacturing.

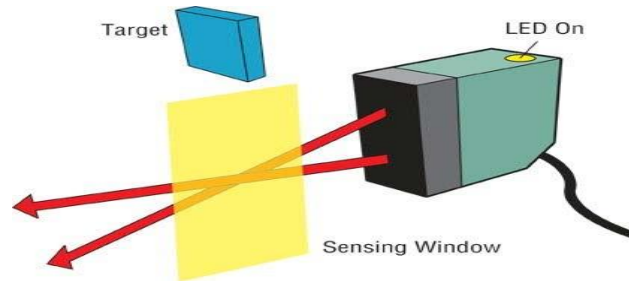


Photoelectric sensor

A retro-reflective arrangement places the transmitter and receiver at the same location and uses a reflector to bounce the light beam back from the transmitter to the receiver. An object is sensed when the beam is interrupted and fails to reach the receiver.

A proximity-sensing (diffused) arrangement is one in which the transmitted radiation must reflect off the object in order to reach the receiver. In this mode, an object is detected when the receiver sees the transmitted source rather than when it fails to see it. As in retro-reflective sensors, diffuse sensor emitters and receivers are located in the same housing. But the target acts as the reflector, so that detection of light is reflected off the disturbance object. The emitter sends out a beam of light (most often a pulsed infrared, visible red, or laser) that diffuses in all directions, filling a detection area. The target then enters the area and deflects part of the beam back to the receiver. Detection occurs and output is turned on or off when sufficient light falls on the receiver.

Programmable Logic Controller



Encoders

The term encoder is used for a device that provides a digital output as a result of angular or linear displacement.



Encoder

An incremental encoder detects changes in angular or linear displacement from some datum position; an absolute encoder gives the actual angular or linear position.

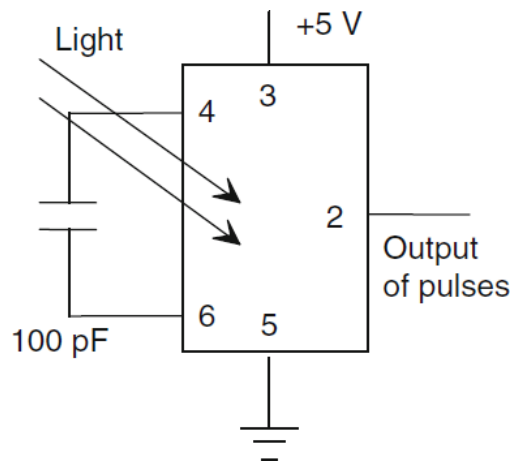


Figure : TSL220.

Temperature Sensors

A simple form of temperature sensor that can be used to provide an on/off signal when a particular temperature is reached is the bimetal element. This consists of two strips of



Temperature sensor

different metals, such as brass and iron, bonded together (Figure a). The two metals have different coefficients of expansion. Thus, when the temperature of the bimetal strip increases, the strip curves in order that one of the metals can expand more than the other. The higher expansion metal is on the outside of the curve. As the strip cools, the bending effect is reversed. This movement of the strip can be used to make or break electrical contacts and hence, at some particular temperature, give an on/off current in an electrical circuit. The device is not very accurate but is commonly used in domestic central heating thermostats because it is a very simple, robust device.

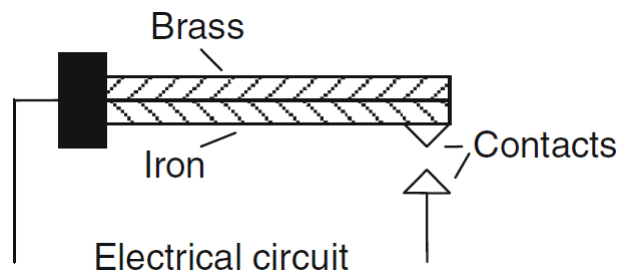


Figure a: Bimetallic strip.

Programmable Logic Controller

Position sensor: A position sensor is any device that permits position measurement. It can either be an absolute position sensor or a relative one (displacement sensor). Position sensors can be linear, angular, or multi-axis. Some position sensors available today: Capacitive transducer.



Position sensor

A. Limit switch: A limit switch is a switch operated by the motion of a machine part or presence of an object.



Limit switch

They are used for controlling machinery as part of a control system, as a safety interlocks, or to count objects passing a point. A limit switch is an electromechanical device that consists of an actuator mechanically linked to a set of contacts. When an object comes into contact with the actuator, the device operates the contacts to make or break an electrical connection.

Programmable Logic Controller

Limit switches are used in a variety of applications and environments because of their ruggedness, ease of installation, and reliability of operation. They can determine the presence or absence, passing, positioning, and end of travel of an object. They were first used to define the limit of travel of an object; hence the name "Limit Switch".

B. Thumb wheel switch: A thumbwheel switch is a multi-position rotary switch. It contains a sprocket that can go forward or backward. As you can imagine from the name, you will be able to use a thumb, or a finger, to move the sprocket each way. They can be on a mechanical or an electronic device. These are sometimes called digital switches, and you can see them in action on a variety of different devices. Some of them are very simple, while others are going to be quite a bit more complex.



Thumb wheel switch

Strain Gauges

When a wire or strip of semiconductor is stretched, its resistance changes. The fractional change in resistance is proportional to the fractional change in length, that is, strain.

$$\frac{\Delta R}{R} = G \times \text{strain}$$

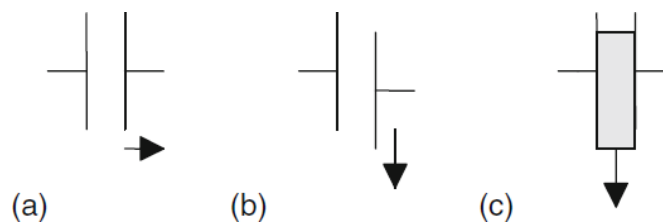


Figure 7: Capacitor sensors: (a) changing the plate separation, (b) changing the area of overlap, and (c) moving the dielectric.

Programmable Logic Controller

where ΔR is the change in resistance for a wire of resistance R and G is a constant called the gauge factor. For metals, the gauge factor is about 2; for semiconductors, about 100. Metal resistance strain gauges are in the form of a flat coil so that they get a reasonable length of metal in a small area. Often they are etched from metal foil and attached to a backing of thin plastic film so that they can be stuck on surfaces, like postage stamps on an envelope. The change in resistance of the strain gauge, when subject to strain, is usually converted into a voltage signal by the use of a Wheatstone bridge. A problem that occurs is that the resistance of the strain gauge also changes with temperature, and thus some means of temperature compensation has to be used so that the output of the bridge is only a function of the strain. This can be achieved by placing a dummy strain gauge in an opposite arm of the bridge, that gauge not being subject to any strain but only the temperature. A popular alternative is to use four active gauges as the arms of the bridge and arrange them so that one pair of opposite gauges is in tension and the other pair in compression. This not only gives temperature compensation; it also gives a much larger output change when strain is applied. The following paragraph illustrates systems employing such a form of compensation.

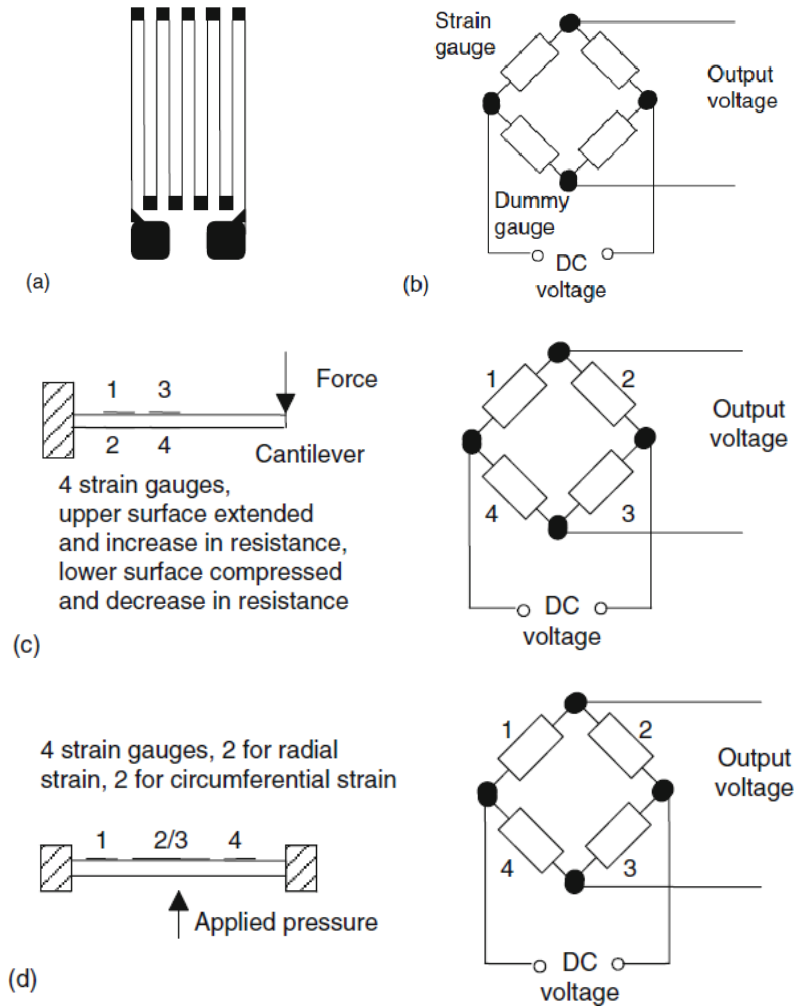


Figure (a) Metal foil strain gauge, (b) a Wheatstone bridge circuit with compensation for temperature changes, (c) strain gauges used for a force sensor, and (d) a pressure sensor.

By attaching strain gauges to other devices, changes that result in strain of those devices can be transformed, by the strain gauges, to give voltage changes. They might, for example, be attached to a cantilever to which forces are applied at its free end (Figure c).

The voltage change, resulting from the strain gauges and the Wheatstone bridge, then becomes a measure of the force. Another possibility is to attach strain gauges to a diaphragm, which deforms as a result of pressure (Figure d). The output from the gauges and associated Wheatstone bridge then becomes a measure of the pressure.

Pressure Sensors

Pressure sensors can be designed to give outputs that are proportional to the difference in pressure between two input ports. If one of the ports is left open to the atmosphere, the gauge measures pressure changes with respect to the atmosphere and the pressure measured is known as gauge pressure.



Pressure sensor

The pressure is termed the absolute pressure if it is measured with respect to a vacuum. Commonly used pressure sensors that give responses related to the pressure are diaphragm and bellows types. The diaphragm type consists of a thin disc of metal or plastic, secured around its edges. When there is a pressure difference between the two sides of the diaphragm, its center deflects. The amount of deflection is related to the pressure difference. This deflection may be detected by strain gauges attached to the diaphragm (see Figure 2 d), by a change in capacitance between it and a parallel fixed plate, or by using the deflection to squeeze a piezoelectric crystal (Figure a). When a piezoelectric crystal is squeezed, there is a relative displacement of positive and negative charges within the crystal and the outer surfaces of the crystal become charged. Hence a potential difference appears across it. An example of such a sensor is the Motorola MPX100AP sensor (Figure b). This has a built-in vacuum on one side of the diaphragm and so the deflection of the diaphragm gives a measure of the absolute pressure applied to the side of the diaphragm.

Programmable Logic Controller

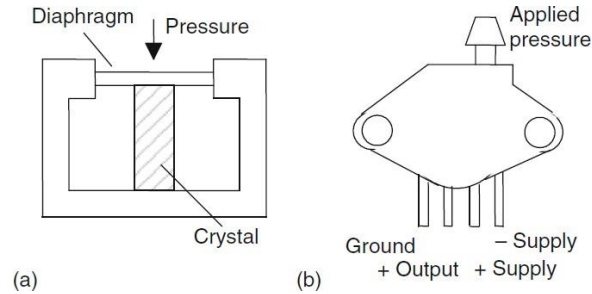


Figure : (a) A piezoelectric pressure sensor and (b) the MPX100AP.

The output is a voltage that is proportional to the applied pressure, with a sensitivity of 0.6 mV/kPa. Other versions are available that have one side of the diaphragm open to the atmosphere and so can be used to measure gauge pressure; others allow pressure to be applied to both sides of the diaphragm and so can be used to measure differential pressures.

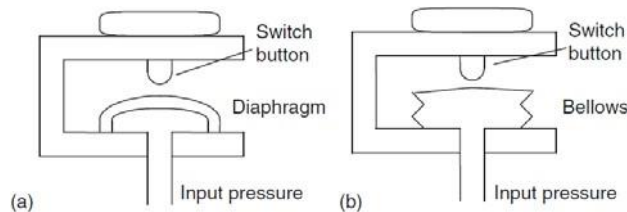


Figure e: Examples of pressure switches.

Pressure switches are designed to switch on or off at a particular pressure. A typical form involves a diaphragm or bellows that moves under the action of the pressure and operates a mechanical switch. Figure e, shows two possible forms. Diaphragms are less sensitive than bellows but can withstand greater pressures.

2.1.9 Liquid-Level Detectors

Pressure sensors may be used to monitor the depth of a liquid in a tank. The pressure due to a height of liquid h above some level is $h\rho g$, where ρ is the density of the liquid and g the acceleration due to gravity. Thus a commonly used method of determining the level of liquid in a tank is to measure the pressure due to the liquid above some datum level



Liquid level detector

Often a sensor is just required to give a signal when the level in some container reaches a particular level. A float switch that is used for this purpose consists of a float containing a magnet that moves in a housing with a reed switch. As the float rises or falls, it turns the reed switch on or off, the reed switch being connected in a circuit that then switches a voltage on or off.

2.1.10 Fluid Flow Measurement

A common form of fluid flow meter is one based on measuring the difference in pressure that results when a fluid flows through a constriction. Figure shows a commonly used form, the orifice flow meter. As a result of the fluid flowing through the orifice, the pressure at A is higher than that at B, the difference in pressure being a measure of the rate of flow.

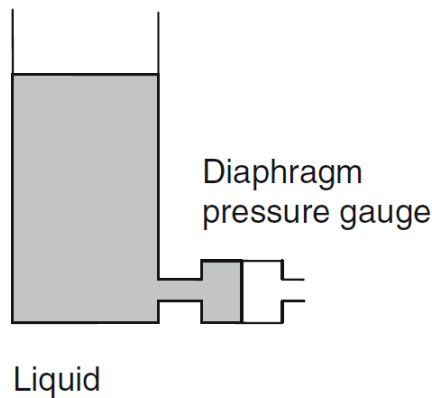
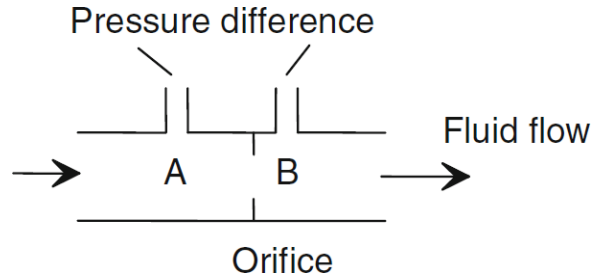


Figure 2.22: Liquid-level sensor.

Programmable Logic Controller



Orifice flow meter.

This pressure difference can be monitored by means of a diaphragm pressure gauge and thus becomes a measure of the rate of flow.

2.1.11 Smart Sensors

To use a sensor, we generally need to add signal conditioning circuitry, such as circuits which amplify and convert from analog to digital, to get the sensor signal in the right form, take account of any nonlinearities, and calibrate it. Additionally, we need to take account of drift, that is, a gradual change in the properties of a sensor over time. Some sensors have all these elements taken care of in a single package; they are called smart sensors.



Smart sensor

The term smart sensor is thus used in discussing a sensor that is integrated with the required buffering and conditioning circuitry in a single element and provides functions beyond that of just a sensor. The circuitry with the element usually consists of data converters, a processor and firmware, and some form of non volatile electrically erasable programmable read only memory (EEPROM, which is similar to EPROM;). The term non volatile is used because the memory has to retain certain parameters when the power supply is removed. Such smart sensors can have all their elements produced on a single silicon chip. Because the elements are processor-based devices, such a sensor can be programmed for specific requirements. For example, it can be

Programmable Logic Controller

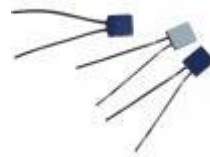
programmed to process the raw input data, correcting for such things as nonlinearities, and then send the processed data to a base station. It can be programmed to send a warning signal when the measured parameter reaches some critical value.

Transducer: A transducer is a device that converts one form of energy to another. Usually a transducer converts a signal in one form of energy to a signal in another.

Transducers are often employed at the boundaries of automation, measurement, and control systems, where electrical signals are converted to and from other physical quantities (energy, force, torque, light, motion, position, etc.). The process of converting one form of energy to another is known as transduction. Different types of transducer are shown below

The Platinum R.T.D. (resistance temperature dependent):

Transducer: Connections Gold contact plates laser trimmed platinum film Ceramic substrate The construction of the platinum R.T.D. transducer, consisting basically of a thin film of platinum deposited on a ceramic substrate and having gold contact plates at each end that make contact with the film. The resistance of the film increases as the temperature increases, i.e. it has a positive temperature coefficient. The increase in resistance is linear, the relationship between resistance change and temperature rise being $0.385\Omega/^{\circ}\text{C}$ for the unit.



Platinum RTD Temperature sensor

$$R_t = R_0 + 0.385t$$

where R_t = Resistance at temperature $t^{\circ}\text{C}$ and R_0 = Resistance at $0^{\circ}\text{C} = 100\Omega$

Normally, the unit would be connected to a D.C. supply via a series resistor and the voltage drop across the transducer is measured. The current flow through the transducer will then cause some self heating, the temperature rise due to this being of the order of $0.2^{\circ}\text{C}/\text{mW}$ dissipated in the transducer.

2.2 Output Devices

The output ports of a PLC are relay or opto-isolator with transistor or triac, depending on the devices that are to be switched on or off. Generally, the digital signal from an output channel of a PLC is used to control an actuator, which in turn controls some process. The term actuator is used for the device that transforms the electrical signal into some more powerful action, which then results in control of the process. The following are some examples.

Relay: A relay is an electrically operated switch. Many relays use an electromagnet to mechanically operate a switch, but other operating principles are also used, such as solid-state relays. Relays are used where it is necessary to control a circuit by a low-power signal (with complete electrical isolation between control and controlled circuits), or where several circuits must be controlled by one signal.



Relay

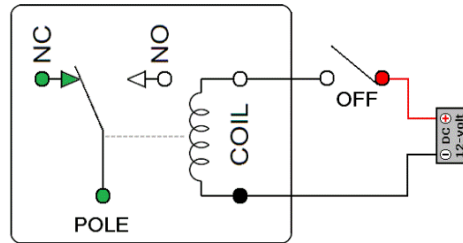
When a current passes through a solenoid, a magnetic field is produced; this can then attract ferrous metal components in its vicinity. With the relay, this attraction is used to operate a switch. Relays can thus be used to control a larger current or voltage and, additionally, to isolate the power used to initiate the switching action from that of the controlled power. For a relay connected to the output of a PLC, when the output switches on, the solenoid magnetic field is produced, and this pulls on the contacts and so closes a switch or switches. The result is that much larger currents can be switched on. Thus the relay might be used to switch on the current to a motor. The solenoid of a relay might be used to operate more than one set of contacts, the term pole being used for each set of contacts. Contacts can also be obtained as, in the absence of any input, either normally open (NO) or normally closed (NC). Thus, when selecting relays for a particular application, consideration has to be given to the number of poles required, the initial contact conditions, and the rated voltage and current. The term latching relay is used for a relay

Programmable Logic Controller

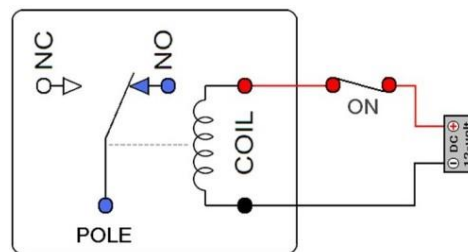
whose contacts remain open or closed even after the power has been removed from the solenoid.

The term contactor is used when large currents are being switched from large voltage sources.

Relay Off:



Relay On:



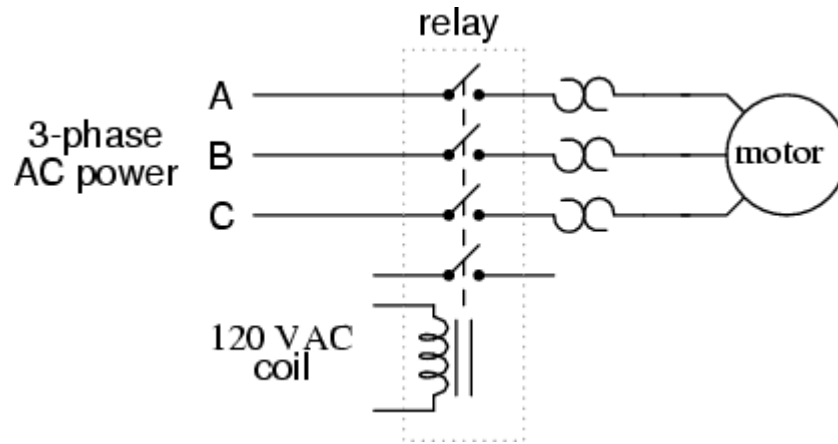
A. Contactor relays

Contactors are often used in control and regulating functions. They are used in large quantities for the indirect control of motors, valves, clutches and heating equipment. In addition to the simplicity which they offer in project engineering, panel building, commissioning and maintenance, the high level of safety which they afford is a major factor in their favor.



Contactors

Contactor connection



Contactor connection

Directional Control Valve :These valves are one of the most fundamental parts in hydraulic machinery as well as pneumatic machinery. They allow fluid flow into different paths from one or more sources. They usually consist of a spool inside a cylinder which is mechanically or electrically controlled. The movement of the spool restricts or permits the flow, thus it controls the fluid flow.



Directional control valve

Motors:

A DC motor has coils of wire mounted in slots on a cylinder of ferromagnetic material, which is termed the armature. The armature is mounted on bearings and is free to rotate. It is mounted in the magnetic field produced by permanent magnets or current passing through coils of wire, which are called the field coils. When a current passes through the armature coil, forces act on the coil and result in rotation. Brushes and a commutator are used to reverse the current through

Programmable Logic Controller

the coil every half rotation and so keep the coil rotating. The speed of rotation can be changed by changing the size of the current to the armature coil.

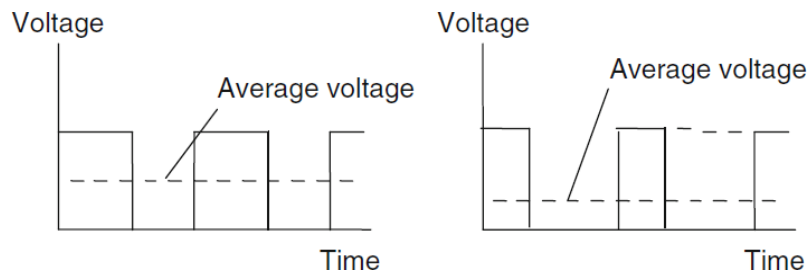


DC Motor

However, because fixed voltage supplies are generally used as the input to the coils, the required variable current is often obtained by an electronic circuit. This can control the average value of the voltage, and hence current, by varying the time for which the constant DC voltage is switched on. The term pulse width modulation (PWM) is used because the width of the voltage pulses is used to control the average DC voltage applied to the armature. A PLC might thus control the speed of rotation of a motor by controlling the electronic circuit used to control the width of the voltage pulses.

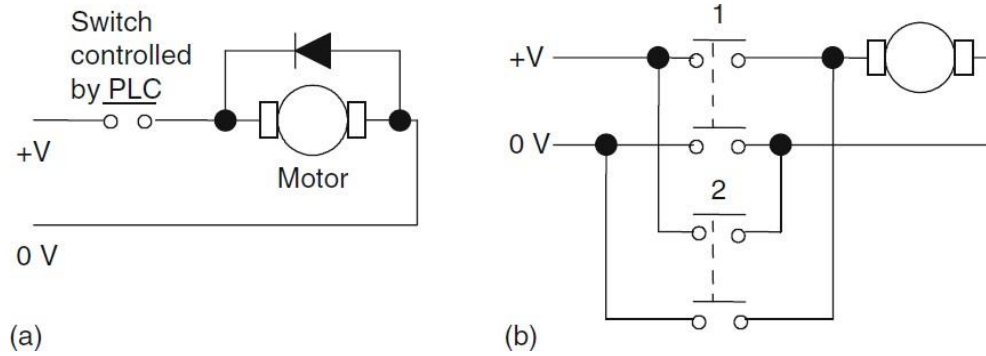
Many industrial processes only require the PLC to switch a DC motor on or off. This might be done using a relay. Fig a shows the basic principle. The diode is included to dissipate the induced current resulting from the back EMF.

Sometimes a PLC is required to reverse the direction of rotation of the motor. This can be done using relays to reverse the direction of the current applied to the armature coil.



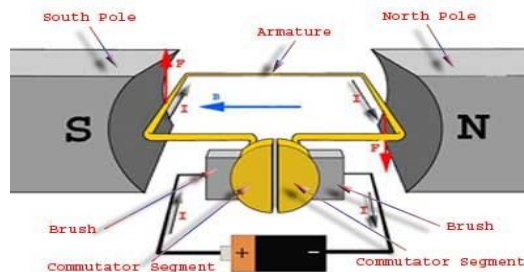
Pulse width modulation.

Programmable Logic Controller



DC motor: (a) on/off control, and (b) directional control.

Figure 2.33b shows the basic principle. For rotation in one direction, switch 1 is closed and switch 2 opened. For rotation in the other direction, switch 1 is opened and switch 2 another form of DC motor is the brushless DC motor.



Working principle of DC motor

With this conventional arrangement DC motor, a commutator has to be used to reverse the current through the coil every half rotation to keep the coil rotating in the same direction. With the brushless permanent magnet motor, electronic circuitry is used to reverse the current. The motor can be started and stopped by controlling the current to the stationary coil. Reversing the motor is more difficult, as reversing the current is not so easy, due to the electronic circuitry used for the commutator function. One method that is used is to incorporate sensors with the motor to detect the position of the north and south poles. These sensors can then cause the current to the coils to be switched at just the right moment to reverse the forces applied to the magnet. The speed of rotation can be controlled using pulse width modulation, that is, controlling the average value of pulses of a constant DC voltage. Though AC motors are cheaper, more rugged, and more reliable than DC motors, maintaining constant speed and controlling that speed is generally more complex than with DC motors. As a consequence, DC motors, particularly brushless permanent magnet motors, tend to be more widely used for control purposes.

Stepper Motors

The stepper or stepping motor is a motor that produces rotation through equal angles, the so-called steps, for each digital pulse supplied to its input. Thus, if one input pulse produces a rotation of 1.8° , then 20 such pulses would give a rotation of 36.0° . To obtain one complete revolution through 360° , 200 digital pulses would be required. The motor can thus be used for accurate angular positioning. If a stepping motor is used to drive a continuous belt, it can be used to give accurate linear positioning. Such a motor is used with computer printers, robots, machine tools, and a wide range of instruments for which accurate positioning is required.

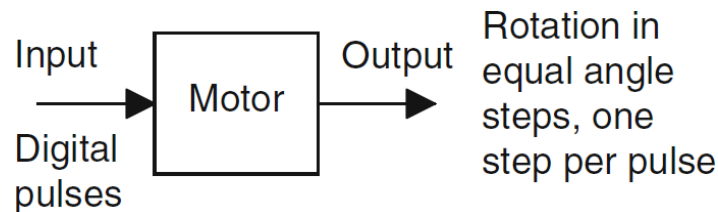


Figure : The stepping motor..

Step	Pole 1	Pole 2	Pole 3	Pole 4
1	North	South	South	North
2	South	North	South	North
3	South	North	North	South
4	North	South	North	South
5	Repeat of steps 1 to 4			

Thus in this case there are four possible rotor positions: 0° , 90° , 180° , and 270° .

Figure shows the basic principle of the variable reluctance type. The rotor is made of soft steel and has a number of teeth, the number being less than the number of poles on the stator. The stator has pairs of poles, each pair of which is activated and made into an electromagnet by a current being passed through the coils wrapped round it. When one pair of poles is activated, a magnetic field is produced that attracts the nearest pair of rotor teeth so that the teeth and poles line up. This is termed the position of minimum reluctance. By then switching the current to the next pair of poles, the rotor can be made to rotate to line up with those poles. Thus by

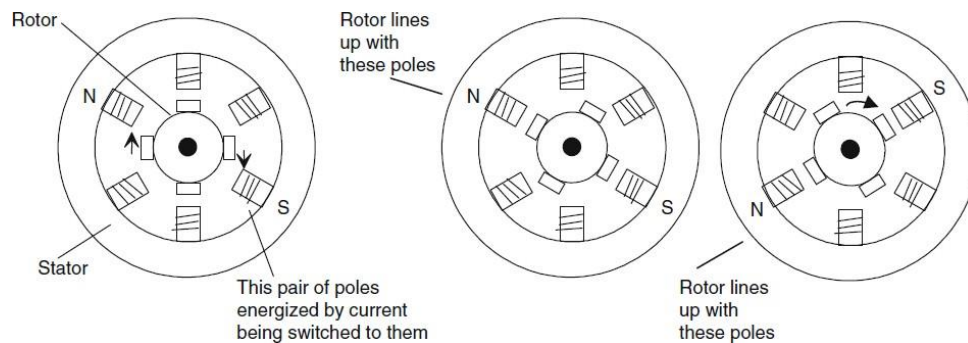
Programmable Logic Controller

sequentially switching the current from one pair of poles to the next, the rotor can be made to rotate in steps.

There is another version of the stepper motor—the hybrid stepper. This version combines features of both the permanent magnet and variable reluctance motors. Hybrid steppers have a permanent magnet rotor encased in iron caps that are cut to have teeth. The rotor sets itself in the minimum reluctance position when a pair of stator coils are energized.

The following are some of the terms commonly used in specifying stepper motors:

- **Phase.** This term refers to the number of independent windings on the stator. Two-phase motors tend to be used in light-duty applications, three-phase motors tend to be variable reluctance steppers, and four-phase motors tend to be used for higher-power applications.



The principle of the variable reluctance stepper motor.

AC Motor: An AC motor is an electric motor driven by an alternating current (AC). The AC motor commonly consists of two basic parts, an outside stationary stator having coils supplied with alternating current to produce a rotating magnetic field, and an inside rotor attached to the output shaft producing a second rotating magnetic field. The rotor magnetic field may be produced by permanent magnets, reluctance saliency, or DC or AC electrical winding.



AC Motor

Programmable Logic Controller

When an AC motor is in steady-state rotation (motion), the magnetic fields of the rotor and stator rotate (move) with little or no slippage (near synchrony). The magnetic forces (repulsive and attractive) between the rotor and stator poles create average torque, capable of driving a load at rated speed. The speed of the stator rotating magnetic field (ω_s) and the speed of the rotor rotating magnetic field (ω_r), relative to the speed of the mechanical shaft (ω_m), must maintain synchronism for average torque production by satisfying the synchronous speed relation (i.e., $\pm\omega_s \pm \omega_r = \omega_m$). Otherwise, asynchronously rotating magnetic fields would produce pulsating or non-average torque.

The two main types of AC motors are classified as induction and synchronous. The induction motor (or asynchronous motor) always relies on a small difference in speed between the stator rotating magnetic field and the rotor shaft speed called slip to induce rotor current in the rotor AC winding. As a result, the induction motor cannot produce torque near synchronous speed where induction (or slip) is irrelevant or ceases to exist. In contrast, the synchronous motor does not rely on slip-induction for operation and uses either permanent magnets, salient poles, or an independently excited rotor winding. The brushless wound-rotor doubly fed synchronous motor system has an independently excited rotor winding that does not rely on the principles of slip-induction of current. The brushless wound-rotor doubly fed motor is a synchronous motor that can function exactly at the supply frequency or sub to super multiple of the supply frequency.

Lightening element:

Bulbs: An incandescent light bulb, incandescent lamp or incandescent light globe is an electric light with a wire filament heated to a high temperature, by passing an electric current through it, until it glows with visible light (incandescence). The hot filament is protected from oxidation with a glass or quartz bulb that is filled with inert gas or evacuated. In a halogen lamp, filament evaporation is prevented by a chemical process that re deposits metal vapour onto the filament, extending its life. The light bulb is supplied with electric current by feed-through terminals or wires embedded in the glass. Most bulbs are used in a socket which provides mechanical support and electrical connections.



Bulb

LED: An LED lamp is a light-emitting diode (LED) product which is assembled into a lamp (or light bulb) for use in lighting fixtures. LED lamps have a lifespan and electrical efficiency which are several times longer than incandescent lamps, and significantly more efficient than most fluorescent lamps, with some chips able to emit more than 300 lumens per watt.



LED

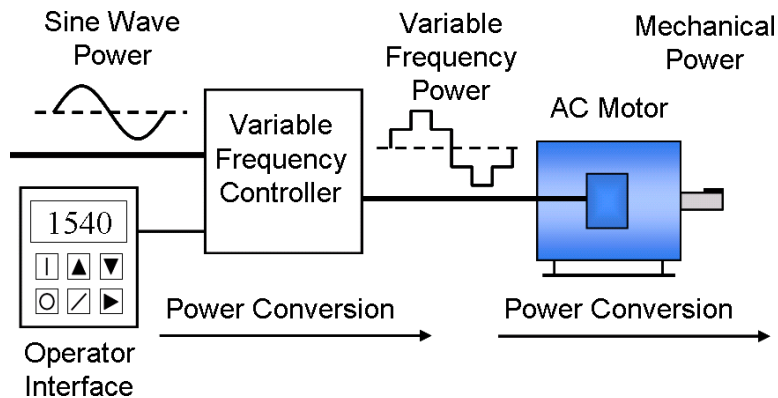
Variable Frequency Drive: A variable-frequency drive (VFD) (also termed adjustable-frequency drive, variable-speed drive, AC drive, micro drive or inverter drive) is a type of adjustable-speed drive used in electro-mechanical drive systems to control AC motor speed and torque by varying motor input frequency and voltage.

Programmable Logic Controller



Variable frequency drive

Ac motor control: The AC electric motor used in a VFD system is usually three-phase induction motor. Some types of single-phase motors can be used, but three-phase motors are usually preferred. Various types of synchronous motors offer advantages in some situations, but three-phase induction motors are suitable for most purposes and are generally the most economical motor choice. Motors that are designed for fixed-speed operation are often used. Elevated-voltage stresses imposed on induction motors that are supplied by VFDs require that such motors be designed for definite-purpose inverter-fed duty in accordance with such requirements as Part 31 of NEMA Standard MG-1.



VFD System

3. PROGRAMMING OF MILLENIUM PLC

PROGRAMMING

Programs for use with PLCs can be written in a number of formats. To make it easier for engineers with no great knowledge of programming to write programs for PLCs, ladder programming was developed. Most PLC manufacturers adopted this method of writing programs; however, each tended to develop its own versions and so an international standard has been adopted for ladder programming and indeed all the methods used for programming PLCs. The standard, published in 1993, is International Electrotechnical Commission (IEC) 1131-3, sometimes referred to as IEC 61131-3.

The IEC Standard

Major Features of IEC 1131-3

The following are some of the major features of the standard.

- 1. Multiple Language Support:** One of the main features of the standard is that it allows multiple languages to be used simultaneously, thus enabling the program developer to use the language best suited to each control task.
- 2. Code Reusability:** The control algorithm can include reusable entities referred to as "program organization units (POUs)" which include Functions, Function Blocks, and Programs. These POUs are reusable within a program and can be stored in user-declared libraries for import into other control programs.
- 3. Library Support:** The IEC-1131 Standard includes a library of pre-programmed functions and function blocks. An IEC compliant controller supports these as a "firmware" library, that is, the library is pre-coded in executable form into a prom or flash ram on the device. Additionally, manufacturers can supply libraries of their own functions. Users can also develop their own libraries, which can include calls to the IEC standard library and any applicable manufacturers' libraries.
- 4. Execution Models:** The general construct of a control algorithm includes the use of "tasks", each of which can have one or more Program POUs. A task is an independently schedulable software entity and can be assigned a cyclic rate of execution, can be event driven, or be triggered by specific system functions, such as startup.

Programmable Logic Controller

The full IEC 61131 standards covers the complete life cycle of PLCs:

Part 1: General definition of basic terminology and concepts.

Part 2: Electronic and mechanical equipment requirements and verification tests for PLCs and associated equipment.

Part 3: Programming languages. Five languages are defined: Ladder Diagram (LAD), Sequential Function Charts (SFC), Function Block Diagram (FBD), Structured Text (ST) and Instruction List (IL).

Part 4: Guidance on selection, installation, and maintenance of PLCs.

Part 5: Software facilities needed for communication with other devices based on the Manufacturing Messaging Specification (MMS).

Part 6: Communications via field bus software facilities.

Part 7: Fuzzy control programming.

Part 8: Guidelines for the implementation of PLC programming languages.

Programming PLCs

External programming units can be used to download programs into the program memory of the CPU. The external field programmers provide several software features that facilitate program entry in graphical form. The programmers also provide comprehensive aids for debugging and execution monitoring support logic and sequence control systems. Printer can be connected to the programmers for the purpose of documenting the program. In some cases, special programming packages that run on Personal Computer can also be used as programming units.

There are two ways of entering the program:

1. Direct program entry to the program memory (RAM) plugged into the central controller. For this purpose, the programmer is connected to the processor or to the programmer interface modules.

2. Programming the EPROM sub modules in the programmer without connecting it to the PC (off-line). The memory sub modules are then plugged into the central controller.

OA programming device can be a handheld device, a desktop console, or a computer. Only when the program has been designed on the programming device and is ready is it transferred to the memory unit of the PLC.

- A handheld programming device normally contains enough memory to allow the unit to retain programs while being carried from one place to another.

Programmable Logic Controller

- Desktop consoles are likely to have a visual display unit with a full keyboard and screen display.

Personal computers are widely configured as program development workstations. Some PLCs only require the computer to have appropriate software; others require special communication cards to interface with the PLC. A major advantage of using a computer is that the program can be stored on the hard disk or a CD and copies can be easily made.

PLC manufacturers have programming software for their PLCs. For example, Mitsubishi has MELSOFT. The company's GX Developer supports all MELSEC controllers, from the compact PLCs of the MELSEC FX series to the modular PLCs, including the MELSEC System Q, and uses a Windows-based environment. It supports the programming methods of IL, LD, and SFC languages. You can switch back and forth between IL and LD at will while you are working. You can program your own function blocks, and a wide range of utilities is available for configuring special function modules for the MELSEC System Q; there is no need to program special function modules, you just configure them. The package includes powerful editors and diagnostics functions for configuring MELSEC networks and hardware, and extensive testing and monitoring functions to help get applications up and running quickly and efficiently. It offers offline simulation for all PLC types and thus enables simulation of all devices and application responses for realistic testing.

As another illustration, Siemens has SIMATIC STEP 7. This fully complies with the international standard IEC 61131-3 for PLC programming languages. With STEP 7, programmer can select from among various programming languages. Besides LAD and FBD, STEP 7 Basis also includes the IL programming language. Other additional options are available for IEC 61131-3 programming languages such as ST, called SIMATIC S7-SCL, or SFC, called SIMATIC S7-Graph, which provides an efficient way to describe sequential control systems graphically. Features of the whole engineering system include system diagnostic capabilities, process diagnostic tools, PLC simulation, remote maintenance, and plant documentation. S7-PLCSIM is an optional package for STEP 7 that allows simulation of a SIMATIC S7 control platform and testing of a user program on a PC, enabling testing and refining prior to physical hardware installation. By testing early in a project's development, overall project quality can be improved. Installation and commissioning can thus be quicker and less expensive because program faults can be detected and corrected early on during development. Likewise, Rockwell

Programmable Logic Controller

Automation manufactures RSLogix for the Allen-Bradley PLC-5 family of PLCs, OMRON has CX-One, and Telemecanique has ProWorx 32 for its Modicon range of PLCs.

Similarly a user friendly application oriented PLC, CROUZET MILLENIUM 3 is programmed using the **CLS M3** software.

CROUZET MILLENIUM 3

The Millennium 3 logic controller can be used to automate small devices requiring between 10 and 50 I/O. Millennium 3's logic functions can be used in numerous applications, including packing, access control, vending, irrigation, pump management and heating and air conditioning system control. Millennium 3 is available in a compact version for simple control systems or an expandable version for enhanced performance. For quick, simple programming, the Millennium 3 software prioritizes dedicated application-functions such as pump switching, PID control, movement, pressure, level and flow. The MILLENIUM 3 is programmed using the **CLS M3** software workshop. All the basic functions, like counting, timing, comparison and display, are also available. The M3 SOFT programming software incorporates fool proofing, so that when the slightest data entry error is made, it flags the incorrect item in red. The M3 SOFT software is multilingual, offering English, French, Italian, German and Spanish. 3rd generation of logic controllers at the core of your industry. With the new Millennium 3, you can take advantage of all the most recent developments in the latest generation of logic controllers. An innovative product, developed, industrialised and marketed by Crouzet, Millennium 3 is the successful synthesis of our expertise in automation systems acquired over a period of more than 30 years. Crouzet develops automation components and products, both standard and customized, meeting the needs expressed by its customers in the fields of machine manufacture, system integration and equipment manufacture. Throughout the world, Crouzet provides its customers with technical and industrial expertise to ensure seamless integration, regardless of the target device or application. Whether you are an electrical engineer or a control systems engineer, you can select the programming language you prefer. With Ladder or FBD/Grafcet language, everything is intuitive, quick and safe. Millennium 3 is capable of reading and converting programs created on the Millennium 2 logic controller.

The important features are

1. Programming: You can choose between two different languages: Ladder and FBD/ Grafcet.
2. Simulation: You can test the result of your programming in real time.
3. Downloading: You can transfer your programs directly to the modules or remotely via local wired or wireless (Bluetooth) modem solutions.
4. Supervision: You can view the status of your application, locally or remotely, by the communication solutions.

Programmable Logic Controller

Selection of PLC: General characteristics

1. Millenium 3 Compact Range



CD12

CD20

CB12

CB20

2. Millenium 3 Expandable Range



XD10

XD26

XB10

XB26

3. Millenium 3 Communication Options



STN

GSM

M3MOD

XN03
4-word
Modbus
extension

XN05
Ethernet
extension

XN06
8-word
Modbus
extension

Programmable Logic Controller

Extensions Crouzet Millenium PLC: Digital and analog extensions.



APPLICATION

Major areas of application are as follows

1. Building Management Systems
 - a) Lighting control systems
 - b) Air conditioning and heating systems
 - c) Lifts, hoists and escalators
 - d) Automatic doors and barriers
2. Industry
 - a) Packing machines
 - b) Woodworking machines
 - c) Conveyors
 - d) Moulding machines
3. Commercial equipment
 - a) Automatic washing equipment
 - b) Vending machines
 - c) Advertising hoardings
 - d) Toll barriers
4. Water treatment/Agriculture
 - a) Farm machinery
 - b) Irrigation/sprinkler systems
 - c) Pump management
5. Renewable energies

Programmable Logic Controller

- a) Solar panels
- b) Wind turbines
- c) Heat pumps

Some of the applications are as follows:

1. Fountain: Pump control with variable flow for different water jet and mood effects, regulation of water neutrality (pH) and disinfecting of water in fountain (ORP).
2. Field irrigation: Irrigation control based on temperature, humidity, and day/night cycle.
3. Hydraulic solar heating: Automation of operation and heating regulation plus remote management of the installation via GSM modem.
4. Opening control for industrial sectional doors: Opening control for doors and associated security devices for restricting access. Synchronisation between the various doors.
5. Automatic barrier Opening control of barriers with automatic vehicle detection and function for selecting opening times/days.
6. Public lighting: Control of public lighting to coincide with sunrise/sunset in order to save energy whilst ensuring optimum security levels.
7. Stretch wrapping machine: Controls the motor that unrolls the packing film, controls cutting of the film after heat sealing, and determines the duration of the motor cycles.
8. Automation of industrial on-board grass-cutter: Control of machine automation and cutting
9. Unit Operating Conditions: control of reverse flow fan (filter cleaning function), control of solenoid valves for locking rear wheels in a straight line, control of cylinders/start function/lights/horn.

Programmable Logic Controller

XD 26 CROUZET MILLENIUM 3 PLC

The model numbering for the XD 26 PLC is given in such a way that as it is having sixteen Digital of which six are analog inputs and ten discrete static relay outputs as shown in Fig.



XD 26 PLC CROUZET MILLENIUM PLC

XD-26 Crouzet Millennium PLC works on the software of MILLENIUM 3 Programming Language. The new Millennium 3, gives us an opportunity to take advantage of all the most recent developments in the latest generation of logic controllers. For quick, simple programming, the Millennium 3 software prioritizes dedicated application-specific functions such as pump switching, PID control, movement, pressure, level and flow. All the basic functions, such as counting, timing, comparison and display, are also available. The M3 SOFT programming software incorporates error checking, so that when the slightest data entry error is made, it flags the incorrect item in red. The numbering for the XD 26 PLC is given in such a way that as it is containing 16 Digital (of which 6 are analog) inputs and 10 discrete static relay outputs. And the total number of these inputs and outputs account to 26.

PROGRAMMING

The IEC 61131 defined the standards for PLCs, with 61131-3 defining the programming languages: ladder diagrams (LAD), instruction list (IL), sequential function charts (SFC), structured text (ST), and function block diagrams (FBD).

PLC's programming is based on the logic demands of input devices and the programs implemented are predominantly logical rather than numerical computational algorithms. Most of

Programmable Logic Controller

the programmed operations work on a straightforward two-state on or off basis and these alternate possibilities correspond to true or false (logical form) and 1 or 0 (binary form) respectively. Thus, PLCs offer a flexible programmable alternative to electrical circuit relay-based control systems built using analog devices.

Every PLC has associated programming software that allows the user to enter a program into the PLC. Before a PLC can perform any control task, it must be programmed to do so.

The Software used for the PLC is Crouzet Millenium 3 Programming Language. The controller offers two programming languages among the following

1. Ladder Programming (LD)
2. Function Block Diagram (FBD)
3. Sequential Flow Chart (SFC)
4. Structured Text
5. Higher level languages such as C.

The first two of the above mentioned languages are discussed below.

1. Ladder Programming (LD):

The most popular language used to program a PLC is ladder Programming. Ladder Programming (LD) is a graphic Language. It can be used to transcribe relay diagrams,

In the ladder programming method, the PLC system provides a design environment in the form of software tools running on a host computer terminal which allows ladder diagrams to be developed, verified, tested, and diagnosed. First, the high-level program is written in ladder diagrams. Then, the ladder diagram is converted into binary instruction codes so that they can be stored in random-access memory (RAM) or erasable programmable read-only memory (EPROM). Each successive instruction is decoded and executed by the CPU.

The function of the CPU is to control the operation of memory and I/O devices and to process data according to the program. Each input and output connection point on a PLC has an address used to identify the I/O bit. The method for the direct representation of data associated with the inputs, outputs, and memory is based on the fact that the PLC memory is organized into three regions: input image memory (I), output image memory (Q), and internal memory (M). The PLC program uses a cyclic scan in the main program loop such that periodic checks are made to the input variables. The program loop starts by scanning the inputs to the system and storing their

Programmable Logic Controller

states in fixed memory locations (input image memory I). The ladder program is then executed rung-by-rung. Scanning the program and solving the logic of the various ladder rungs determine the output states. The updated output states are stored in fixed memory locations (output image memory Q).

The output values held in memory are then used to set and reset the physical outputs of the PLC simultaneously at the end of the program scan.

2. Function Block Diagram Language (FBD):

FBD mode allows graphic programming based on the use of predefined function blocks. It offers a large range of basic functions: timer, counter, logic, etc

In the Function Block Diagram, the PLC system provides a design environment in the form of software tools running on a host computer terminal which allows Function Block Diagram to be developed, verified, tested, and diagnosed. First, the high-level program is written in Function Block Diagram. Then, the Function Block Diagram is converted into binary instruction codes so that they can be stored in random-access memory (RAM) or erasable programmable read-only memory (EPROM). Each successive instruction is decoded and executed by the CPU.

The function of the CPU is to control the operation of memory and I/O devices and to process data according to the program. Each input and output connection point on a PLC has an address used to identify the I/O bit. The method for the direct representation of data associated with the inputs, outputs, and memory is based on the fact that the PLC memory is organized into three regions: input image memory (I), output image memory (Q), and internal memory (M). The PLC program uses a cyclic scan in the main program loop such that periodic checks are made to the input variables.

The program loop starts by scanning the inputs to the system and storing their states in fixed memory locations (input image memory I). The Function Block Diagram is then executed. Scanning the program and solving the logic of the various blocks determine the output states. The updated output states are stored in fixed memory locations (output image memory Q). The output values held in memory are then used to set and reset the physical outputs of the PLC simultaneously at the end of the program scan.

For the given PLC, the time taken to complete one cycle or the scan time is 0.18 ms/K (for 1000 steps) and with a maximum program capacity of 1000 steps. The development system comprises a host computer (PC) connected via an RS232 port to the target PLC.

Programmable Logic Controller

The host computer provides the software environment to perform file editing, storage, printing, and program operation monitoring. The process of developing the program to run on the PLC consists of: using an editor to draw the source Function Block Diagram, converting the source program to binary object code which will run on the PLC's microprocessor and downloading the object code from the PC to the PLC system via the serial communication port. The PLC system is online (monitoring mode) when it is in active control of the machine and monitors any data to check for correct operation.

Operating Modes of the Controller

A PLC executes an initialization step when placed in run mode, then repeatedly executes a scan cycle sequence. The basic PLC scan cycle consists of three steps an input scan, a user program scan, an output scan. The total time for one complete program scan is a function of processor speed, I/O modules used, and length of user program. Typically, hundreds of complete scans can take place in 1 second.

There are several operating modes for the program to simulate and monitor in the workspace. The various operating modes are listed below:

1. Edit Mode: Edit mode is used to construct programs in FBD mode, which corresponds to the development of the application.

2. Simulation Mode: In simulation mode, the program is executed offline directly in the programming workshop .In this mode, each action on the chart (changing the state of an input, output forcing) updates the simulation windows.

3. Monitoring mode: In Monitoring mode, the program is executed on the controller, the programming workshop is connected to the controller (PC ↔ controller connection). The different windows are updated cyclically.

FEATURES

- 1. Macro function:** Integrating your repetitive functions into dedicated macro functions saves time and makes your life easier, as it enables you to reuse your expertise directly within your programs. You can access and modify the content of your macro functions, or choose to protect them with a password.
- 2. Division of the wiring sheet into several edit windows:** This kind of division makes it possible to display two different sections of the wiring sheet on the same screen. This makes it easier to carry out debugging and wiring for your program.
- 3. Easy moving of links:** The fact that you can move the links means you can develop your program by replacing function blocks but without losing your existing links.
- 4. Simulating program timing:** The “Next event” key enables the user to set the time of the time simulator to the start of next timed event that has been programmed.
- 5. Customizing your program with your own images:** The software enables you to import images into your program so you can customize your wiring sheet, your input/output icons and your macro functions.
- 6. Pressure transmitter:** The pressure transmitter measures the compressor’s supply and outlet pressures to control the motor according to the required displayed pressure, thereby ensuring maximum efficiency and avoid breakdowns of equipments.
- 7. Ready-to-use:** The pressure transmitter’s reference and specifications are preset in the Millenium 3 logic controller, allowing safe, speedy and effective installation, using dedicated function blocks.
- 8. The logic controller at the heart of your equipment:** The Millenium 3 logic controller has everything you need to control your process effectively. We can adapt function blocks and preset applications to gather data and process it, to co-ordinate operation of one or more output devices easily. A dedicated function ensures simultaneous management of devices, in order to extend their working life.

PLC Trainer kit:

PLC Trainer Kit is a setup of Crouzet Millenium PLC and power circuit interfacing unit consisting of various components. PLC trainer kit provides an advanced real time hardware implementation of various programmable switching schemes, logical and process control. On this PLC trainer kit we can also perform application oriented experiment by making use of in-built applications of FBD programming. This trainer kit facilitates to perform various experiments and project work for research and development of existing schemes for the students to wide spread their opportunities in core and industrial sector.

PLC Trainer Kit consists of following blocks:

1. SMPS (SWITCHED MODE POWER SUPPLY)
2. XD 26 Crouzet Millenium PLC
3. Relay Unit and Contactor
4. Manual Control Panel Block
5. Output Testing Unit
6. Protective Devices

Block Diagram of PLC Trainer Kit:

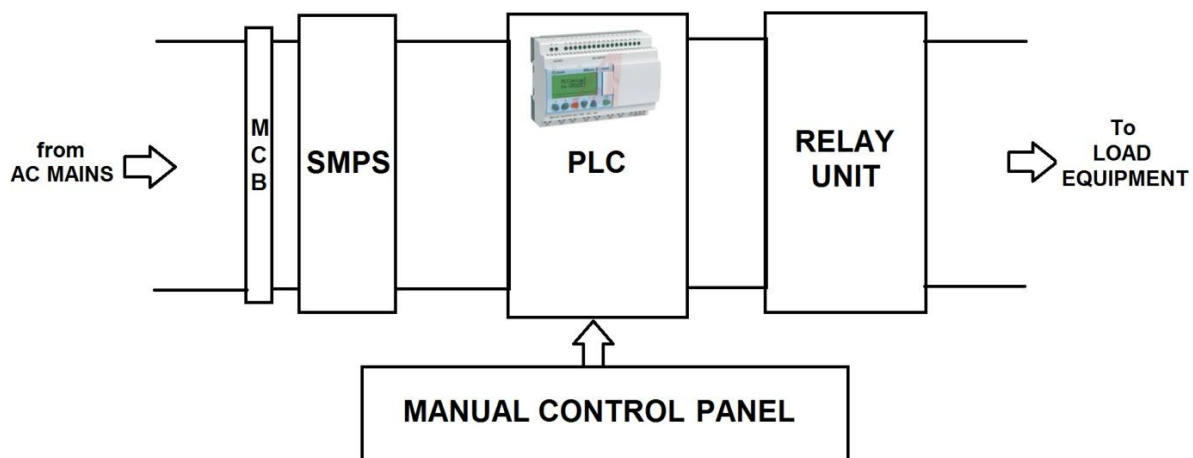
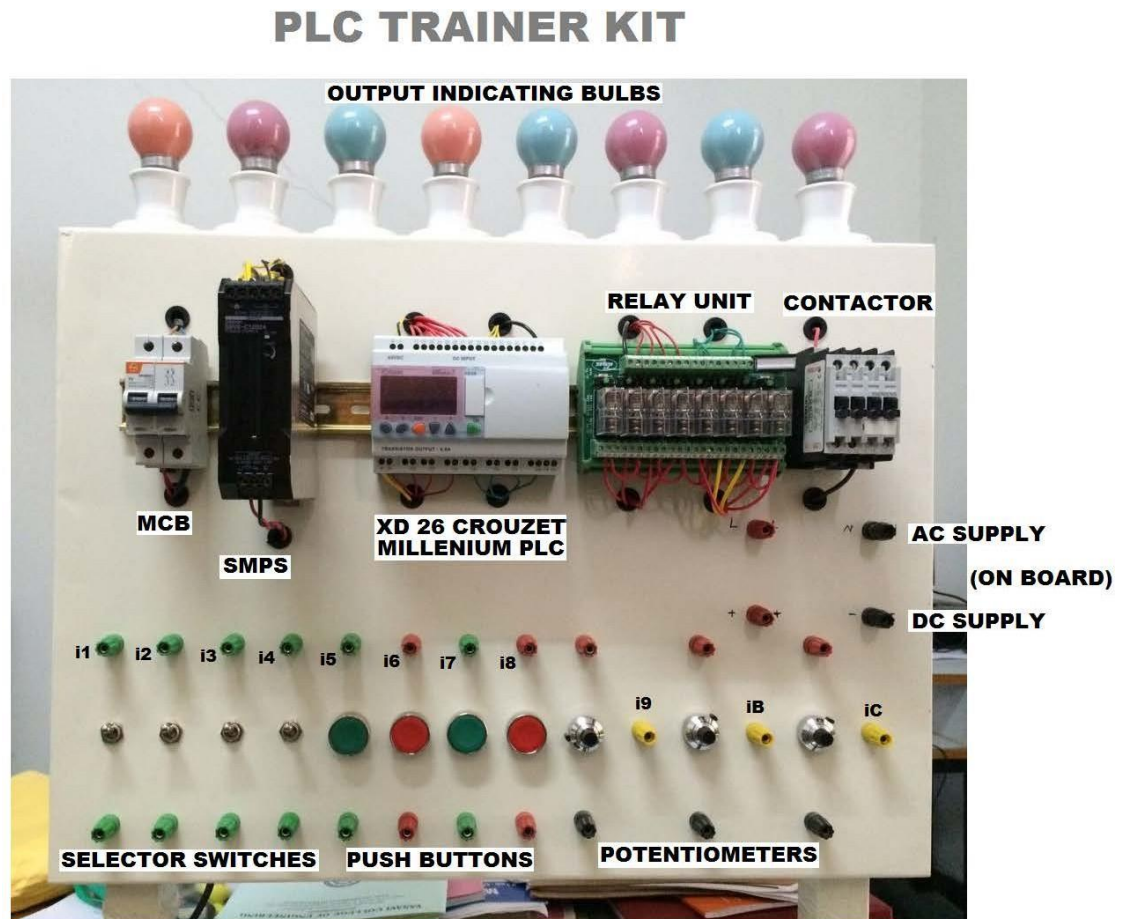


Fig Block diagram of PLC trainer kit

PLC Trainer Kit can be sectionalized for better understand of the kit to use it safely and logically as shown below:



PLC Trainer Kit blocks are discussed in briefly as follows:

1. SMPS (SWITCHED MODE POWER SUPPLY):

SMPS (Switched Mode Power Supply) is a power electronic device which converts variable input ac (230 V) to fixed dc (24 V as per requirement) to energize PLC.

Terminals:

Input: ac 230v, 1-phase (L+) and neutral (N-)

Output: dc 24v, 2 + terminals and 1 - terminal

Programmable Logic Controller

SMPS assembled in our trainer kit is as shown in figure below:



Fig SMPS

2. XD 26 Crouzet Millenium PLC:

XD 26 Crouzet Millenium PLC is a user friendly, application oriented and inbuilt functions powered PLC. The numbering for the XD 26 PLC implies it contains 16 Digital (10 discrete and 6 analog) inputs and 10 discrete static relay outputs. And the total number of these inputs and outputs account to 26. The Software used for the PLC is Crouzet Millenium 3 Programming Language. The controller offers two programming languages

1. Ladder Programming (LD)
2. Function Block Diagram (FBD)

Terminals:

Programmable Logic Controller

Input: dc 24v, 1 + terminals, 1 – terminal, I1-IG (16 input terminals)

Output: dc 24v, O1-OA (10 output terminals)

PLC powered and programmed in our trainer kit is as shown in figure below:



Fig XD 26 Crouzet Millenium PLC

3. Relay Unit and Contactor:

Relay unit:

It consist of a set of relays which are controlled and actuated by PLC, upon an actuating signal generated by the programmer through PLC programming to control load equipment. A **relay** is an electrically operated switch. Many relays use an electromagnet to mechanically operate a switch.

Terminals:

Input: dc 24v, 1 + terminals, 1 – terminal, I1-IG (16 input terminals)

Output: dc 24v, O1-OA (10 output terminals)

Relay Unit assembled in our trainer kit is as shown in figure below:

Programmable Logic Controller



Fig Relay Unit.

Contactor:

A type of relay that can handle the high power required to directly control an electric motor or other loads is called a contactor. Contactor gets activated by PLC through relay to drive heavy load equipments.

Terminals:

Input: Control terminals are ac 230v, A1 and A2

Power terminals are ac 440 v, L1, L2, L3 and NC

Output: Power terminals are ac 440v, T1, T2, T3 and NC

Contactor assembled in our trainer kit is as shown in figure below:



Fig Contactor.

4. Manual Control Panel Block

Manual Control Panel Block facilitates the option of controlling the output of PLC manually by operating it correctly. This block contains 3 types of input parameter switching devices. They are as follows

- a) Selector switches
- b) Push Buttons
- c) Potentiometers

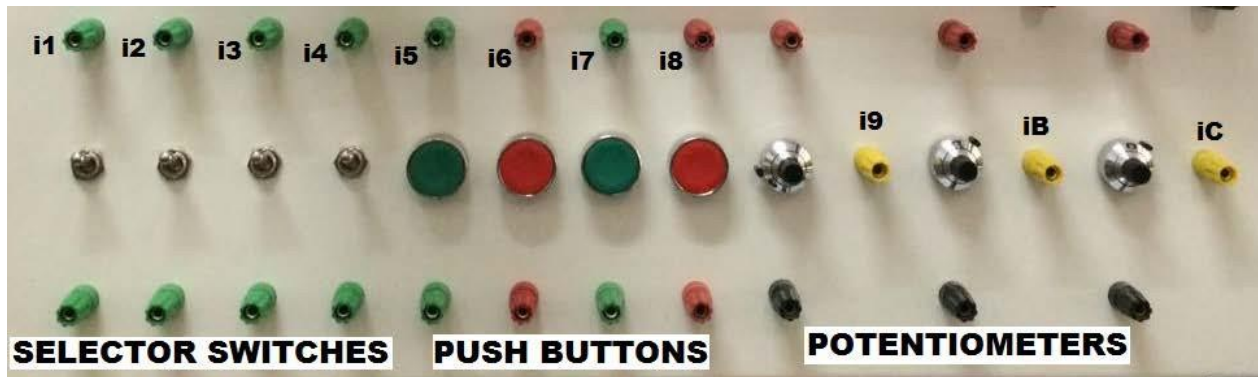


Fig Manual Control Panel Block

The different types of input switching devices are discussed briefly.

a) Selector switches:

Selector switch: A manually operated multi-position switch, which is usually adjusted by a knob or [handle](#), and may have detents to hold in a given [position](#). Used for instance, in devices or instruments with multiple functions, ranges, or modes of [operation](#). Such a switch is usually rotary also called [selector](#). There are 2 positions NC and NO. When selected in NC the device is in OFF mode whereas when selected in NO position the device gets connected and it is in ON mode.

Terminals:

In this trainer kit there are 4 selector switches dedicated for controlling input of PLC i1-i4 (4 inputs). The upper terminals are connected to input of PLC and the lower are connected to positive (+) terminal.

Programmable Logic Controller

b) Push Buttons

Push Button: A push-button or simply button is a simple switch mechanism for controlling some aspects of machine or a [process](#). A push button is a momentary or non-[latching switch](#) which causes a temporary change in the state of an [electrical circuit](#) only while the switch is physically actuated. An automatic mechanism (i.e. a spring) returns the switch to its default position immediately afterwards, restoring the initial circuit condition.

Terminals:

In this trainer kit there are 4 Push Button (2NC and 2NO) dedicated for controlling input of PLC i5-i8 (4 inputs). The upper terminals are connected to input of PLC and the lower are connected to positive (+) terminal.

c) Potentiometers

A potentiometer or a pot is a three-[terminal resistor](#) with a sliding or rotating contact that forms an adjustable [voltage divider](#). If only two terminals are used, one end and the wiper, it acts as a variable resistor or rheostat. The measuring instrument called a [potentiometer](#) is essentially a voltage divider used for measuring [electric potential](#) (voltage)

Terminals:

In this trainer kit there are 3 Potentiometers dedicated for controlling input of PLC i9-iA (3 inputs). The middle terminals are connected to input of PLC, the lower are connected to positive (+) terminal and the upper terminals are connected to negative (-) terminal.

5. Output Testing Unit

Prior to connection of all the hardware input output devices, the output can be tested on a set of bulbs as a precautionary measurement for safe through monitoring of PLC and output devices. There are 8 bulbs of ac 10W, 230v. These all are connected to output of relay unit powered by 1-phase supply. Figure shows a set of 8 bulbs for output indication.



6. Protective Devices

The main important device in all electrical equipment is the protective device. In this kit before giving the supply to SMPS, it's passed through a MCB which isolate the whole equipment from power supply and shut down the system during fault condition, by isolating healthy part from the faulty one.



Fig MCB

For avoiding lengthy and complex power circuit connection, both the ac and dc power terminals are dropped at one corner of the kit for ease of connections.

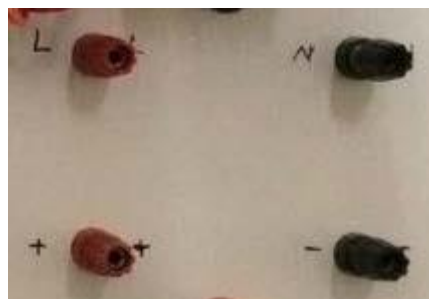
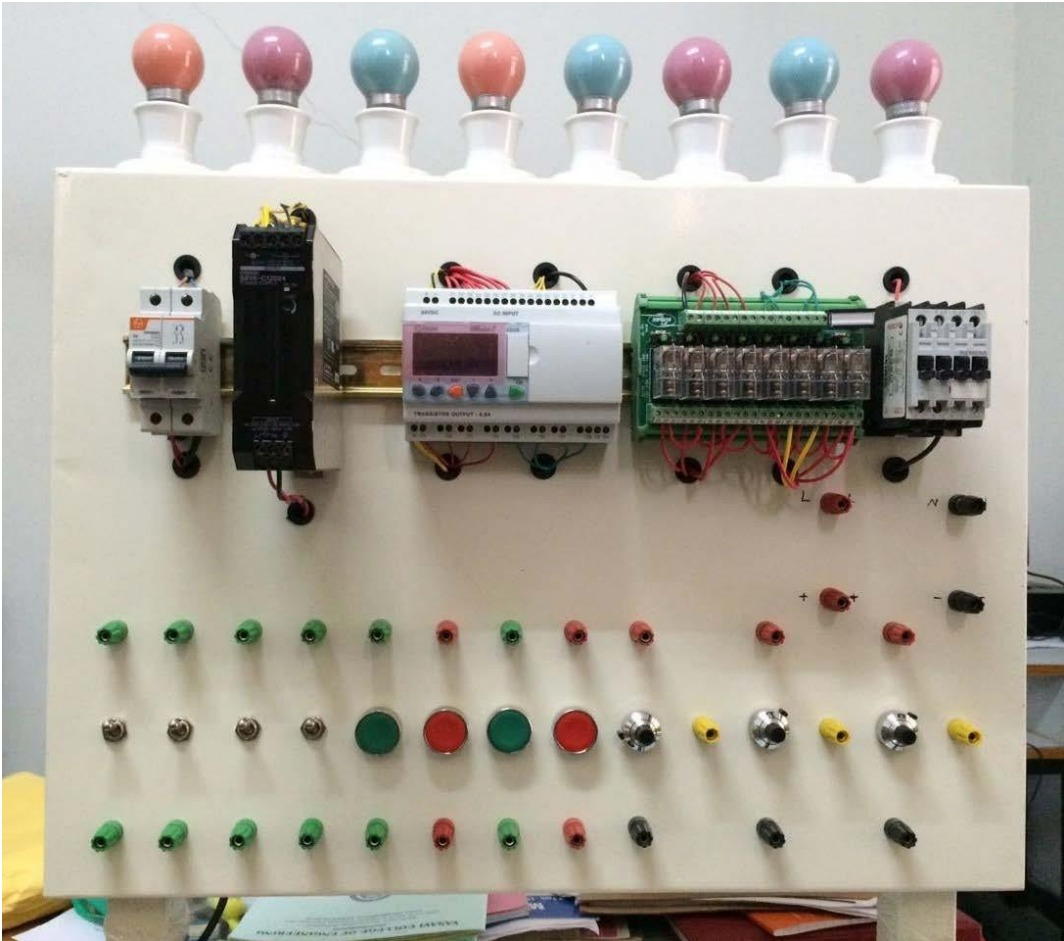


Fig ac and dc power terminals

Programmable Logic Controller

PLC Trainer Kit:



Photograph of PLC TRAINER KIT

4. FUNCTIONAL BLOCK DIAGRAM (FBD)

FBD MODE OF CROUZET MILLENIUM 3 PROGRAMMING

FBD mode allows graphic programming based on the use of predefined function blocks.

It offers a large range of basic functions: timer, counter, logic, etc.

In FBD programming, there are two types of window and two displays: They are

1. The edit window
 - a). Program view
 - b). Setting view



2. The supervision window

Let us discuss in detail as follows

1. Edit Window

a) Program View

The FBD programs are created in the edit window in **Program view**. This can be accessed by

using the  buttons on the controller bar and the  button on the standard toolbar.

The edit window is made up of three zones:

1. The wiring sheet, where the functions that make up the program are inserted.
2. The Inputs zone on the left of the wiring sheet where the inputs are positioned.
3. The Outputs zone on the right of the wiring sheet where the outputs are positioned.

The input and output are specific to the type of controller and extension chosen by the user.

The program in the edit window corresponds to the program that is:

1.

Compiled

2. Transferred to the controller

3.

Compared to the contents of the controller

4. Used in simulation mode

5. Used in supervision mode

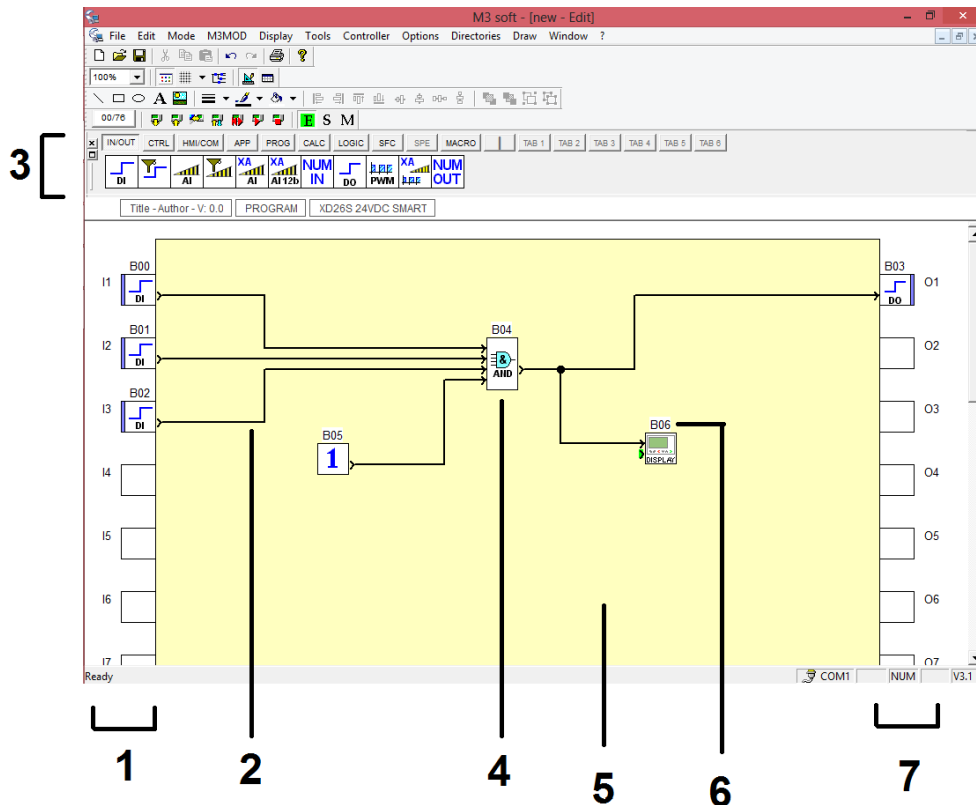
Lists of different elements in the edit window are

1. Function block input zone.
2. Connection between two function blocks.

Programmable Logic Controller


3. Function bar. 4.
- Function block. 5.
- Wiring sheet. 6.
- Function block number. 6.
7. Output function block zone

The figure below shows an example of an edit window in FBD language:



b) Settings View

The

Settings view can be accessed in Edit mode using the  button. It is used to list all automation functions with parameters used in the application. The general interface allows the user to view all the information:



1. **Block:** function block diagram
2. **Function:** Timer, Counter, etc.
3. **Block num:** function block ID
4. **Parameters:** the target value for a counter, etc. ,

Programmable Logic Controller

- 5. Save on power failure:** indicates whether the Save on power failure option has been selected
- 6. Modification authorized:** indicates whether or not parameter modification is possible from the controller front panel
- 7. Comment:** comments associated with the function

2. Supervision Window

The Supervision window can also be accessed from the following modes:

- 1. Simulation :** In the Mode/Simulation menu or using the simulation button  on the controller bar
- 2. Monitoring:** In the Mode/Monitoring menu or using the monitoring button  on the controller bar. It contains the functions, without their connections, that the programmer has extracted (using Drag/Drop or Copy/Paste) from the edit window. The window can also contain drawings, text and images. In simulation and monitoring mode, the parameters and outputs of the functions present are updated.

Function Bar

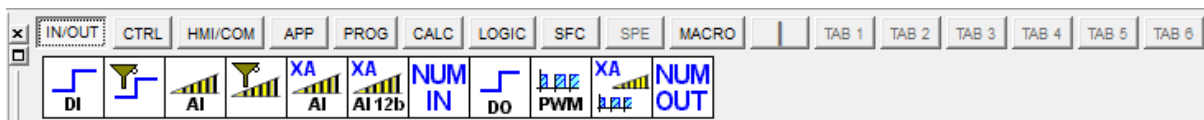
To create an FBD program, the different functions to be inserted in the wiring sheet are available in a function bar. Each of the tabs in the function bar groups a function type.

When the mouse is moved over one of the tabs, the dialog box displays the list of available variables.

A) Inputs Function Bar and Outputs Function Bar

The following figure shows the input/output function bar and HMI/communication function bar:

1. Input/Output Bar (IN/OUT)



2. Human Machine Interface/Communication(HMI/COM)

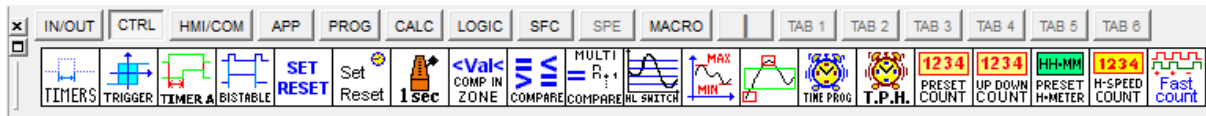
Programmable Logic Controller



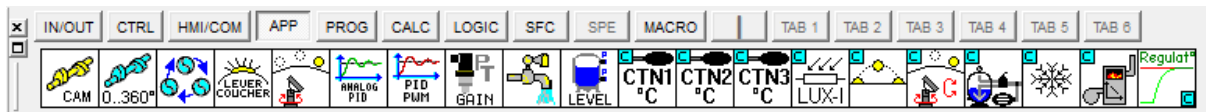
B) Standard Functions Function Bar

The following figure shows the standard functions function bar:

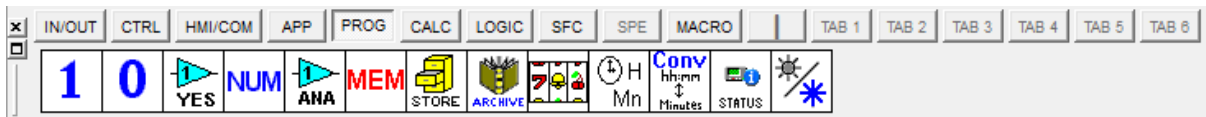
1. Control Bar (CTRL)



2. Application Bar (APP)



3. Programmation Bar (PROG)



4. Calculation Function Bar (CALC)



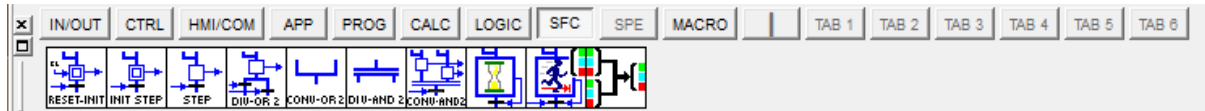
5. Logic Functions Function Bar

Programmable Logic Controller



C) SFC Functions Function Bar

The following figure shows the SFC function bar:



FBD Language Elements

The various blocks/elements available in FBD are as follows

1. Input/Output functions (IN/OUT)

1. Discrete Input
2. Filtered Discrete Input
3. Analog Input 0-10v
4. Filtered Analog Input
5. Analog Input Expansion 10-bits
6. Analog Input Expansion 12-bits
7. Input Integer
8. Discrete Output
9. PWM Output
10. Analog Output Expansion 10-bits
11. Input Integer
12. Output Integer

2. Human Machine Interface/Communication functions (HMI/COM)

1. LCD Display
2. Text
3. Menu Scroll
4. Controlled LCD Backlighting
5. Button A
6. Button B

Programmable Logic Controller

7. Escape Button
8. Minus Button
9. Plus Button
10. Ok Button
11. Serial Port Input
12. Serial Link Protected Input
13. Alarm

3. Control functions (CTRL)

1. Timers (Function A-C, BW, Li/L, B/H, Totaliser)
2. Schmitt Trigger
3. Timer A
4. Impulse Relay (Bi-stable)
5. RS switching (SET RESET)
6. SET RESET switching
7. 1 sec Clock
8. Compare in zone
9. COMPARE
10. Multi Compare
11. HL switch
12. Min Max function
13. Reduced Average
14. Time Programmer
15. Weekly Time Programmer
16. Preset count up/down Counter
17. Up/down Counter
18. Preset Hour Counter
19. High Speed Counter
20. Fast Counter

4. Application functions (APP)

1. CAM Block
2. Angular CAM Timer

Programmable Logic Controller

3. Pump Management
4. Sunrise and Sunset Time
5. Solar Tracking one Axis
6. Analog PID
7. PID PWM
8. Pressure Gain
9. FLOW
10. LEVEL
11. CTN-1
12. CTN-2
13. CTN-3
14. LUX-1
15. TWILIGHT
16. Solar Tracking dual Axis
17. Swimming pool filtration
18. DEFROST
19. Heat Curve
20. Analog PID regulator

5. Programmation functions (PROG)

1. Constant ON
2. Constant OFF
3. YES function
4. Numerical Constant
5. YES NUM
6. MEMORY
7. STORAGE
8. ARCHIVE
9. RANDOM
10. Hour minute
11. Hour minute conversion
12. Controller Status

Programmable Logic Controller

13. Summer Time

6. Calculation Functions (CALC)

1. Gain
2. ADD/SUB
3. MUL/DIV
4. ADD SUB
5. SIN COS
6. SQUARE ROOT
7. BIT MULTIPLEXER
8. MULTIPLEXING
9. DE-MULTIPLEXER
10. MULTIPLEXER
11. CONVERSION WORD-BIT
12. CONVERSION BIT-WORD
13. SPLIT BY 4
14. SPLIT BY 2
15. WORD SHIFT
16. SHIFT REGISTER
17. TRANSFER FUNCTION
18. TRANSFER FUNCTION 50
19. TIMER TRANSFER FCT
20. TIMER TRANSFER FCN 50

7. Logic Functions Functions (LOGIC)

1. Logical NOT
2. Logical AND (2 input)
3. Logical AND (4 input)
4. Logical AND (6 input)
5. Logical OR (2 input)
6. Logical OR (4 input)
7. Logical OR (6 input)
8. Logical NOT-AND

Programmable Logic Controller

9. Logical NOT-OR
10. Logical EXCLUSIVE-OR
11. BOOLEAN function (1 output)
12. BOOLEAN function (2 output)

8. SFC Functions (SFC/Grafcet)

1. Resettable initial SFC step
2. Initial SFC step
3. SFC step
4. Divergence to OR with SFC branches
5. Convergence to OR with SFC branches
6. Divergence to AND with SFC branches
7. Convergence to AND with SFC branches
8. WAIT SFC STEP
9. MOVE SFC STEP
10. Motor Multiplexer

9. MACRO

1. 15 DISPLAYS SCROLL
2. 4 DISPLAYS SCROLL

Programmable Logic Controller

4.1 Study of basic control function

4.1.1 Types of I/O blocks: Input and output blocks can be accessed from IN/OUT functions bar. These symbols are helpful in easy understanding and construction of program.

The input blocks are as follows:

A) Discrete Blocks

1. Discrete type input
2. Filtered discrete type input

B) Analog Blocks

1. Analog input
2. Filtered analog input

C) Integer type

D) Special input in FBD Language

1. Button
2. Discrete constants
3. Numerical constants
4. Summer time
5. Blinking for 1 second

E) Extension Bit Integer Input

1. Extension 10-Bit integer input
2. Extension 12-Bit integer input

The output blocks are as follows:

A) Discrete-Type Outputs

B) LCD Screen Backlighting Output

C) Integer Type Output

D) PWM-Type Output

Programmable Logic Controller








E) Extension 10-Bit Analog Output

Types of input blocks

The different type of inputs can be selected from the Parameters window which is displayed in the edit and supervision windows. Different inputs are discussed as follows

A) Discrete Blocks:

1. Discrete type input: The type of Discrete input can be selected from the Parameters window. This is then displayed in the edit and supervision windows.

- | | | |
|----|---|-------------------------|
| a) |  | Discrete input |
| b) |  | Contact |
| c) |  | Limit Switch |
| d) |  | Proximity sensor |
| e) |  | Presence sensor |
| f) |  | Illuminated push-button |
| g) |  | Selector switch |
| h) | | Push-button |

Programmable Logic Controller



i)

Normally open relay



2. Filtered discrete type input:



The filtered Discrete input function is accessible from the **IN** window. Along with the discrete input, a filter is added to reduce or even eliminate disturbances. A Discrete input is filtered using a constant level detection algorithm (1 or 0) on the "sensor" signal, measured over a certain time frame. If the signal is stable throughout the entire detection period, the output of the symbol from the filtered Discrete input takes the value of the measured signal. Otherwise it remains unchanged. The filtered discrete inputs can be arranged at any controller input. The value of the parameter (between 1 and 255) entered in the **Parameters** window may be used to define the minimum time during which the signal must be stable. This value is a multiple of the duration of the basic cycle of the controller.

B) Analog Blocks:

1. Analog input



The Analog input function is accessible from the IN window. The Analog input is available on all types of controller supplied with a DC voltage. The Analog input voltage is converted into a numerical integer value by a 10-bit analog/digital converter. The integer value of the output is between 0 and 1023. Analog inputs can only be arranged on the inputs between IB and IG. By default, this voltage varies between 0 and 10V DC. The type of electrical connection at the input can be configured in the Parameters window: 0 - 10 V Potentiometer, selected if the input is connected to a potentiometric device with a power supply between 0 Volts and the controller supply voltage. The type of Analog input can be selected from the Parameters window which is displayed in the edit and supervision windows.

Programmable Logic Controller

- a) Analog Input (default analog input)



- b) Analog Input (on scale)



- c) Temperature



- d) Potentiometer



2. Filtered Analog Input



The filtered Analog input function is accessible from the IN window. Along with the analog input, a low pass filter is added. This function is available on all the types of controllers supplied with a DC voltage. The Analog input voltage is converted into a digital integer value by a 10-bit analog/digital converter. The whole output value is between 0 and 1023. The Analog inputs can only be placed on the inputs numbered from IB to IG. A low pass filter restores the entire input signal (frequency, amplitude and phase-shift), whose frequency is considerable lower, to a typical filter frequency, called a cut-off frequency. When the frequency of the input signal nears the cut-off frequency, the output signal, of the same frequency, becomes increasingly lower and phase-shifted. When the frequency of the input signal is equal to the cut-off frequency, the output signal is lowered by around 30%, and phase-shifted by 45°. For a frequency above and rising from the cut-off frequency, the reduction becomes greater (until it reaches total elimination) and the phase-shifting approaches 90°. The Parameters window is used to define the input voltage and the cut-off frequency of the low pass filter (between 0.06 and 88.25 Hz).

Care should be taken when the input is connected to a potentiometric device supplied between 0 Volts and the tester supply voltage. Whenever a modification is made to the basic cycle time, you must check or modify the cut-off frequency to avoid malfunctioning and damage of equipment.

C) Integer type:

Integer Type Input



The integer input function is accessible from the IN window. This function is used to enter a 16-bit (-32768, +32767) integer from the outputs of certain connected extensions. Integer type inputs can be positioned on the inputs (J9 to JB) of the extension controllers.

D) Special Inputs in FBD Language

In FBD, various special inputs are available that can be accessed from the IN window. These inputs cannot be inserted in the input plots of the diagram sheet.

1. Button



Button-type Inputs:

Button-type inputs correspond to the keys available on the front panel of the controller. These inputs can be inserted in an FBD diagram and, in Simulation and Monitoring mode it act as simulate contacts and can be controlled.

2. Discrete constants:

Discrete Constant-Type Inputs:

- a) 1 constant



- b) 0 constant



There are two types of Discrete constants: the 1 constant and the 0 constant . These two constants can be used to set the function inputs to 1 or 0. In Simulation or Monitoring modes, we can force these inputs in the reverse order.

3. Numerical constants

Numerical Constant-Type Inputs:

Programmable Logic Controller



The numerical constant NUM is an integer with a value between -32768 and +32767. This constant can be used to set values to the functions whose inputs cannot be connected to other than numerical blocks. The value of the constant can be set in the Parameters window. In Simulation or Monitoring modes, it is also possible to modify the constant value.

4. Summer time

Summer Time Input:

- a) During Summer Time



- b) During Winter Time



The summer time input function is active throughout summer time, and inactive throughout Winter time. While configuring this function we should check Program configuration window, the date format, status of the Activate the summer/winter time change box and selection of either preset geographical zone or manually configuring the dates when the time change takes place.

5. Blinking for 1 second

Blinking Input:



The blinking input function is active every second. It can be used for generating 1 sec ON/OFF pulses which can be utilised to activate another block.

E) Extension Bit Integer Input

- 1. Extension 10-Bit integer input



Programmable Logic Controller

The extension 10-bit integer input function is accessible from the **IN** window and can be configured from the **Analog extension** tab of the **Program**. In Simulation or Monitoring modes, you can force (between 0 and 4095) the output of the analog inputs. The extension **10-bit integer** inputs are available on controllers that are compatible with the XAO4 24VDC analog I/O extension. 10-bit integer inputs can only be arranged on IPXA and IQXA input squares of the XAO4 24VDC analog I/O extension. The analog input voltage is converted into a numerical integer value by a 10-bit analog/digital converter. The converter output integer value is between 0 and 1023. In Simulation or Monitoring modes, you can force (between 0 and 1023) the output of the analog inputs

2. Extension 12-Bit integer input



The function Extension 12-bit integer input is accessible from the IN/OUT window. The extension 12-bit integer inputs are available on controllers that are compatible with analog I/O 12-bit extension. The analog input voltage is converted into a numerical integer value by a 12-bit analog/digital converter. The integer value of the converter output is between 0 and 4095. In Simulation or Monitoring modes, you can force (between 0 and 4095) the output of the analog inputs

Types of output blocks

The controllers feature two types of Discrete outputs

- a) Solid state outputs for certain controllers supplied with a DC voltage,
- b) Relay outputs for controllers supplied with an AC or DC voltage.

The different type of output can be selected from the Parameters window which is displayed in the edit and supervision windows. Different output are discussed as follows

A) Discrete-Type Outputs:



Programmable Logic Controller

The Discrete output function is accessible from the OUT window. The type of Discrete output can be selected from the Parameters window which is then displayed in the edit and supervision windows. The selection is made using the output's inactive-state symbol

a) Discrete output



b) Normally open relay



c) Lamp



d) Solid state relay



e) Valve



f) Actuator



g) Motor



h) Resistance



i) Audible signal



j) Green light signal

Programmable Logic Controller



k) Red light signal



l) Orange light signal



m) Indicator light



n) Heating



o) Fan



B) LCD Screen Backlighting Output

LCD Screen Backlighting Output:



The LCD Screen Backlighting Output function is accessible from the **OUT** window. The LCD screen **BACK LIGHT** output enables the program to control the backlighting of the LCD display of the controller. As long as the connected input is active, the backlighting is on. This function cannot be arranged on the controller outputs. The following table lists the symbols of the LCD Screen Backlighting function in Simulation or Monitoring modes.

C) Integer Type Output



The integer output function is accessible from the **OUT** window. This function is used to create a 16-bit (-32768, +32767) integer output towards the integer-type inputs of certain extensions connected to the controller. Integer-type outputs can be positioned on the outputs (O9 to OB) of the extension controllers. If the function input is not connected, the output is 0.

D) PWM-Type Output



The PWM output function can be accessed from the OUT function bar. Depending on the model, controllers with solid state outputs have one or more discrete outputs that can be controlled using PWM (pulse width modulation). The mean value of the PWM output voltage is then proportional to the 8-bit analog set point. This enables a value between 0 and 100% of its maximum value to be controlled using a discrete output. 0% corresponds to set point 0 and 100% corresponds to set point 255. The basic frequency of all the PWM-type controller outputs can be set from the Configuration tab of the Program Configuration window. The Frequency of all controller PWMs parameter enables the basic frequency for PWM-type outputs to be selected from as 1806 Hz, 452 Hz, 226 Hz, 113 Hz, 56 Hz and 14 Hz. The physical outputs of the controller in PWM mode are those with high indices as follows: O4 for controllers reference CD12 S and XD10 S, O4 to O7 for controllers reference CD20 S, O4 to O7 for controllers reference XD26 S.

E) Extension 10-Bit Analog Output



The 10-bit integer output function is accessible from the OUT window. The 10-bit integer outputs are configured from the Analog extension tab of the Program Configuration window accessible by clicking on the XAO4 24VDC button, see XAO4 24VDC Analog. 10-bit analog outputs are available on controllers that are compatible with XAO4 24VDC analog I/O extensions. 10-bit analog outputs can only be arranged on OF XA and OG XA output blocks of

Programmable Logic Controller

the XAO4 24VDC analog I/O extensions. A 10-bit analog/digital converter converts the numerical integer value into an output voltage that may be either:

- a) PWM (pulse width modulation): In this case, the output voltage is periodic and takes the values 0 V and 10 V so that its average value is proportional to the analog setpoint. This enables a value between 0 and 100% of its maximum value to be controlled using a discrete output.
- b) Analog: The analog output voltage varies between 0 and 10 V (1023 gives 10 V).

Some basic function of FBD are as follows:

1. Bistable Impulse Relay Function:

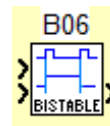
The BISTABLE impulse relay function switches the OUTPUT state on each rising edge (change from inactive to active) of the COMMAND input. The impulse relay function is accessible from the FBD function bar.

It is represented as follows:

Function Bar icon:



Wiring Sheet block:



It is 2 input and 1 output function.

Inputs:

1. COMMAND: This is the input that controls changes in the output state, whose type is Discrete.
2. RESET: When this command is active, the OUTPUT always remains inactive, regardless of the COMMAND input transitions. If the RESET input is not connected, it is considered to be inactive.

Output:

1. OUTPUT: This is the impulse relay output, whose type is Discrete. This value depends upon the state of the RESET input.

If the RESET input is:

- a) Inactive: the OUTPUT changes state in line with the transitions of the COMMAND input,
- b) Active: the OUTPUT always remains inactive.

Programmable Logic Controller

2. SET and RESET function

It is 4 and 1 output function.

The SET RESET function operates as follows:

- a) Activation of the SET input activates the output, which remains so even if the SET input is then deactivated,
- b) Activation of the RESET input deactivates the output,
- c) If both inputs are active, the state of the output depends on the configuration of the function:
 - i. The output is active if the SET has priority option is configured,
 - ii. The output is inactive if the RESET has priority option is configured.
 - iii. Non-connected inputs are set to the Inactive state.

This function is accessible from the FBD function bar. It is represented as follows:

Function Bar icon:



Wiring Sheet block:



Programmable Logic Controller

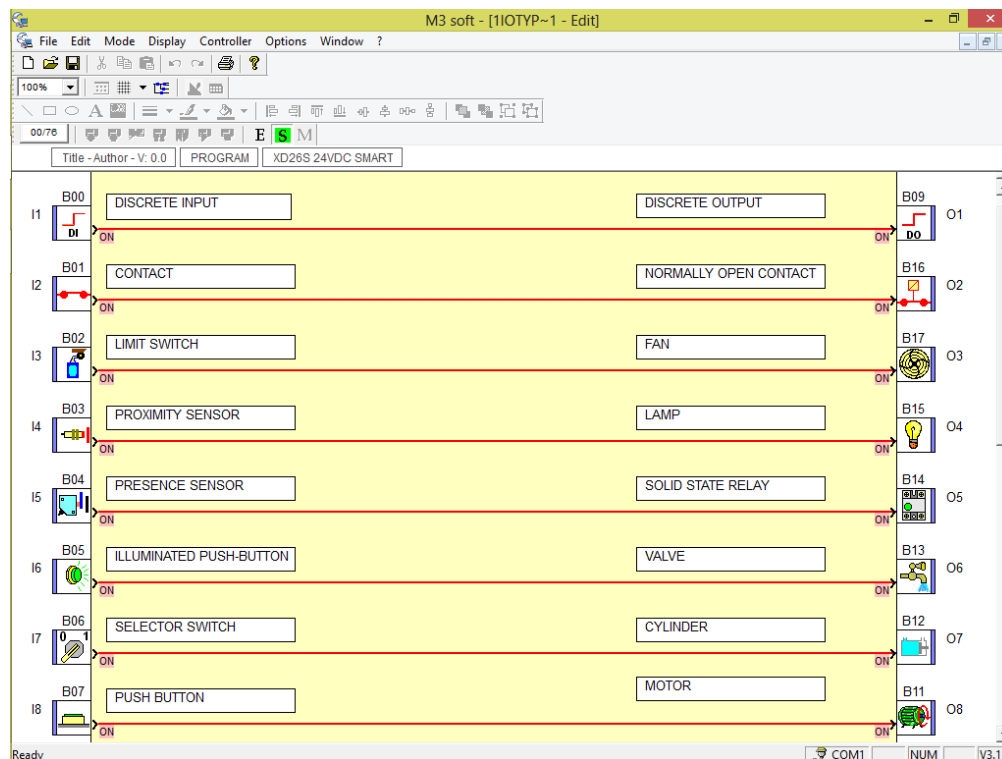
1. A program to illustrate input output blocks:

Program details:

1. Choose controller type and parameter window setting properly for input and output compatibility and validity as per requirement.
2. Make proper connection between blocks, input-output to avoid invalidity and malfunction.
3. Inputs (8 blocks) and output (8 blocks)

Hardware requirement:

1. Proximity sensors, selector switch, push buttons, Relays (2 normal and 1 solid state), Fan, bulb, valve, cylinder and motor.
2. Set of single stranded wires
3. Relay card



Programmable Logic Controller

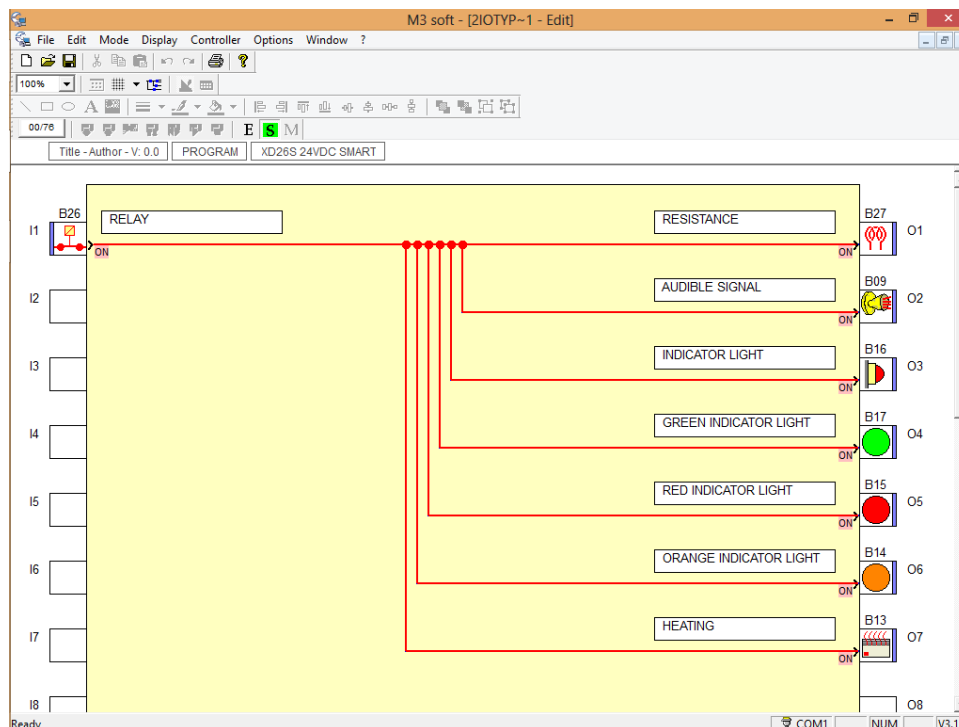
2. A program to illustrate input output blocks:

Program details:

1. Choose controller type and parameter window setting properly for input and output compatibility and validity as per requirement.
2. Make proper connection between blocks, input-output to avoid invalidity and malfunction.
3. Input (1 block) and output (7 blocks)

Hardware requirement:

1. Light (indicator light, green indicator, red indicator, orange indicator), Resistive load, speakers and heating element
2. Set of single stranded wires
3. Relay card



Programmable Logic Controller

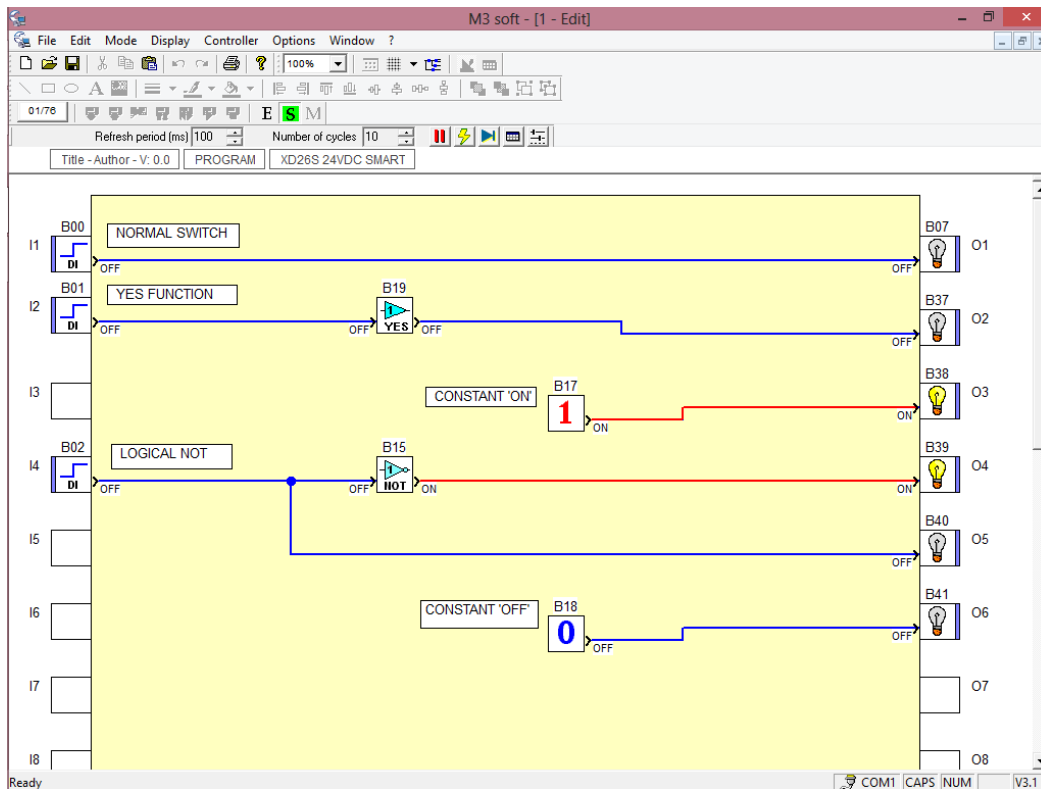
3. A program to illustrate special FBD blocks: YES, CONSTANT 1 and 0

Program details:

1. Choose controller type and parameter window setting properly, for input and output compatibility and validity as per requirement.
2. Make proper connection between blocks, input-output to avoid invalidity and malfunction.
3. Input (3 blocks), output (6 blocks), YES function, NOT function, constant ON and OFF

Hardware requirement:

1. Bulbs (6 No's)
2. Set of single stranded wires
3. Relay card



Programmable Logic Controller

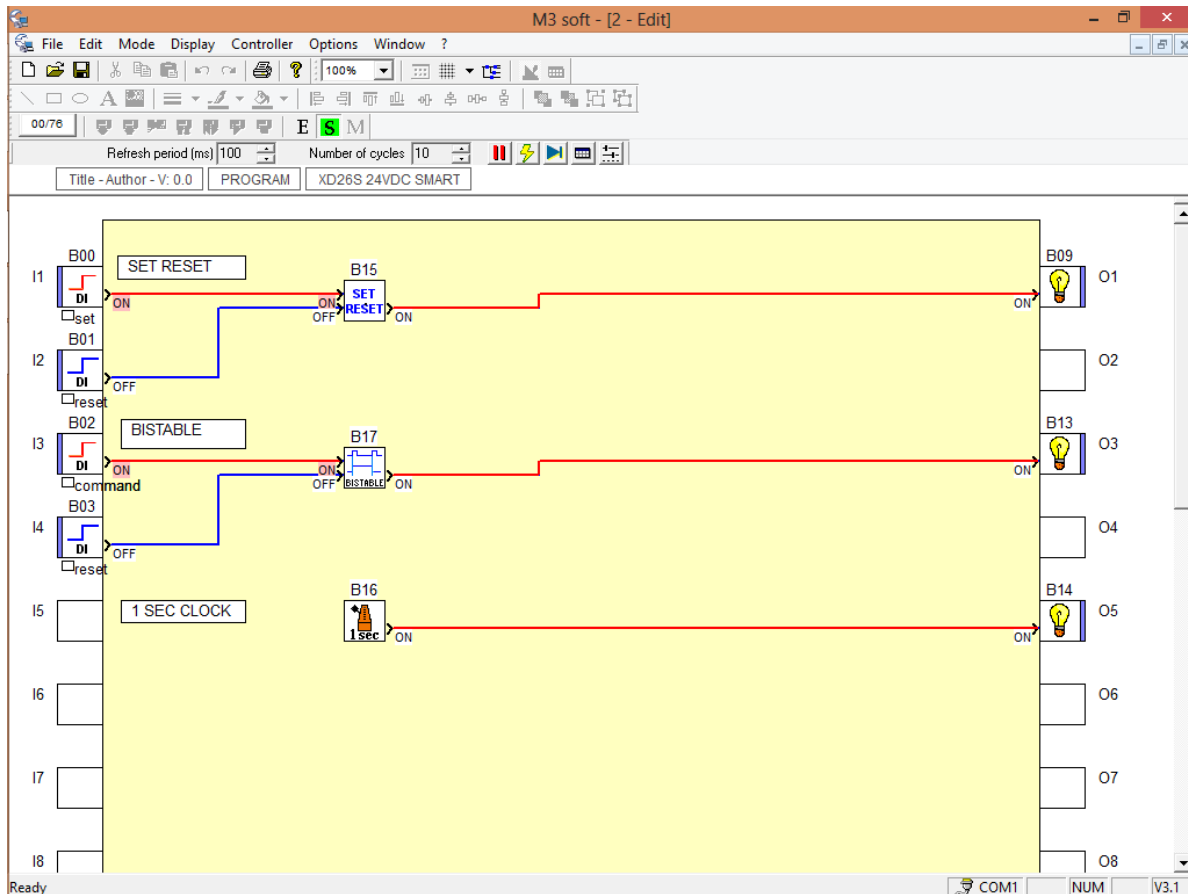
4. A program to illustrate special FBD blocks: 1 SEC CLOCK, SET RESET and Bi-Stable function

Program details:

1. Choose controller type and parameter window setting properly, for input and output compatibility and validity as per requirement.
2. Make proper connection between blocks, input-output to avoid invalidity and malfunction.
3. Input (4 blocks), output (3 blocks), 1sec clock, SET RESET function and bi-stable function.

Hardware requirement:

1. Bulbs (3 No's)
2. Set of single stranded wires
3. Relay card



4.2 Study of logic gates and Boolean functions

Logical Functions

In FBD language, various in-built logic functions are available .They are

1. The NO function,
2. The AND function,
3. The OR function,
4. The NO AND function,
5. The NO OR function,
6. The EXCLUSIVE OR function.
7. BOOLEAN Function

These inputs can be accessed from the LOGIC window.

The various logic functions is discussed in briefly as follows

1. The NO Function:

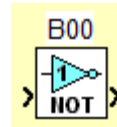
- a) If the input is inactive or not connected, the output is active.
- b) If the input is active, the output is inactive.

It is represented as follows:

Function Bar icon:



Wiring Sheet block:



2. The AND Function:

It is 2/4/6 input and 1 output function.

- a) If all the inputs are active or not connected, the output is active.
- b) If at least one input is inactive, the output is inactive.

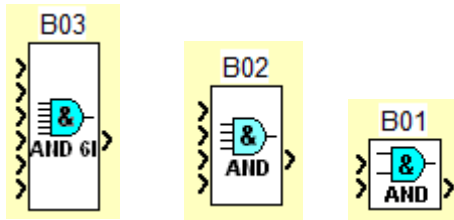
It is represented as follows:

Function Bar icon:



Programmable Logic Controller

Wiring Sheet block:



3. The OR Function:

It is 2/4/6 input and 1 output function.

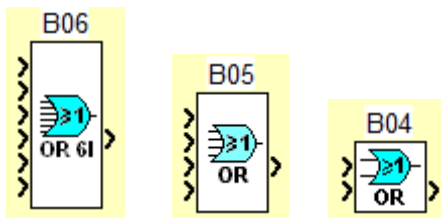
- a) If at least one input is active, the output is active.
- b) If all the inputs are inactive or not connected, the output is inactive.

It is represented as follows:

Function Bar icon:



Wiring Sheet block:



4. The NO AND Function:

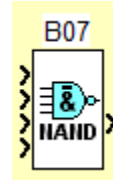
It is 4 and 1 output function.

- a) If at least one input is inactive, the output is active.
- b) If all the inputs are active or not connected, the output is inactive.

It is represented as follows:

Function Bar icon:

Wiring Sheet block:



5. The NO OR Function:

It is 4 and 1 output function.

- a) If all the inputs are inactive or not connected, the output is active.
- b) If at least one input is active, the output is inactive.

It is represented as follows:

Function Bar icon:

Wiring Sheet block:



6. The EXCLUSIVE OR Function:

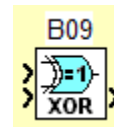
It is 2 and 1 output function.

- a) If an input is inactive and the other input is active or not connected, the output is active.
- b) If both inputs are active or inactive or not connected, the output is inactive.

It is represented as follows:

Function Bar icon:

Wiring Sheet block:



7. BOOLEAN Function:

The BOOLEAN function gives the value of the output according to the combination of inputs.

The function has four inputs, and therefore 16 combinations.

Programmable Logic Controller

These combinations can be found in parameter window as a truth table. For each of these, the output value can be adjusted.

The number of configurable combinations depends on the number of inputs connected to the function. Non-connected inputs are set to 0.

It has two configurations as 4 input, 1 output and 6input, 2output as shown below.

Having connected at least one input, you can configure the value of the output in the truth table, in the Parameters window.

The output values can be 0 for the Inactive state, and 1 for the Active state.

a) By selecting the Output ON if result is TRUE option, the output takes the value configured in the truth table.

b) By selecting the Output OFF if result is TRUE option, the output takes the inverse value of the value configured in the truth table. This function is accessible from the FBD function bar.

It is represented as follows:

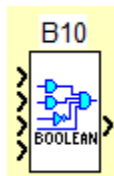
Function Bar icon:



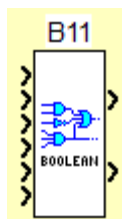
Wiring Sheet block:

It has two configurations as 4 input, 1 output and 6input, 2output as shown below.

a) 4 input, 1 output Boolean function block



b) 6 input, 2 output Boolean function block

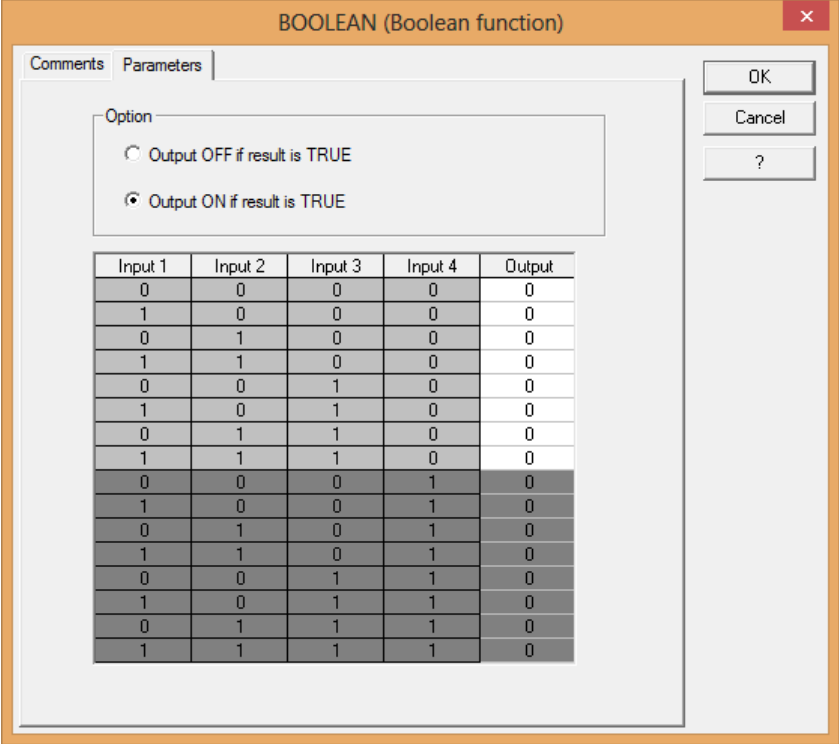


Programmable Logic Controller

Note: For example by connecting only 3 inputs to the Boolean block, we get 8 output combination as $2^3=8$ in the parameter windows as shown below.

Parameter Window for Boolean function:

a) a) 4 input, 1 output Boolean function block



b) 6 input, 2 output Boolean function block

Programmable Logic Controller

BOOLEEN

Comments Parameters

Option

- O1 OFF if result is TRUE
- O1 ON if result is TRUE
- O2 OFF if result is TRUE
- O2 ON if result is TRUE

I1	I2	I3	I4	I5	I6	O1	O2
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0
1	1	0	0	0	0	0	0
0	0	1	0	0	0	0	0
1	0	1	0	0	0	0	0
0	1	1	0	0	0	0	0
1	1	1	0	0	0	0	0
0	0	0	1	0	0	0	0
1	0	0	1	0	0	0	0
0	1	0	1	0	0	0	0
1	1	0	1	0	0	0	0

OK
Cancel
?

Programmable Logic Controller

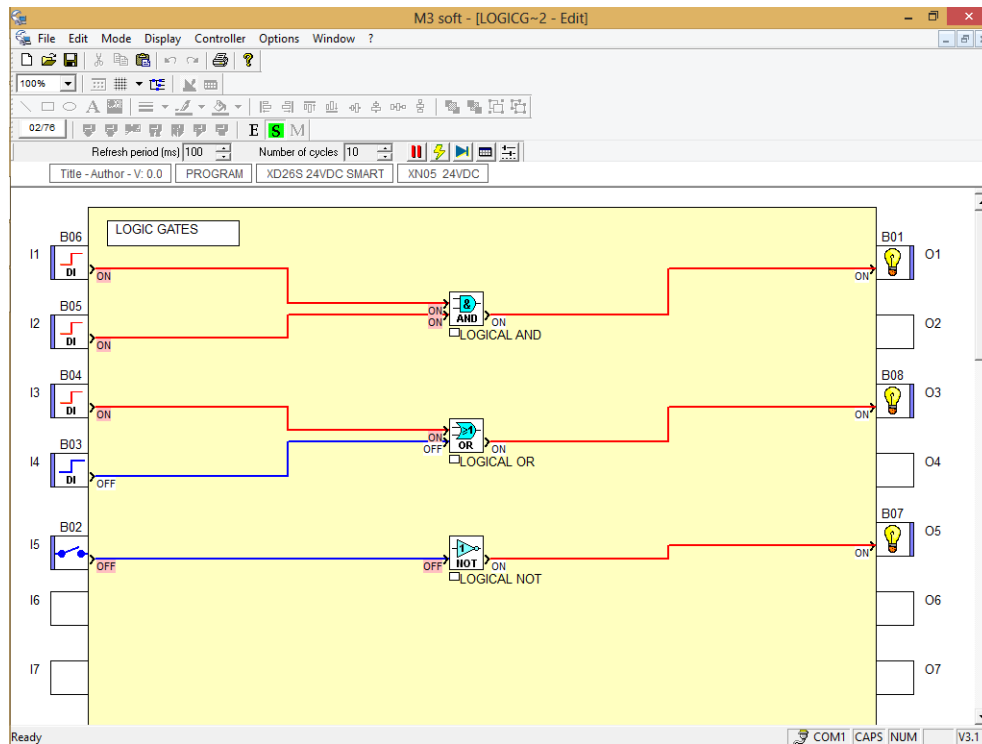
5. A program to illustrate logic gates: AND, OR and NOT

Program details:

1. Choose controller type and parameter window setting properly for input and output compatibility and validity as per requirement.
2. Make proper connection between blocks, input-output to avoid invalidity and malfunction.
3. Input (5 blocks), output (3 blocks), logical AND, OR and NOT.

Hardware requirement:

1. Bulbs (3 No's)
2. Set of single stranded wires
3. Relay card



Programmable Logic Controller

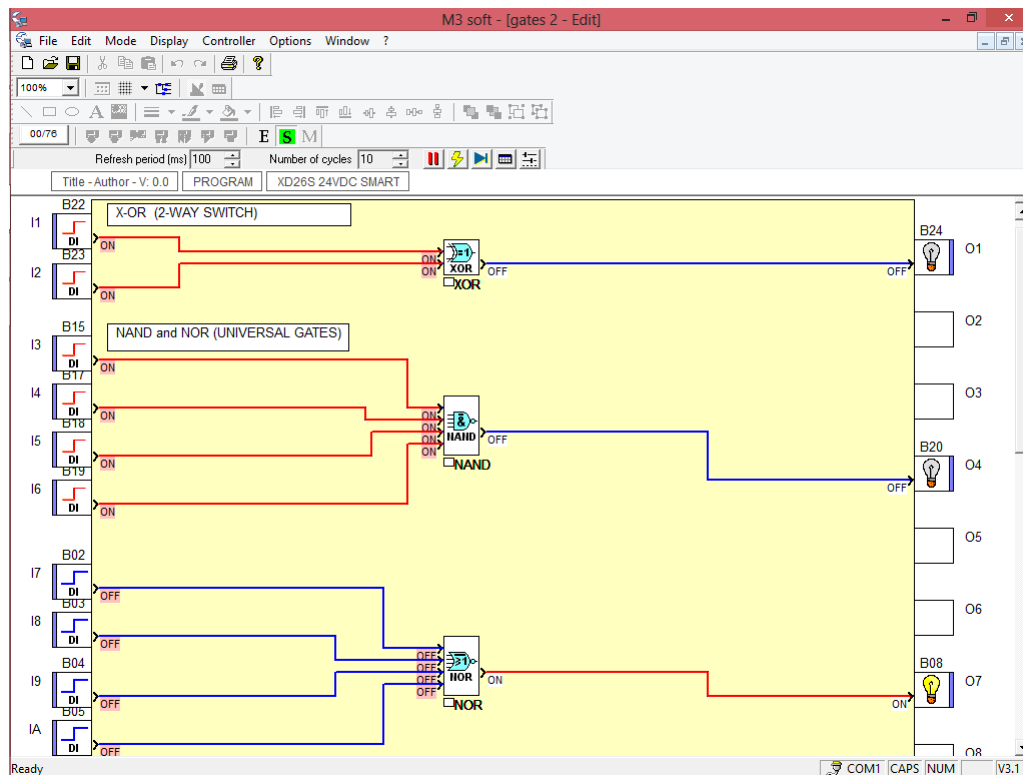
6. A program to illustrate logic gates: XOR, NAND and NOR

Program details:

1. Choose controller type and parameter window setting properly for input and output compatibility and validity as per requirement.
2. Make proper connection between blocks, input-output to avoid invalidity and malfunction.
3. Input (10 blocks), output (3 blocks), logical XOR, NAND and NOR.

Hardware requirement:

1. Bulbs (3 No's)
2. Set of single stranded wires
3. Relay card



Programmable Logic Controller

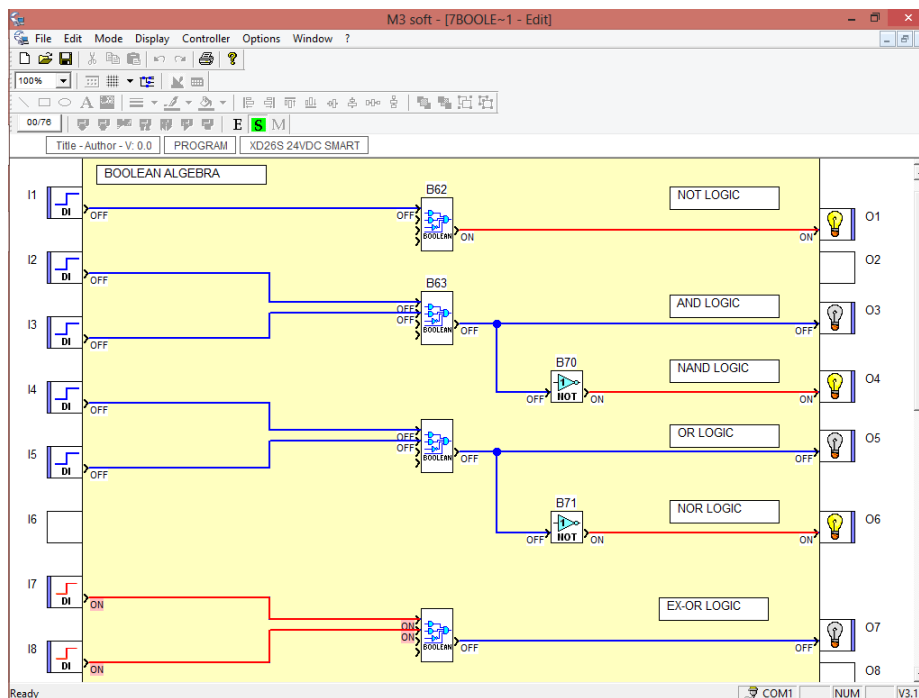
7. A program to illustrate Boolean functions: Realizing AND, OR, NAND, NOR, NOT.

Program details:

1. Choose controller type and parameter window setting properly for input and output compatibility and validity as per requirement.
2. Make proper connection between blocks, input-output to avoid invalidity and malfunction.
3. Input (7 blocks), output (6 blocks), Boolean functions realizing AND, OR, NAND, NOR, NOT.

Hardware requirement:

1. Bulbs (6 No's)
2. Set of single stranded wires
3. Relay card



Programmable Logic Controller

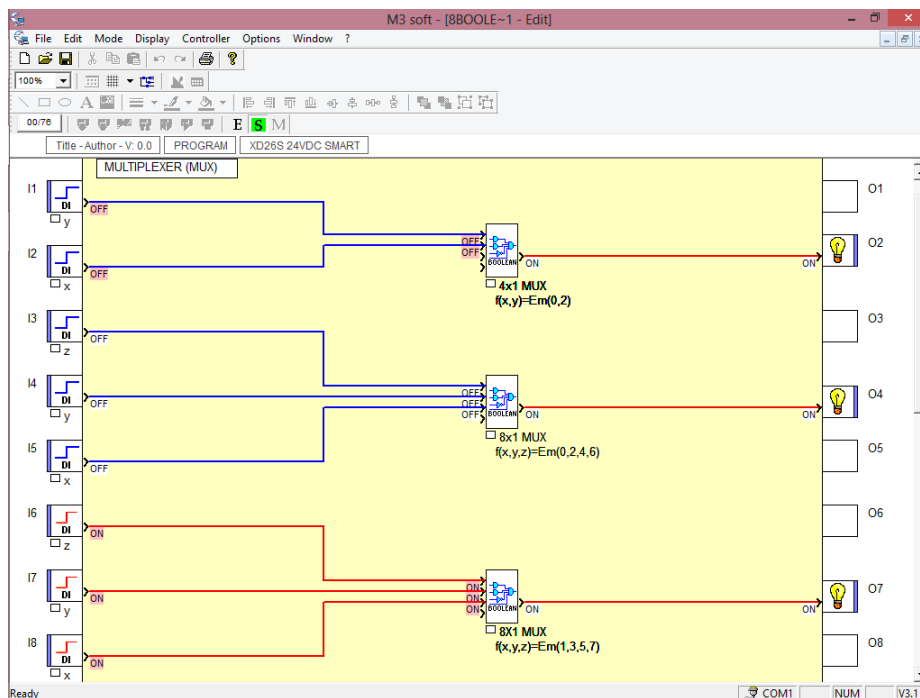
8. A program to realize multiplexers using Boolean functions: Realizing Multiplexers

Program details:

1. Choose controller type and parameter window setting properly for input and output compatibility and validity as per requirement.
2. Make proper connection between blocks, input-output to avoid invalidity and malfunction.
3. Input (8 blocks), output (3 blocks), Boolean functions realizing Multiplexers (MUX): 4x1even MUX, 8x1even MUX, 8x1odd MUX.

Hardware requirement:

1. Bulbs (3 No's)
2. Set of single stranded wires
3. Relay card



Programmable Logic Controller

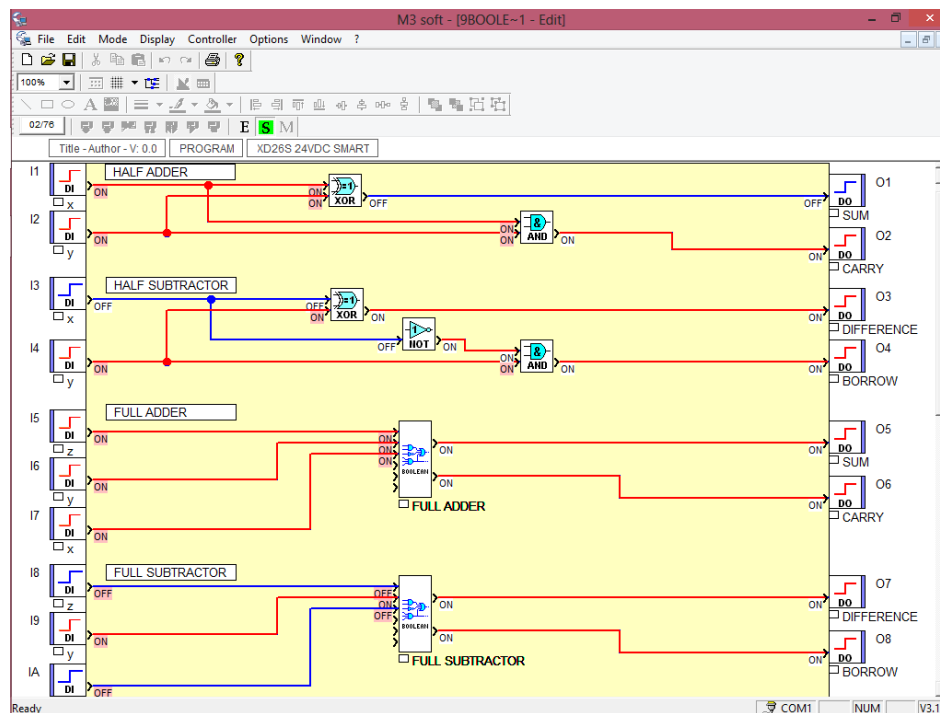
9. A program to realize ADDERS and SUBTRACTORS using Boolean functions:

Program details:

1. Choose controller type and parameter window setting properly for input and output compatibility and validity as per requirement.
2. Make proper connection between blocks, input-output to avoid invalidity and malfunction.
3. Input (10 blocks), output (8 blocks), Half adder, Half subtractor, Full adder and Full subtractor.

Hardware requirement:

1. Bulbs (8 No's)
2. Set of single stranded wires
3. Relay card



4.3 Study of PLC timer functions

TIMERS

In FBD there are various timer function block with different functionalities and applications.

They are as follows

1. Timer A
2. Timer A-C
3. Timer BW
4. Timer Li
5. Timer B/H
6. Timer Totaliser function

There are two timer icons on function bar (CTRL bar), with one icon we can have a direct access for timer A and on opting another icon we get a message in parameter window to opt one timer from mentioned above list.

The TIMERS function block provides access to the following types of timers:

1. Timer A

It is 1 input and 1 output function block. This timer is used to only for on-delay action over preset period of time.

Input:

COMMAND: a) When the input is inactive the output is inactive

b) When the input is active the output will be active after a predefined time **Output:**

OUTPUT: Output Value

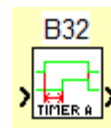
corresponding to the input as defined as UNIT used to select the time unit for the delays, which can be expressed in seconds, tenths of seconds or in number of cycles. Setting for Timer is done to set the ON DELAY value.

Timer A is represented as follows:

Function Bar icon:

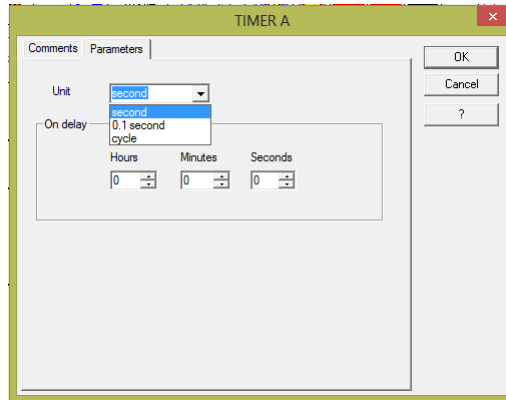


Wiring Sheet block:

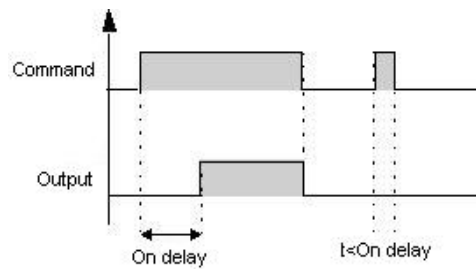


Programmable Logic Controller

Parameter Window for TIMER A function:



Timer A timing diagram:



2. Timer A/C:

It is 2 input and 1 output function block. This timer is used for both on-delay and off-delay action over preset period of time.

Function A: Timer on-delay

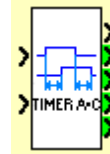
Function C: Timer off-delay

Function A-C: Combination of functions A and C

Timer A/C is represented as follows:

Function Bar icon:

Wiring Sheet block:




Selection of Timer:

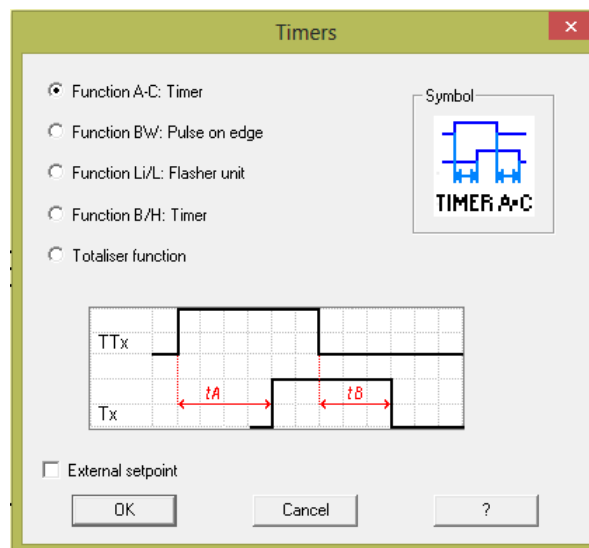
To place a timer on the wiring sheet and select its type, proceed as follows:

Step 1:



Click on the  icon of the FBD function bar, hold down the mouse button and slide the icon to the required place into the wiring sheet.

Then a window appears as shown below



2 Select the desired type of timer from the 5 types available in this window.

3 If required, you can check the External set point box: in this case the on time(s) or off time(s) will be integer type timer block inputs rather than being internal configurable parameters.

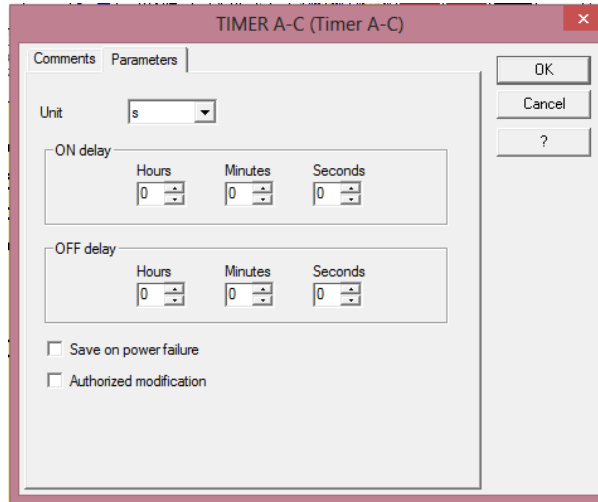
Timer A/C Parameters:

The Parameters window is used to:

1. Select the time unit of the delays; these delays can be expressed in seconds, tenths of seconds or number of cycles.

Programmable Logic Controller

2. Set the ON delay value for function A, only if the External setpoint box was left unchecked when the timer type was selected.
3. Set the OFF delay value for function C, only if the External setpoint box was left unchecked when the timer type was selected.



Possibly activate the Save on power failure parameter. It enables the timer to be restarted at the point where it stopped after a power failure.

The combination of both the ON and OFF delays can be used to obtain a function A/C.

Input:

COMMAND: a) When the input is inactive the output is inactive

b) When the input is active the output will be active after a predefined time

c) Then when input goes low then the output remains high until the off delay time is met.

RESET: When this terminal goes active then the process of timer stops and output gets inactive at that instant and delay time count gets reset to 0.

Output: There are 5 output terminal as follows

1. Output Value corresponding to the input as defined as UNIT used to select the time unit for the delays, which can be expressed in seconds, tenths of seconds or in number of cycles. Setting for Timer is done to set the ON DELAY value. Each pulse on the COMMAND input of the TIMERS timer block resets the current value to 0.
2. ON delay set point
3. ON delay current value
- 4.

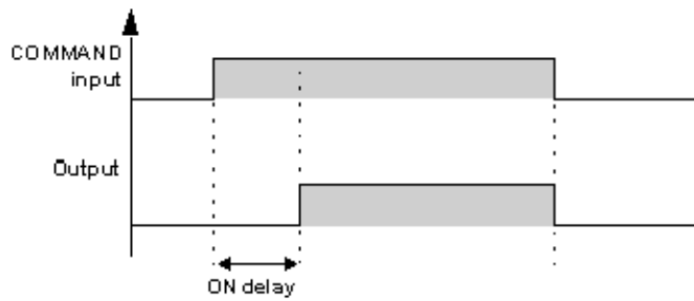
OFF delay set point

5.

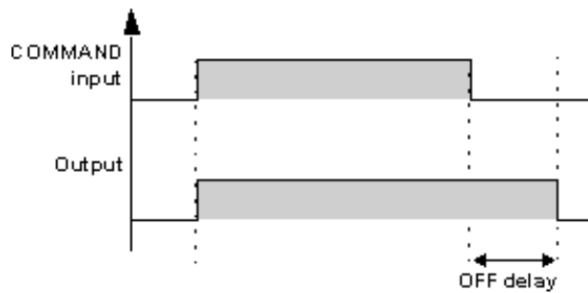
OFF delay current Value

Timing Diagrams For Timer A/C are as follows:

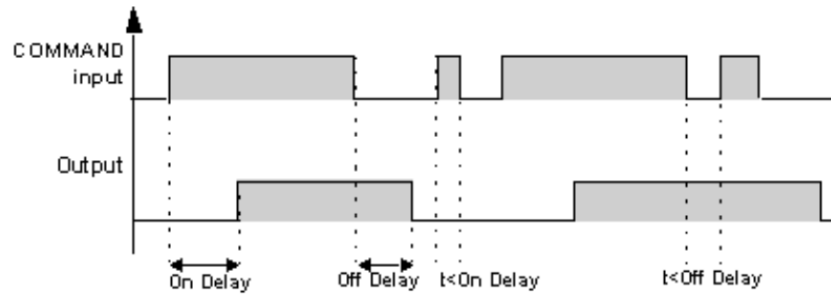
Function A:



Function C:

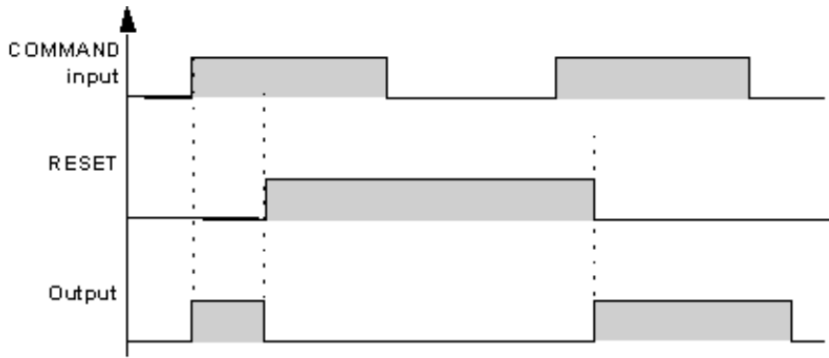


Function A-C:



Function A-C with reset:

Programmable Logic Controller



3. Timer BW: It is used to create a pulse for the duration of a cycle on the output from an input edge. It is 1 input and 1 output function block. Timer BW is represented as follows:

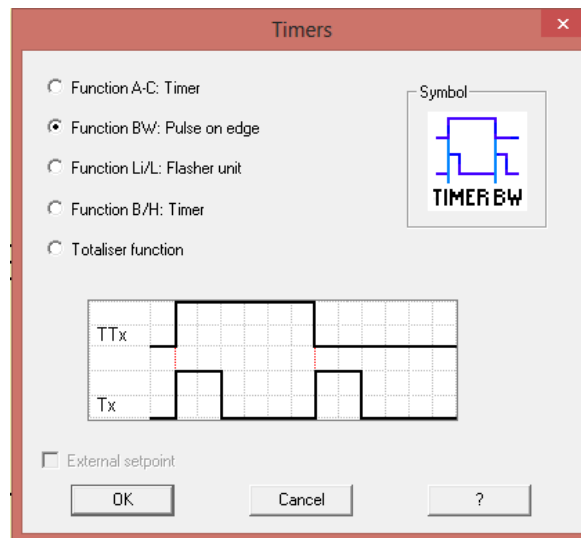
Functional Bar icon:



Wiring Sheet Block:



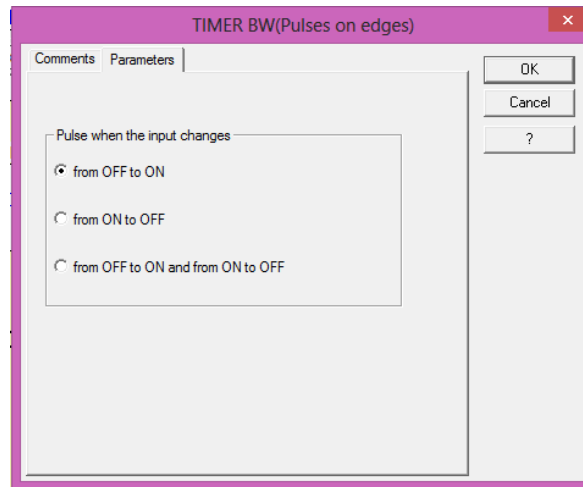
Selection window:



The **Parameters** window is used to select the type of edge on the input that will generate the pulse on the output. Select:

- From OFF to ON** to generate a pulse on each rising edge of the input
- From ON to OFF** to generate a pulse on each falling edge of the input
- From OFF to ON and from ON to OFF** to generate a pulse on each rising edge and each falling edge of the input

Parameter Window:



4. Timer Li:

It is used to create flashing (the durations of on and off states can be configured):

Function Li: The flashing cycle starts with an ON state.

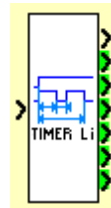
Function L: The flashing cycle starts with an OFF state.

It is 1 input and 1 output function block. Timer BW is represented as follows:

Functional Bar icon:



Wiring Sheet Block:

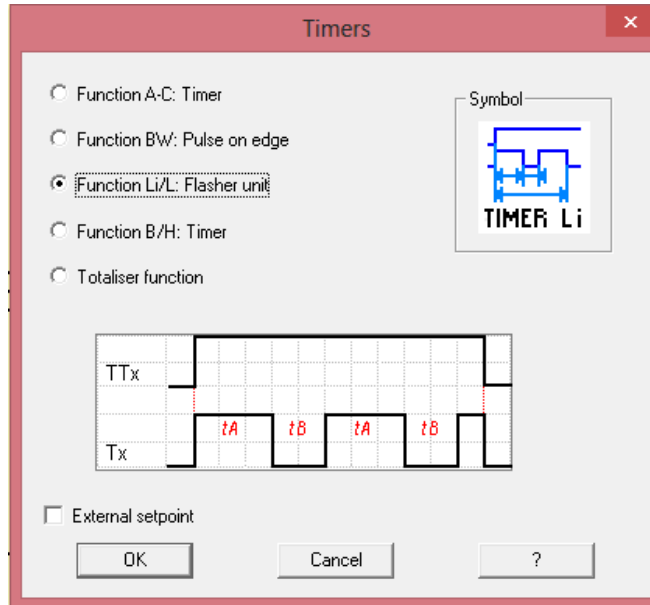


Selection of Timer:

Timer Li Parameters can be selected as follows

The Parameters window is used to:

- a) **Select the time unit of the delays**, these delays can be expressed in seconds, tenths of seconds or number of cycles.
- b) **Select the type of flashing:**
 - i. select **function Li** for the flashing to begin with an **On** state, and
 - ii. select **function L** for the flashing to begin with an **Off** state.



c) Set the On time value

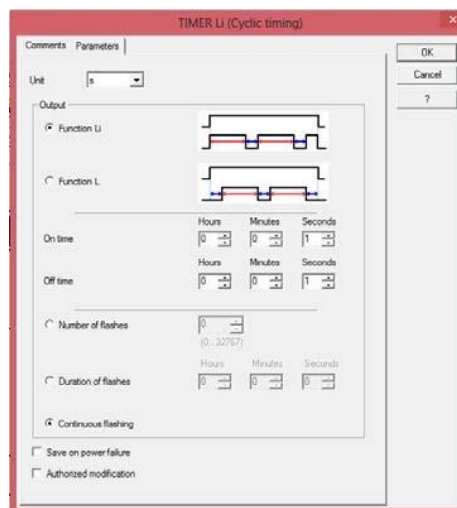
d) Set the Off time value

e) Select the stop mode of the flashes, select:

- i. **Number of flashes** in order to stop after a set number of flashes
- ii. **Duration of flashes** in order to stop after a set time
- iii. **Continuous flashing** so that the flashes do not stop until the Command input is active.

Possibly activate the Save on power failure parameter. It enables the timer to be restarted at the point where it stopped after a power failure.

Parameter Window:



Programmable Logic Controller

Input:

COMMAND: When this input is activated timer starts operating and generates pulses according to settings made in parameter window.

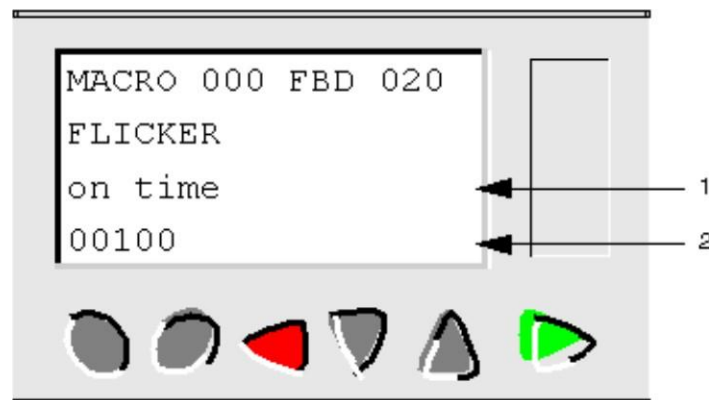
Output:

1. Output
2. ON set point value
3. ON current value
4. OFF set point value
5. OFF current value
6. Number/Duration
7. Current Value Number/Duration of flashes

Parameters can also be controlled from the front panel and the PARAMETERS menu can be set as follows:

- a) The value of the ON time parameter
- b) The value of the OFF time parameter
- c) The value of the counting set point parameter corresponding either to the duration of flashing, or to the number of flashes

Illustration:

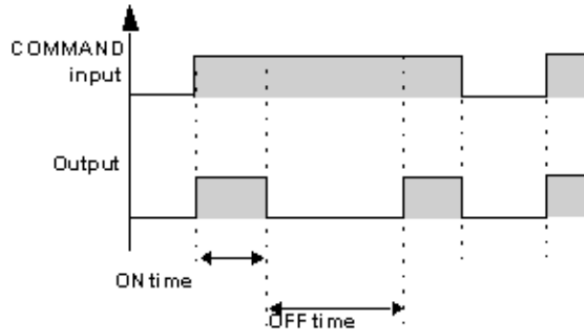


1 Name of the parameter displayed

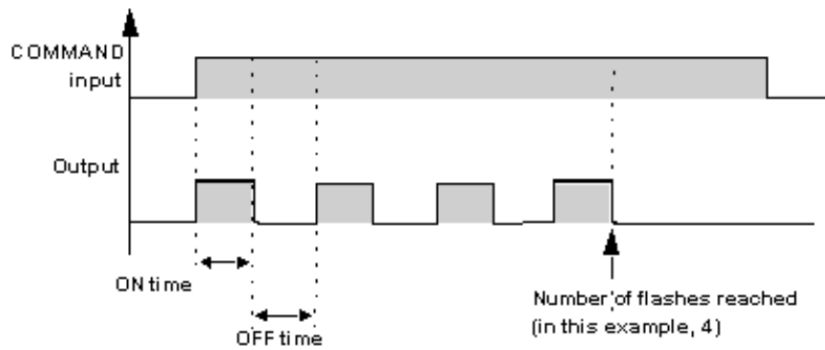
2 Value of the parameter displayed

Timing Diagrams For Timer Li

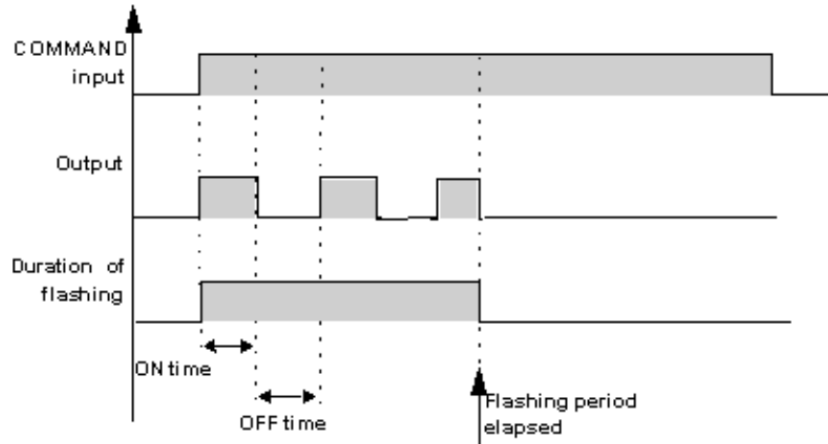
a) Continuous flashing mode:



b) Number of flashes mode:



c) Duration of flashes mode:



4. Timer B/H

Timer B/H creates a pulse on the output of the rising edge of the input.

Function B: Regardless of the duration of the command pulse, the output is active for a duration that has been set.

Function H: The output is inactive at the end of a set time or on the falling edge of the command.

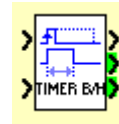
It is a 2 input , 3 output Timer function.

Timer B/H function is represented as follows:

Function Bar icon:

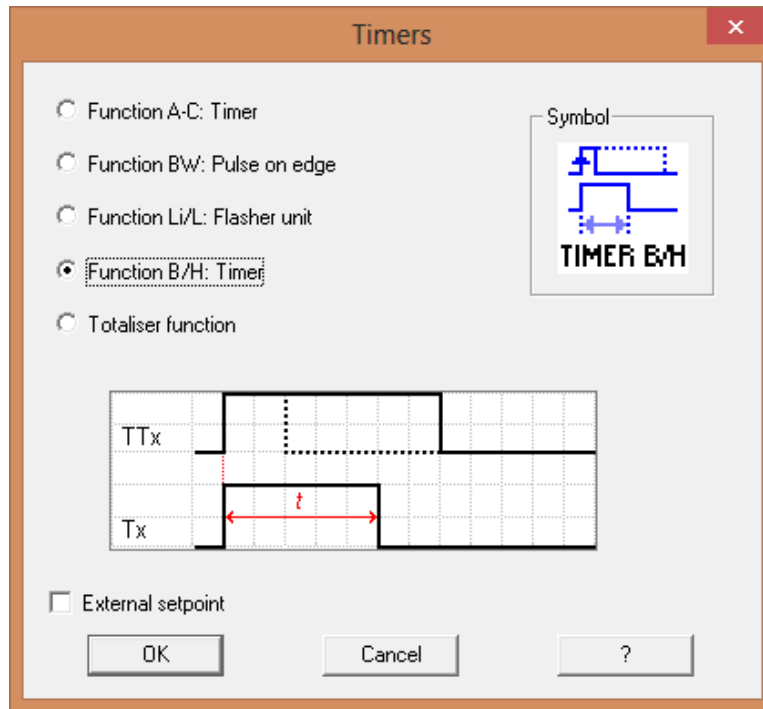


Wiring sheet block:

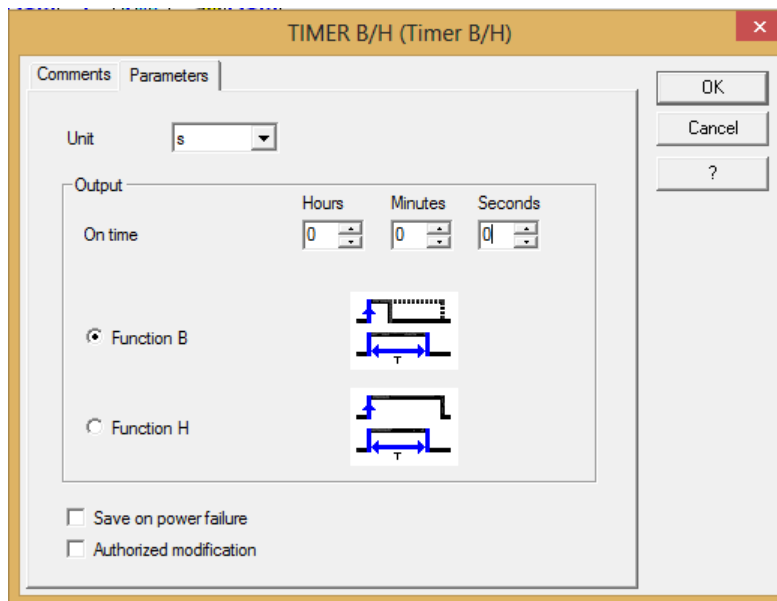


Selection of B/H Timer:

Programmable Logic Controller



Parameter Window:



Timer B/H Parameters

The **Parameters** window is used to:

- a) **Select the time unit of the ON time**; this time can be expressed in seconds, tenths of seconds or number of cycles.

Programmable Logic Controller

b) Set the **On time value**, only if the **External setpoint** box was **left unchecked** when the timer type was selected.

c) **Select the operating mode of the timer,**

- i. select **Function B** so that the output remains active regardless of the duration of the command pulse;
- ii. select **Function H** so that the output switches to the inactive state on the falling edge of the command.

Possibly **activate** the **Save on power failure** parameter. It enables the timer to be restarted at the point where it stopped after a power failure.

Inputs:

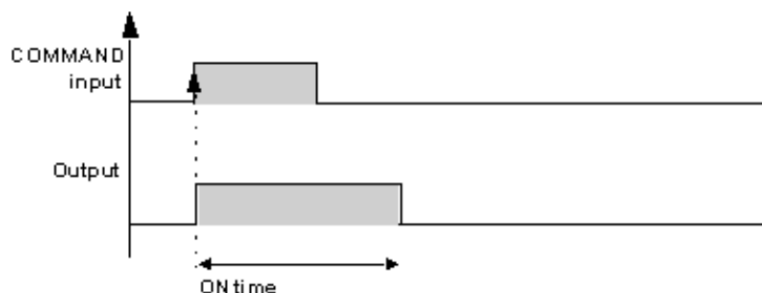
1. **COMMAND:** A positive or negative pulse of input activates the timer and performs according to the set parameters.
2. **RESET:** On activation of this input, the output goes inactive and the timer values set to initial conditions.

Output:

1. Output
2. ON set point value
3. ON Current value

Timing Diagrams for Timer B/H

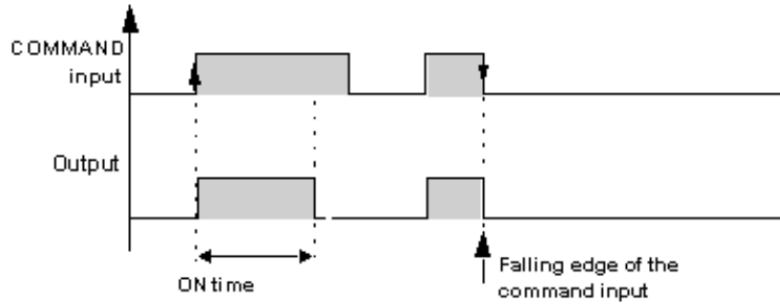
Function B:



Note: Each pulse on the timer **COMMAND** input resets the current value to 0.

Function H:

Programmable Logic Controller



6. TOTALISER Function:

The totaliser creates a pulse on the output when the period during which the input was active reaches (one or more times) a set value.

It is a 2 input, 3 output Timer function.

Timer B/H function is represented as follows:

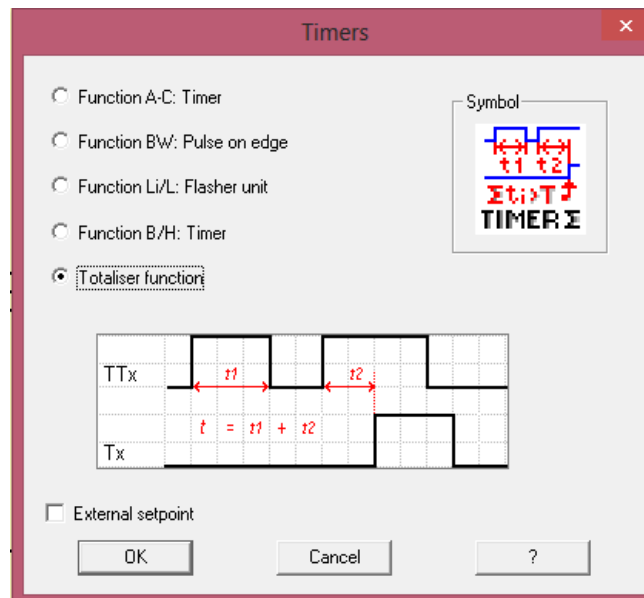
Function Bar icon:



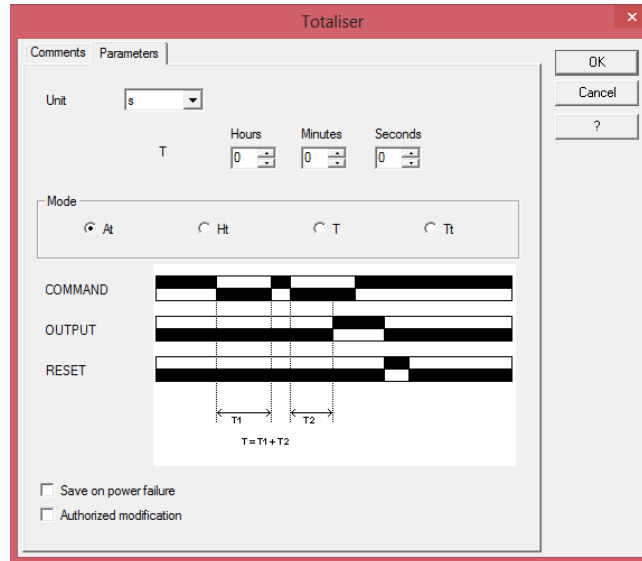
Wiring sheet block:



Selection Window:



Parameter Window:



Totaliser Parameters

The **Parameters** window is used to:

- a) **Select the time unit of the ON time**; this time can be expressed in minutes, seconds, tenths of seconds or number of cycles.
- b) **Set the value T** of the time to be reached, only if the **External setpoint** box was **left unchecked** when the timer type was selected
- c) **Select the totaliser operating mode** , select:
 - i. **At** so that the totaliser output switches to the **active** state when the time spent in the **inactive** state by the input reaches the set time
 - ii. **Ht** so that the totaliser output switches to the **inactive** state when the time spent in the **inactive** state by the input reaches the set time
 - iii. **T** so that the totaliser output switches to the **active** state when the time spent in the **active** state by the input reaches the set time
 - iv. **Tt** so that the totaliser output switches to the **active** state for a set time when a pulse is detected on the input

Possibly **activate** the **Save on power failure** parameter. It enables the timer to be restarted at the point where it stopped after a power failure

Inputs:

1. **COMMAND**: A positive or negative pulse of input activates the totaliser function and the count value of ON time or OFF time is recorded and performs according to the set parameters.

Programmable Logic Controller

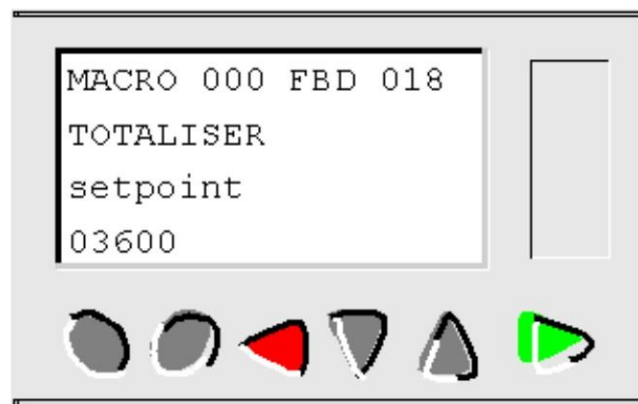
2. RESET: On activation of this input, the output goes inactive and the timer values set to initial conditions.

Output:

1. Output
2. set value
3. Current value

Parameters can also be controlled from the front panel PARAMETERS menu, parameters can be set and modified: The value of the on time i.e; pulse duration.

Illustration:



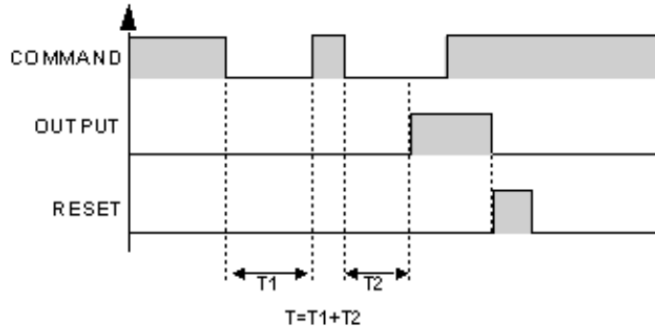
Parameter Modification

To modify the parameters from the controller front panel, check the Modification authorized box in the Parameters window.

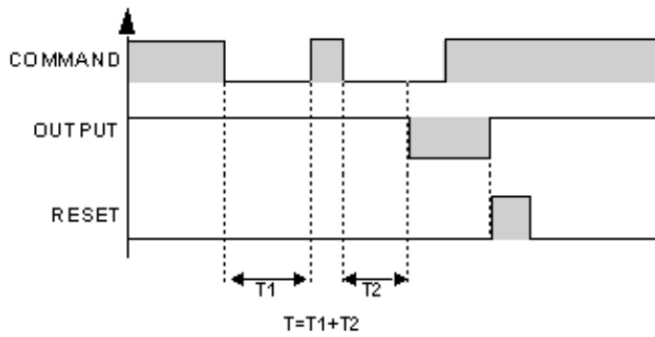
Totaliser Timing Diagrams:

Function At:

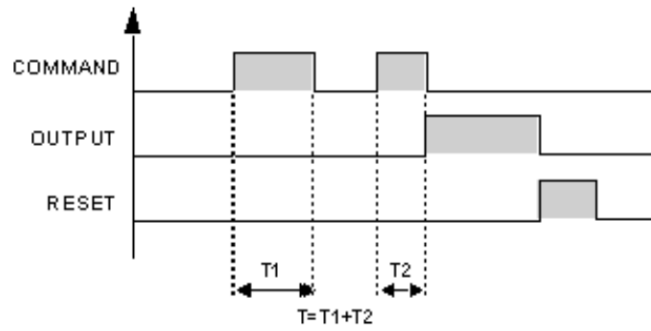
Programmable Logic Controller



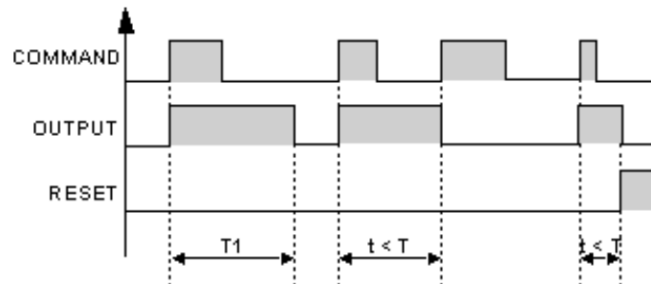
Function Ht:



Function T:



Function Tt



Programmable Logic Controller

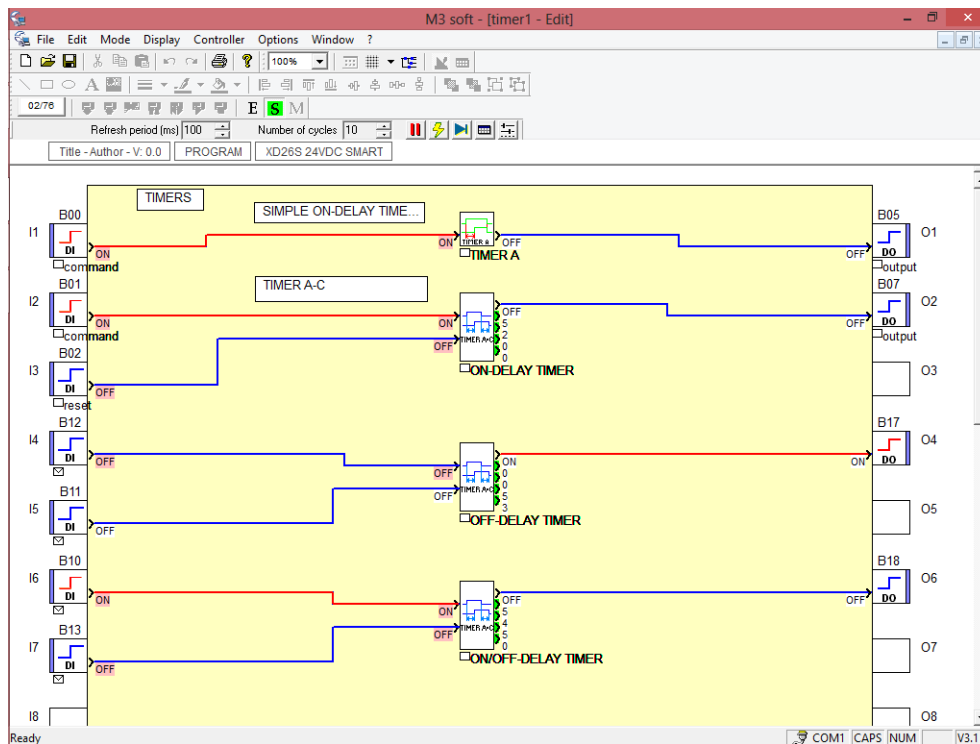
10. A program to study timer functions: Timer A and Timer AC

Program details:

1. Choose controller type and parameter window setting properly for input and output compatibility and validity as per requirement.
2. Make proper connection between blocks, input-output to avoid invalidity and malfunction.
3. Input (7 blocks), output (4 blocks), Timer A, Timer AC (3 No's).

Hardware requirement:

1. Output indication Bulbs (4 No's)
2. Set of single stranded wires
3. Relay card



Programmable Logic Controller

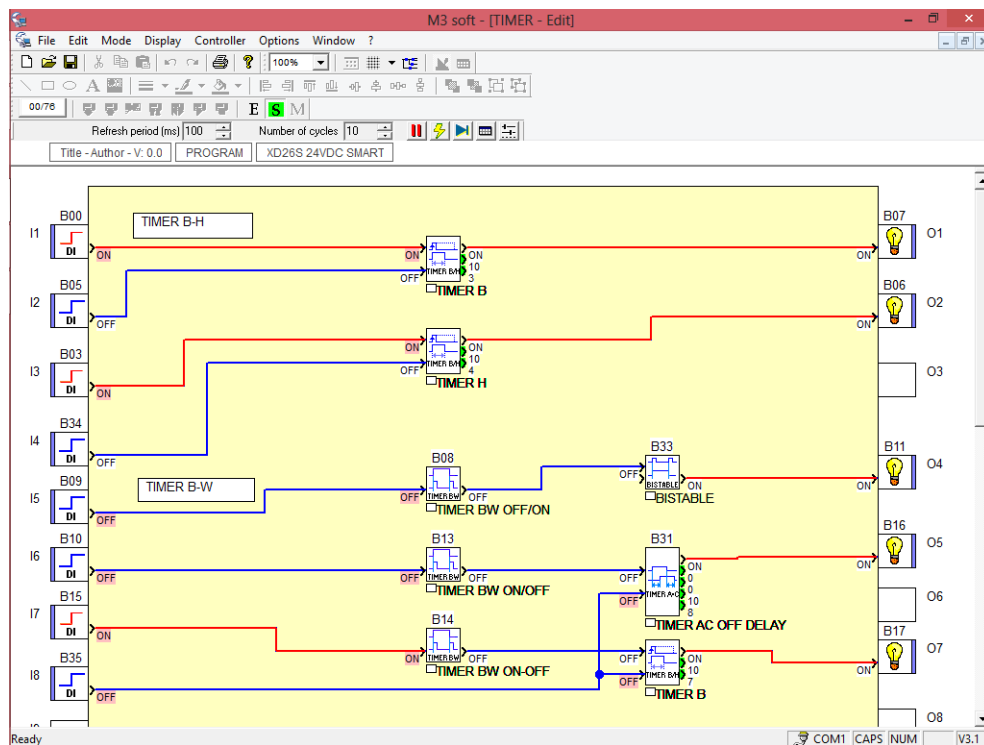
11. A program to study timer functions: Timer BH and Timer BW.

Program details:

1. Choose controller type and parameter window setting properly for input and output compatibility and validity as per requirement.
2. Make proper connection between blocks, input-output to avoid invalidity and malfunction.
3. Input (8 blocks), output (5 blocks), Timer BH (3 No's), Timer BW (3 No's), Timer AC (1No's) and bi-stable

Hardware requirement:

1. Output indication Bulbs (5 No's)
2. Set of single stranded wires
3. Relay card



Programmable Logic Controller

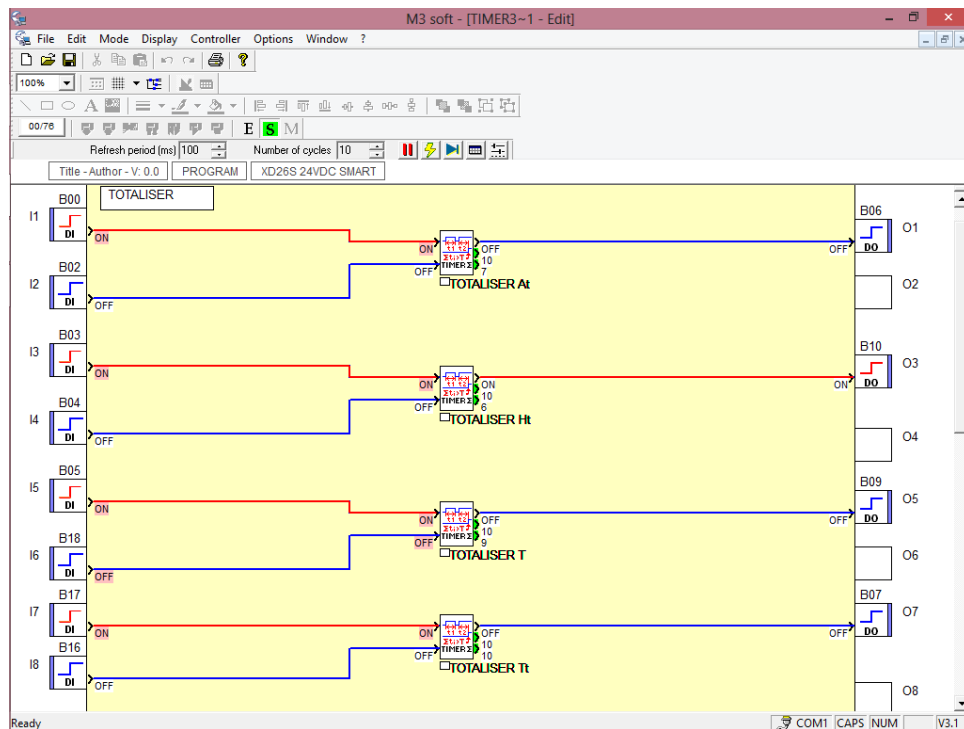
12. A program to study timer functions: TOTALISER At, Ht, T and Tt.

Program details:

1. Choose controller type and parameter window setting properly for input and output compatibility and validity as per requirement.
2. Make proper connection between blocks, input-output to avoid invalidity and malfunction.
3. Input (8 blocks), output (4 blocks), Timer TOTALISER (4 No's of type At, Ht, T and Tt).

Hardware requirement:

1. Output indication Bulbs (4 No's)
2. Set of single stranded wires
3. Relay card



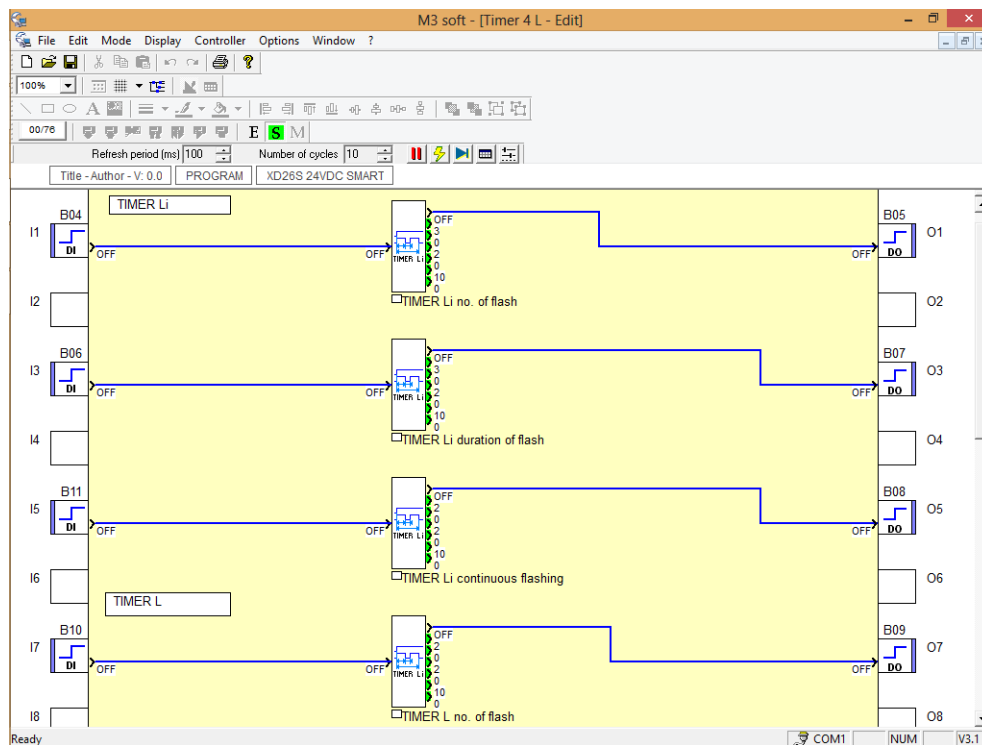
13. A program to study timer functions: Timer Li and Timer L.

Program details:

1. Choose controller type and parameter window setting properly for input and output compatibility and validity as per requirement.
2. Make proper connection between blocks, input-output to avoid invalidity and malfunction.
3. Input (4 blocks), output (4 blocks), Timer Li (3 No's, type of flashing as no of flash, duration of flash, continuous flashing) and Timer L

Hardware requirement:

1. Output indication Bulbs (4 No's)
2. Set of single stranded wires
3. Relay card



4.4 Study of PLC counters functions

1. PRESET COUNT Up/Down Counter

The PRESET COUNT up/down counter function is used to up-count from 0 to the preset value, or to down-count from this value to 0. This function is accessible from the FBD function bar. It is a 3 input, 4 output function block.

Several functions are available:

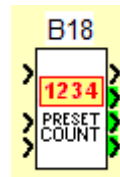
- Up-counting and forcing the counter to 0 on initialization,
- Up-counting and forcing the counter to 0 on initialization and when the count value has been reached,
- Down-counting and forcing the counter to the preset value on initialization,
- Down-counting and forcing the counter to the preset value on initialization and when 0 has been reached.

It is represented as follows

Functional Bar icon:



Wiring Sheet Block:



Parameter Window:

The image shows a parameter window titled 'PRESET COUNT (Preset up/down counter)'. The window has a red border and a white background. It contains the following settings:

- Counting:** Two radio buttons are present. The first, 'Upcounting to the preset value', is selected. The second, 'Downcounting from the preset value', is unselected.
- Preset:** A numeric input field contains the value '0'. To the right of the field, the range '(0...32767)' is displayed.
- Cycle:** Two radio buttons are present. The first, 'Single', is selected. The second, 'Repetitive', is unselected.
- Duration of pulse:** A numeric input field contains the value '1'. To the right of the field, the unit 'x 100ms' is displayed. Below the field, the range '(1...32767)' is displayed.
- Checkboxes:** Two checkboxes are located at the bottom of the window. The first is 'Save on power failure' and the second is 'Authorized modification'. Both are currently unchecked.

On the right side of the window, there are three buttons: 'OK', 'Cancel', and a button with a question mark '?'.

Programmable Logic Controller

Inputs/Outputs

Input:

The up/down counter uses:

- i. A discrete UP-COUNT input,
- ii. A discrete DOWN-COUNT input,
- iii. A discrete INITIALIZATION input.

Output:

The up/down counter provides:

- i. A discrete OUTPUT,
- ii. The preset value
- iii. The current counter value
- iv. The output timer value

These integer values are displayed in Simulation and Monitoring mode.

Parameters setting and modification:

From the Parameters window, you can adjust:

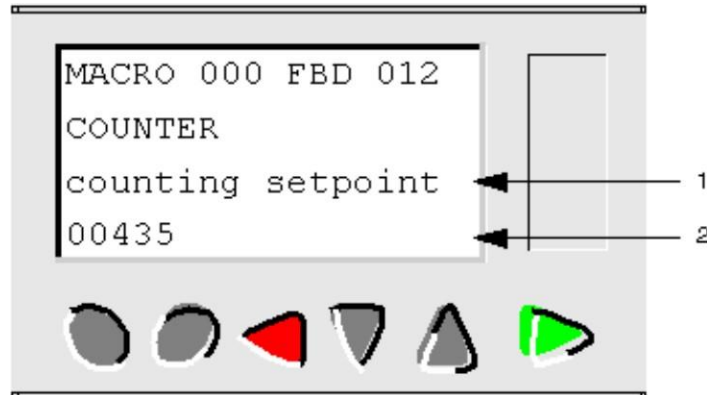
- a) The Upcounting to the preset value or Downcounting from the preset value,
- b) The Preset or Setpoint value
- c) Cycle type:
 - i. The Single cycle for initializing the counter only on initialization,
 - ii. The Repetitive cycle for initializing the counter on initialization, and when the current count value reaches 0 or the preset value. which the output is Active.

The Save on power break parameter, if selected, enables the current value of the timer to be retrieved following a power failure.

From the front panel using the PARAMETER menu, setting and modification can be done.

- a) The Preset or Counting setpoint value,
- b) The DURATION OF PULSE value (for a repetitive cycle).

Illustration: Parameters of the counter



1. Name of the parameter displayed
2. Value of the parameter displayed

Parameter Modification

To modify the parameters from the front panel of the controller, check the Authorized modification box of the Parameters window.

2. UP/DOWN COUNT Up/Down Counter:

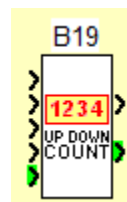
The UP/DOWN COUNT function is used to up-count or down-count from a preset value resulting from a calculation outside the function. This function is accessible from the FBD function bar. It is 5 input, 2 output counter function.

It is represented as follows:

Functional Bar icon:



Wiring Sheet Block:



A level 1 on the PRESET FORCING input is used to change the counter with the value available at the PRESET input. The PRESET input can be connected to the NUM constant, to an analog input, or to any other kind of output on a function block which delivers an INTEGER-type value.

1. A rising edge on the: .
 - a) UPCOUNTING: increments the counter.
 - b) DOWNCOUNTING: decrements the counter.
2. State of the OUTPUT:

Programmable Logic Controller

1: when the counting number has been reached, the OUTPUT switches to 1 and remains so for as long as the counting number is greater than or equal to the PRESET value,

0: if the transitions on the DOWNCOUNT input switch the counting number back to a value less than the PRESET value.

Activation of the RESET or PRESET FORCING inputs enable the counter to be relaunched.

As long as the RESET input is at 1, the OUTPUT remains at 0. When the RESET input switches to 0, the up/down count operation is relaunched from zero.

Inputs/Outputs:

Inputs:

The up/down counter uses the following inputs:

- i. UPCOUNTING, Discrete type,
- ii. DOWNCOUNT, Discrete type,
- iii. RESET, Discrete type,
- iv. PRESET FORCING, Discrete type,
- v. PRESET, integer type.

Output:

The up/down counter provides the following outputs:

- i. OUTPUT, Discrete type,
- ii. CURRENT VALUE, integer type, between -32768 and 32767.

Parameters:

The Save on power break parameter, if selected, enables the current value of the timer to be retrieved following a power failure.

Programmable Logic Controller

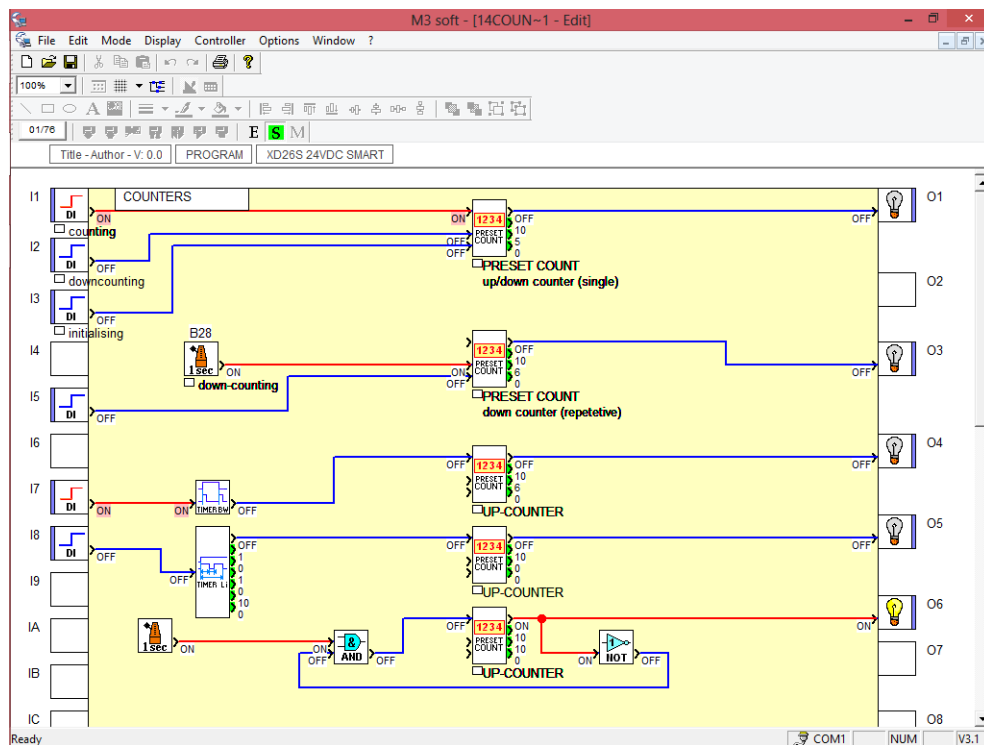
14. A program to study counter functions: PRESET COUNTER.

Program details:

1. Choose controller type and parameter window setting properly for input and output compatibility and validity as per requirement.
2. Make proper connection between blocks, input-output to avoid invalidity and malfunction.
3. Input (6 blocks), output (5 blocks), preset counter (5 No's), 1sec clock, Timer BW, Timer Li, AND and NOT gates.

Hardware requirement:

1. Output indication Bulbs (5 No's)
2. Set of single stranded wires
3. Relay card



Programmable Logic Controller

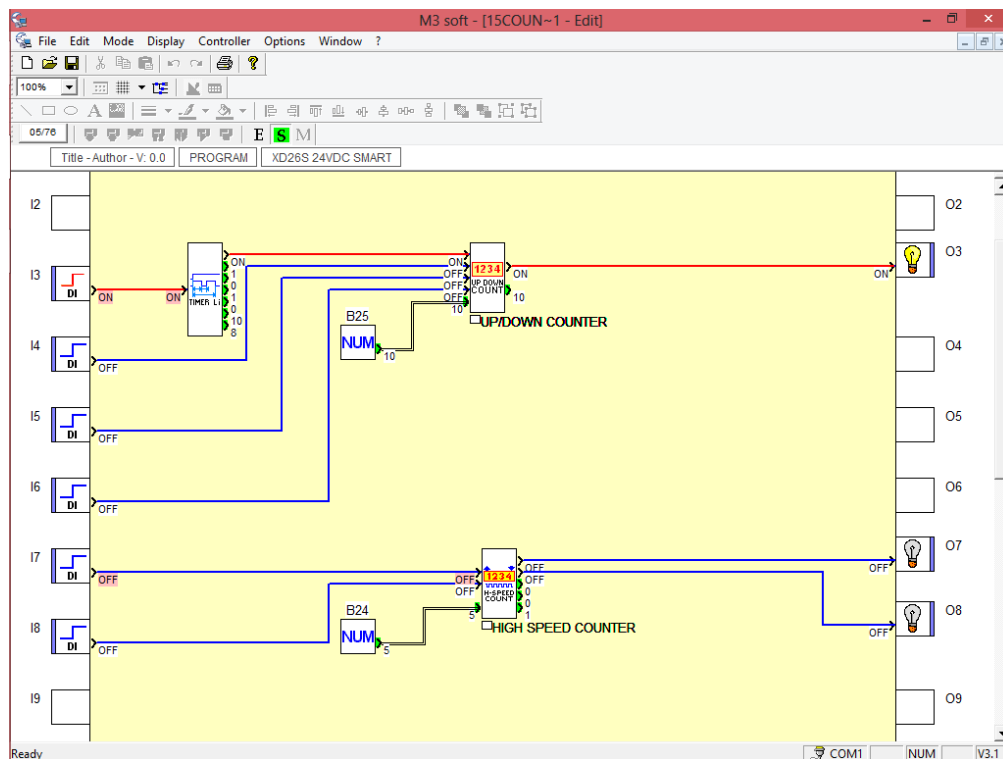
15. A program to study counter functions: UP-DOWN COUNTER.

Program details:

1. Choose controller type and parameter window setting properly for input and output compatibility and validity as per requirement.
2. Make proper connection between blocks, input-output to avoid invalidity and malfunction.
3. Input (6 blocks), output (3 blocks), UP-DOWN Counter (2 No's), Timer Li and NUM (2 No's).

Hardware requirement:

1. Output indication Bulbs (3 No's)
2. Set of single stranded wires
3. Relay card



Programmable Logic Controller

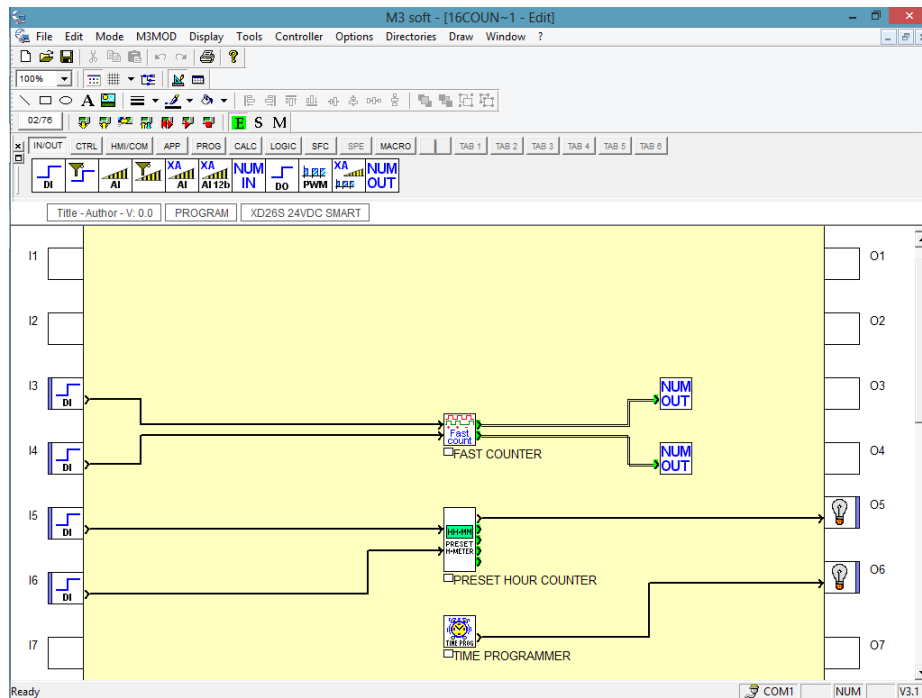
16. A program to study counter functions: Fast counter, Preset Hour counter, Time programmer.

Program details:

1. Choose controller type and parameter window setting properly for input and output compatibility and validity as per requirement.
2. Make proper connection between blocks, input-output to avoid invalidity and malfunction.
3. Input (4 blocks), output (2 blocks), Fast counter, Preset Hour counter, Time programmer NUM OUT (2 No's).

Hardware requirement:

1. Output indication Bulbs (2 No's)
2. Set of single stranded wires
3. Relay card



4.5 Study of PLC Arithmetic functions

ADD/SUB:

The ADD-SUB (Addition or Subtraction) function is used to perform simple operations on integers

ADDITION: CALCULATION OUTPUT = INPUT 1 + INPUT 2

SUBTRACTION: CALCULATION OUTPUT = INPUT 1 - INPUT 2

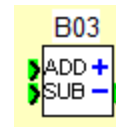
This function has 2 inputs and 1 output. The function is accessible from the FBD function bar.

ADD/SUB function is represented as follows:

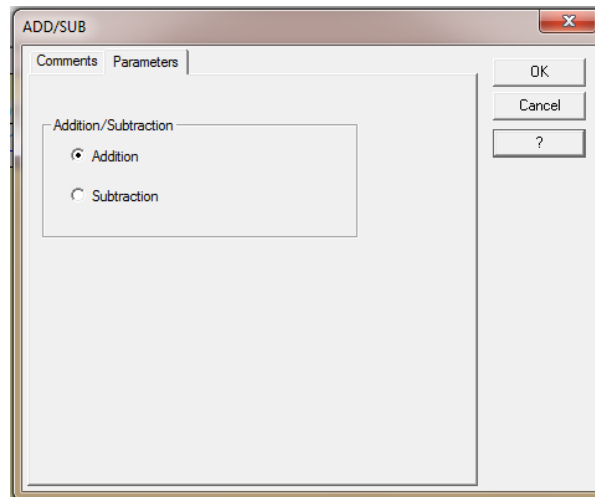
Function Bar icon:



Wiring sheet block:



Parameter window:



Input/Output:

Input:

1. INPUT 1: First input value in the formula.
2. INPUT 2: Second input value in the formula

Output:

CALCULATION OUTPUT: This is the calculation formula output value.

ADD-SUB Function:

The ADD-SUB Addition and/or Subtraction function enables simple operations to be carried out on integers:

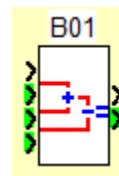
1. Addition,
2. Subtraction.

The ADD-SUM function is accessible from the FBD function bar(CAL). This function has 4 inputs and 2 output. ADD-SUB function is represented as follows:

Function Bar icon:



Wiring sheet block:



Calculation formula:

```
CALCULATION OUTPUT = INPUT1+INPUT2-INPUT3
```

Inputs/Outputs:

Inputs:

1. INPUT 1: first input value of the formula (integer),
2. INPUT 2: second input value of the formula (integer),
3. INPUT 3: third input value of the formula (integer).

Note that If the INPUTS are not connected, they are set to 0.

4. ERROR PROPAGATION: this input, whose type is Discrete, is used to propagate errors (or saturations) from calculation functions (ADD-SUB or MUL-DIV) carried out upstream.

Note that

a) If ERROR PROPAGATION is at 1, then the operations are not performed and the ERROR/OVERRUN output is set to 1.

Programmable Logic Controller

b) If ERROR PROPAGATION is not connected, it is set to 0.

Outputs:

1. **CALCULATION OUTPUT:** this is the value of the calculation formula output (integer).

2. **ERROR/OVERRUN:** this output, whose type is Discrete, indicates any presence of saturation errors). This output is activated in the following cases. The consequence of the operations is a result that is not included in the interval $[-32768, +32767]$, The **ERROR PROPAGATION** input is active.

For an operation as one functioning out of two the following options can be used:

For addition: simply do not use the **INPUT 3** input.

For subtraction: simply do not use one of the **INPUT 1** or **2** inputs

MUL-DIV Arithmetic Function

Description

The **MUL-DIV** Multiplication and/or Division function enables simple operations to be carried out on integers:

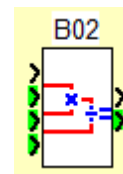
1. Multiplication,
2. Division.

The **MUL-DIV** function is accessible from the FBD function bar(CAL). This function has 4 inputs and 2 output. **MUL-DIV** function is represented as follows:

Function Bar icon:



Wiring sheet block:



Calculation formula:

```
CALCULATION OUTPUT = INPUT1*INPUT2/INPUT3
```

Inputs/Outputs

Inputs:

1. **INPUT 1:** first input value of the formula (integer).
2. **INPUT 2:** second input value of the formula (integer)

Programmable Logic Controller

3. INPUT 3: third input value of the formula (integer)

Note: If the INPUTS are not connected, they are set to 1.

4. ERROR PROPAGATION: this input, whose type is Discrete, is used to propagate errors (or saturations) from calculation functions (ADD-SUB or MUL-DIV) carried out upstream.

Note that

- i. If ERROR PROPAGATION is at 1, then the operations are not performed and the ERROR/OVERRUN output is set to 1.
- ii. If ERROR PROPAGATION is not connected, it is set to 0.

Outputs:

1. CALCULATION OUTPUT: this is the value of the calculation formula output (integer).

2. ERROR/OVERRUN: this output, whose type is Discrete, indicates any presence of saturation errors). This output is activated in the following cases:

- a) The consequence of the operations is a result that is not included in the interval $[-32768, +32767]$,
- b) The ERROR PROPAGATION input is active,
- c) The INPUT 3 equals 0.

For an operation as one functioning out of two the following options can be used:

For **multiplication**: simply do not use the INPUT 3 input.

For **division**: simply do not use one of the INPUT 1 or 2 inputs

GAIN Function

The Gain function enables analog values to be converted by changing the scale and offset.

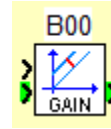
Gain calculation formula:

$$\text{CALCULATION OUTPUT} = A / B * \text{CALCULATION INPUT} + C$$

The gain function is accessible from the FBD function bar. This function has 2 inputs and 1 output. The function is accessible from the FBD function bar (CAL). ADD/SUB function is represented as follows:

Function Bar icon:

Wiring sheet block:



Inputs/Outputs:

Inputs:

1. ENABLE FUNCTION: Gain function input is command of the type Discrete. The state of this input determines operation of the block: if the ENABLE FUNCTION input is inactive, the CALCULATION OUTPUT retains the last calculated value.

2. CALCULATION INPUT: value of the analog input connected to the gain function.
This is an integer between -32768 and 32767.

Output:

1. CALCULATION OUTPUT:

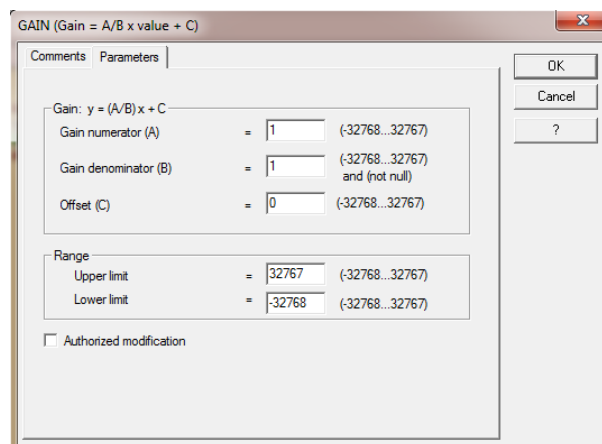
This value depends upon the state of the ENABLE FUNCTION input.

If the ENABLE FUNCTION input is:

- i. inactive, then the CALCULATION OUTPUT equals the last calculated value, when the ENABLE FUNCTION input is inactive,
- ii. active, then the CALCULATION OUTPUT is equal to the result of the gain calculation formula.

Note that if the ENABLE FUNCTION input is not connected, then it is considered to be active.

Parameters Window:



Parameters

Programmable Logic Controller

In the programming software

From the Parameters window, you can adjust:

a) A/B which corresponds to the gain applied by the function with:

A: being a numerator (from -32768 to 32767),

B: denominator (from -32768 to -1 and from 1 to 32767)

b) C which is the offset applied by the function, and is an integer between -32768 and 32767.

In addition, it is possible to define an operating range by setting limits for the function output:

Lower limit: integers between -32768 and 32767,

Upper limit: integers between -32768 and 32767.

Parameters can also be controlled from the front panel PARAMETERS menu, parameters can be set and modified from the PARAMETER menu, the following parameter values can set:

A: gain numerator

B: gain denominator (value 0 prohibited),

C: offset,

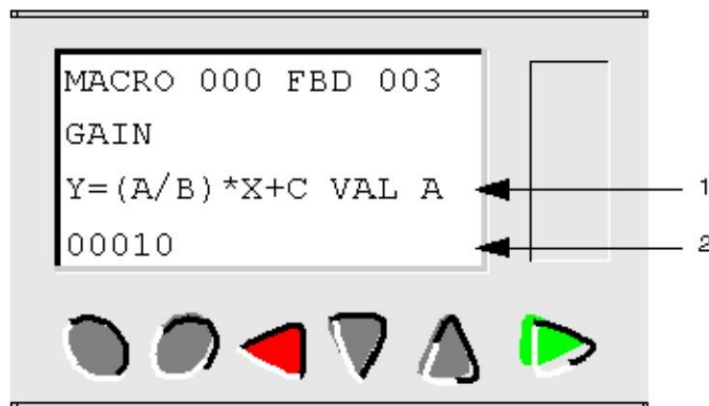
upper limit,

lower limit.

Parameter Modification:

To modify the parameters from the front panel of the controller, check the Authorized modification box of the Parameters window.

Illustration:



1 Name of the parameter displayed

2 Value of the parameter displayed

SQUARE ROOT

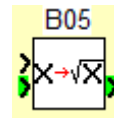
This function is used to calculate the square root of the number present as an input with accuracy to two decimal points

The SQUARE ROOT function is accessible from the FBD function bar. This function has 2 inputs and 1 output. The function is accessible from the FBD function bar (CAL). SQUARE ROOT function is represented as follows:

Function Bar icon:



Wiring sheet block:



Input/Output:

Input:

1. Validation: Function validation input. Until this input is activated, the function remains inert. Validation is active implicitly if it has not been connected
2. Calculation input: The value must be between 0 and 32767

Output:

1. Calculation output: The result is presented in the format "Root" x 100 and the calculations is accurate to 0.01.

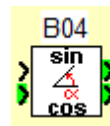
SIN-COS:

This function is used to calculate the cos and sin of an angle between 0° and 90°. The function calculates the cos and the sin to the nearest 0.0001 by rounding up or down as appropriate. The SIN-COS function is accessible from the FBD function bar. This function has 2 inputs and 1 output. The function is accessible from the FBD function bar (CAL). SIN-COS function is represented as follows:

Function Bar icon:



Wiring sheet block:



Input/Output:

Input:

Validation: Function validation input. Until this input is activated, the function remains inert. Validation is active implicitly if it has not been connected.

Programmable Logic Controller

Angle: Represents the angle in degrees. Its value must be between 0 and 900 for an angle between 0° and 90° .

Output:

Sin: Result of ("Angle" sin) x 10000

Cos: Result of ("Angle" cos) x 10000

Programmable Logic Controller

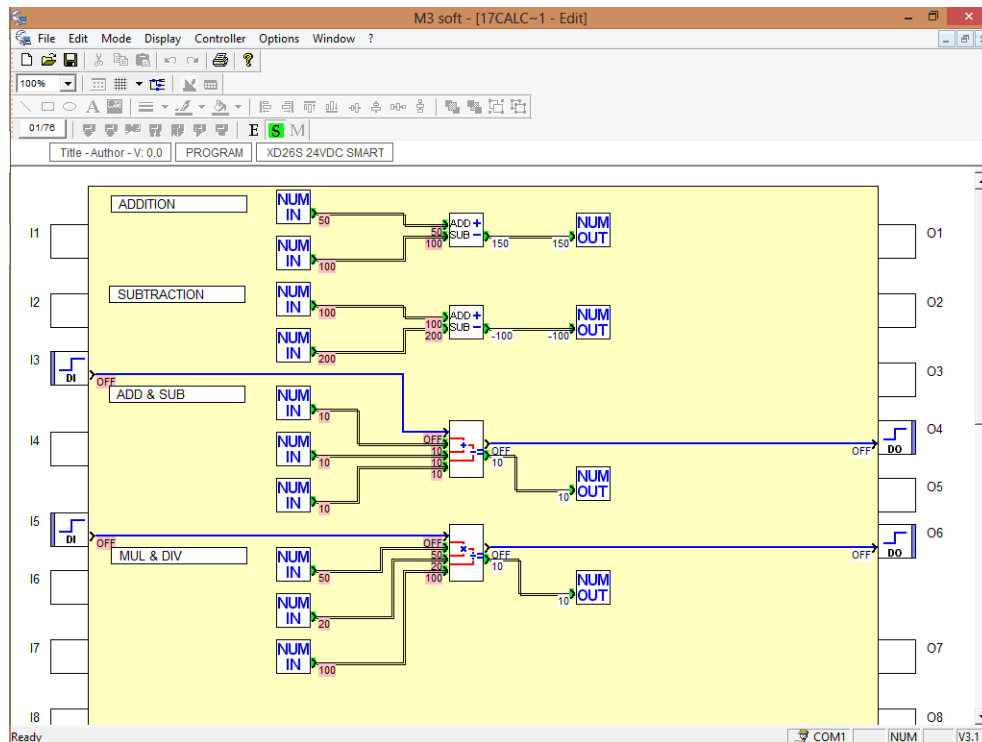
17. A program to study arithmetic functions: ADD/SUB and MUL/DIV.

Program details:

1. Choose controller type and parameter window setting properly for input and output compatibility and validity as per requirement.
2. Make proper connection between blocks, input-output to avoid invalidity and malfunction.
3. Input (2 blocks), output (2 blocks), NUM IN (10 No's), NUM OUT (4 No's), ADD/SUB (3No's) and MUL/DIV

Hardware requirement:

1. Output indication Bulbs (2 No's)
2. Set of single stranded wires
3. Relay card



Programmable Logic Controller

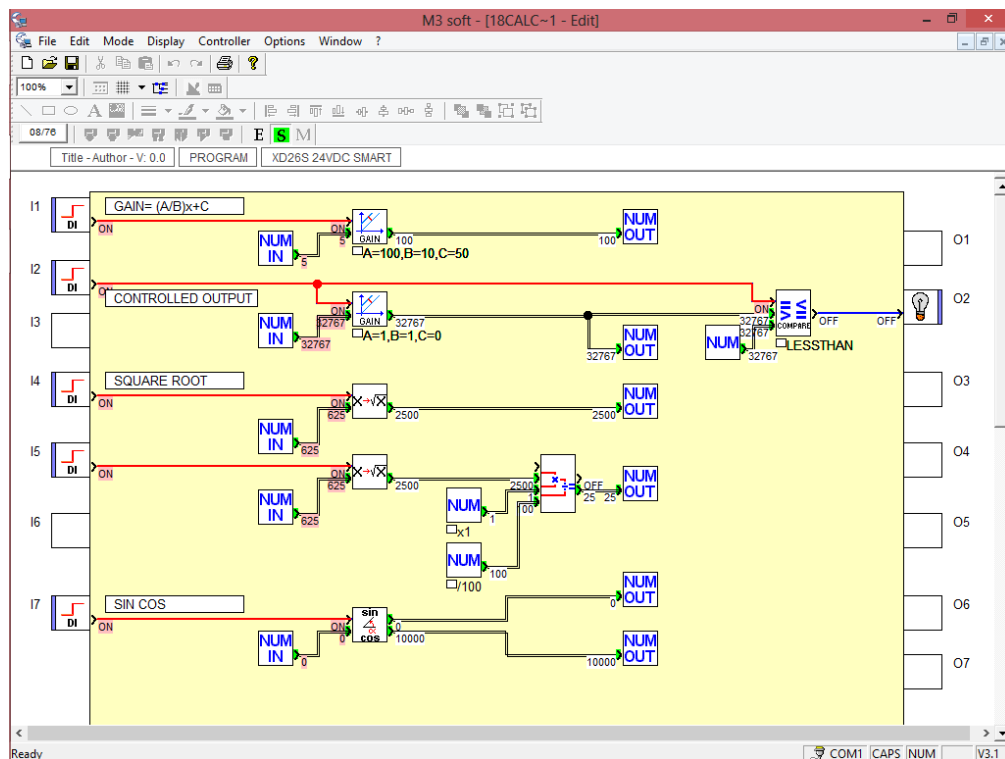
18. A program to study arithmetic functions: GAIN, SQUARE ROOT and SIN-COS

Program details:

1. Choose controller type and parameter window setting properly for input and output compatibility and validity as per requirement.
2. Make proper connection between blocks, input-output to avoid invalidity and malfunction.
3. Input (5 blocks), output (1 blocks), NUM IN (5 No's), NUM OUT (6 No's), NUM (3No's) GAIN (2 No's), SQUARE ROOT (2 No's), SIN-COS, MUL/DIV and COMPARE function

Hardware requirement:

1. Output indication Bulb (1 No's)
2. Set of single stranded wires
3. Relay card



4.6 Study of Number Comparison functions

COMPARE Function for Comparing Two Analog Values

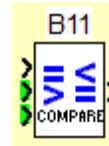
The COMPARE function is used to compare two analog values.. This function uses 3 input and provides 1 discrete-type OUTPUT. The function is accessible from the FBD function bar

COMPARE function is represented as follows:

Function Bar icon:



Wiring sheet block:



Inputs/Outputs:

This function uses 3 input and provides a discrete-type OUTPUT.

Input:

1. an ENABLE FUNCTION Discrete-type input.
2. a VALUE 1 Integer-type input,
3. a VALUE 2 Integer-type input.

Note that if the VALUE 1 or VALUE 2 input is not connected, the value is set to 0.

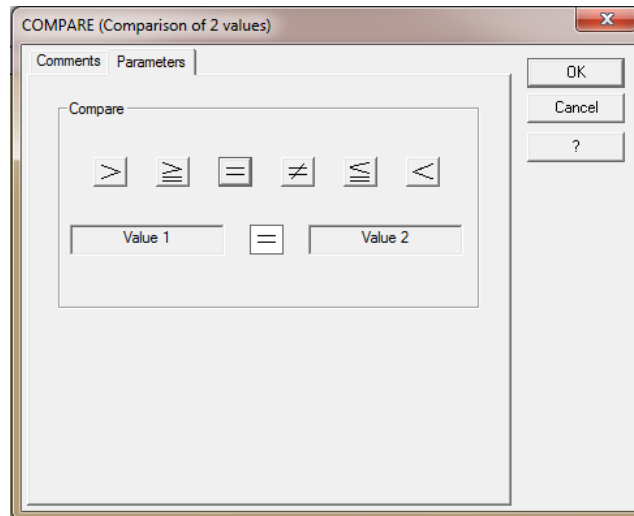
Output:

The output is active if the result of the comparison between VALUE 1 and VALUE 2 is true and if the ENABLE FUNCTION input is active or not connected.

The output does not change state if the ENABLE FUNCTION input changes from Active to Inactive state.

Parameters Window:

Programmable Logic Controller



The comparison operators that can be chosen from the **Parameters** window are as follows:

Symbol	Description
>	Greater than.
≥	Greater than or equal to.
=	Equal to.
≠	Different(not equal).
≤	Less than or equal to.
<	Less than.

COMP IN ZONE Comparison

The COMP IN ZONE comparison function is used to compare one value between two set points (the MIN and MAX values of the zone).The function is accessible from the FBD function bar. It is a 4 input, 1 output comparison function.

COMP IN ZONE function is represented as follows:

Function Bar icon:

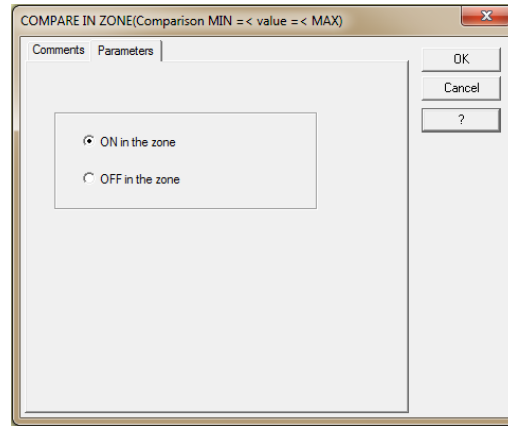


Wiring sheet block:



Parameters Window:

Programmable Logic Controller



From the **Parameters** window, you can select

1. the state of the output according to the result of the comparison:

- a) **ON in the zone:** the output will be active if the input value is between the two setpoints (MIN and MAX),
- b) **OFF in the zone:** the output will be inactive if the input value is between the two setpoints (MIN and MAX).

Note that, If MIN value is greater than MAX value, then for:

- a) **ON in the zone:** the output always remains inactive,
- b) **OFF in the zone:** the output always remains active.

Inputs/Outputs: The comparison function comprises of 4 inputs and 1 output.

Inputs:

1. A discrete ENABLE FUNCTION input; this input is Active if it is not connected,
2. A VALUE TO COMPARE input whose type is Integer,
3. A MIN VALUE input, whose type is Integer,
4. A MAX VALUE input, whose type is Integer,

Output:

1. A discrete OUTPUT.
 - a) The OUTPUT indicates the result of the comparison when the ENABLE FUNCTION input is active.
 - b) The OUTPUT does not change state when the ENABLE FUNCTION input is inactive.

MULTI COMPARE:

Programmable Logic Controller

This function is used to activate the output corresponding to the value present on the "Value" input. The comparison value (Value N) can be configured. It must be between 0 and 32760. This function uses 2 inputs and 8 outputs. The function is accessible from the FBD function bar. MULTI COMPARE function is represented as follows:

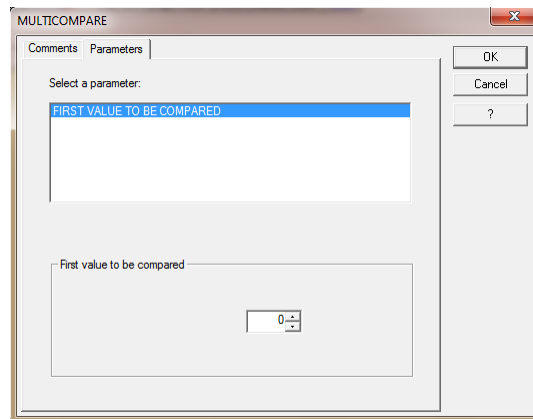
Function Bar icon:



Wiring sheet block:



Parameter Window:



Input:

1. VALIDATION: Function validation input. Until this input is activated, the function remains inert. Validation is active implicitly if it has not been connected
2. VALUE: Value to be compared.

Output:

1. VALUE N: Output ON if Value = Value N.
2. VALUE N + 1: Output ON if Value = Value N + 1
3. VALUE N + 2: Output ON if Value = Value N + 2
4. VALUE N + 3: Output ON if Value = Value N + 3

Programmable Logic Controller

5. VALUE N + 4: Output ON if Value = Value N + 4
6. VALUE N + 5: Output ON if Value = Value N + 5
7. VALUE N + 6: Output ON if Value = Value N + 6
8. VALUE N + 7: Output ON if Value = Value N + 7

Programmable Logic Controller

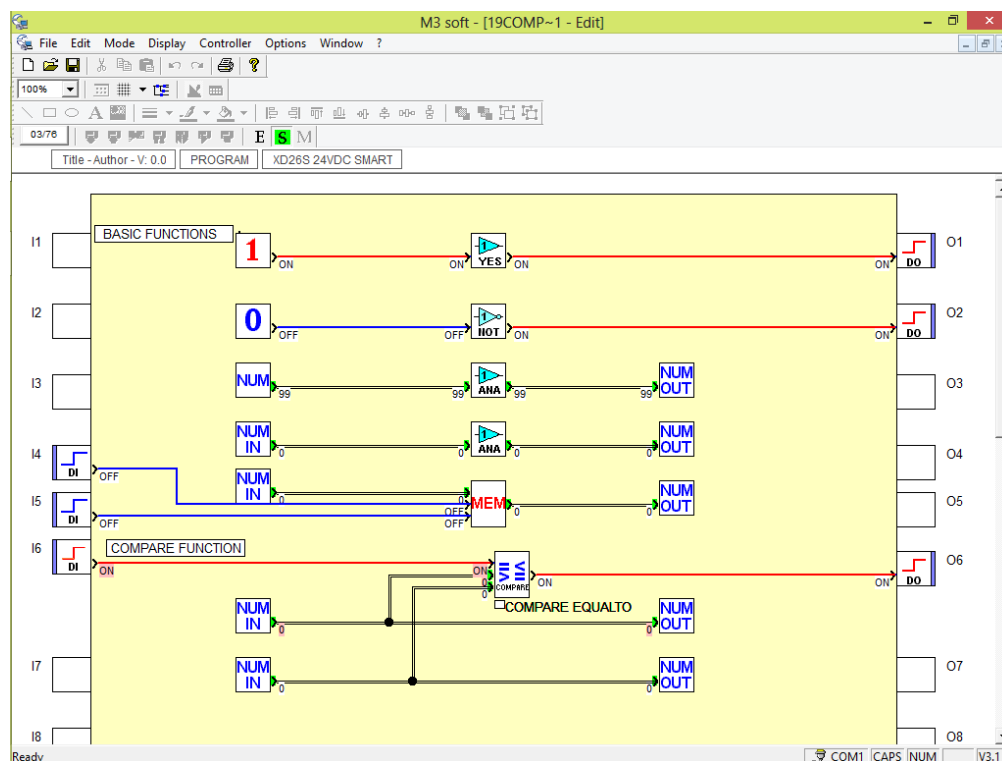
19. A program to study compare functions: COMPARE

Program details:

1. Choose controller type and parameter window setting properly for input and output compatibility and validity as per requirement.
2. Make proper connection between blocks, input-output to avoid invalidity and malfunction.
3. Input (3 blocks), output (3 blocks), NUM IN (4 No's), NUM OUT (5 No's), NUM (1No's), constant 1 and 0, ANA (2 No's), YES, NOT, MEM and COMPARE function

Hardware requirement:

1. Output indication Bulbs (3 No's)
2. Set of single stranded wires
3. Relay card



Programmable Logic Controller

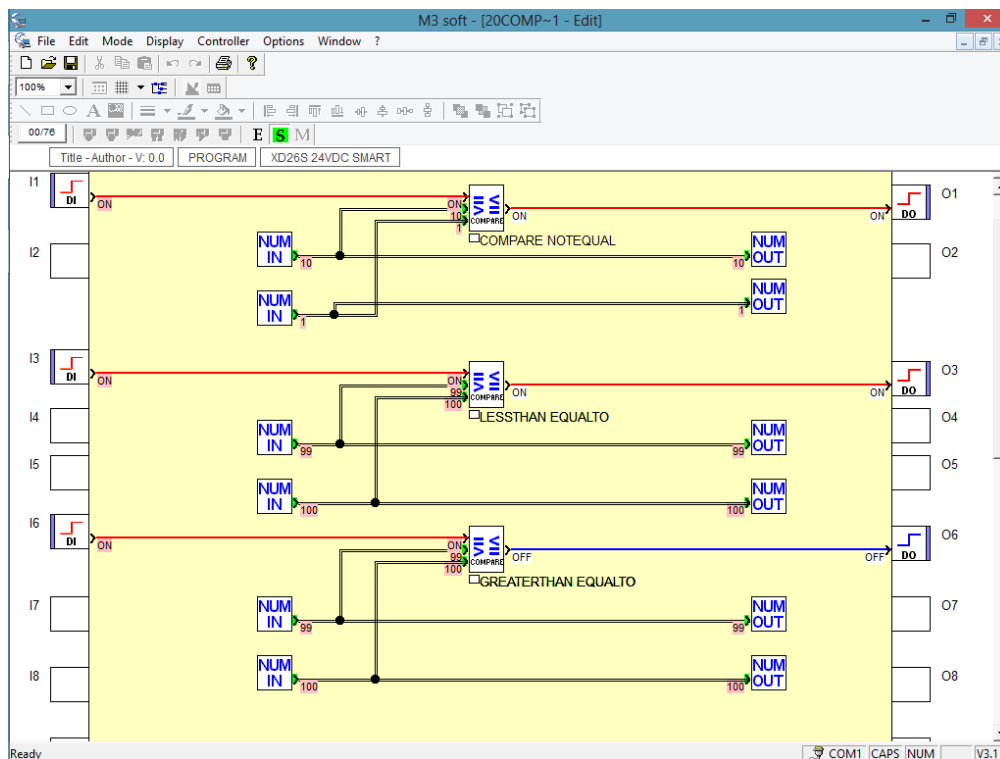
20. A program to study compare functions: COMPARE

Program details:

1. Choose controller type and parameter window setting properly for input and output compatibility and validity as per requirement.
2. Make proper connection between blocks, input-output to avoid invalidity and malfunction.
3. Input (3 blocks), output (3 blocks), NUM IN (6 No's), NUM OUT (6 No's) and COMPARE function (3 No's)

Hardware requirement:

1. Output indication Bulbs (3 No's)
2. Set of single stranded wires
3. Relay card



Programmable Logic Controller

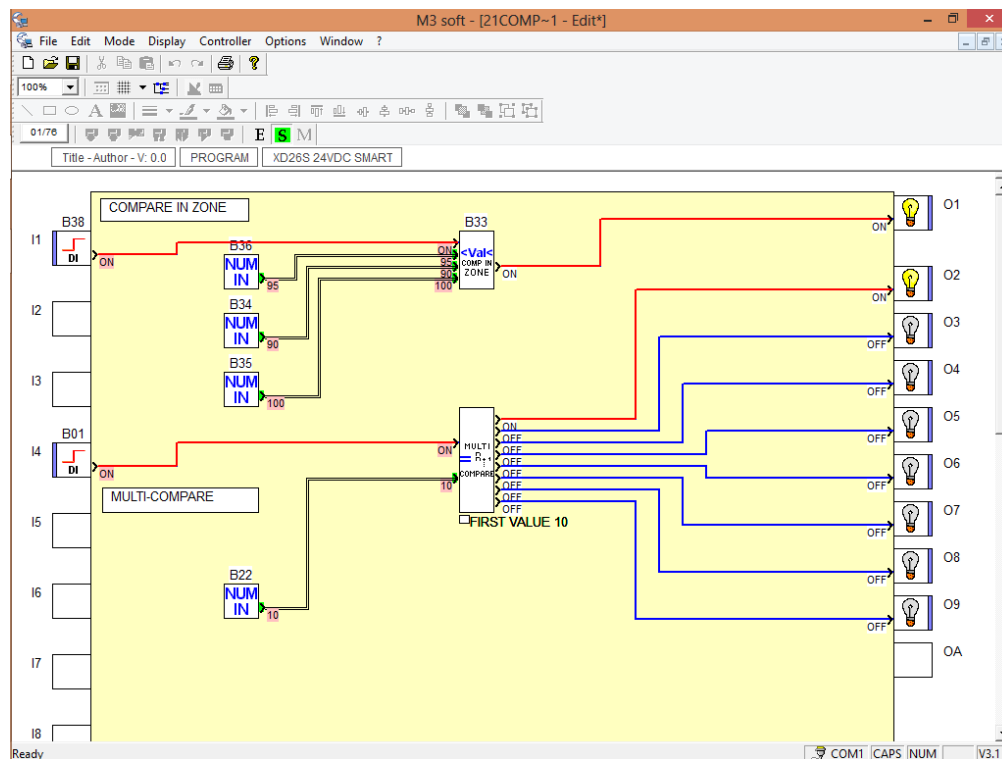
21. A program to study compare in-zone and multi-compare functions:

Program details:

1. Choose controller type and parameter window setting properly for input and output compatibility and validity as per requirement.
2. Make proper connection between blocks, input-output to avoid invalidity and malfunction.
3. Input (2 blocks), output (9 blocks), NUM IN (4 No's) and COMPARE IN ZONE function and MULTI-COMPARE

Hardware requirement:

1. Output indication Bulbs (9 No's)
2. Set of single stranded wires
3. Relay card



4.7 Study of sequencer

4.7.1 Study of Pump Management Sequencer

SEQUENCER: APPLICATION-SPECIFIC FUNCTION

PUMP MANAGEMENT: (TANK MANAGEMENT WITH CIRCULAR PUMP CHANGEOVER)

This function is used to set to ON a maximum of four digital outputs which can be activated (OUTPUT 1 ... OUTPUT 4). This number is equal to the maximum number of digital inputs (from 2 to 4) in the ON state. In addition, the outputs set to ON are selected so that in the event of prolonged operation, each output will have been set to ON the same number of times

The SQUARE ROOT function is accessible from the FBD function bar. This function has 4 inputs and 5 output. The function is accessible from the FBD function bar (CAL). SQUARE ROOT function is represented as follows:

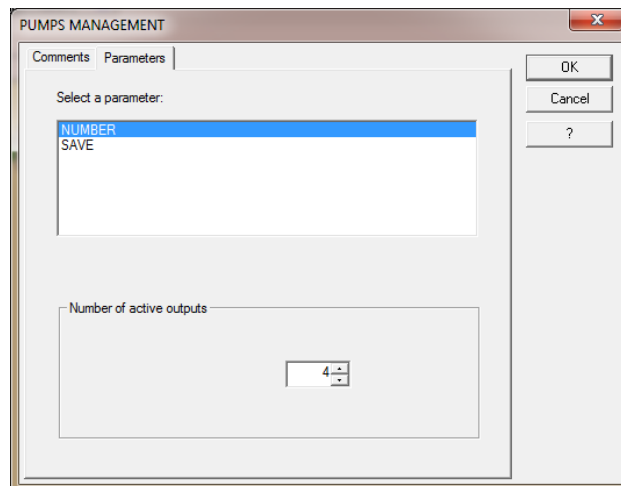
Function Bar icon:



Wiring sheet block:



Parameter Window:



In the parameter window the number of active outputs can be set and save off on power failure option can also be selected.

Input/Output:

Programmable Logic Controller

Input:

1. Input 1
2. Input 2
3. Input 3
4. Input 4

Output:

1. Output 1
2. Output 2
3. Output 3
4. Output 4
5. Pilot output number

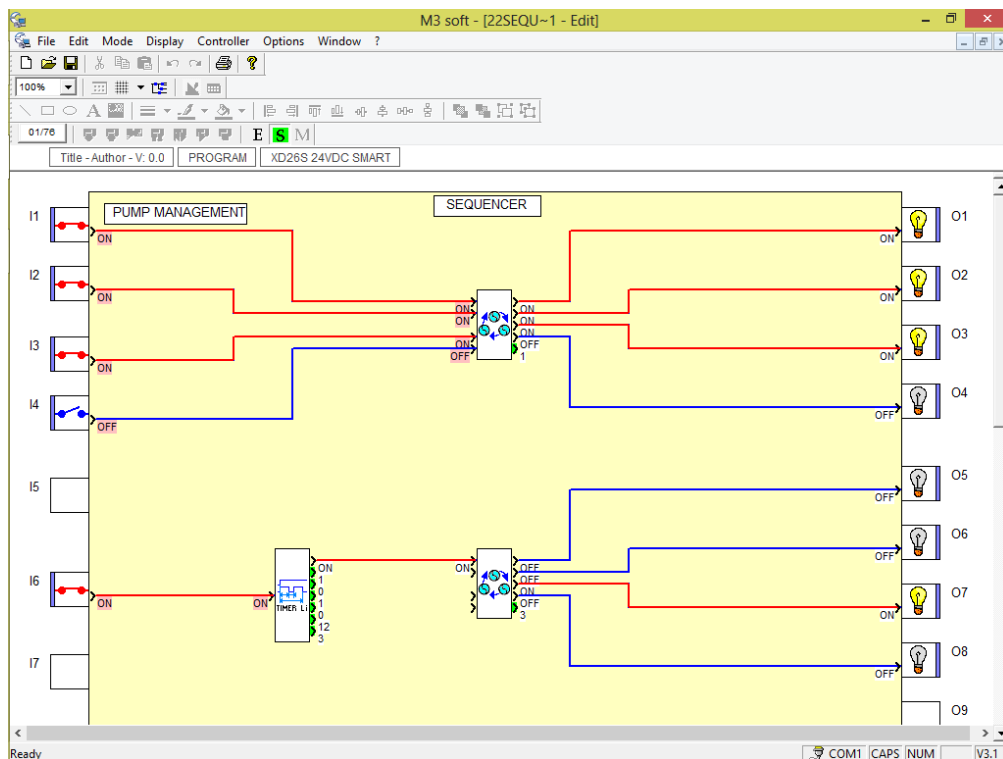
22. A program to study sequencer functions: Pump Management Block

Program details: Sequential lighting of 2 set of 4 bulbs using pump management block.

1. Choose controller type and parameter window setting properly for input and output compatibility and validity as per requirement.
2. Make proper connection between blocks, input-output to avoid invalidity and malfunction.
3. Input (5 blocks), output (8 blocks), PUMP MANAGEMENT block (2 No's) and Timer Li function (1 No's)

Hardware requirement:

1. Output indication Bulbs (8 No's)
2. Set of single stranded wires
3. Relay card



Programmable Logic Controller

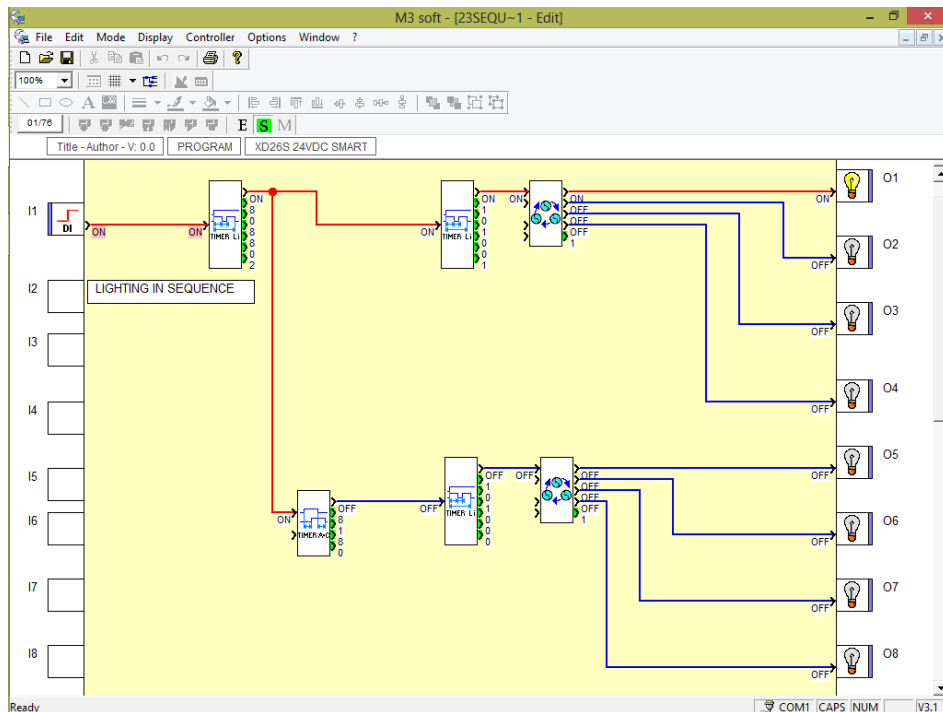
23. A program to study sequencer functions: Pump Management Block

Program details: Sequential lighting ON/OFF of a set of 8 bulbs using pump Management blocks.

1. Choose controller type and parameter window setting properly for input and output compatibility and validity as per requirement.
2. Make proper connection between blocks, input-output to avoid invalidity and malfunction.
3. Input (1 blocks), output (8 blocks), PUMP MANAGEMENT block (2 No's), Timer AC and Timer Li function (3 No's)

Hardware requirement:

1. Output indication Bulbs (8 No's)
2. Set of single stranded wires
3. Relay card



4.7.1 Study of CAM Sequencer

CAM BLOCK Cam Programmer

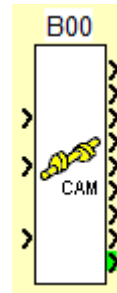
The cam programmer function CAM BLOCK controls a set of 8 built-in cam wheels.

On its 8 outputs (representing the 8 wheels), the function provides the state corresponding to the current position of the shaft wheels. The cam configuration can be set; for each position, output state is adjustable. Once the maximum value has been reached, the cam restarts from its initial position (output returns to 0). The CAM BLOCK function is accessible from the FBD function bar (APP). This function has 3 inputs and 9 output. The function is accessible from the FBD function bar (APP). CAM BLOCK function is represented as follows:

Function Bar icon:



Wiring sheet block:



Inputs/Outputs

Inputs:

1. MOVE FORWARD: this is the input that controls cam progress; it moves one step forward at each rising edge (change from inactive to active).

2. MOVE BACKWARD: this is the input that controls backward cam movement; it moves one step backward at each rising edge (change from inactive to active).

Note: The MOVE FORWARD input takes priority over the MOVE BACKWARD input.

Note: If the MOVE FORWARD and MOVE REVERSE inputs are not connected, they are set to inactive.

3. RESET (initialization): When this input is active, the cam is replaced to its initial position: the POSITION output is forced to 0.

Note: The RESET input takes priority over the MOVE FORWARD and MOVE BACKWARD inputs.

Note: If the RESET input is not connected, it is set to inactive.

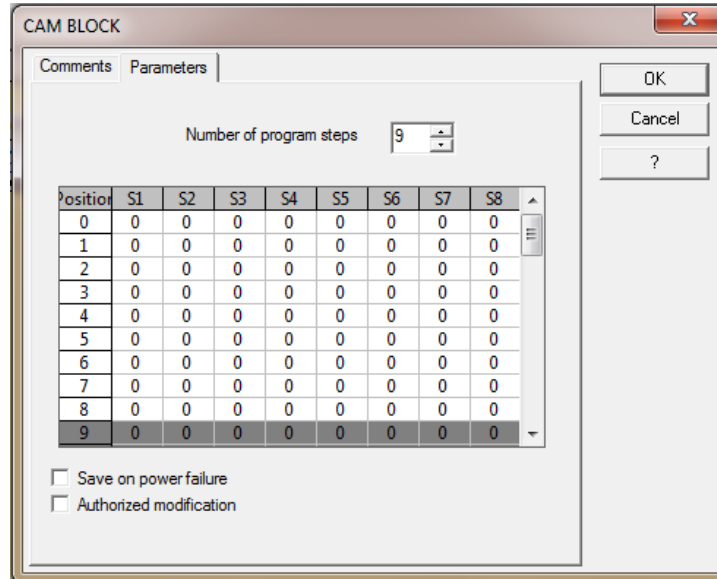
Outputs:

Programmable Logic Controller

OUTPUT 1 to 8: state corresponding to the current position of the shaft (representing the 8 wheels),

POSITION: current cam position (0 to 49).

Parameter Window:



In the Parameters window, the following parameters can be adjusted

1. The number of program steps: Its value is between 1 and 50,
2. Output status [1..8]: for each position of the shaft.
3. When selected, the Save on power fail parameter enables the current value of the timer to be retrieved following a power failure.

Modification of Parameters from the Front Panel can be done as follows:

To modify the parameters from the front panel of the controller,

1. check the Authorized modification box of the Parameters window.
2. From the PARAMETERS menu, it is possible to modify bit-wise the contents of all the cam programmer steps, but it is not possible to modify the number of steps.

you have entered the block number,

3. Enter the following data:

The step number: Value between [0..49],

Programmable Logic Controller

Output status [1..8]: For each output one can set the value to INACTIVE (empty diamond) or ACTIVE (black diamond).

Programmable Logic Controller

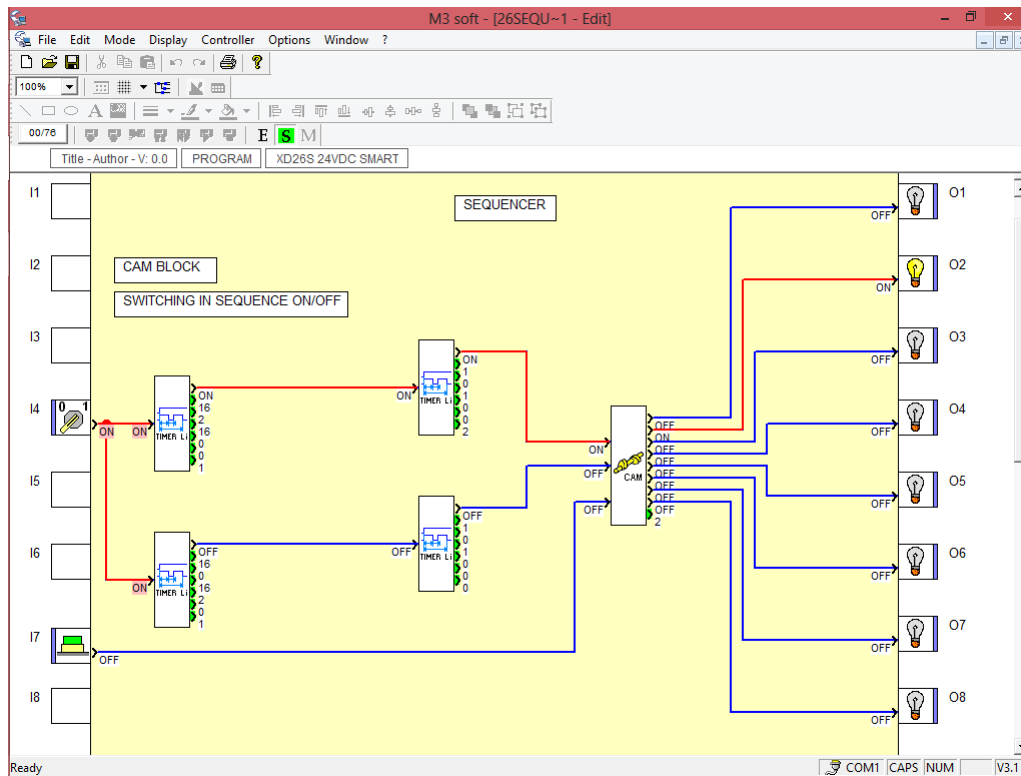
24. A program to Study of Sequencer: CAM Block

Program details: Sequential lighting ON/OFF of a set of 8 bulbs using CAM block.

1. Choose controller type and parameter window setting properly for input and output compatibility and validity as per requirement.
2. Make proper connection between blocks, input-output to avoid invalidity and malfunction.
3. Input (2 blocks), output (8 blocks), CAM block (1 No's) and Timer Li function (4 No's)

Hardware requirement:

1. Output indication Bulbs (8 No's)
2. Set of single stranded wires
3. Relay card



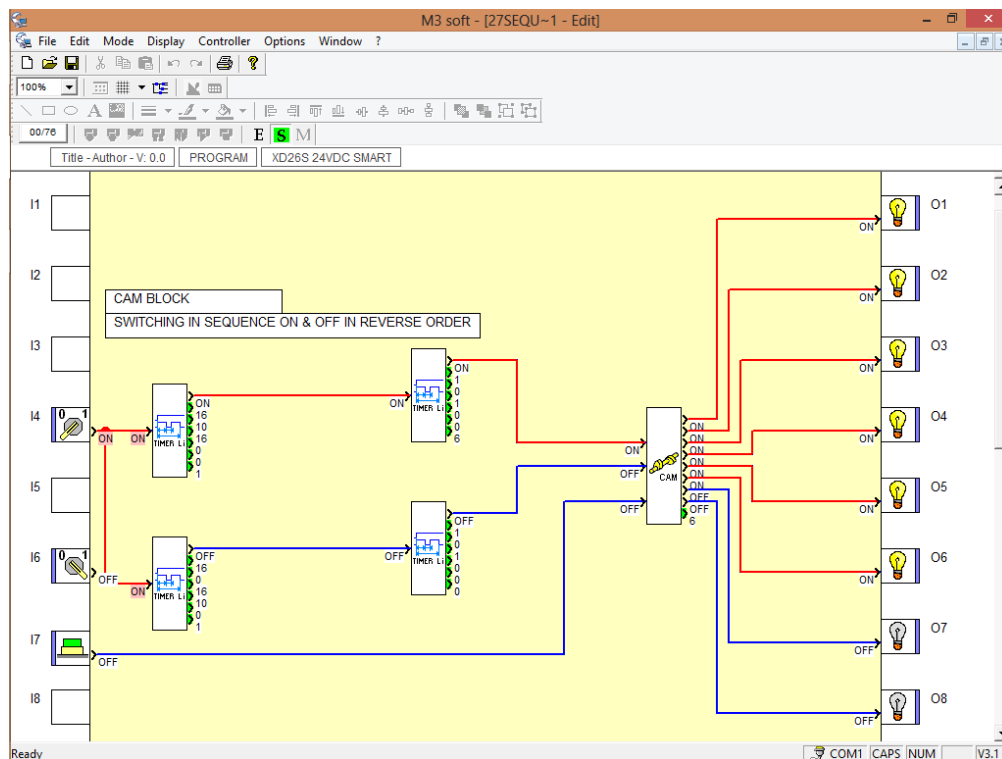
25. A program to Study of Sequencer: CAM Block

Program details: Sequential lighting ON of a set of 8 bulbs using CAM block.

1. Choose controller type and parameter window setting properly for input and output compatibility and validity as per requirement.
2. Make proper connection between blocks, input-output to avoid invalidity and malfunction.
3. Input (3 blocks), output (8 blocks), CAM block (1 No's) and Timer Li function (4 No's)

Hardware requirement:

1. Output indication Bulbs (8 No's)
2. Set of single stranded wires
3. Relay card



4.8 Study of HMI/COM

4.8.1 Study of LCD Display

LCD DISPLAY Screen

The DISPLAY function is used to display text, a date, a time or a numerical value on the LCD display instead of the controller INPUTS-OUTPUTS screen:

The DISPLAY function is used to display information on the following:

1. Text (maximum 72 characters),
2. Numerical values corresponding to the output of a block function used in the application.

Up to 8 DISPLAY blocks can be enabled simultaneously in one program. If this number is exceeded, only the first 8 blocks activated are displayed.

- a) Pressing the green OK and red ESC keys simultaneously replaces the display on the DISPLAY screen with the menu display.
- b) Pressing the ESC key again returns the display to the DISPLAY screen.

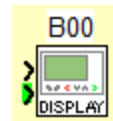
Both ASCII-standard characters and accented characters can be used. Characters and symbols that are not displayed in the data entry window when keyed are not supported.

The DISPLAY function is accessible from the FBD function bar. This function has 3 inputs and 9 output. The function is accessible from the FBD function bar (CAL). DISPLAY function is represented as follows:

Function Bar icon:

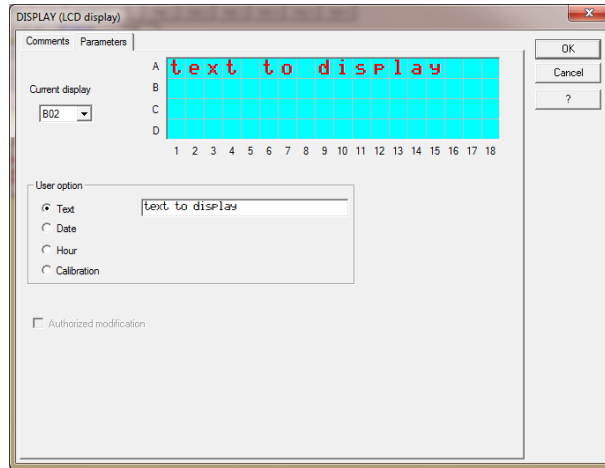


Wiring sheet block:

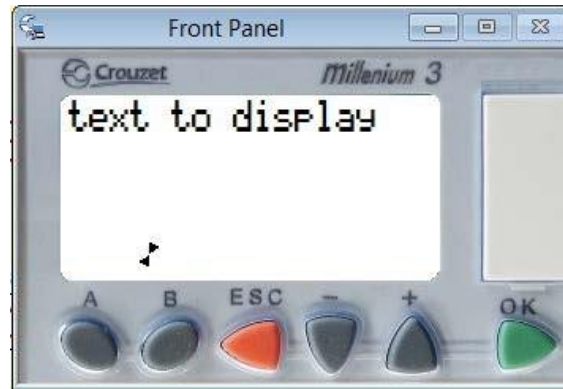


Parameter Window:

Programmable Logic Controller



Front Panel Window in simulation mode:



Input/Output:

Input:

1. ENABLE FUNCTION: This is the Discrete-type DISPLAY function control input. The state of this input determines block operation: if the ENABLE FUNCTION input is active, the information is displayed on the LCD; otherwise there is no display. If this input is not connected, then it is considered to be active.
2. VALUE INPUT: This is the selection input that determines the nature of the information to be displayed. If this input is:
 - i. Not connected: The display corresponds to the selection made in the User options zone.
 - ii. Connected to the output of a function block: The display corresponds to the value sent by this output.

Output: Output message is displayed on the front panel and can also be viewed in simulation and monitoring mode in computer.

Parameters

Programmable Logic Controller

The adjustable parameters depend on whether or not the VALUE INPUT is connected

1st case: VALUE INPUT not connected

The display corresponds to the selection made in the User options zone.

Depending on the options chosen, the following can be displayed:

Text: a string of characters

Date: the current value of the internal date of the device on which the program is executed (controller or simulator)

Time: The current internal time value

Calibration: Drift value of the controller's internal clock

2nd case: VALUE INPUT connected

The integer value present on the input is converted into a string of characters, whose display format depends on the option that has been selected:

Integer 1/1 - 1/10000

Calendar date

Bar chart

Maxinumber

Entry procedure:

Entering one of the parameters of a DISPLAY block can be done as follows:

Step 1:

Is the VALUE INPUT connected?

yes, then specify the display format.

If no, then enter text in the User options box.

Step 2:

Position the start of the text using the mouse:

Step 3:



Confirm using the green OK key.

Result: The new DISPLAY block is saved and the parameters window is closed.

Modifying Data from the Front Panel:

When the Modification authorized option is checked, it is possible to modify the data displayed directly from the display screen by proceeding as follows:


Programmable Logic Controller

Step 1: Use the  and  keys to place the cursor on the data to modify.

Step 2: Confirm by pressing the OK key .

Results the selected data flashes.

Step 3: Use the  and  keys to scroll to the desired value.

Step 4: Confirm by pressing the OK key .

Programmable Logic Controller

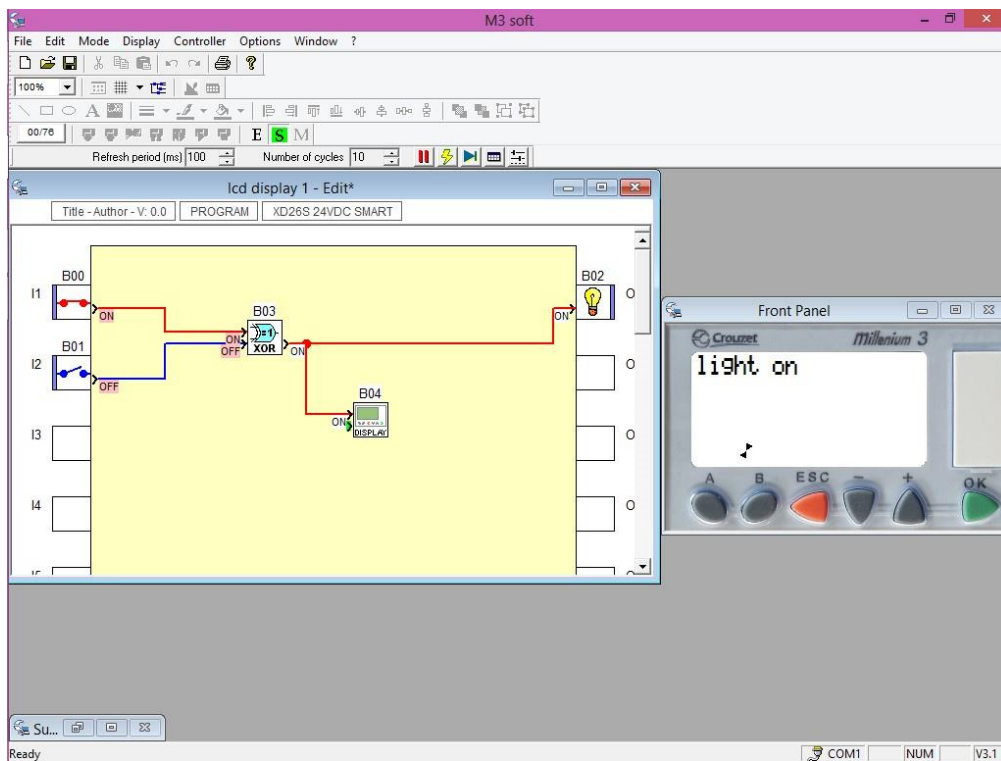
26 A program to display data on LCD screen.

Program details: Displaying data on LCD screen.

1. Choose controller type and parameter window setting properly for input and output compatibility and validity as per requirement.
2. Make proper connection between blocks, input-output to avoid invalidity and malfunction.
3. Input (2 blocks), output (1 blocks), LCD display (1 No's) and XOR gate (1 No's)

Hardware requirement:

1. Output indication Bulbs (1 No's) and Millenium PLC display
2. Set of single stranded wires
3. Relay card



4.8.2 Study of Menu Scroll

MENUSCROLL

This function is used to set one of the digital outputs to ON. The MENUSCROLL function is accessible from the FBD function bar.

Only one output is active at a time. It is selected by means of the Plus and Minus inputs. At the start, position 1 is at ON then subsequent ones can be set to ON in succession until "Active position number" = "Number of outputs". While the Reset input is active, all outputs are set to OFF and once deactivated, only position 1 is at ON. This function can be useful for menu scrolling if displays are wired as outputs. This function has 4 inputs and 9 outputs. The function is accessible from the FBD function bar (HMI/COM). MENUSCROLL function is represented as follows:

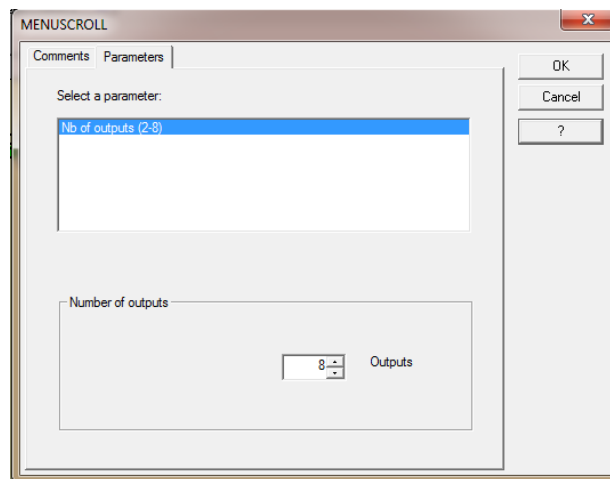
Function Bar icon:



Wiring sheet block:



Parameter Window:



Input/Output:

Programmable Logic Controller

Input:

1. Validation: Function validation input. Until this input is activated, the function remains inert. Validation is active implicitly if it has not been connected.
2. Plus input: Sets the active output to OFF and the next to ON provided that the number in "Number of outputs" has not been reached.
3. Minus input: Sets the active output to OFF and the previous one to ON provided that the "Active position number" is other than 1.
4. Reset: Resets the first output to ON and all the others to OFF.

Output:

1. Position 1
2. Position 2
3. Position 3
4. Position 4
5. Position 5
6. Position 6
7. Position 7
8. Position 8
9. Active Position Number

Programmable Logic Controller

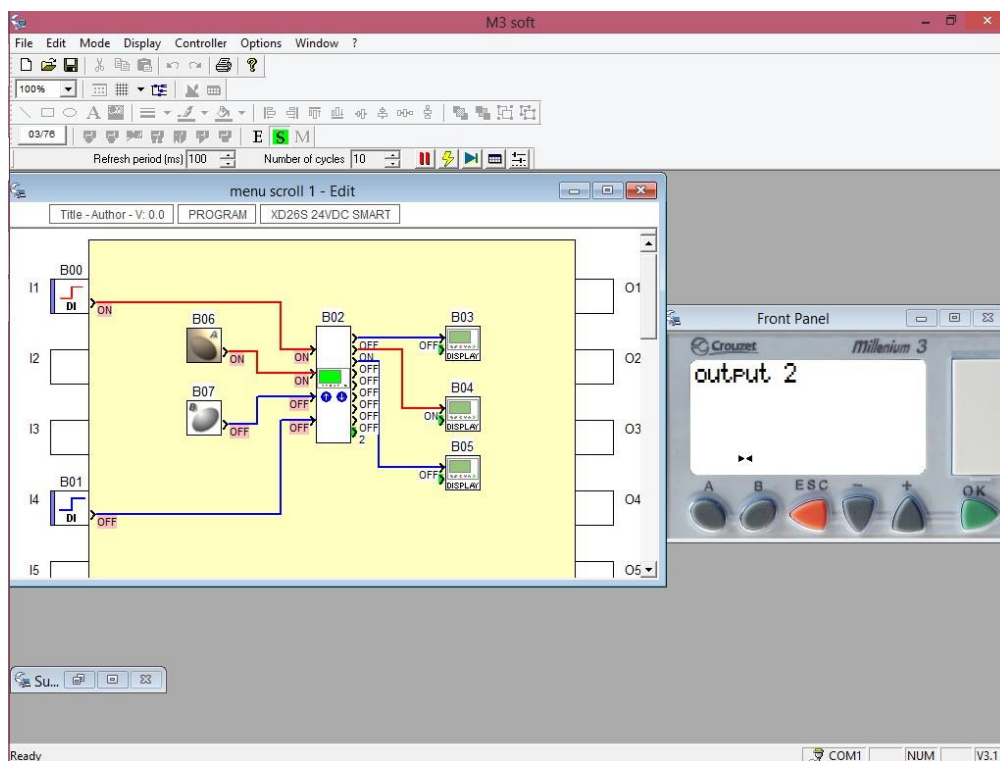
27. A program to study Menu Scroll.

Program details: Displaying data on LCD screen using Menu scroll (from program) and PLC display screen.

1. Choose controller type and parameter window setting properly for input and output compatibility and validity as per requirement.
2. Make proper connection between blocks, input-output to avoid invalidity and malfunction.
3. Input (2 blocks), MENUSCROLL, LCD display (3 No's), button A and button B.

Hardware requirement:

1. Millenium PLC display
2. Set of single stranded wires
3. Relay card



4.8 Applications

4.8.1 Motor control using PLC

P28. Programs for motor control:

a) Latching and unlatching of motor:

b) Interlocking concept:

P29. A Program for motor control: Control of industrial process:

4.8.2 Sequential lighting of bulbs

P30. A program on Sequential lighting 1 of bulbs using timers:

P31. A program on Sequential lighting 2 of bulbs using timers:

4.8.3 Automatic Traffic control

P32. A program for implementation of Manual Traffic control using timer functions:

P33. A program for implementation of Automatic Traffic controls 1 using timer functions:

P34. A program for implementation of Automatic Traffic controls 2 using timer functions:

4.8.4 Industrial applications

P35. A program to illustrate Alarm System:

P36. A program to illustrate Tank level System:

P37. A program to illustrate temperature control System:

P38. A program for speed control of dc motor using PLC and drive system:

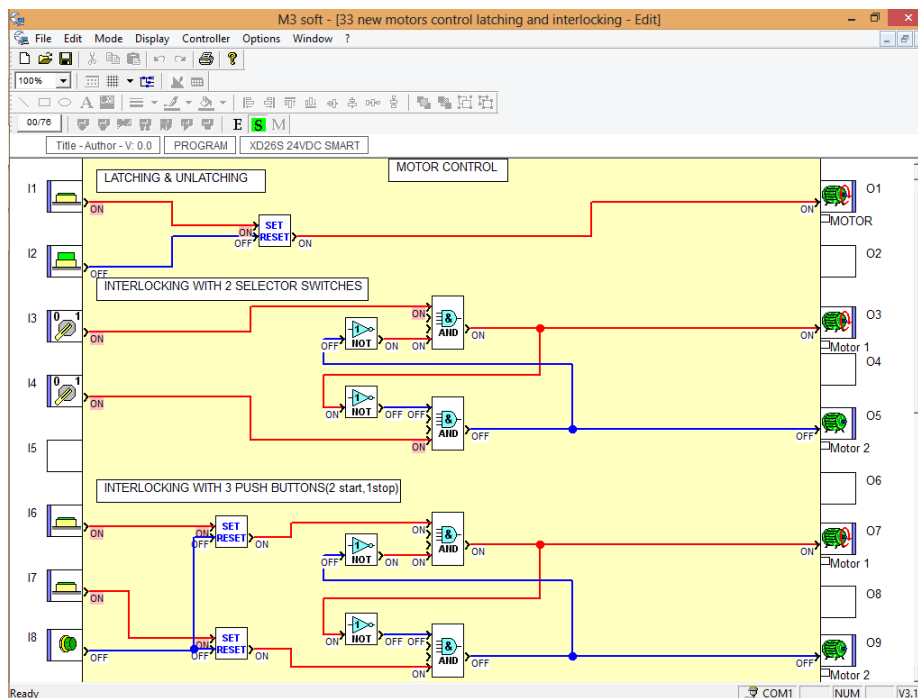
28. Programs for motor control: Latching and interlocking of motor

Program details:

1. Choose controller type and parameter window setting properly for input and output compatibility and validity as per requirement.
2. Make proper connection between blocks, input-output to avoid invalidity and malfunction.
3. Input (7 blocks), output (5 blocks), SET RESET function (3 No's), AND (4 No's) and NOT (4 No's)

Hardware requirement:

1. Push-buttons (5 No's), selector switches (2 No's) and Motors (5 No's).
2. Set of single stranded wires
3. Relay card



Programmable Logic Controller

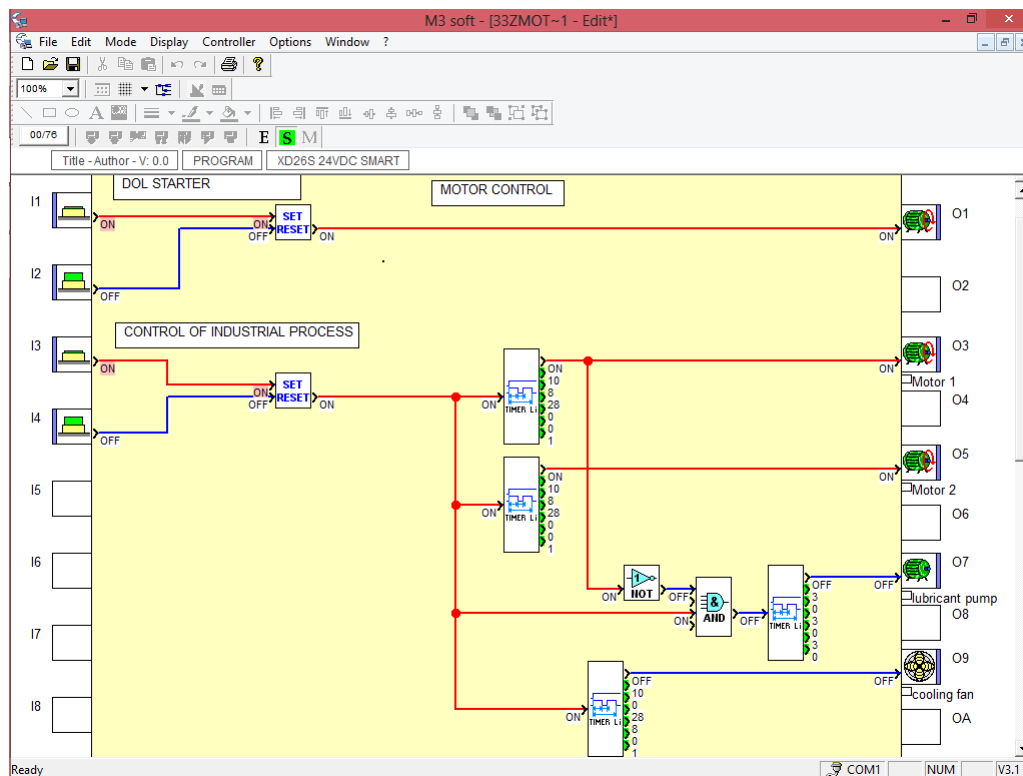
29. Programs for motor control: Control of industrial process

Program details:

1. Choose controller type and parameter window setting properly for input and output compatibility and validity as per requirement.
2. Make proper connection between blocks, input-output to avoid invalidity and malfunction.
3. Input (5 blocks), output (8 blocks), SET RESET function (2 No's) and Timer Li function (3 No's)

Hardware requirement:

1. Push-buttons (4 No's), Motors (4 No's) and a fan
2. Set of single stranded wires
3. Relay card



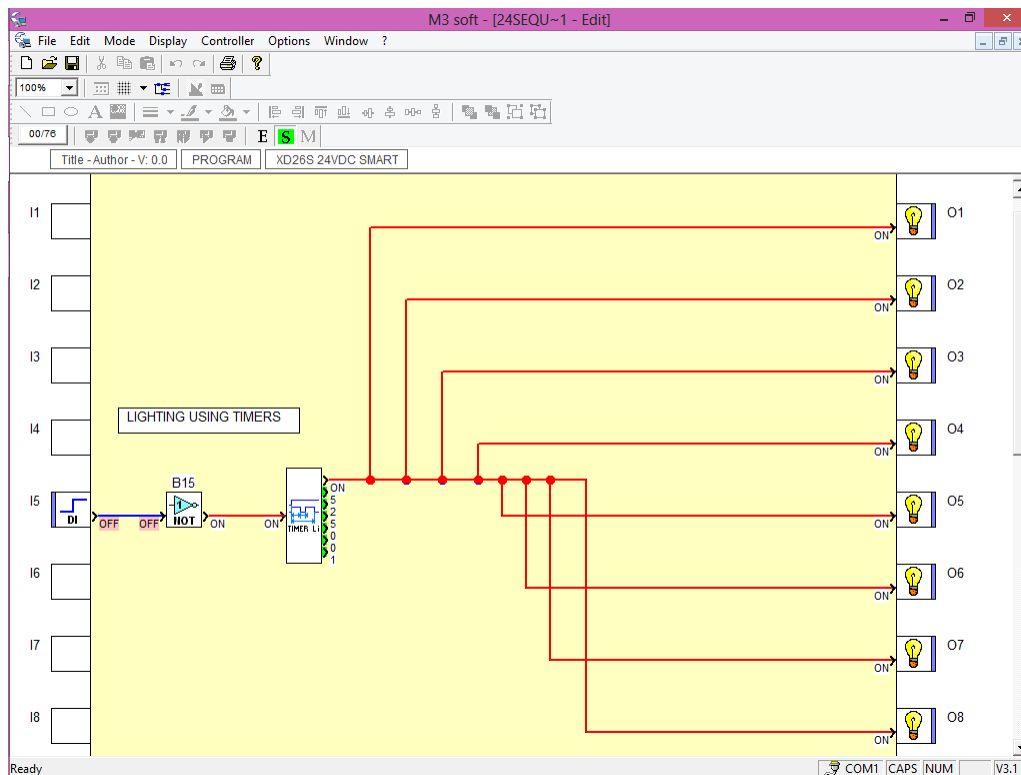
30. A program on Sequential lighting of bulbs using timers

Program details:

1. Choose controller type and parameter window setting properly for input and output compatibility and validity as per requirement.
2. Make proper connection between blocks, input-output to avoid invalidity and malfunction.
3. Input (1 blocks), output (8 blocks), NOT gate and Timer Li function (1 No's)

Hardware requirement:

1. Selector switch (1 No's), Bulbs (8 No's)
2. Set of single stranded wires
3. Relay card



Programmable Logic Controller

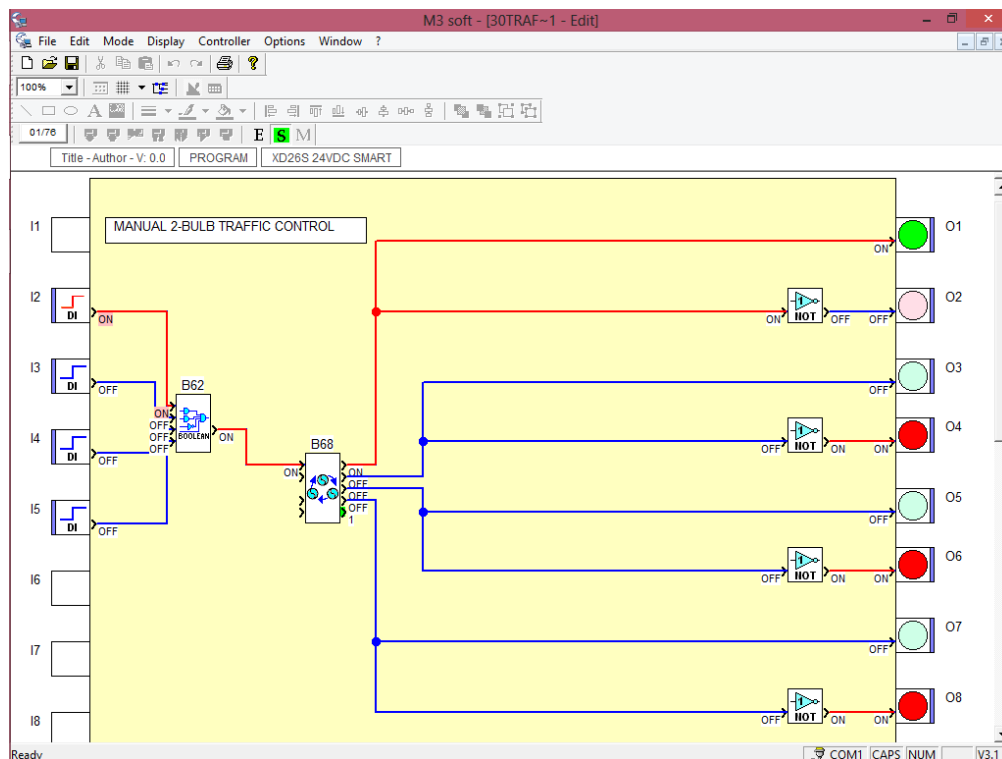
32. A program for implementation of Manual Traffic control using timer functions.

Program details:

1. Choose controller type and parameter window setting properly for input and output compatibility and validity as per requirement.
2. Make proper connection between blocks, input-output to avoid invalidity and malfunction.
3. Input (4 blocks), output (8 blocks), Sequencer Pump Management Block (1 No's), Boolean function (1 No's) and NOT gates (4 No's).

Hardware requirement:

1. Selector switches (4 No's), Bulbs (8 No's, 4 red and 4 green)
2. Set of single stranded wires
3. Relay card



Programmable Logic Controller

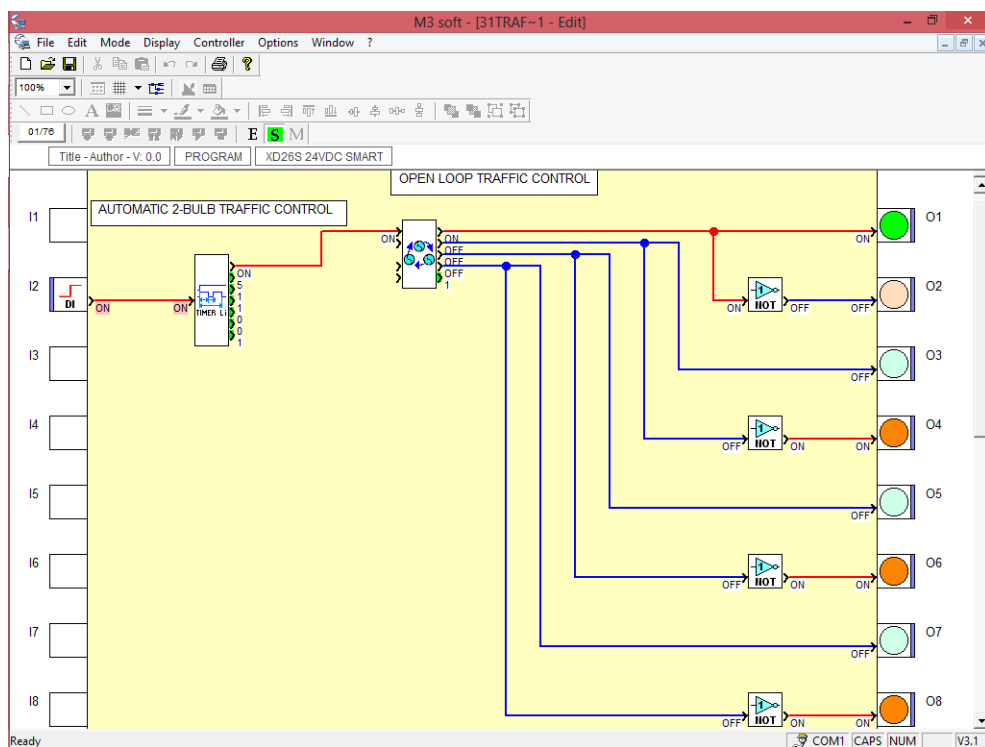
33. A program for implementation of Automatic Traffic control using timer functions:

Program details:

1. Choose controller type and parameter window setting properly for input and output compatibility and validity as per requirement.
2. Make proper connection between blocks, input-output to avoid invalidity and malfunction.
3. Input (1 blocks), output (8 blocks), Sequencer Pump Management Block (1 No's) and Timer Li (1 No's).

Hardware requirement:

1. Selector switches (1 No's), Bulbs (8 No's, 4 red and 4 green)
2. Set of single stranded wires
3. Relay card



Programmable Logic Controller

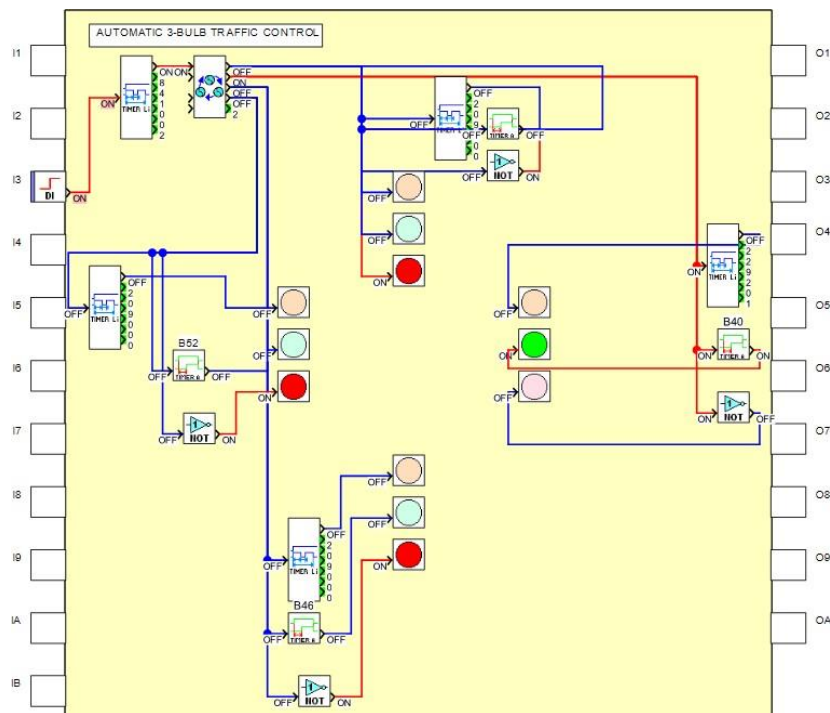
34. A program for implementation of Automatic Traffic control using timer functions:

Program details:

1. Choose controller type and parameter window setting properly for input and output compatibility and validity as per requirement.
2. Make proper connection between blocks, input-output to avoid invalidity and malfunction.
3. Input (1 blocks), output (8 blocks), Sequencer Pump Management Block (1 No's) and Timer Li (1 No's).

Hardware requirement:

1. Selector switches (1 No's), Bulbs (12 No's; 4 orange, 4 red and 4 green)
2. Set of single stranded wires
3. Relay card



Programmable Logic Controller

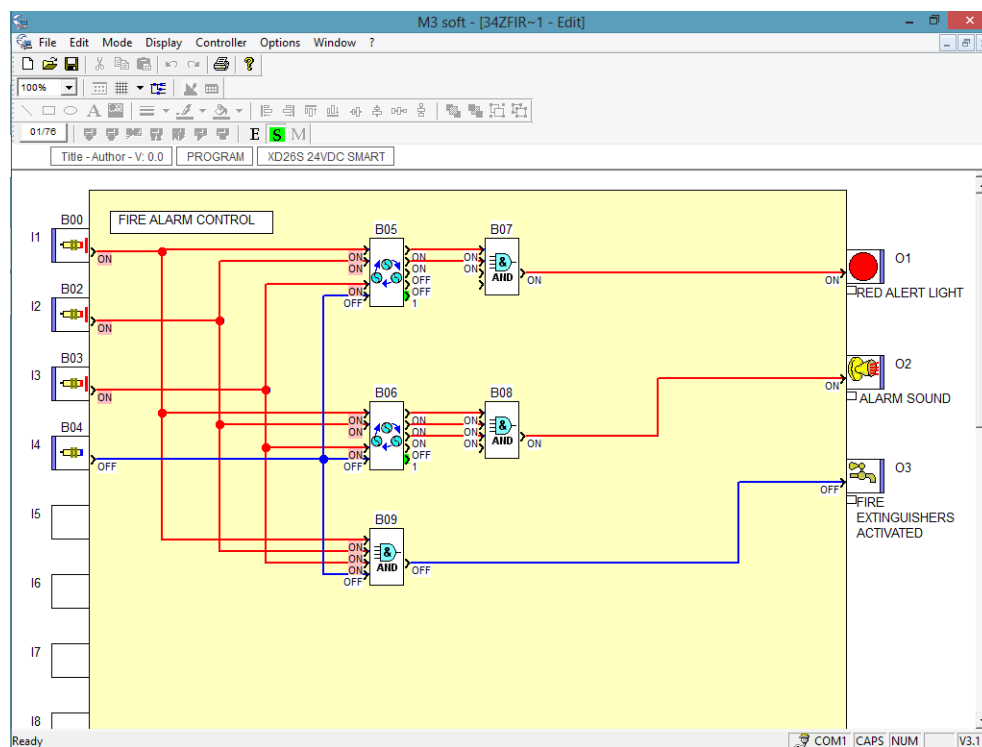
35. A program to illustrate Alarm System:

Program details:

1. Choose controller type and parameter window setting properly for input and output compatibility and validity as per requirement.
2. Make proper connection between blocks, input-output to avoid invalidity and malfunction.
3. Input (4 blocks), output (3 blocks), Sequencer Pump Management Block (2 No's) and 4-input AND gates (3 No's).

Hardware requirement:

1. Different types of Sensors (4 No's), red light, Siren and Fire Extinguisher
2. Set of single stranded wires
3. Relay card



Programmable Logic Controller

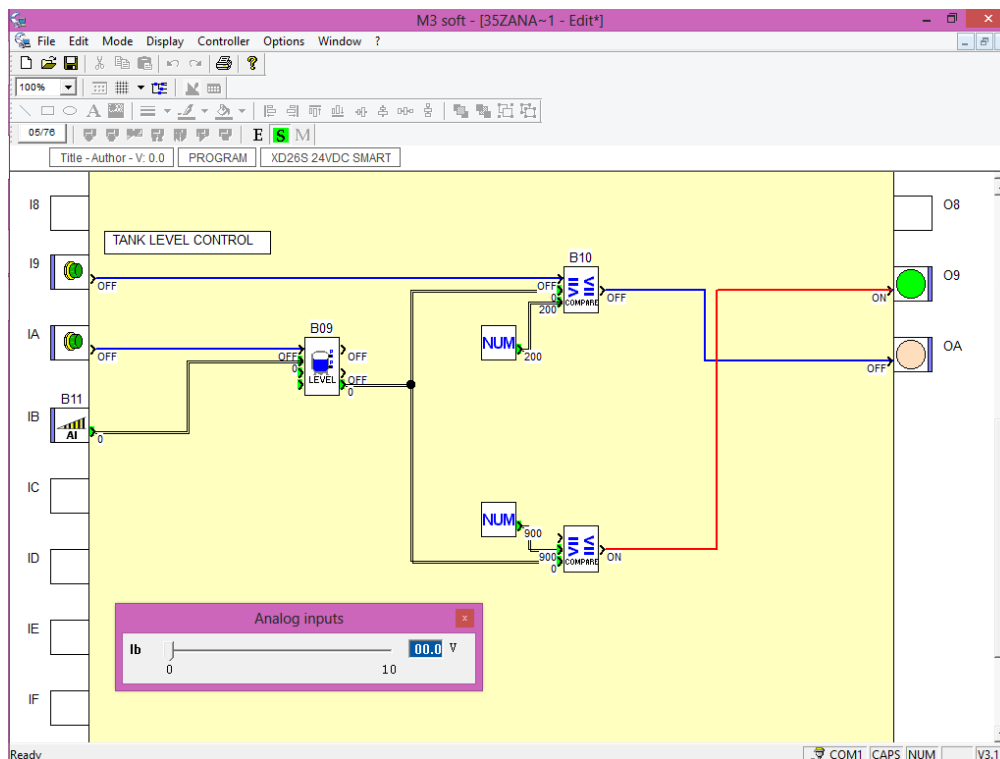
36. A program to illustrate Tank level System:

Program details:

1. Choose controller type and parameter window setting properly for input and output compatibility and validity as per requirement.
2. Make proper connection between blocks, input-output to avoid invalidity and malfunction.
3. Input (1 analog and 2 Discrete blocks), output (2 blocks), LEVEL function (1 No's), NUM block (2 No's) and COMPARE block(2 No's).

Hardware requirement:

1. Push button (2 No's), sensor with A/D converter, green light and red light
2. Set of single stranded wires
3. Relay card



Programmable Logic Controller

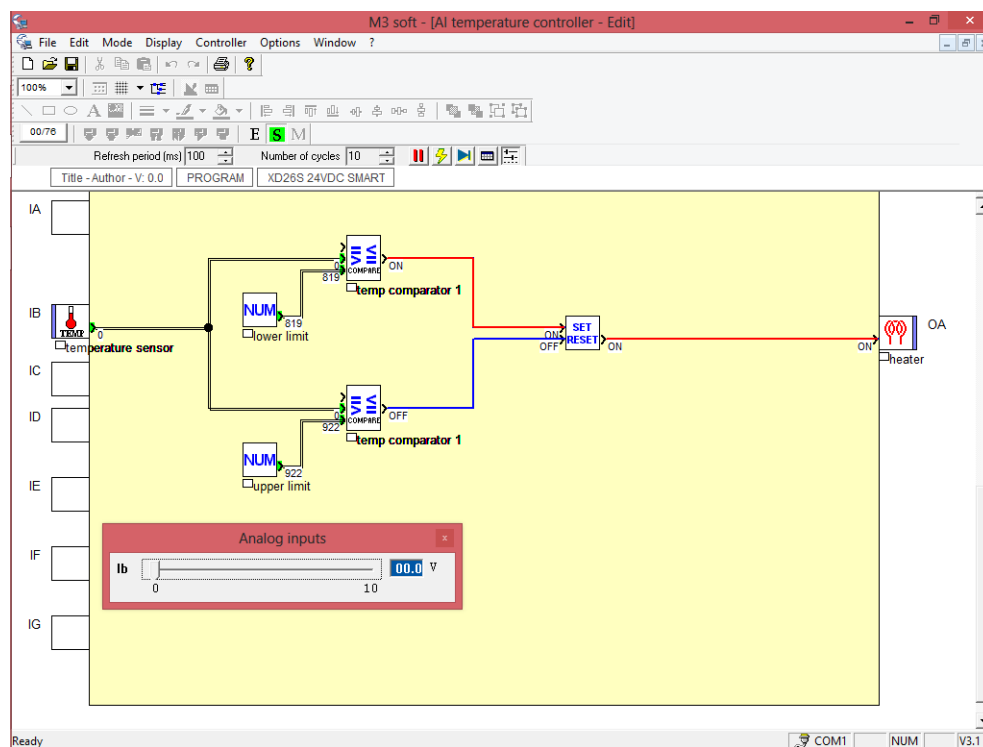
37. A program to illustrate temperature control System:

Program details:

1. Choose controller type and parameter window setting properly for input and output compatibility and validity as per requirement.
2. Make proper connection between blocks, input-output to avoid invalidity and malfunction.
3. Input (1 Discrete blocks), output (1 blocks), NUM block (2 No's) and COMPARE block (2 No's) and SET RESET function.

Hardware requirement:

1. Temperature sensor with A/D converter and heater.
2. Set of single stranded wires
3. Relay card



Programmable Logic Controller

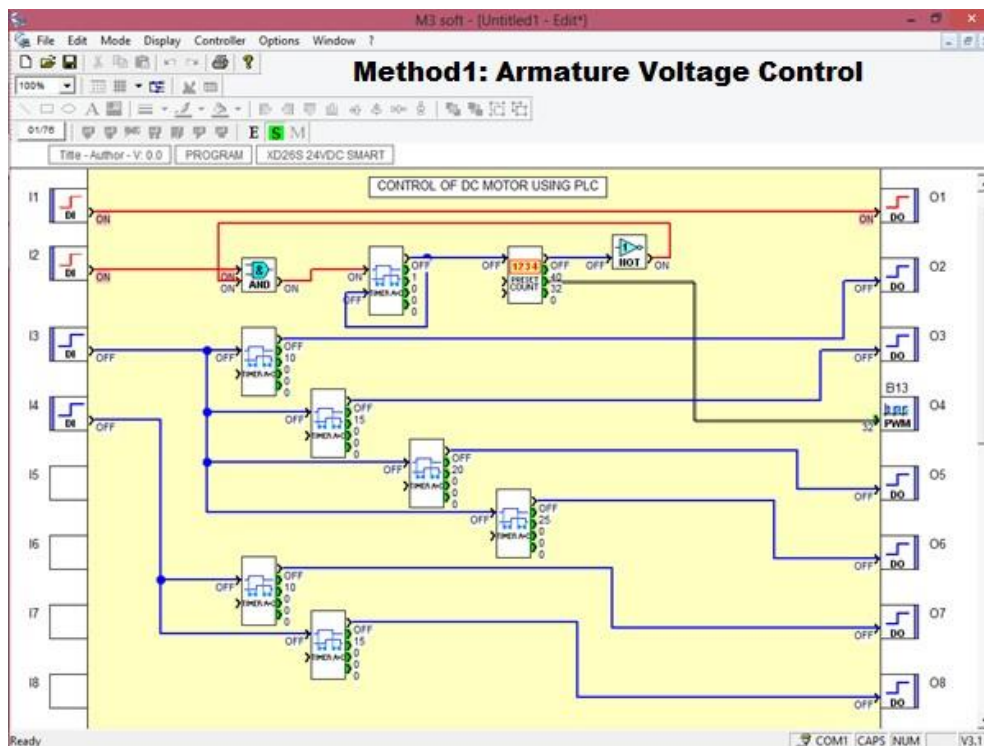
38. A program for speed control of dc motor using PLC and drive system:

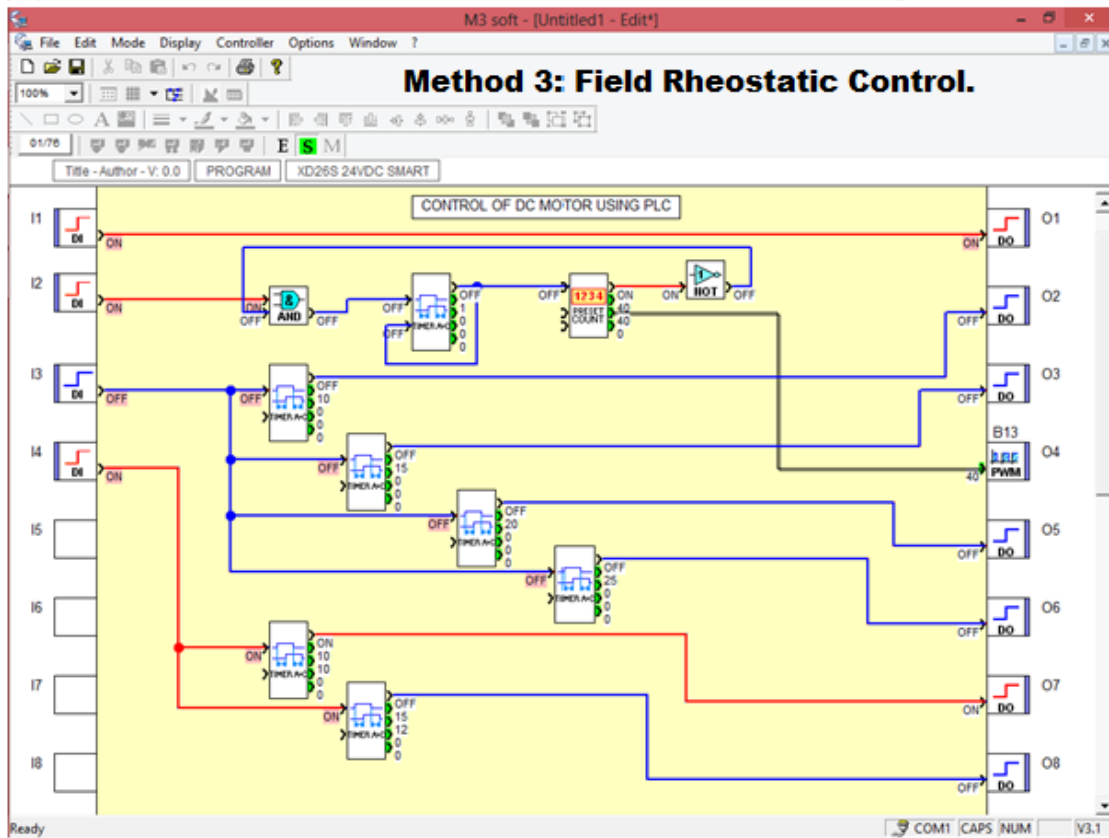
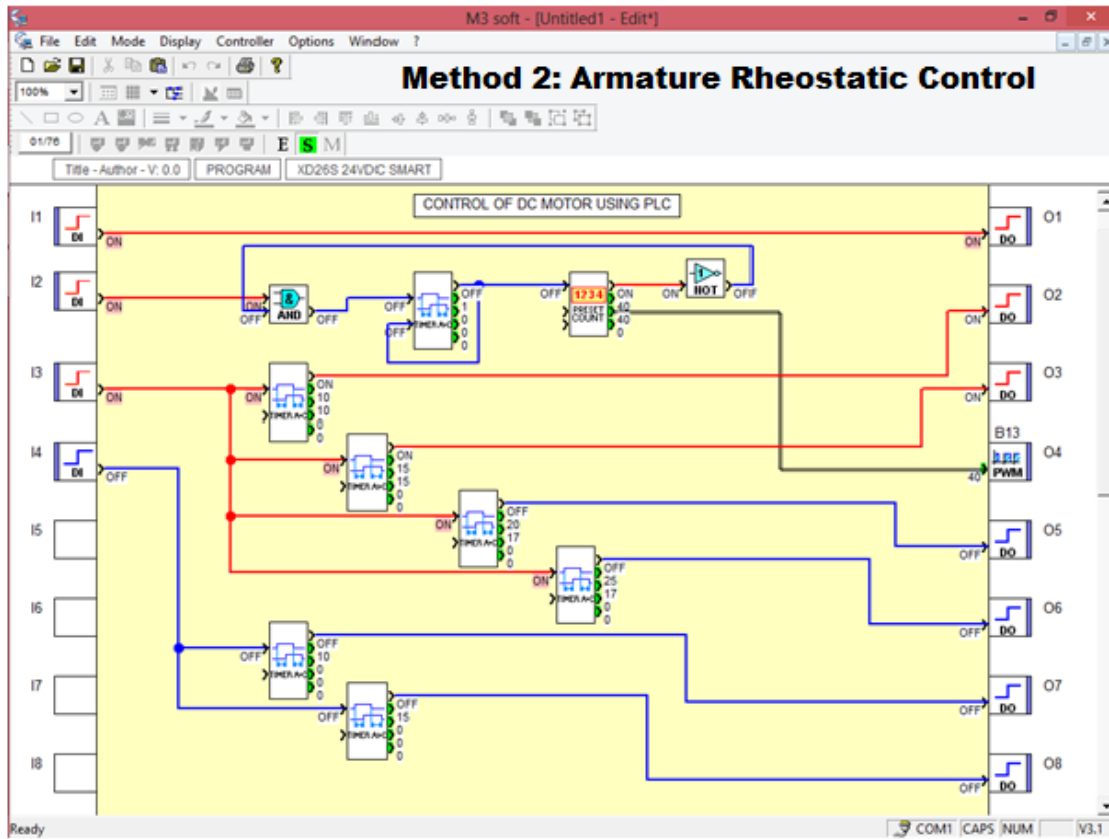
Program details: 3 methods of speed control: 1. Armature voltage control, 2. Armature rheostatic control and 3. Field rheostatic control

1. Choose controller type and parameter window setting properly for input and output compatibility and validity as per requirement.
2. Make proper connection between blocks, input-output to avoid invalidity and malfunction.
3. Input (4 blocks), output (8 blocks), Timer AC function (7 No's), preset counter (7 No's), AND gate and NOT gate

Hardware requirement:

1. DC Motors, D/A converter and dc drive circuitry.
2. Relay card and Set of single stranded wires.





5. LADDER PROGRAMMING (LAD)

5. LADDER PROGRAMMING (LAD)

5.1 Study of basic control function

Ladder language

Ladder language (LD) is a graphic language. It can be used to transcribe relay diagrams, and is suited to combinational processing. It provides basic graphic symbols: contacts, coils, blocks. Specific calculations can be executed within the operation blocks. The ladder network is between the first "contact" column (**Contact 1**) and the "coil" column.

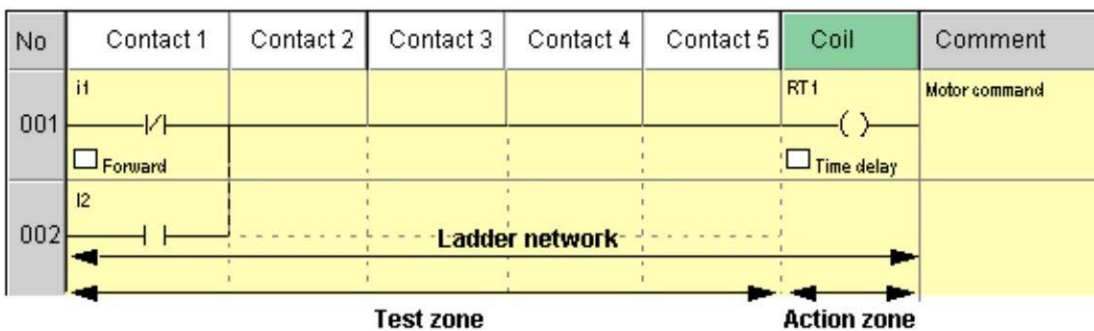
Ladder Network Description

A ladder network is made up of a collection of graphic elements set out over a grid with A maximum 120 program lines, Each line comprising a maximum of 5 contacts and a coil.

It is divided into two zones:

The **test zone**, in which the conditions necessary for triggering an action (contacts) are displayed,

The **action zone**, which applies the result following a logical test combination (coils).



Definition

Definition of an Action An action is applied to an automation function (timer, counter, etc.), an auxiliary relay or a PLC output. An action causes a change in state specified for each associated function. For example: An RT1 action causes a reset of the T1 timer, An SM1 action causes a set of the M1 auxiliary relay.

Definition of an Automation Function

An automation function (timer, counter, auxiliary relay, etc.) is defined by:

1. Input data or actions,
2. Output data or statuses,
3. Adjustment parameters.

Programmable Logic Controller

The following diagram shows the structure of a function

General

Graphic elements are ladder language instructions. **Graphic elements** function it that The inputs/outputs of the PLC (push-buttons, sensors, relays, LEDs, etc.), Automation functions (timers, counters, etc.), Logic operations, Internal variables (auxiliary relays) of the PLC.

Comments For each line of a ladder network (optional).

Contacts

Graphic elements of the contacts are programmed in the test zone and take up one cell (one row high by one column wide).

Normally open Contact



Conducting contact when its controlling input (switch, sensor, etc.) is active.

Normally closed Contact



Conducting contact when its controlling input is inactive.

Linking Elements

Linking graphic elements are used to connect test and action graphic elements.

Horizontal connection



Used to link test and action graphic elements together between the two potential bars.

A horizontal connection represents a logical **AND**. It sends the state of the contact located immediately to its left to the contact located immediately to its right.

Vertical connection



Used to link test and action graphic elements in parallel.

A vertical connection represents the logical **OR** of the **active** states of the horizontal connections located to its left, i.e.:

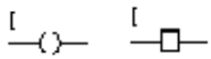
Programmable Logic Controller

- a) Inactive if the states of all the horizontal contacts located to the left are inactive,
- b) Active if at least one of the horizontal contacts located to the left is active.

Coils

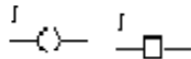
The graphic elements of the coils are programmed in the action zone and take up one cell (one row high by one column wide).

1 Direct coil



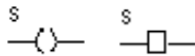
The coil is energized if the contacts to which it is connected are conducting (contact mode).

Impulse coil



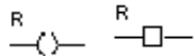
The coil is energized if the contacts to which it is connected change state (impulse relay mode).

Set or latch coil



The coil is energized once the contacts to which it is connected are conducting, then stays triggered even if later the contacts are no longer conducting (SET mode).

Reset or unlatch coil



The coil is deactivated when the contacts to which it is connected are conducting. It remains inactive even if later the contacts are no longer conducting (RESET mode).

Note: For upward compatibility, the four types of functions for a given Q output coil or M auxiliary relay may be used in the same wiring diagram.

Programming Rules for a Ladder Network

General

Ladder networks are programmed using graphic elements, observing the following programming rules.

Programming Rules

The programming of a ladder network must obey the following rules:

- a) Test and action graphic elements each occupy one cell within a network,
- b) All ladder networks end with at least one action (coil) in the final column,
- c) Actions are always located in the coil column,
- d) A coil corresponds to the triggering of an action applied to an automation function (timer, counter, auxiliary relay, PLC output, etc.),
- e) The status of an automation function may be used as a test (contact). The contact is then designated by the name of the associated function,

for example: T1 represents the status of the "T1" timer t1 represents the complementary status of the "T1" timer
Links are read (interpreted) from left to right,

- f) If, in a network, we use an S (Set) action for an automation function (output, auxiliary relay, etc.), it is generally advisable to use an R (Reset) action for the same function. **Exception:** An S action is used without an R action for detecting operating anomalies that can only be reset on receiving a "RESET-INIT" action from the automation program,
- g) The R (Reset) actions of an automation function always take priority over S (Set) actions applied to the same function at the same time,
- h) Network tests combine in the same way as an electrical voltage circuit from the left-hand network column (+V) to the right-hand network column (+0v).

Programmable Logic Controller

Program View

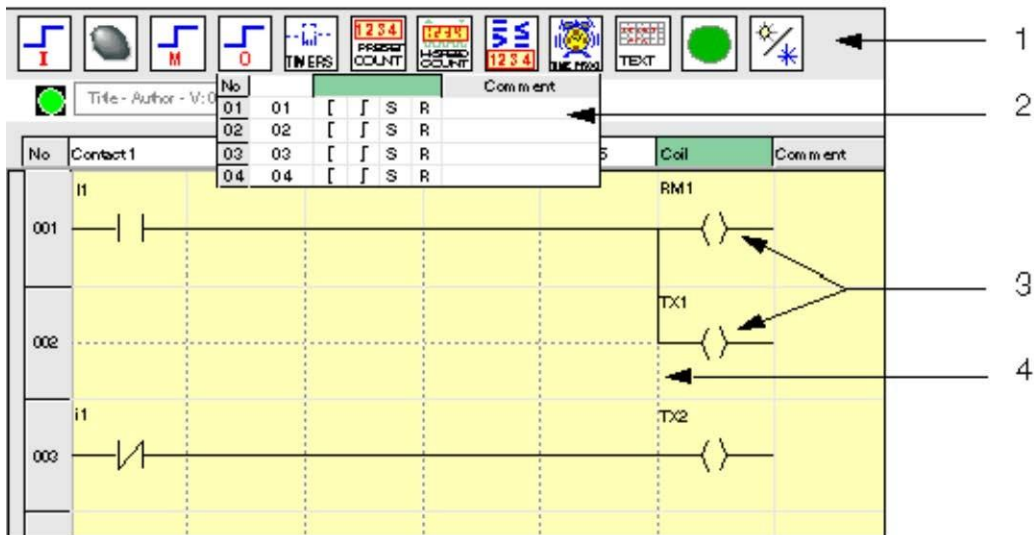
Overview

Using the software workshop in **Program view** mode allows you to adopt a software-based approach to programming:

1. Use of toolbars
2. Creation of the application by dragging and dropping automation functions
3. Use of parameter setting windows
4. Clear overall visibility of the application

In this form, the workspace is made up of a wiring sheet to which the various automation functions are added.

This approach is designed for people with experience using the programming software workshops commonly used in the automation world.



The elements are described below:

1 Function bar

Groups together the available automation function types. Click on a type of function to display the table of available functions.

2: Function type table

Shows the functions available for each function:

- a) The list of its outputs (or states)

Programmable Logic Controller

- b) The list of its inputs (or actions)
- c) The comment associated with the function

Enter comments directly in the **Comment** column. Drag the functions from the table to the wiring sheet. The color of the first line indicates the place in the wiring sheet where the function may be placed:

- White: An output (or state) to a contact column
- Green: An input (or action) to the coil column

3 : Symbol

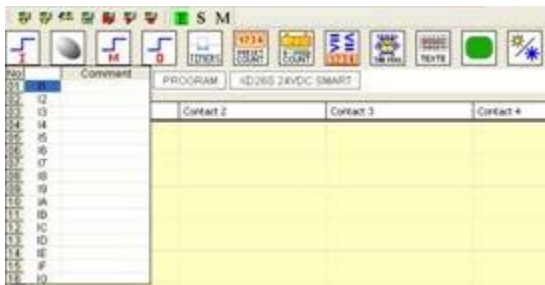
Represents a **Ladder symbol** or an **Electrical symbol** function according to the choice made in the **Display** menu. Double-click on the symbol to open the parameters window.

4 : Pre-drawn connection

Used to make a connection on the wiring sheet. Click with the mouse on the horizontal and/or vertical pre-drawn connections to draw a connection.

LD Language Elements

1. Discrete Inputs



2. Discrete (DISCR) Outputs

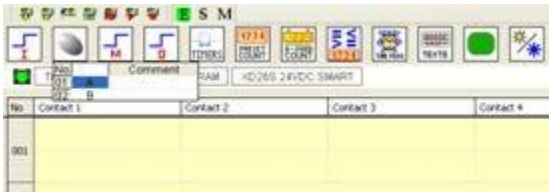


3. Auxiliary Relays

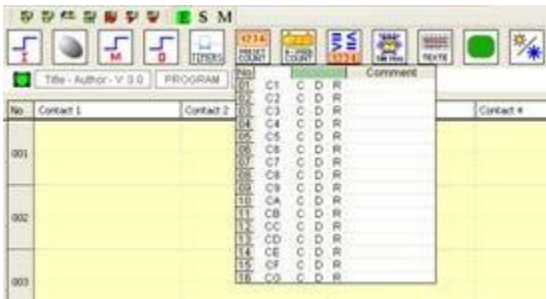
Programmable Logic Controller



4. A/B Keys



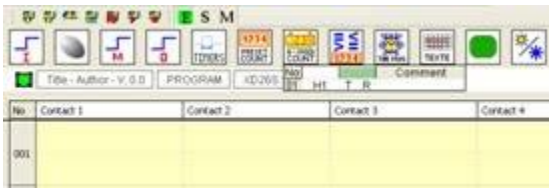
5. Counters



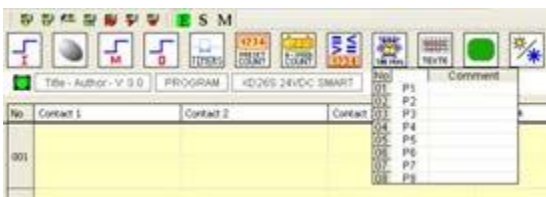
6. Counter Comparator



7. High-Speed Counter



8. Clocks

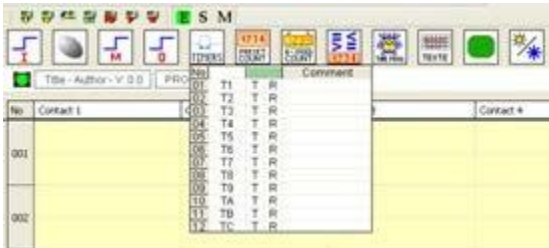


9. Change to Summer / Winter Time

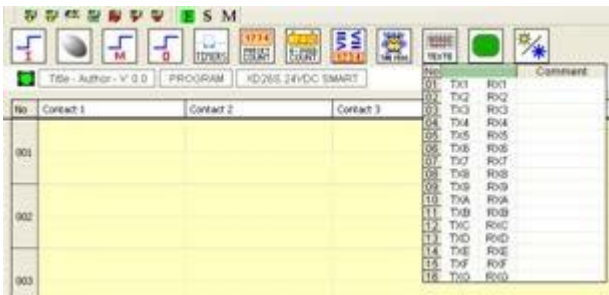
Programmable Logic Controller



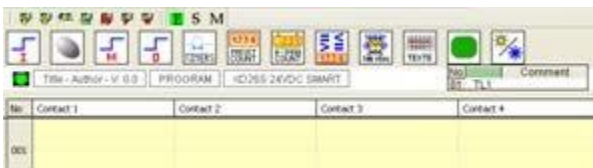
10. Timers



11. Texts



12. LCD Screen Backlighting



Discrete Inputs

Description

Discrete inputs can be used exclusively as contacts in the program. These contacts represent the status of the input of the controller connected to a sensor (pushbutton, switch, sensor, etc.). The contact number corresponds to the number of terminals of the associated input: 1 to 9, then A to R (except for letters I, M and O) according to the controller and any possible extension.

Access



This function is accessible from the **LD** function bar.

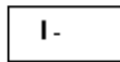
Programmable Logic Controller

Use as a Contact

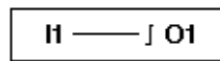
This contact may use the direct state of the input (normally open mode) or its inverse state (normally closed mode), see below.

Normally open mode:

Symbol of a normally open contact:



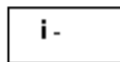
A normally open contact corresponds to the use of the direct state of the input. If the input is supplied, the contact is said to be conducting. For example: Switching a lamp on and off with a pushbutton.



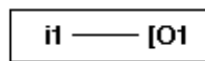
If input **1** is powered, contact **I1** is closed, and coil **Q1** is activated.

Normally closed mode:

Symbol of a normally closed contact:



A normally closed contact corresponds to the use of the reverse state (logical complement of the direct state) of the input. If the input is supplied, the contact is said to be non-conducting. For example: Controlling a lamp using an input in reverse state.



If input **1** is supplied, contact **i1** is open, and coil **Q1** is non-activated.

Modification of the Mode of a Contact

In the programming software, to modify the mode of a contact, simply position the cursor on it, then:

- With the mouse: Right-click to display a list of possible states (left-click to validate),
- With the space bar: Scroll through all possible states.

Initialization

Status of contacts on program initialization:

Programmable Logic Controller

- a) The direct state is inactive,
- b) The reverse state is active.

Discrete (DISCR) Outputs

Description

Discrete outputs correspond to the controller output relay coils (connected to the actuators). These outputs are numbered from 1 to 9, then from A to G, according to the controller and any extensions. Any Discrete output may be used, in the program, either as a coil or a contact.

Access



This function is accessible from the **LD** function bar.

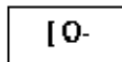
Use as a Coil

To use a Discrete output as a coil, four modes are available:

1. Contactor mode,
2. Impulse relay mode,
3. Latch mode,
4. Unlatch mode

1 Contactor mode:

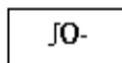
Symbol of a Discrete output, used as a coil in contactor mode:



The coil is energized if the contacts to which it is connected are conducting. Otherwise it is not energized.

2 Impulse relay mode:

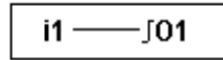
Symbol of a Discrete output, used as a coil in impulse relay mode:



Pulse energization, the coil changes state on each pulse it receives.

For example: Switching a lamp on and off with a pushbutton:

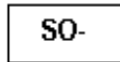
Programmable Logic Controller



A push button is connected to input **I1** and a lamp to output **Q1**. Every time the button is pressed, the lamp switches on or off.

3. Latch mode:

Symbol of a Discrete output, used as a coil in latch mode:

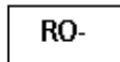


The **Set** coil, also called the latch coil, is energized as soon as the contacts to which it is connected are conducting, then stays set even if later the contacts are no longer conducting.

This behavior is identical to that of an RS logic flip-flop. For example: Switching a lamp on and off using two pushbuttons: See Latching Mode below.

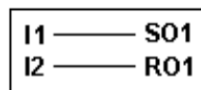
4. Unlatch mode:

Symbol of a Discrete output, used as a coil in latch mode:



The **RESET** coil, also called the unlatch coil, is deactivated when the contacts to which it is connected are conducting. It remains inactive even if later the contacts are no longer conducting.

For example: Switching a lamp on and off using two pushbuttons: See Unlatch Mode



BPI1 is connected to input I1 and BPI2 is connected to input I2. The lamp is controlled by output Q1. The lamp turns on when pushbutton BPI1 is pressed, and it turns off when pushbutton BPI2 is pressed.

Note: Rule for using outputs:

- a) An output must only be used at one single point in the program as a coil.
- b) If a SET coil is used for a Discrete output, it is advisable to provide a RESET coil for this output. The RESET coil takes priority over the SET coil. The use of a Set coil alone is only justified for activating an alarm signal that can be reset only by an INIT + ON action from the program.

Programmable Logic Controller

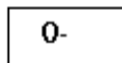
Note: For upward compatibility, the four types of modes for a given Q output coil or M auxiliary relay may be used in the same wiring diagram. In this case, the operating mode is determined by the first activated coil.

Use as a Contact

An output can be used as an auxiliary contact as many times as necessary. This contact may use the direct state of the output (normally open mode) or its inverse state (normally closed mode), see below.

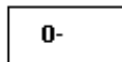
Normally open mode:

Symbol of a Discrete output, used as a contact in normally open mode:



An output used as a **normally open** auxiliary contact corresponds to the use of the direct state of the output. If it is **powered**, the contact is said to be **conducting**.

Normally closed mode:



Symbol of a Discrete output, used as a contact in normally closed mode:

An output used as a **normally closed** auxiliary contact corresponds to the use of the reverse state (logical complement of the direct state) of the output. If it is **powered**, the contact is said to be **nonconducting**.

Modification of the Mode of a Coil or a Contact

In the programming software, to modify the mode of a coil or a contact, simply position the cursor on the element then:

- a) With the mouse: Right-click to display a list of possible states (left-click to validate),
- b) With the space bar: Scroll through all possible states.

Initialization

Status of contacts on program initialization:

- a) Normally open mode (direct state) is inactive,
- b) Normally closed mode (reverse state) is active.

Save on Power Failure

Programmable Logic Controller

By default, after a power failure, the relay is in the state that corresponds to program initialization. To restore the state of the output backed up on a power failure, it is essential to enable save on power failure.

From the front panel: From the **PARAMETER** window, or

In the programming software: Enable the **Save on power failure** option in the parameters window associated with the output.

Auxiliary Relays

Description

Auxiliary relays marked **M** behave in exactly the same way as **O** Discrete outputs (see Discrete (DISCR) Outputs), but do not have an electrical output contact. They can be used as internal variables. There are 31, numbered from 1 to 9 and from A to Y except for letters I, M, O. Any auxiliary relay may be used, in the program, indifferently as coil or contact. They are used to memorize a state that will be used as the associated contact.

Access



This function is accessible from the **LD** function bar.

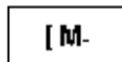
Use as a Coil

To use an auxiliary relay as a coil, 4 modes are available:

1. Contactor mode,
2. Impulse relay mode,
3. Latch mode,
4. Unlatch mode.

1. Contactor mode:

Symbol of an auxiliary relay, used as a coil in contactor mode:

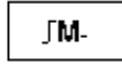


The relay is energized if the contacts to which it is connected are conducting. Otherwise it is not energized.

2. Impulse relay mode:

Symbol of an auxiliary relay, used as a coil in impulse relay mode:

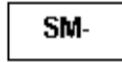
Programmable Logic Controller



Pulse energization, the coil changes state on each pulse it receives.

3. Latch mode:

Symbol of an auxiliary relay, used as a coil in latch mode:



The **SET** relay, also called the latch relay, is energized as soon as the contacts to which it is connected are conducting, then stays set even if later the contacts are no longer conducting.

This behavior is identical to that of an RS logic flip-flop.

4. Unlatch mode:

Symbol of an auxiliary relay, used as a coil in latch mode:



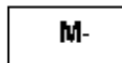
The **RESET** relay, also called the unlatch relay, is deactivated when the contacts to which it is connected are conducting. It remains deactivated even if later the contacts are no longer conducting. For upward compatibility, the 4 types of modes for a given Q output coil or M auxiliary relay may be used in the same wiring diagram.

Use as a Contact

Auxiliary relays may be used as contacts as many times as necessary. This contact may use the direct state of the relay (normally open mode) or its inverse state (normally closed mode),

Normally open mode:

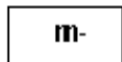
Symbol of an auxiliary relay, used as a contact in normally open mode:



A relay used as normally open contact corresponds to the use of the direct state of the relay. If it is powered, the contact is said to be conducting.

Normally closed mode:

Symbol of an auxiliary relay, used as a contact in normally closed mode:



Programmable Logic Controller

An auxiliary relay used as a normally closed contact, corresponds to the use of the reverse state(logical complement of the direct state) of the relay. If it is powered, the contact is said to be nonconducting.

Modification of the Mode of a Coil or a Contact

In the programming software, to modify the mode of a coil or a contact, simply position the cursor on the element then:

- a) With the mouse: Right-click to display a list of possible states (left-click to validate),
- b) With the space bar: Scroll through all possible states.

Initialization

Status of contacts on program initialization:

- a) Normally open mode (direct state) is inactive,
- b) Normally closed mode (reverse state) is active.

Save on power failure

By default, after a power failure, the relay is in the state that corresponds to program initialization. To restore the state of the output backed up on a power failure, it is essential to enable save on power failure.

From the front panel: From the PARAMETER window or In the programming software: Enable the Save on power failure option in the parameters window associated with the relay.

A/B Keys

Description

The A and B keys behave exactly like the **I** physical inputs (Discrete inputs). The only difference is that they do not correspond to the controller's connection terminals, but to the gray A and B buttons on the front panel. They are used as pushbuttons, and can only be used as contacts.

Access



This function is accessible from the **LD** function bar.

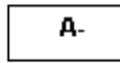
Use as a Contact

This contact may use the direct state of the key (normally open mode) or its inverse state (normally closed mode).

Programmable Logic Controller

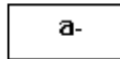
Normally open mode:

Symbol of the normally open contact, representing a key:



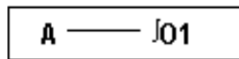
The normally open mode corresponds to the use of the direct state of the key. If the key is pressed, the corresponding input is said to be conducting.

Normally closed mode: Symbol of the normally closed contact, representing a key:



The **normally closed** mode corresponds to the use of the reverse state (logical complement of the direct state) of the key. If the key is pressed, the corresponding input is said to be non-conducting.

Example Creating an impulse relay operated by the **A** key and the **O1** output:



Each time the A key is pressed, the O1 output changes state.

Modification of the Mode of a Contact

In the programming software, to modify the mode of a contact, simply position the cursor on it, then:

- a) With the mouse: Right-click to display a list of possible states (left-click to confirm),
- b) With the space bar: Scroll through all possible states.

Initialization

Status of contacts on program initialization:

- a) Normally open mode (direct state) is inactive,
- b) Normally closed mode (reverse state) is active.

Entering a Contact or a Coil

This section describes the procedures for performing the following operations:

This is valid for either type of element: contact or coil, whether its parameters can be set or not.

1. Entering an element
2. Modifying an element
3. Deleting an element

Programmable Logic Controller

1. Entering an Element

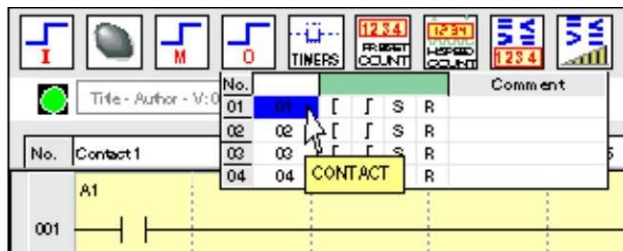
When entering an element, the following rules must be observed:

Contact: in any column except the last two

Coil: in the second-last column (the last column is reserved for comments)

Entry procedure:

1 Select the type of element required in the toolbar:



The list of available elements is displayed at the bottom of the screen. When the mouse is moved over one of the elements, the dialog box displays the list of available variables:

- The element number
- The element label
- The associated comment

2 Select the required element in the dialog box by placing the mouse pointer over it.

3 If necessary, enter a comment by clicking in the comment zone.

4 Left-click with the mouse.

5 Release the mouse button over the selected cell.

Deleting an Element

To delete an element, select the element then press one of the keys on the keyboard:

Del

Back space

Right-click/Clear

Control X

Entering a Link

This section describes the procedures for performing the following operations:

- Entering links between elements

Programmable Logic Controller

- b) Deleting links between elements
- c) Replacing a link with a contact

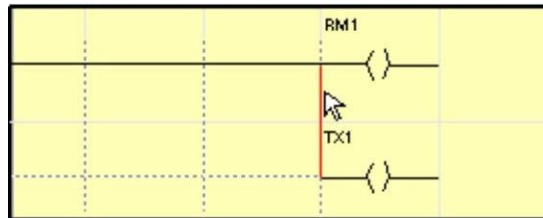
Entering a Link

Links are entered exclusively in cells framed by dotted lines.

1 Select the segment to transform, by placing the mouse pointer over it.

2 Left-click with the mouse: the validated segment turns red.

Illustration



3 Release the mouse button: The segment is created.

4 Connect the elements on the wiring sheet by clicking on the dotted lines that separate them.

Deleting a Link

To delete the links between elements, simply click again on the link.

Replacing a Link with a Contact

To replace a link with a contact, simply Follow the procedure for entering an element.

Place the contact over the segment to modify. This operation is only possible in cells reserved for contacts.

Automation Function Parameters

When entering a ladder diagram, the parameters of the configurable automation functions must be completed.

Once the automation function has been entered on the wiring sheet, double-click on it and the corresponding parameter setting window opens.

1. Discrete outputs
2. Auxiliary memories
3. Clocks
4. Timers
5. Counters

Programmable Logic Controller

6. High-speed counter
7. Counter comparators
8. Texts

This window has two tabs:


Parameters: these are the specific parameters associated with the variable

Comments: associated comments

Direct access

Once the automation function has been entered on the wiring sheet, double-click on it and the corresponding parameter setting window opens.

Access via the Settings View

The **Settings view** enables you to list all automation functions with parameters used in the application. This view is accessible in the edit window by clicking on the  button. The general interface allows the user to view all the information:

No: Function number

Function: Timer, Counter, etc.

Label: Identification of the function block on the wiring sheet

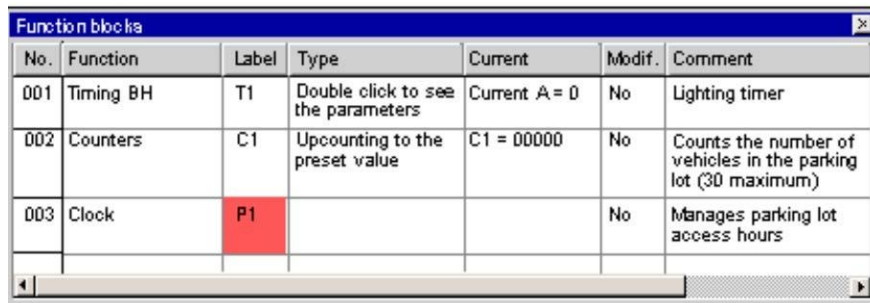
Parameters: The target value for a counter, etc.

Save on power failure: Indicates whether the Save on power failure option has been selected

Modification authorized: Indicates whether or not parameter modification is possible from the controller front panel

Comment : Comments associated with the function

Position: Position on the wiring sheet



No.	Function	Label	Type	Current	Modif.	Comment
001	Timing BH	T1	Double click to see the parameters	Current A = 0	No	Lighting timer
002	Counters	C1	Upcounting to the preset value	C1 = 00000	No	Counts the number of vehicles in the parking lot (30 maximum)
003	Clock	P1			No	Manages parking lot access hours

To adjust the various parameters, double-click on the desired line.

Programmable Logic Controller

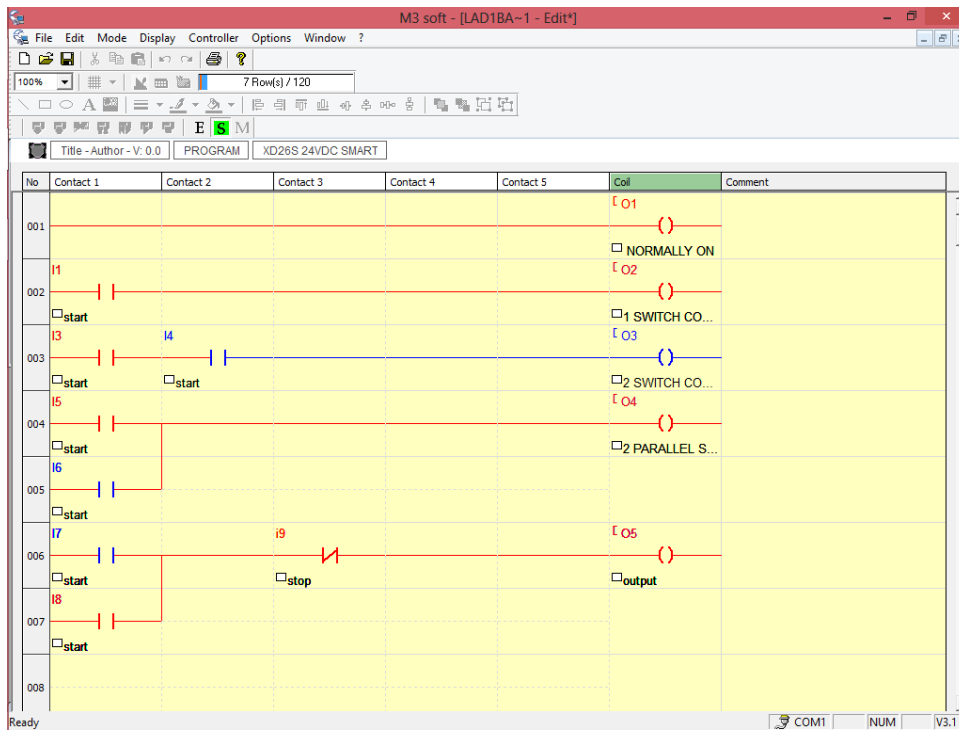
1. A program to illustrate start, stop, single switch, 2 switch, parallel switch and push button controls.

Program details:

1. Choose controller type and parameter window setting properly for input and output compatibility and validity as per requirement.
2. Make proper connection between elements, input-output to avoid invalidity and malfunction.
3. Input (8 elements), output (5 coils).

Hardware requirement:

1. Bulbs (5 No's)
2. Set of single stranded wires
3. Relay card



5.2 Implementation of logic gates and Boolean functions

The basic logic gates are as follows:

1. The NO logic,
2. The AND logic,
3. The OR logic,
4. The NO AND logic,
5. The NO OR logic,
6. The EXCLUSIVE OR logic.

1. The NO logic Function:

- a) If the input is inactive or not connected, the output is active.
- b) If the input is active, the output is inactive.

2. The AND logic Function:

- a) If all the inputs are active or not connected, the output is active.
- b) If at least one input is inactive, the output is inactive.

3. The OR logic Function:

- a) If at least one input is active, the output is active.
- b) If all the inputs are inactive or not connected, the output is inactive.

4. The NO AND logic Function:

- a) If at least one input is inactive, the output is active.
- b) If all the inputs are active or not connected, the output is inactive.

5. The NO OR logic Function:

- a) If all the inputs are inactive or not connected, the output is active.
- b) If at least one input is active, the output is inactive.

6. The EXCLUSIVE OR logic Function:

- a) If an input is inactive and the other input is active or not connected, the output is active.
- b) If both inputs are active or inactive or not connected, the output is inactive.

Programmable Logic Controller

2. A program to implement logic gates: AND, OR, NOT, NAND, NOR, X-OR and X-NOR

Program details:

1. Choose controller type and parameter window setting properly for input and output compatibility and validity as per requirement.
2. Make proper connection between elements, input-output to avoid invalidity and malfunction.
3. Input (5 elements), output (7 coils).

Hardware requirement:

1. Bulbs (7 No's)
2. Set of single stranded wires
3. Relay card



5.3 Study of PLC timer functions

Timers

The **TIMERS** function block provides access to the following types of timer:

Timer A/C is used to delay or prolong actions over a predetermined time:

Function A: Timer on-delay

Function C: Timer off-delay

Function A-C: Combination of functions A and C

Timer BW is used to create a pulse for the duration of a cycle on the output from an input edge.

Timer Li is used to create flashing (the durations of on and off states may be configured):

Function Li: The flashing cycle starts with an ON state.

Function L: The flashing cycle starts with an OFF state.

Timer B/H creates a pulse on the output of the rising edge of the input:

Function B: Regardless of the duration of the command pulse, the output is active for a duration that has been set.

Function H: The output is inactive at the end of a set time or on the falling edge of the command.

The **totaliser** creates a pulse on the output period during which the input was active reaches (one or more times) a set value.

Access



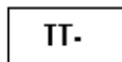
This function is accessible from the **LD** function bar.

Use of Coils

Each timer has two associated coils:

The use of these coils is described below.

Command input:

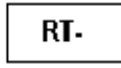


Symbol for the Command input coil of a timer:

Each type involves a specific operation, which can be used to manage all the possible scenarios in an application.

Reset input:

Programmable Logic Controller



Symbol for the Command input coil of a timer: Energization of the coil causes a reset of the current timer value: contact T is deactivated and the function is ready for a new timer cycle.

TT coil: Command input

RC coil: Reset input

Note: This coil is only necessary for pulsed on/off type timers.

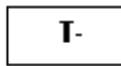
Use as a Contact

The contact associated with the timer indicates whether the timer has stopped.

It can be used as many times as necessary in the program. It can be used in one of 2 modes – normally open or normally closed - described below.

Normally open mode:

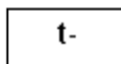
Symbol for the normally open contact associated with a timer:



The normally open contact corresponds to use of the direct state of the Timer function block output. If this output is **active**, the contact is said to be **conducting**.

Normally closed mode

Symbol for the normally closed contact associated with a timer:



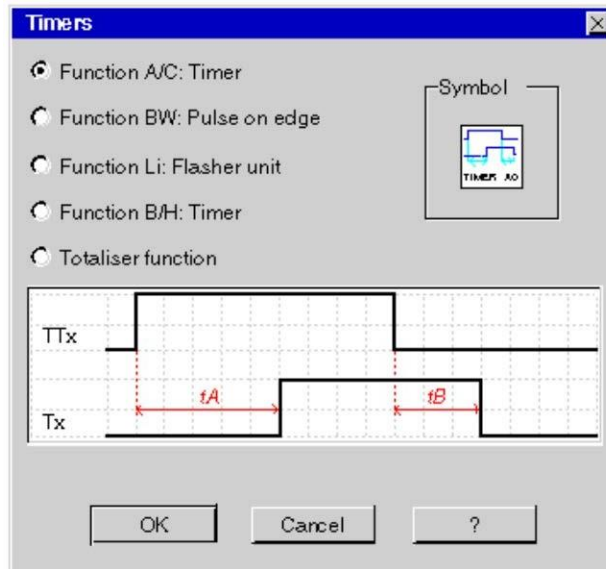
The normally closed contact corresponds to use of the reverse state (logical complement of the direct state) of the Timer function block output. If this output is **active**, the contact is said to be **nonconducting**.

Parameter Setting

Timer settings are described in the following sections:

1. **Timer A-C**
2. **Timer BW**
3. **Timer Li**
4. **Timer B/H**

5. Totaliser



Timing Diagrams

Timing diagrams are described in the following sections:

1. **Timer A-C**
2. **Timer Li**
3. **Timer B/H**
4. **Totaliser**

Modifying the Mode of a Coil or a Contact

In the programming workshop, to modify the mode of a coil or a contact, simply position the cursor on

the element then:

With the mouse: Right-click to display a list of possible states (left-click to confirm)

With the space bar: Scroll through all possible states

Initialization

State of the contacts and current values on initialization of the program:

The **normally open** mode (direct state) is **inactive**

The **normally closed** mode (inverse state) is **active**

the **current value(s)** is (are) **zero**

Programmable Logic Controller

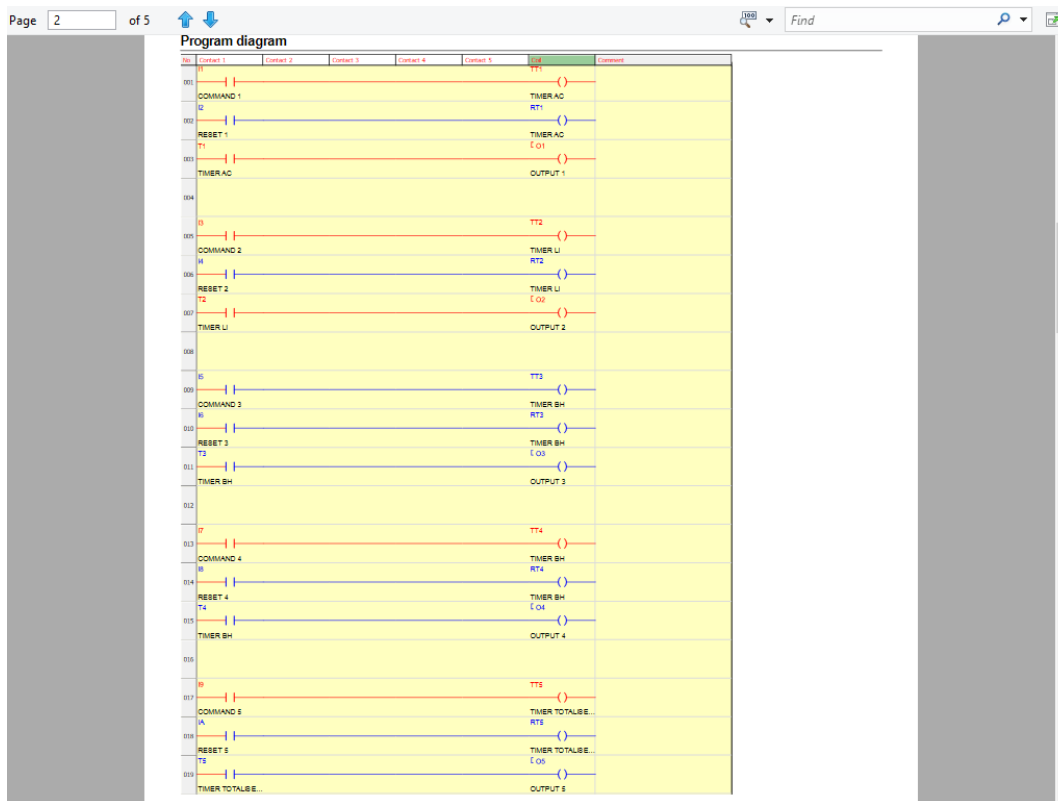
3. A program to study TIMERS: Timer AC, Timer LI, Timer BH and Totaliser

Program details:

1. Choose controller type and parameter window setting properly for input and output compatibility and validity as per requirement.
2. Make proper connection between elements, input-output to avoid invalidity and malfunction.
3. 15 elements, 5 timers and 15 coils.

Hardware requirement:

1. Bulbs (5 No's)
2. Set of single stranded wires
3. Relay card



Programmable Logic Controller

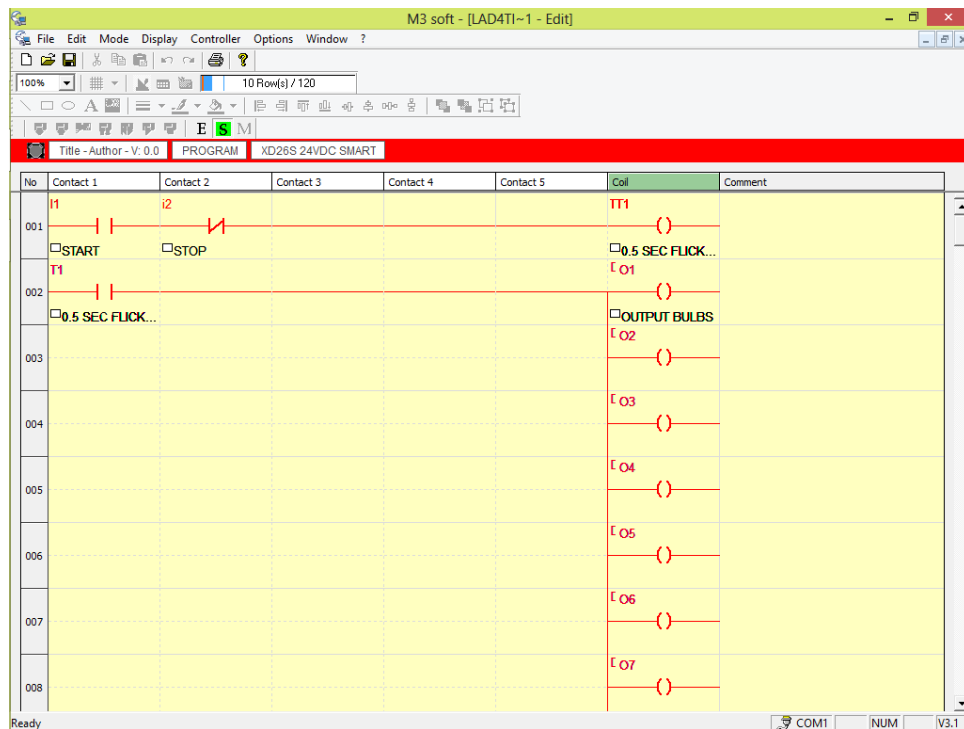
4. A program to implement timer to blink a set of bulbs periodically.

Program details:

1. Choose controller type and parameter window setting properly for input and output compatibility and validity as per requirement.
2. Make proper connection between elements, input-output to avoid invalidity and malfunction.
3. 3 elements, 1 timer and 7 coils.

Hardware requirement:

1. Bulbs (7 No's)
2. Set of single stranded wires
3. Relay card



5.4 Study of PLC counters functions

Counters

Description

The **Counter** function is used to upcount or downcount pulses. The controller has 16 counters, numbered from 1 to 9 then from A to G. The Counter function can be reset to zero or to the preset value (depending on the chosen parameter) during use. It can be used as a contact to find out whether: The preset value has been reached (**TO** upcounting mode)The counter has reached 0 (**FROM** downcounting mode)

Access



This function is accessible from the **LD** function bar.

Use of Coils

Each counter has 3 associated coils:

CC coil: Counting pulse input

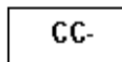
RC coil: Reset initial counter state input

DC coil: Counting direction input

The use of these coils is described below.

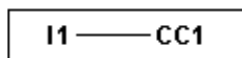
1. Counting pulse input:

Symbol for the Counting pulse input coil of a counter:



When used as a coil in a control diagram, this element represents a counter input for the function. Every time the coil is energized, the counter is incremented or decremented by 1 according to the chosen counting direction.

Example: Counting pulses delivered by the input by counter no. 1.

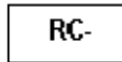


Every time input I1 is energized, the counter no. 1 is incremented by 1.

2. Reset initial state input:

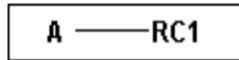
Programmable Logic Controller

Symbol for the Reset initial counter state input coil:



When used as a coil in a control diagram, this element represents an input that resets the counter function to its initial state. Energizing the coil has the following effect:

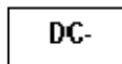
Example: Counter No.1 reset to zero on pressing key A.



Every time key A is pressed, the counter starts from 0.

3. Counting direction input:

Symbol for the Counting direction input coil of a counter:

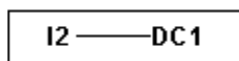


This input determines the counting direction according to its state. The counter: Resets the current counter value to **zero** if the count type is **TO** (upcounting from the preset value). Reset the current value to the **preset value** if the count type is **FROM** (downcounting from the preset value).

Down counts if the coil is energized.

Up counts if the coil is not energized.

Note: By default, if this input is not hard-wired, the automation function upcounts. Example: Up/down counting, depending on the state of controller input I2. If input **I2** is active, the automation function down counts.



Use as a Contact

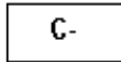
The contact associated with the high-speed counter indicates whether the preset value (**TO** mode) or zero (**FROM** mode) has been reached.

It can be used as many times as necessary in the program. It can be used in one of 2 modes: normally open or normally closed, described below.

Normally open mode:

Symbol of the normally open contact associated with a counter:

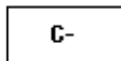
Programmable Logic Controller



This contact is **conducting when**:

- a) The current counter value **has reached** the preset value, if the counter is in **TO** mode (upcounting mode).
- b) The current counter value **is equal to 0**, if the counter is in **FROM** mode (downcounting mode).

Normally closed mode:



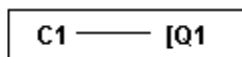
Symbol of the normally closed contact associated with a counter:

The contact is **conducting as long as** :

- a) The current counter value **has reached** the preset value, if the counter is in **TO** mode (upcounting mode).
- b) The current counter value **is equal to 0**, if the counter is in **FROM** mode (downcounting mode).
- c) The current counter value **has not reached** the preset value, if the counter is in **TO** mode (upcounting mode).
- d) The current counter value **does not equal 0**, if the counter is in **FROM** mode (downcounting mode).

Example: Lighting an LED connected to the counter No. 1 output (**TO** mode).

When the preset value has been reached: the LED is lit; otherwise it is off.



Parameter Setting from the Programming Workshop

Pulses:

This value is between 0 and 32767 (preset value).

Type of counting:

Two modes are available:

1. Upcounting to the preset value: incrementation of the current counter value

Programmable Logic Controller

2. Downcounting from the preset value: decrementation of the current counter value

Save on power failure:

By default, after a power cut, the counter is set to the state that corresponds to program initialization. To restore the counter status that is saved on power failure, it is essential to check the Enable save on power failure option.

Modification authorized:

By default, parameters can only be modified from the controller front panel. To be able to modify parameters from the controller front panel using the PARAMETERS menu, check the **Modification authorized** option.

Parameter Setting from the Front Panel

The block parameters can be accessed and set from the **PARAMETERS** menu if the **Modification authorized** option has been selected. The only modifiable parameter is the **Counting setpoint**, i.e. the value up to or down from which the counter counts.

Current Counter Value

The current counter value is the value at any given time resulting from the successive up/downcounting actions that have occurred since the last time the counter was reset to its initial state. This value is between 0 and 32767. Once these values have been reached, downcounting will leave the current value at 0 and upcounting will leave the current value at + 32767.

Modifying the Mode of a Coil or a Contact

In the programming workshop, to modify the mode of a coil or a contact, simply position the cursor on

the element then:

1. With the mouse: Right-click to display a list of possible modes (left-click to confirm)
2. With the space bar: scroll through all possible states

Initialization

State of the contacts and current value on initialization of the program:

1. The **normally open** mode (direct state) is **inactive**
2. The **normally closed** mode (inverse state) is **active**

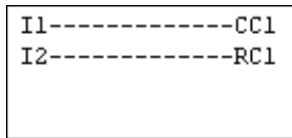
Programmable Logic Controller

3.

The **current value** is zero

Below, three simple examples of use of a counter (configured in upcounting to the preset value mode):

Screen 1

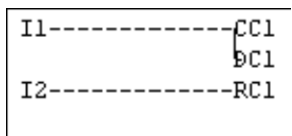


Upcounting and resetting:

The counter is incremented each time input **I1** is activated.

The counter is reset each time input **I2** is activated.

Screen 2

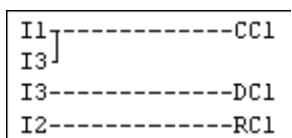


Downcounting and resetting:

The counter is decremented each time input **I1** is activated.

The counter is reset each time input **I2** is activated.

Screen 3



Upcounting, downcounting and resetting:

The counter is incremented each time input **I1** is activated. The counter is decremented each time input **I3** is activated. The counter is reset each time input **I2** is activated

Programmable Logic Controller

5. A program to activate bulbs implementing counter.

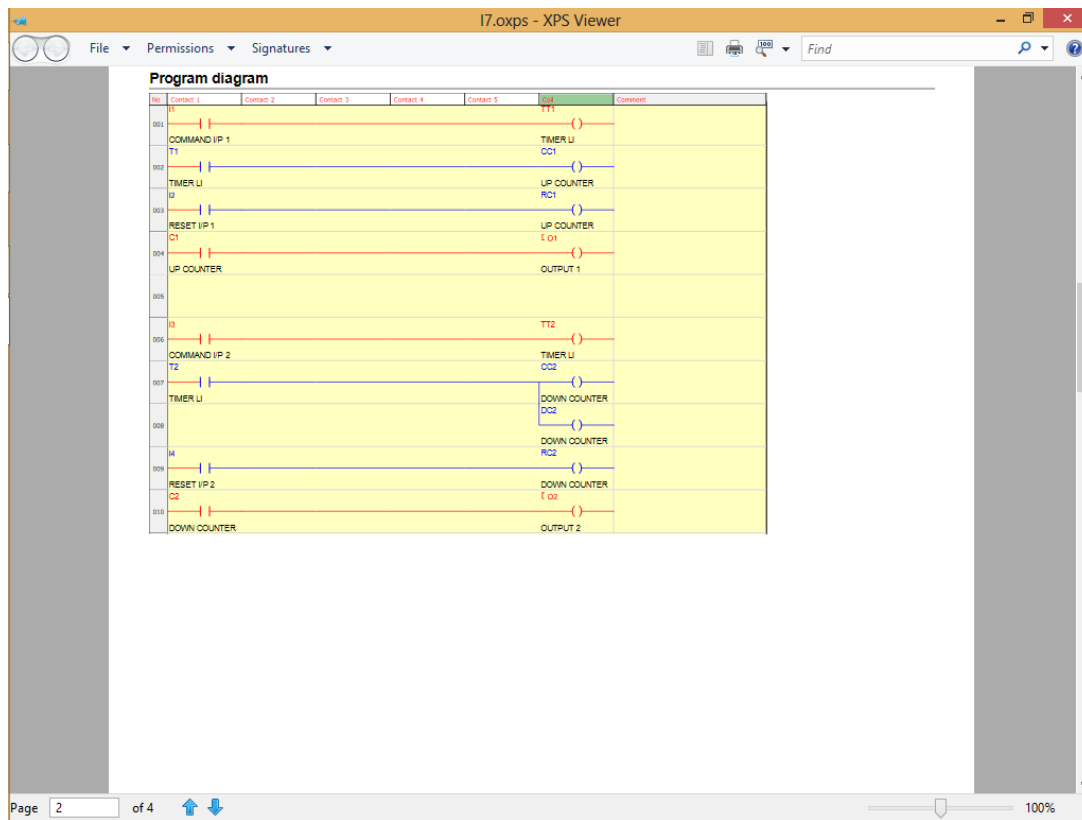
Program details:

1. Choose controller type and parameter window setting properly for input and output compatibility and validity as per requirement.
2. Make proper connection between elements, input-output to avoid invalidity and malfunction.
3. 8 elements, 2 counter (1 up counter, 1 down counter), 2 timer and 9 coils.

Hardware requirement:

1. Bulbs (2 No's)
2. Set of single stranded wires
3. Relay card

Programmable Logic Controller



6. A program to implement counter to activate a bulb after a count has reached.

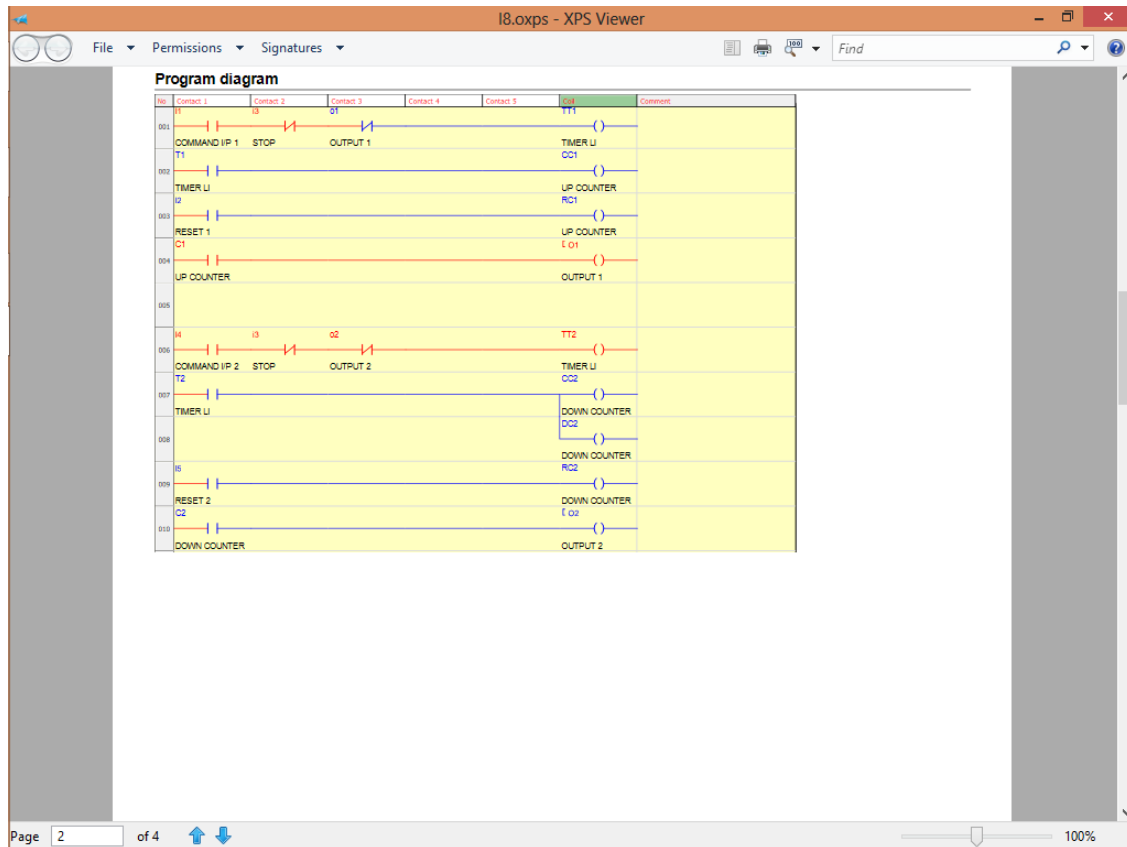
Program details:

1. Choose controller type and parameter window setting properly for input and output compatibility and validity as per requirement.
2. Make proper connection between elements, input-output to avoid invalidity and malfunction.
3. 12 elements, 2 counter (1 up counter,1 down counter), 2 timer and 9 coils.

Hardware requirement:

1. Bulbs (2 No's)
2. Set of single stranded wires
3. Relay card

Programmable Logic Controller



5.5 Study of Number Comparison functions

Counter Comparator

Description

This function is used to compare the current counter values of two counters or of a counter and a constant value.

Note: The **Counter comparator** function block can only be configured from the programming workshop in **Program view** mode.

Access



This function is accessible from the **LD** function bar.

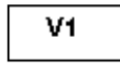
Use as a Contact

The counter comparator indicates whether the chosen condition has been verified. It is used as a contact, in normally open mode or in normally closed mode.

Normally open mode:

Programmable Logic Controller

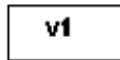
Counter comparator symbol, in normally open mode:



The contact is **conducting** when the condition has been **verified**.

Normally closed mode:

Counter comparator symbol, in normally closed mode:



The contact is **conducting** when the condition **has not been verified**.

Parameter Setting from the Programming Workshop

The different parameters to fill in are the following:

1. Comparison formula
2. Parameter modification authorized

1. Comparison formula:

The comparison formula is as follows:

$$Cx + x < \text{Comparison Operator} > Cy + y$$

Where:

1. **Cx and Cy**: Represent the counters to compare; these are selected from the associated drop-down menu
2. **x and y**: These are constants (offset) between: - 32,768 and 32,767

The **comparison operators** that may be chosen are the following:

Symbol with description

- > Greater than.
- ≥ Greater than or equal to.
- = Equal to.
- ≠ Different.
- ≤ Less than or equal to.

Programmable Logic Controller

< Less than.

Modification authorized:

By default, parameters can only be modified from the controller front panel. To be able to modify parameters from the controller using the PARAMETERS menu, check the **Modification authorized** option.

Modifying the Mode of a Coil or a Contact

In the programming workshop, to modify the mode of a contact, simply position the cursor on it, then:

- a) With the mouse: Right-click to display a list of possible states (left-click to confirm)
- b) With the space bar: Scroll through all possible states

Initialization

State of contacts on program initialization:

Normally open mode (direct state) is inactive

Normally closed mode (reverse state) is active

7. A program to illustrate compare function and generate output accordingly.

Program details:

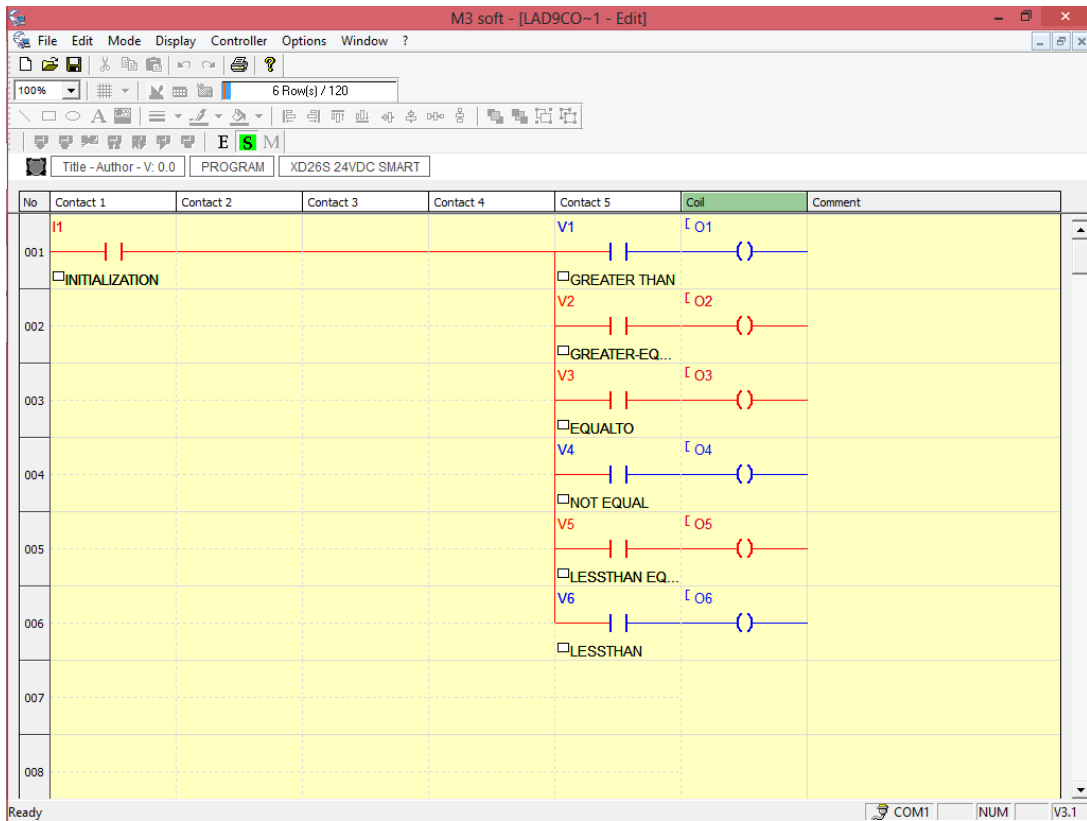
1. Choose controller type and parameter window setting properly for input and output compatibility and validity as per requirement.
2. Make proper connection between elements, input-output to avoid invalidity and malfunction.
- 3.1 elements, 6 compare function and 6 coils.

Hardware requirement:

1. Bulbs (6 No's)
2. Set of single stranded wires

Programmable Logic Controller

3. Relay card



8. A program to implement timer, counters and compare with off set values .

Program details:

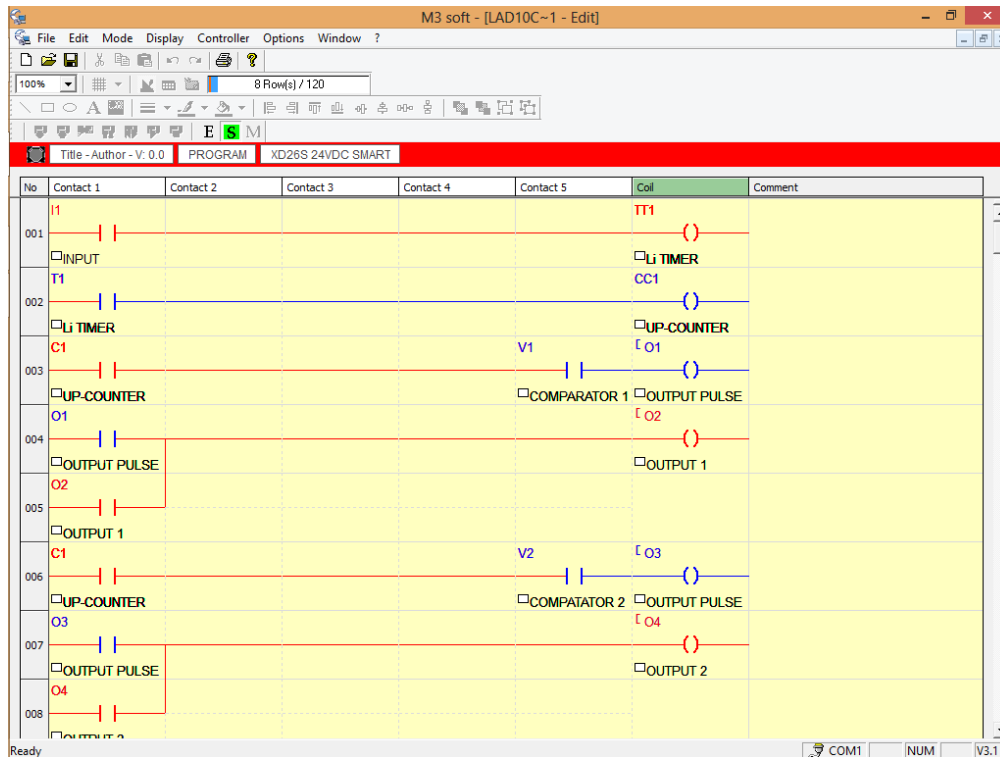
1. Choose controller type and parameter window setting properly for input and output compatibility and validity as per requirement.
2. Make proper connection between elements, input-output to avoid invalidity and malfunction.
3. 10 elements, 1 timer, 1 counter, 1 compare function and 6 coils.

Hardware requirement:

1. Bulbs (2 No's)
2. Set of single stranded wires

Programmable Logic Controller

3. Relay card



9. A program to implement timer, counters and compare with off set values .

Program details:

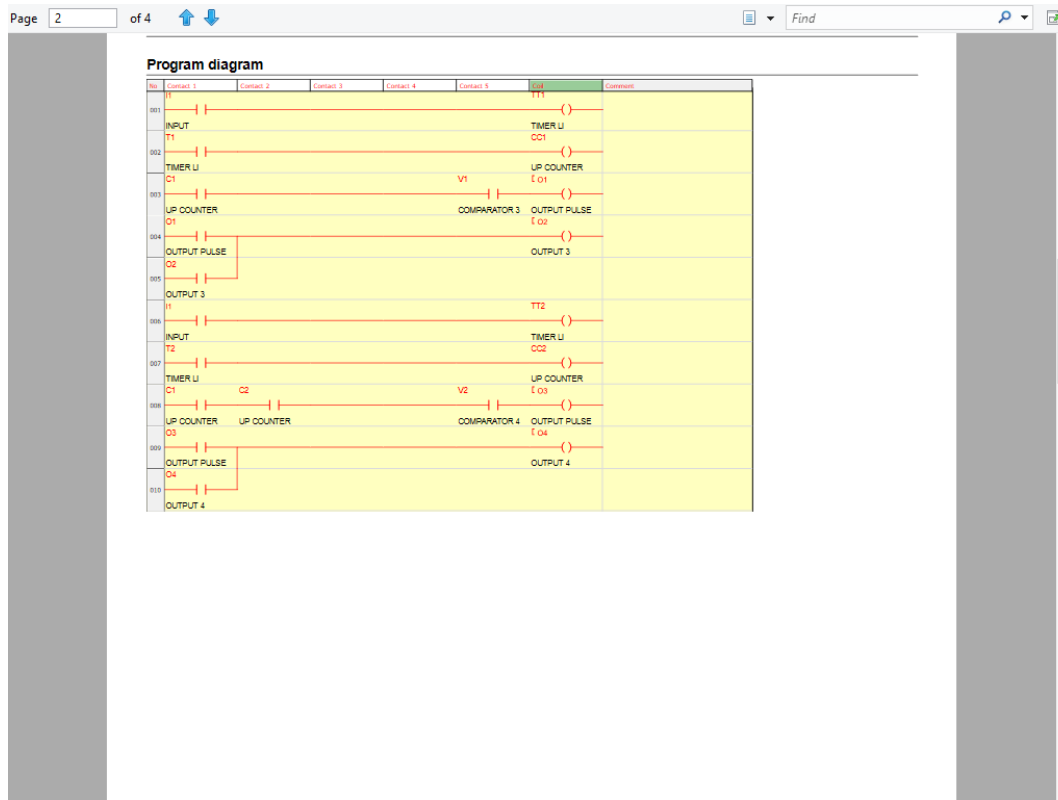
1. Choose controller type and parameter window setting properly for input and output compatibility and validity as per requirement.
2. Make proper connection between elements, input-output to avoid invalidity and malfunction.
3. 10 elements, 1 timer, 1 counter, 1 compare function and 6 coils.

Hardware requirement:

1. Bulbs (2 No's)
2. Set of single stranded wires

Programmable Logic Controller

3. Relay card



5.6 Applications

5.6.1 Motor control using PLC

P10. A program to illustrate latching and multiple start –single stop control of motor.

P11. A program to illustrate schemes for switching of motor.

P12. A program to illustrate schemes for switching of motor.

5.6.2 Sequential lighting of bulbs

P13. A program to implement timer to turn ON /OFF 3 bulbs sequentially.

P14. A program to implement timer to turn ON /OFF 6 bulbs sequentially.

10. A program to illustrate latching and multiple start –single stop control of motor.

Program details:

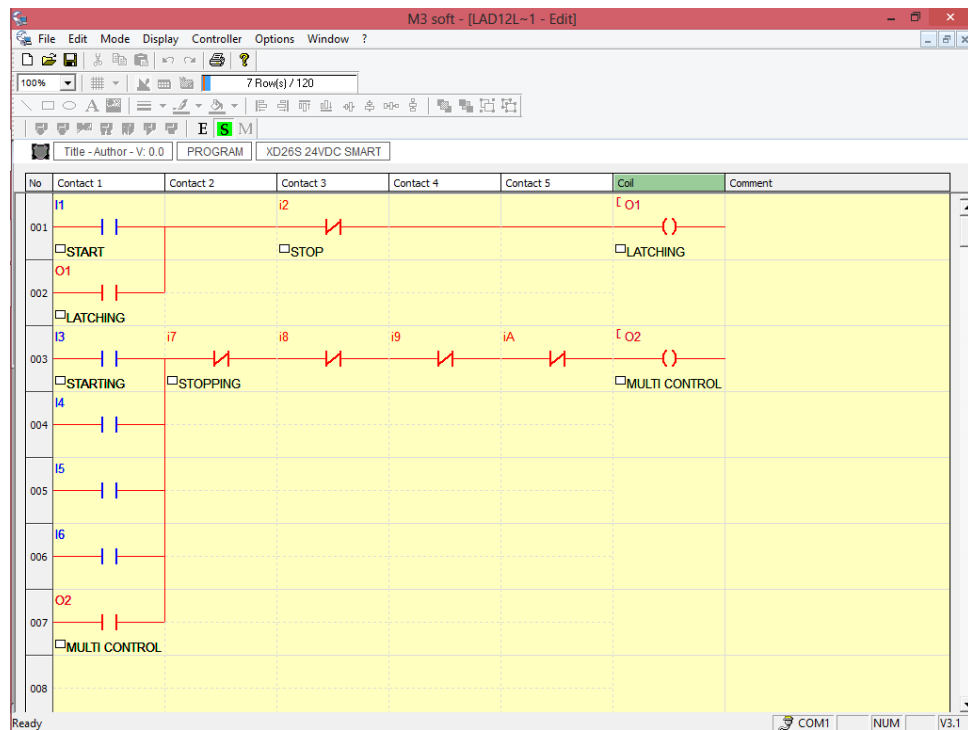
1. Choose controller type and parameter window setting properly for input and output compatibility and validity as per requirement.
2. Make proper connection between elements, input-output to avoid invalidity and malfunction.
3. 12 elements, 2 coils.

Hardware requirement:

1. Bulbs (2 No's)
2. Set of single stranded wires

Programmable Logic Controller

3. Relay card



11. A program to illustrate schemes for switching of motor.

Program details:

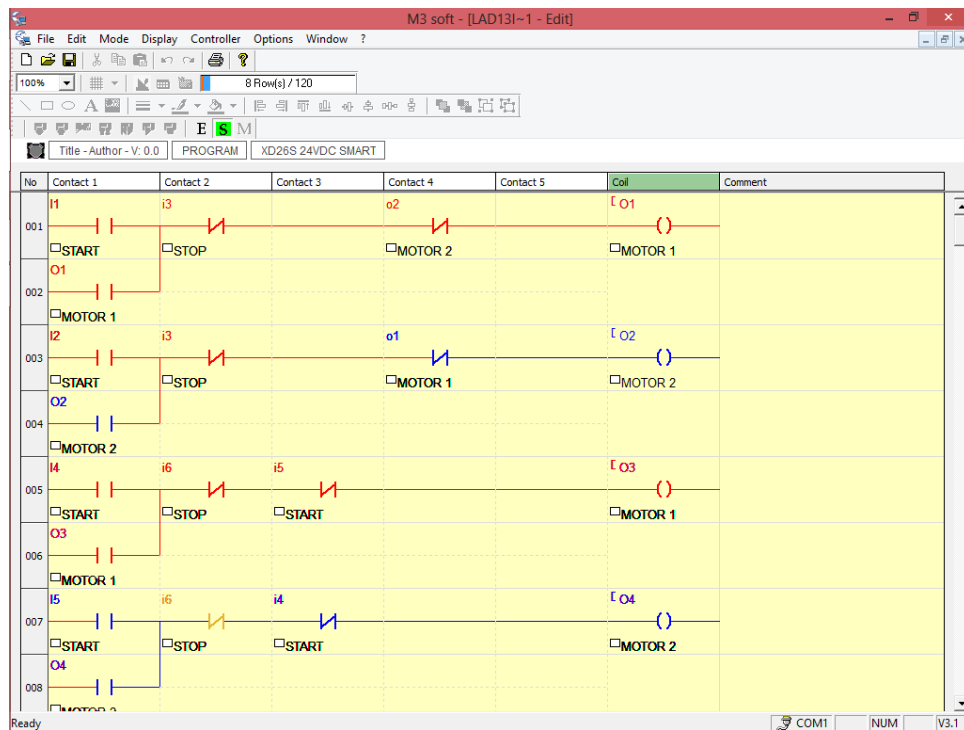
1. Choose controller type and parameter window setting properly for input and output compatibility and validity as per requirement.
2. Make proper connection between elements, input-output to avoid invalidity and malfunction.
3. 16 elements, 4 coils.

Hardware requirement:

1. Bulbs (4 No's)
2. Set of single stranded wires

Programmable Logic Controller

3. Relay card



12. A program to illustrate schemes for switching of motor.

Program details:

1. Choose controller type and parameter window setting properly for input and output compatibility and validity as per requirement.
2. Make proper connection between elements, input-output to avoid invalidity and malfunction.
3. 32 elements, 7 coils.

Hardware requirement:

1. Bulbs (7 No's)
2. Set of single stranded wires

Programmable Logic Controller

3. Relay card



Sequential lighting of bulbs

13. A program to implement timer to turn ON /OFF 3 bulbs sequentially.

Program details:

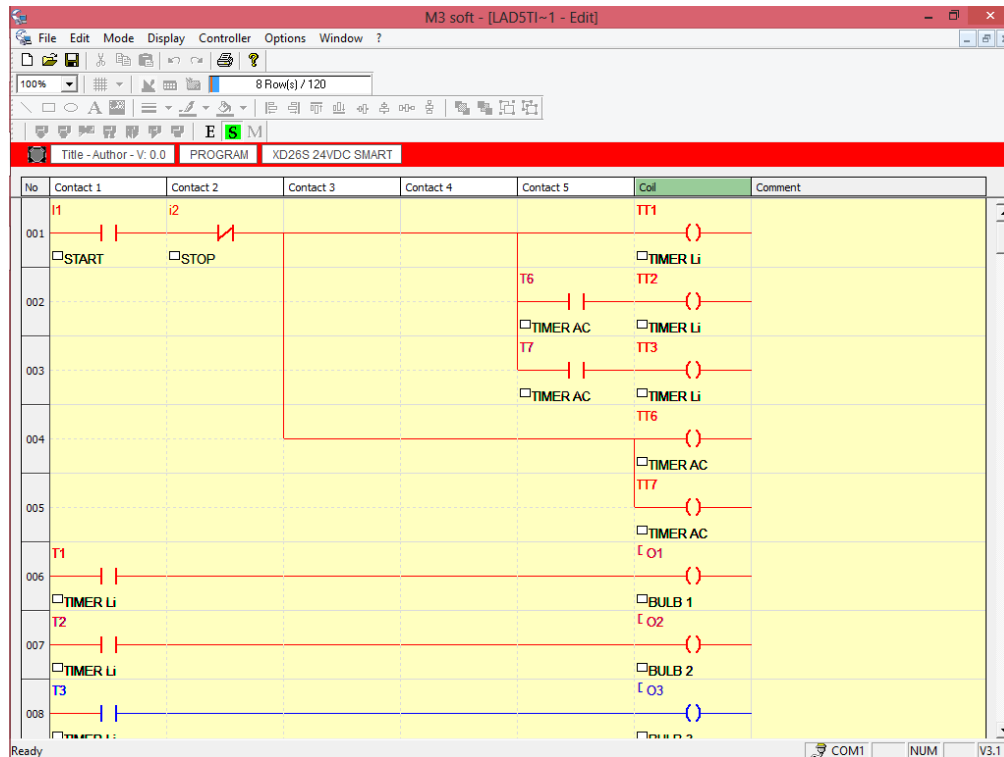
1. Choose controller type and parameter window setting properly for input and output compatibility and validity as per requirement.
2. Make proper connection between elements, input-output to avoid invalidity and malfunction.
3. 7 elements, 5 timer and 8 coils.

Hardware requirement:

1. Bulbs (3 No's)
2. Set of single stranded wires

Programmable Logic Controller

3. Relay card



14. A program to implement timer to turn ON /OFF 6 bulbs sequentially.

Program details:

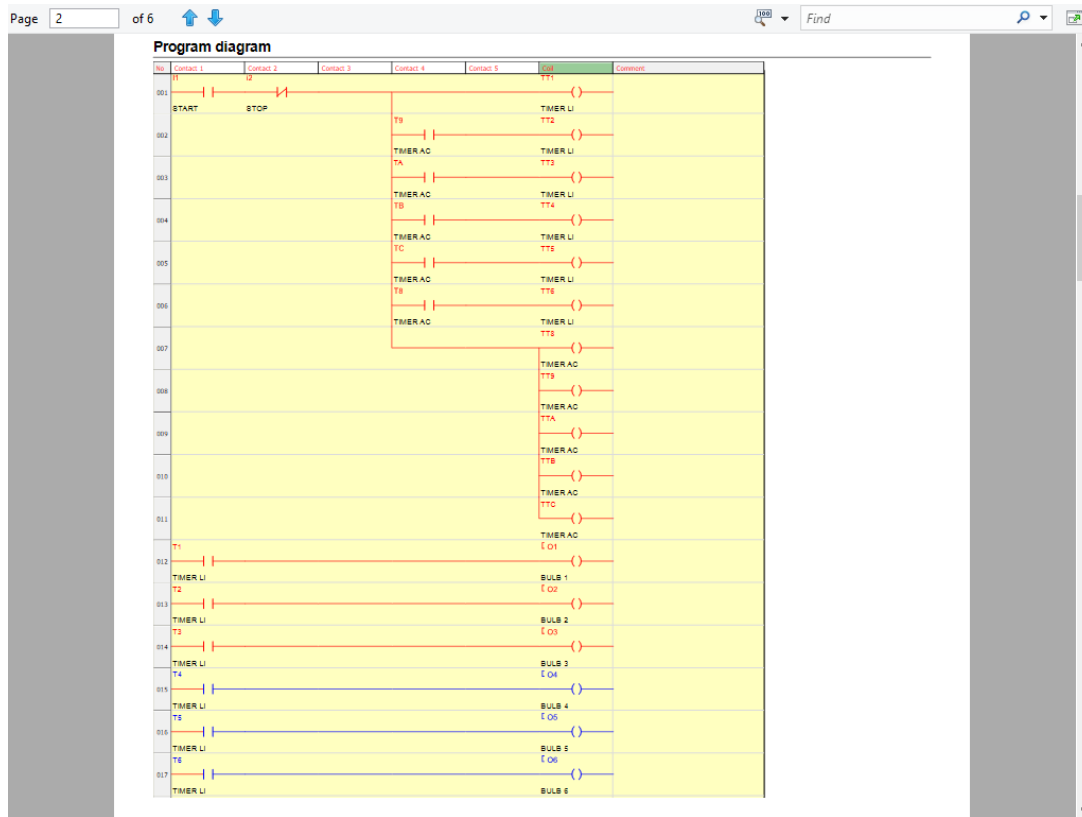
1. Choose controller type and parameter window setting properly for input and output compatibility and validity as per requirement.
2. Make proper connection between elements, input-output to avoid invalidity and malfunction.
3. 13 elements, 11 timer and 17 coils.

Hardware requirement:

1. Bulbs (3 No's)
2. Set of single stranded wires

Programmable Logic Controller

3. Relay card



For any queries kindly contact
Dr.M.Chakravarthy
hodeee@staff.vce.ac.in
Cell No: 9849979136

