



PDHonline Course E448 (9 PDH)

PLC & Fieldbus Control Systems - Automation Guidebook, Part 4

Instructor: Jurandir Primo, P.E.

2020

PDH Online | PDH Center

5272 Meadow Estates Drive
Fairfax, VA 22030-6658
Phone: 703-988-0088
www.PDHonline.com

An Approved Continuing Education Provider

**PLC & FIELDBUS CONTROL SYSTEMS
AUTOMATION GUIDEBOOK – PART 4**

CONTENTS:

- 1. PLC HISTORY**
- 2. INTRODUCTION**
- 3. INPUT/OUTPUT (I/O) CAPABILITIES**
- 4. LADDER LOGIC PROGRAMS**
- 5. HUMAN-MACHINE INTERFACE (HMI)**
- 6. HOW THE PLCs WORK**
- 7. FIELDBUS CONTROL SYSTEMS**
- 8. DCS AND FIELDBUS DEVELOPMENT**
- 9. FIELDBUS MODELS AND NETWORK STRUCTURE**
- 10. DIAGNOSTICS & COMMUNICATION INTERFACES**
- 11. BASIC COMMUNICATION STANDARDS**
- 12. REFERENCES & LINKS**

OBS.: This is a didactic and professional handbook. It's highly recommended to downloading and printing the course content for your study, before answering the quiz questions.

1. PLC HISTORY

The early history of the PLC (Programmable Logic Controller) goes back to the 1960's when control systems were still handled using relay control. During this time the control rooms consisted of several walls containing many relays, terminal blocks and mass of wires.

In 1968 Bill Stone, who was part of a group of engineers at the Hydramatic Division of General Motors Corporation, presented a paper at the Westinghouse Conference outlining their problems with reliability and documentation for the machines at this plant. He also presented a design criteria developed by the GM engineers for a "standard machine controller", to eliminate costly scrapping of assembly-line relays.

These specifications along with a proposal request to build a prototype were given to four control builders:

- Allen-Bradley, by way of Michigan-based Information Instruments, Inc.
- Digital Equipment Corporation (DEC).
- Century Detroit.
- Bedford Associates.

Considering the proposal request, the team of Digital Equipment brought a "mini-computer" into GM, which was rejected for many reasons, from which static memory was one of its serious limitations.

Allen-Bradley was at the time, already well known for its rheostats, relays and motor controls, responded at the risk of competing with one of its most successful core business, which were the electromechanical relays. The first attempt, the PDQ-II or "Program Data Quantifier", was considered too large, too complex and too hard to program. The second attempt, the PMC or "Programmable Matrix Controller", was smaller and easier to program, but yet not able to fully serve the customer needs for machine controls.

The Bedford Associates, with Richard Morley, Mike Greenberg, Jonas Landau, George Schwenk and Tom Boissevain, were already working on the design of a unit, named as 084, since it was the 84th project for the company, whose characteristics included a modular and rugged design, using no interrupts for processing, as well as, direct mapping into memory.

After finding some financial support, the team decided to form a new company called Modicon (Modular Digital CONTroller) working closely with Bedford to create the controller, and build the 084, designated as the programmable controller (PC).

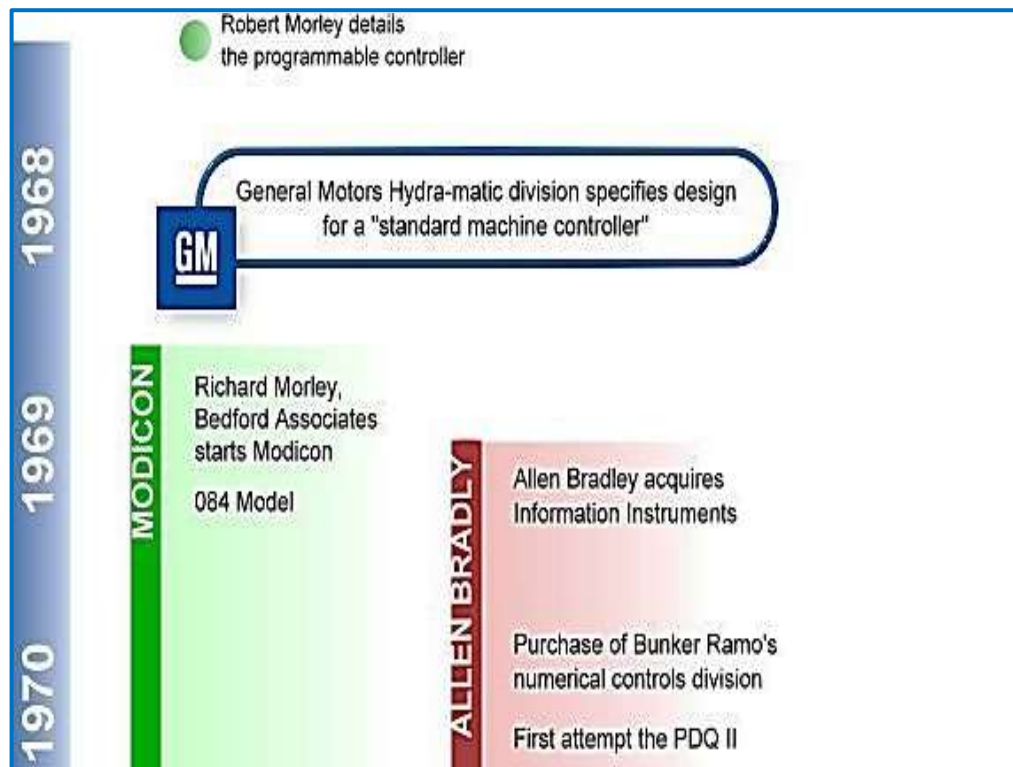
Finally in 1969, the winning proposal came from Bedford Associates and Modicon, when they demonstrated the Modicon 084 solid-state sequential logic solver, in GM, consisted of three distinct components including the processor board, the memory, and the logic solver board, to calculate the dominant algorithms associated with the ladder logic.

By 1971, Odo Struger and Ernst Dummermuth engineers of Allen-Bradley, had begun to develop a new concept that included improvements based on the customer needs. Allen-Bradley named this new device as the “Programmable Logic Controller” (PLC), over the then accepted term “Programmable Controller”. The PLC terminology became the industrial standard especially when the PC became associated with personal computers. In 1985, Rockwell Automation acquired Allen-Bradley. The PLC name is still today, associated to Allen-Bradley.

PLC Evolution: The early 1980s saw a cross pollination between PLCs and DCSs (Distributed Control Systems). Building on these trends, software companies appeared, and during the 90s, standardization and open systems were the main themes. Ethernet peer-to-peer networking became available from virtually all PLC manufacturers, with EEPROM and Flash memories that replaced the EPROMs of the 1980s.

Personal Computers (PCs) and Cathode Ray Tubes (CRTs) in general, became accepted and started to replace switches and lights on control system panels. Small PLCs called “Bricks” were introduced to the marketplace. The first few years of the 21st century have seen a consolidation of PLC manufacturers, and Safety PLCs featuring triple redundancy were introduced. Very small nano or pico PLCs, some as small as industrial relays, have appeared. Nowadays, the LCD base operator interface panels have largely displaced the CRTs, especially on the plant floor.

There is much more to say about the history of the PLC, but we have only focused on the two main vendors, which saw the birth of the PLC and kept still on the market: Modicon as part of Schneider Electric, and Allen-Bradley as part of Rockwell Automation.



2. INTRODUCTION:

Programmable Logic Controller or PLC is a general-purpose controller, applicable to many different types of process control applications. The word “programmable” in its name reveals just why PLCs are so useful: the end-user is able to program, or instruct, the PLC to do virtually any control function imaginable. Unlike PID loop controllers, which are special-purpose devices intended to perform a single type of control function, a PLC may be instructed to do almost anything with the signals it receives from input devices.

The re-programmability of a PLC also meant changes could be implemented to the control system strategy must easier than with relay circuits, where re-wiring was the only way to alter the system's function. Additionally, the computer-based nature of a PLC meant that process control data could now be communicated by the PLC over networks, allowing process conditions to be monitored in distant locations, and by multiple operator stations.

Programmable Logic Controllers are essentially nothing more than special-purpose, industrial computers. As such, they are built far more ruggedly than an ordinary personal computer (PC), and designed to run extremely reliable operating system software. Typical devices that may be connected to PLC's inputs include hand switches, process switches, sensors, analog transmitters (4-20 mA), thermocouples, thermistors, and strain gauges. Typical devices connecting to a PLC's outputs include electric lamps, solenoids, relay coils, motor contactors, analog final control elements (e.g. throttling control valves, variable-speed motor drives), and audible buzzers.

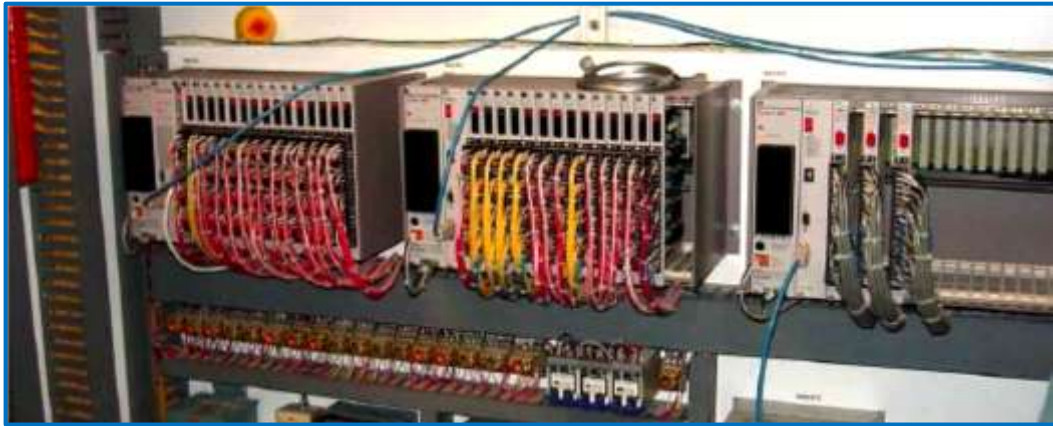
PLCs, as a rule, do not contain disk drives, cooling fans, or any other moving parts. This is an intentional design decision, intended to maximize the reliability of the hardware in harsh industrial environments, where the PLC chassis may be subjected to extreme temperature, vibration, humidity, and airborne particulates (dust, fibers, and/or fumes).

Large PLC systems consist of a rack into which circuit “cards” are plugged. These cards include processors, input and output (I/O) points, communications ports, and other functions necessary to the operation of a PLC system. Such “modular” PLCs may be configured differently according to the specific needs of the application. Individual card failures are also easier to repair in a modular system, since only the failed card need be replaced, not all the cards or the whole card rack.

Small PLC systems consist of a monolithic “brick” containing all processor, I/O, and communication functions. These PLCs are typically far less expensive than their modular cousins, but are also more limited in I/O capability and must be replaced as a whole in the event of failure. The following photographs show several examples of real PLC systems, some modular and some monolithic. These selections are not comprehensive by any means, as there are many more manufacturers and models of PLC than those I have photographed. They do, however, represent some of the more common brands and models in current industrial use.

The programming language of PLCs, as relay-replacements, is a graphical convention known as Ladder Diagram. This PLC program is called Ladder Diagram because it resembles ladder-style electrical schematics, where vertical power “rails” convey control power to a set of parallel “rung”

circuits containing switch contacts and relay coils. While Ladder Diagram programming definitely has limitations compared to other computer programming languages, it is relatively easy to learn and diagnose, which is why it remains popular as a PLC programming language today.



The example above is a modular Siemens PLC, installed in a control panel of a municipal wastewater treatment plant with individual processor, I/O, and communication cards plugged into a rack. Three racks appear in this photograph (two completely filled with cards, and the third only partially filled). The programming of these PLCs involves timers, counters, sequencers, and other functions to properly manage the continuous operation of the modular racks.

Some of the inputs to this PLC include water level switches, pressure switches, water flow meters, and conductivity meters (to measure the purity of the water, greater electrical conductivity indicating the presence of more dissolved minerals, which is a bad thing for this particular process application). In turn, the PLC controls the starting and stopping of water pumps and the switching of water valves to manage the water purification and storage processes.

The power supply and processor card for each rack is located on the left-hand end, with I/O cards plugged into slots in the rest of the rack. Input devices such as switches and sensors connect by wire to terminals on input cards, while output devices such as lamps, solenoids, and motor contactor coils connect by wire to terminals on output cards.

One of the benefits of modular PLC construction is that I/O cards may be changed out as desired, altering the I/O configuration of the PLC as needed. If, for example, the PLC needs to be configured to monitor a greater number of sensors, more input cards may be plugged into the rack and subsequently wired to those sensors. Or, if the type of sensor needs to be changed – perhaps from a 24 volt DC sensor to one operating on 120 volts AC – a different type of input card may be substituted to match the new sensor(s).

A modern PLC manufactured by Allen-Bradley (Rockwell) is this ControlLogix 5000 system, shown below, is used to control a cereal manufacturing process. The modular design of the ControlLogix 5000 system follows the more traditional scheme of individual cards plugged into a rack of fixed size. The photograph shown aside is a semi-modular PLC design, which maybe the least expensive PLC on the market is the Koyo “CLICK” PLC series, the processor module with eight discrete input

and six discrete output channels, and with free programming software. This PLC has a minimum of input/output (I/O) channels built into the processor module, but having the capacity to accept multiple I/O modules plugged in to the side, much like the Siemens S7-300 PLC.



ControlLogix 5000 system

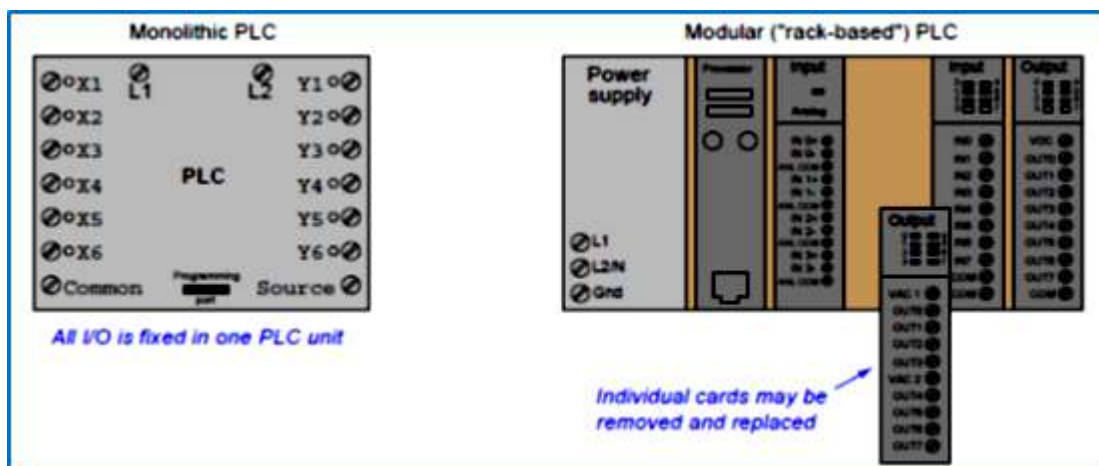


Koyo "CLICK" PLC series

3. INPUT/OUTPUT (I/O) CAPABILITIES:

Every Programmable Logic Controller must have some means of receiving and interpreting signals from real-world sensors such as switches, and encoders, and also be able to effect control over real-world control elements such as solenoids, valves, and motors. This is generally known as input/output, or I/O, capability.

Monolithic ("brick") PLCs have a fixed amount of I/O capability built into the unit, while modular ("rack") PLCs use individual circuit board "cards" to provide customized I/O capability. The advantages of using replaceable I/O cards instead of a monolithic PLC, is the fact that individual I/O cards may be easily replaced in the event of failure without having to replace the entire PLC.

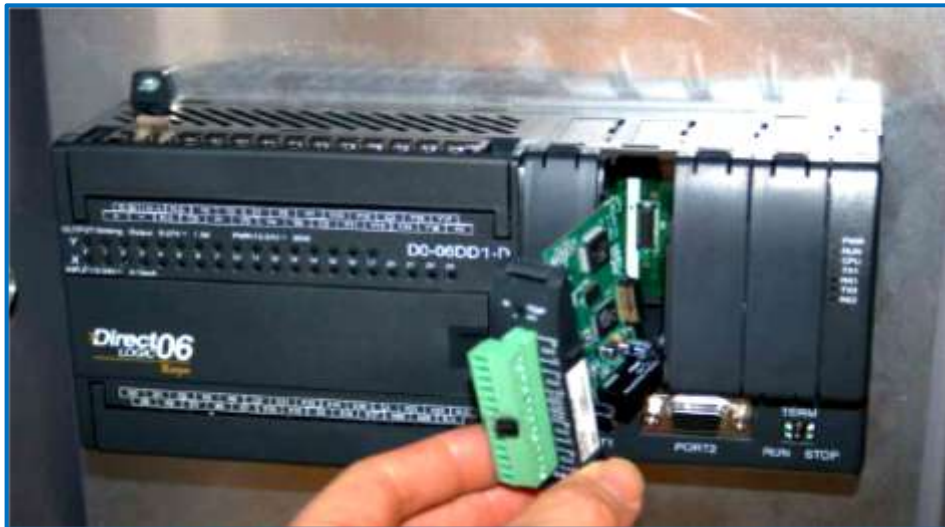


Some PLCs have the ability to connect to processor-less remote racks filled with additional I/O cards or modules, thus providing a way to increase the number of I/O channels beyond the capacity

of the base unit. The connection from host PLC to remote I/O racks usually takes the form of a special digital network, which may span a great physical distance. An alternative scheme for system expansion is to network multiple PLCs together, where each PLC has its own dedicated rack and processor.

Specific I/O cards may be chosen for custom applications, biasing toward discrete cards for applications using many on/off inputs and outputs, or biasing toward analog cards for applications using many 4-20 mA and similar signals. Some PLCs even offer the feature of hot-swappable cards, meaning each card may be removed and a new one inserted without de-energizing power to the PLC processor and rack.

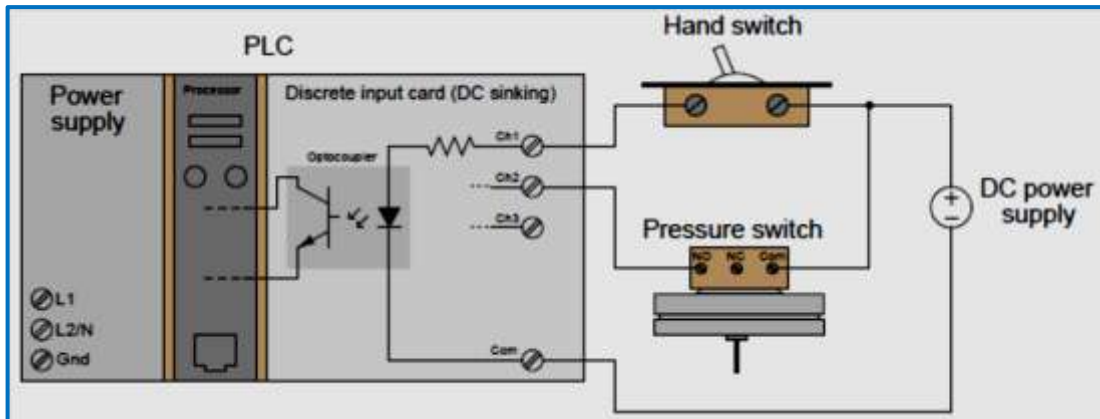
Through the use of communication instructions, one PLC may be programmed to read data from and/or write data to another PLC, effectively using the other PLC as an extension of its own I/O. Although this method is more expensive than remote I/O (where the remote racks lack their own dedicated processors), it provides the capability of stand-alone control in the event the network connection between PLC processors becomes severed. Input/output capability for programmable logic controllers comes in three basic varieties: discrete, analog, and network. The photograph below shows a PLC base unit with 20 discrete input channels and 16 discrete output channels, accepting an analog input card.



Discrete I/O: A “discrete” data point is one with only two states on and off. In order for a PLC to be aware of a discrete sensor’s state, it must receive a signal from the sensor through a discrete “*input*” channel. Inside each discrete “*input*” module is (typically) a set of light-emitting diodes (LEDs) which will be energized when the corresponding sensing device turns on.

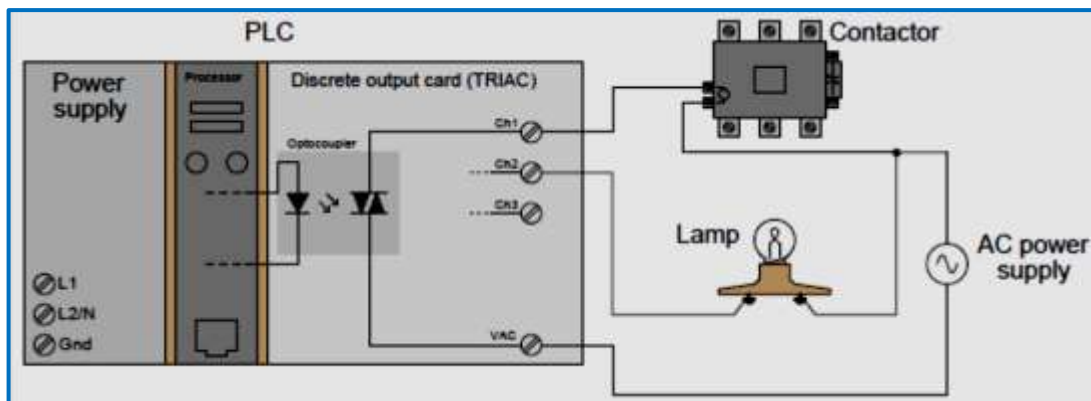
Light from each LED shines on a photo-sensitive device such as a phototransistor inside the module, which in turn activates a bit (a single element of digital data) inside the PLC’s memory. This opto-coupled arrangement makes each input channel of a PLC rather rugged, capable of isolating the sensitive computer circuitry of the PLC from transient voltage “spikes” and other electrical phenomena capable of causing damage.

The internal schematic diagram for a discrete input module (“card”) showed below reveals the componentry typical for a single input channel on that card. Energizing an input channel lights the LED inside the opto-coupler turning on the phototransistor, sending a “high” signal to the PLC’s micro-processor, setting (1) that bit in the PLC’s input register. Each input channel has its own opto-coupler, writing to its own unique memory register bit inside the PLC’s memory. Discrete “input” cards for PLCs typically have 4, 8, 16, or 32 channels.



Indicator lamps, solenoid valves, and motor starters (assemblies consisting of contactors and over-load protection devices) are all examples of discrete control devices. In a manner, similar to discrete inputs, a PLC connects to any number of different discrete final control devices through a discrete “output” channel. Discrete “output” modules typically use the same form of opto-isolation to allow the PLC’s computer circuitry to send electrical power to loads: the internal PLC circuitry driving an LED which then activates some form of photosensitive switching device.

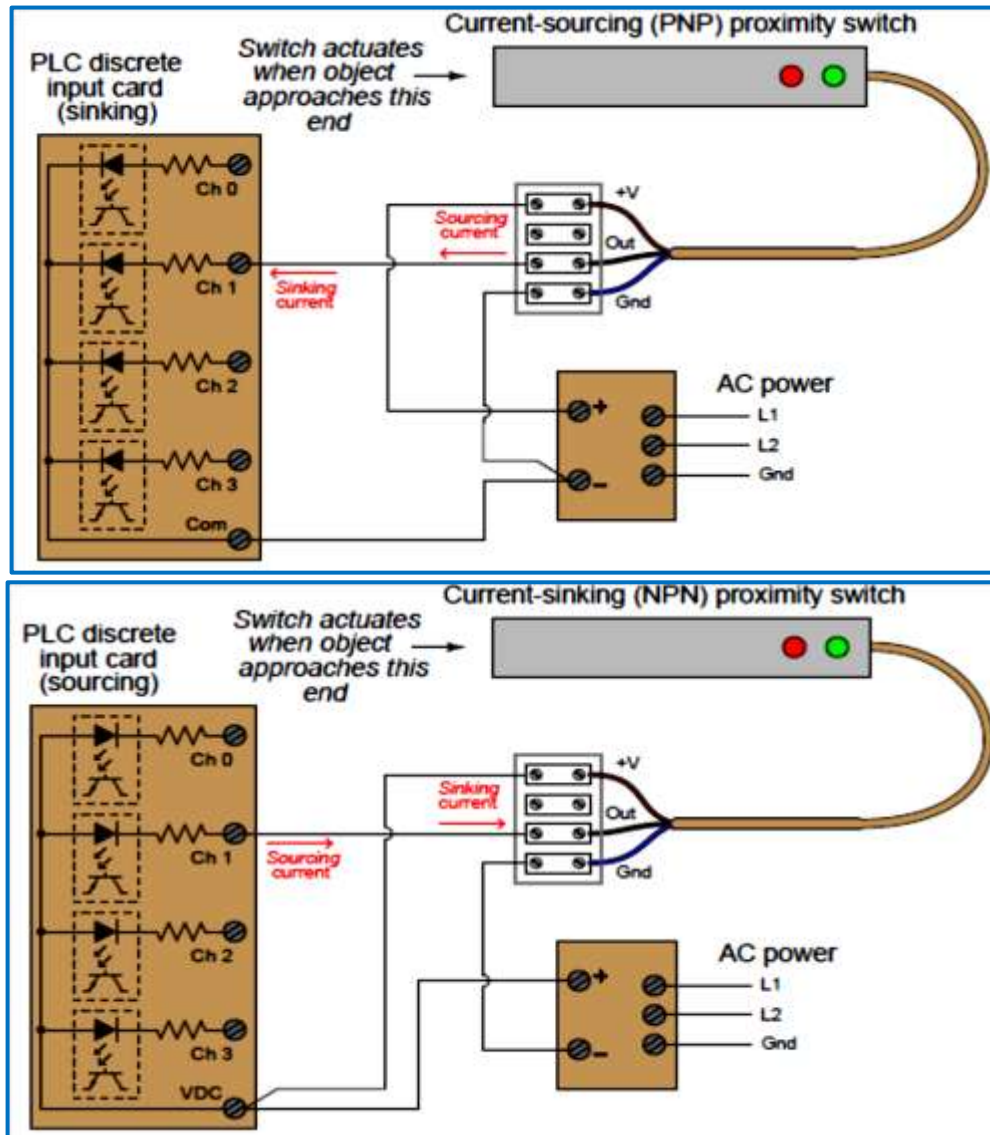
Each output channel has its own opto-coupler, driven by its own unique memory register bit inside the PLC’s memory. Discrete “output” cards for PLCs also typically have 4, 8, 16, or 32 channels. An important concept to master when working with DC discrete I/O is the distinction between current-sourcing and current-sinking devices.



The terms “sourcing” and “sinking” refer to the direction of current (as denoted by conventional flow notation) into or out of a device’s control wire. The schematic diagram shown below is a discrete output module typical for a single channel on that card. Setting a bit (1) in the PLC’s output register

sends a "high" signal to the LED inside the opto-coupler, turning on the photo-TRIAC, sending AC power to the output channel to energize the load.

Some discrete sensing devices are polarity-sensitive, such as electronic proximity sensors containing transistor outputs. A "sourcing" proximity switch can only interface with a "sinking" PLC input channel, and visa-versa. In all cases, the "sourcing" device sends current out of its signal terminal while the "sinking" device takes current into its signal terminal. On the "sinking" card, the input channel terminal is positive while the common ("Com") terminal is negative. On the "sourcing" card, the input channel terminal is negative while the common ("VDC") terminal is positive.



Two photographs of a **DC input** module (sinking) for an Allen-Bradley model SLC 500 are shown below: one with the plastic cover closed over the connection terminals, and the other with the plastic cover opened up for viewing the terminals. A legend on the inside of the cover shows the purpose of each screw terminal: eight input channels (numbered 0 through 7) and two redundant "DC Com" terminals for the negative pole of the DC power supply to connect.

The third photograph shows a discrete **AC output** module for an Allen-Bradley SLC 500 PLC, using TRIACs, as power switching devices rather than transistors, of a DC (sinking) discrete input module. This particular eight-channel module provides two sets of TRIACs for switching power to AC loads, each set of four TRIACs receiving AC power from a “hot” terminal (VAC 1 or VAC 2), the other side of the load device being connected to the “neutral” (grounded) conductor of the AC power source.



Analog I/O: All PLCs are digital devices, however, in order to interface with an analog sensor or control device, some “translation” is necessary between the analog and digital worlds. Inside every analog input module is an ADC, or Analog-to-Digital Converter, circuit designed to convert an analog electrical signal into a multi-bit binary word. Conversely, every analog output module contains a DAC, or Digital-to-Analog Converter, circuit to convert the PLC’s digital command words into analog electrical quantities. Analog I/O is commonly available for modular PLCs for many different analog signal types, including:

- Voltage (0 to 10 volt, 0 to 5 volt)
- Current (0 to 20 mA, 4 to 20 mA)
- Thermocouple (millivoltage)
- RTD (millivoltage)
- Strain gauge (millivoltage)

Network I/O: Many different digital network standards exist for PLCs to communicate with, from PLC to PLC and between PLCs and field devices. One of the earliest digital protocols developed for PLC communication was the Modbus, originally for the Modicon brand of PLC. Modbus was adopted by other PLC manufacturers as a standard, and actually remains as the most universal digital protocol available for industrial digital devices today.

Another digital network standard developed by a particular manufacturer and later adopted as a standard is the Profibus, originally developed by Siemens. **Modbus and Profibus** networks are considered “de facto” standards because those networks were designed, built, and marketed by pioneering firms prior to their acceptance as standards by others. By contrast, another standard is the **FOUNDATION Fieldbus**, where a committee arrives at a consensus for a network design and specifications prior to that network being built and marketed by some other firm.

Logic Control Programming: There is an international standard for controller programming that is the IEC 61131-3 standard. In the early 1990s, the International Electrotechnical Commission (IEC) began developing the IEC 61131 standard for PLCs encompassing everything, from hardware and test protocol to communications. The standard defines five programming languages: function block, instruction list, ladder diagram, sequential flowchart, and structured text.

However there are programs for over a half-dozen different manufacturers of PLCs (such as, Allen-Bradley, Siemens, Square D, Koyo, Fanuc, Moore Products, and Modicon), with multiple PLC models within most of these brands. After learning how to program one model of PLC, it is quite easy to adapt to other models of PLC programming. The IEC 61131-3 standard specifies five distinct forms of programming language for industrial controllers:

- Ladder Diagram (LD)
- Structured Text (ST)
- Instruction List (IL)
- Function Block Diagram (FBD)
- Sequential Function Chart (SFC)

PLCopen: Is a not-for-profit, vendor and product-independent worldwide trade organization founded in 1992 to extend the IEC 61131-3 standard, provide certifications and benchmarking tools, and define training requirements, guidelines, and other functions. The core activities of PLCopen are focused around IEC 61131-3, the only standard for industrial control programming. It harmonizes the way people design and operate industrial controls by standardizing the programming interface, including specification, design, implementation, testing, installation, and maintenance.

PLCopen, working with member vendors and users, has defined motion function blocks that provide standards to reduce cost, simplify training, and lower engineering costs. This standardization makes programming motion control applications less hardware-dependent and enables application logic to be reused for greater efficiency. There are currently six parts to the motion standard defining standard function blocks for use in a wide range of applications, including machine control, process automation, and off-road equipment controls. The motion control standard includes:

- Part 1 - PLCopen Function Blocks for Motion Control;
- Part 2 - PLCopen Motion Control - Extensions (merged with Part 1 in the new release 2.0);
- Part 3 - PLCopen Motion Control - User Guidelines;
- Part 4 - PLCopen Motion Control - Coordinated Motion;
- Part 5 - PLCopen Motion Control - Homing Extensions;
- Part 6 - PLCopen Motion Control - Fluid Power Extensions.

ISA Safety: Another reason for programming limitations is safety: the more flexible and unbounded a programming language is, the more potential there will be to unintentionally create complicated “run-time” errors when programming. The ISA safety standard number 84 classifies industrial programming languages as Fixed Programming Languages (FPL), Limited Variability Languages (LVL), or Full Variability Languages (FVL).

4. LADDER LOGIC PROGRAMS:

Ladder Logic Programming: Are written methods to document the design and construction of relay racks used in manufacturing and process control, where each device is represented by a symbol on the ladder diagram. The ladder logic is used for Programmable Logic Controllers (PLCs) widely used in industrial control applications. The name is based on the observation that the programming resembles ladders, with two vertical rails and a series of horizontal rungs between them.

Ladder Logic Beginning: According to Dick (Richard) Morley, known as the "father" of the Programmable Logic Controller, the ladder language was not the invention of Modicon. The language is very old, originated in Germany to describe relay circuitry. It is a mnemonic form of rule-based language, designed in a Darwinian fashion over a period of many decades, gave birth to the ladder logic, a very modern and very high level PLC language application.

The name ladder comes from the fact that on the right-hand of the drawing is one power rail and the left-hand side is the other power rail; and in between in a horizontal fashion, is the statement or sequential connection of logical elements which we call relays or relay logic.

The initial logic controller named as 084 had only logic in its functionality, and as a result, was marginal. A software engineer, Chuck Schelberg trying to ascertain how to make a generic call to functionalities that far exceeded the relay symbology and representation, came up with the "DX function.

This programming had also block functions that were elements on the ladder logic representation performing many functionalities including arrays, motor drive functions, servo functions, extended to mathematical functions, PID loops, etc. According to this, "DX function" was the first language used for logic programming.

Thus, "DX function" was the first beginning to the actual ladder programming. The ladder logic is a very powerful construct used today in expert systems and other rule-based languages. However, the symbology, allowing normally open and normally closed situations, as well as, parallel and serial representation, was used for many decades before the invention of the programmable controller.



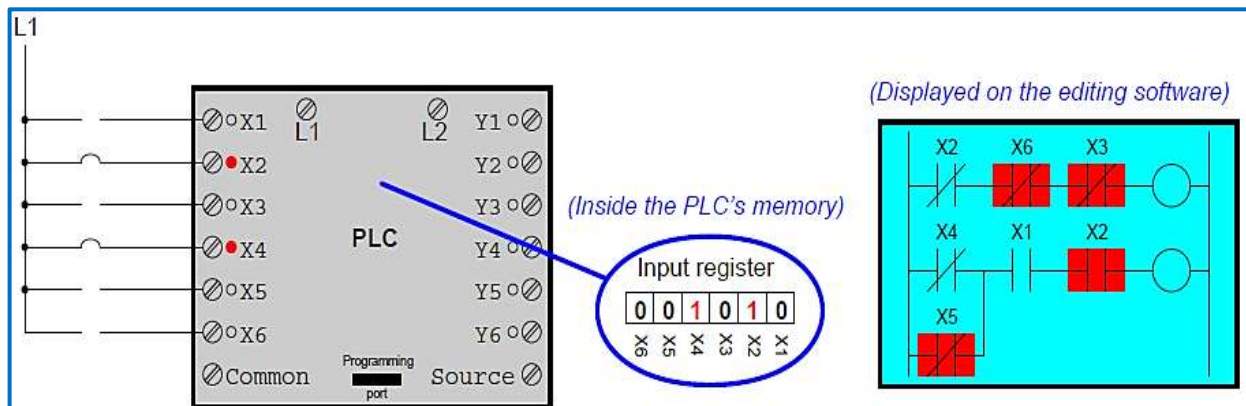
Modicon 084: Dick Morley, Tom Boissevain, George Schwenk, and Jonas Landau.

Although the diagrams themselves have been used since the days when logic could only be implemented using switches and electromechanical relays, the term “*ladder logic*” was only latterly adopted with the advent of the solid state programmable logic. Often the ladder logic program is used in conjunction with an HMI (Human Machine Interface) program operating on a computer workstation.

Ladder Diagrams: The first fundamental rule when examining a Ladder Diagram PLC program, is that each virtual contact shown in the program actuates whenever it reads a “1” state in its respective bit and will be at rest whenever it reads a “0” state in its respective bit (in the PLC’s memory).

If the contact is a normally-open (NO) type, it will open when the contact bit is “0”, and will close when the contact bit is “1”. If the contact is a normally-closed (NC) type, it will close when its bit is “0” and open when its bit is “1”. The “0” bit state causes the contact to be in its “*normal*” (resting) condition, while a “1” bit actuates the contact, forcing it into its “*non-normal*” (actuated) state.

Another rule to remember when examining a Ladder Diagram PLC program is that the programming “*software offers color highlighting*” to display the virtual status of each program element: a “*colored*” contact is closed, while an “*un-colored*” contact is open. Then, as indicated below, the color-highlighted contacts in the programming editor software’s display, shows a collection of contacts addressed to those input bits in various states (colored = closed ; un-colored = open).



Obs: To clarify, the fundamental rules when interpreting contact instructions in Ladder Diagram PLC programs are:

- ✓ Each input bit in the PLC’s memory will be a “1” when its input channel is powered, and will be a “0” when its input channel is unpowered;
- ✓ Each virtual contact shown in the program actuates whenever it reads a “1” state in its respective bit, and will be at rest whenever it reads a “0” state in its respective bit;
- ✓ A colored contact is closed (passes virtual power in the PLC program), while an un-colored contact is open (blocks virtual power in the PLC program).
- ✓ Ladder diagrams are types of electrical notations, using a frequent symbology to illustrate how electromechanical switches and relays are interconnected.
- ✓ The two vertical lines are called “rails”, attached to opposite poles of a power supply, usually 120 volts AC.

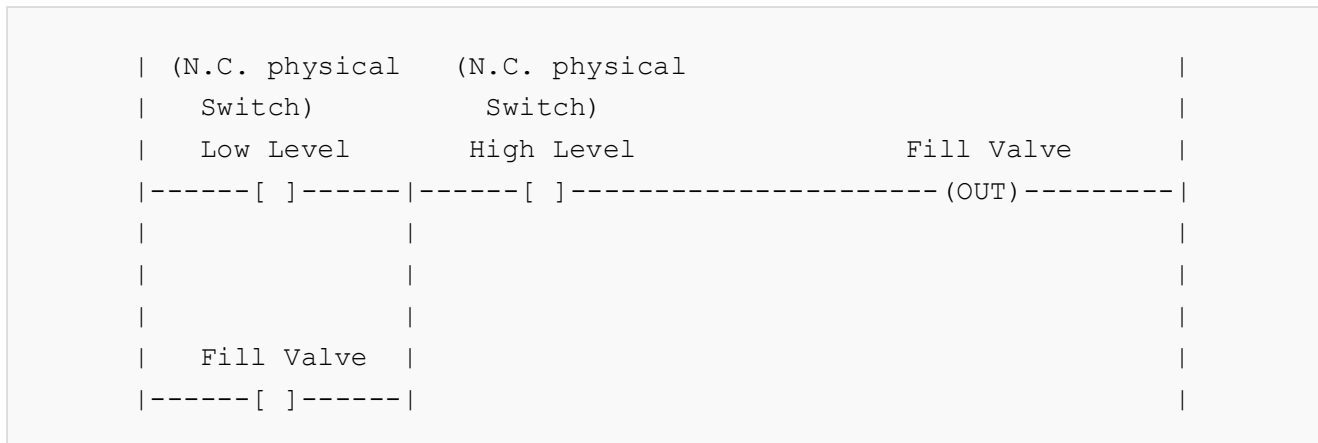
- ✓ Horizontal lines in a ladder diagram are called "rungs," each one representing a unique parallel circuit branch between the poles of the power supply.

Analog Signals: Are like *volume controls*, with a range of values between “zero and full-scale”, that can be typically interpreted as integer values (counts) by the PLC, with various ranges of accuracy depending on the device and the number of bits available to store the data. As PLCs typically use 16-bit signed binary processors, the integer values are limited between -32,768 and +32,767.

Discrete Signals: Are sent using either voltage or current, where a specific range is designated as *On* and another as *Off*. For example, a PLC might use 24 V DC I/O, with values above 22 V DC representing *On*, and values below 2 V DC representing *Off*, and intermediate values undefined. Initially, the PLCs had only discrete I/O. Push buttons, Limit switches, and photoelectric sensors are examples of devices providing a discrete signal.

Pressure, temperature, flow, and weight are often represented by analog signals. Analog signals can use voltage or current with a magnitude proportional to the value of the process signal. For example, an analog 0 - 10 V input or 4 - 20 mA would be converted into an integer value of 0 - 32,767.

Example: A facility needs to store water in a tank. The water is drawn from the tank by another system, and the water level in the tank must be managed by controlling the valve that refills the tank. The ladder diagram is the method used, as shown below:



In this example, the physical state of the float switch contacts must be considered when choosing "normally open" or "normally closed" symbols in the ladder diagram. The PLC has two discrete inputs from float switches (Low Level and High Level). Both float switches (normally closed) open their contacts when the water level in the tank is above the physical location of the switch.

When the water level is below both switches, the float switch physical contacts are both closed, and a true (logic 1) value is passed to the “Fill Valve” output. Water begins to fill the tank. The internal "Fill Valve" contact latches the circuit so that even when the "Low Level" contact opens (as the water passes the lower switch), the fill valve remains on. Since the “High Level” is also normally closed, water continues to flow as the water level remains between the two switch levels.

Once the water level rises enough so that the "High Level" switch is off (opened), the PLC will shut the inlet to stop the water from overflowing (latching) logic. The output is sealed in until a high level condition breaks the circuit. After that the fill valve remains off until the level drops so low that the Low Level switch is activated, and the process repeats again.

In ladder diagram, the contact symbols represent the state of bits in processor memory, which corresponds to the state of physical inputs to the system. If a discrete input is energized, the memory bit is a 1, and a "normally open" contact controlled by that bit will pass a logic "true" signal on to the next element of the ladder. Internal status bits, corresponding to the state of discrete outputs, are also available to the program.

A complete program may contain thousands of rungs, evaluated in sequence. Typically the PLC processor will alternately scan all its inputs and update outputs, then evaluate the ladder logic. Input changes during a program scan, will not be effective, until the next I/O update. A complete program scan may take only a few milliseconds, much faster than changes in the controlled process.

Thus, discrete input and output channels on a PLC correspond to individual bits in the PLC's memory array. Similarly, analog input and output channels on a PLC correspond to multi-bit words (contiguous blocks of bits) in the PLC's memory. The association between I/O points and memory locations is by no means standardized between different PLC manufacturers, or even between different PLC models designed by the same manufacturer.

Ladder Logic Programming: The language itself can be seen as a set of connections between logical checkers (contacts) and actuators (coils). If a path can be traced between the left side of the rung and the output, through asserted (true or "closed") contacts, the rung is true and the output coil storage bit is asserted (1) or true. If no path can be traced, then the output is false (0) and the "coil" by analogy to electromechanical relays is considered "de-energized".

Ladder logic has contacts that make or break circuits to control coils. Each coil or contact corresponds to the status of a single bit in the programmable controller's memory. Unlike electromechanical relays, a ladder program can refer any number of times to the status of a single bit, equivalent to a relay with an indefinitely large number of contacts. PLCs have many types of special blocks. They include timers, arithmetic operators and comparisons, table lookups, text processing, PID control, and filtering functions.

Powerful PLCs can operate on a group of internal memory locations and execute an operation on a range of addresses, for example, to simulate a physical sequential drum controller or a finite state machine. In some cases, users can define their own special blocks, which effectively are subroutines or macros. The large library of special blocks along with high speed execution has allowed use of PLCs to implement very complex automation systems.

Overview of Bit Logic Instructions: Contacts and coils work with two digits, *1 and 0*, which form the base of a number system called as binary system, binary digits or bits. In the world of contacts and coils, the *1* indicates activated or energized, and *0* indicates not activated or not energized. The bit logic instructions interpret signal states of *1 and 0* and combine them according to the Boolean

logic. These combinations produce a result of 1 or 0 that is called the “result of logic operation” (RLO). The common basic bit logic instructions are:

- ---| |--- = Normally Open Contact = True 1
- ---| / |--- = Normally Closed Contact = False 0
- ---(SAVE) = Save RLO into BR Memory
- XOR = Bit Exclusive OR
- ---() = Output or Coil
- ---(#)--- = Midline Output
- ---|NOT|--- = Invert Power Flow

The following instructions react to an RLO (result of logic operation) of 1:

- ---(S) = Set Coil
- ---(R) = Reset Coil
- SR = Set-Reset Flip Flop
- RS = Reset-Set Flip Flop

Other instructions react to a positive or negative edge transition to perform the following functions:

- ---(N)--- = Negative RLO Edge Detection
- ---(P)--- = Positive RLO Edge Detection or Positive Transition Sensing Coil
- NEG = Negative Edge Detection or Negative Transition Sensing Coil
- POS = Positive Edge Detection or Positive Transition Sensing Coil
- ---(M)--- = Retentive memory coil.
- ---(SM)--- = Set retentive memory coil
- ---(RM)--- = Reset retentive memory coil

Boolean Functions:

- AND (value, value, destination) - Binary and function
- OR (value, value, destination) - Binary or function
- XOR (value, value, destination) - Binary exclusive or function
- NOT (value, destination) - Binary not function

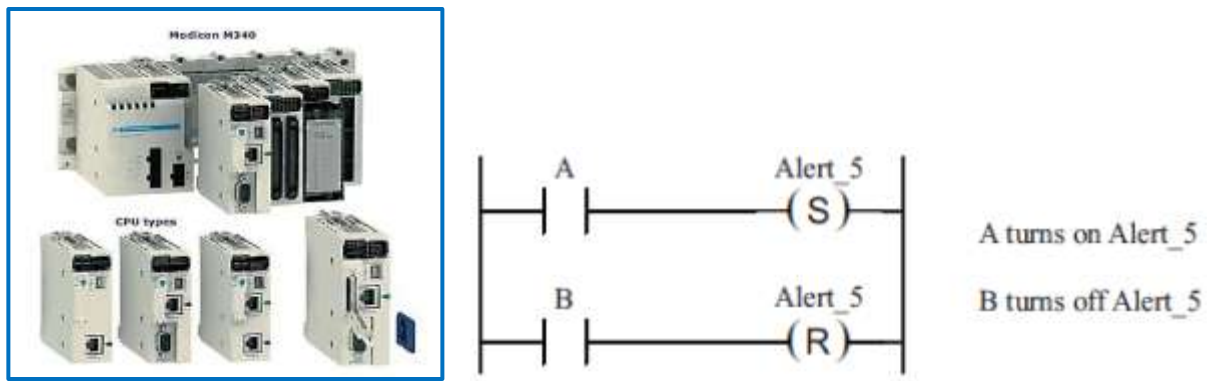
The uses of the Boolean functions require two arguments. The AND function will only turn on bits in the result that are true in both of the source words. The OR function will turn on a bit in the result word if either of the source word bits is on. The XOR function will only turn on a bit in the result word if the bit is on in only one of the source words. The NOT function reverses all of the bits in the source word.

The industry trend is toward using the IEC 61131-3 symbols (formerly IEC 1131-3) standard, but, since IEC 61131-3 is only a voluntary standard, individual manufacturers have some freedom in implementation. These symbols are called as *ladder logic instructions* that are scanned (executed) by

the PLC. The ladder logic is a very visual and graphical language, very different from textual languages like C++, Fortran, Basic, and Java. The main ladder languages are:

- ✓ Modicon (IEC compliant)
- ✓ Allen-Bradley ControlLogix PLC-5/SLC-500/ SLC-5000 (IEC compliant and not compliant)
- ✓ Siemens S7 (IEC compliant)
- ✓ GE Fanuc (IEC compliant)

Modicon: The Modicon Schneider M340 is programmed in ladder logic compatible with IEC 61131-3 and can be used individually, but is also a perfect companion of Modicon Premium and Modicon Quantum, increasing the performance, quality and the profitability of industrial processes. The Modicon *basic ladder logic coil symbols* are similar to described above, as basically indicated below:



Allen-Bradley ControlLogix: The RsLogix family from Rockwell Automation has been developed to operate on Microsoft Windows operating systems. The first PLC introduced is known as PLC 5 and the next is the SLC 500 family. The RsLogix 500 was the first PLC programming software to offer unbeatable productivity with an industry-leading user interface. The RSLogix 5000 Enterprise Series software is an IEC 61131-3 compliant software package that offers relay ladder, structured text, function block diagram, and sequential function chart editors to develop application programs.







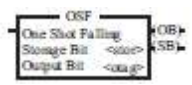
PLC 5



SLC 500 Family

For the Allen-Bradley PLCs, the basic output ladder logic coil symbols are:

-  = Output or coil


-  = Latch coil
-  = Unlatch coil
-  = One shot rising output
-  = One shot falling output

Siemens Family (S7-200, S7-300, S7-400, S7 1200 and S7 1500 series): The three types of S7 processors (S7-200, S7-300/400, and S7-1200) have the same ladder basic instructions. The most Siemens PLCs provide a Profinet or Ethernet interface (S7-1200 and 1500 family). Older versions (S7-300 family) can be connected via an MPI/Profibus interface with an additional MPI-Adapter. The only exception is the midline output coil that is not valid for the S7-200 and S7-1200 processors and the negated and transitional coils are valid only for the S7-1200, as shown below:

- ---() = Output or Coil
- ---(/) = Negated coil (S7-1200 only).
- ---(#) = Midline output coil (S7-300/400 only)
- ---(P) = Positive transition sensing coil (S7-1200 only).
- ---(N) = Negative transition sensing coil (S7-1200 only).

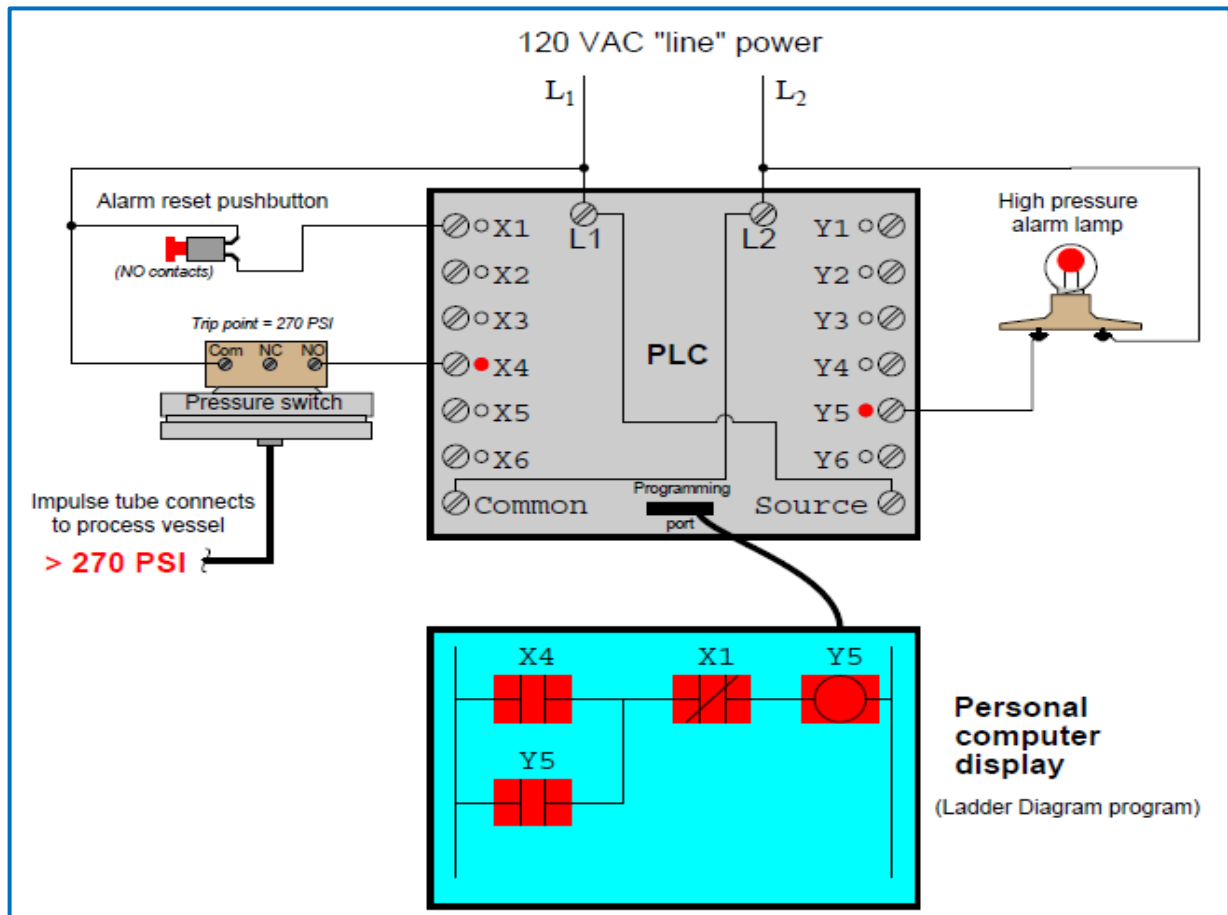
GE PLC's: GE Fanuc and Fanuc GE CNC Europe were joint ventures of FANUC (Japan) and General Electric (USA). In August 2009 GE and FANUC agreed to dissolve the joint venture, completed in December 2009. After this, the Software, Embedded and Control Systems businesses became a part of GE and the CNC division went to FANUC.

The **Series 90-30** family of controllers, came in 1990, with more than 200,000 applications, such as high speed packaging, material handling, complex motion control, water treatment, continuous emissions monitoring, mining, food processing, elevator control, injection molding, etc. The **Series 90-70** family of controllers, came in 1991, designed to meet the most complex applications that require a large number of I/O and process memory. The **VersaMax PLC**, came in 1998, with a broad selection of I/O modules and its communication modules link to a variety of networks. The basic ladder logic coil (output) symbols are:

- ---() = Output or coil.
- ---(/) = Negated coil.
- ---(S) = Set coil.
- ---(R) = Reset coil.
- ---() = Positive transition sensing coil (POSCOIL).

- ---(P) = Positive transition sensing coil (PTCOIL).
- ---(ψ) = Negative transition sensing coil (NEGCOIL).
- ---(N) = Negative transition sensing coil (NTCOIL).

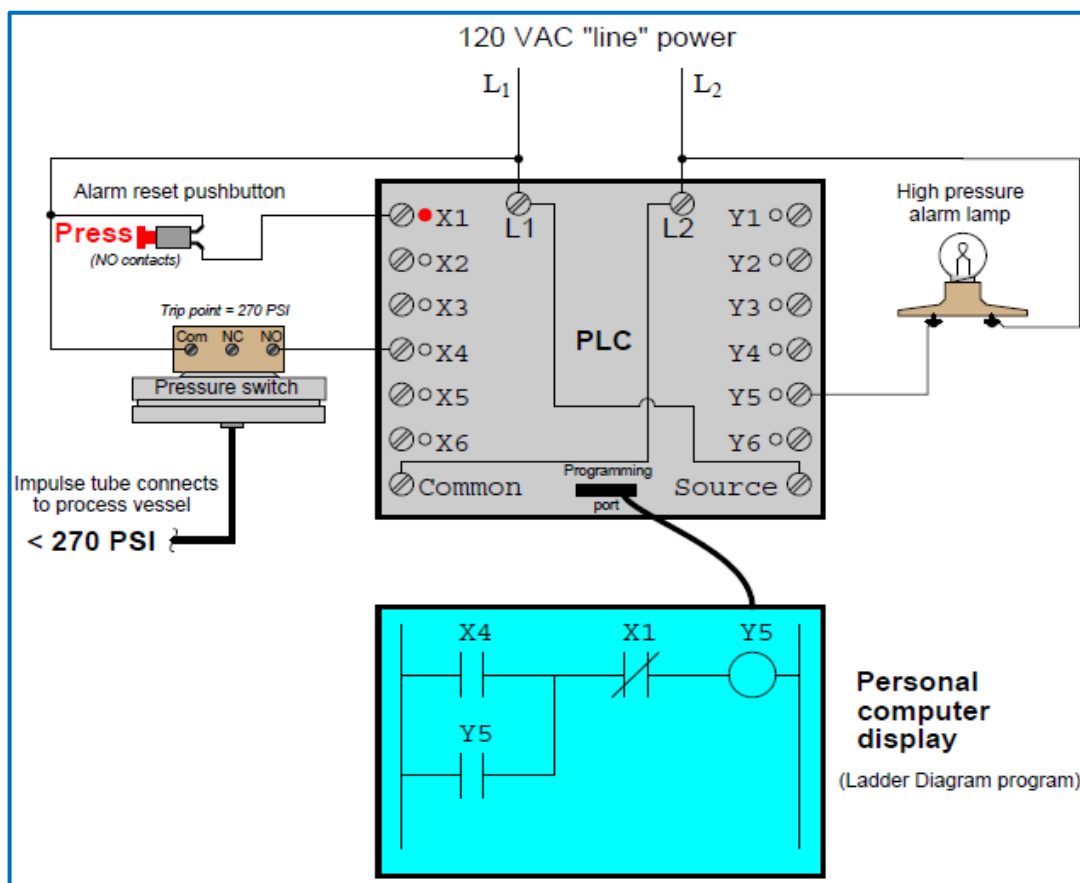
Example.1: As indicated schematically below, the PLC's task is to energize a warning lamp when the process vessel pressure always exceeds 270 PSI, and keep that warning lamp energized even if the pressure falls below the trip point of 270 PSI. This way, all operators will be alerted to both past and present process vessel over-pressure events. A 120 volt AC "line" power (L1 and L2) provides electrical energy for the PLC to operate.



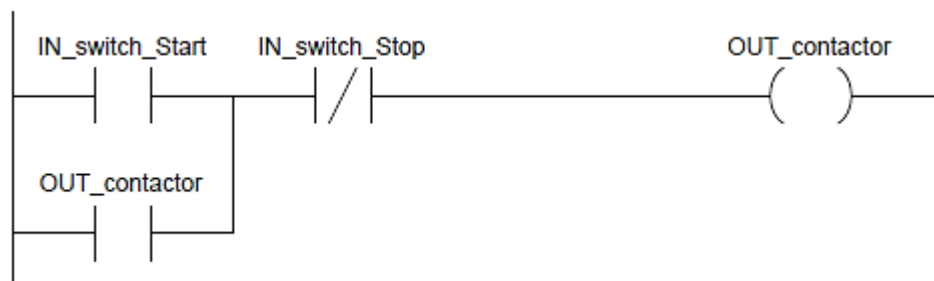
When the process vessel experiences a high pressure (> 270 PSI), the pressure switch actuates, closing its normally-open contact and energizes the input X4 on the PLC, which will "close" the virtual contact X4 in the ladder program sending a virtual power to the "coil" Y5, which in turn latches itself on through the virtual contact Y5, also energizes the discrete output Y5 thus, energizing the warning lamp.

When the process pressure falls below 270 PSI, the pressure switch will return to its normal state (open), thus de-energizing the discrete input X4 on the PLC. Because of the latching contact Y5 in the PLC's program, the output Y5 remains on, to keep the warning lamp in its energized state. The

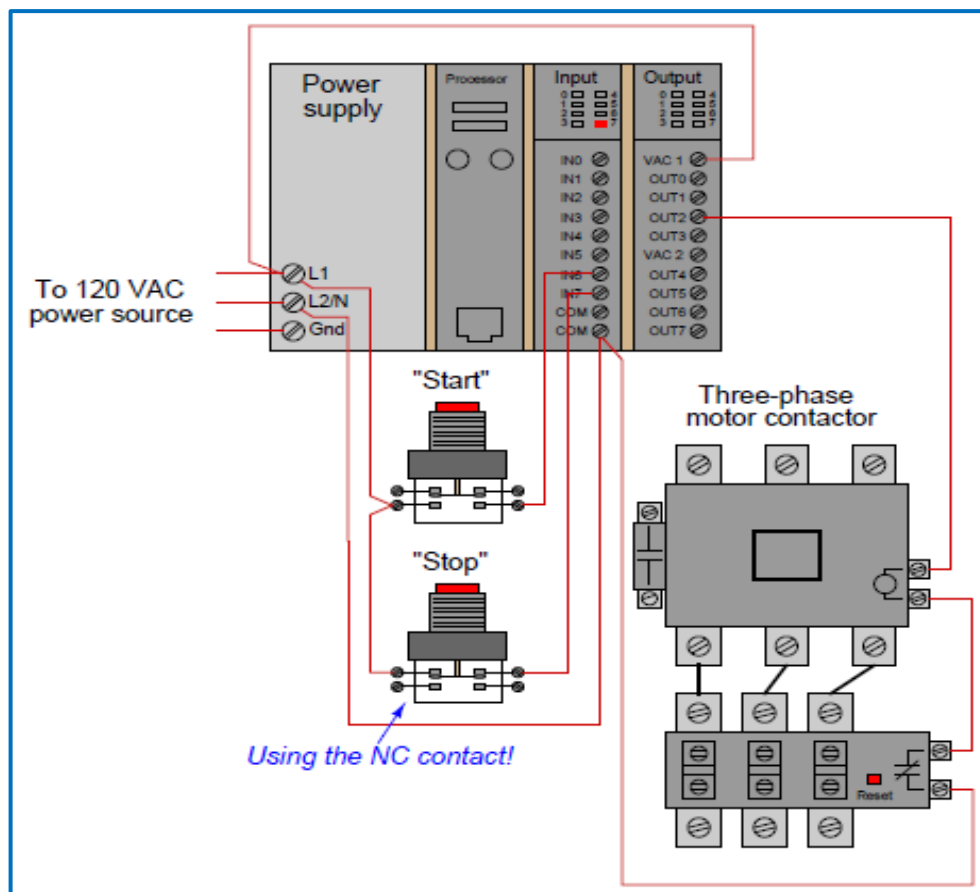
only way for a human operator to reset the warning lamp is to press the pushbutton, then, energizing the input X1 on the PLC, opening the virtual contact X1 (normally closed) in the program, and interrupting the power to the virtual coil Y5, powering down the warning lamp and un-latching the virtual power in the program, as can be seen below:



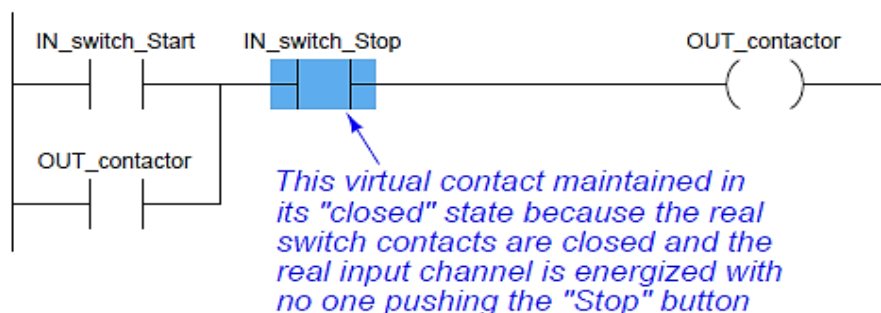
Example.2: A very common application of a PLC system is a latching start/stop program for controlling electric motors, by means of momentary-contact push button switches. In this system, two pushbutton switches are connected to discrete inputs on a PLC, and the PLC in turn energizes the coil of a motor contactor relay by means of one of its discrete outputs. An overload contact is wired directly in series with the contactor coil to provide motor overcurrent protection, even in the event of a PLC failure where the discrete output channel remains energized. The ladder program for this motor control system would look like this:



The functionality can be illustrated by means of a hypothetical example circuit shown below. Pressing the "Start" pushbutton energizes the discrete input channel 6 on the PLC, which "closes" the virtual contact in the PLC program labeled IN switch Start. The normally-closed virtual contact for input channel 7 (the "Stop" pushbutton) is already closed by default when the "Stop" button is not being pressed, and so the virtual coil will receive "power" when the "Start" pushbutton is pressed and the "Stop" pushbutton is not.



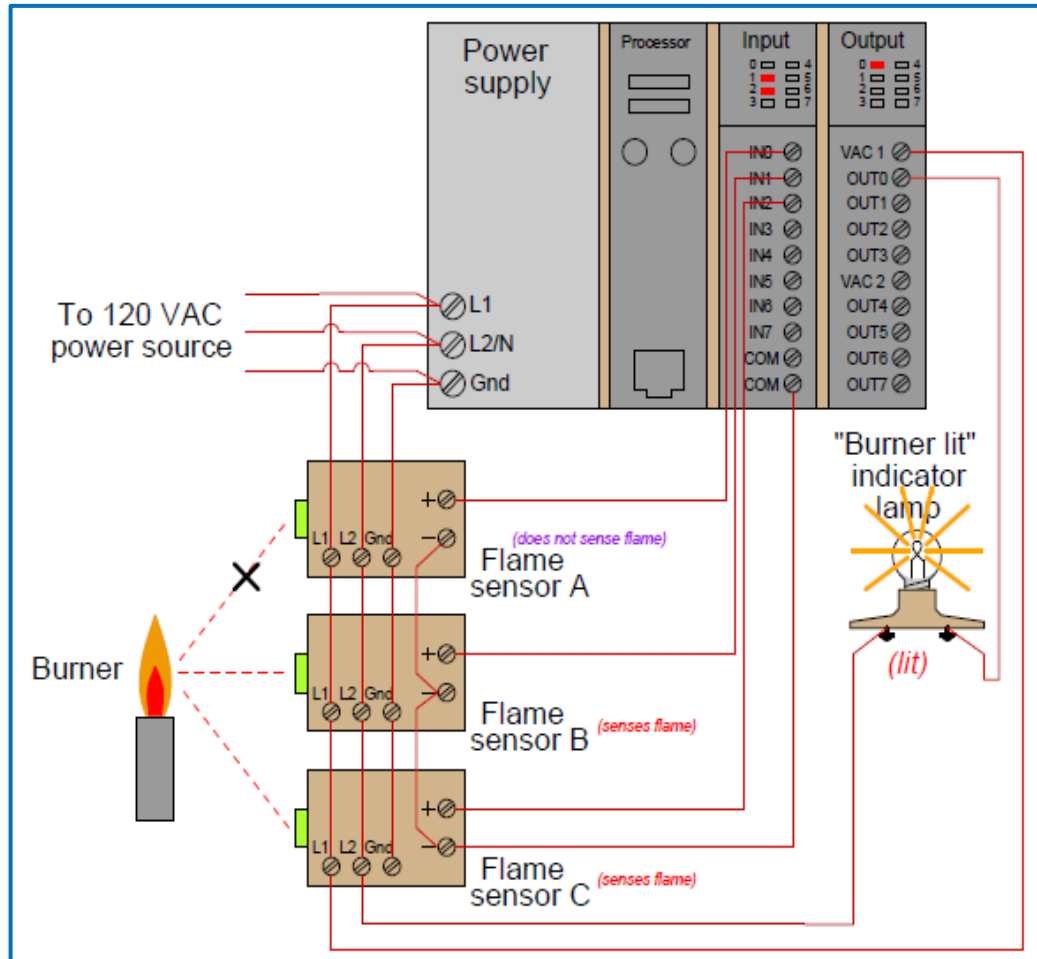
In order for the PLC program to work properly with this new switch wiring, the virtual contact for IN switch Stop must be changed from a normally-closed (NC) to a normally-open (NO).



Note: The Allen-Bradley corporation (Rockwell) uses the terms: *examine if closed (XIC)* and *examine if open (XIO)*, to describe "normally open" and "normally closed" virtual contacts, respectively, in their Ladder Diagram programming.

Example.3: Imagine the construction and programming of a redundant flame-sensing system to monitor the status of a *burner flame* using three sensors. The purpose of this system will be to indicate a “lit” burner if, at least, two of the three sensors indicate flame. If only one sensor indicates flame (or if no sensors indicate flame), the system will declare the burner to be un-lit.

The burner’s status will be visibly indicated by a lamp that human operators can readily see inside the control room area. To illustrate how this program would work, consider that the *flame sensors B and C* detect flame, but *sensor A* does not. This represents a two-out-of-three condition, and the PLC should turn on the “*burner light*” indicator, as programmed:



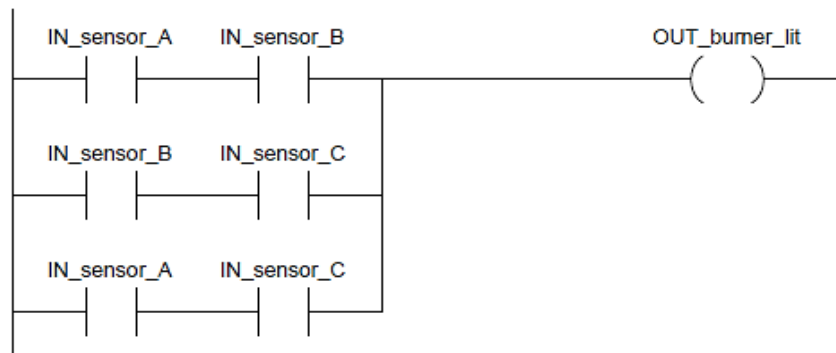
The flame sensor A bit is “clear” as “0” state, because its corresponding input channel is de-energized. The PLC program has “set” that corresponding bit in the PLC’s output memory register to “1” state, noticing the fact that the “burner light” of the indicator lamp is energized. A display of input and output register bits shows the “set” and “reset” states for the PLC at this moment:

Input register								Output register							
0	0	0	0	0	1	1	0	0	0	0	0	0	0	1	
IN7	IN6	IN5	IN4	IN3	IN2	IN1	IN0	OUT7	OUT6	OUT5	OUT4	OUT3	OUT2	OUT1	OUT0

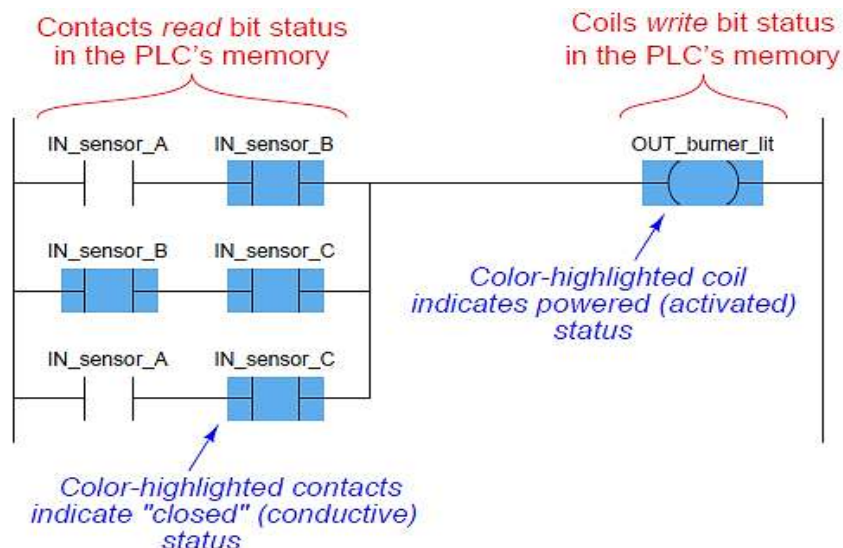
The series-connected contacts in a Ladder Diagram perform the logical *AND function*, while parallel contacts perform the logical *OR function*. Thus, this two-out-of-three flame-sensing program could be verbally described as:

“Burner is lit if either A and B, or either B and C, or A and C”.

Then, the ladder program to determine if at least two out of the three sensors detect flame, with the tag names referencing each contact and coil, is indicated below:

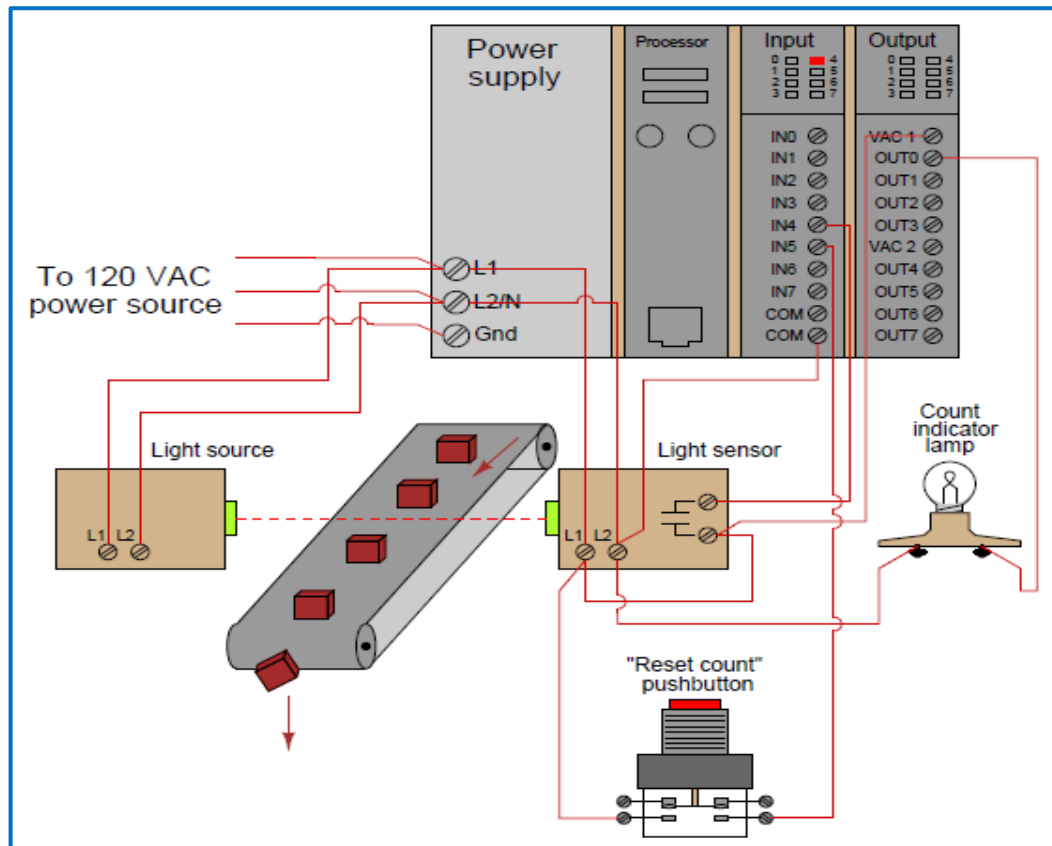


The main purpose of a coil in a PLC program is to write the status of a bit in the PLC's memory. Here, the “energized” coil sets the bit for the PLC output “0” to a “1” state, thus activating the real-world output and sending electrical power to the “burner” lamp. Examining the Ladder Diagram program with status indication enabled, we would see how just one of the series-connected contact pairs are passing “virtual power” to the output coil:

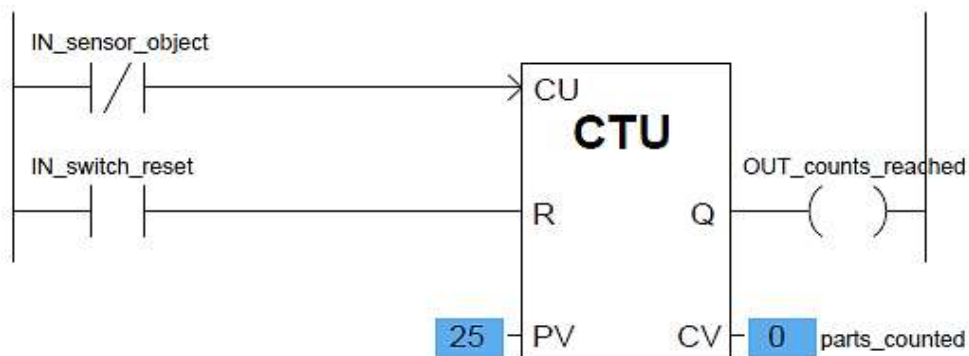


Counters: Are PLC instructions that either increment (count up) or decrement (count down) an integer number value when prompted by the transition of a bit from 0 to 1 (“false” to “true”). Counter instructions come in three basic types: *up counters*, *down counters*, and *up/down counters*. Both “up” and “down” counter instructions have single inputs, whereas “up/down” counters have two inputs: one to make the counter increment and one to make the counter decrement.

To illustrate the use of a counter instruction, we will analyze a PLC-based system designed to count objects, as they pass down a conveyor belt:



In this system, a continuous light beam causes the light sensor to close its output contact. When an object on the conveyor belt interrupts the light beam, the sensor's contact opens, interrupting the power and causing to input IN4. A pushbutton switch is used to activate the discrete input IN5, when pressed, and serve as a manual “reset” of the count value. An indicator lamp connected to one of the discrete output channels serve as an indicator, when the object count value has exceeded the pre-set limit. The Ladder Diagram program is:



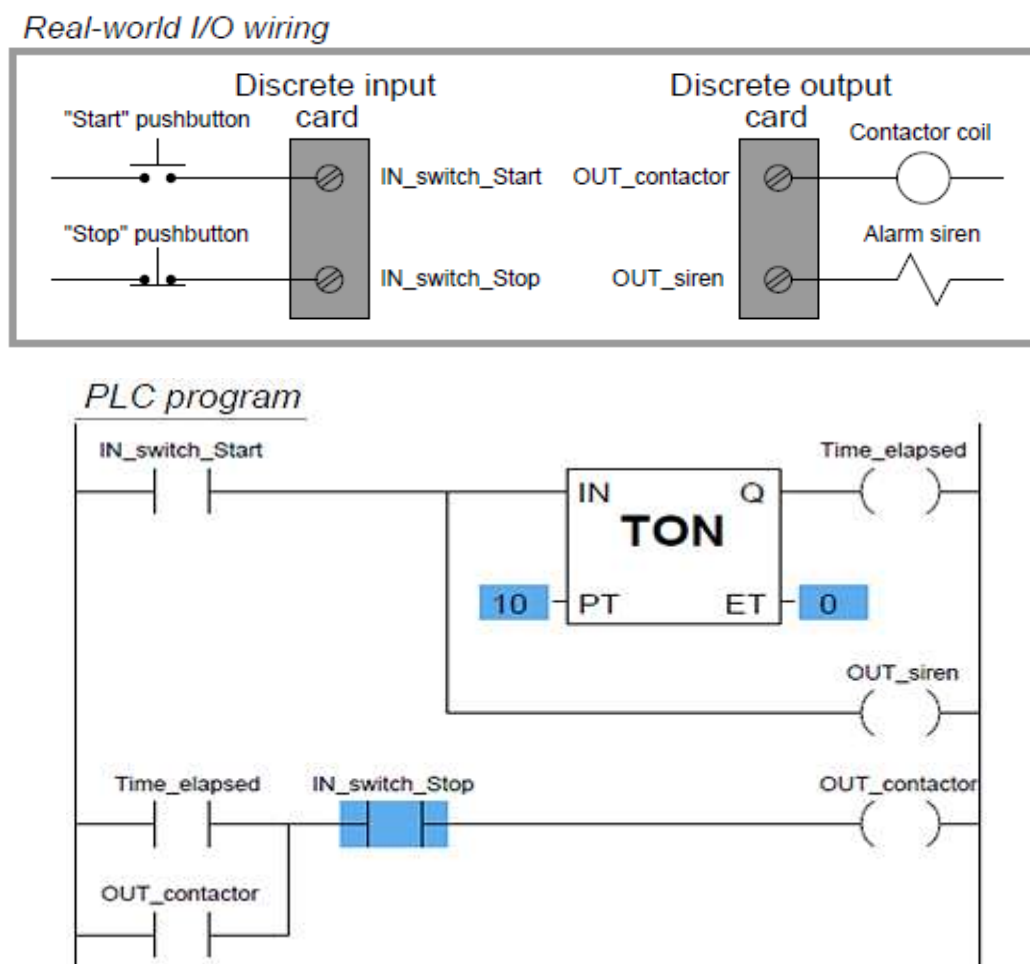
The counter instruction (CTU) is an incrementing counter, which means it counts “up” using its “CU” input. The normally-closed virtual contact (IN sensor object) is held in the “open” state when the light

beam is continuous, with its discrete input channel energized. When the beam is broken by a passing object on the conveyor belt, the input channel de-energizes, causing the virtual contact (IN sensor object) to “close”.

In this case, the counter's preset value (PV) is 25 and the counter's current value (CV) is 0, shown highlighted in blue, and it serves for activating the counter's output (Q), which turns on the count indicator lamp (OUT counts reached coil). When the reset pushbutton is pressed, the counter immediately resets its current value (CV) to zero. The second input of the counter instruction box (“R”) is the reset input, receiving virtual power from the contact (IN switch reset).

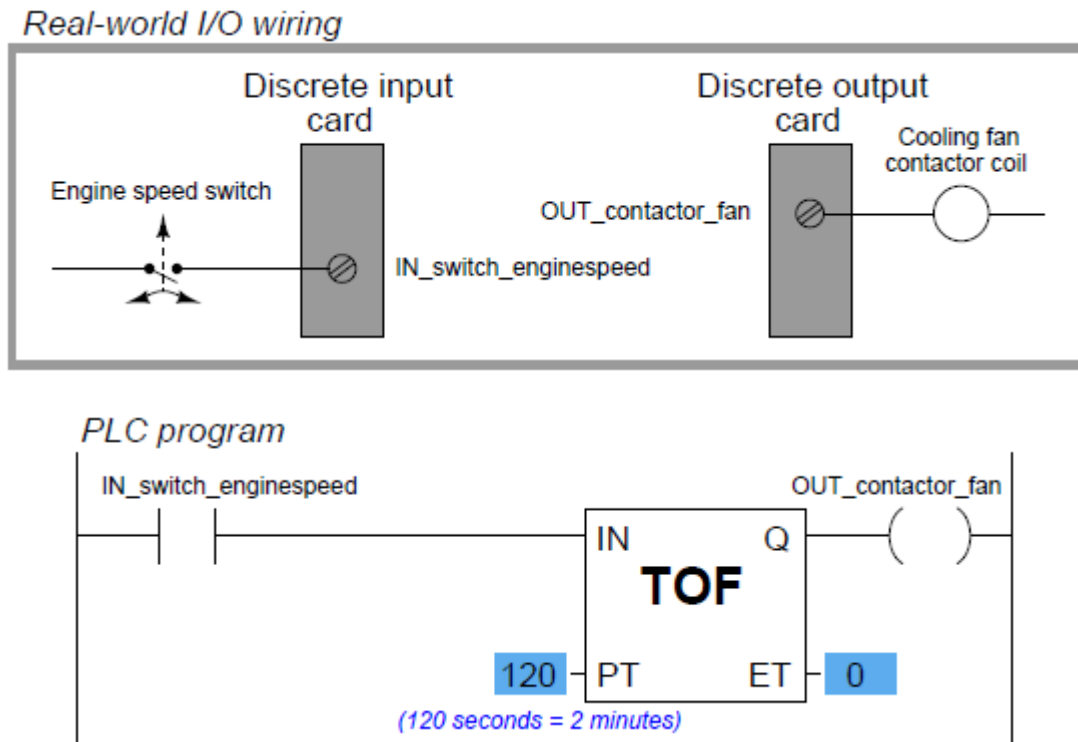
Timers: Are PLC instructions to measure the amount of time elapsed following an event. Timer instructions come in two basic types: *on-delay timers* and *off-delay timers*. Both have single inputs for the timed function. The “on-delay” timer activates an output only when the input has been active for a minimum amount of time.

On-delay Timers (TON): For instance, a PLC program is designed to sound an audio alarm siren prior to starting a conveyor belt and the operator must press and hold the “Start” pushbutton for 10 seconds, warning people to clear away from the conveyor belt that is about to start. Then, only after this 10-second the start delay makes the motor actually start (latch “on”), as shown below:

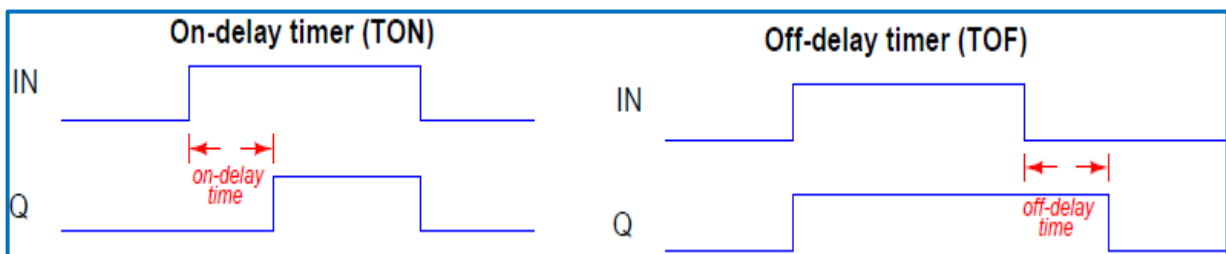


The timer instruction (TON) is an incrementing timer, similar to an “up” counter, the on-delay timer’s elapsed time (ET) value increments once per second until the preset time (PT) is reached, doing the output (Q) activates. The preset time value is 10 seconds, however, the Q output will not activate until the “Start” switch is pressed by the operator. The alarm siren, not activated by the timer, energizes immediately when the “Start” pushbutton is pressed.

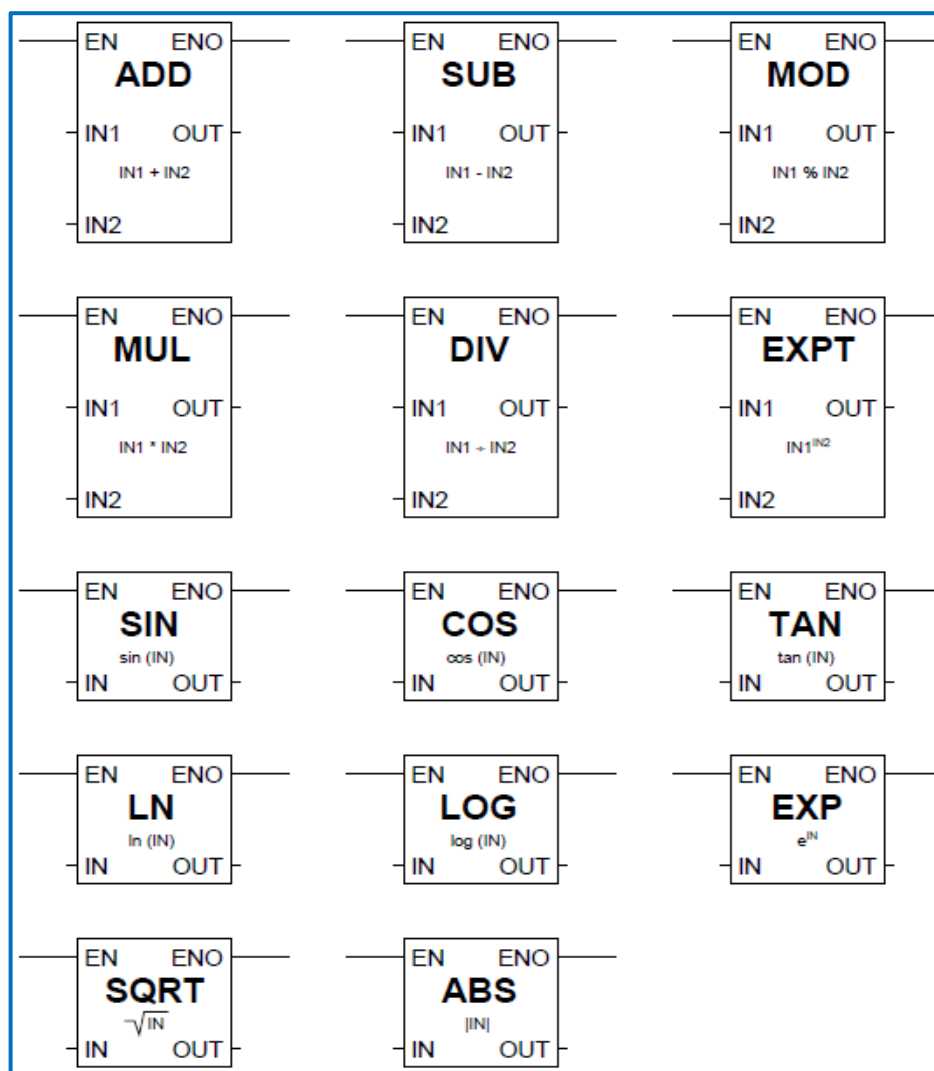
Off-delay Timers (TOF): These timer instructions differ from the on-delay type, as the timing function begins as soon as the instruction is deactivated, not when it is activated. An application for an off-delay timer is a cooling fan motor control for a large industrial engine, where the PLC starts an electric cooling fan, as soon as the engine is detected as rotating, and keeps the fan running for two minutes to dissipate residual heat, as shown below:



When the input (IN) is activated, the output (Q) immediately turns on the cooling fan motor contactor, providing the engine with cooling, as soon as it begins to rotate. When the engine stops rotating, the speed switch returns to its *normally-open position*, de-activating the timer’s input signal. The Q output only remains active while the timer counts from 0 seconds to 120 seconds. When it reaches 120 seconds, the output (Q) de-activates (shutting off the cooling fan motor) and the elapsed time value remains at 120 seconds until the input re-activates, as time resets back to zero.



Math Instructions: The IEC 61131-3 standard specifies several dedicated ladder instructions for performing arithmetic calculations. Each of these math instructions must be enabled by an “energized” signal to the enable (EN) input. Input and output values are linked to each math instruction by tag name. Some of them are shown here:

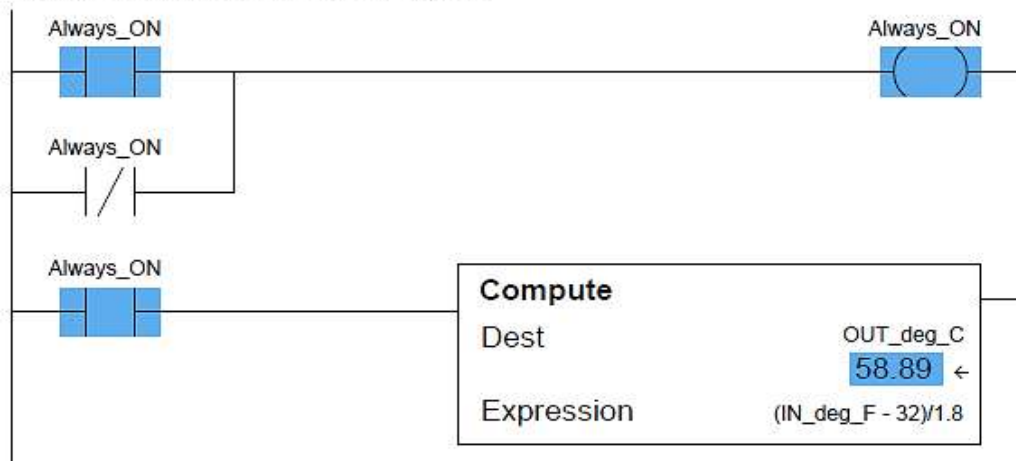


The Allen-Bradley Logix5000 programming, for example, has the “Compute” (CPT) function, which allows any typed expression to be computed in a single instruction, opposed to several dedicated math instructions such as “Add,” “Subtract,” etc.

A general-purpose math instructions dramatically shorten the length of a ladder program when compared to the use of dedicated math instructions for any applications requiring non-trivial calculations. For example, the same Fahrenheit-to-Celsius temperature conversion implemented in Logix5000 programming, only requires a single math instruction and no declarations of intermediate variables:

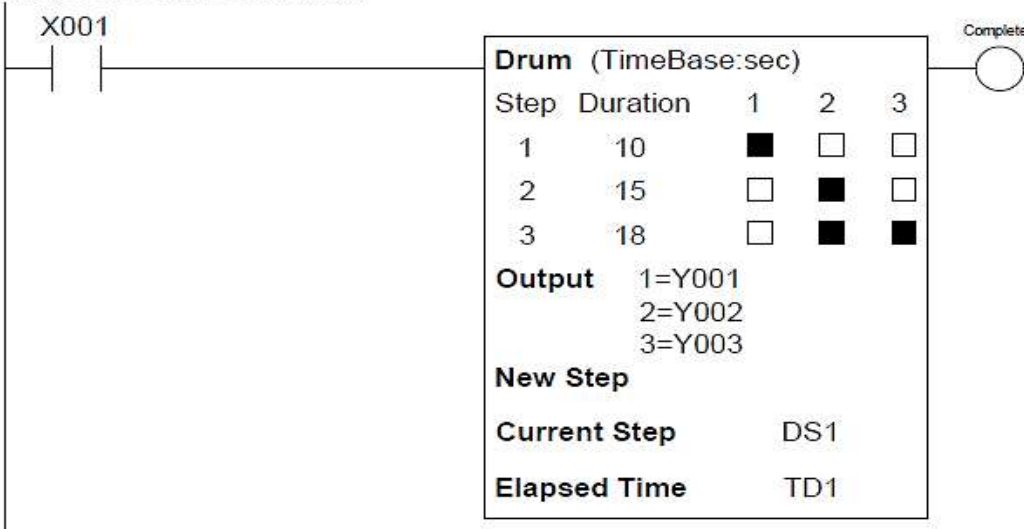
Example: The PLC should convert a temperature measurement of 138°F to units of degrees Celsius. In this particular case, the program inputs the temperature measurement and calculates the equivalent temperature as 58.89 °C, as shown below:

Rockwell Logix5000 PLC program



Sequencers: Many industrial processes require predefined sequences. Traffic lights and batch processes are perhaps the clearest examples, where parameters such as *warning lights*, *alarms*, *temperature* and *pressure* should be controlled and monitored. The drum instruction offered in Koyo PLCs is a model of simplicity itself. This instruction is practically self-explanatory, as shown in the following example:

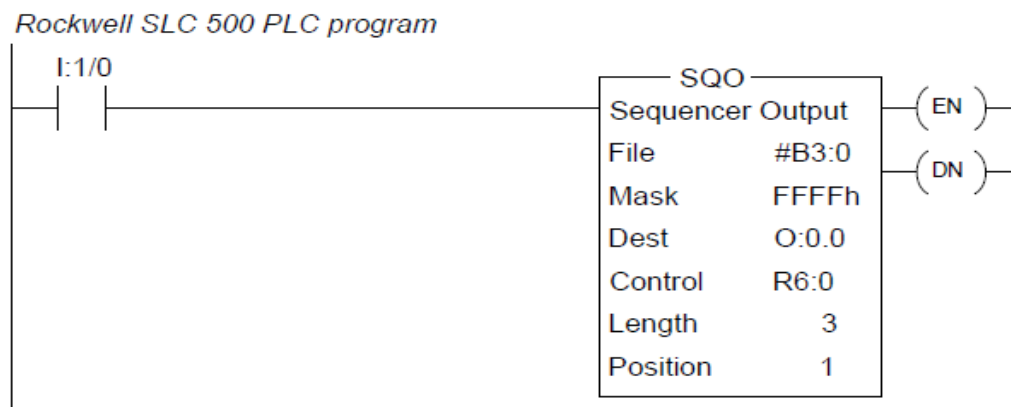
Koyo CLICK PLC program



The first step in this particular example has a duration of 10 seconds, the second step 15 seconds, and the third step 18 seconds. At the first step, only output bit Y001 is set. In the second step, only output bit Y002 is set. In the third step, output bits Y002 and Y003 are set (1), while bit Y001 is reset (0). The colored versus uncolored boxes reveal which output bits are set and reset with each step.

The current step number is held in memory register DS1, while the elapsed time (in seconds) is stored in timer register TD1. A “complete” bit is set at the conclusion of the three-step sequence. Koyo drum instructions may be expanded to include more than three steps and more than three output bits, with each of those step times independently adjustable and each of the output bits arbitrarily assigned to any writable bit addresses in the PLC’s memory.

Allen-Bradley Sequencer: Rockwell (Allen-Bradley) PLCs use a more sophisticated set of instructions to implement the sequences. The closest equivalent to Koyo's drum instruction is the Allen-Bradley SQO (Sequencer Output) instruction, shown below:



Sequencer instructions in Allen-Bradley PLCs use a notation called *indexed addressing* to specify the locations in memory for the set of 16-bit words. In the example shown above, we see the "File" parameter specified as #B3:0. The "#" symbol tells the instruction that this is a starting location in memory for the first 16-bit word, when the instruction's position value is zero.

If B3:0 is the word referenced at position 0, then B3:1 will be the memory address read at position 1, B3:2 will be the memory address read at position 2, and so on. Thus, the "position" value causes the SQO instruction to "point" or "index" to successive memory locations. To illustrate, let us examine a set of bits held in the B3 file of an Allen-Bradley SLC 500 PLC:

Data File B3 (bin) -- BINARY															
B3:0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B3:1	0	1	1	1	1	0	0	0	0	0	0	0	0	0	1
B3:2	1	0	0	0	0	0	1	1	0	0	0	0	0	1	0
B3:3	0	0	0	1	0	0	0	0	0	1	0	1	0	1	0
	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1

If File = #B3:0, then . . .

← Read at position = 1

← Read at position = 2

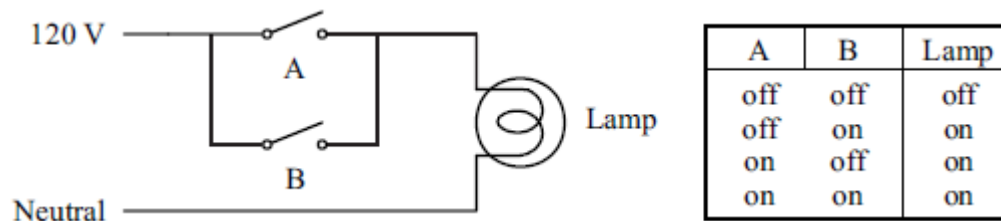
← Read at position = 3

Thus, if the data file is specified in the SQO instruction box as #B3:0, then B3:1 will be the row of bits read when the sequencer's position value is 1, B3:2 will be the row of bits read when the position value is 2, and so on. A mask value of FFFFh (FFFF in hexadecimal format) means all 16 bits of each B3 word will be read and written to the destination.

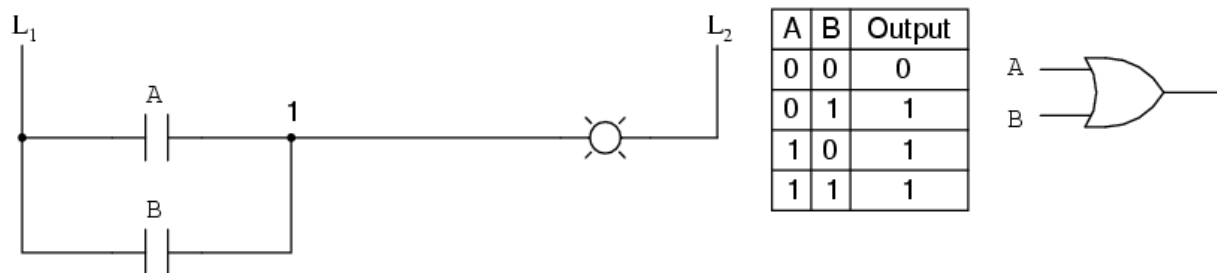
Allen-Bradley PLCs offer a third sequencing instruction called Sequencer Load (SQL), which is the opposite function as the Sequencer Output (SQO). The SQL instruction takes data and writes it into an indexed register according to a position value, rather than reading data from an indexed register and sending it to a destination, as the SQO instruction. SQL instructions are useful for reading data from a live process and storing it in different registers within the PLC's memory at different times.

Relay Ladder Logic Circuit: Typically in industrial relay logic circuits, but not always, the operating voltage for the switch contacts and relay coils are 120 volts AC. Lower voltage AC and even DC systems are sometimes built and documented according to "ladder" diagrams.

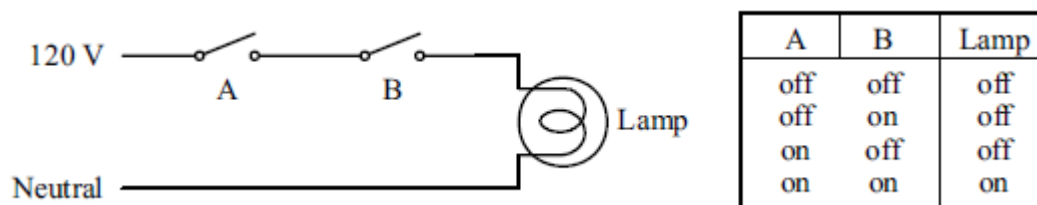
Example.1 - OR Circuit: Two switches labeled A and B are wired in parallel controlling a lamp. The lamp is **on** when the switch A is **on** (closed) or switch B is **on** (closed), as shown below:



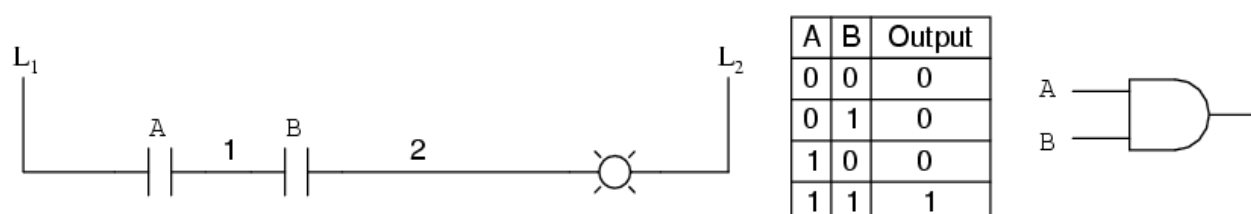
In ladder logic circuit, the wires leading to the switch are labeled "L1" and "1," respectively. Wires leading to the lamp are labeled "1" and "L2," respectively. The standard binary notation is used to show the status of the switches and lamp (0 for unactuated or de-energized; 1 for actuated or energized), and a truth table is showing how the logic works.



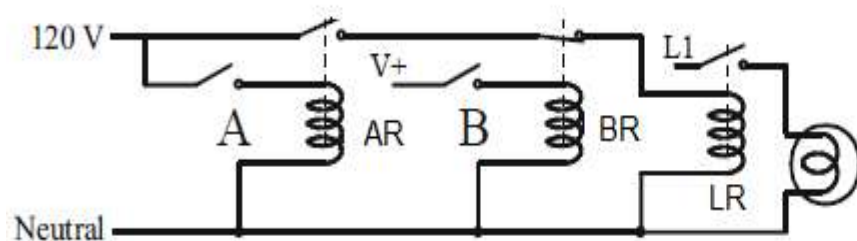
Example.2 - AND Circuit: Two switches labeled A and B are wired in series controlling a lamp. The lamp is **on** when switch A is **on** (closed) and switch B is **on** (closed), as shown below:



Now, the lamp energizes only if contact A and contact B are simultaneously actuated. A path exists for current from wire L1 to the lamp (wire 2) if and only if both switch contacts are closed.



Example.3 - NOT Circuit: A lamp needs to be turned **on** when switch A is **on** (closed) and switch B is **off** (open). The possible combinations of the 2 switches, are:



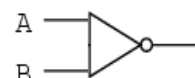
NOT Truth Table

A	B	Lamp
off	off	off
off	on	off
on	off	on
on	on	off

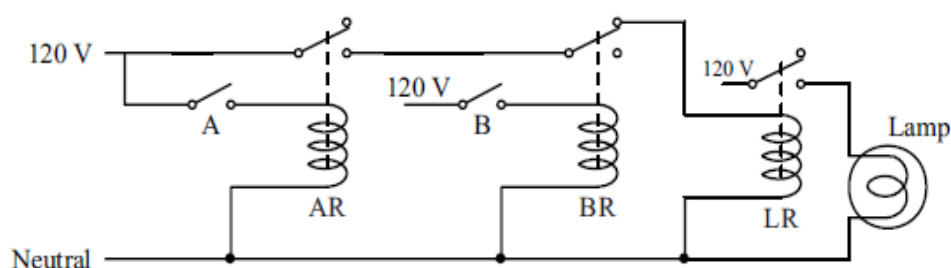
The logical inversion, or NOT, function can be performed on a contact input simply by using a normally-closed contact instead of a normally-open contact. When input (switch) A is **on** (closed) and input (switch) B is **off** (open) then the lamp is **on**.



A	B	Output
0	0	0
0	1	0
1	0	1
1	1	0



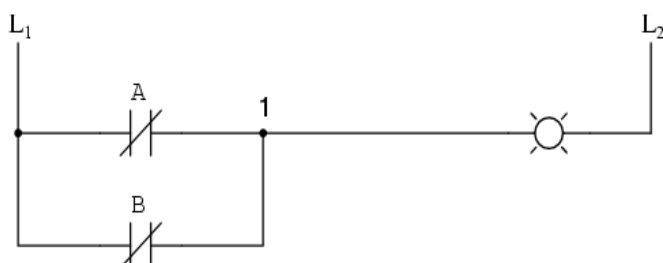
Example.3.1: NAND (NOT AND) Circuit: The lamp turns on only when both switch A is off (open) and switch B is off (open).



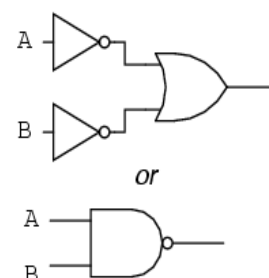
NAND Truth Table

A	B	Lamp
off	off	on
off	on	on
on	off	on
on	on	off

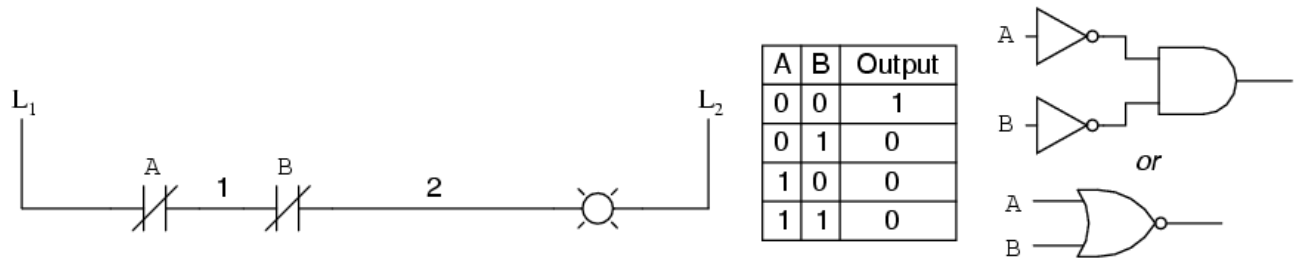
Take the OR function and invert each "input" through the use of normally-closed contacts, ending up with a NAND function. This effect of gate function is identified with the inversion of input signals.



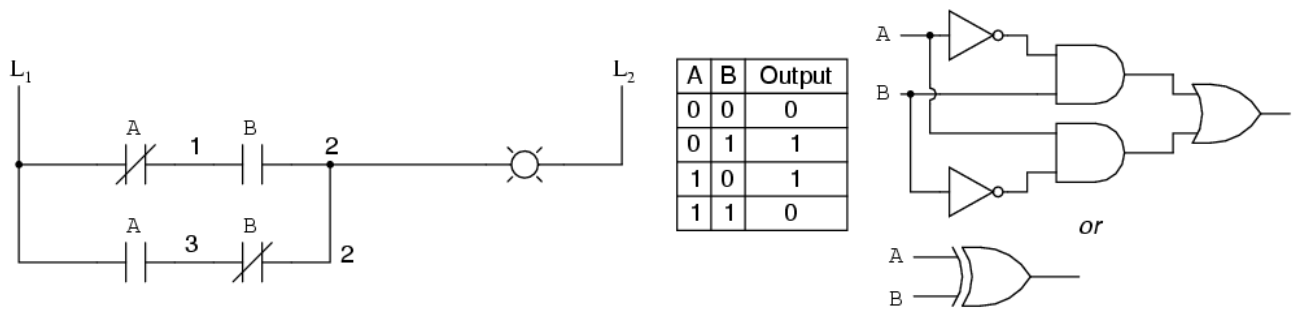
A	B	Output
0	0	1
0	1	1
1	0	1
1	1	0



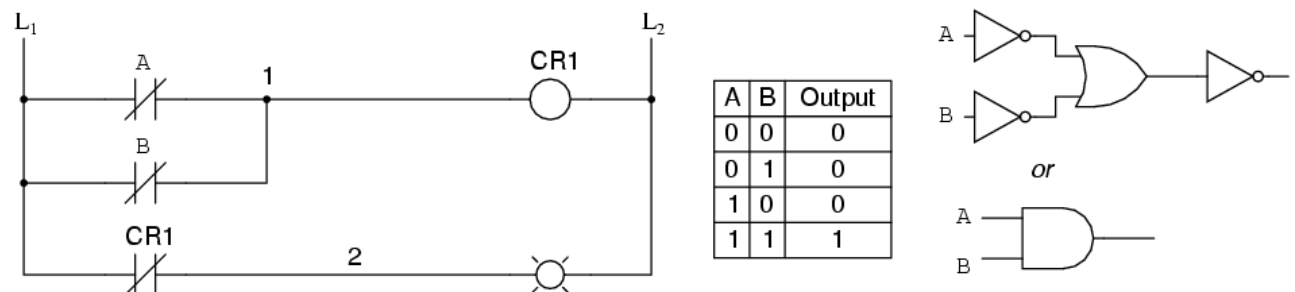
Example.3.2: NOR Circuit: Likewise, if we take our AND function and invert each "input" through the use of normally-closed contacts, we will end up with a NOR function.



Example.3.3: Exclusive-OR Circuit: We can build combinational logic functions by grouping contacts in series-parallel arrangements, as well. In the following example, we have an Exclusive-OR function built from a combination of AND, OR, and inverter (NOT) gates.



Control Relay 1 (CR₁): The normally-closed contact actuated by relay coil CR₁ provides a logical inverter function to drive the lamp opposite that of the switch's actuation status. From switch A to the coil of CR₁, the logic function is non-inverted. Applying this inversion strategy, such as the OR-to-NAND, we can invert the output with a relay to create a non-inverted function.

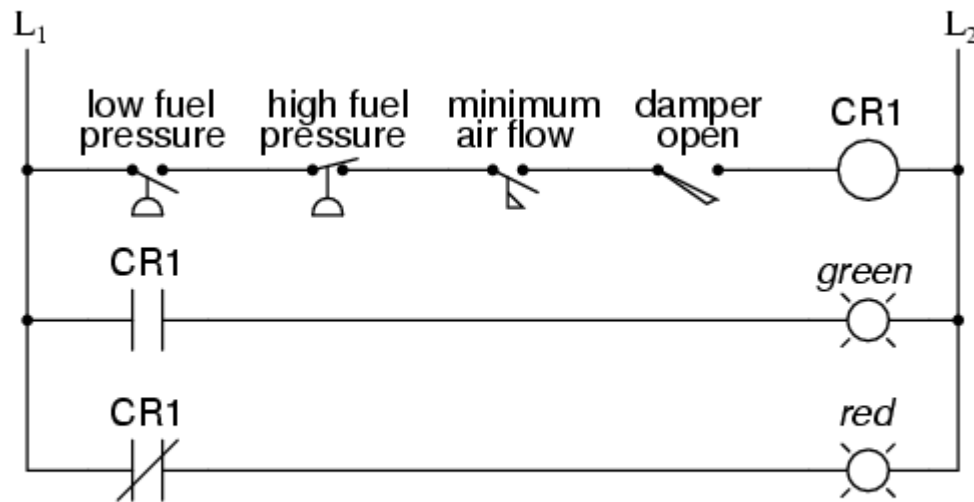


Burner Control: A good example of this, is a burner control for large combustion furnaces, to be started safely. The control system requests a "permission" from *several process switches*, including high and low fuel pressure, air fan flow check, exhaust stack damper position, access door position, etc. Each process condition is called as "*permissive*", and each switch contact is wired in series, so that if any one of them detects an unsafe condition, the circuit will not start.

If all permissive conditions **are met**, the CR1 is energized and the safety green lamp is always lit. In real life, more than just a green lamp would be energized, usually a control relay or fuel valve solenoid would be placed in that rung of the circuit to be energized when all the permissive contacts

were all closed. If any one of the permissive conditions **are not met**, the series string of switch contacts will be broken, CR2 will de-energize, and the warning red lamp will light.

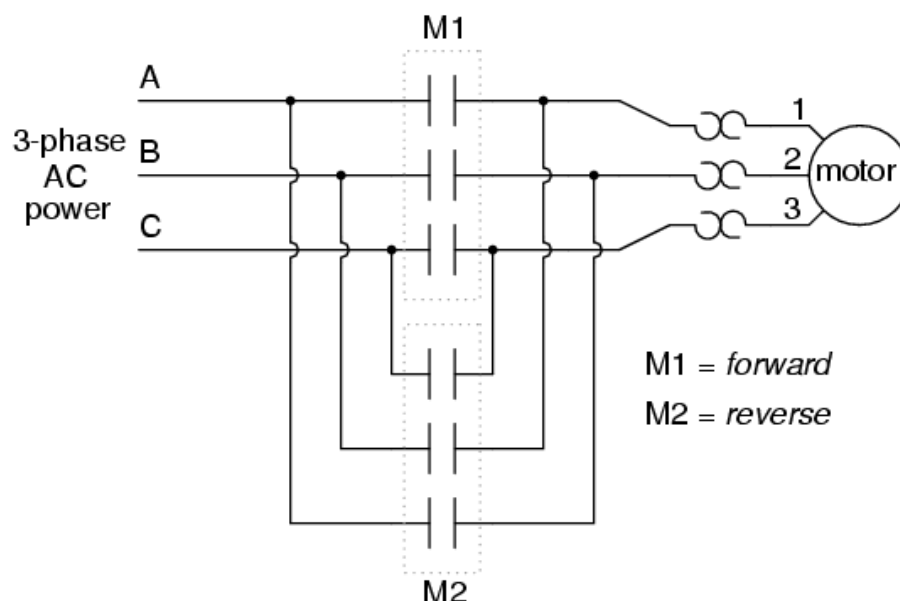
Burner Control



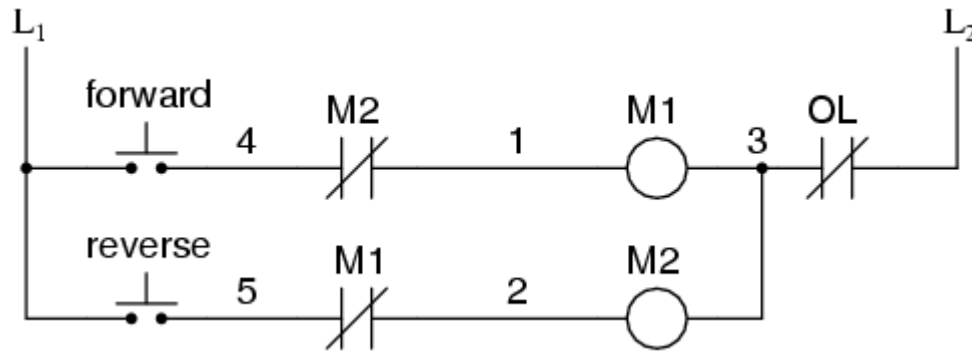
Green light = *conditions met: safe to start*

Red light = *conditions not met: unsafe to start*

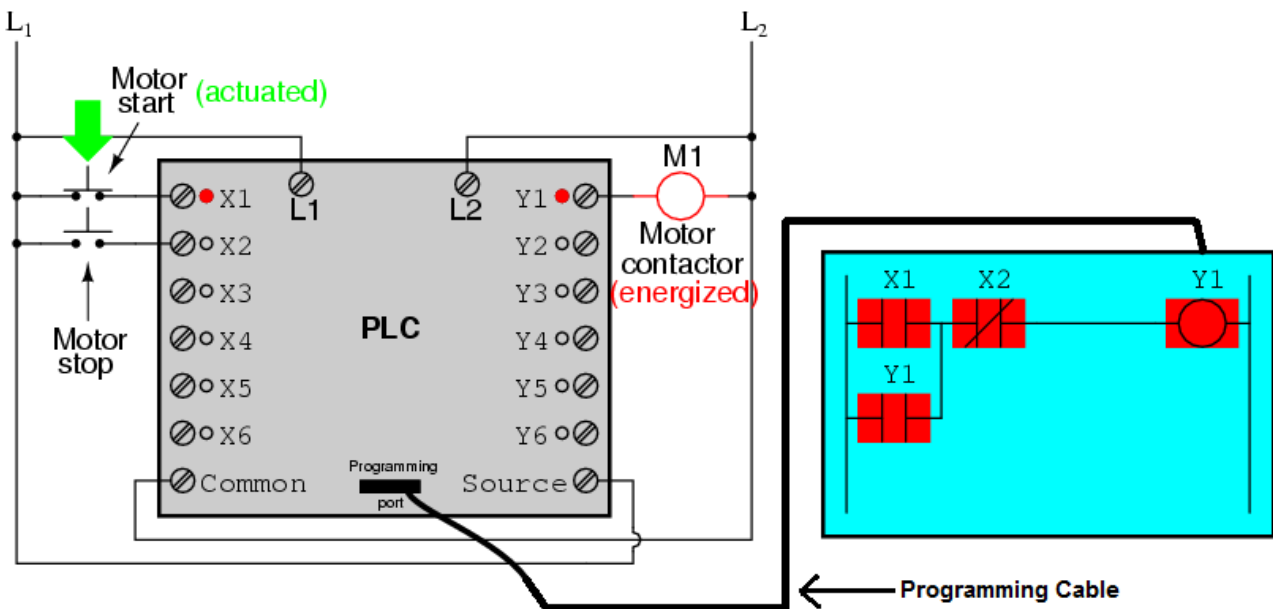
Electric Motor Control: An example of this is a reversible motor control, where two motor contactors are wired to switch polarity (or phase sequence) to an electric motor, and it's not necessary the forward and reverse contactors be energized simultaneously. As can be seen below, when contactor M₁ is energized, the 3 phases (A, B, and C) are connected directly to terminals 1, 2, and 3 of the motor, respectively. However, when contactor M₂ is energized, phases "A" and "B" are reversed. "A" is connected to motor terminal 2 and "B" is connected to motor terminal 1. This reversal of phase wires results in the motor spinning in the opposite direction.



The circuit is designed, so that the energization of one contactor prevents the energization of the other. This is called *interlocking*, and it is accomplished through the use of auxiliary contacts on each contactor. Thus, when M_1 is energized, the normally-closed auxiliary contact on the second rung will be open, thus preventing M_2 from being energized, even if the "Reverse" pushbutton is actuated. Likewise, M_1 's energization is prevented when M_2 is energized. Note, as well, how additional wire numbers (4 and 5) were added to reflect the wiring changes.



PLC Input Terminals: PLCs are industrial computers, with "input and output" signals, typically 120 volts AC. The "input" terminals are to interpret the logical states from sensors and switches. The output terminals, are to output signals to power lights, solenoids, contactors, small motors, and other devices lending themselves to on/off control. Take, for instance, a motor start-stop control circuit:



When the "Stop" pushbutton is released, the input X2 de-energizes, returning to its normal, "closed" state. The motor, however, will not start again until the "Start" pushbutton is actuated, because the "seal-in" of Y1 is lost. The *fail-safe* design is very important in PLCs, as it is in electromechanical relay-controlled systems. In this motor circuit, for example, there is a problem: if the input wiring for X2 (the "Stop" switch) fails to open, there would be no way to stop the motor.

Decimal Numerical System: Decimal numerical system is defined with its basis 10 and decimal positioning from right to left, and it consists of digits 0,1,2,3,4,5,6,7,8,9. This means that the right-most digit is multiplied by 1 in total sum; next digit to it is multiplied by 10, next one by 100, etc. Operations of addition, subtraction, division and multiplication in decimal numerical system are well known, so we will not detail these.

Example:

$$\begin{array}{r} 4631 \\ \begin{array}{l} \text{---} 1 * 10^0 = 1 \\ \text{---} 3 * 10^1 = 30 \\ \text{---} 6 * 10^2 = 600 \\ \text{---} 4 * 10^3 = 4000 \end{array} \\ \hline \text{total is } 4631 \end{array}$$

Binary Numerical System: Binary numerical system is quite different from the decimal that we got used to in common life. Its basis is 2 and each digit can have one of two values, “1” or “0”. Binary numerical system is used for computers and microcontrollers, because it is much easier for processing than decimal. Usually, binary number consists of 8, 16 or 32 binary digits.

Example:

10011011 - binary number with 8 digits

The conversion of binary number to decimal value is done by totaling the sum on the right. Depending on the position in the binary number, digits carry different “weight” multiplied by themselves, and totaling them all gives us an understandable decimal number.

$$\begin{array}{r} 10011011 \\ \begin{array}{l} \text{---} 1 * 2^0 = 1 \\ \text{---} 0 * 2^1 = 0 \\ \text{---} 0 * 2^2 = 0 \\ \text{---} 1 * 2^3 = 8 \\ \text{---} 1 * 2^4 = 16 \\ \text{---} 0 * 2^5 = 0 \\ \text{---} 0 * 2^6 = 0 \\ \text{---} 1 * 2^7 = 128 \end{array} \\ \hline \text{total is } 155 \end{array}$$

The basic rules that apply to binary additions work similar to decimal numerical system, that is, we add the digits of the same weight. If both digits added are zero, the result remains zero, while “0” and “1” total “1”. Two ones give zero, but one is carried to the left position.

Example:

$$\begin{array}{r} 1 \\ + 0 \\ \hline 1 \end{array} \quad \begin{array}{r} 0 \\ + 1 \\ \hline 1 \end{array} \quad \begin{array}{r} 0 \\ + 0 \\ \hline 0 \end{array} \quad \begin{array}{r} 1 \\ + 1 \\ \hline 10 \end{array}$$

We can check by converting these numbers to decimal system and adding them. Value of the first number is 10, value of the second is 9 and 19 as result, which means that operation was done correctly. Problem occurs when the result is greater than can be represented with given number of binary digits. There are various solutions, one of them being expanding the number of binary digits like in the example below:

$$\begin{array}{r} 1010 \text{ first number} \\ 1001 \text{ second number} \\ \hline 10011 \text{ sum} \end{array}$$

Subtraction works on the same principles as addition does. Two zeros give zero in result, as do two ones, while subtraction of one from zero requires borrowing one from the higher position in binary number. Conversion of numbers to decimal system gives as values 10 and 9, with the result of subtraction of 1, which is correct.

Example:

$$\begin{array}{r} 1010 \text{ first number} \\ 1001 \text{ second number} \\ \hline 0001 \text{ difference} \end{array}$$

Hexadecimal Numerical System: Hexadecimal numerical system has number 16 for basis. Therefore, there are 16 different digits used in this system. These are "0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, and F". Letters A, B, C, D, E and F represent values 10, 11, 12, 13, 14, 15 and are used for the sake of easier notation. As with binary numerical system, we can apply the same formula here for determining the greatest decimal number that can be represented with a given number of hexadecimal digits.

Example:

$$16^2 - 1 = 256 - 1 = 255$$

Usually, hexadecimal numbers have prefix "\$" or "0x" to emphasize the fact that hexadecimal system is used. Thus, number A37E should be represented with \$A37E or 0xA37E. No calculations are needed for converting the hexadecimal number to binary system - it is simple substituting of hexadecimal digits with binary ones. Since maximum value of hexadecimal digit is 15, 4 binary digits are required per one hexadecimal.

$$\begin{array}{r} \$E4 = 1110 \ 0100 \\ \quad \quad \quad \underline{\quad} \quad \underline{\quad} \\ \quad \quad \quad E \quad 4 \end{array}$$

Example: Converting both numbers to decimal system, gives us value 228 which is correct. In order to calculate decimal equivalent of hexadecimal number, each digit of number should be multiplied by 16 raised to power equal to the position in the number and then added altogether.

$$\begin{array}{r} A37E \\ \begin{array}{l} \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \end{array} \begin{array}{l} 14 * 16^0 = 14 \\ 7 * 16^1 = 112 \\ 3 * 16^2 = 768 \\ 10 * 16^3 = 40960 \end{array} \\ \hline \text{total is } 41854 \end{array}$$

Addition works similar to two previous numerical systems:

$$\begin{array}{r} \$3A2B \quad \text{first number} \\ +\$A9C1 \quad \text{second number} \\ \hline \$E3EC \quad \text{sum} \end{array}$$

Conclusion: Binary numerical system remains the most commonly used, decimal system the most intelligible, while hexadecimal is somewhere in between, however, binary and decimal, the most important numerical system to us.

Design of a Process Control System: *First*, select a system to be controlled. Automated systems can be a machine or a process and can also be called an process control system. PLC controllers send a signal to external output devices (operative instruments) to control the system functions in an standard manner (recommended to draw a block diagram of operations' flow).

Secondly, is necessary to specify all input and output instruments that will be connected to the PLC controller, with an identification of all input and output instruments, and corresponding designations. Input devices are various switches, sensors and so on. Output devices are solenoids, electromagnetic valves, motors, relays, magnetic starters, as well as, instruments for sound alarm and light signalization.

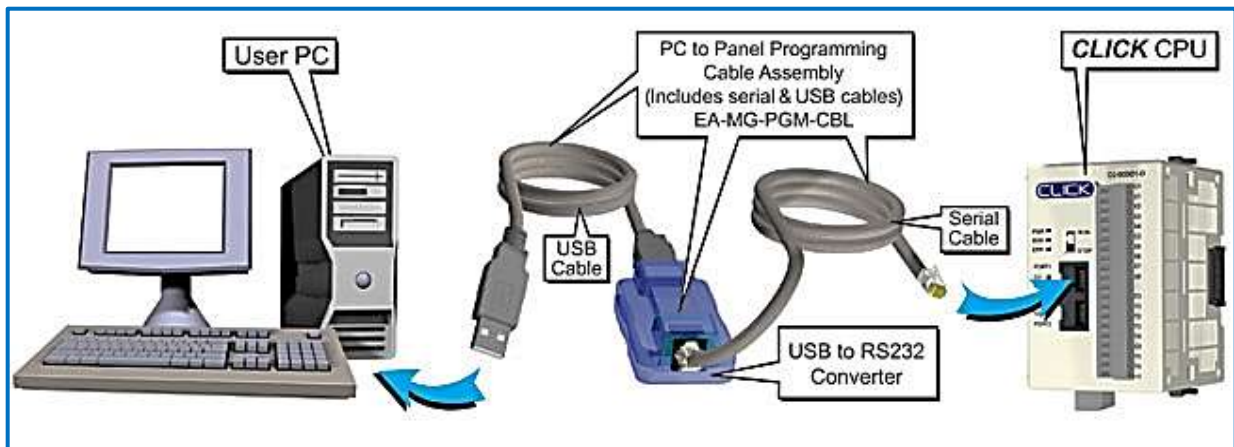
Third, make a ladder diagram for a program by following the sequence of operations, determined in the first step. When the programming is finished, make a checkup for any existing errors in the program code (using functions for diagnostics) and, if possible, an entire operation is simulated. Before this system is started, you need to check once again whether all input and output instruments are connected to correct inputs or outputs. Finally, program is entered into the PLC controller memory.

Components of a PLC: The power supply used is 100- 240V AC or 24V DC, and all PLCs consist of these basic components:

- **CPU:** This is the brain of the PLC which performs arithmetic and logical operations, monitors input and outputs, communicates with memory, scanning of program, etc.
- **Memory:** Memory is used to store data for use in programs or to change the state of input and output. There are two types of memory, RAM and ROM. RAM (Random Access Memory) *can be written, deleted and rewritten*, used to store Program Memory. ROM (Read Only Memory) *can only be written once* and cannot be deleted. It is used to store the core programs of the PLC and cannot be altered by the user.
- **Inputs and Outputs:** Inputs and Outputs are used to communicate the PLC with the outside world. Depending upon the type of PLC, the number of inputs and outputs may vary. Modular type of PLCs can increase the number of inputs and outputs depending upon the application at hand.

Communication Cable: A PLC controller is linked with a PC computer through an RS-232 cable, for simulating and downloading ladder logic programs, but actually some of the communication cables now are USB supported. One end of the cable is connected to a serial PC port (9-pin or 25-pin connector), while the other end is connected to an RS-232C connector.

If a USB port is available on the PC, an automation direct USB to RS232 PC to panel programming cable assembly, EA-MG-PGM-CBL can be used, to connect between the USB port on the PC and the RJ12 connector on the CPU's PORT1. The RS232 port is used to download or upload a program and for communication between the personal computer and a PLC.



Programming Basics: All PLCs are programmed either using:

- ✓ **Ladder Logic:** Each program line occupies a rung in the Ladder. The CPU scans the first rung then the second and it goes on in the ascending order.
- ✓ **Functional Block Diagram:** Each input and output is denoted by using block diagram and logic gates. It is simpler compared to Ladder Logic.

5. HUMAN-MACHINE INTERFACE (HMI):

Human-Machine Interfaces: Is the part of the machine that provides a graphics-based visualization of a control and monitoring system and handles with the human-machine interaction. A Human Machine Interface (HMI) is exactly what the name implies; a graphical interface that allows humans and machines to interact, which can vary widely, from control panels for nuclear power plants, to the basic screen on an iPhone.

The Human Machine Interface (HMI) includes the electronics required to signal and control the state of industrial automation, where multiple equipment are linked by a host control system that can be accessed and controlled. The term "user interface" is often used in the context of personal computer systems and electronic devices. An HMI is typically local to one machine or piece of equipment, and is the interface method between the human and the equipment/machine.

HMI is very broad term that also includes MP3 players, industrial computers, household appliances, and office equipment, however, an HMI is much more specific to manufacturing and process control systems. An HMI provides a visual representation of a control system and provides real time data acquisition. This broad concept of user interfaces include the interactive aspects of computer operating systems, hand tools, heavy machinery operations, and process controls.

In industrial processes, an HMI is the centralized control unit for manufacturing lines, equipped with data recipes, event logging, video feed, and safety triggering, so that one may access the system at any moment for any purpose. For a manufacturing line to be integrated with an HMI, it must first be working with a Programmable Logic Controller (PLC). It is the PLC that takes the information from the sensors, and transforms it to a Boolean algebra, so the HMI can decipher and make decisions.



History of HMI: HMI products originated from the need to make machinery easier to operate, while producing optimal outputs. Predecessors of HMI include the Batch Interface (1945-1968), Command-Line User Interface (1969-Present), and the Graphical User Interface (1981-Present).

- ✓ The Batch Interface is a non-interactive user interface, where the user specifies the details to the batch process in advance, and receives the output when all the processing is done. Sin-

ce the batch process does not allow additional inputs, once the process has begun; it is very problematic in modern manufacturing lines.

- ✓ The Command-Line Interface is a mechanism that interacts with a computer operating system or software by typing commands to perform specific tasks. The concept of the Command-Line interface originated when teletypewriter machines were connected to computers in the 1950s. A basic example of Command-Line Interface would be the Windows Disk Operating System “DOS” which dominated up to 1990. Over time, interfaces became highly complex and extremely easy to use.
- ✓ The Graphical User Interface has allowed people to interact with programs in more ways than typing, such as computers, hand-held devices such as MP3 Players, Portable Media Players or Gaming devices, household appliances, and office equipment with images, rather than text commands.

HMI Applications: Is used throughout various industries including manufacturing plants, vending machines, food and beverage, pharmaceuticals, and utilities, just to name a few. The HMI allows supervisory control and data acquisition in the entire system, and parameter changes are feasible to the operator's choosing. For example, in metals manufacturing, an HMI might control how metal is cut and folded, and how fast to do so. An HMI offers improved stock control and replenishment, so the fewer journeys are required out to the vendors. HMIs are used in bottling processes to control all aspects of the manufacturing line, such as speed, efficiency, error detection and error correction. Utility companies may use HMIs to monitor water distribution and waste water treatment.



Advantages of an HMI: The greatest advantage of an HMI is the user-friendliness of the graphical interface, that contains color coding that allows for easy identification (for example: red for trouble). Some technological advantages the HMI offers are: converting hardware to software, eliminating the need for mouse and keyboard, and allowing kinesthetic computer/human interaction. Pictures and icons allow for fast recognition, easing the problems of illiteracy, can reduce the cost of product manufacturing, and potentially increase profit margins and lower production costs. HMI devices are now extremely innovative, capable of higher capacity and elaborate functions than ever before.

HMI Functions: As an example, first consider all components that are necessary to make a manufacturing control system to operate and the machinery that performs the work required in the production line. Next, consider the various input/output sensors that monitor temperature, speed, pressure, weight and feed rate. Third, decide on the Programmable Logic Controller (PLC) that will receive the data from the input/output sensors, and converts the data into logical combinations.

It is important to make the appropriate selection. An HMI that is located in a water plant might have various water seals around its perimeter, as opposed to an HMI that is located in a pharmaceutical warehouse. Some applications may only require a small, black and white touch screen monitor. The physical properties are extremely important considering the operating environment, and what safety measures the HMI needs to attend the codes. A specific size may be needed due to space limitations and physical properties may include the processor and memory of the HMI.

Programming Softwares: There are three main categories of programming softwares to choose: proprietary, hardware-independent and open software, as defined below:

- Proprietary software is the software that the manufacturer provides, which is typically easy to use and allows for quick development. The drawback is that proprietary software will only run on a specific hardware platform.
- Hardware-independent software is a third party software developed to program on several different types of HMIs, and gives the user much more freedom for an HMI selection. The downside to hardware-independent software is not as user-friendly as the proprietary.
- Open software should only be selected by the advanced programmer. It allows the user to have a complete openness in the design process.

HMI Basic Types: There are three basic types of HMIs: the pushbutton replacer, the data handler, and the overseer, as defined below:

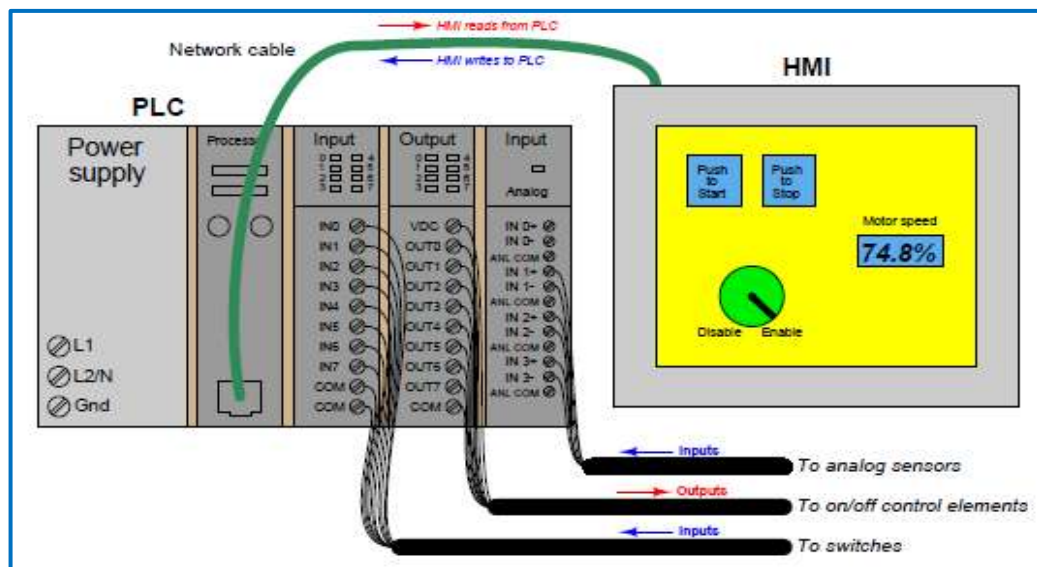
- The pushbutton replacer consist of hundreds of pushbuttons and LEDs performing different operations, before the HMI came into existence. The pushbutton replacer HMI has streamlined manufacturing processes, centralizing all the functions of each button into one location.
- The data handler is perfect for applications requiring constant feedback from the system, or printouts of the production reports. It's necessary to ensure the HMI screen is big enough for such things as graphs, visual representations and production summaries. The data handler includes such functions as recipes, data trending, data logging and alarm handling/logging.
- The overseer HMI is very beneficial and involves SCADA or MES softwares. SCADA (Supervisory Control and Data Acquisition) is a type of industrial control system (ICS), with coded signals that provide control of remote equipment. MES (Manufacturing Execution Systems) is used in manufacturing and work in real time to enable the control of multiple elements of the production process. The overseer HMI will most likely need to run Windows, and have several Ethernet ports.

HMI & PLC Combination: In order to program an HMI to operate a PLC properly, all the registers of the PLC must be known. A good way to learn how to program a PLC via an HMI is to first start working with the PLC and the software it came with. This also helps to understand how to operate the

PLC without the HMI. That knowledge will easily transfer over when the user is ready to connect the two units together.

Wiring: This connection is as simple as connecting a USB, RS-232, RS-485, RJ-45 between the HMI and PLC. In many of them, both units are equipped with wireless features, making the connection much easier. However, the wiring between the PLC and the actual automation line could be much more difficult. Depending on the complexity of the application, a fieldbus protocol may be required for the PLC. A wiring schematic from the HMI to the PLC is highly recommended.

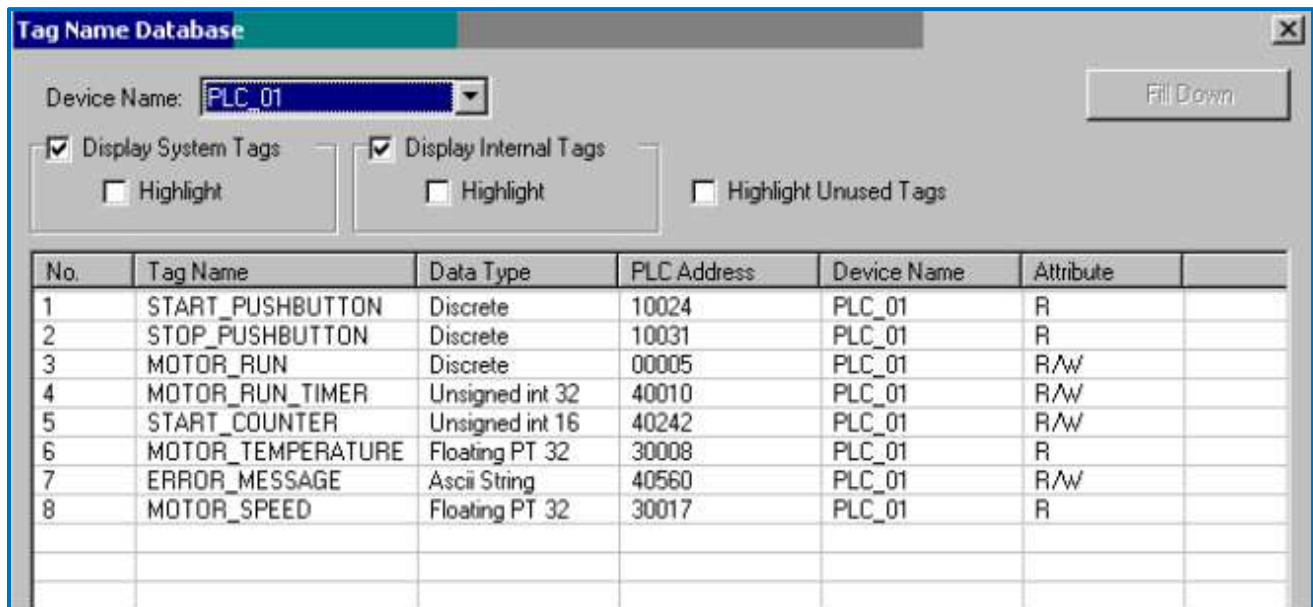
HMI Panels: Most industrial HMI panels come equipped with touch-sensitive screens, allowing operators to press their fingertips on displayed objects to change screens, view details on portions of the process, etc. HMI displays to read and write data via a digital network to one or more PLCs. Graphical objects arrayed on the display screen, often show real-world indicators and switches, in order to provide a familiar interface for operations personnel. A “pushbutton” object on the face of an HMI panel, for example, can be configured to write one bit of data to the PLC, in a manner similar to a real-world switch writing one bit of data to the PLC’s input register.



Communication Protocols: Selecting a PLC depends on what protocol this PLC or controller uses. If the controller uses Modbus RTU, then the user should select Modbus RTU Slave. If the controller uses ASCII, then the user should select the Universal ASCII slave. The Profibus extension is similar to a power strip that extends one input/output to multiple input/outputs by connecting to the expansion port of the PLC.

The HMI must be connected to a device, whether it is through Ethernet (RJ45), serial communication (RS232, USB or RS422), or wireless. The two devices must be in synchronous baud (rate symbols per second, or pulses per second) rates, to avoid miscommunication. Windows CE is an embedded operating system, which translate to a minimalistic copy of windows for smaller devices such as HMIs and cellular phones. The Codesy's packet is a proprietary third party software, which allows the user to program an HMI and a PLC.

Modern HMI panels and software are almost exclusively tag-based, with each graphic object on the screen associated with at least one data tag name, or data points (bits, or words) in the PLC, by way of tag names database files resident in the HMI. Graphic objects on the HMI screen either accept (read) data from the PLC to present useful information to the operator, writing data to the PLC. The *task of programming an HMI* unit consists of building a tag name database and then drawing screens to illustrate the process to a good level of detail, as shown below:



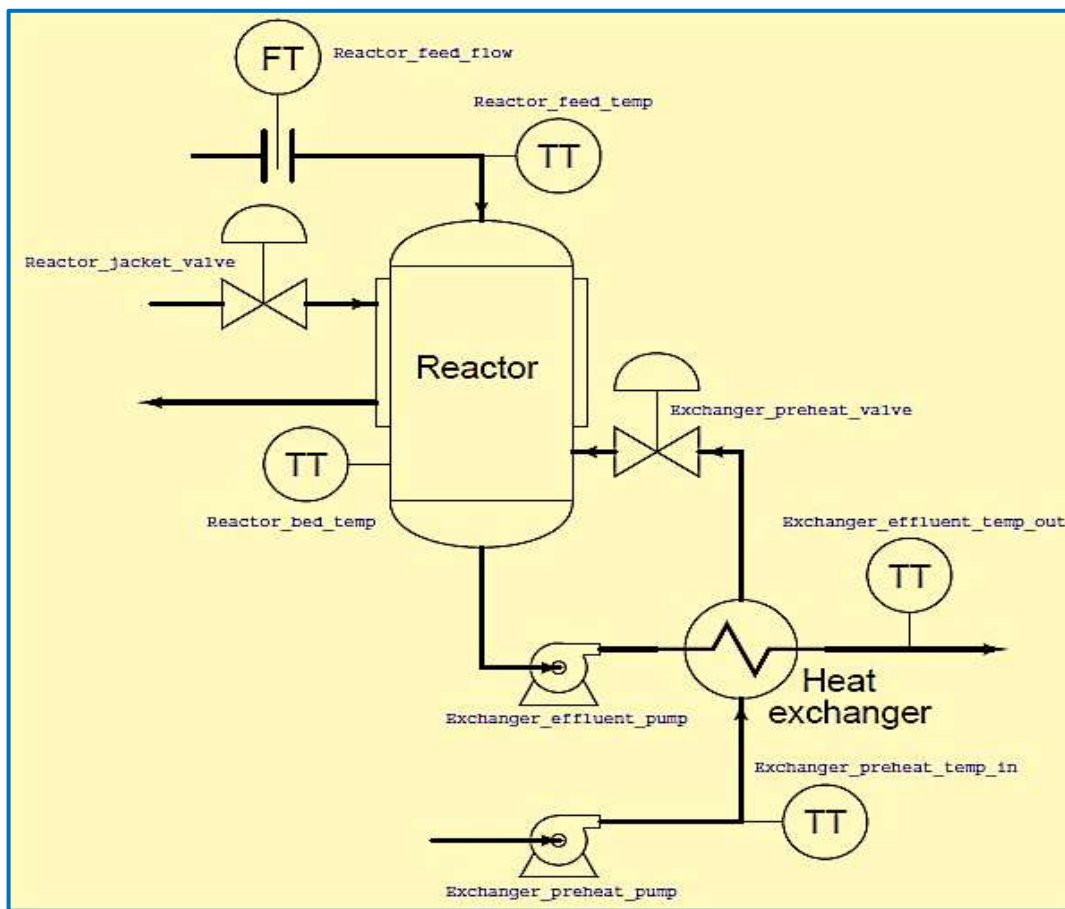
No.	Tag Name	Data Type	PLC Address	Device Name	Attribute
1	START_PUSHBUTTON	Discrete	10024	PLC_01	R
2	STOP_PUSHBUTTON	Discrete	10031	PLC_01	R
3	MOTOR_RUN	Discrete	00005	PLC_01	R/w
4	MOTOR_RUN_TIMER	Unsigned int 32	40010	PLC_01	R/w
5	START_COUNTER	Unsigned int 16	40242	PLC_01	R/w
6	MOTOR_TEMPERATURE	Floating PT 32	30008	PLC_01	R
7	ERROR_MESSAGE	Ascii String	40560	PLC_01	R/w
8	MOTOR_SPEED	Floating PT 32	30017	PLC_01	R

The tag database is accessed and edited using the same software to create graphic images in the HMI. In this particular example there are several tag names (e.g. START PUSHBUTTON, MOTOR RUN TIMER, ERROR MESSAGE, MOTOR SPEED) associated with data points within the PLC's memory (in this example, the PLC addresses are shown in Modbus register format). In many cases the tag name editor will be able to display corresponding PLC memory points in the same manner as they appear in the PLC programming editor software (e.g. I:5/10, SM0.4, C11, etc.).

HMI Programming: Like Programmable Logic Controllers, the capabilities of HMIs have been steadily increasing while their price decreases. Modern HMIs support graphic trending, data archival, advanced alarming, and even web server ability allowing other computers to easily access certain data over wide-area networks. The ability of HMIs to log data over long periods of time relieves the PLC of having to do this task, which is very memory-intensive.

This way, the PLC merely “serves” current data to the HMI, and the HMI is able to keep a record of current and past data using its vastly larger memory reserves. Some modern HMI panels even have a PLC built inside the unit, providing control and monitoring in the same device. Such panels provide terminal strip connection points for discrete and even analog I/O, allowing all control and interface functions to be located in a single panel-mount unit.

Take for instance, this example of a process with several data points defined in a PLC control system and displayed in an HMI:



Listing all these tags in *alphabetical order*, the association is immediately obvious:

- Exchanger effluent pump
- Exchanger effluent temp out
- Exchanger preheat pump
- Exchanger preheat temp in
- Exchanger preheat valve
- Reactor bed temp
- Reactor feed flow
- Reactor feed temp
- Reactor jacket valve

As can be seen from this tag name list, all the tags are directly associated with the heat exchanger located in one contiguous group, and all the tags are directly associated with the reactor located in the next contiguous group. In this way, naming of tags serves to group them in hierarchical fashion, making them easy for the programmer to locate at any future time in the tag name database.

All the tag names shown here lack space characters between words, that is, instead of “Reactor bed temp”, a tag name uses hyphens or underscore marks as spacing characters, “Reactor_bed_temp”, since spaces are generally assumed by computer programming languages to be delimiters (separators between different variable names).

Interface Flexibility: The HMI can fully support the most complex applications, with multiple screens and several routines running. If the user is looking to program an HMI with something more simplistic, he/she can have instructions for the controller directly written onto the HMI. Every HMI comes with different features; some may play sound, play video, or even may have remote access control. The design of the actual interface should be optimized for specific applications, taking into consideration all the capabilities of the typical user, as well as the environmental aspects such as noise, lighting, dust, vision and technological curves.

HMI combines all the control features that are typically found throughout the automation line and places them in one centralized location, eliminating the need to run to a red pushbutton that will stop your line in an emergency. With remote access, the operator does not need to be anywhere near the automation line to start/stop or monitor production. With remote access, the operator can have all the same features, on your centralized unit in a smaller compact form. Simplicity is also a big factor in the usability of an HMI. The screens and functions provide for easy training to supervise the automation line.

PLC Training: A very good way of learning the PLC language and programming is this unit, a student-built PLC trainer, housed inside an attached case. This trainer unit contains an Allen-Bradley MicroLogix 1100 PLC along with input switches and output indicator lights, and also includes an HMI touch-screen panel on a fold-down bracket:



Once you have learned the basic steps for entering, running, and saving a PLC program, you are ready to begin building your knowledge of the language's vocabulary and grammar. In computer programming (of all types), there are different functions of the language one must become familiar with in order to do useful tasks.

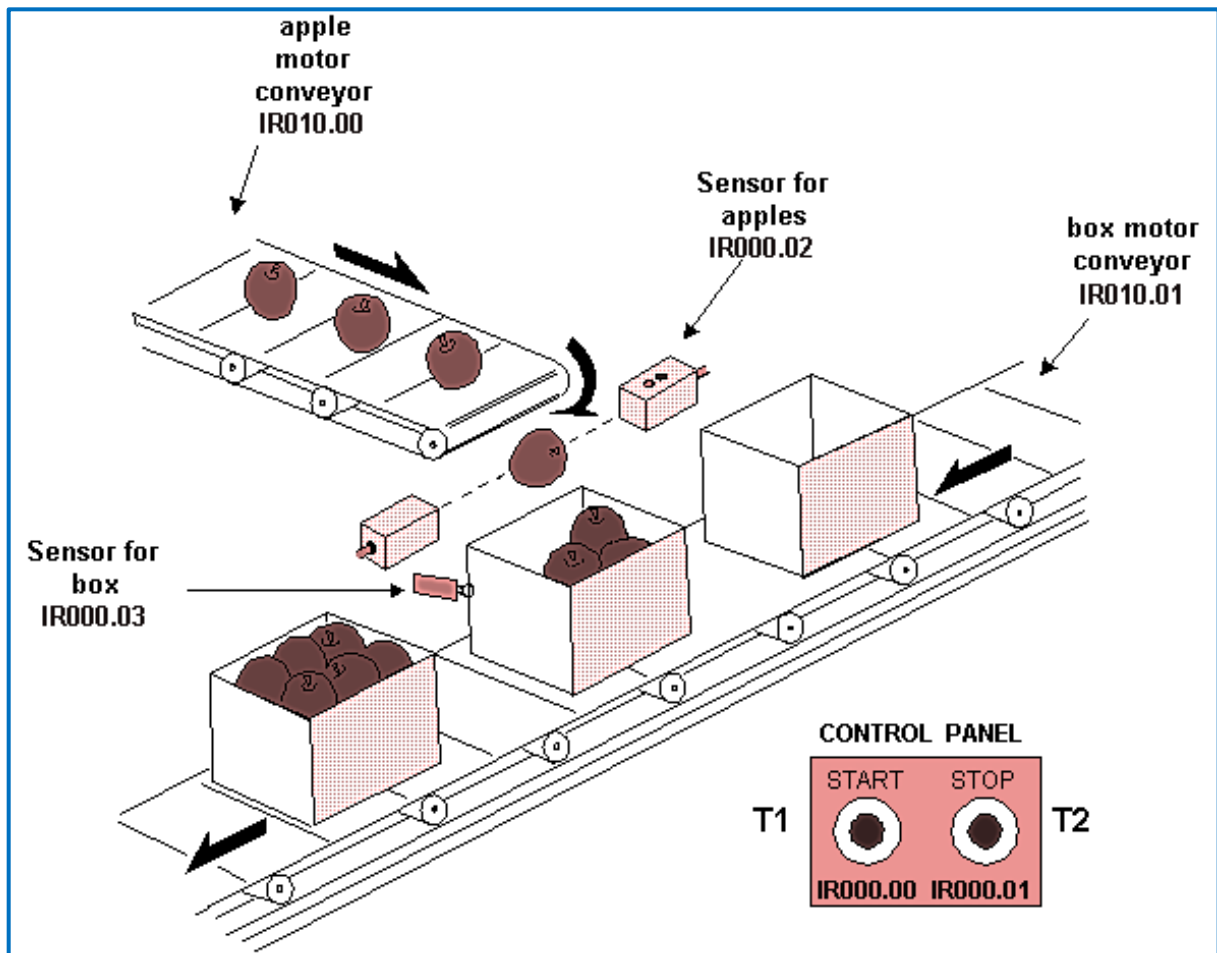
For example, if you open up the pages of almost any computer programming book, somewhere near the beginning you will find a demonstration program called "Hello World", on the computer screen. It is an entirely useless program to run, but it is highly useful for the basics of program construction and text message functionality.

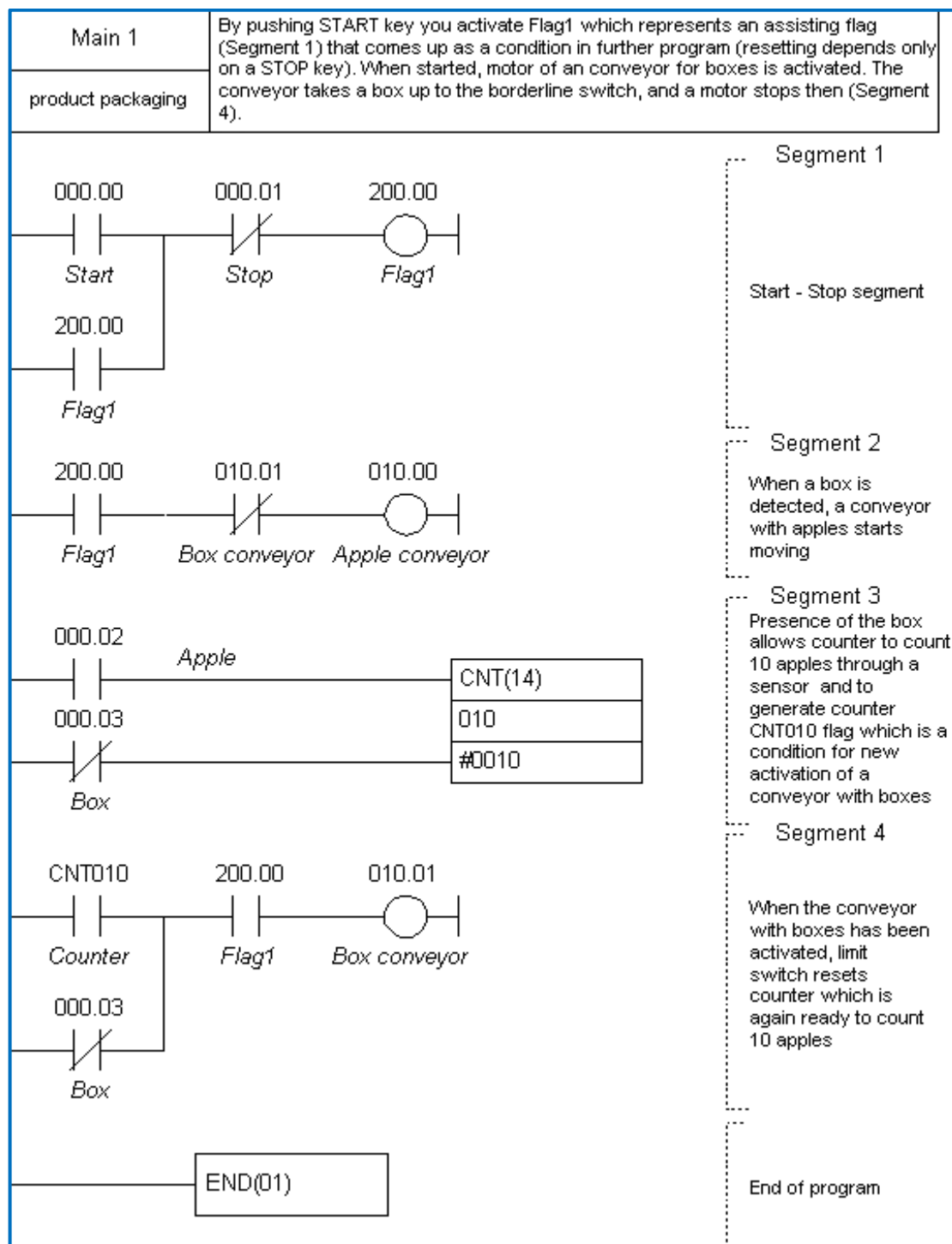
For example, every PLC provides instructions to perform the following tasks:

- Turn discrete outputs on and off;
- Count discrete events;
- Time events;
- Control events in a specific sequence;
- Compare numerical values (greater than, less than, equal, not equal);
- Perform arithmetic functions.

The reference manuals provided for a PLC describes in detail how to use each function, allowing you to directly explore how each function works, and to gain an understanding of each function by observing its behavior and also by making (inevitable) mistakes.

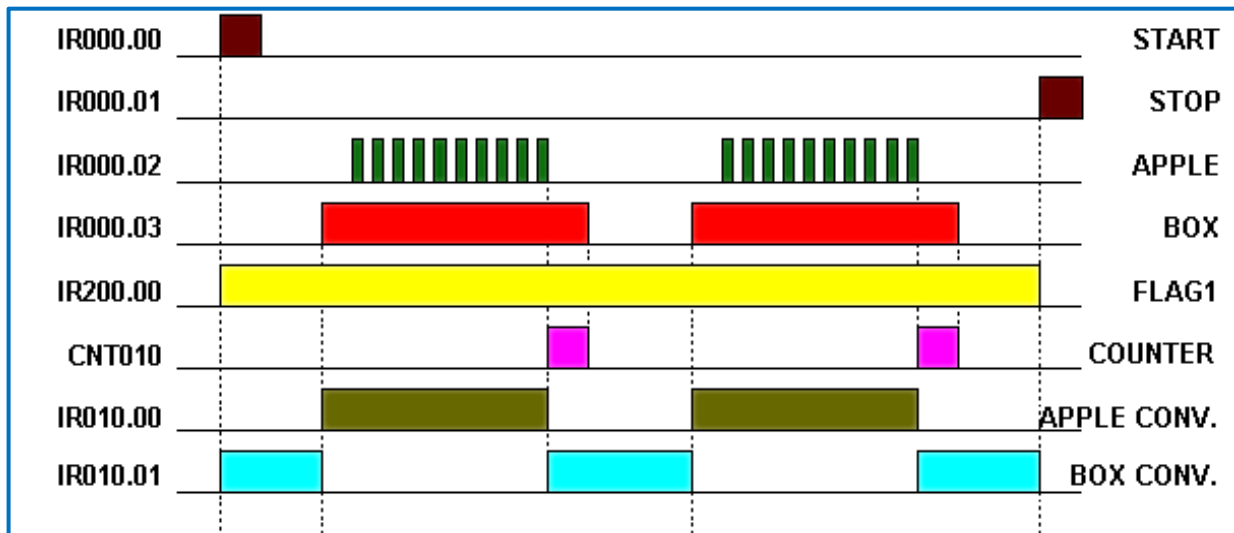
Example: Automation of Product Packaging: Product packaging is one of the most frequent cases for automation in industry. It can be encountered with small machines (packaging for food products) and large systems such as machines for packaging medications. The example, showed here may solve the classic packaging problem, with few elements of automation. There are a small number of needed inputs and outputs provided for the use of a PLC controller, which represent a simple and economical solution.



Ladder diagram:


When pushing the START key, the Flag1 is activated, which represents an assisting flag (Segment 1) that comes up as a condition in further program (resetting depends only on a STOP key). When started, the electric motor of a conveyor for boxes is activated. The conveyor takes the box up to the limit switch, and a motor stops (Segment 4). Condition for starting a conveyor with apples is actually a limit switch for a box. When a box is detected, a conveyor with apples starts moving (Segment 2).

Presence of the box allows counter to count 10 apples through a sensor used for apples and to generate counter CNT010 flag, which is a condition for new activation of a conveyor with boxes (Segment 3). When the conveyor with boxes has been activated, limit switch resets counter which is again ready to count 10 apples. Operations repeat until STOP key is pressed when condition for setting the Flag1 is lost. Picture below gives a time diagram for a packaging line signal.



PLC Simulators: Simulators are powerful ladder logic programs softwares, which allow the user to edit the ladder logic programs graphically, so that the programmer can test and debug the program before installation into its operating environment. The object of a PLC simulator is to 'fake out' the input into a PLC, if you are a programmer or an engineer who programs PLCs or even a technician in need of a quick way to test PLC functionalities, the simulator devices are recommended.

The main simulators for Programmable Logic Controllers are: Allen Bradley, GE Fanuc, Siemens, Modicon, Mitsubishi, Omron, Automation Direct and many others, free, educational or purchased softwares for PLC simulations, as described below:

RSLogix 500 is used with the earlier generation SLC500 and MicroLogix family of Programmable Logic Controllers. RSLogix 5000 is used with the new generation of "ControlLogix" Programmable Automation Controllers. The RSLogix 5000 uses tags, which are a powerful method of programming PLCs, which is an ideal tool for learning the fundamentals of ladder logic with tag-based addressing.

ProSim-II Simulations for LogixPro 500 is a basic interactive educational tool developed to assist students in the acquisition of the programming skills, used in the control of process equipment and systems. Simulating process equipment such as conveyors, bottling plants, and so on, presents the student with a far more realistic programming experience.

GE Fanuc PLCs family is Series 90-30, Series 90-70, Versamax and the PacSystems Rx3i, Rx7i. GE Fanuc programming software known as Cimplicity, now uses the family name of Proficy Machine Edition, as well as, PE for the HMI. Intellution FIX32, iFix and i Historian were all converted to the GE Fanuc family of software named Proficy.

GE Fanuc PLC Simulator is a program collection with 19 downloads. The most lightweight of them are the Modbus Slave and the Modbus Slave Simulator, while the largest one is C-more micro, including 12 freeware products like 3DView and EasyVeep.

Siemens S5 (6ES5-1XX to 6ES5-9XX) PLCs family provides the necessary tools to program Siemens PLCs. However, is worthwhile to completely replace the existing SIMATIC S5 by the modern SIMATIC S7. The Siemens SIMATIC S7 PLC family tree S7200/300/400/1200/1500, shows how the PLC's can be structured, programmed, documented, maintained and installed, also uses the SIMATIC S7-CFC (Continuous Function Chart) also designated as STEP 7 package.

S5 Simulator built in with the integrated simulation PLC, it is possible to test your S5 PLC programs without additional hardware, directly performed on your PC. SIMATIC S7-PLCSIM from Version 5.4 SP3, simulates a controller for functional testing of user blocks and programs for S7-300 and S7-400 on the programming device/PC. The facility to simulate the communication via MPI, PROFIBUS DP and TCP/IP is new and ensures a high degree of flexibility in the simulation.

Modicon Simulator allows users to program, run, and test Modicon ladder logic and simulated I/O without using a PLC. When a program is online with Modicon Simulator, the simulator responds in a similar manner to the type of PLC that is being simulated. Programs may be loaded, saved, edited, and executed while online with the simulator.

Modicon Simulator's I/O simulation capability allows a sequence of values to be created and written to the simulator's memory, opposed to actual I/O setup, which requires time-consuming and complicated wiring of physical devices. This sequence may be set up to reproduce the same I/O values that are received by the PLC from the equipment to which it is physically attached, and expected output values can be generated through the I/O Simulator.

Mitsubishi Electric PLCs family FX, FX2C, FX0N, FX0S, FX0, FX2N, FX2NC and MELSEC Models FX1S FX1N FX2N FX3U are the Programmable Controllers. GX Developer-FX is a condensed version of the currently available SW5D5C-GPPW-E, designed as a programming tool for the Mitsubishi programmable controller FX series. GX Simulator also enables you to simulate all your devices and application responses for realistic testing.

Omron family CS1G/H, CS1D and CJ1M, CJ1G-P, CJ2M, CJ2H use the CX-Simulator and a Plug-In MatrikonOPC Omron. CX-Programmer or CX-Designer simulates the interaction between an HMI and PLC programs. The CS1 medium-building-block PLCs can be configured from a wide array of CPU Units. The CJ1 Series of small Programmable Controllers provide all the functions required for everything from machine control to process control that enables flexible system configuration.

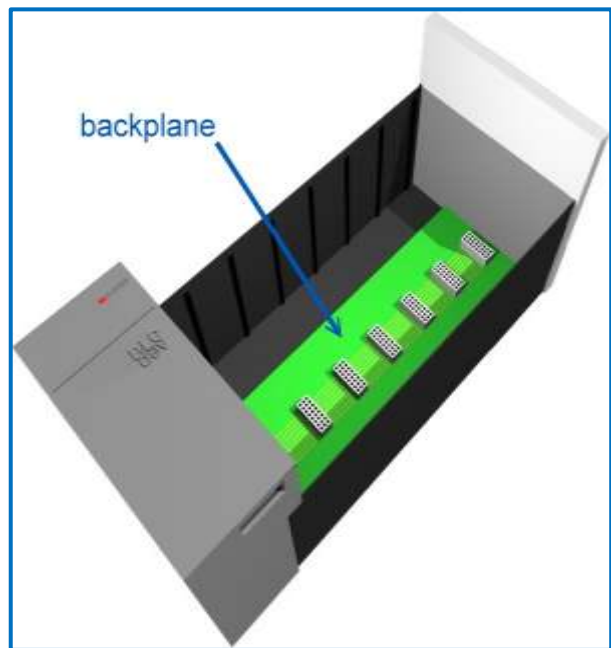
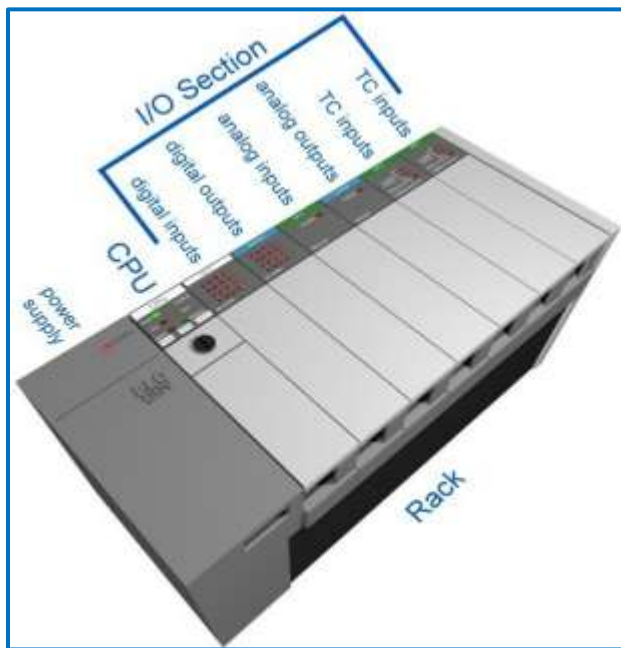
Automation Direct, Do-more Designer is the full-featured PLC programming software for the Do-more Automation Direct series Programmable Controllers (PLCs). Do-more Designer Software is offered as a free download from the Automationdirect.com. A CD-ROM version is also available for purchase. When Do-more Designer is installed on your PC, the following tool is also installed: the Do-more PLC Simulator (offline simulator of ladder program execution and PID control).

6. HOW THE PLCs WORK:

Common terms are typical for PCs, like Central Processing Unit (CPU), Memory, Software and Port Communications. Unlike a personal computer, the PLC is designed to survive in a rugged industrial atmosphere and to be very flexible in how it interfaces with Inputs and Outputs to the real world. The components that make a PLC work can be divided into three core areas.

- ✓ The Power Supply Unit (PSU) and rack;
- ✓ The Central Processing Unit (CPU);
- ✓ The Input/Output (I/O) section;

PLCs come in many shapes and sizes. They can be so small, as to fit in a shirt pocket, while more involved controls systems require large PLC racks. The smaller PLCs, also known as a “brick type”, are typically designed with fixed I/O points as low as 6 but also up to 256. Then, let’s take the more modular rack based systems. It’s called “modular” because the rack can accept many different types of I/O modules that simply slide into the rack and plug in. So, let’s start off by removing all our modules which leaves us with a naked PLC with only the power supply and the rack.



Power Supply Unit (PSU) and Rack: The rack is the component that holds everything together. Depending on the needs of the control system it can be ordered in different sizes to hold more modules. Like a human spine the rack has a backplane at the rear which allows the cards to communicate with the CPU.

The most popular power supply is the universal AC supply type, which works from 100V AC to 240V AC or 24V DC sources. PLCs work with the inputs and outputs generally using 24V DC, as well as, the 5V DC for the PLCs internal power and the universal AC input terminals. The 24V DC supplied by a PLC supply is generally low capacity and is only used to power the inputs for the system.

The Power Supply Unit (PSU) plugs into the rack, as well, and supplies a regulated DC power to other modules that plug into the rack. The rack is an important part of how PLCs work. The rack below is a Mitsubishi “A” series with eight I/O slots and a Power Supply Unit slot.



Like the spine in the human body, the rack has a backplane at the rear which gives the physical support needed to the PLC modules/cards. As well as allowing all the modules/cards to communicate with the CPU via the backplane data bus. This data bus is a very important part of how PLCs work, allowing the CPU direct access to each individual module.

The Central Processing Unit (CPU): The brain of the whole PLC is the CPU module. This module typically lives in the slot beside the power supply. Manufacturers offer different types of CPUs based on the complexity needed for the system. The CPU consists of a microprocessor, memory chip and other integrated circuits to control logic, monitoring and communications. The CPU has different operating modes. In *program-ming mode* it accepts the downloaded logic from a PC. The CPU is then placed in *run mode* so that it can execute the program and operate the process.

The CPU is made up of several components, a microprocessor, memory chip or chips, I/O interfacing and other integrated circuits (ICs) to control logic, monitoring and communications. The CPU of the PLC, itself has a few different operating modes:

- ✓ PLC Programming Mode;
- ✓ PLC Run Mode;
- ✓ PLC Stop Mode;
- ✓ PLC Reset Mode;

PLC Programming Mode: The CLP accepts the downloaded ladder logic program from a *laptop* or *PC* to install the controlling program. The CPU is placed in *run mode*, the way it can execute the program and operate the required process.

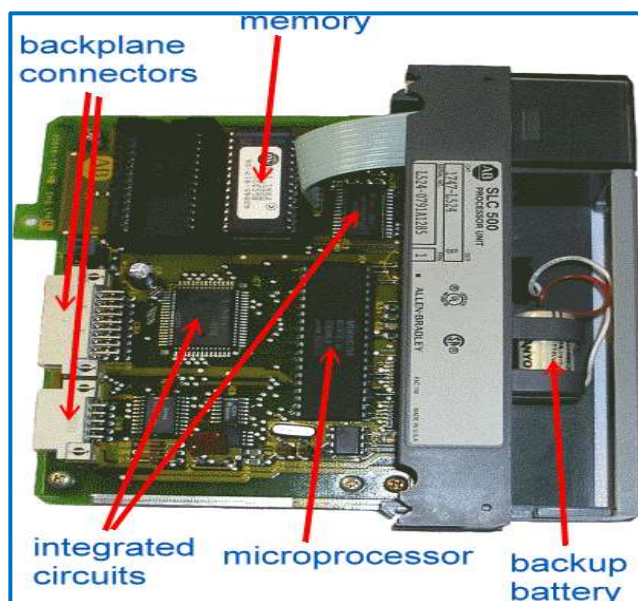
PLC Run Mode: Is when the PLC is in full operation, doing all self-checks controlled by the installed logic program, reading the inputs and setting the outputs accordingly, or even conversing with other units via the RS232, Profibus, Scada or CC-Link.

PLC Stop Mode: Some PLC programming functions require to be stopped. The Stop Mode function turns off all the outputs.

PLC Reset Mode: Resets the PLC from operating conditions back to switch on position. When this is done without resetting any data memory registers, this is called a “warm reset”. When the reset performed is full, for example, resetting all I/O and data registers is called as “cold reset”.



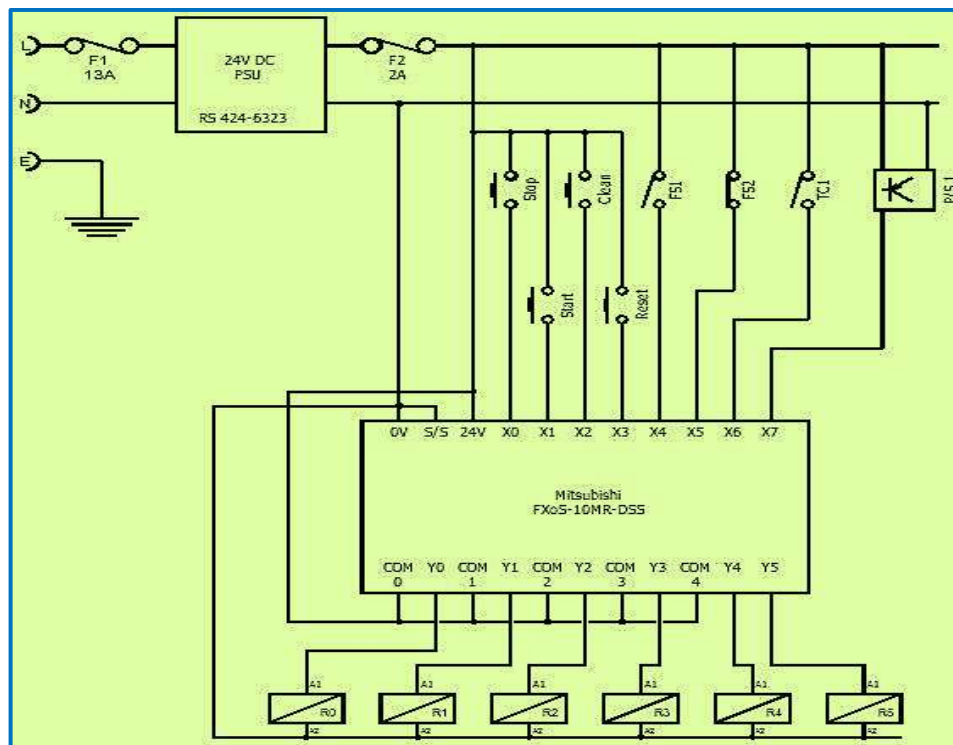
The picture above of a working panel, where can be seen the rack data cable connecting the expansion rack on top of the CPU rack, the white cable snaking along the top of trunking, and some of the input and output LED (Light Emitting Diode) actually on. The PLC in this panel (left) is a controller used by the London Underground Tube Railway, gives an indication of just how PLCs work very reliably day in and day out.



The picture above, at right, shows the PSU slotted in place (this rack does not have a slot for the CPU). The rack provides all the slotted in I/O modules, the PSU, and communications between the I/O modules installed and the PLC. The rack pictured below, is how it looks in a typical installation, also known as expansion rack, as provides an I/O expansion to the PLC CPU rack.

Input/Output (I/O) System: The I/O system provides the physical connection between the equipment and the PLC. Opening the doors on an I/O card reveals a terminal strip where the devices connect. There are many different kinds of I/O cards which serve to condition the type of input or output so the CPU can use it for its logic.

It's simply a matter of determining what inputs and outputs are needed, filling the rack with the appropriate cards and then addressing them correctly in the CPU's program. The I/O connectors on the PLC system provide the physical connection between the equipment and the PLC. Below, is a simple Mitsubishi PLC wiring diagram, where the PLC is a 24V DC supply type.



There are many different types of input and output cards which are picked depending on what type of control system is needed. How PLCs work can be customized by the type of input and output cards picked, the way the CPU can use for its logic control. It's simply a matter of defining what kind of inputs, outputs and Comms cards are needed for a PLC specification. Filling the rack with the appropriately picked cards and then addressing them correctly from within the CPU's program.

Inputs: Input devices can consist of digital or analog devices. *Digital input cards* handle discrete devices which give a signal that is either on or off such as a pushbutton, limit switch, sensors or selector switches and even Comms modules. *Analog input cards* convert a voltage or current (a signal that can be anywhere from 0 to 20mA) into a digitally equivalent number that can be understood by

the CPU. Examples of analog devices are pressure transducers, flow meters and thermocouples for temperature readings.

A PLC input device means anything that can influence the programs operation. These can consist of digital, analogue, switches, sensors, intelligent devices. Digital input cards handle discrete devices such as push-buttons, micro-switches, selector switches, photocells and proximity sensors which give a signal that has only two states. They are either on or off, what's called "bit device".

This is because the full scale of their signal range, (Full Scale Deflection or FSD for short), can be represented by one bit. There are eight bits in a byte, computers *talk* in bytes or multiples of bytes. To help to understand more on how PLCs work see 32-Bit and 64-Bit computer data bus.



PLC digital input cards that handle discrete devices are available with anywhere from 8 to 128 inputs on a single card that is slotted into the rack. However any more than 16 inputs on a card usually means having a breakout connector as it's just not possible to connect that many wires onto the top of an input card on the PLC with built in screw terminals.

A breakout connector is merely a means of *fanning* out the physical connection from the input or output card. This is achieved by using a multi-pole connector on top of the card which connects via a data cable to another card with all the screw terminals on it. It's this card that the discrete bit devices will connect to.

Comm Modules: Are bi-directional devices and part of how PLCs work, used as input for intelligent devices conveying all types of signals in bits and bytes, or values representing levels from previously digitized analogue devices, or lots of individual bit devices. Intelligent devices would include things like DC and AC drives for motors, other PLCs, HMI screens (Human Machine Interface), remote I/O stations as well as sophisticated sensors such as cameras and position arrays.

Analogue devices can also be input devices to PLCs, but need special cards to translate the infinitely variable signals (which could be voltage or current) into something the CPU can understand. This simply means converting say a 0V to 10V DC signal that you might get from load cell or speed control potentiometer into a value. For example when 10V DC is equal a value of 4000, then 2.5V DC would be equal to 1000 and 5V to 2000, then the FSD would be, 4000.

Outputs: Output devices can also consist of digital or analog types. A digital output card either turns a device on or off such as lights, LEDs, small motors, and relays. An analog output card will convert a digital number sent by the CPU to its real world voltage or current.



Typical outputs signals can range from 0-10 V DC or 4-20mA used to drive mass flow controllers, pressure regulators and position controls. Comms modules and intelligent devices are bi-directional devices and part of how PLCs work.

Bi-directional Devices: By definition, talk in both directions, simultaneously inputs and outputs at the same time. Looking at the *RS232 pinouts* can be seen a *Tx pin* and an *Rx pin*, standing for transmit and receive respectively. More basically, this translates into talk (Tx) and listen (Rx).

Input Converting Cards: Are modules also called A/D Converters, and means Analogue and Digital converters. D/A Converters (the process is reversed) are called *converter output cards*. An analogue output card converts a digital value or number held in a memory location by the CPU into a real world voltage or current. Typical outputs signals can range from 0-10V DC, -10V to +10V DC or 4-20mA and are used to control servo drives and positioning controls as well as pressure regulators and level control systems. This type of system is called “Closed Loop” control.

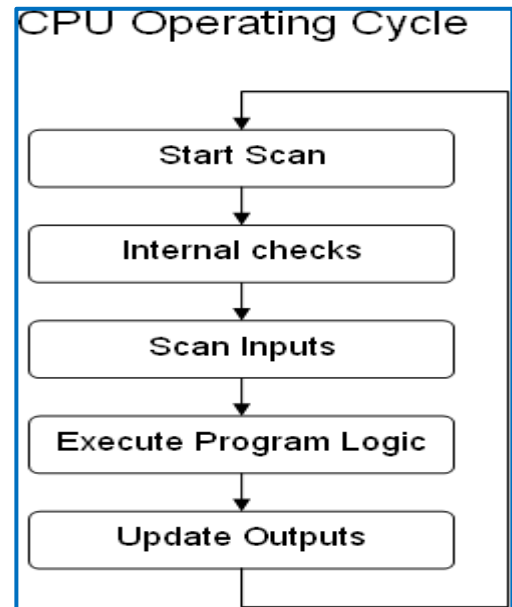
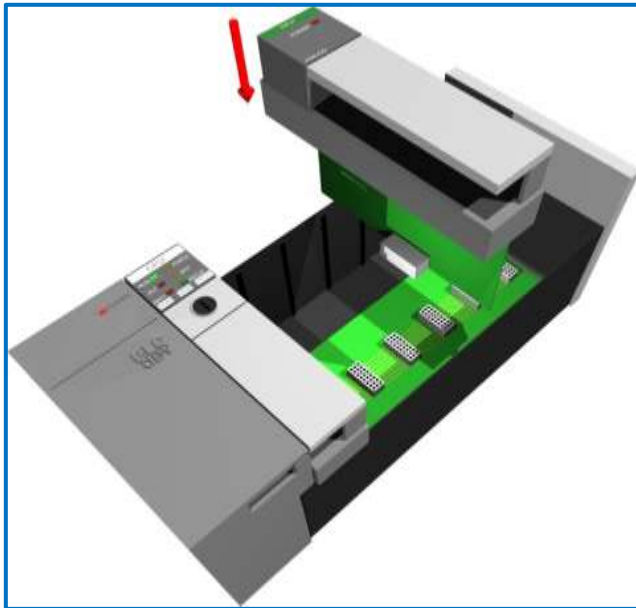
The digital output card is the complement to the digital input card and turns a (bit) device *on or off*, such as lights, LEDs, small motors, solenoids (electromagnets), and relays. Digital output cards are available with 8 up to 128 per card, but like the input cards, more than 16 would need a connector breakout card because of the physical space needed for the wire screw connections.

The Scan-time: The scan-time speed is very quick, and has to be as they're dealing with real time situations. The scan-time of a PLC happens in the order of 1/1000th's of a second. Sometimes it can even vary on a scan by scan basis depending on program loops being switched in and out by inputs and/or program equations.

Before it executes the user program, the PLC will scan the input modules, after that's complete it will execute the user program. But since a PLC is a dedicated controller, it will only process this one program, it will go through the whole program once per scan. The memory in the CPU stores the program in non-volatile RAM, which means it won't lose the program if the power is lost.

It uses volatile and non-volatile RAM for holding the status of the I/O and providing a means to store values. Some are kept at power down some are not. Then the PLC will update the outputs according to the condition of the inputs and the program logic instructions. Then the PLC repeats this process over and over again.

Since a PLC is a dedicated controller it will only process this one program over and over again. One cycle through the program is called a scan time and involves reading the inputs from the other modules, executing the logic based on these inputs and then updated the outputs accordingly.



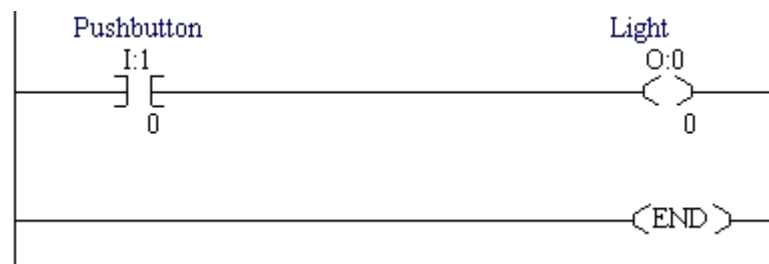
The scan time happens very quickly in the range of 1/1000th of a second, the memory in the CPU stores the program while also holding the status of the I/O and providing a means to store values. The CPU is then put into run mode so that it can start scanning the logic and controlling the outputs. PLC Operation: A PLC works by continually scanning a program, consisting of 3 important steps.

- **Check Input Status:** First, the PLC takes a look at each input to determine if it is on or off. In other words, check if the sensor connected to the first input. Then, the second input. Finally, it records all data into its memory to be used during the next step.
- **Execute Program:** Next the PLC executes your program in one instruction at a time. When a program says that the first input was on, then it turns on the first output. Since it already knows which inputs are on/off from the previous step, it will be able to decide whether the first output should be turned on, based on the state of the first input. It will store the execution results for use later during the next step.
- **Update Output Status:** Finally, the PLC updates the status of the outputs. It updates the outputs based on which inputs were on, during the first step and the results of executing a program during the second step. Based on this conception, now it turns on the first output because the first input was on, if such a program says to turn on the first output when this condition is true. After the third step the PLC goes back to step one and repeats all the steps continuously.

Programming a PLC: Today, a PC with especially dedicated software from the PLC manufacturer is used to program a PLC. The most widely used form of programming is called *ladder logic*. The *ladder logic* uses symbols, instead of words, to emulate the real world relay logic control, which is a relic from the PLC's history. These symbols are interconnected by lines to indicate the flow of current through relay like contacts and coils. Over the years the number of symbols has increased to provide a high level of functionality.

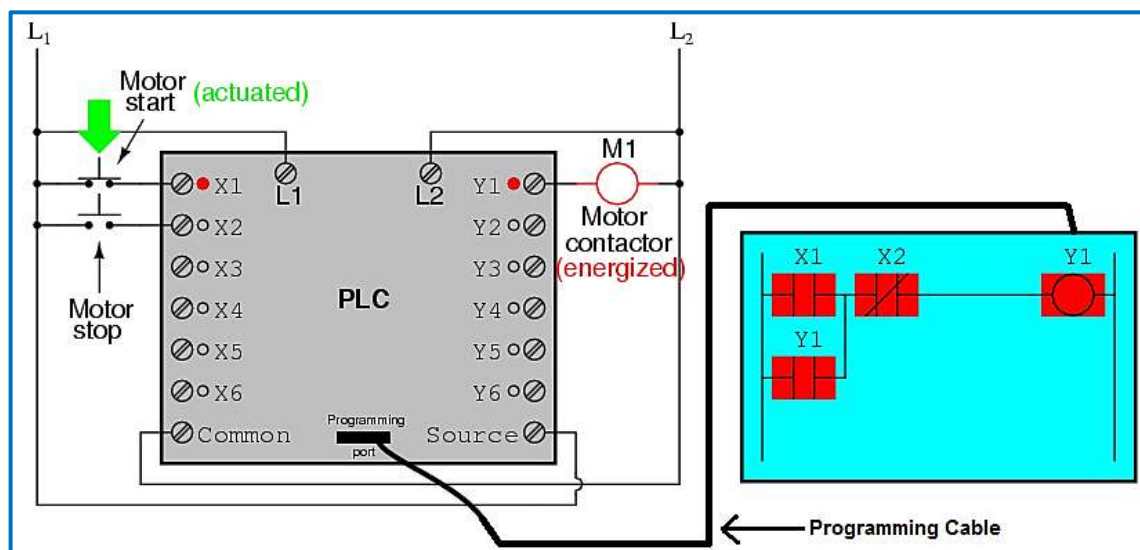
The completed program *looks like a ladder* but in actuality *it represents an electrical circuit*. The left and right rails indicate the positive and ground of a power supply. The *rungs* represent the wiring between the different components, which in the case of a PLC are all in the virtual world of the CPU. Understanding how basic the electrical circuits work, then anyone can understand ladder logic. In this simplest of examples a digital input (like a button connected to the first position on the card) when it is pressed turns on an output which energizes an indicator light.

The input symbols are representations of real world normally open and normally closed switches, the output symbols represents relays and lights connected by lines as though to show a wire. So the flow of current through the switch and relay or light, like relay contacts and coils can be viewed like a real circuit, like this:



The completed program **is downloaded** from the PC to the PLC with a special programming and a **connection cable**. It connects between a serial port on the PC to the programming connector on the front of the PLC/CPU, usually RS232 wiring. When *downloaded* the CPU is then put into **Run Mode**, so that it can start scanning the inputs, and controlling the outputs, as this example below:

When the "Stop" pushbutton is released, the input X2 de-energizes, returning to its normal, "closed" state. The motor, however, will not start again until the "Start" pushbutton is actuated, because the "seal-in" of Y1 is lost. The *fail-safe* design is very important in PLCs, as it is in electromechanical relay-controlled systems. In this motor circuit, for example, there is a problem: if the input wiring for X2 (the "Stop" switch) fails to open, there would be no way to stop the motor.

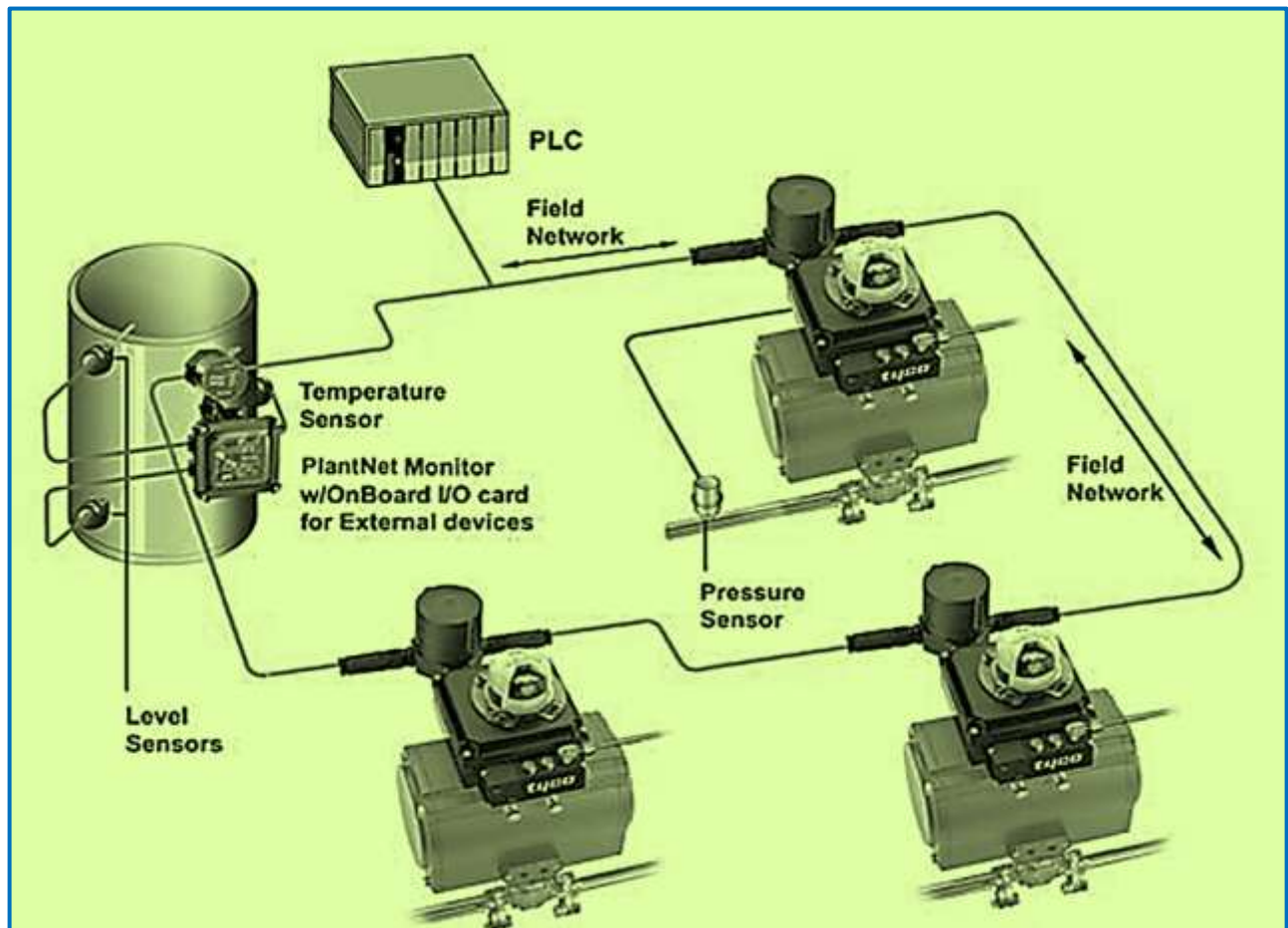


7. FIELDBUS CONTROL SYSTEMS:

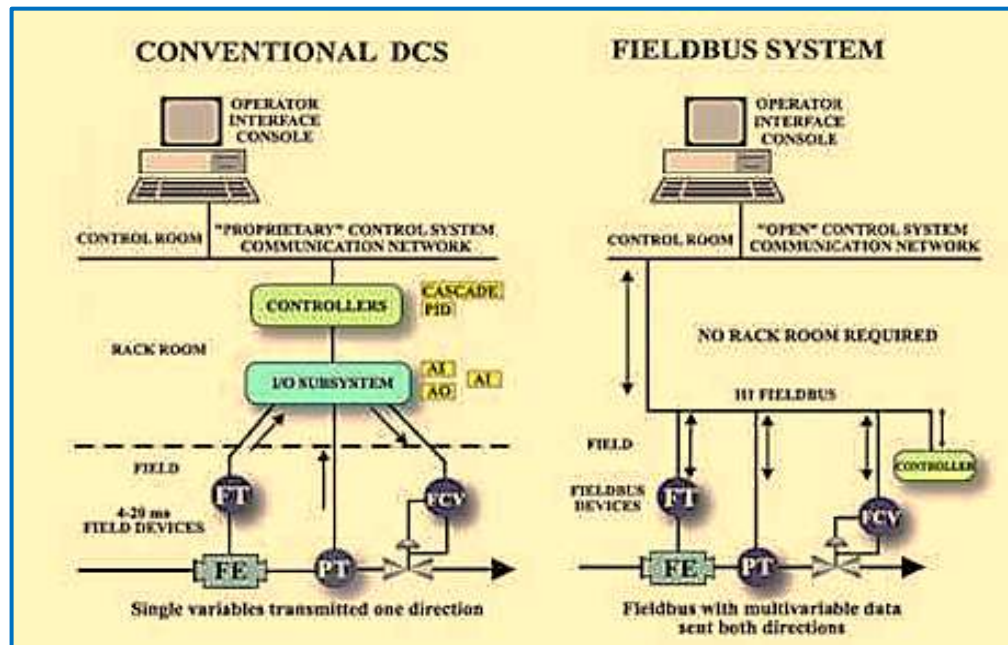
This chapter presents an introduction to the Fieldbus communication protocols and Distributed Control Systems (DCS), early used to control manufacturing processes continuous or batch-oriented, such as oil refining, petrochemicals, central station power generation, fertilizers, pharmaceuticals, food and beverage manufacturing, cement production, steelmaking, papermaking and many other consolidated processes.

Fieldbus: Is an industrial network system used in process control and industrial automation, as a way to connect instruments in a manufacturing plant. Fieldbus is the name of a family of industrial computer network protocols used for a real-time distributed control, standardized today, as IEC 61158. **Field** is an abstraction of equipment plant levels. **Bus** is a well-known word in computer science as a set of common line that electrically (or even optically) connects various units (circuits) in order to transfer the data among them.

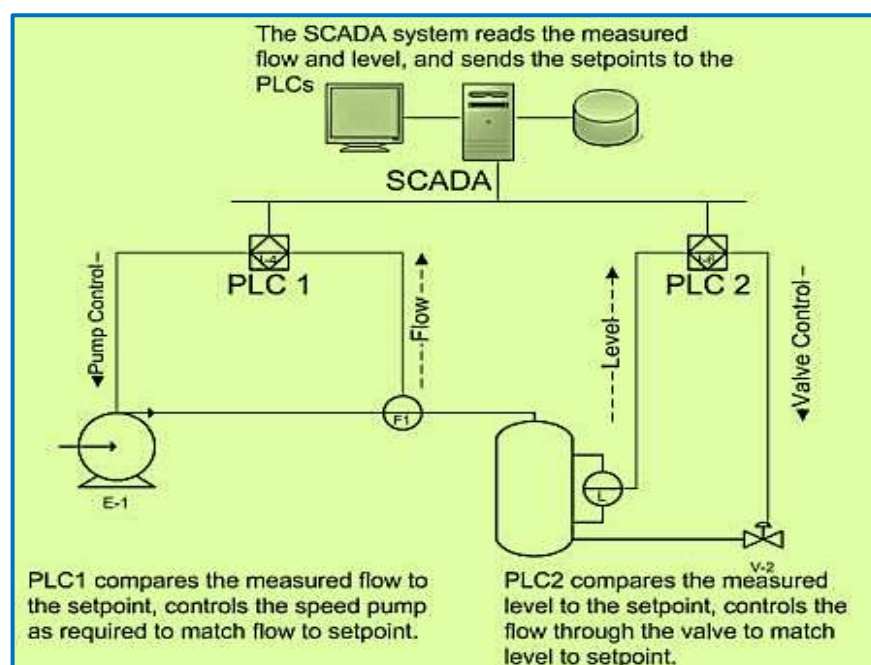
Fieldbus can also be defined as an industrial network system, of a family of industrial *computer network protocols* for real-time distributed control, as a way to connect instruments in a manufacturing plant. A complex automated industrial system, such as a manufacturing assembly line, usually needs a distributed control system or an organized hierarchy of pneumatic, hydraulic and electro-electronic instrumentations and controller systems, to operate faithfully.



Difference between DCS and Fieldbus: DCS is a control platform while Foundation Fieldbus is a communication protocol. A DCS is a centralized control for inputs distributed geographically, and the operator station is normally intimately connected with its I/O (Fieldbus, local wiring, networks, etc.). Foundation Fieldbus is a digital signal that replaces the traditional 4-20 mA signal and deals with communication between instruments, actuators, and controllers, whether distributed or centralized.



SCADA (Supervisory Control & Data Acquisition): More similar to DCS (distributed geographically), is a system operating with coded signals over communication channels to provide control of remote equipment (using typically one communication channel per remote station). However, DCS and SCADA have different functions.



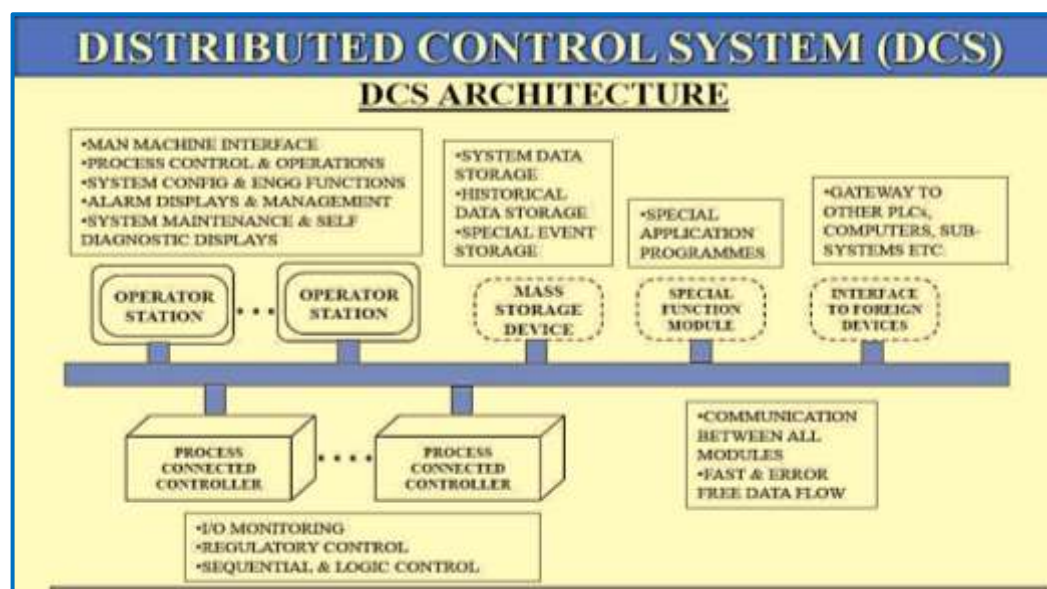
The SCADA systems were among the first to enjoy the benefits of digital technology in the 1960's. If the flow of information is one-way (simplex, from measurement devices to human operators), the system is more properly referred to as a telemetry system rather than a SCADA system. SCADA implies in a two-way (duplex) information flow, where human operators not only monitor process data, but also issue commands back to the remote terminal units to effect change.

The term SCADA usually refers to centralized systems which monitor and control entire sites, or complexes of systems spread out over large areas (anything from an industrial plant to a nation). Most control actions are performed automatically by RTUs or by PLCs. Host control functions are usually restricted to basic overriding or supervisory level intervention.

Process data in a SCADA system is sensed by various measurement devices (transmitters), converted to digital form by a device called an RTU (Remote Terminal Unit), and communicated to one or more MTUs (Master Terminal Units) at a central location where human operators may monitor the data and make command decisions. SCADA systems are very versatile and most can communicate with almost anything, in one way or another.

For example, a PLC may control the flow of cooling water through part of an industrial process, but the SCADA system may allow operators to *change the set points* for the flow, and enable alarm conditions, such as loss of flow and high temperature, to be displayed and recorded. The feedback control loop passes through the RTU or PLC, while the SCADA system monitors the overall performance of the loop.

Distributed Control Systems: The DCSs use custom designed processors as controllers, and both proprietary interconnections and communications protocols for communication. The input modules receive information from input instruments in the process (or field) and the output modules transmit instructions to the output instruments in the field. Computer buses or electrical buses connect the processors to distributed controllers and the Human–Machine Interfaces (HMI) or control consoles, through multiplexer or demultiplexers.



DCSs may employ one or more workstations; can be configured by an off-line personal computer, and local communication can be handled by a control network, by over twisted pair, coaxial, or fiber optic cables. The input/output devices (I/O) can be integral with the controller or located remotely via a field network, and a server may be included in the system for extra data collection.

8. DCS AND FIELBUS DEVELOPMENT:

The DCS was introduced in 1975. Both Honeywell and the Japanese electrical engineering company Yokogawa introduced their own independently produced DCSs at the same time, with the TDC 2000 and CENTUM systems, respectively. The US-based Bristol also introduced their UCS 3000 universal controller in 1975.

In 1978 Metso (known as Valmet in 1978) introduced its own DCS system called Damatic. In 1980, Bailey (now part of ABB) introduced the NETWORK 90 system, Fisher Controls (now part of Emerson Electric) introduced the PROVoX system, Fischer & Porter Company (now also part of ABB) introduced DCI-4000 (DCI stands for Distributed Control Instrumentation).

Era of the 1980s: In the 1980s, users began to look at DCSs as more than just basic process control. A very early example of a Direct Digital Control DCS was completed by the Australian business Midac in 1981-82 using R-Tec Australian designed hardware. The first attempts to increase the openness of DCSs resulted in the adoption of the operating system of the day: *UNIX*. *UNIX* and its companion networking technology TCP-IP were developed by the US Department of Defense, precisely the issue the process industries were looking to resolve.

As a result, suppliers also began to adopt Ethernet-based networks with their own proprietary protocol layers. The full TCP/IP standard was not implemented, but the use of Ethernet made it possible to implement the first global management data access technology, and the first PLCs integrated into the DCS infrastructure. The first DCS supplier to adopt *UNIX* and Ethernet networking technologies was Foxboro, who introduced the I/A Series system in 1987.

Era of the 1990s: The introduction of Microsoft at the desktop, resulted in the development of technologies such as OLE for Process Control (OPC), which became an industry connectivity standard. The Internet technology also began to make its mark in the DCS world, with HMI Internet connectivity. The first fieldbus installations occurred in the 1990s, and this year is also known as the "Fieldbus Wars", where rival organizations competed to define what should become the IEC fieldbus standard for digital communication with field instrumentation, instead of 4-20 mA analog communications.

Era of the 2000s: The technology began to develop with the market consolidating the Ethernet I/P, Foundation Fieldbus and Profibus PA for process automation applications. Some suppliers built new systems to maximize functionality with fieldbus, such as the Rockwell PlantPAX System, Honeywell with Experion & Plantscape SCADA, ABB with System 800xA, Emerson Process Management with the DeltaV Control System, Siemens with the SPPA-T3000 or Simatic PCS 7, Azbil Corporation with the Harmonas-DEO system, and with Metso DNA system. The Fieldbus technology has been used to integrate machine, drives, quality and condition monitoring applications to one DCS.

Fieldbus Development: ARCNET was conceived in 1975 for office connectivity and later uses in industry; the majority of fieldbus standards *were developed in the 1980s* and became fully established during the mid-1990s. Attached Resource Computer Network (abbreviated ARCNET or ARCnet) is a communications protocol for local area networks. ARCNET was the first available networking system for microcomputers and became popular in the 1980s for office automation tasks.

ARCNET: Was developed originally to connect groups of the *Datapoint 2200 terminals* to talk to a shared 8" floppy disk system, by the engineer John Murphy at Datapoint Corporation in 1976, and announced in 1977. ARCNET remained proprietary until the early-to-mid 1980s. In 1979, the Open Systems Interconnection Reference Model (OSI model) was published. Then, in 1980, Digital, Intel and Xerox (the DIX consortium) published an open standard for Ethernet that was soon adopted as the basis of standardization by the IEEE and the ISO.

IBM responded by proposing the *Token Ring system* as an alternative to Ethernet, but kept a tight control over standardization, that competitors were wary of using it. ARCNET was less expensive, and more flexible, and by late 1980s it had a market share almost equal to Ethernet. When Ethernet moved from *co-axial cable* to *twisted pair* and "interconnected stars" cabling topology *based on active hubs*, became more attractive. Easier cabling, combined with greater speed of Ethernet (10 Mbit/s, as compared with 2.5 Mbit/s of ARCnet) helped to increase the Ethernet demand.

In response to greater bandwidth needs, and the challenge of Ethernet, a new standard called ARCnet Plus was developed by Datapoint, introduced in 1992. ARCnet Plus ran at 20 Mbit/s, and was backward compatible with original ARCnet equipment. However, by the time, Ethernet had captured the majority of the network market, and there was little incentive for users to move back to ARCnet. As a result, very few ARCnet Plus products were ever produced. ARCNET was eventually standardized as ANSI ARCNET 878.1, and the name changed from ARCnet to ARCNET.

Controller Area Network (CAN): Started originally in 1983 at Robert Bosch, GmbH. The protocol was officially released in 1986 at the Society of Automotive Engineers (SAE) in Detroit, Michigan. The first CAN controller chips, produced by Intel and Philips, came on the market in 1987. CAN bus is one of five protocols used in the on-board diagnostics (OBD)-II vehicle diagnostics standard, which became mandatory for all cars and light trucks sold in the United States since 1996, and for all petrol vehicles sold in the European Union since 2001, and all diesel vehicles since 2004.

Robert Bosch published the CAN 2.0 specification in 1991. In 2012 Bosch has specified the improved CAN data link layer protocol, called CAN FD, which will extend the ISO 11898-1. The concept of CAN was that every device can be connected by a *single set of wires*, and every device that is connected *can freely exchange data* with any other device. CAN soon evolved into the factory automation marketplace (as many others).

CAN bus is a vehicle bus standard designed to allow microcontrollers and devices to communicate with each other within a vehicle without a host computer. CAN bus is a message-based protocol, designed specifically for automotive applications but now also used in other areas such as aerospace, maritime, industrial automation and medical equipment.

Fieldbus Standardization: In 1999, the IEC SC65C/WG6 standards committee met to resolve difference in the draft IEC fieldbus standard, resulting the initial form of the IEC 61158 standard with eight different protocol sets called "Types" as follows:

- Type 1 - FOUNDATION Fieldbus H1 & H2 (Low-Speed Ethernet);
- Type 2 – ControlNet;
- Type 3 – PROFIBUS – DP, PA;
- Type 4 - P-Net;
- Type 5 - FOUNDATION Fieldbus HSE (High-Speed Ethernet);
- Type 6 - SwiftNet (a protocol developed for Boeing, since withdrawn);
- Type 7 – WorldFIP;
- Type 8 – Interbus.

The fieldbus technology needed to be implemented differently in different applications, since automotive fieldbus is functionally different from process plant control. The final edition of IEC standard IEC 61158 allows 8 technologies. IEC 61158 consists of the following parts, under the general title Digital data communications for measurement and control fieldbuses for use in industrial control systems. Below is the basic hierarchic layer of the automation protocols.

- Part 1: Overview and guidance for the IEC 61158 series;
- Part 2: Physical Layer specification and service definition;
- Part 3: Data Link Service definition;
- Part 4: Data Link Protocol specification;
- Part 5: Application Layer Service definition;
- Part 6: Application Layer Protocol specification.

Ethernet and Fieldbus: Recently a number of Ethernet-based industrial communication systems have been established, most of them with extensions for real-time communication. These have the potential to replace the traditional fieldbuses in the long term. Here is a partial list of the new Ethernet-based industrial communication systems:

- FDX, EtherCAT, EtherNet/IP, Ethernet Powerlink, FOUNDATION HSE, BACnet, PROFINET IO, PROFINET IRT, SafetyNET p, SERCOS III, TTEthernet, VARAN, RAPIEnet.

Ethernet and EtherNet: There are dozens of protocols that run on top of Ethernet. Some of them use TCP/IP and some use UDP/IP, but it's still different. "UDP" stands for "User Datagram Protocol". The "IP" in "EtherNet/IP" stands for "Industrial Protocol". The "IP" in "TCP/IP" stands for "Internet Protocol". Note that "EtherNet/IP" is a specific thing, not "Ethernet TCP/IP". The EtherNet/IP (note the capital N) is actually a set of protocols; some of them use TCP/IP and some use UDP/IP, applying multi-cast or uni-cast, peer-to-peer and scanner-node functionality.

Fieldbus Evolution: From early applications of digital computing in the 1960's to the first distributed control systems (DCS) in the 1970's to the "smart" transmitter revolution of the 1980's, digital technology has expanded on the capabilities and information-sharing ability of measuring and con-

trol instruments. The advent of digital electronic circuitry has brought a steady stream of technological progress to industrial instrumentation.

IEC 61158 (1999- 2008) – Main Fieldbus Standards:

- (Type 1) - FOUNDATION Fieldbus H1 & H2 (Low-Speed Ethernet);
- (Type 2) - ControlNet;
- (Type 3) - Profibus- DP, PA;
- (Type 4) - P-Net;
- (Type 5) - FOUNDATION Fieldbus HSE (High-Speed Ethernet);
- (Type 6) – Swiftnet;
- (Type 7) – WorldFIP;
- (Type 8) - Interbus-S.

Modbus History:

- Modicon (Schneider Electric) – 1979;
- Modbus IDA – 2004.

PROFIBUS History:

- Fieldbus – 1987: German companies and Institutes.

FOUNDATION Fieldbus History:

- ISA SP50 – 1988;
- ANSI/ISA – S50.02 – 1994;
- ISP Foundation – 1993: Merged with WorldFIP North America, 1996;
- Foundation Fieldbus – 1996.

HART History:

- Rosemount (Emerson) – 1986;
- Open Protocol -1990;
- HART User Group – 1990;
- HART Communication Foundation – 1993;
- IEC 61158/IEC 61784 – 2007.

IEC 61158 (Updated 2008) – Fieldbus Development:

- 1 – FOUNDATION Fieldbus: H1, H2, HSE;
- 2 – CIP: ControlNet, EtherNet/IP, DeviceNet;
- 3 – Profibus: DP, PA, Profinet (4 types);
- 4 – P-NET: P-NET RS-485, P-NET RS-232, P-NET on IP;

- 5 – WorldFIP: 3 types;
- 6 – InterBus: Several types;
- 8 – CC-Link: 3 types;
- 9 – HART;
- 10 – Vnet/IP;
- 11 – Tcnet: 2 types;
- 12 – EtherCat, Ethernet Powerlink;
- 13 – EPA;
- 14 – MODBUS-RTPS: MODBUS-RTPS; MODBUS TCP;
- 15 – SERCOS: Types I, II, and III.

Fieldbuses in the Process and Manufacturing Industries:

- IEC Standard buses: ControlNet, FF H1, FF HSE, Profibus; Interbus-S, P-net, CAN, CIM;
- Standard buses: HART, MODBUS (now IEC too);
- Others: AS-i, LON.

Fieldbuses with Focus on Process Technologies:

- FF H1, FF HSE;
- Profibus PA (Process Automation);
- ProfiNet;
- Profibus DP (Decentralized Peripherals);
- MODBUS;
- HART.

Connectivity: Previously, computers were connected using the RS-232 (serial connections), by which only two devices could communicate, equivalent of the currently used 4-20 mA, which requires that each device had its own communication point at the controller level. For example, it is possible for PCs to communicate using a particular protocol, like Kermit, via Com ports that are RS-232 or RS-485.

Kermit: Is an open protocol and communication software that provides a consistent approach to file transfer, and set character conversions across many different computer hardware and platforms. A version of Kermit is supplied with Linux is the C-Kermit, which runs from a command line. However, RS-232, RS-422, and RS-485 are not technically protocols, but line interface standards, described by characteristics such as connectors, signaling (voltage, timing, single-ended, differential, etc.).

Protocol: Is the structure of the bits or octets (unit of digital information in computing and telecommunications that consists of eight bits), being transmitted, as transparency techniques, forming of the messages, frames, etc. Protocols are the codes and a rule the transmitter uses to compose the message and the receiver uses to decompose the message and extract the information. Many protocols have been developed for each of the line interface standards. Consequently, it is usually easy to connect interfaces together correctly, and still have each end not understanding the other.

9. FIELDBUS MODELS AND NETWORK STRUCTURE:

Fieldbus works on a network structure which typically allows daisy-chain, star, ring, branch, and tree network topologies. Though each technology shares the generic name of fieldbus the various fieldbus are not readily interchangeable. The differences between them are so profound that they cannot be easily connected to each other. To understand the differences among fieldbus standards, it is necessary to understand how fieldbus networks are designed.

For each technology the physical medium and the physical layer standards fully describe, in detail, the implementation of bit timing, synchronization, encoding/decoding, band rate, bus length and the physical connection of the transceiver to the communication wires. The application layer standard, defines how the data communication layers are interfaced according to the application. It describes message specifications, network management, and response from the application of services.

Basically, the fieldbus is equivalent to the LAN-type connections, which require only one communication point at the controller level and allow multiple (hundreds) of analog and digital points to be connected at the same time. Devices that communicate through fieldbus require multiple points typically provided by the same device. Some fieldbus devices support control schemes such as PID control on the device side instead of forcing the controller to do the processing.

OSI model: The **Open Systems Interconnection Model (OSI)** is a conceptual model that characterizes and standardizes the internal functions of a communication system by partitioning it into abstraction layers. The physical media of Fieldbus standards are determined by the cabling types, and mainly by layers *one, two and seven* of the reference model.

The model is a product of the Open Systems Interconnection project at the International Organization for Standardization (ISO), maintained by the identification ISO/IEC 7498-1. The OSI standards documents are available from the ITU-T as the X.200-series of recommendations that describes seven layers, labeled 1 to 7. Note that the Layer 1 is the lower layer in this model.

OSI Model			
	Data unit	Layer	Function
Host layers	Data	7. Application	Network process to application
		6. Presentation	Data representation, encryption and decryption, convert machine dependent data to machine independent data
		5. Session	Interhost communication, managing sessions between applications
	Segments	4. Transport	Reliable delivery of packets between points on a network.
Media layers	Packet/Datagram	3. Network	Addressing, routing and (not necessarily reliable) delivery of datagrams between points on a network.
	Bit/Frame	2. Data link	A reliable direct point-to-point data connection.
	Bit	1. Physical	A (not necessarily reliable) direct point-to-point data connection.

Comparison with TCP/IP Model: The TCP/IP model for Internet protocols are deliberately not as rigidly designed into strict layers as in the OSI model. However, TCP/IP recognizes four broad layers of functionality which are derived from operating scope of the contained protocols: the scope of the software application; the end-to-end transport connection; the internetworking range; and the scope of the direct links to other nodes on the local network. These layers are compared with the OSI layering scheme, in the following way:

- The Internet application layer includes the OSI application layer, presentation layer, and most of the session layer.
- Its end-to-end transport layer includes the graceful close function of the OSI session layer as well as the OSI transport layer.
- The internetworking layer (Internet layer) is a subset of the OSI network layer (see above).
- The link layer includes the OSI data link and physical layers, as well as parts of OSI's network layer.

Bus Technology Family: In most cases *Fieldbus and Ethernet* networks use the same application layer, that is, the same data types, object structure, and functions. Fieldbus and Ethernet complement each other, thus, all major bus technology organizations have both Fieldbus and Ethernet. To be clear, the table below shows only a small fieldbus selection of what is available.

Organization	Fieldbus Level	Remote I/O Level	Ethernet Level
ODVA	DeviceNet	ControlNet	EtherNet/IP
Modbus-IDA	-	Modbus/RTU	Modbus/TCP
PNO	Profibus-PA	Profibus-DP	PROFINET
Fieldbus Foundation	FF H1	-	FF HSE
	-	SERCOS	SERCOS-III

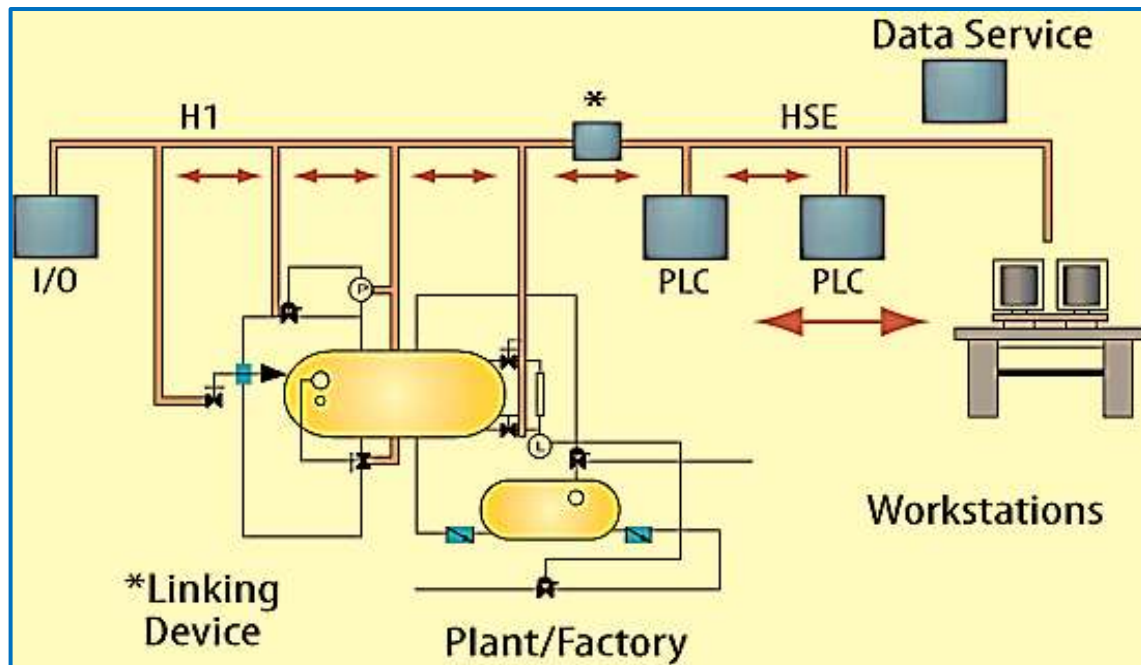
There are about 20 industrial network systems that can be called industrial fieldbus and ethernet protocols, so an exhaustive description is not our scope. The table below lists a number of popular (serial) fieldbus protocols and their corresponding Ethernet-based industrial protocols, and described only the most known and broadly recognized in international standards:

Popular Industrial Fieldbus and Industrial Ethernet Protocols:	
Serial Protocols	Ethernet-Based Protocols
DeviceNet/ControlNet	Ethernet/IP
PROFIBUS	PROFINET
Modbus-RTU	Modbus-TCP
SERCOS I/II	SERCOS III
Fieldbus Foundation FF H1	FF HSE
LonWorks	Lon over Ethernet
CANopen	EtherCAT/ Ethernet Powerlink

Fieldbus Foundation: Is actually the organization responsible for the definition and application rules of the FOUNDATION fieldbus specification, which certifies products to be compliant with this standard. In 1990 a number of partners from Japan and America merged the *FIP to the WorldFIP* standardization group (later became the Fieldbus Foundation group). The name "FIP fieldbus" was originally given as an abbreviation of the French "Flux d'Information vers le Processus" and later known with the English name "Factory Instrumentation Protocol" (some references also use the hybrid "Flux Information Protocol").

Along with the competing German PROFIBUS the fieldbuses were submitted for European standardization by CENELEC in 1996. These standards were included to the international IEC 61158 and IEC 61784 standards by 1999 where FIP is listed as the Communication Profile Family 5. However, the WorldFIP homepage has seen no press release since 2002 (since US based Fieldbus Foundation have taken the lead in ongoing development with H1 fieldbus for process automation).

FOUNDATION Fieldbus: This system is an all-digital, serial, two-way communications, commonly used as a *Local Area Network (LAN)* for automation devices and process automation, which integrates low-speed sensors and actuators with high-speed controllers and servers in a single system, using a distributed architecture where the control is in the devices themselves.



FOUNDATION fieldbus is made up of two networks: *H1 (lower-speed)* and *HSE (high-speed)*. The "H1" and the "HSE" are based on UDP/IP (Ethernet) at the host-level. The function block control is primarily decentralized to transmitters and positioners, whose primary function is to pass data from the H1 level to the HSE level. Host computers also on the HSE network act as operator interfaces.

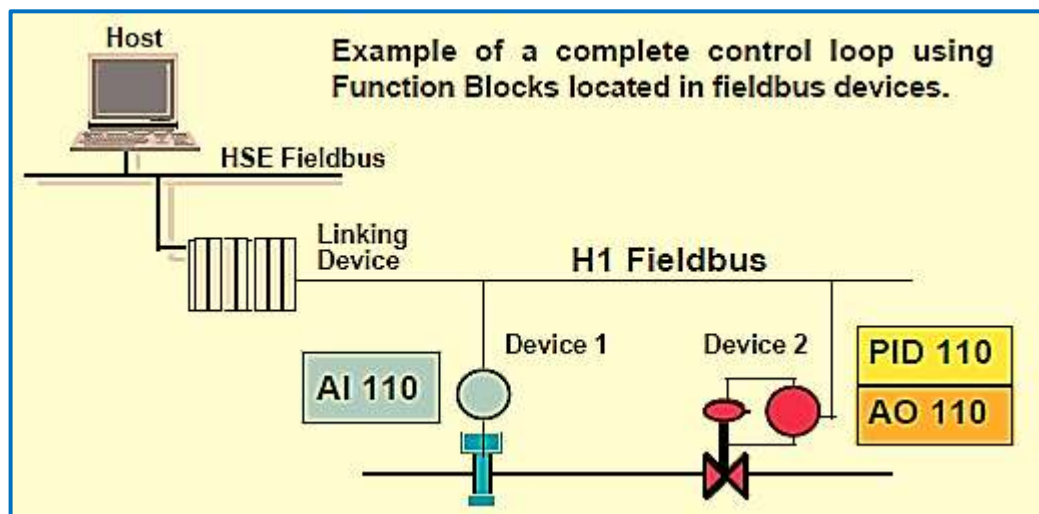
- ✓ **The H1 (Low-Speed Ethernet) Communication:** Runs at 31.25 kbit/s, up to 1900 meters, over twisted pair cables or fiber optic links, provide an open and interoperable solution for most field instruments and applications including intrinsically safe networks.

- ✓ **The HSE (High-Speed Ethernet) Communication:** Runs at 100 Mbit/s, up to 100 meters, the high speed connection between various H1 segments and host systems, including PLCs with a “backbone” network.

The FOUNDATION fieldbus H1 is a conventional data communication protocol designed for short messages up to 255 bytes, and secured by a 24-bit cyclic redundancy check. The HSE (High-Speed Ethernet) provides 100 Mbp/s between PLCs, servers and workstations, and can also be connected to H1 networks via linking devices that multiplexes H1 segments together. Introduced in 1996, HSE provides services at layers 1, 2, 3, 4 (Ethernet and TCP/IP) and layer 7 of the OSI model.

FOUNDATION fieldbus was designed to connect "smart" field instruments with each other, and can be required to host a DCS, PLC, or HMI or may be a PV (measurement), alarm status, secondary variable, etc. at a specific synchronous interval. FOUNDATION fieldbus instruments are capable of processing raw measurement signals by linearizing, filtering, and converting to engineering units.

The function block may be configured to execute in these field instruments or to subscribe to external values in the same segment, and perform many standard computations including PID closed loop control. The basic instrument communicates in FOUNDATION fieldbus H1 protocol, as previously described.

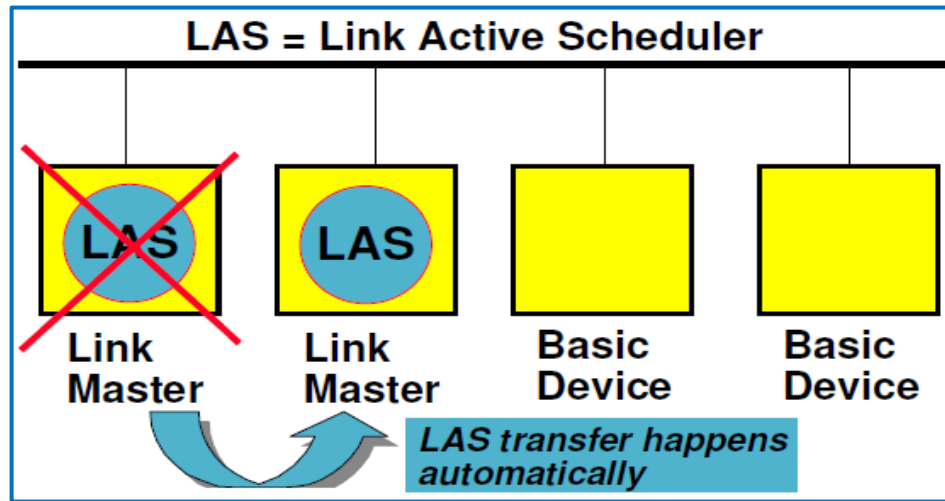


All messages are acknowledged. Critical loop timing is controlled by the LAS (Link Active Scheduler), a function that must exist for each control loop, and may be resident in any FOUNDATION fieldbus H1 device. The LAS performs link arbitration and assures that measurements are delivered just-in-time to the PID control function block.

FOUNDATION fieldbus (FF) is primarily used in new plants built from scratch. Existing plants that use 4-20 mA tend to be upgraded to AI and AO card with HART since HART-enabled devices and the hardwiring are already in place. There have been cases where very old plants using pneumatics are upgraded to FF. A new plant can be built with digital FF networking instead of hardwiring. Using Fieldbus enables a completely digital control loop, from sensor to actuator.

FOUNDATION fieldbus is used for temperature transmitters (or RTD sensors) all the time, necessary in any process plant. One of the beauties of Fieldbus is that it is not limited to one signal per pair of wires, so you have 8-channel Fieldbus temperature transmitters like the Rosemount 848T which takes the place of 8 conventional transmitters, 8 pairs of wire, and an 8 channel AI card.

LAS (Link Active Scheduler): Is a link master device that transmits schedule times for all data buffers in all basic devices that need to be cyclically transmitted. When it is time for a device to send a buffer, the LAS issues a Compel Data (CD) message to the device. Upon receipt of the CD, the device broadcasts or “publishes” the data in the buffer of all devices on the fieldbus.



Market: The market is dominated by FOUNDATION fieldbus and PROFIBUS PA. Both technologies use the same physical layer (2-wire manchester-encoded current modulation at 31.25 kHz) but are not interchangeable. Applications controlled and monitored by PLCs tend towards PROFIBUS, and applications controlled and monitored by a DCS (digital/distributed control system) tend towards FOUNDATION fieldbus.

FOUNDATION Fieldbus and PROFIBUS-PA are two very widely used fieldbuses, neither of which is superimposed on a DC signal. Both use Manchester Encoding to form essentially an AC baseband signal with a trapezoidal wave form to reject noise. The protocol for each is quite different, but they are designed to carry the same Process Variable value in digital format, along with other data that is available in the smart field transmitter.

PROFIBUS: Is the abbreviation for Process Field Bus and is the standard for fieldbus communication. In 1987, 21 institutions in Germany joined forces to create a new serial Fieldbus system. The group, named as Central Association for the Electrical Industry (ZVEI), completed its goal with the creation of PROFIBUS FMS (Fieldbus Message Specification). In 1993, the group introduced a new standard, PROFIBUS DP (Decentralized Periphery). This new version featured more simplicity, including easier configuration and faster messaging.

Profibus is divided into three variations:

- **FMS (Fieldbus Message Specification):** Was the initial version of PROFIBUS, but the FMS technology was not as flexible as needed. This protocol was not appropriate for less complex messages or communication on a wider, more complicated network. PROFIBUS FMS is still in use today, though the vast majority of users find newer solutions to be more appropriate.
- **DP (Decentralized Peripheral) Version:** Replaced the first complex communication protocol version FMS (Fieldbus Message Specification) in 1993. *PROFIBUS DP* is the high speed solution, optimized especially for communication between automation systems and decentralized devices. It can operate at data rates of up to 12 Mbit/s over twisted pair cables or fiber optic links. The usual operations are: Data rate: 93.75 Kbp/s and less - 1200 meters per segment; 500 Kbp/s - 400 meters; 1.5 Mbp/s - 200 meters; 12 Mbp/s - 100 meters.
- **PA (Process Automation) Protocol Version:** Is the lesser-used and is commonly provided for monitoring measurement equipment. *PROFIBUS PA* is used to monitor measuring equipment via a process control system. The disadvantage of this protocol is its slow data rate of only 31.25 kbit/s, but the weak current flow makes it intrinsically safe and ideal for use in explosion-hazardous areas. Usual operation is: Data rate: 31.25 Kbp/s – 1900 meters:

PROFIBUS is defined by the OSI layers one and two. However, the data link layer is only completed through a Fieldbus Data Link (FDL), to combine two common schemes, master-slave methodology and token passing. In the master-slave network, masters usually controllers, send requests to slaves, sensors and actuators. “Token” signal is passed between nodes. The token node communication concept is like a speaking conch; only the person with the conch is allowed to talk.

The PROFIBUS token passing procedure uses a simplified version of the Timed-token protocol. The medium access functions which are implemented at the *layer-2 of the OSI reference model*, called as *Fieldbus Data Link (FDL)*. In addition, to controlling the bus access and the token cycle time, the FDL is also responsible for the provision of data transmission services for the application layer.

The primary vendor of this industrial network is Siemens and today is managed by the organization PROFIBUS and PROFINET International (PI). PROFIBUS supports four data transmission services: *Send Data with No-Acknowledge (SDN)*; *Send Data with Acknowledge (SDA)*; *Request Data with Reply (RDR)* and *Send and Request Data (SRD)*. The SDN is an unacknowledged service used for broadcasts from a master station to all other stations on the bus, an important characteristic with response to an acknowledgement.

PROFIBUS enables someone to connect any bus communicator to a device that is a stand-alone protocol converter. It doesn't matter if this device runs CAN, RS232/422/485, Modbus RTU, DF1 or any other proprietary protocol. It is easy to configure the connection, in a configuration manager software, which means that no programming is needed.

PROFIBUS DP runs at speeds between 9.6kbit/s and 12Mbit/s. A particular speed can be chosen for a network to give enough time for communication with all the devices present in the network. Balanced transmission RS 485 only allows 32 devices to be connected at once. However, more devices can be connected, and the network can be expanded using hubs or repeaters.

PROFIBUS PA is slower and runs at fixed speed of 31.2kbit/s. This protocol is commonly used when there is risk of explosion or for systems that intrinsically need safer equipment. However, the message formats for PROFIBUS PA are identical to PROFIBUS DP.

The PROFIBUS DP master can also communicate with the Siemens SIMATIC S7, other CPUs, and any other programming devices, HMI or with PROFIBUS DP slaves. The maximum number of field devices that can be connected is 16 DP slaves. However, all Siemens products within the range can be integrated into an existing automation system, via PROFIBUS or PROFINET.

From a field wiring perspective, PROFIBUS and FOUNDATION fieldbus are physically identical, as both uses the same twisted pair cables and device couplers, and require the same segment terminators. Both handle up to 32 devices per segment.

Either PROFIBUS PA or FOUNDATION fieldbus H1 works with a single twisted pair wire carrying a digital signal and DC power that connects up to 32 fieldbus devices to a DCS or similar control system. Most devices are two-wire-bus-powered units requiring 10 to 20mA. Instead of running individual cables, both systems allow multiple instruments to use a single cable, called a “trunk” or a “segment”; each instrument connects to the cable as a “drop.”

The fieldbus segment begins with an interface device at the control system. On a FOUNDATION fieldbus (FF) H1 system, the interface is called an H1 card; on a PROFIBUS PA system, it is a PROFIBUS DP/PA segment coupler. In terms of signal wiring and power requirements for the segment, FF and PA are identical:

- Minimum device operating voltage of 9V;
- Maximum bus voltage of 32V;
- Maximum cable length of 1900m (shielded twisted pair).

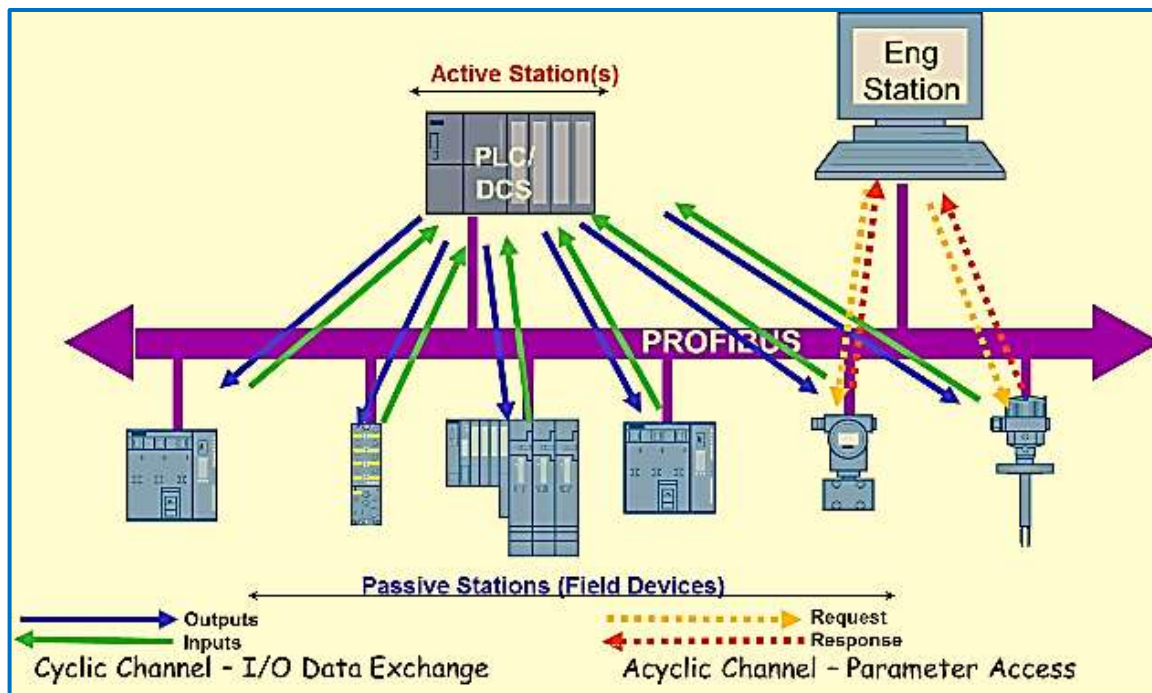
Other main difference between PROFIBUS PA and FOUNDATION fieldbus H1 are:

- ✓ PROFIBUS is a polling system, while FOUNDATION fieldbus utilizes a cyclic transmission (see definition below for cyclic and acyclic data). Polling involves the bus master asking for information from the devices. Polling operation is most often used in terms of input/output I/O, and is also referred to as polled I/O or software-driven I/O.
- ✓ PROFIBUS does not have function blocks capabilities, and instrumentation reports to the PA master; if communications are lost, the instruments must go to a fail-safe position or maintain their last settings until otherwise directed. Profiles are available for encoders, linear transducers, CNC machines, Drives, PLC, SCADA, digital instruments, etc.
- ✓ PROFIBUS PA uses PROFIBUS DP, which is an RS-485 network, or PROFINET, an Ethernet-based network, to connect its PA devices to the DCS. FOUNDATION fieldbus (FF) uses an HSE (high speed Ethernet) network to connect remote H1 cards to the DCS; and continue to operate if communications are lost to the control system.

PROFIBUS supports include cyclic and acyclic data exchange, diagnosis, alarm-handling, and isochronous messaging. The ability to embed *software commands* into the memory of devices represents the real difference between *digital and analog I/P* segments. The most common fieldbus networks methods are:

- **PROFINET:** Is a standard for computer network, which usually uses TCP/IP and Ethernet. It defines the communication with field connected peripheral devices in a cascading real-time basis. PROFINET IO is designed for the fast data exchange between Ethernet-based field devices and can be integrated in the system via an IO-Proxy (subordinate bus system). The PROFINET Component Based Automation (CBA) system consists of various automation components and covers all mechanical, electrical and IT variables.
- **Cyclic data:** Is information that is pre-configured to pass from one device to another at a known rate. Cyclic data is the sender and the receiver end of the message. Therefore if this cyclic data is not delivered with the proper timing, faults will occur on the network to be monitored for reliability assurance.
- **Acyclic data:** Are messages sent and received at any time as they are generated by the sender and generally have a lower priority than cyclic messages. The system incorporates a “request” and “response” communications scheme where the message sender waits to receive a response from the target before generating another message.

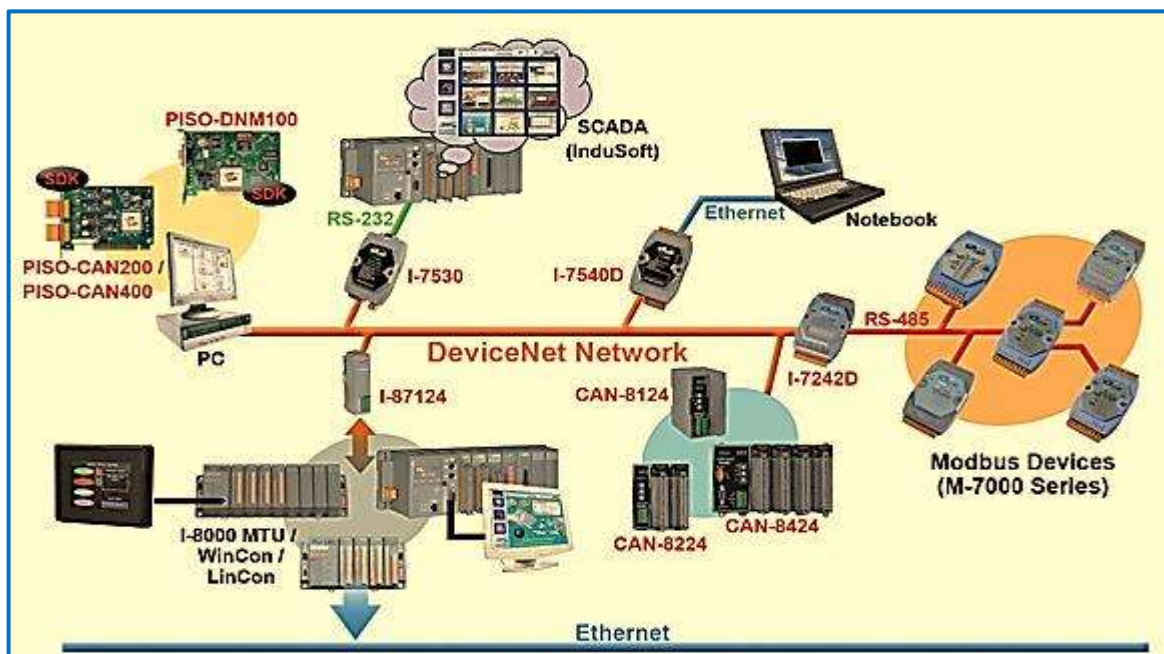
GSD File: GSD stands for “General Station Description.” The manufacturer of the devices is responsible for providing the GSD file, which describes the PROFIBUS functionality of a device or PLC, required in order to have the device recognized by the controller. Integration and configuration of a new device is done by importing a GSD file and synchronizing the address of the device.



DeviceNet: Is a digital, multi-drop network to connect sensors, actuators and automation systems in general. It was developed for maximum flexibility between field equipment and interoperability among different manufacturers. Originally introduced in 1994 by Allen Bradley, today Rockwell Automation, the DeviceNet transferred its technology to ODVA (Open DeviceNet Vendor Association) in 1995.

ODVA is a non-profit organization made up by hundreds of companies worldwide that maintains, promotes and disseminates the DeviceNet networks, based on the CIP (Common Industrial Protocol). The DeviceNet network is classified as a device bus network, whose characteristics are high speed, byte-level communication that includes analogical equipment communication and high diagnostic power by the network devices.

The DeviceNet technology is an open automation standard aiming at transporting 2 types of information: *Cyclic data* from sensors and actuators directly related to control and *acyclic data* indirectly relates to control, like configuration and diagnostics. The *cyclic data* represent information exchanged periodically between the field equipment and the controller. The *non-cyclic* are information exchanged eventually during the configuration or the field equipment diagnostics.



Physical Layer: The access to the DeviceNet network is based on the *CAN (Controller Area Network)* technology and the CIP protocol upper layers, which is defined by an architecture based on objects and the connection between them. Using CAN as a backbone, DeviceNet requires minimal bandwidth to transmit and package messages. The CAN provides fast and reliable response for application mainly used on the automotive area.

A DeviceNet network may have up to 64 devices, from 0 to 63. A DeviceNet node is modeled as a collection of objects. An object example and an object class have attributes or data that provide services (methods or proceedings) and implement behaviors, nodes attributes, examples, classes and

addresses (0 - 63) are addressed with a number. There is no restriction for using the addressing nodes, although the use of the 63 is not recommended, as this is normally used for commissioning.

The DeviceNet uses a trunk line/drop line type of network topology that allows for both the signal wiring and the power source to share the same cable. Since the physical layer is optically isolated from the device, communication power and device power can share the same bus. The power supplied by a source connected directly to the network has the following description:

- 24 Vdc;
- DC outlet isolated from the AC inlet;
- Current capacity compatible with the installed equipment;
- The total length of the network varies according to the transmission rate (125,250, 500Kbps)

DeviceNet supports 125 kbit/s, 250 kbit/s and 500 kbit/s data rates, depending on the chosen cable types. DeviceNet can support communication up to 500 meters (using round, large diameter cable). Typical round cable supports up to 100 meters, while flat-style cable supports up to 380 meters at 125 kbit/s and 75 meters at 500 kbit/s.

The basic trunk line-drop line topology uses 1 cable (two twisted pairs separated for power and signal). Thick or thin cable may be used for trunk lines or drop lines. The distance between network extremes varies with the data rate and cable length:

RATE DATA	125 Kbps	250 Kbps	500 Kbps
Length of main bus with thick-trunk cable	500 m	250 m	100 m
Length of main bus with thin-trunk cable	100 m	100 m	100 m
Maximum length of a main bus maximum-drop	6 m	6 m	6 m
Accumulated length of a main bus cumulative-drop	156 m	78 m	39 m

In-Cabinet (IP20) Distributed I/O: Modular I/O is a system of interface cards and communications adapters that interface directly to the sensors and actuators of the machine/process and communicate their status to the controller via a communication network. It allows the designer to mix and match I/O interfaces and communications adapters.

Block I/O is a complete assembly of sensor and actuator interface points including a network adapter. Safety block I/O can be used with Rockwell Automation safety controllers, to communicate on DeviceNet using CIP Safety. The following software option is available for your DeviceNet network:

Product	Description
IOLinux Software	Help to design an application software to control and collect information from a network.

ControlNet: Is a real-time control network that provides high-speed transport of both time-critical I/O and interlocking data and messaging data, including upload/download of programming and configuration data on a single physical media link. The ControlNet network's highly efficient data transfer capability significantly enhances I/O performance and peer-to-peer communication in any system or application where it is used.

ControlNet was earlier supported by ControlNet International, but as DeviceNet, in 2008 the management of ControlNet was transferred to ODVA, which now supports all protocols in the Common Industrial Protocol (CIP) family. Today, Rockwell Automation provides the ControlNet Traffic Analyzer software to analyze ControlNet packets.

ControlNet and EtherNet/IP: Together with DeviceNet, there is a common object structure, in other words, it is independent from the physical layer and the data link layer. ControlNet offers good real-time capabilities, and used with EtherNet I/P layer, provides a high-speed deterministic transmission for time-critical I/O data and messaging data.

Ethernet uses the CSMA/CD (Carrier Sense Multiple Access with Collision Detection) mechanism for resolving contention on the communication medium. The CSMA/CD protocol specified in the IEEE 802.3 network standard means that, when a node wants to transmit, it listens to the network. If the network is busy, it waits until the network is idle; otherwise it transmits immediately.

If two or more nodes listen to the idle network and decide to transmit simultaneously, the messages of these transmitting nodes collide and the messages are corrupted. While transmitting, a node must also listen to detect a message collision. On detecting a collision between two or more messages, a transmitting node stops transmitting and waits a random length of time to retry its transmission.

When a transceiver detects a collision, it truncates the current frame, which means that stray bits and pieces of frames frequently appear on the cable. Second, it prevents a node from completing the transmission of a short frame, before the first bit has reached the far end of cable, where it may collide with another frame. For a 10 Mbps Ethernet with a maximum length of 2500 m and four repeaters, the minimum allowed frame time or slot time is 51.2 μ s, which is the time required to transmit 64 bytes at 10 Mbps.

ControlNet and PROFIBUS are typical examples of token-passing bus control networks. These are deterministic networks because the maximum waiting time before sending a message frame can be characterized by the token rotation time. The token bus protocol (IEEE 802.4) allows a linear, multidrop, tree-shaped, or segmented topology.

ControlNet connection has the following characteristics:

- ✓ Provides complete services for configuring, controlling and assessing intelligent devices over a single network and repeatable performance for both discrete and process applications.
- ✓ Supports various topology options like, trunk/drop line, ring, star, tree, and combination of any of these, using repeaters.

- ✓ Allows seamless removing and replacing nodes in the network, can be used in potentially explosive atmosphere;
- ✓ Can detect duplicate node ID, supports multiple media types like, coaxial cable, fiber optics, RG-6 quad shield-cable, etc.

Physical Layer: ControlNet can support up to 99 nodes, bus speed Manchester code of 5 Mbps/s, and I/O or implicit messages involving time-critical data, as well as explicit messages like protocol information or service requests. At the physical layer, offers multiple choices of topologies and media options. The following table describes the options briefly:

Media	2 nodes	32 nodes	48 nodes	49 – 99 nodes
Coax	1000 m (3280 ft)	500 m (1520 ft)	250 m (820ft)	Requires repeaters
Fiber	Depends on fiber and termination quality, repeater capability.			
Coax or fiber with repeaters	Depends upon no. of repeaters, repeater capability, fiber and termination quality.			
Maximum drop length	1 m (3.3 ft)	1 m (3.3 ft)	1 m (3.3 ft)	1 m (3.3 ft)

ControlNet can operate with a single RG-6 coaxial cable bus, or a dual RG-6 coaxial cable bus for cable redundancy of quad-shield variety. Maximum cable length without repeaters is 1000 m and maximum number of nodes on the bus is 99. Repeater can be used to further extend the cable length. The network can support up to 5 repeaters (10 when used for redundant networks). The repeaters do not utilize network node numbers and are available in copper or fiber optics choices.

With fiber optics media, the ControlNet incorporates a full-duplex, point-to-point link using a transmitter and receiver at both ends. Repeater is used for extending the network and increasing the number of nodes per network segment. The following table states the uses and examples of the devices under each class:

Device class	Supports	Example
Messaging class	Unscheduled explicit messaging received from all other classes	Computer Interface cards, network configuration and diagnostic tools etc. , software application that do not require real-time I/O response
Adapter class	Receiving scheduled I/O data connection requests from the scanner class and unscheduled explicit message requests from other classes	I/O rack adapters, different robots, weigh scales, welders, and drivers that send and receive real time scheduled data at the request of the PLCs and other controllers.
Scanner class	Sending scheduled I/O data connection requests to the adapter class	PLCs, controllers and robots etc.

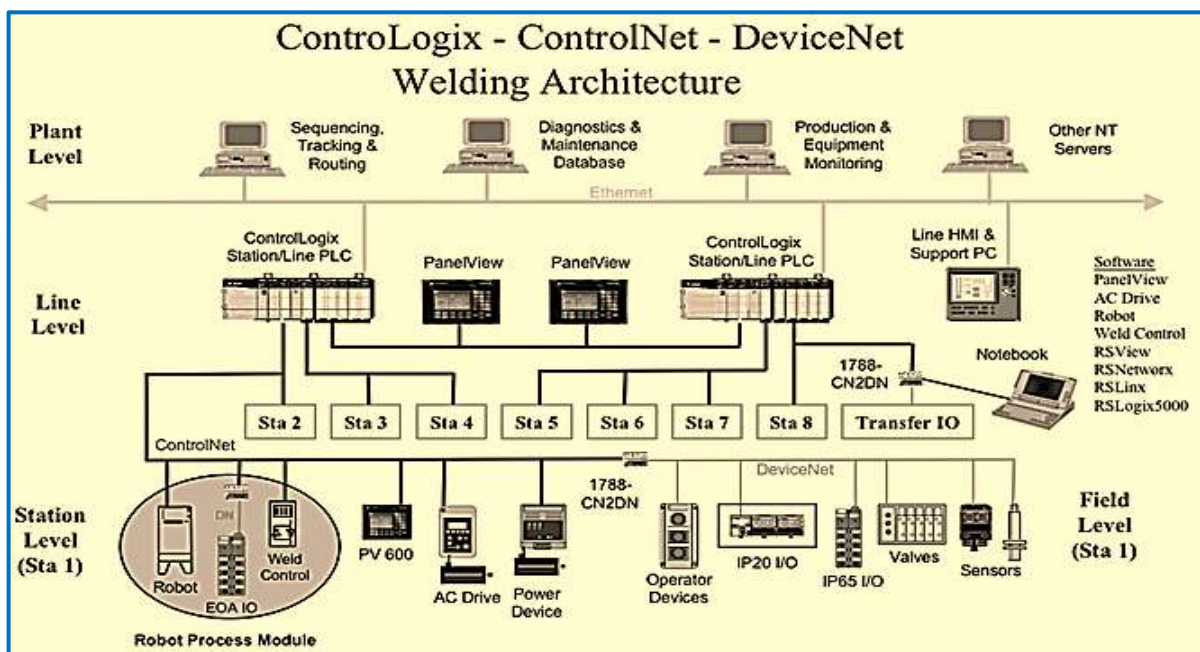
Rockwell Automation offers a full array of ControlNet products, including controllers, I/O, operator interfaces, media, drives, motion control, and softwares, all the products and know-how that is needed to set up a ControlNet system, to maintain and grow existing applications, including:

- ✓ Fiber media for optical isolation from noise and distances up to 20 km;

- ✓ Fiber ring option for additional topology flexibility;
- ✓ Media option to ensure that a system can maintain operation during a cable fault condition;
- ✓ Intrinsic safety option to install a ControlNet network in hazardous, explosive locations;
- ✓ IP67 installation rated for adherence to standards.

The ControlNet network is most often used as a default network for the ControlLogix platform. Substitute/replacement for the Universal remote I/O network, because ControlNet handles large numbers of I/O points, as defined:

- ✓ Backbone to multiple distributed DeviceNet networks;
- ✓ Peer communication network;
- ✓ High-speed I/O network.



ControlNet is a member of the family of networks that implement common industrial protocol (CIP) at the upper layers. ControlNet combines CIP with the Concurrent Time Domain, Multiple Access (CTDMA) technology, thus producing a serial communication system, which allows deterministic, high-speed I/O and peer-to-peer connection in time-critical applications. The ControlNet Software is:

RSNetWorx for ControlNet:	Provides graphical network management, including an intuitive network browser for multi-network viewing. Available separately or with RSLogix programming software packages.
RSNetWorx MD for ControlNet Add-On:	Add-on to an existing RSNetWorx for ControlNet software. Maintenance and diagnostic components that provide pre-configured analysis and troubleshooting information for the ControlNet network. Includes the RSNetWorx for ControlNet and the MD subsystem.
RSNetWorx MD for ControlNet Bundle:	

WorldFIP: The WorldFIP is standardization to the old fieldbus FIP (Factory Interoperable Protocol) which was defined In France 1989 by the standard NFC46. The FIP standard is based on a French initiative in 1982 to create a requirements analysis for a future field bus standard, when CENELEC (European Committee for Electrotechnical Standardization) proposed a European standard for three existing protocols; one of them was the WorldFIP.

In 1990 a number of partners from Japan and America merged with FIP to the World-FIP Standardization group (that later merged into the Fieldbus Foundation group), with the competing German Profibus, were submitted for European standardization by CENELEC in 1993. The WorldFIP (Factory Instrumentation Protocol) was jointly created by Honeywell, Allen-Bradley, Telemecanique (today Schneider Electric), and several other companies, after updated in 1998.

WorldFIP is a fieldbus network protocol designed to provide links between levels zero (sensors and actuators) and controllers (PLCs, CNCs) in automation systems, so that can be used with all types of automation, including the centralized, the synchronous and the asynchronous applications. This fieldbus protocol is not dedicated only for the real-time requirements, but also for transferring the monitoring information from the plant, along the network to the supervisory equipment.

The function of the physical layer is to ensure the transfer of information bits from one node to all other nodes that are connected to the bus. The transmission medium can be by shielded twisted pair wire or optical fibers. The network architecture of the WorldFIP is relying on the Bus topology, which means that all the devices on the network are attached to the same line (bus).

The physical layer encodes the bits transmitted by the data link layer using the Manchester code. Each cable segment has a maximum of 64 nodes. The repeaters can connect up to four segments at the same time. Although the cable segment length depends on the data bit rate, the standard recommends that the ideal segment length is 1 km. The four transmission speeds of the WorldFIP protocol are: 31.25 Kbp/s (low speed); 1.0 Mbp/s (high speed); 2.50 Mbp/s (high speed); 5.00 Mbp/s (high speed for fiber optic).

Modbus: Is an open, royalty-free serial communications protocol, originally developed by Modicon (now Schneider Electric) in 1979, for use with its Programmable Logic Controllers (PLCs). Today the infrastructure is managed by the Modbus-IDA, a group of independent users and suppliers of automation devices since April 2004, when Schneider Electric transferred rights to that organization, signaling a clear commitment to openness, seeking to drive the adoption of this protocol standard.

In 2011, Modbus has joined with Fieldbus Foundation, PROFIBUS and the HART Communication Foundation, to work cooperatively on the development of wireless gateway specifications based on WirelessHART and the emerging ISA SP 100.11a standard. Headquartered in Massachusetts, the objective is to design specifications as common as possible, ensuring complete compatibility with the existing wired versions of each participant's technology.

Modbus enables communication among many (approximately 240) devices connected to the same network, for example a system that measures temperature and humidity and communicates the results to a computer. Modbus is often used to connect a supervisory computer with a Remote Termi-

nal Unit (RTU) in supervisory control and data acquisition (SCADA) systems. Many of the data types are named from its use in driving relays: a single-bit physical output is called a *coil*, and a single-bit physical input is called a *discrete input* or a *contact*.

The basic Modbus commands can instruct an RTU (Remote Terminal Unit) to change a value in one of its registers, control or read an I/O port, as well as, command the device to send back one or more values contained in its registers. There are many modems and gateways that support Modbus, some of them specifically designed for this protocol. The main Modbus protocols are:

- **Modbus RTU:** Is an open, serial (RS-232 or RS-485) protocol derived from the Master/Slave architecture widely accepted, as the messages are a simple 16-bit CRC (Cyclic-Redundant Checksum). Modbus RTU packets are intended to send data and do not have the capability to send parameters, such as point name, resolution, units, etc. When such parameters are needed to send, one should investigate a BACnet, EtherNet/IP, or other modern protocols.
- **Modbus-TCP:** Is also known as Modbus IP, Modbus EtherNet, and Modbus TCP/IP and means that the Modbus protocol is used on top of Ethernet-TCP/IP, that is, Modbus TCP runs on an Ethernet physical layer. The most basic difference between Modbus RTU and Modbus TCP is that Modbus RTU is a serial level protocol.
- **Modbus ASCII:** There are two basic transmission modes found in serial Modbus connections, ASCII and RTU. These transmission modes determine the way in which the messages are coded. In ASCII, the messages are encoded with hexadecimal values. The characters used for this encoding are 0...9 and A...F. For every byte of information, two communication-bytes are used because every communication-byte can only define 4 bits in the hexadecimal system. In RTU the characters are binary 0...255.

Modbus-WirelessHART: Different implementations use wireline, wireless communication, such as the ISM band, Short Message Service (SMS) and General Packet Radio Service (GPRS). One of the more common wireless networks, makes use of Mesh Networking. The ST81 is a kit suited for testing T810 and learning about the WirelessHART. It provides an entry-level platform and guidance for instrument vendors seeking WirelessHART functionality in their products.

Modbus-RSLogix: The modules of the Rockwell Automation CompactLogix L1x and 1734 Point I/O Adapters can also be configured as a Modbus Master or Slave, with an Add-On Instruction used to configure the modules to create logical definitions for I/O, status, and control information. These include in-chassis interfaces compatible with the large automation controllers such as Rockwell Automation and Schneider Electric, as well as, protocol gateways and industrial wireless solutions.

Controller Area Network (CAN): The Controller Area Network (CAN) was developed by Robert Bosch, GmbH, in the early 1980's, was specially developed for fast serial data exchange between electronic controllers in motor vehicles, but can also be used in industrial microcontroller networks, as an internal bus in machine tools, to interconnect distributed measurement, control and monitoring functions, to highest automation level connecting sensors, actuators and/or user interfaces.

CAN bus is a message-based protocol, designed specifically for automotive applications but now also used in other areas such as aerospace, maritime, industrial automation and medical equipment. The protocol was officially released in 1986 at the Society of Automotive Engineers (SAE). Bosch published the CAN 2.0 specification in 1991. In 2012 Bosch has specified the improved CAN data link layer protocol, called CAN FD, which will extend the ISO 11898-1.

CAN bus is one of five protocols used in the OBD-II (On-Board Diagnostics) for vehicle diagnostics standard. The OBD-II standard has been mandatory for all cars and light trucks sold in the United States since 1996, and the EOBD standard has been mandatory for all petrol vehicles sold in the European Union since 2001 and all diesel vehicles since 2004

Automotive: The modern automobile may have as many as 70 ECUs (Electronic Control Units) for various subsystems. Typically the biggest processor is the engine control unit. Others are used for transmission, airbags, antilock braking/ABS, cruise control, electric power steering, audio systems, power windows, doors, mirror adjustment, battery and recharging systems for hybrid/electric cars, etc. CAN is a multi-master serial bus standard for connecting ECUs and each node requires:

Central processing unit or host processor:

- The host processor decides what received messages mean and which messages it wants to transmit itself.
- Sensors, actuators and control devices can be connected to the host processor.

CAN controller; hardware with a synchronous clock:

- Receiving: the CAN controller stores received bits serially from the bus until an entire message is available, which can then be fetched by the host processor (usually after the CAN controller has triggered an interrupt).
- Sending: the host processor stores its transmit messages to a CAN controller, which transmits the bits serially onto the bus.

Transceiver:

- Receiving: it adapts signal levels from the bus to levels that the CAN controller expects and has protective circuitry that protects the CAN controller.
- Transmitting: it converts the transmit-bit signal received from the CAN controller into a signal that is sent onto the bus.

Each node is able to send and receive messages, but not simultaneously. A message consists primarily of an ID (identifier), which represents the priority of the message, and up to eight data bytes. The improved CAN FD extends the length of the data section to up to 64 bytes per frame. It is transmitted serially onto the bus.

The signal pattern is encoded in non-return-to-zero (NRZ) format and may be received by all nodes. The devices that are connected by a CAN network are typically sensors, actuators, and other control devices. These devices are not connected directly to the bus, but through a host processor and a CAN controller.

Data Transmission: Uses a lossless bit-wise arbitration method of contention resolution. This arbitration method requires all nodes on the CAN network, to be synchronized to a sample with every bit on the network at the same time. This is why some call CAN synchronous, however the term synchronous is imprecise as the data is transmitted without a clock signal in an asynchronous format.

If a logical 1 is transmitted by all transmitting nodes at the same time a logical 1 is seen by all of the nodes, including both the transmitting node(s) and receiving node(s). If a logical 0 is transmitted by all transmitting node(s) at the same time then a logical 0 is seen by all nodes. If a logical 0 is being transmitted by one or more nodes, and a logical 1 is being transmitted by one or more nodes, then a logical 0 is seen by all nodes including the node(s) transmitting the logical 1.

When a node transmits a logical 1 but sees a logical 0, there is a contention and it quits transmitting. By using this process, any node that transmits a logical 1 when another node transmits a logical 0 "drops out" or loses the arbitration. Since the 11 (or 29 for CAN 2.0B) bit identifier is transmitted at the start of the CAN frame, the node with the lowest identifier transmits more zero's at the start of the frame, this is the node that has the highest priority.

For example, consider an 11-bit ID CAN network, with two nodes with IDs of 15 (binary representation, 0000001111) and 16 (binary representation, 0000010000). If these two nodes transmit at the same time, each will transmit the first six zeros of their ID with no arbitration decision being made. When the 7th bit is transmitted, the node with the ID of 16 transmits a 1 (recessive) for its ID, and the node with the ID of 15 transmits a 0 (dominant) for its ID.

When this happens, the node with the ID of 16 will realize that it lost its arbitration, and allow the node with ID of 15 to continue its transmission. This ensures that the node with the lower bit value will always win the arbitration. The ID with the smaller number will win the right to use. Bit rates up to 1 Mbit/s are possible at network lengths below 40 m. Decreasing the bit rate allows longer network distances (as, 500 m at 125 kbit/s). The improved CAN FD extends the speed of the data section by a factor of up to 8 of the arbitration bit rate.

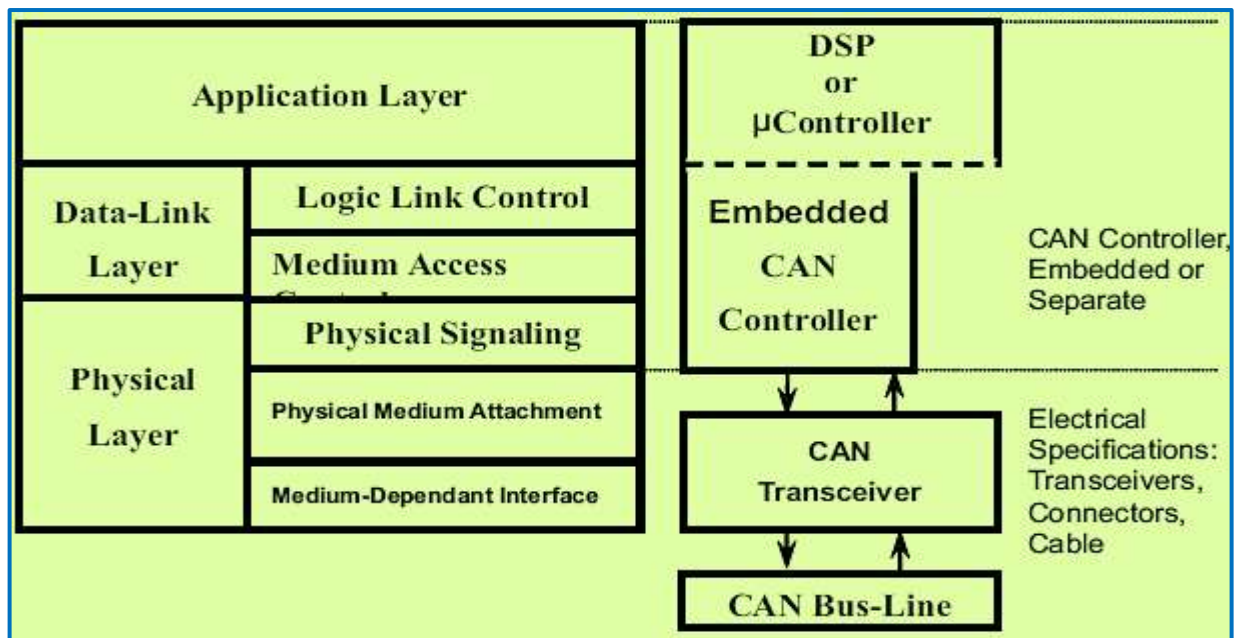
The ISO 11898 standard also describes the physical and the data link layer. The lower protocol levels of CAN (layer 1 and 2) are standardized in the ISO/OSI layer model. Protocols based on layer 7 (application layer) are summarized in different, partly manufacturer-specific standards. Examples for these CAN-based protocols are CANopen and J1939.

CAN FD (CAN with Flexible Data Rate): is an enhancement of the CAN protocol. To improve CAN regarding bandwidth Bosch specified the improved CAN data link layer protocol, called CAN FD. The main differences to CAN are the extended user data from 8 up to 64 bytes, and the ability to send user data with higher data rates. In this way, the requirements for higher bandwidth networks in the automotive industry are fulfilled, while profiting from the experiences in CAN development.

The idea is that after the phases of the classic CAN, the data rate can be increased. The data bits are transferred with a higher bit rate than in the arbitration phase. The speed is limited only by the CAN transceiver signal delay and the signal delay on the cable, integrating the ISO 11898-1 standard as an optional feature. CAN FD has the following properties:

Prioritization of messages; Guarantee of latency times; Configuration flexibility; Multicast reception with time synchronization; System wide data consistency; Error detection and signaling; Automatic retransmission of corrupted messages as soon as the bus is idle again; Distinction between temporary errors and permanent failures of nodes and; Autonomous switching off of defect nodes.

The CAN communications protocol, ISO-11898: 2003, describes how information is passed between devices on a network and conforms to the Open Systems Interconnection (OSI) model that is defined in terms of layers. Actual communication between devices connected by the physical medium is defined by the physical layer of the model. The ISO 11898 architecture defines the lowest two layers of the seven layer OSI/ISO model as the data-link layer and physical layer, as below:



CANopen: Is a communication protocol and device profile specification for embedded systems used in automation. The CANopen standard consists of an addressing scheme, several small communication protocols and an application layer defined by a device profile. The communication protocols have support for network management, device monitoring and communication between nodes, including a simple transport layer for message segmentation/desegmentation.

In terms of the OSI model, CANopen implements the layers above and including the network layer. The lower level protocol implementing the data link and physical layers is usually Controller Area Network (CAN), although devices using some other means of communication (such as Ethernet Powerlink, EtherCAT) can also implement the CANopen device profile.

CANopen is also a CAN-based higher layer protocol, further developed as a standardized embedded network with highly flexible configuration capabilities, basically designed for motion-oriented machine control networks and used in various application fields, such as handling systems, medical equipment, off-road vehicles, maritime electronics, railway applications or even modern building automation.

CANopen provides standardized communication objects for real-time data, configuration data as well as, network management data. CANopen device and application profiles enable device manufacturers to provide their products with standardized interfaces to achieve CANopen devices with "plug and play" capability in CANopen networks. Nevertheless, CANopen also allows to implement manufacturer-specific functionality.

Communication: CAN bus, which is the data link layer of CANopen, can only transmit short packages consisting of an 11-bit, a remote transmission request (RTR) bit and 0 to 8 bytes of data. The CANopen standard divides the 11-bit CAN frame id into a 4-bit function code and 7-bit CANopen node ID. An extension to the CAN bus standard (CAN 2.0 B) allows extended frame ids of 29 bits, but in practice CANopen networks big enough to need the extended id range are rarely seen.

In CANopen the 11-bit id of a CAN-frame is known as communication object identifier, or COB-ID. In case of a transmission collision, the bus arbitration used in the CAN bus allows the frame with the smallest id to be transmitted first and without a delay. Using a low code number for time critical functions ensures the lowest possible delay.

Standard CANopen devices and useful application profiles simplify the tasks of integrating a CANopen system. For system designers, it is very important to reuse application software. This requires not only communication compatibility, but also interoperability and interchangeability of devices. The table below shows the typical transmission speeds and corresponding bus lengths:

Bit Rate	Bit Time	Bus Length
1 Mb/s	1 μ s	25m
800 kb/s	1.25 μ s	50m
500 kb/s 2 μ s 100m	2 μ s	100m
250 kb/s	4 μ s	250m
125 kb/s	8 μ s	500m
62.5 kb/s	16 μ s	1000m
20 kb/s 50	μ s	2500m
10 kb/s 100	μ s	5000m

The consideration of compatibility and choice of cables and connectors belongs also to the tasks of a system integrator. Table below shows the possible cable sections and types for selected network configurations, according to bus cable characteristics:

Bus speed	Cable type	Cable Resist./m	Terminator	Bus Length
50 kb/s at 1000 m	0.75/0.8 mm ² (AWG18)	70 m Ω	150 ... 300 Ω	600 .. 1000 m
100 kb/s at 500 m	0.5/0.6 mm ² (AWG20)	< 60 m Ω	150... 300 Ω	300 ... 600 m
500 kb/s at 100 m	0.34/0.6 mm ² (AWG22)	< 40 m Ω	127 Ω	40 ... 300 m
1000 kb/s at 40 m	0.25/0.34 mm ² (AWG23)	< 26 m Ω	124 Ω	0 ... 40 m

Contents of a standard CANopen frame:

	COB-ID	RTR	Data length	Data
Length	11 bits	1 bit	4 bits	0-8 bytes

Hart Communications Protocol: "Hart" is an acronym for *Highway Addressable Remote Transducer*. The protocol was developed by Rosemount Inc., built off the Bell 202 early communications standard, in 1980 as a communication protocol for their smart field instruments. In 1986, it was made an open protocol and ever since, the capabilities of the protocol have been enhanced by successive revisions to the specification.

In 1993, the registered trademark and all rights were transferred to the HART Communication Foundation (HCF). The protocol remains open and free for all, to use without royalties. According to Emerson due to the huge installed base of 4-20 mA systems throughout the world, the HART Protocol is one of the most popular industrial protocols today.

This protocol makes use of the Bell 202 FSK (Frequency Shift Keying) standard to superimpose digital communication signals at a low level on top of the 4-20mA. Industries are using Profibus DP/PA and Foundation fieldbus (also by Emerson/Rosemount), as become familiar with later technology and look to take advantage of the enhanced diagnostics they can provide.

HART is mainly a master/slave protocol, which means that a slave device speaks only when connected to a master. HART is considered one of the members of the family of digital protocols called "fieldbus", type 20 of the IEC standard along with FOUNDATION Fieldbus, PROFIBUS, Control Net, and others. However, only the digital portion of HART is covered by this standard, which also specifies a 9600 Hz version, also FSK modulated, but not merged with a 4 - 20 mA DC carrier.

The main value of HART, however, is to carry the data produced by smart field instruments such as temperature transmitters, differential pressure transmitters (both high and low pressures), to be used for device configuration and diagnostics, and other secondary variables that can be carried by the Process Variable (PV), which is a digital representation of the current between 4 - 20 mA DC.

As an example, the Rosemount 3144P Fieldbus temperature transmitter (accepts RTD and other sensors) provides process monitoring diagnostics not found in non-FF transmitters. The SPM (Statistical Process Monitoring) is a system available to measure, monitor and trend data centers, provide process engineers with a better view of what is going on in the process, usually by a handheld field communicator or from a centrally intelligent device management software.

ESD means *Emergency Shutdown System*, used for shutdown a particular process or stop an entire plant in the case of emergencies. The ESD PLC uses 4-20 mA and on/off signals for safety functions and can be used for HART transmitter diagnostics. Similarly, in Fieldbus valve positioners there is also performance diagnostics, such as continuous friction monitoring, but not transmitters for control valves.

HART also provides digital integration and diagnostics from discrete devices such as two-wire intelligent on/off valves etc. as well as, tank gauging systems, gas chromatographs, process gas analyzers, etc. If the existing plant has aging temperature multiplexers, where someone cannot find spares (very common scenario), it is better to replace them with multi-channel temperature transmitters, using either Fieldbus or WirelessHART.

However, if a particular category of device, such as turbidity or humidity transmitter is not available, a HART-enabled transmitter can be used instead, because systems that support FOUNDATION fieldbus also support the 4-20 mA/HART, thanks to the magic of EDDL diagnostics from 4-20 mA/HART. Fieldbus and PROFIBUS devices (such as motor starters and drives) can be integrated into the same intelligent device management software.

Electronic Device Description Language (EDDL) technology is used by major manufacturers to describe the information accessible in digital devices. This technology is also used by tool suppliers in process control systems and maintenance to support device diagnostics and calibration. EDDL is a text file, not a software. EDDL is a structured textual declaration similar to an XML, HTML web page, or SGML document, and hence independent of operating systems.

EDDL is useful to anyone who is working with smart transmitters and other intelligent devices applicable to portable tools which are not PC-based, such as handheld communicators and calibrators. EDDL is also applicable to different communication protocols, independent of the underlying communication hierarchy. This makes it possible to integrate data from HART, WirelessHART, Foundation fieldbus, and PROFIBUS devices in the same tool.

WirelessHART: Is a purely digital communication standard, since all data points from the field devices are stored in the gateway in a digital form, and must be accessed digitally. In the case of Emerson's Smart Wireless Gateway, the data may be accessed by any host system via Modbus query commands, communicated either serially (EIA/TIA-485, Modbus RTU format) or encapsulated in Ethernet packets (Modbus TCP).

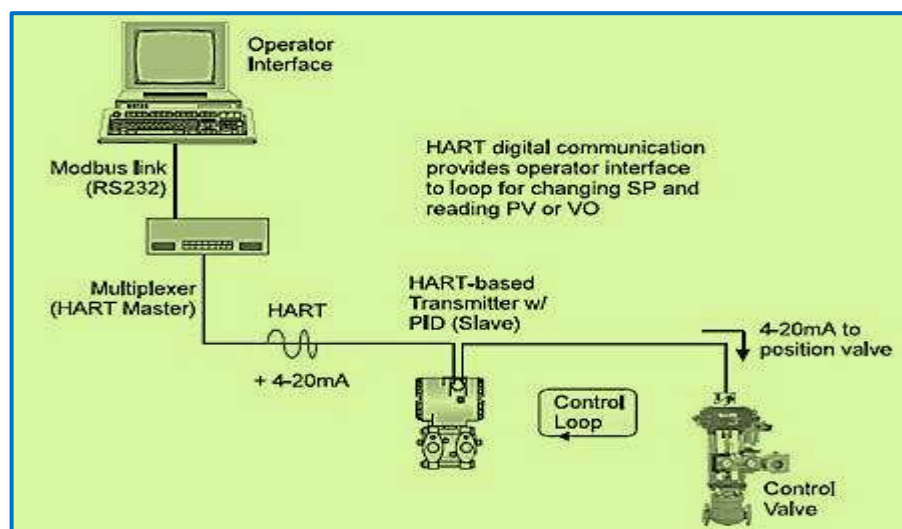
WirelessHART field instruments are multi-variable devices, capable of reporting more than one variable to the gateway. WirelessHART field instruments are like the wired counterparts, with the obvious addition of an antenna. A WirelessHART Rosemount model 648 temperature transmitter appears in this photograph. Removing the large cover on this transmitter reveals the lithium battery:



There are screw terminal connections on the Emerson gateway for an EIA/TIA-485 (RS-485) cable to connect, as well as, multiple RJ-45 Ethernet ports for connection to a hub or switch where Ethernet-based computers and systems can be connected. There is also a pair of metal terminals marked "Comm" on the transmitter, where the battery plugs in, provides a place to connect a standard HART communicator device.

WirelessHART Simulator: Is the first tool that allows simulating a communication in a plant. Someone just needs to open a desired map design (such as tanks, buildings, other obstacles), add the sensors, and is ready to go. The WirelessHART Simulator offers the ability to compute the RF density and potential node path diversity.

Most automation networks in operation today are based on traditional 4-20 mA analog wiring. Then, HART technology serves a critical role because the digital information is simultaneously communicated with the 4-20mA signal. Without it, there would be no digital communication. HART technology is easy to use and very reliable when used for commissioning and calibration of smart devices as well as for continuous online diagnostics.



The price for a Fieldbus device is not that much higher than a hardwired transmitter, and in a new plant you need to lay much less cable using fieldbus and can reduce cabinet footprint, particularly using the new *DeltaV H1* card with integrated power where the fieldbus cable lands directly on the H1 card eliminating the need for an intermediate marshalling cabinet. The many advantages are:

1. There are about 50 times as many HART enabled devices than Foundation Fieldbus devices. Any brownfield plant may have a large investment in HART that could be duplicated with FF.
2. HART connects to the DCS via the same wiring whether it is using the "fieldbus type" data or just the 4-20. The FF needs different wiring configurations.
3. HART is easier for technicians for troubleshooting than FF, without a large amount of training that many employers don't want to provide.
4. HART works exactly the same way when wired or when it is wireless (IEC62591-WirelessHART).
5. PV and diagnostic data from HART transmitters are essentially the same as from FF transmitters.
6. HART is compatible with all the existing wiring in older plants, understood, accepted, and appreciated by virtually everyone involved.
7. Fieldbus costs a lot more to implement, especially for existing equipment, comparing FF over HART for most applications.
8. A lot of instruments are not as readily available with FF and it is much the same reasons that few people want to use Profibus instruments.

SERCOS (Serial Real-time Communication System): Is used in Industrial Control Systems, for interfacing of various control components, for example, metal cutting machine tools, metal forming equipment, assembly machinery, packaging machinery, robotics, printing machinery and material handling equipment.

The SERCOS is a globally standardized open digital interface for the communication between industrial controls, motion devices (drives) and input output devices (I/O). It is classified as standard IEC 61491 and EN 61491. Provides a standard, real-time, high-performance communications interface between motion controls, digital servo drives, and input/output (I/O) devices, ideal for applications where the precise coordination of movement along multiple axes of motion control is critical.

SERCOS I was released in 1991 and the transmission medium used is optical fiber. The data rates supported are 2 and 4 Mbit/s, and cyclic update rates as low as 62.5 microseconds. A ring topology is used. This system also supports a "Service Channel" which allows asynchronous communication with slaves for less time-critical data. **SERCOS II** was introduced in 1999 to expand the data rates supported to 2, 4, 8 and 16 Mbit/s. **SERCOS III** merges the hard-real-time interface aspects of the Ethernet standard.

CIM: This system also is known as the *Computer Integrated Manufacturing*. The CIM is defined at least by three levels of interconnection. Those levels are:

- **Factory Level (Level 2):** This level is responsible to connect different departments within the industrial area (management, product development, maintenance, etc.)
- **Cell Level (Level 1):** In this level can be connected various automation equipment at the factory floor (Robot Controllers, PLCs, CNCs, etc.).
- **Sensor/Actuator Level (Level 0):** In this level are linked sensors and actuators to the controllers, found in level 1.

LonWorks (Local Operating Network): Is a networking platform specifically created to address the control applications needs. The platform was built on a protocol created by Echelon Corporation for networking devices. The technology has its origins over media, such as twisted pair, power lines, fiber optics, RF, chip designs, signaling, routers, network management software, and other products to be used for the automation of various functions within buildings, such as lighting and HVAC.

The platform include diverse functions as embedded machine control, municipal and high-way/tunnel/street lighting, heating and air conditioning systems, intelligent electricity metering, subway train control, building lighting, stadium lighting and speaker control, security systems, fire detection and suppression, and newborn location monitoring and alarming.

The physical-layer signaling technologies are: the twisted pair "free topology" and the power line carrier that are typically inserted in the protocol of the LonWorks technology. The two-wire layer operates at 78 kbit/s, using differential Manchester encoding, while the power line achieves either 5.4 or 3.6 kbit/s, depending on frequency. LonWorks is now an open standard, since it solves a problem of traditional control networks, as nodes has to be specifically designed to work together.

The LonWorks platform also uses the affiliated Internet Protocol (IP) tunneling standard ISO/IEC 14908-4 (ANSI/CEA-852) to connect the devices previously deployed, and new LonWorks platform-based networks to IP-aware applications or remote network-management tools.

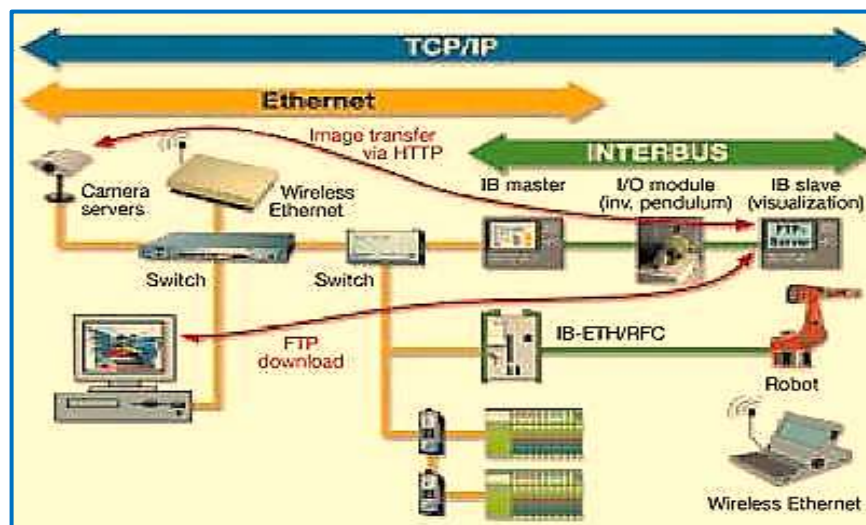
By 2010 approximately 90 million devices were installed with LonWorks technology. Manufacturers in a variety of industries including building, home, street lighting, transportation, utility, and industrial automation have adopted the platform as the basis for their product and service offerings.

INTERBUS: Is a serial bus system which transmits data between control systems (PCs, PLCs, computers, robot controllers, etc.), and spatially distributed as I/O modules to be connected to sensors and actuators (temperature sensors, position switches), developed by Phoenix Contact, then available since 1987. It is one of the leading Fieldbus systems in the automation industry and is fully standardized according to European Standard EN 50254 and IEC 61158.

All PLCs are supported, regardless of manufacturer, and the system offers connection to open computer systems such as PCs in addition to the entire range of field devices used in automation, (drives, encoders, robots, sensors, etc.). The I/O devices are independent of the type of control system. The fieldbus replaces the bundle of parallel cables with a single bus cable and connects all levels, from the field to the control level, regardless of the type of the automation devices used.

Regardless of the type of automation device used, as Programmable Logic Controllers (PLCs) from various manufacturers or PC-based control systems, the fieldbus transmission medium, networks all components, distributed anywhere in the field all connected locally. This provides a powerful communication network for today's rationalization concepts.

There are numerous advantages to a fieldbus system in comparison to parallel wiring. The reduced amount of cabling saves time during planning and installation. The INTERBUS master/slave system enables the connection of up to 512 devices, across 16 levels of networks. The ring is automatically closed by the last device. Self-diagnostics, which are carried out by the system using plain text displays, minimize downtimes and maintenance times.



SWIFTNet: Provides an application-independent, single window interface to all the connected applications of all the institutions participating in the global financial community. Actual access is controlled by the business policy decisions of each Service Administrator, not by the technical limitations of the infrastructure. SWIFTNet provides a basis for assuring business continuity and disaster recovery for the infrastructure of financial applications for institutional boundaries, designed to satisfy institutional community requirements for interoperability of critical financial software solutions.

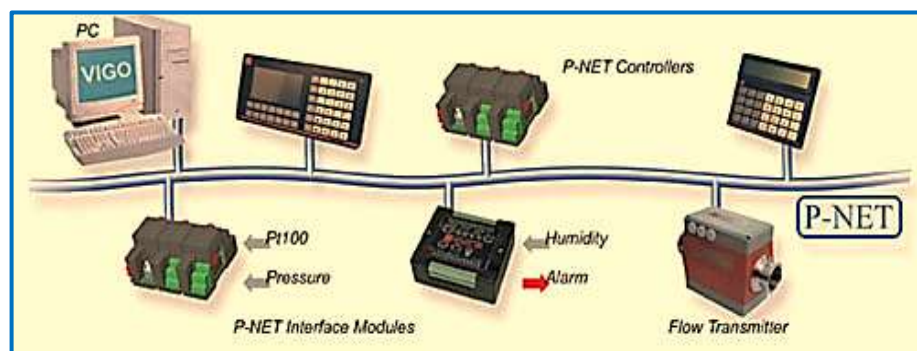
SWIFTNet Link (SNL): Is a business software that uses the application programming interface (API) to access and use SWIFTNet services. The SNL is the mandatory network interface to SWIFTNet, for all external interfaces. The SNL also includes background processes that support messaging, security, and service management functions.

P-NET: Is a multi-master and multi-net standard. All communication is based on a principle, where a Master sends a request and the addressed Slave returns a response. Not only can several masters be connected to each bus segment, but a complex network holding a great number of bus segments can be built also. Communication is routed through the different bus-segments by means of gateways with two or more P-NET interfaces.

P-NET was conceived in 1983 by Process-Data Silkeborg ApS Denmark. The first product using this multi-master fieldbus was launched in 1984. The multi-networking and gateway functions were added to the protocol specification in 1986. The first operational P-NET gateway product (PD3000) was made available in 1987. In 1987, P-NET was adopted by Ultrakust Electronic GmbH, Germany, in the manufacture of its own network instrumentation.

The master on the network can access any node within the network without the need for special programs in the gateways or masters. This feature is a part of the P-NET protocol and is built into the standard operating system. The segmentation also makes it possible to have independent local traffic on each bus-segment, which increases the data throughput throughout the total system.

P-NET is specified and implemented according to the Open Systems Interconnection Reference Model (OSI) on layer 1, 2, 3, 4, and 7. The usage of the application layer (Layer 7) provides the plug-compatibility in relation to other layers. The application layer profiles for individual I/O points, gives the interfaces for different nodes defined as objects, which contain not only real time data but also predefined function switches, diagnostics, maintenance data and error messages.



Physical Layers Comparison for the Main Fieldbus Systems:

Bus Technology	Standards	Pwr w/Comm	Comm Type	Comm Speed	IS Possible	Max Distance	# devices
FF H1	IEC 61158, ISA SP50	Yes	All Digital	31.25 Kbs	Yes	1.9km, 9.5 km	32 per seg
Profibus PA	IEC 61158	Yes	All Digital	31.25 Kbs	Yes	1.9km, 9.5 km	32 per seg
FF HSE	IEC 8802, IEEE 802.3	No	All Digital	100 Mbs, 1 Gbs	No	100 m	Unlimited
ProfiNet	IEC 8802, IEEE 802.3	No	All Digital	100 Mbs, 1 Gbs	No	100 m	Unlimited
MODBUS	IEEE 1451.2, TIA-485	No	All Digital	9.6 Kbs – 12 Mbs	No	1512 m	247 per seg
Profibus DP	IEEE 1451.2, TIA-485	No	All Digital	9.6 Kbs – 12 Mbs	No	1512 m	247 per seg
HART	Bell 202, 4-20mA	Yes	Digital over analog	1.2 Kps – 9.6 Kps	Yes	3.0 km	1 w/Analog, 64

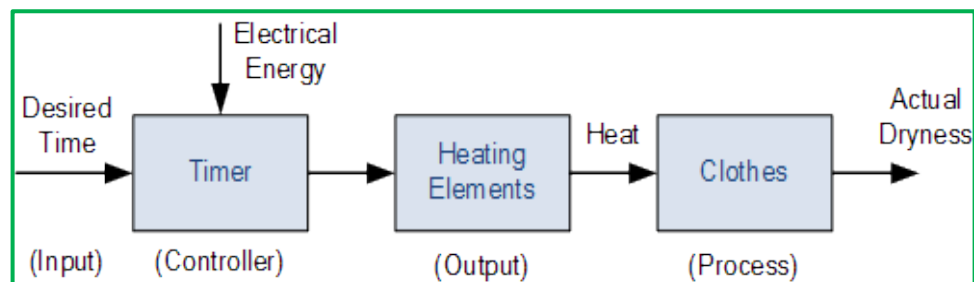
Bus Technology	Standards	Data Link Type	Error Detection	Deterministic	Comm Relationships	Time Features
FF H1	IEC 61158, ISA SP50	Token Passing	16-bit CRC	Yes	Client/server, pub/sub, sink/source	TM distributes time
Profibus PA	IEC 61158	Token Passing	16-bit CRC	Yes	Master/slave	None
FF HSE	IEC 8802	Token Passing	16-bit CRC	No	Client/server, pub/sub, sink/source	TM distributes time
ProfiNet	IEC 8802	Token Passing	16-bit CRC	No	Master/slave	None
MODBUS	None	master/slave address scheme	1-bit	No	Master/slave	None
Profibus DP	IEC 61158	master/slave address scheme	1-bit	No	Master/slave, pub/sub	None
HART	None	Flat addressing	CRC	No	Master/slave	None

Bus Technology	Standards	Data Transfer	Supports Control in the Field	Peer to Peer Comm	Alerts and Trends in Devices	Time Features
FF H1	IEC 61158, ISA SP50, Function block application based on IEC 61804 (Draft)	AI, AO, DI, DO, PID, PD, CS, MIO, many more	Yes	Yes	Yes	Single sense of time
Profibus PA	IEC 61158	AI, AO, DI, DO	No	No	Yes	None
FF HSE	IEC 61158	Same as H1	Yes	Yes	Yes	Single sense of time
ProfiNet	IEC 61158	Same as DP	No	No	Yes	None
MODBUS	IEC 61158	Registers	No	No	No	None
Profibus DP	IEC 61158	AI, AO, DI, DO	No	No	No	None
HART	IEC 61158	Commands	Yes	No	No	None

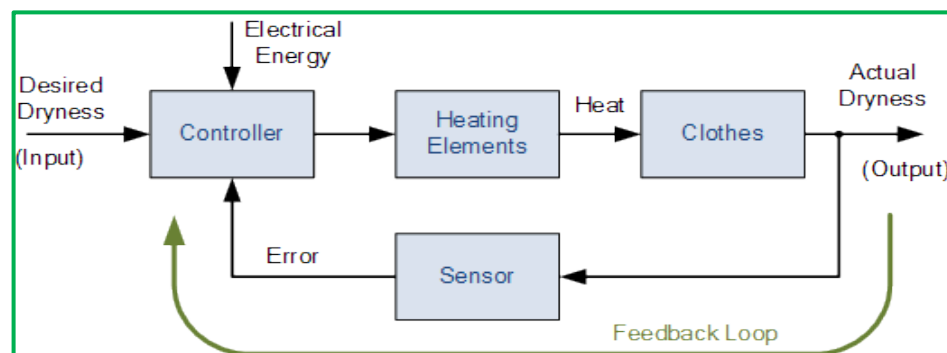
Control Systems: Are a set of devices that manages, commands, directs or regulates the behavior of other devices or systems and may be applied and completed by five elements: *Detectors, Transducers, Transmitters, Controllers, and a Final Control Element*. These functions may be implemented electronically by a proportional control, a PI controller, PID controller, bistable, hysteretic control or Programmable Logic Controller (PLC). The final control changes an input or output that affects the manipulated or controlled variables.

The most common classes of control systems are: *the open loop control systems and the closed loop control systems*. The open loop control systems mean there are no feedbacks from the controlled condition; in other words, no information is sent back from the process or system. In closed loop control systems current output takes into consideration the adjustments and corrections based on *feedbacks*. A closed loop system is also called a feedback control system.

Open-loop System: Is a type of continuous control system in which the output has no influence or effect on the control action of the input signal, also referred to as non-feedback system, is. Then, an open-loop system is expected to faithfully follow its input command or set point, regardless of the final result, as example below:

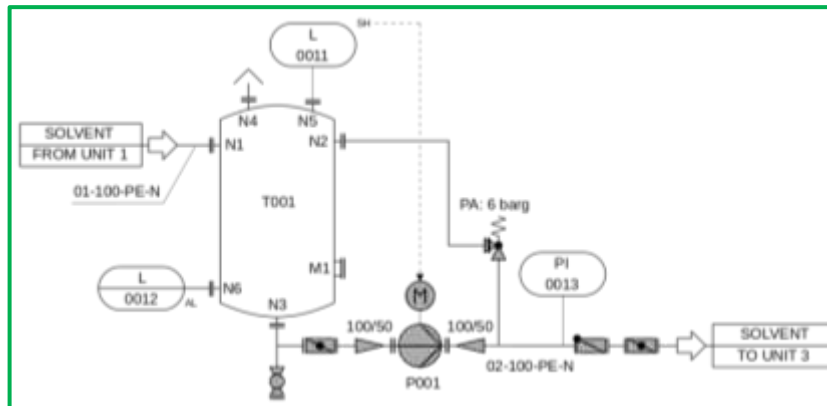


Closed-loop Systems: Are designed to automatically achieve and maintain the desired output condition by comparing it with the actual condition. In other words, a “closed-loop system” is a fully automatic control system in which its control action, gives feedbacks from the controlled condition.



Proportional-Integral-Derivative (PID) Controller: Is a control loop feedback mechanism (controller) widely used in industrial control systems. The PID controller calculates an *error* value as the difference between a measured process variable and a desired setpoint, also called as PI, PD, P or I controller whereas the absence of an integral term may prevent the system from reaching its target value due to the control action.

Piping and Instrumentation Diagram - P&ID: Is a schematic illustration of the functional relationship of piping, instrumentation and system equipment components. The P&ID are used to operate the process system, and shows all of the piping installation through diagrams with standardized symbols, including the physical sequence of branches, reducers, valves, equipment, instrumentation and control interlocks, also allowed for further safety and operational investigations, such as a “Hazard and Operability Study” commonly pronounced as HAZOP.



10. DIAGNOSTICS & COMMUNICATION INTERFACES:

Digital Controllers: Utilizing Distributed Control System (DCS), PC software tools, or handheld communicators, process professionals can diagnose the health of instrumentations while it is in the line. Digital controllers incorporate predefined system devices for maintenance diagnostics to provide alerts if there are problems with instruments, electronics, hardware and valves performances.

DeltaV System: The DeltaV is a Distributed Control System (DCS) developed to protect and improve plant performance. The safety integrity is provided by continuously monitoring sensors, logic solvers and final elements, with faults diagnosed, before causing process failures.

Flow Scanner: Can diagnose the health of valves through a series of off-line tests. The Flow Scanner system consists of a portable computer with pressure sensors that can be connected to valves to enable diagnostic tests. It is possible to diagnose the health of a valve remotely via HART or Foundation fieldbus, which enable predictive maintenance without disrupting the process.

Fieldvue Instruments: Enable new diagnostic that can be accessed remotely. This single element requires a look at the potential impact of the technology as it applies to control valves.

HART-based Handheld Field Communicators: When connected to the digital valve controllers it enables user-configured alerts and alarms, providing notification of current status and potential valve and instrument problems, including travel deviation, travel limit, cycle count accumulation.

Human Machine Interface (HMI): Is typically local to one machine or piece of equipment, and is the interface method between the human and the equipment/machine, where an operator can monitor or operate the system.

11. BASIC COMMUNICATION STANDARDS:

In order to successfully communicate digital data along a network, there must not only be a standard agreed upon between *transmitting and receiving data* devices for encoding bits. Thus, with the objective of a basic knowledge, the signal codes described here are:

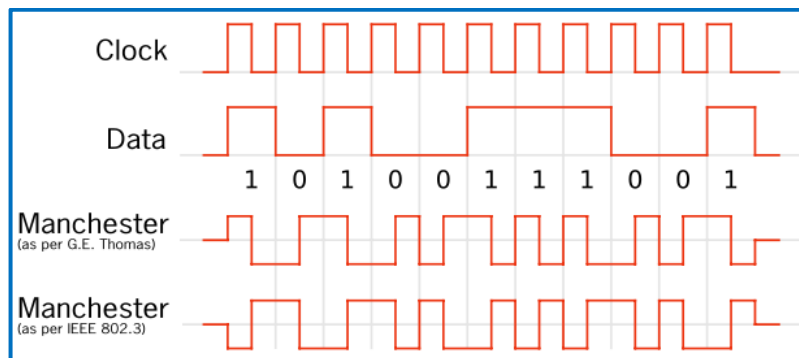
DSR (Data Signaling Rate): Is the total rate at which data pass a point in the transmission path of a data transmission system. The DSR is usually expressed in bits per second (bps).

Self-Clocking Signal: Is a line code commonly used in telecommunications and electronics, decoded without the need of a separate clock signal or other source of synchronization, by including embedded synchronization information within the signal, and adding constraints on the coding of the data payload, such that false synchronization can be easily detected.

In clock language, "1" transitions or remains high on the trailing clock edge of the previous bit and "0" transitions or remains low on the trailing clock edge of the previous bit, or just the opposite. The "1" is represented by one physical level (such as a DC bias on the transmission line) and "0" is represented by another level (usually a negative voltage).

Manchester Coding: Is a line code (also known as Phase Encoding, or PE), where the encoding of each data bit has at least one transition and occupies the line at the same time. Manchester code always has a transition at the middle of each bit period and may (depending on the information to be transmitted) have a transition at the start of the period.

Each bit is transmitted in a fixed time (period). The "0" is expressed by a low-to-high transition, and the "1" is expressed by high-to-low transition. Electrical connections using a Manchester code are either easily *galvanically isolated* (network isolator), or a simple *one-to-one isolation transformer*. It has no DC component, then, is "self-clocking", which means that it may be inductively or capacitively coupled.



Example of a Manchester encoding, showing both conventions.

ASK (Amplitude-Shift Keying): Is a form of amplitude modulation that represents digital data as variations in the amplitude of a carrier wave. In this system, the binary symbol "1" is represented by transmitting a fixed-amplitude carrier wave and fixed frequency for a bit duration of "T" seconds. If

the signal value is "1" then the carrier signal will be transmitted; otherwise, a signal value of "0" will be transmitted.

FSK (Frequency-Shift Keying): Is a frequency modulation scheme in which digital information is transmitted through discrete frequency changes of a carrier wave. The simplest FSK is the binary FSK (BFSK), which uses a pair of discrete frequencies to transmit binary (0s and 1s) information. With this scheme, the "1" is called the mark frequency and the "0" is called the space frequency.

AFSK (Audio Frequency-Shift Keying): Is a modulation technique by which digital data is represented by changes in the frequency (pitch) of an audio tone, yielding an encoded signal suitable for transmission via radio or telephone. Normally, the transmitted audio alternates between two tones: one, the "mark", represents a binary one; the other, the "space", represents a binary zero.

NRZ (Non-Return-to-Zero): This line code is a binary code, where "1" is represented by one significant condition (usually a positive voltage) and "0" is represented by any other significant condition (usually a negative voltage), with no other neutral or rest condition. NRZ is not inherently a "self-clocking signal", thus some additional synchronization technique (for example a run length limited constraint, or a parallel synchronization signal) must be used for avoiding the bit slip.

RZ (Return-to-Zero): Describes a line code used in telecommunications signals in which the signal drops (returns) to zero between each pulse. This takes place even if a number of consecutive "0s" or "1s" occur in the signal. The signal is self-clocking. This means that a separate clock does not need to be sent alongside the signal, but suffers from using twice the bandwidth to achieve the same data-rate as compared to non-return-to-zero format.

NRZI (Non return to zero, inverted): Is a method of mapping a binary signal to a physical signal for transmission over some transmission media. The two level NRZI signal has a transition at a clock boundary if the bit being transmitted is a logical "1", and does not have a transition if the bit being transmitted is a logical "0". In Universal Serial Bus (USB), NRZI signaling has an opposite convention, when in Mode 1 a transition occurs when signaling 0, and a steady level if signaling a 1.

PSK (Phase-Shift Keying): Is a digital modulation scheme that conveys data by changing or modulating the phase of a reference signal (carrier wave). PSK uses a finite number of phases, each assigned a unique pattern of binary digits.

DPSK (Differential Phase-Shift Keying): Is when the demodulator determines the changes in the phase of the received signal, but depends on the difference between the successive phases.

Communication Speed: The essential parameter in a serial data communication system is the *bit rate, measured in bits per second (bps)*. Some communications standards have fixed *bit rates*, such as FOUNDATION Fieldbus H1 and Profibus PA, both standardized at exactly 31.25 kbps. Some, such as Ethernet, have a few pre-defined speeds (10 Mbps, 100 Mbps, 1 Gbps) defined by the specific transmitting and receiving hardware used. Others, such as EIA/TIA-232 may be arbitrarily set by the user at speeds ranging from 300 bps to over 115 kbps.

Baud Rate and Bit Rate: “Baud” means symbols per second, pulses per second or alternations per second of time. “Bits per second” refers to the number of binary data bits communicated per second of time, sometimes used synonymously, however, “bits per second” and “baud” are different things. Baud is useful when determining whether or not the bandwidth (the maximum frequency capacity) of a communications channel is sufficient for a certain communications purpose.

Nevertheless, using NRZ encoding, the *baud rate* is equivalent to the *bit rate*. For a string of alternating bits (0101010101), there will be exactly one voltage transition for each bit. In some clever encoding schemes, it is possible to encode multiple bits per signal transition, such that the bit rate will actually be greater than the baud rate.

Data Frames: Sometimes, in order to do communication, data must be sent in “frames” or “packets” of fixed (maximum) length, each frame preceded by a special “start” signal and concluded with a special “stop” signal. When the transmitting device issues the “start” signal, the receiving device synchronizes exactly that start time, and runs at the pre-determined clock speed to gather the successive bits of the message until the “stop” signal is received.

That is, as long as the internal clock circuits of the transmitting and receiving devices are running at approximately the same speed, the devices are synchronized closely enough to exchange a short message without any bits being lost or corrupted. The synchronous digital network works with all transmitting and receiving devices, locked into a common clock signal. The advantage of synchronous communication, is that there is no time wasted on “start” and “stop” bits, since data transfer may proceed continuously rather than in packets.

Master-Slave: The first network method is the principle of having only one device on the network designated as the “master”, with permission to arbitrarily transmit data. All other devices on the network are “slaves,” which may only respond in direct answer to a query from the “master”. If the network happens to be simplex in nature, “slave” devices don’t even have the ability to transmit data, all they can do is “listen” and receive data from the unique “master” device.

Token-Passing: Another method of arbitrating which device gets to transmit on a channel in a half-duplex network, is the token-passing method. Here, a special data message called the “token” serves as temporary authorization for each device to transmit. Any device in possession of the token is allowed to act as a master device, transmitting at will. Token-passing ensures only one device is allowed to transmit at any given time, and it also solves the problem inherent to master-slave networks of what happens when the master device fails.

If one of the devices fails, its silence will be detected after the last token-holding device transmits the token message to the failed device. Examples of token-passing networks are the Token Ring network standard (IEEE 802.5) and the passed Token Bus (IEEE 802.4). Some proprietary industrial networks, such as Honeywell’s TDC 3000 network (called the Local Control Network, or LCN) utilize token-passing arbitration.

TDMA (Time Division Multiple Access): Is a method of channel arbitration, very similar to token-passing. Here, each device is assigned an absolute “time slot” in a repeating schedule when it alone is allowed to transmit. When TDMA permission to transmit is granted by an appointment on time

schedule, this method is less time-efficient than token-passing, because devices with no data to transmit occupy the same amount of time in the schedule, as when they have data to transmit.

Examples of TDMA networks include the GSM cellular telephone standard as well as the WirelessHART and ISA100.11a radio instrumentation standards. TDMA arbitration works very well for wireless networks where the communication channel is inherently unreliable due to physical obstacles. If a device on a TDMA wireless network falls out of range or becomes blocked, the rest of the network carries on without missing a step.

CSMA (Carrier Sense Multiple Access): Is where any and all devices have permission to transmit, when the network is silent. There are no dedicated master and slave devices with CSMA, nor are devices permitted to transmit in a pre-determined order as with token-passing or in a pre-determined schedule as with TDMA. Any device on a CSMA network may “talk” in any order and at any time whenever the network is free.

CSMA/CD (Carrier Sense Multiple Access with Collision Detection): When two or more devices begin communicating simultaneously, this is called “collision”, and must be addressed in order for any CSMA network to be practical. Multiple methods exist to overcome this problem used in Ethernet. With CSMA/CD, all devices are not only able to sense an idle channel, but are able to sense when they have “collided” with a transmitting device. In the event of a collision, the colliding devices cease transmission, and set random time-delays to wait before re-transmission.

CSMA/BA (Carrier Sense Multiple Access with Bitwise Arbitration): A different method of avoiding collisions, is to pre-assign each device on the network with a priority number to determine the order of re-transmission. This is the strategy used in DeviceNet, an industrial network based on CAN technology, one of the more popular data networks used in automotive engine control systems.

CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance): Is the technique used for WLAN networks (IEEE 802.11 specification). This device with higher-priority (shorter wait times) will always have an advantage in transmitting data over devices of lower priority. The degree of disparity in network access grows as more devices occupy the network.

EIA/TIA-232, 422, and 485: Are some of the basic types of digital communication networks defined by the EIA (Electronic Industry Alliance) and TIA (Telecommunications Industry Alliance) groups, under the numerical labels 232, 422, and 485.

EIA/TIA-232C: Formerly known as RS-232, is the standard of layer 1 of the OSI Reference Model (voltage signaling, connector types) and some details found at layer 2 of the OSI model (asynchronous transfer, with signals between transmitting and receiving devices).

In the early days of personal computers, almost every PC had either a 9-pin or a 25-pin connector (and sometimes multiple of each) dedicated to this form of digital communication. It was also the way peripheral devices such as keyboards, printers, modems, and mice were connected to the PC. The USB (Universal Serial Bus) has now replaced EIA/TIA-232 for personal computers, but it still lives on in the world of industrial devices.

EIA/TIA-422 and 485: Are very different from 232, as designs were intended to optimize both maximum cable length and maximum data rate. The electrical signaling used for both EIA/TIA-422 and EIA/TIA-485 is a differential single-ended, that is, a dedicated pair of wires is used for each communications channel rather than a single wire whose voltage. Using dedicated wire pairs instead of single conductors, means that EIA/TIA-422 and EIA/TIA-485 networks enjoy much greater immunity to induced noise than EIA/TIA-232.

RSSI (Received Signal Strength Indication): Is a measure for each device receiving RF signals, in units of dBm. Radio frequency (RF) is a rate of oscillation in the range of around 3 kHz to 300 GHz, which corresponds to the frequency of radio waves, and the alternating currents which carry radio signals. Problems related to antennas, path loss, fade loss, and interference will result in decreased RSSI for that device. WirelessHART devices are also within the range of RSSI.

Ethernet: An engineer named Bob Metcalfe conceived the idea of Ethernet in 1973, while working for the Xerox research center in Palo Alto, California. His fundamental invention was the CSMA/CD method of channel arbitration, allowing multiple devices to share a common channel of communication while recovering gracefully from inevitable “collisions.” In Metcalfe’s vision, all of the “network intelligence” would be built directly into “controller” devices situated between the DTE devices (computers, terminals, printers, etc.) and a completely passive coaxial cable network.

Metcalfe’s original network design operated at a data rate of 2.94 Mbps, impressive for its time. Around 1980, the three American computer companies DEC (Digital Equipment Corporation) Intel, and Xerox collaborated to revise the Ethernet design to a speed of 10 Mbps, and released a standard called the DIX Ethernet standard (the acronym “DIX” representing the first letter of each company’s name). Later, the IEEE Local and Metropolitan Networks Standards Committee codified the DIX Ethernet standard under the numeric label 802.3. At the present time there exist many “supplemental” standards underneath the basic 802.3 definition, a few of them listed here:

- 802.3a-1985 10BASE2 “thin” Ethernet;
- 802.3d-1987 FOIRL fiber-optics link;
- 802.3i-1990 10BASE-T twisted-pair cable Ethernet;
- 802.3u-1995 100BASE-T “Fast” Ethernet and Auto-Negotiation;
- 802.3x-1997 Full-Duplex standard;
- 802.3ab-1999 1000BASE-T “Gigabit” Ethernet over twisted-pair cable.

IEEE 802.3 Standard: Is limited to layers 1 and 2 of the OSI Reference Model: the “Physical” and “Data link” layers. In the physical layer (1), the various supplements describe all the different ways in which bits are electrically or optically represented, as well as permissible cable and connector types.

In the data link layer (2), the IEEE standard describes how devices are addressed (each one with a unique identifier known as a MAC address, consisting of a 48-bit binary number usually divided into six bytes, each byte written as a two-character hexadecimal number), and also how data frames are organized for Ethernet transmissions.

LRV and URV: Stand for Lower and Upper-Range Values (LRV and URV) in a smart transmitter, to calibrate the analog-to-digital and digital-to-analog converter circuits independently of each other. Thus, this means that a full calibration procedure on a smart transmitter, potentially requires more work and a greater number of adjustments than an all-analog transmitter.

A common mistake made among students and experienced technicians are about the range settings (LRV and URV) for actual calibration adjustments. The digitally setting of the LRV of a pressure transmitter to 0.00 PSI and the URV to 100.00 PSI does not necessarily mean the accurately register at points within that range.

The Future of the Fieldbus Systems: Like all other technological products Fieldbus is up to continuing process of updating. Further developments are going on in the fieldbus technology especially in the vehicle systems. A new era has been born which some specialists called the "**X-by-wire**". This technology tends to replace all the mechanical linkages that are found in the vehicles with digital links and wire all these links into one network protocol that entirely runs the vehicle.

Rising now in the horizon of this era are two protocols. The first is called **FlexRay**, and the second is called **TTP**. These two are based on the older kin; the **CAN**. Among the anticipated benefits: better fuel economy, better vehicle performance in adverse conditions, and advances in safety features such as collision warning and even automatic collision avoidance systems.

FlexRay: Is a hybrid protocol that allocates portions of network time to both a time triggered protocol and to prioritized message access. While the CAN prioritization scheme is based on dominant and recessive bit-values, FlexRay uses timing offset values proportional to priority. As its name suggests, FlexRay adds flexibility—permitting coexistence of both prioritized and time-triggered messages on the same network.

TTP (Time-Triggered Protocol), **FTT-CAN** (Flexible Time-Triggered Communication on CAN), and **TCN** (Train Communication Network). Are other new proposed protocols.

12. REFERENCES & LINKS:

REFERENCES: To learn more about fieldbus look at the yellow book "Fieldbuses for Process Control: Engineering, Operation, and Maintenance". Also check out other fieldbus courses.

Kuphaldt, Tony R., Lessons in Industrial Instrumentation.

FOUNDATION Fieldbus Technical Overview, FD-043 Revision 3.0.

Foundation Fieldbus Overview, National Instruments, May 2003 Edition.

Romero, Vanessa S. and Theorin, Alfred, History of Control History of PLC and DCS, 2012-06-15.

LINKS:

<http://www.isa.org/fieldbuses>;

<http://www.profibus.com>;

<http://www.fieldbus.org>.

<http://www.modbus.org/>

<http://pt.hartcomm.org/>
<http://www.ni.com/white-paper/2732/en/> (CAN Overview)
<http://www.interbus.com/get.php?object=497>
<http://www.emerson.com/en-US/Pages/Default.aspx>
<http://www.schneider-electric.com/site/home/index.cfm/ww/>
<http://www.siemens.com/entry/cc/en/>
<http://www.geindustrial.com/>
<http://www.plantwebuniversity.com;>
<http://www.rockwellautomation.com/>
<http://www.mikroe.com/old/books/plcbook/chapter1/chapter1.htm>
<http://www.automationnc.com/>
<http://www.plcprofessor.com/>
<http://www.plcdev.com/>

VIDEOS:

https://www.youtube.com/watch?v=Cm_xlqmPm0c&list=PLE98A38E00B457A81&index=2
<https://www.youtube.com/watch?v=-8DVa3SBu38>
https://www.youtube.com/watch?v=BHpVcS_QDBc
https://www.youtube.com/watch?v=BHpVcS_QDBc
<https://www.youtube.com/watch?v=CyvMRxLfm8w>
https://www.youtube.com/watch?v=zvS_BuQISXo
<https://www.youtube.com/watch?v=J5dMQlvwa4o>
<https://www.youtube.com/watch?v=z-qojCqX8iw>
<https://www.youtube.com/watch?v=95hIMjxwy8o>
<https://www.youtube.com/watch?v=k993tAFRLSE>
<https://www.youtube.com/watch?v=Khvhi-O2uOI>