Learning resource

Demonstrate knowledge of programmable logic controllers (PLCs)

Level 4 | Credits 3

Te Pūkenga

Contents

Part 1: Programmable Logic Controller Principles 1
What is a PLC? 2
Why PLCs are better than relay and hard-wired logic 3
PLCs vs Relays 3
Hard-wired logic 4
PLC functional components and hardware modules 5
Input Output (I/O) devices 7
I/O modules 8
Power supply 11
Central processing unit (CPU) 12
Input relays (contacts) 12
Internal auxiliary relays (contacts) 13
Counters 13
Timers 13
Output relays (coils) 13
Data storage 14
High-speed counter modules 14
High-speed timer modules 14
I/O signal types 15
PLC Memory, Types and Uses 16
ROM — Read Only Memory 16
PROM — Programmable Read Only Memory 16
EPROM — Erasable Programmable Read Only Memory 17
EEPROM — Electrically erasable programmable read only memory 17
RAM — Random access memory 17
Flash RAM 18
Operating sequence of a PLC 19
Operating modes 19
Cycle or Scan time 22
PLC programmers 23

Dedicated hand-held programmers 23
Personal computers 24
Programming Methods 25
Ladder logic 25
Symbolic block or Function block diagram 28
Sequential Function Chart 30
Structured Text 30
Instruction List 31
IEC program examples 32
PLC Terms 33
Bit level functionality 33
Word level functionality 35
Special functionality timing 36
Flags 38
Emergency Stop Logic 39
Hard-wired emergency stop 39
PLC emergency stop considerations 39
Simple PLC emergency stop application 40
Review questions 41
Part 2: Design, Write and Store a PLC Program 44
Process and instrumentation diagram (P&ID) 45
Function description (FD) 46
Writing Your Program 48
Programming the sequence 48
The Emergency Stop 49
Programming the outputs 50
Storing your program 51
Answers to review questions 52

Part 1: Programmable Logic Controller Principles

For this module of learning, you should have prior knowledge of Relays, Contactors, and Control Circuits for Motor Starting, control systems, heat control, and so on.

You need to have access to a personal computer and have a good general knowledge and understanding of its components and their interfaces.

In the module for unit standard15862 you learned about the various forms of closed loop control systems and how they can be used to keep a controlled variable at a precise set value. A Programmable Logic Controller, commonly known as PLC, can also be used to perform this function.

What is a PLC?

The definitions for PLC vary, depending on the experience of various authors of brand, size, type and complexity. However, a general definition for PLC is as follows:

A PLC is a dedicated computer, which is designed to operate in an industrial environment, specifically for the purpose of monitoring and controlling plant and equipment.

The PLC is connected to the plant using Inputs and Outputs that are connected to sensors and actuators.

Generally, PLCs or their Inputs and Outputs (I/O) are able to operate in extremes of temperature, corrosive atmosphere, and shock or vibration.

Originally PLCs were designed to be programmed by a simple statement (instruction) list or with ladder logic diagrams rather than the common computer languages of the time such as BASIC or FORTRAN. If you are familiar with relay logic diagrams, you could be trained to start programming a PLC in a few hours.

The term PLC is copyrighted to Allen Bradley even though the Allen Bradleys weren't the first Programmable Controllers available. The use of the term PLC therefore involves trademark issues. For this reason, you will find that many texts and manufacturers refer to PLCs generically, as **programmable controllers (PCs)**.
It is important not to confuse this term with the same abbreviation used for personal computers (PCs). To avoid confusion, this unit refers only to programmable logic controllers (PLCs).

Why PLCs are better than relay and hard-wired logic

PLCs vs Relays

Before the days of the PLC the only way to control machinery was through the use of relays. Relays work by utilising a coil that, when energised, creates a magnetic force to effectively pull a switch to the ON or OFF position. When the relay is de-energised, the switch releases and returns the device to its standard ON or OFF position.

So, for example, to turn a motor ON or OFF, a power relay would be attached between the power source and the motor. The power to the motor is controlled by either energising or de-energising the relay.

Since there could be several motors in a factory that need to be controlled, many power relays get added. Furthermore, to control the power to the coils in the power relays more relays would be added, and this could go on and on. You can now start to see some of the problems with this system of electromechanical control via relays.

Some of the advantages of using PLCs over relays are:

- **Number of contacts.** Relays have a limited number of contacts available for use before an additional relay is required (usually no more than 8) whereas a PLC can have thousands of contacts for a programmed relay.
- **Variety of control tasks.** Relays are limited in what functions they can perform without using multiple devices (on/off, changeover, time delay, counting, impulse control and so on). A PLC on the other hand can perform hundreds of different control tasks, often by using single dedicated instructions rather than lots of complex interconnected ‘hard wired’ logic.
- **Flexibility.** One single PLC can easily run many machines.
- **Ease of alteration and duplication.** Relay logic is difficult to modify or make addition to as you need to re-design the control circuit, rewire the relay panel, re-test and commission. With a PLC, you can modify the code either on-line or at your desk, saving both time and expense.
- **On-line documentation.** When documenting a relay logic circuit, there is considerable time and effort in getting it right, building and wiring the panel, then marking up any alterations and so on, and redrawing the alterations. Documentation for PLCs is done on-line.
- **Time and cost saving.** Designing relays and documenting them, even with modern CAD (Computer Aided Design) programs, can take considerable time and expense. With modern PLCs, the documentation is a part of the programming software functionality, and therefore happens as a part of the programming process, rather than an additional task, saving time and keeping it aligned to the actual logic. The time to “install” the changes is quick as all that is required is to download the modified program to the PLC. In a relay-based system the system must be shut down while the rewiring changes are made.

- **Ease of Troubleshooting.** Back before PLCs, wired relay-type panels required time for rewiring of panels and devices. With PLC control any change in circuit design or sequence is as simple as editing the logic. Correcting errors in PLC is both fast and cost effective.
- **Space Efficient.** Fewer components are required in a PLC system than in a conventional hardware system. The PLC performs the functions of timers, counters, sequencers, and control relays, so these hardware devices are not required. The only field devices that are required are those that directly interface with the system such as switches and motor starters.
- **Low Cost.** Prices of PLCs vary from few hundreds to few thousands of dollars. This is minimal compared to the prices of the contacts, coils, and timers that are required to do the same things. Using PLCs also saves on installation cost and shipping.
- **Testing.** A PLC program can be tested, evaluated, and validated offline prior to implementation in the field.
- **Visual observation.** When running a PLC program, a visual operation displays on a screen or module mounted status lamps assist in making troubleshooting a circuit quick, easy, and relatively simple.

Hard-wired logic

Hard-wired control units are implemented through use of combinational logic units, featuring a finite number of gates or diode matrices that can generate specific results based on the design of the circuit.

A controller that uses this approach can operate at high speed. However, it has little flexibility, and the complexity of the instruction set it can implement is limited.

The main reason that using a PLC is better than using a hard-wired logic circuit is that it is flexible because it is reprogrammable. For fixed applications that will never require modification, such as automatic washing machines, vending machines and the like, they are a perfect solution. However, when compared to PLCs in the industrial environment they are difficult to modify, difficult to fault find, and are generally very limited in function.

PLC functional components and hardware modules

The PLC consists of the following hardware components:

- Power supply
- Central Processing Unit (CPU)
- Memory
- Input/Output (I/O) sections – appropriate devices and circuits to receive or send data
- Programming device – a device to physically write the program into memory.

These components work together to bring information into the PLC from the field, evaluate that information, and send information back out to various field devices.

We can consider the PLC to be a box full of hundreds or thousands of separate relays, counters, timers and data storage locations. These don't physically exist but are simulated by software instructions. Relays, having only two states (on or off), are easily simulated by a single bit of memory or simulated through 'bit locations' in registers which are just combinations of bits.

Figure 1 shows the components of a PLC.

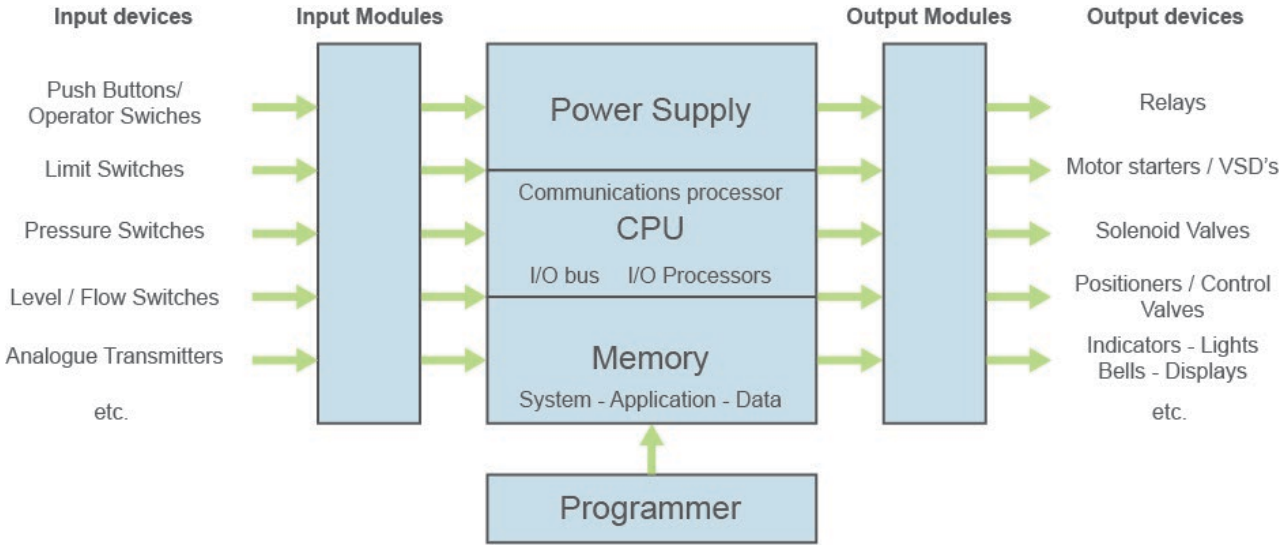
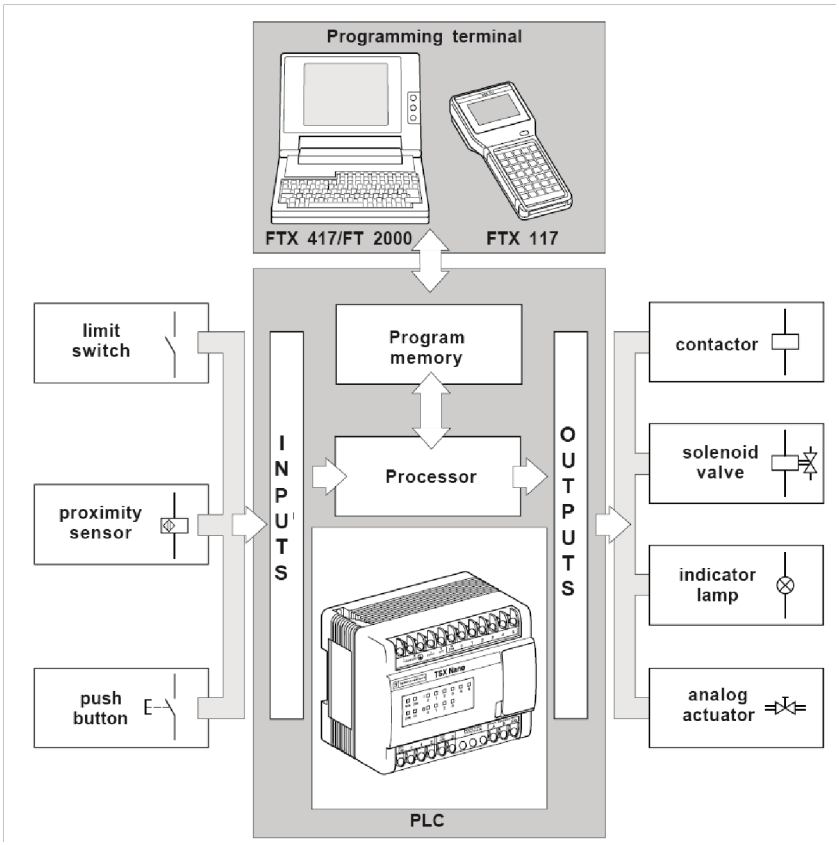


Figure 1: PLC components

In modern PLCs, these components are not as clearly defined. In lower end 'brick' style PLCs, everything is contained in a plastic brick form with the only externally accessible parts being indicators, wiring terminals and a socket to plug in peripheral devices such as a programming terminal or PC.

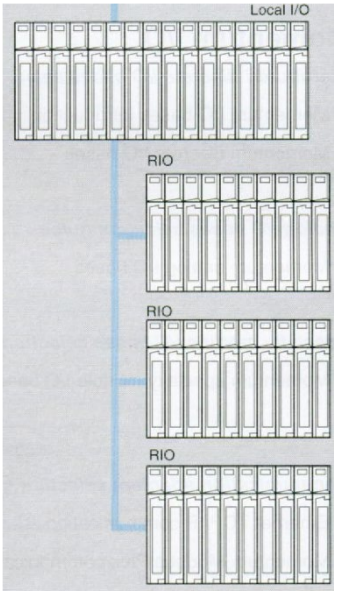
The following picture from Schneider Electrics, TSX NANO PLC Manual shows the ‘brick’ style of lower end PLCs along with the same component relationships seen in the older style PLC.



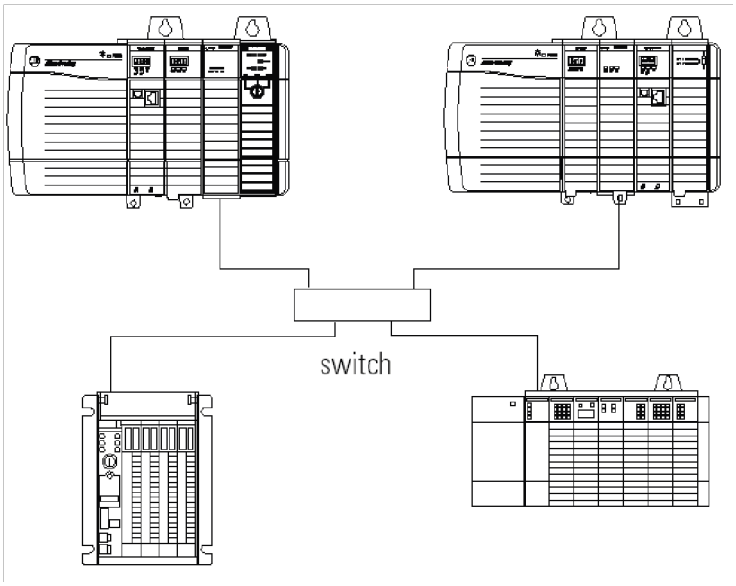
Brick style PLC Copyright © Schneider Electronics

Field devices run relatively high ‘industry standard’ voltages and currents, whereas the electronics in the PLC processor run at much lower values.

To get around this, input and output (I/O) modules are used to isolate the higher field values from the PLC internal electronics and to convert the corresponding value to a level that can be used internally in the PLC.



Rack style PLC



Brick style PLC

Unlike the small brick style PLC, the rack style PLCs shown above have individual I/O modules plugged into their various backplanes or chassis. The larger systems also allow for remote I/O chassis, which can be located, several meters away from the main or ‘local’ PLC rack, which contains the processor or CPU module. This allows for much greater versatility where further I/O modules can be added as and when required as a part of the one larger PLC system.

Let us look at these PLC hardware components and modules in a little more detail.

Input Output (I/O) devices

I/O is information representing the data that is received from sensing devices and the commands that are sent to actuating and indicating devices. The I/O system is the collection of physical elements of the control system that either provide or use I/O data.

The ‘typical’ input devices are switches and pushbuttons hard wired in control panels that the plant operators use to ‘control’ their plant along with the micro-switches and other field devices that give ‘feedback’ from the actual equipment in the process. We could cover literally hundreds of devices, but it would probably be more use to look at the types of signal that the PLC receives from these devices and how the PLC uses these signals (see I/O signal types).

Output devices are also many and varied, from simple indicator lamps to motor contactors, level controllers, valve positioners and more. Once again, we could look at hundreds of devices, but it is more appropriate to look at the types of output signal and how these are used (see I/O signal types).

There are two major types of I/O devices:

- Digital - binary devices which must be in one of only two states: on or off.
- Analogue - continuous devices – sense and respond to a range of values.

Digital I/O

Digital input devices are binary – they may be either on or off, open/closed.

Common digital field input devices include pushbuttons, limit switches, and photo-sensors.

Common digital output devices include relays, motor starters, and solenoid valves.

Analogue I/O

Analogue input devices sense continuous signals. The information that they provide is given as a continuous range of values, not just an on or off indicator.

Common analogue inputs are pressure, temperature, speed, etc.

Analogue output devices respond to a range of output values from the controller.

Common analogue output signals include motor speed, valve position, air pressure, etc.

I/O modules

I/O modules connect “real world” field devices to the controller. They convert the electrical signals used in the field devices into electronic signals that can be used by the control system and translate real world values to I/O table values.

These are the actual terminal blocks and associated electronics that the input and output devices are connected to. In the ‘brick type’ PLC, the I/O modules are built into the overall controller.

There are several ‘Speciality’ I/O modules designed to perform specific functions. Two common ‘Speciality modules’ are the high-speed timer and high-speed counter modules, which we will look at separately in more detail.

The type of input modules used by a PLC depends on the type of input device. For example, some respond to digital inputs, which are either on or off, while others respond to analogue signals. Analogue signals may represent machine or process conditions as a range of voltage or current values. The PLC input circuitry converts signals into logic signals that the CPU can use. The CPU evaluates the status of inputs, outputs, and other variables as it executes a stored program. The CPU then sends signals to update the status of outputs.

Output modules convert control signals from the CPU into digital or analogue values that can be used to control various output devices. The programming device is used to enter or change the PLCs program or to monitor or change stored values. Once entered, the program and associated variables are stored in the CPU. In addition to these basic elements, a PLC system may also incorporate an operator interface device to simplify monitoring of the machine or process.

The standard PLC module types and their descriptions are as follows:

- Digital Input – The input is an AC or DC voltage from input field device such as contacts and limit switches.
- Analog Input - The input is a variable voltage or current signal from input field devices such as temperature transducers.
- Digital Output - Provides a discrete DC or AC voltage output to control output field devices such as solenoids or relay coils.
- Analogue Output - Provides a variable voltage or current output signal to field devices such as valves or ventilation dampers.

I/O modules are available with various numbers of field device points, such as 4, 8, 16 and 32 point. Optocouplers in the modules are used to electrically isolate the module from the CPU.

Input Modules

Input modules provide the electrical connection between field devices (pushbuttons, limit switches, photo-eyes) and internal process of the PLC. They differ in voltages and types of signals produced such as on, off or a variable voltage. Figure 2 shows an example of a generic input module.

Figure 2: Input module

Output modules

Outputs are the devices that the PLC uses to send changes out to the world. These are the actuator the PLC can change to adjust or control the process - motors, lights, relays, pumps, etc. Figure 3 is an example of an output module.

Figure 3: Output Module

Some of the speciality type modules you may come across include the following;

- **ASCII** input and output modules for receiving ASCII data from weighing machines, modems and outputting to computers, displays, printers and so on.
- **RTD** and **Thermocouple** input modules for directly reading temperatures from field mounted temperature probes.
- **Servo** control modules which has both outputs to control the servomotors and inputs to read the servo position from field mounted encoders.
- **Communications** modules which allows computers and other controllers to communicate with the PLC directly. These can vary from bar code readers to smart instruments, variable speed drives and so on. They come in a variety of types and protocols such as Ethernet, Devicenet, Modbus Plus, Controlnet and Profibus.

As you can see that there are a large number of speciality I/O modules. If you come across any of these, treat them with caution, as they each have their own particular requirements and if you are not careful you could inadvertently damage a module. In these cases, always RTM (read the manual).

Power supply

PLCs require power to operate. They require a source of voltage/current to run:

- their internal operations.
- the input and output modules.
- the field devices (both input and output).

PLC power supply

The power supply is a module located in the PLC system module rack. Most PLC power supply modules provide a low voltage 24V d.c. output for powering field devices as well as the internal supply for the PLC and the I/O modules.

When we look at the simple PLCs, the power supply is set, and if you want to supply the PLC from 230V mains or from a 24-volt battery, you must buy the appropriate PLC already set up to accept the voltage input you require.

With modular PLCs, you have the option of purchasing a range of power supplies that can accept virtually any standard input voltage you could require. They also come in various current ratings to cover varying numbers of I/O racks and field devices.

Central processing unit (CPU)

The CPU controls the PLC. The function of the CPU is to store and run the PLC software programs. It also interfaces with the co-processor modules, the I/O modules, the peripheral device, and runs diagnostics. It is essentially the "brains" of the PLC.

Most people think of the whole box when they talk of the CPU, however the actual CPU is one processor chip inside the CPU module. Most modern PLCs will have additional processors that look after the Inputs and Outputs and any PLC communications.

The CPU, shown in Figure 4, contains a microprocessor, memory, and interface adapters.

The items shown inside the CPU and their basic functions are as follows:

- The microprocessor codes, decodes, and computes data.
- The memory (ROM, PROM/EEPROM/UVPROM, and RAM) stores both the control program and the data from the field devices.
- The I/O Interface adapter connects the Co-Processor Modules, the I/O Modules and the Peripheral Device to the CPU.

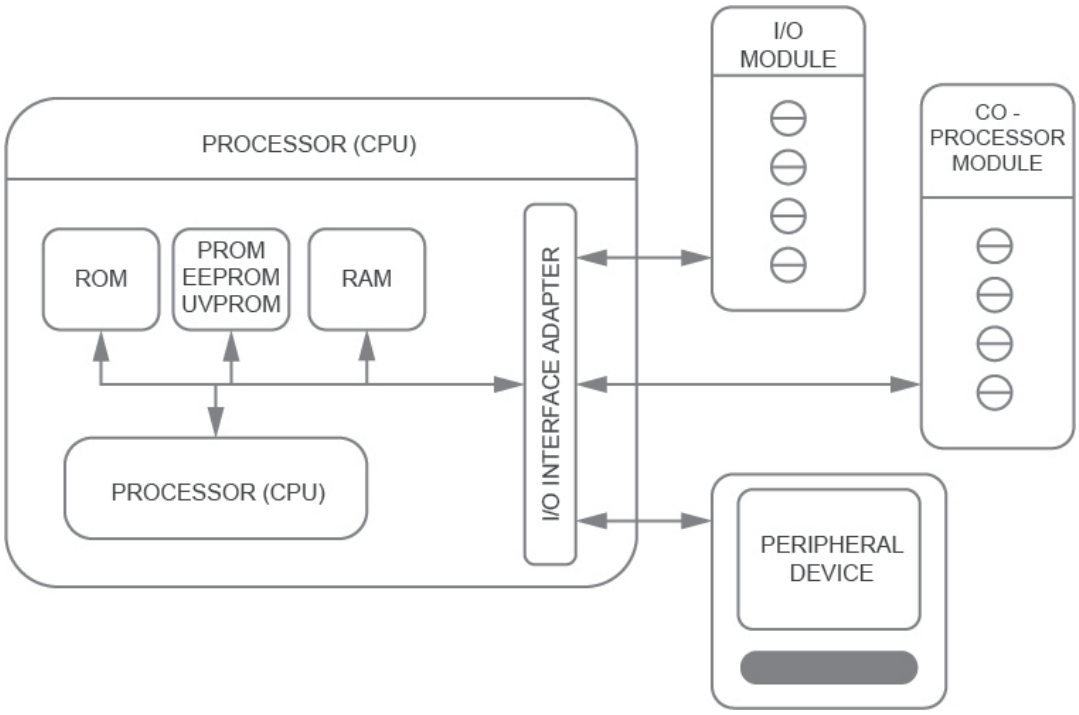


Figure 4: CPU

Input relays (contacts)

Input relays, also known as input interface, are the interface between the outside world and the PLC. They physically exist and receive signals from switches, sensors and so on. Typically, they are not relays but transistors.

Internal auxiliary relays (contacts)

Internal auxiliary relays do not receive signals from the outside world, nor do they physically exist (they are simulated in the PLC memory). They are simulated relays and are what enable a PLC to eliminate external relays. There are also some special relays that are dedicated to performing only one task. Some are always on, while some are always off. Some are on only once during power-on and are typically used for initialising data that was stored.

Counters

These again do not physically exist. They are simulated counters and they can be programmed to count pulses. Typically, these counters can count up, down or both up and down. Since they are simulated, they are limited in counting speed. Some manufacturers also include high-speed counters that are hardware based (we will look at these later). Most times, these counters can count up, down, or up and down.

Timers

Timers also do not physically exist. They come in many varieties and increments. The various types of timers include:

- on-delay (the most common type).
- off-delay.
- retentive and non-retentive.

Increments vary from 1ms through 1s. As for the counters, they are simulated and are limited in speed. Some manufacturers also include high-speed timers that are hardware based (we will also look at these later). Most manufacturers include timers that can increment, or decrement and retentive timers remember how far they have timed since they were last activated.

Output relays (coils)

These are connected to the outside world. They physically exist and send on/off signals to solenoids, lights and so on. They can be transistors, relays, or triacs depending upon the model chosen.

Data storage

The memory in a PLC can be split into three parts, namely, System, Application and Data.

Some memory can be used to store data when power is removed from the PLC. Upon power-up, it will still have the same contents as before power was removed. Very convenient and necessary!

High-speed counter modules

When PLCs run their program, they do so sequentially, either from the top rung to the bottom or by the left column to the right across the rungs. Once the PLC has completed its run through the program, it returns to where it started from, and repeats the scan. It does this continually. The time taken to complete a run through the program (or a scan) is known as the ‘scan time’.

The time taken for each program scan cycle can take from tens to several hundred milliseconds. If we rely on the PLC scan to read inputs and update counters within the program, sometimes the input can change faster than the PLC logic can read, and counts are lost (or missed).

Because of this, the majority of PLCs have what is termed a high-speed counter input or high-speed counter module that can count up to several kHz (thousands of pulses per second). Unlike the programmed counters the high-speed counter is an actual hardware device that works independently to the rest of the PLC. When these counters reach their pre-set, they operate an interrupt, which halts the PLC scan cycle and updates the relevant logic immediately. This immediate interrupt can then operate the associated output devices as soon as they need to be operated. This ensures high-speed processes can be effectively controlled by the PLC, without the need for external dedicated controllers.

High-speed timer modules

The high-speed timer is also a hardware device that operates much faster than the programmed ‘soft’ timers uses in the program. These modules can time extremely accurately and for very short time periods (as little as 1 millisecond). They also use interrupts in the same way as the high- speed counters.

I/O signal types

If we ignore the range of communications modules, which can send or receive serial data and so on, most other I/O modules fall into two categories. These are either Analogue or Digital.

Analogue I/O

Analogue I/O comes in various forms with the input or output one of two distinct types:

- Voltage.
- Current.

Some standard industrial analogue values from field instruments are 0 to 20mA, 4mA to 20mA, 1 to 5V, 0 to 10V, –10V to +10V, –20mA to +20mA.

Ultimately, the PLC doesn't care what the field signal is, as the input electronics convert the signal to a value using an analogue to digital converter that outputs a binary number equating to the signal level. This numeric value is stored in a register or memory location dedicated to that particular analogue input.

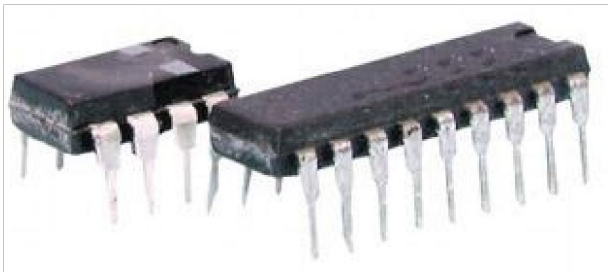
Digital I/O

Digital I/O, as for analogue I/O, comes in various forms with differing levels of input being accepted e.g. 12 or 24V ac or dc. This is converted to a 1 or a 0 in a bit or register location in memory for each input.

Each digital output is energised bit by bit from an output word dedicated to the particular output module.

PLC Memory, Types and Uses

Typical PLCs use several different types of memory for specific applications. Here we look at the various types of memory, what they are used for, and whether they are volatile or not.



Examples of memory chips

A **Volatile** memory loses its contents when it loses power, whereas **Non-Volatile** memory retains its contents regardless of whether it is powered or not.

ROM — Read Only Memory

ROM, or Read Only Memory, is designed to store information that can only be read by the PLC. This is generally only used for the operating system of the PLC itself.

The ROM chip is written in the factory and cannot be altered in the field. Once written, it cannot be changed. A new version of ROM requires replacement of the ROM chip.

PROM — Programmable Read Only Memory

PROM or Programmable Read Only Memory is a special type of ROM. The PROM can be written to only once or added to until all memory locations are full.

The programming of these chips is accomplished by using current pulses that melt fusible links within the chip itself. While some older PLCs used PROMs for permanent machine control programs, very few modern PLC would use PROM chips, as any program alterations require the PROM chips to be replaced.

EPROM — Erasable Programmable Read Only Memory

EPROM or Erasable Programmable Read Only Memory is similar in operation to the PROM, with one major difference. Shining ultraviolet light through a window in the chip case can erase the EPROM. When erased, the chip can be reprogrammed from scratch.

EPROM chips are used in a large range of devices, from dedicated controllers to personal computers to PLCs. EPROMs are programmed by use of a special interface unit that the chip is inserted into. They can be programmed then installed into PLCs for mass produced machine control, without the necessity of programming each individual PLC. In some PLCs, these were used to hold operating system software that could be updated by re-writing the chip. They are non-volatile and therefore do not require battery backup.

Because ultra-violet (UV) light erases the program, care must be taken with these chips wherever sunlight can penetrate. After the chip is programmed, a sticker is placed over the window in the chip case to prevent UV from entering it. This sticker must not be removed except for erasing or reprogramming the chip.

EEPROM — Electrically erasable programmable read only memory

EEPROM is non-volatile memory and does not require battery backup. The big advantage of EEPROM over the EPROM is that the data it contains can be electrically over-written, thus removing the necessity of using ultra-violet light to erase the chip.

Typically, the EEPROM is used to backup, store or transfer PLC programs. It can both be written to and read by the PLC, so, in effect, it is a form of non-volatile RAM.

RAM — Random access memory

RAM or Random Access Memory is designed to allow data to be written to it and read from it continually. This ‘solid state’ memory is an integrated circuit (chip or chips) or several integrated circuits that store the user program, timer and counter values, I/O status and so on.

This form of memory is considered ‘volatile’, that is, it requires power to hold the data. If you momentarily remove the power supply to RAM, the data is lost. To stop this data loss in the case of power failure, most PLCs have a battery providing power to the RAM. Typically, this battery will last anywhere from 2 to 5 years before requiring replacement.

Flash RAM

Flash RAM is a ‘Non-Volatile’ RAM and is used to store the following:

- A backup copy of the application that is reloaded into normal RAM on power up.
- Loadables, that is, instructions that are specific to specialist I/O modules that were not included in the initial operating system supplied with the PLC.
- The PLC operating system (rather than ROM) as it can be modified therefore allowing the ability to ‘upgrade’ your PLC to a new version of operating system software, without the need to replace any electronics.

Operating sequence of a PLC

Operating modes

While it varies from PLC to PLC and manufacturer to manufacturer, all PLCs will have at least two ‘modes’ they can be in: **program** mode and **run** mode.

Program mode

The program mode is when the PLC is not reading the inputs or operating the outputs but is completely under the control of the programming device. When in this mode, the application or user program can be written, modified saved, replaced or cleared altogether, and any configuration changes can be made.

Some PLCs can be programmed ‘on-line’. That is, they are actually running, (including their I/O) while their program code is being altered. A majority of modern modular (medium to high end) PLCs can be programmed on-line. This adds a whole new layer of risk to your running plant when you are making changes.

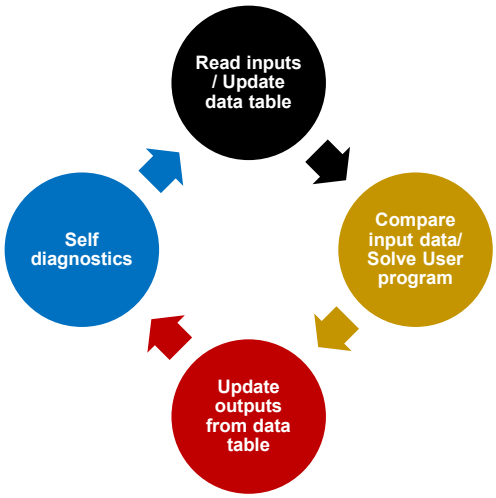
Some brands of PLC have a function called ‘Force’. This is a form of online programming which allows the programmer to override a contact or even entire rungs of program and operate outputs, regardless of the actual state of inputs and other control elements.

If you are operating a program with forces on, it is very important to take great care as it is possible to bypass safety interlocks and systems to turn on machinery, despite trips and soft controls being off. This is one of the reasons that PLCs should not be used, or relied on, for isolation of equipment for persons to work on.

Isolation for personal safety must include physically isolating the machine’s power supply as well as the PLC control.

Run mode

During the run mode, the PLC is reading the inputs, comparing the inputs with the user or application program, controlling the outputs as a result of this comparison, and performing self-checks.



While the sequence may differ slightly from PLC to PLC, all follow this general sequence of operation.

Some high-end PLCs can scan asynchronously, that is, the input and output tables are read and written independently to the Program scan. While this makes

for much faster processing, it can lead to some problems if an input changes mid-way through the scan of the program.

Now let us look at each step of the scan process, starting with the machine looking at the input status.

Step 1: Read inputs/Update data table

The input modules, whether fixed or modular, measure a voltage or current signal from some real device, and transfer the current status of each input to a memory location commonly termed the ‘input table’.

To understand this better, we should look at how these memory locations store information and how the CPU finds the particular piece of information it is looking for. The simplest analogy is to think of the memory area as a series of pigeonholes or post boxes that each can store a piece of information (a binary word — 16 or more bits). As with post office boxes, each location has a unique address that the CPU can locate, and that you can use in your program to identify each particular piece of information that you may wish to use or change.

The actual addressing labelling conventions differ from PLC manufacturer to PLC manufacturer and can also differ between PLC models from the same manufacturer.

The Input table is a particular area of the total memory that contains the current status of the real-world input signals.

The status of inputs and outputs, the elapsed time of timers, the current status of internal (software) relays and all other bits of information are stored in the ‘data’ area of memory.

The application memory is where the program resides. It doesn’t matter which programming system is used; the result is converted to data that can be read by the PLCs processor.

Step 2: Run the program (Scan or execute the programmed instructions)

The application program is run. This is done by executing instructions one at a time. In a ladder program the instruction sequence would be by scanning the logic rung by rung, from left to right and top to bottom, just like reading a book, as shown in Figure 6.

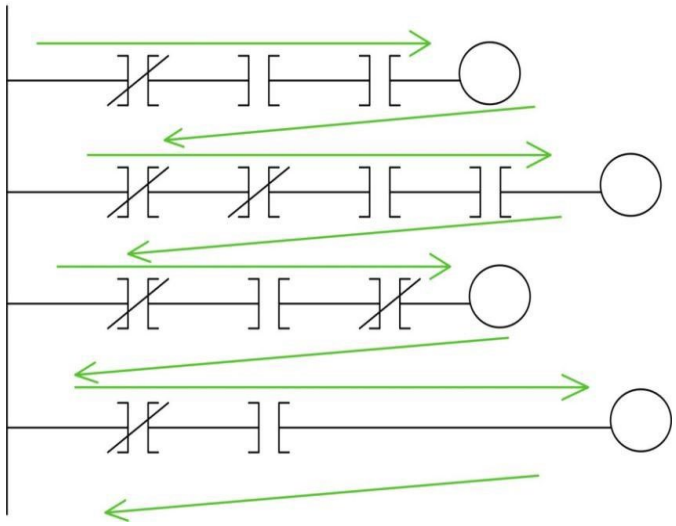


Figure 6

Unlike hard wiring where the current can flow through the wiring in any direction, in **PLC Ladder diagrams, the logical flow of current only travels from LEFT to RIGHT.**

Step 3: Update outputs from data table

The CPU then updates the data tables in memory to what is required by the user program and then updates the outputs to reflect these changes in the output data table.

Step 4: Self diagnostics and housekeeping

The internal diagnostics of the PLC then checks what has happened in that particular PLC scan to make sure the PLC hardware and software is still operating correctly.

A special timer called the **watchdog** timer checks the amount of time taken to execute the scan. If a pre-set time limit has been exceeded, the watchdog timer will fault the processor and either shut the PLC immediately, or on some more advanced models run a special fault routine/shutdown program.

Any faults with the I/O modules, or communications to remote I/O or internal problems with the PLC program or memory are recorded in the system area of memory, and appropriate action, depending on the severity of the fault, is taken.

If no faults exist or the severity of any fault is not great enough to shut down the PLC, the operational scan starts again from the beginning.

Cycle or Scan time

Typically, a scan can take from 5 to 500 milliseconds depending on the speed of the processor, the size of the machine and size of the program.

Often, the time it takes to complete a scan is a major factor in whether a PLC is up to the task for high-speed machine control.

PLC programmers

When we talk of programmers, we are not referring to the person who is writing or inputting the program.

Programmers or programming devices come in a variety of types, depending on the age of the PLC and the brand or manufacturer. We can break down the list into three relatively distinctive groups:

- Dedicated hand-held programmers.
- Software that runs on a PC or laptop.

Dedicated hand-held programmers

Early PLCs were developed with the programming device attached to the PLC. The program was keyed into the PLC, one instruction at a time, and monitored one instruction at a time.

Because the PLCs were generally located inside cabinets, the manufacturers developed versions of their programming devices that could be attached via cables so the programmers (people) could perform their programming tasks in relative comfort.

Hand-held programmers (device) now are more sophisticated and can have LCD displays that show several rungs of logic at once.

The hand-held programmer shown in Figure 10 is for the Toshiba EX series controllers. The LCD display can show several rungs of logic at once, which allows the programmer to more easily see how the logic flows.

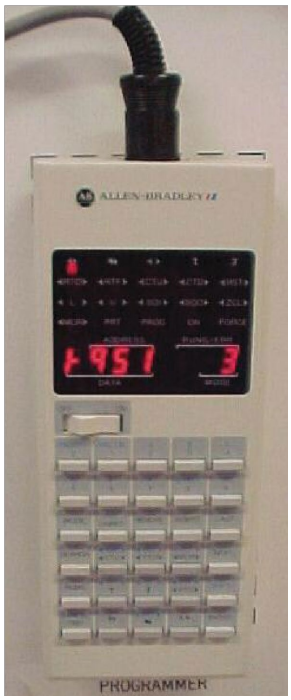


Figure 9: AB hand-held programmer

Figure 10: Hand-held programmer with LCD display

Personal computers

The most common Programmer is now the “personal computer”. This may take the form of desktop, laptop, notebook, or even smartphone devices.

A specific program of application for the PLC to be programmed must first be installed on the computer. Most of these have a graphical user interface in which one can set up the names and addresses of the I/O, and program the PLC using multiple methods or languages.

Programming Methods

IEC 61131-3 is the international electrical standard for PLC programming methods which specifies syntax and semantics of programming languages for programmable controllers as defined in part 1 of IEC 61131. This allows systems integrators, engineers, and maintenance personnel to be able to follow PLC code, regardless of the manufacturer.

However, IEC 61131-3 is a manufacturer's standard, not an end user's standard, therefore, each PLC manufacturer implements it differently. The look and feel of the finished code, and how we read it when fault finding a process, is very similar. How we write the code within the individual programming software is dramatically different between manufacturers.

Also, many programmers aren't formally trained in the standard and many don't even know it exists.

Each programmer develops his own style and methods for programming, just as computer programmers do.

The most common programming methods or languages are:

- Ladder logic or Ladder diagram.
- Symbolic block or Function block diagram.
- Sequential function chart.
- Structured text.
- Instruction list.

Let's look at these methods in detail.

Ladder logic

Ladder logic programming was developed as replacement for relay logic. Today, this is the most common of the PLC Programming methods. The diagram looks like a wiring schematic for a relay circuit with the power line on the left and the outputs on the right. This method is particularly suited to machine control and is the main programming method for PLCs in industrial controls. Math instruction while available can be very complex and/or difficult to program.

It is referred to as a ladder diagram because when you look at it, it looks like a ladder with the inputs and outputs of the program contained on each rung.

In the example on the following page, we can see the ease of converting from the hard-wired relay circuit for a Direct Online (DOL) motor starter, to an equivalent PLC 'Ladder Logic' diagram, or program.

Figure 12 shows a hard-wired logic diagram. Let us see how this can be converted to a PLC Ladder logic program.

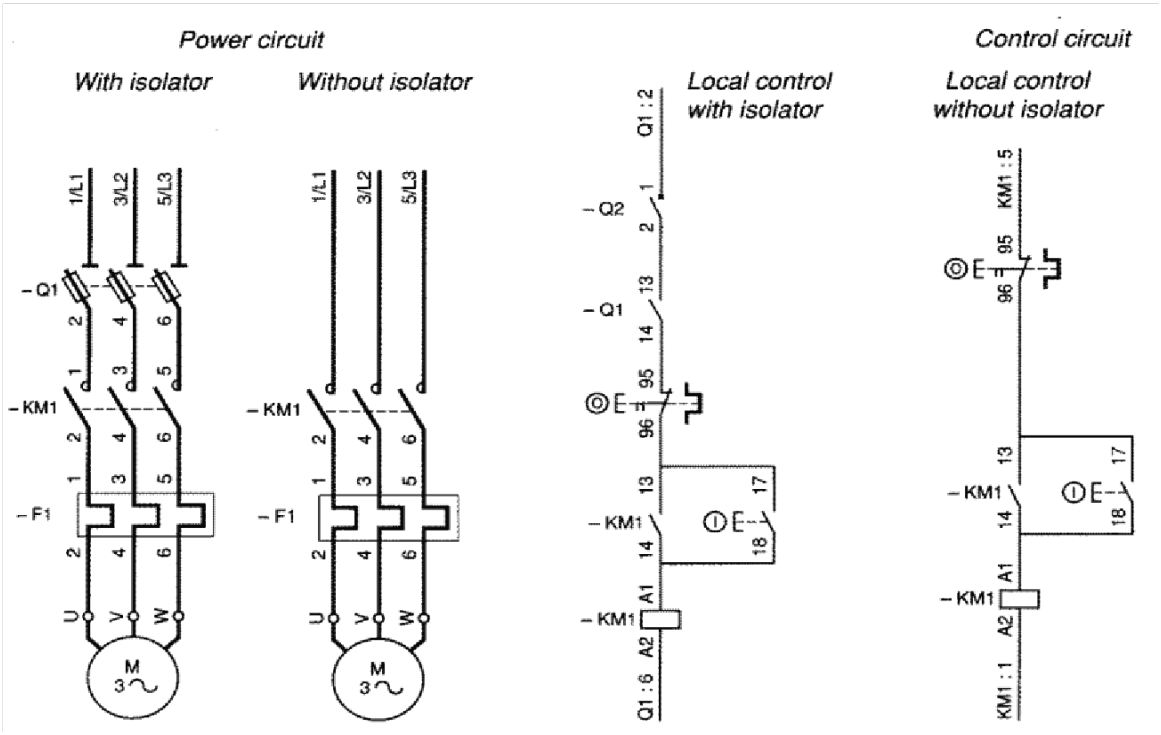


Figure 12: Hard-wired logic diagram courtesy of Groupe Schneider from their book 'Practical Aspects of Industrial Control Technology' Telemecanique Technical Collection.

For ease of understanding, the control section of the above diagram has been turned 90 degrees and had the output coil shifted to the right-hand end, to follow PLC convention (Figure 13).

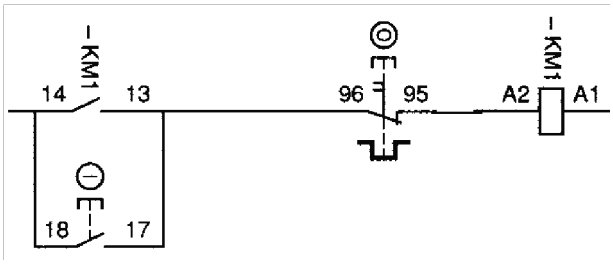


Figure 13: Control section turned 90 degrees

Each element is labelled to identify which device it represents, along with the terminal numbers to connect the wiring to when building the device.

In this example, the stop button and thermal overload trip are combined in one device and is shown N/C (Normally Closed), which is the normal convention.

When converting this to a PLC ladder logic program, you must look at things a little differently. As you will recall, the Input modules on the PLC are monitored to see what the current state of the real-world inputs actually are.

To understand how this affects the way we program, you should consider the actual input as the coil of a relay. When you power the coil, any N/O (normally open) contacts will close and any N/C contacts will open.

The N/O contact is sometimes referred to as XIC (examine if closed). When the input is live, the ‘PLC input relay’ is active and all N/O contacts associated with it will be closed. This leaves the stop button wired as per usual; normally closed represented in the logic as a normally open contact (it will be active as the input is energised).

The N/C contact we refer to as XIO (examine if open). When the input is NOT live, the ‘PLC input relay’ is not active and all N/C contacts associated with it will be closed.

Subsequently, the equivalent PLC logic for the above starter would be as shown in Figure 14.

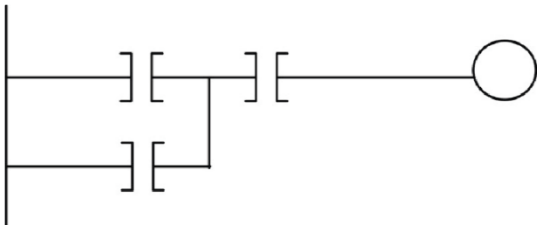


Figure 14: Equivalent PLC logic

However, when we write a PLC ladder program, the PLC must know which inputs and outputs you are referring to in your diagram. This means we must assign addresses to each element of the ladder diagram. An example is shown in Figure 15.

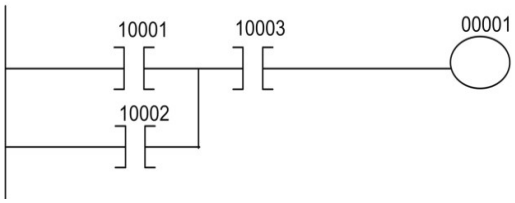


Figure 15: Ladder diagram with addresses assigned

The numbering convention used depends on the PLC.

This addressing, forms part of your as-built documentation as you can print the ladder listing whenever you wish.

Additional documentation can also be added to show what each instruction and rung of logic is there for.

In this case, the addresses used relate to the inputs and outputs as shown below.

10001	Start Button Input to PLC Input module (address 10001)
10002	Hold in contact from contactor KM1 to PLC Input module (address 10002)
10003	Stop Button Input to PLC Input module (address 10003)
00001	Output from PLC Output module (Address 00001) to Contactor KM1 coil

Symbolic block or Function block diagram

In some instances, the boundaries between what we would consider a PLC and other forms of controller such as DCSs (Distributed Control Systems) are becoming very blurred and the symbolic block methods of programming commonly used in DCS systems are also crossing from one platform to the others. Symbolic block programming can make programming very simple.

Symbolic block diagram is essentially a graphical method of programming a PLC by connecting a few or many blocks together to achieve your desired process. It consists of blocks for each function that show the inputs and outputs for more complex sequences and lines drawn between each block illustrating what each output will do and what will affect each input. For example, you may have a scale in your process and if you want an alarm to sound if the weight measured on the scale is too high or too low, then the scale will have a box with the line drawn from the weight output to the variable input of the alarm box. The output of the alarm box for either the too high or too low alarm will go to an alarm horn and/or light.

With many elementary blocks available you can build virtually any control scenario you can imagine. However, some PLC vendors also allow you to build your own function blocks to create specialist functions where it is better to ‘black box’ a piece of code for re-use, or to create a special function that would otherwise take up a lot of code.

Function block is well suited to analogue control with the relationships between analogue blocks easy to follow.

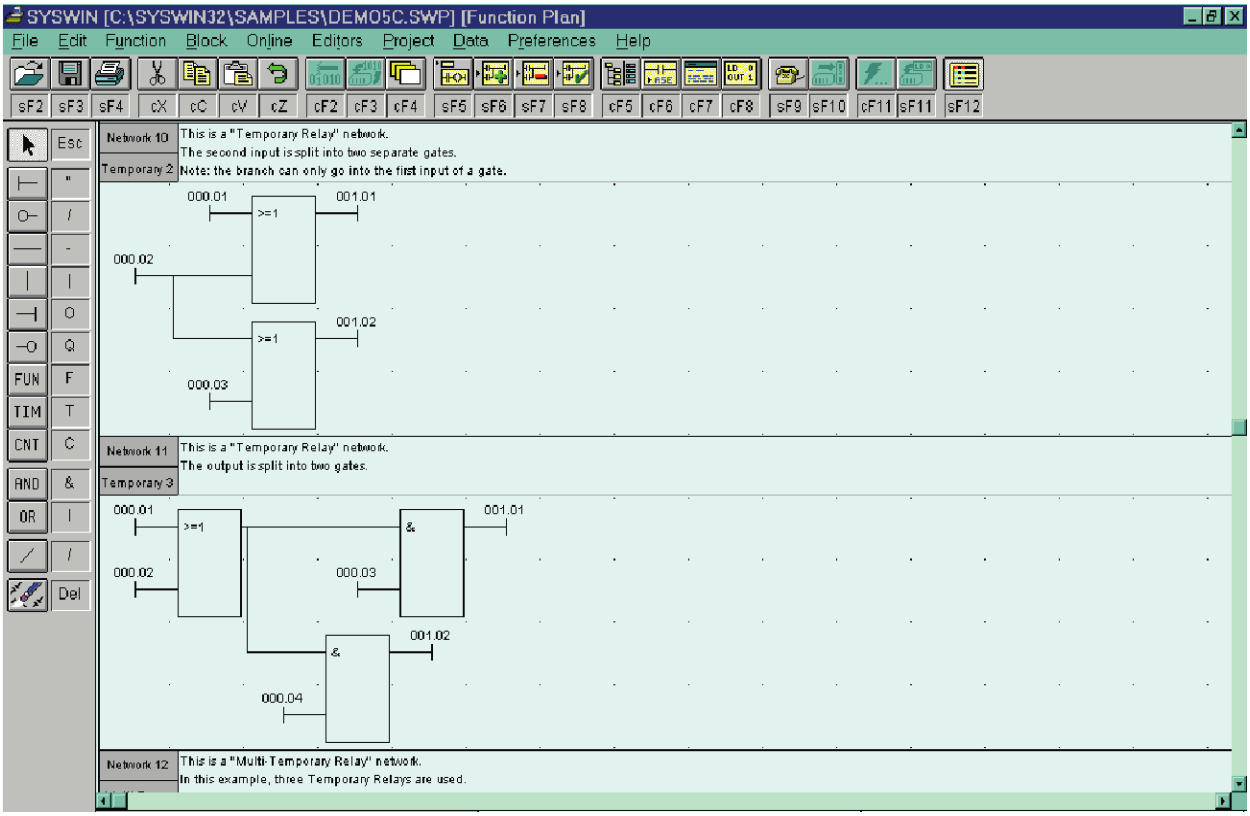
There are many variations but for the purpose of this unit, we will look at one variant of the symbolic block form of programming.

Function Plan

Often drawn as logic gates, the same way as they would be used in designing a dedicated (hard-wired) logic circuit.

This method of programming shows IEC617 compatible logic symbols.

The following example is a screenshot taken from OMRON's, Syswin programming software.



Sequential Function Chart

The sequential function chart method is another pictorial method. It most closely resembles a flow chart, only it's more complex. In the SFC (sequential function chart) sequence language, a section is split into single configured sequential steps, through steps and transitions, which alternate in the sequence plan.

There are three primary elements in a sequential function chart: steps, actions and transitions. Each step contains the logic for a particular portion of the process.

The sequential function chart is simply built from program states or steps and transitions to get from one step to the next. When a step is active, its associated outputs and flags are active. If a transition to another step becomes 'true', then that new step becomes active (current) and the previous one will now become inactive.

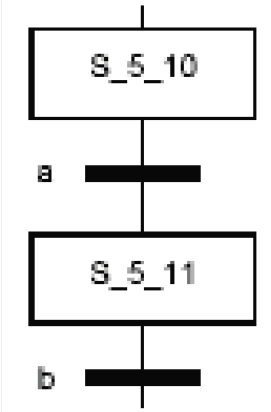


Figure 16: Sequential function chart

In Figure 16 process of S_5_10 to S_5_11 only takes place, if step 5_10 is in an active state and the condition for transition 'a' is true.

As an example: weighing an item, checking for alarms and sounding the alarm if the weight is out of limits. The actions are the individual activities of performing the steps. Transitions move the process from one step to the next.

Structured Text

Structured text is a text language and is not used often with PLCs, though many manufacturers do allow for this within their PLCs' programming software. It is very similar to Pascal or BASIC, and for people trained with computer programming, it can be the easiest. Complex math, decision-making processes and looping code are often easier to accomplish with structured text as it can be done on one page versus many rungs of a ladder diagram. It uses many of the same programming constructs such as FOR Loops, Case statements and IF statements. It also contains many of the high-level mathematical instructions in the same or similar format.

The majority of high-level languages (programming languages that resemble human language) are specifically written for programming personnel and mainframe computers. In older PLCs, higher-level languages were confined to speciality modules such as BASIC modules, ASCII modules, and co-processors and similar. With the advent of IEC1131-3, Structured Text has been specifically written for use in PLC programming.

Instruction List

An instruction list, as you would expect from the name, is composed of a list or series of instructions, which are executed in the order they appear in the list.

Each ‘instruction’ begins on a new line and consists of:

- An Operator (if necessary, with modifier).
- If necessary, one or more operands.
- A comment can follow the instruction.

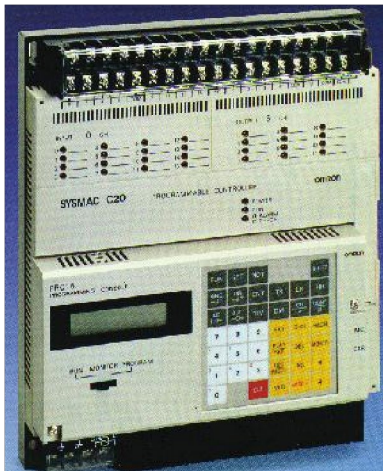
Tag	Operators	Operands	Comments
START:	LD	A	(* Keyboard 1 *)
	ANDN	B	(* AND keyboard 2 *)
	ST	C	(* Ventilator on *)

(In this case the AND Operator has the Modifier N ‘Negate or Not’ assigned to it.)

As with all languages, you need to understand the syntax and what instructions are available to you, along with how they operate.

The instruction list method is probably the most complicated method, as it most closely resembles Assembly language. This can be useful for processes that repeat a small function often. Though it is a powerful method, it is often easier to just program the process in a ladder diagram than it is to learn how to program with an instruction list.

This programming method came about mainly by the need to create some form of programming that was cheap and easily transportable.



Most manufacturers had a dedicated programmer/loader, or a keypad attached to the front of their controllers. These were very expensive and could not safely be used in many industrial environments.

They usually had a one-line text display that had to be interpreted for each instruction entered into your program. One example of this style of programming device is the Omron PRO15 shown in Figure 17 as attached to the front of a SYSMAC C20 PLC.

Later versions attached via a short flexible cable.

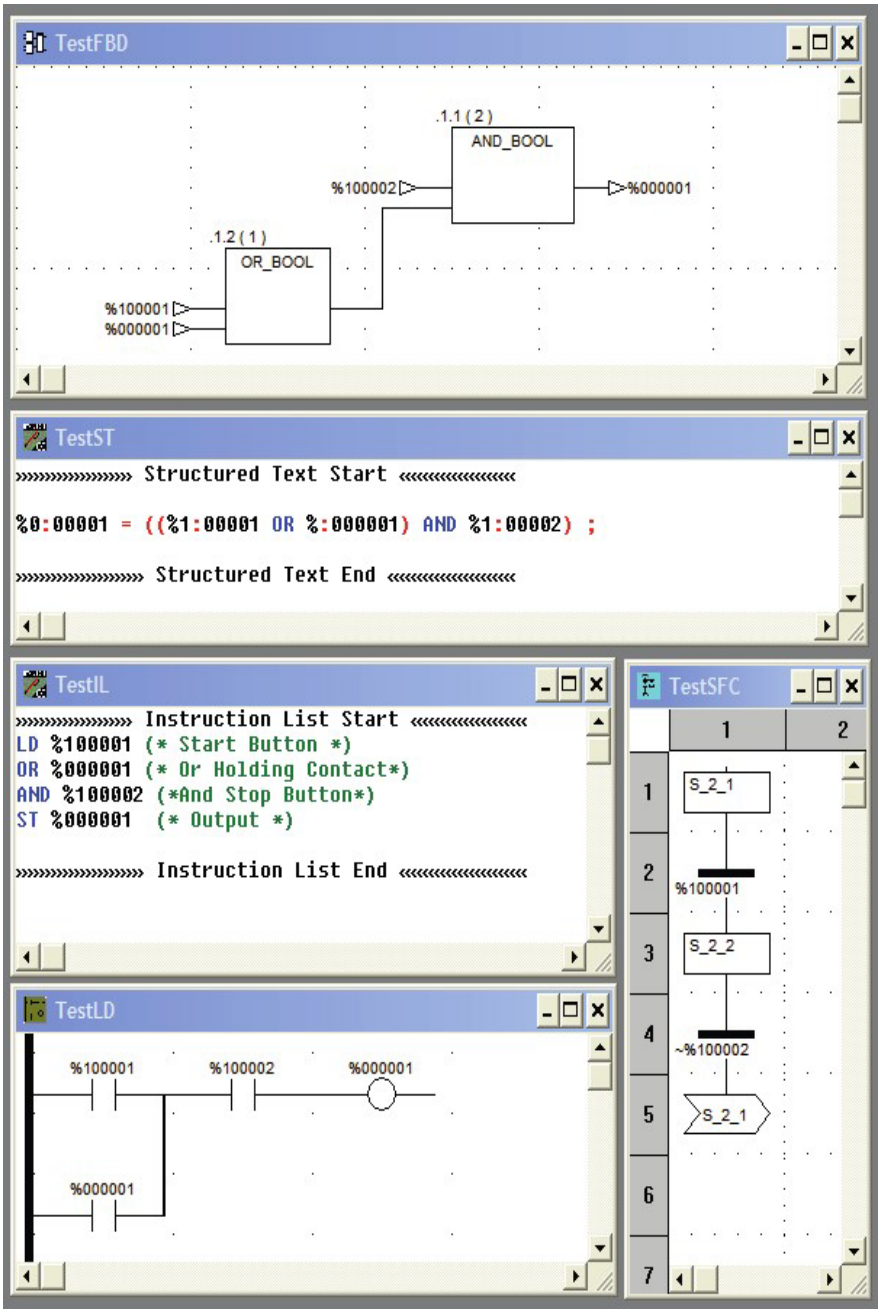
Figure 17: Omron PRO15

With the demand for technology where companies could do their own programming with a simple transportable programming terminal, smaller and dedicated programming devices with sealed keypads and LED displays were developed.

IEC program examples

The IEC Code below shows the same simple stop start circuit using all five IEC languages we have just discussed. For this piece of code, you will see that certain languages lend themselves better than others to easily follow this code.

Function Block diagram



Structured Text

Instruction List

PLC Terms

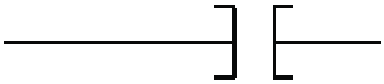
When we start looking at PLC terms, we can go straight to the glossaries found in the PLC manuals from the many PLC vendors out there. From these, we could very easily create a list of several hundred terms and variations. Many of the terms used are vendor specific, so we will examine the most important and common terms now.

We have deliberately avoided the more advanced instruction terms, and while several of these terms can be used across the different IEC programming languages, we are looking at Ladder Logic only.

Bit level functionality

Where there are multiple names for an instruction/term, these terms all mean the same thing. They are simply the names used by the manufacturers for that instruction.

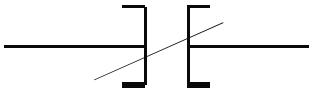
Examine On, XIC, Normally Open



When ‘True, ON or Closed’ (Logic or Bit value = 1) the instruction passes power.

The PLC equivalent of a normally open relay contact.

Examine Off, XIO, Normally Closed, Not

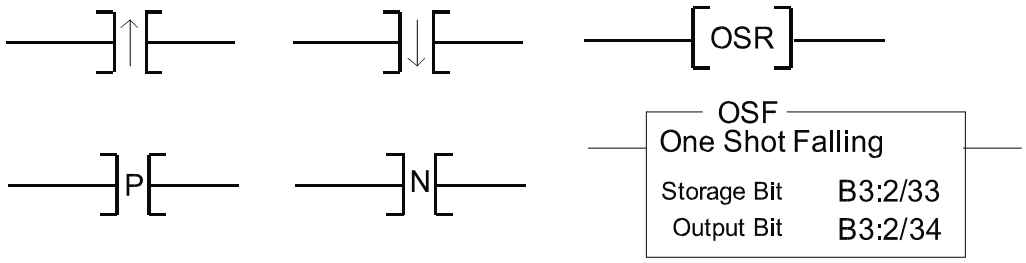


When ‘False, OFF or Open’ (Logic or Bit value = 0) the instruction passes power.

The PLC equivalent of a normally closed relay contact.

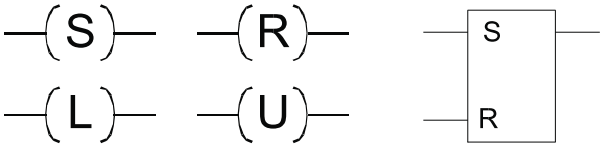
Transitional contacts

A transitional or One-Shot contact varies between manufacturers. Some use a contact off any coil with a Positive or Negative transitional contact as shown by an arrow pointing either up or down, or a P or an N to designate positive or negative. Others use a special instruction with its own unique address for the Positive ‘One Shot Rising’, and often an instruction with 2-bit addresses for the negative or ‘One Shot Falling’.



Regardless of the configuration, the transitional contact is active for ONE scan of the PLC. Positive is active for 1 scan when the contact or instruction transitions from OFF to ON (the rising edge) and the Negative is active for 1 scan when the contact or instruction transitions from ON to OFF (the falling edge).

Set and Reset



Set/Reset Coils (Latch/Unlatch) are often used where a flag or output needs to be turned on (Set — Energised) by one piece of logic and turned off (reset — de-energised) by a separate piece of code.

You can place the Set or Latch coil in one rung of logic and when the coil is active it ‘Sets’ the bit in the PLC memory. The bit stays ‘Set’ until such time as the ‘Reset’ coil is activated when the bit is ‘Reset’.

Some PLCs have a Set/Reset instruction or the Opposite Reset/Set, which has an output that we use as the controlled Bit. These RS or SR blocks are often known as Flip/Flops. Where both inputs are energised, the precedence goes to the first in the name, that is, an RS will be reset where an SR will be set.

Word level functionality

As we have seen previously, these instructions can vary considerably from manufacturer to manufacturer.

Move or block move

A Move instruction (MOV) takes the value in a source word and moves it to a destination word. It is often used with a transitional in step control logic.

Block Move (BLKM) or Copy (COP) instructions perform much the same way with the addition of a length or number of words moved or copied.

Test or compare

Some PLCs have a single TEST block with 3 outputs:

(A > B, A = B, A < B).

Others have a range of instructions or blocks, as follows:

- (GRT) greater than.
- (GEQ) greater than or equal to.
- (EQU) equal to.
- (LEQ) less than or equal to.
- (LES) less than.

Higher end PLCs, often have more advanced compare instructions or blocks where you can define more complex relationships involving more than two variables.

Simple math

Almost all PLCs available today will have some mathematical capability.

Simple ADD, SUB, MUL and DIV blocks are used very commonly to process analogue values.

More advanced math functions such as SQRT, (square root), exponentiations and so on are commonplace in medium level PLCs and above.

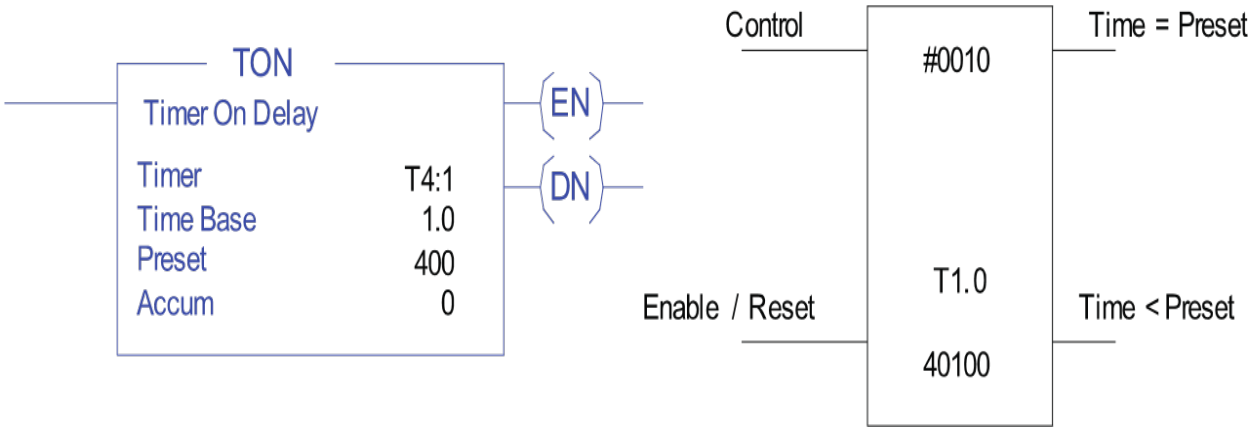
Special functionality timing

Timing

All PLCs use timers such as On delay, Off delay, and retentive or accumulative timing.

Simple PLC timers generally have a timing input, a preset, and an output which energises when the ‘Accumulated value’ = the ‘Preset value’.

They will often be reset when power is removed from the timer or may have a separate reset coil associated with them.



The two-timer instructions shown above are from Allen Bradley and MODICON.

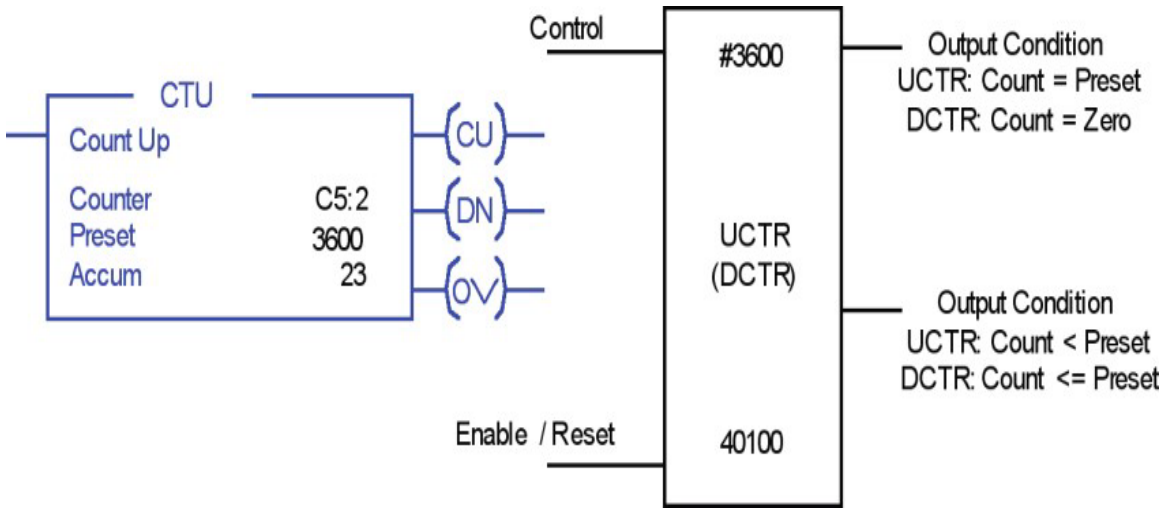
The AB Timer TON is an On Delay only and have three output bits (EN Enabled, TT Timer Timing, and DN Done) available along with the Preset and Accumulator values that can be used elsewhere in your logic.

The MODICON Timer has two inputs and two outputs along with a timer register that holds the accumulated value. The Preset in this case is shown as a fixed value, but it can be a register address allowing you to use it elsewhere in your program, as was the case with the AB timer.

The Time base in each case is set at 1.0 Seconds. Depending on the PLC, this could be set to 0.1, 0.01 or even 0.001 second increments.

Counting

As with timers, counters are very common in PLCs. They are often used in industrial processes and can be Count up, Count down, or even count up and down. Counters are also commonly used to count timed pulses, where a timer is set up as self-resetting, so it produces a one scan pulse every second or minute (or any other desired period), and the pulses are counted to accumulate a large time period.



The two counter instructions shown above are from Allen Bradley and MODICON.

The AB Counter (CTU) is an Up Counter only and has three output bits (CU Enabled, OV Overflow, and DN Done) available along with the Preset and Accumulator values that can be used elsewhere in your logic.

The MODICON Counter as with the timer has 2 inputs and 2 outputs along with a Counter register that holds the accumulated value and a configuration mnemonic (UCTR/DCTR), which designates either up or down counting. The Preset in this case is shown as a fixed value (#3600), but it can be a register address allowing you to use it elsewhere in your program, as is the case with the AB counter.

You can also construct counters by using the PLC ADD and SUB instructions.

Flags

When we think of flags, we usually think of our national flag or maybe the flag a touch judge uses during a game of rugby or football. Either way, the flag is seen to represent something specific, such as New Zealand, or the ball is out of play and where it went out. We can often get even more information from a flag when we look at it, such as which way the wind is blowing, and possibly how strong it is blowing.

In PLC logic, a flag performs much the same function. It tells us the state of a piece of Logic or a piece of plant or our process, to allow us to decide whether we need to perform an action.

Internal relays and coils as flags

When we need to signal an alarm because a part of our process is not in a safe condition or a bin is full or a temperature is low or high or a box has been detected on a conveyor or we can energise an internal coil and use the state of that coil to ‘Flag’ the state.

Similarly, we can Latch a coil to say that a state occurred in the past.

These ‘Flags’ can be used both in the normal operation of your process or in fault-finding.

Internal registers as flags

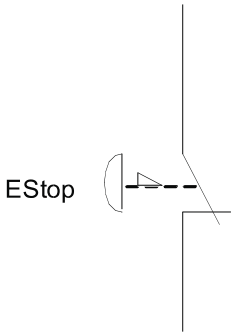
When a value is varying, such as the wind direction and strength around a tower crane, we will often use an analogue ‘Flag’ to show us what the current state is, or as previously stated, we can capture these states for future observation.

We may even trend the state over time to build up a picture of how a process is performing.

Emergency Stop Logic

Hard-wired emergency stop

In a hard-wired system, an Emergency Stop (EStop) is always wired ‘Normally Closed’. This is to ensure fail-safe operation, that is, if the wiring fails (a wire is broken), the failure leaves the device in a ‘Safe’ state. With most modern machines and processes, the EStop buttons around the plant are wired to EStop relays, which cut control to the process, until such time as you physically reset the stop button and reset the EStop relay.



PLC emergency stop considerations

With PLC Emergency Stop control, you must consider two things:

1. Is this Emergency Stop required to meet any specific SIL (Safety Integrity Level) rating?

   The SIL rating system has four levels based on the likelihood of an event times the severity of the resulting injury/damage, to persons, the environment and, finally, the plant.

   For our purposes, we will consider an SIL rating of 1 (low risk, low impact). This level most likely does not require a Safety PLC or Safety relay but will require the standard PLC and control system to immediately stop the running plant.

2. What do I need to stop and how?

   Under ‘normal’ plant operation, we generally stop the plant using the PLC logic and/or remote stop buttons.

   For a lot of plant, we will do exactly the same for an emergency stop. The exception is often where high inertia loads or processes need to dissipate the stored or kinetic energy in a hurry.

   Here, we will need to apply specific emergency stop code to apply brakes, dump compressed air or hydraulic pressure, and so on, or lock machinery in a safe position.

Simple PLC emergency stop application

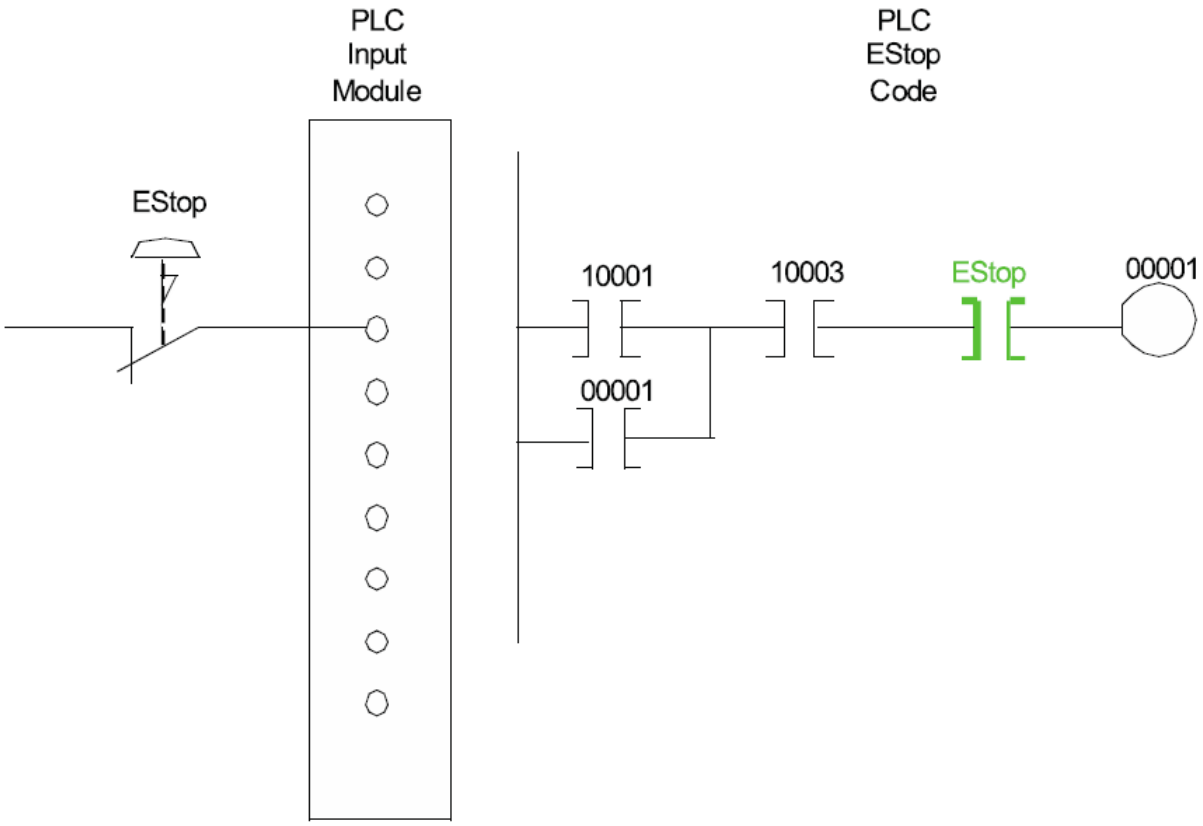


Figure 19: PLC emergency stop

In the example in Figure 19, you can see the ‘Normally Closed’ Emergency Stop push button is wired to a standard Discrete (or Digital) input module.

The code is programmed with a Normally Open or XIC instruction, which is Active as the EStop input is energised.

Pressing the EStop button will de-energise the input, and de-activate the EStop Normally Open contact in the PLC code.

Review questions

1. TRUE or FALSE

a. Compared to relay and hard-wired logic, PLCs can have many times more contacts for each ‘soft’ relay.

b. The two major types of I/O devices are Digital and Analogue.

c. PLCs typically scan the logic left to right from top to bottom like reading a book.

d. PLCs normally operate in a fixed sequence or scan.

e. The Programmer is the device you use to enter your user program into the PLC Memory.

f. All PLC elements must have specific addresses to identify their memory locations for processing.

g. ALL Emergency Stop circuits should use normally closed Emergency Stop buttons, which you program with normally open contacts, and should self-latch so the equipment cannot restart just by releasing the Emergency Stop button.

2. Match the columns:

a. All about memory

ROM (Read Only Memory)	Non-volatile form of RAM, used to back up user programmes.
PROM (Programmable Read Only Memory)	Non-volatile, used for mass produced program code. Can be erased and reprogrammed in a special interface unit.
EPROM (Erasable Programmable Read Only Memory)	Non-volatile, used for mass produced programs and program backups. Can be erased and reprogrammed or altered.
EEPROM (Electrically Erasable Programmable Read Only Memory)	Non-volatile, used in older PLCs for non-changeable programmes. Cannot be erased but can be added to until the memory is full.
RAM (Random Access Memory)	Volatile, used for the user programme, the I/O tables and running code in the PLC. Can be read, written to and altered quickly.
Flash RAM	Non-volatile, used for the PLC operating system. Cannot be erased or altered.

b. Programming methods

Instruction List	A graphical representation of the code that is scanned and processed in the block priority order from inputs to outputs.
Function Block	A graphical representation of the code that very much follows the same constructs as an electrical wiring diagram — with the exception that the ‘power flow’ is only from left to right and vertical.
Ladder Diagram	A high level textual-based programming language that supports, constructs and processes very similar to ‘high level’ computer programming languages.
Sequential Function Chart	A list of instructions that are processed one at a time.
Structured Text	A graphical representation of your code based on steps and transitions between steps.

(For answers, go to the section titled ‘Answers to review questions’ at the end of this book.)

Part 2: Design, Write and Store a PLC Program

Before you can build a house, you need to have a plan of exactly how it will be put together.

Before you bake a cake, you need a recipe showing exactly what ingredients are required and how they are mixed, what baking tin is required, and what temperature it needs to be baked at, and for how long.

When we start PLC programming, most beginners will try to get right in there, and start putting code together. This invariably ends in disaster. Either it takes an extraordinary long time to achieve anything, or the code never quite makes the grade, or someone else is called in to unravel the spaghetti and get the code working.

To avoid the pitfalls, we first need to have a plan to get our process to run smoothly.

In PLC terms, we usually have the following two documents ready before we start:

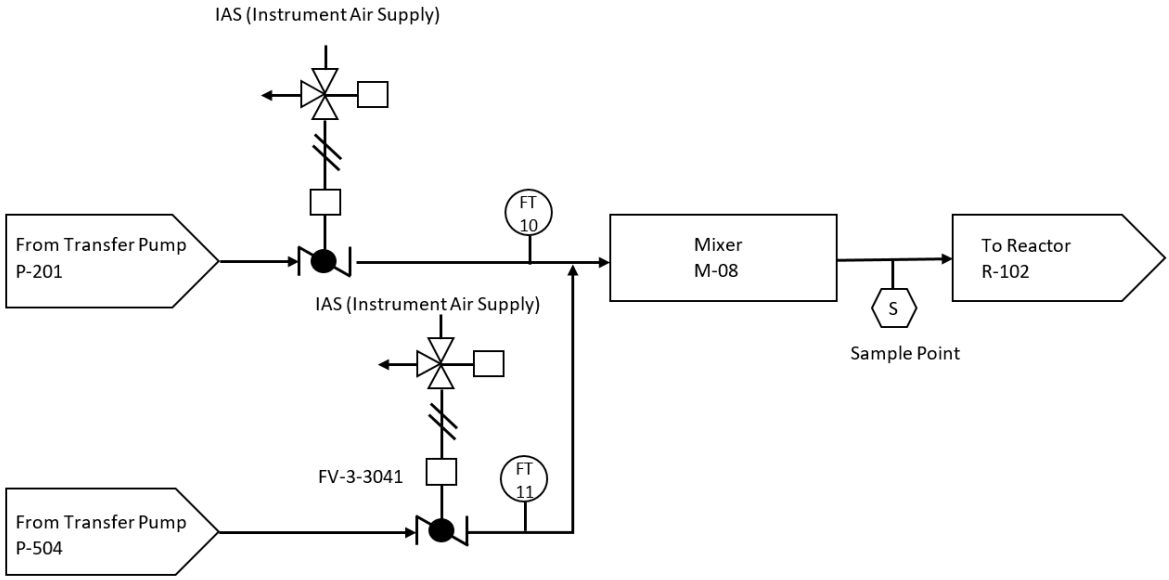
1. **Process and Instrumentation Diagram,** commonly referred to as the **P&ID.**
2. **Function Description or FD** that is developed based on the P&ID.

When we have these (the Plan and the Recipe) we can then confidently code the PLC with a very high likelihood that we will be successful, with very little need for follow up work.

Let’s look at these two documents in detail.

Process and instrumentation diagram (P&ID)

The P&ID when used in fluid processes is sometimes called a Piping and Instrumentation Diagram.



The P&ID example (above) has two inflows from transfer pumps, being controlled by air actuated butterfly valves, which are monitored by flow transmitters (FT10 and FT11) before being mixed and forwarded to a reactor. It also shows a sample point prior to the reactor.

When we look at mechanical processes, it is often more appropriate to look at the mechanical/construction drawings for instrumentation such as limit switches and photo eyes and to create a flow chart of the machine operation. We usually create an FD (function diagram) for the installation. An FD is a form of flow chart, which we will discuss very soon.

This system for developing FDs is based on the French '**Grafcet'** method of process control flow-charting. The same standard has been adopted by the IEC (with some minor modifications) as an international standard for flowcharting process control systems. The equivalent joint New Zealand Australian standard is AS/NZS 4382:1996 Preparation of Function Charts for Control Systems.

This Standard allows process control engineers and plant designers to specify the process flow of a plant or piece of equipment in a manner that can then be used to program the PLC without major re-writing of the information.

Function description (FD)

The following example is taken from Programmable Automation Systems (1990 edition) published by CITEF (ISBN 2-907314-00-9).

The example shows a simple automated drilling machine and the various function charts associated with it.

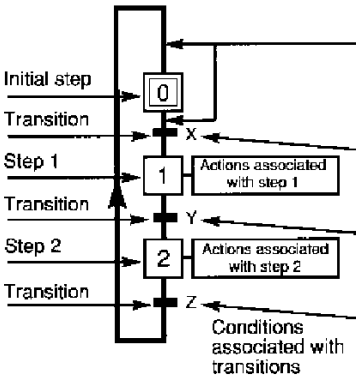
■ FUNCTION CHART (GRAFCET) in the evolution of automated system representation

Timing diagrams, phase diagrams, PETRI networks and flow charts are graphic languages that have been used to represent automated systems. Each language has contributed in building the basis for the development of Function Charts.

Now a standard, Function Chart is recognized as the graphic language best suited to representing the sequential part of automated production systems.

Function Chart represents the sequence of steps in the cycle. Step-by-step progression of the cycle is controlled by "transitions" located between each step. Each step may consist of one or more actions. A "transition condition" is associated with each transition. This condition must be satisfied to allow the transition to be cleared and allow progress to the next step.

The cycle advances step by step. The initial step (step 0 in the figure on the right), is activated at the beginning of operation and enables the transition which follows. It is cleared when transition condition x is true. Step 1 is then activated and step 0 deactivated. Actions associated with step 1 will then take place until the transition conditions of the next transition are satisfied.



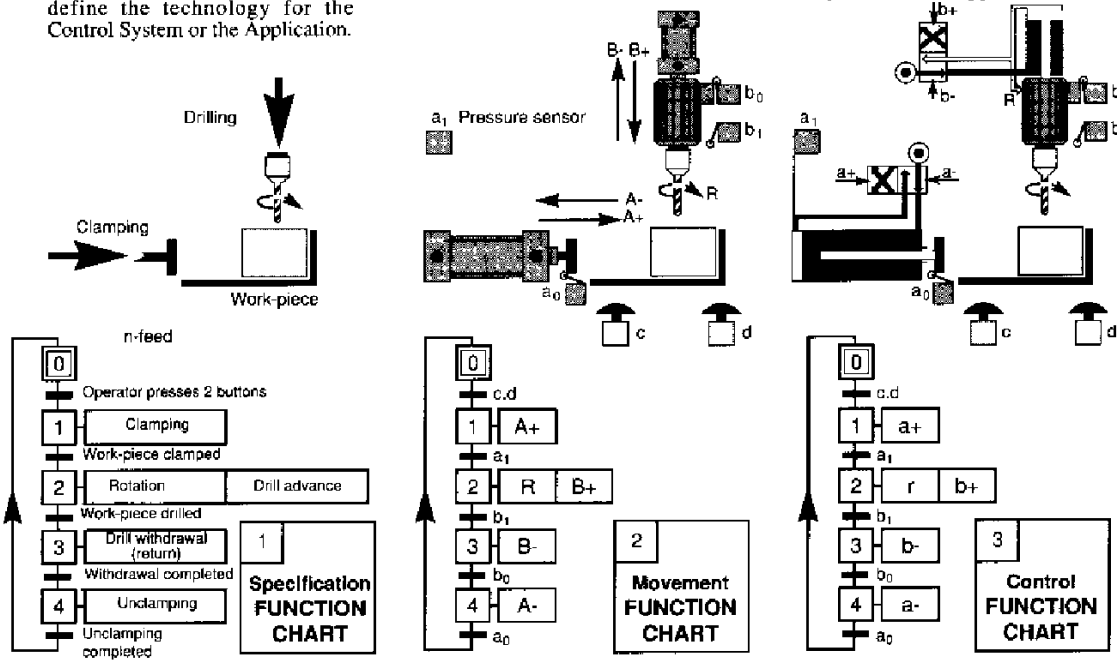
■ FUNCTION CHART, based on acquired experience

The example on page 14 is described at various stages of its design by the FUNCTION CHART below :

1 - In the initial specification the FUNCTION CHART does not define the technology for the Control System or the Application.

2 - Once actuators and sensors, have been selected, the "movement FUNCTION CHART" defines the actions and transitions.

3 - Lastly, when the preactuators have been selected, the "control FUNCTION CHART" represents the signals exchanged between the Control System and the Application.



From GRAFCET to the IEC (1) Standardized FUNCTION CHART

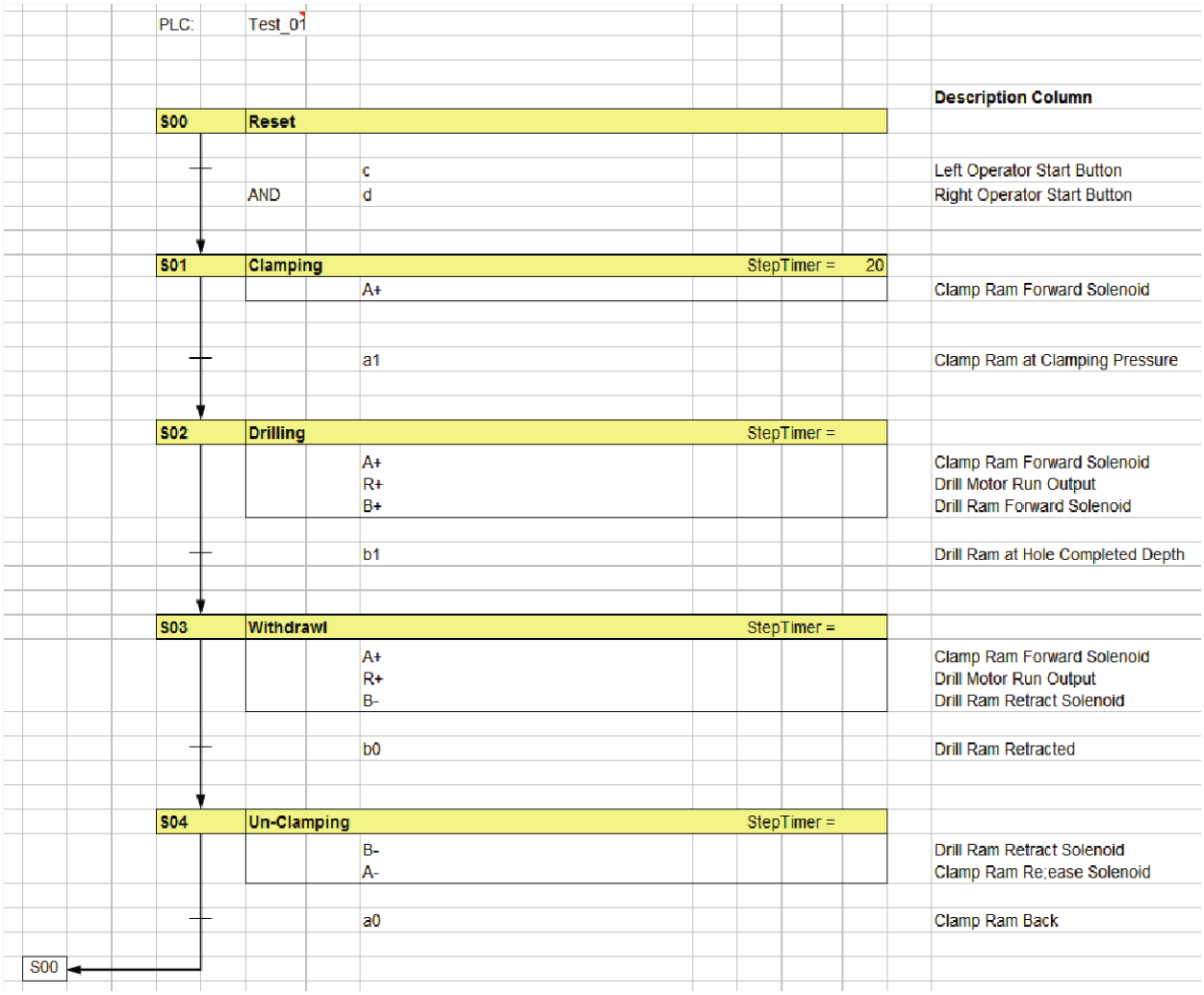
GRAFCET - a French coined word, is quickly establishing itself as a very powerful language in the practice of automation.
The IEC 848 international standard, published in 1988, which refers to GRAFCET as FUNCTION CHART or FC and defines the rules.

(1) IEC : International Electrotechnical Commission.

In the example above, we can see the process or specification is defined first.

This specification is expanded to show the movements of the machine components.

Lastly, the control functions for each step and transition are defined ready to be coded.



From these process flow charts or graphical Function Descriptions and the P&I Drawings shown above them, we can see what Inputs and Outputs are required, and what the operational sequence requires on a step-by-step basis. When the design engineer is satisfied as to the accuracy of the ‘FD’ in place, it becomes a simple exercise to code the PLC so the machine performs exactly as the designer intended first time around.

Above is an actual FD created for the drilling machine example detailed in the FD description. Note, we have used the same tag names as used in the grafcet.

Writing Your Program

Before you start writing code, you should document all the inputs, outputs and internal coils and registers you know you are going to use.

This allows you to visually check you are using the correct bits as you add them to your code, saving you lots of time and effort trying to check your code later, if you don't get it right.

Programming the sequence

From your flow chart (FC), you know we have 5 steps.

1. Reset.
2. Clamping.
3. Withdrawal.
4. Retracting.
5. Unclamping.

You also know what is required to get from one step to the next.

Although there are lots of different ways you can do it, this is what you should code first.

When you have this coded, you will be able to test the operating sequence without operating any actual outputs, proving your operational sequence and, if required, making any adjustments to get it right first time out.

Error checking — What IF?

When you have the operating sequence correct, you should look at 'What IF' scenarios.

- What if the clamp ram doesn't clamp, or unclamp?
- What if the drill motor doesn't run?
- What if the drill ram doesn't advance or retract?

In each of these cases, we need to consider the consequences. Do we need to allow for these errors in our code and automatically take some action, or do we just wait and let the operator intervene?

Let us look at the first couple of steps.

Reset

With the machine at Reset, the operator places a piece of work to be drilled in the machine, and then presses the two start buttons. If the work piece is not clamped, the operator will see it, and react so we probably don't need to add anything to the code for this error.

Clamping

Once the work piece has been clamped, the operator can leave it to the PLC to complete the task. So, we now need the PLC to take care of any errors that may occur.

Withdrawal

If the drill motor doesn't run (no run feedback) we certainly don't want the drill ram to advance, so we need to add this feedback to the advance condition.

We can add an additional step in the process, or we can add a fault routine that stops the process and sets an alarm indicator (lamp, bell and so on) to show the operator there is a problem.

Retracting

If the Drill runs OK but the drill ram fails to advance (move off its back limits switch) or takes too long to drill the hole (before it activates the Forward limit), we will again need to run our fault routine.

Unclamping

Obviously, we need to make similar judgment calls for the remaining steps.

The Emergency Stop

When we look at programming our 'Emergency Stop', we need to look at what is happening to our drilling machine at the time. If we are in Step 2 (clamping) or Step 4 (retracting) we certainly don't want to release the clamp, as the drill will still be spinning and could throw the work piece, creating an additional hazard.

In the other steps, **de-energising** everything is probably the best answer.

Once we have established what is safe, we need to program the Emergency Stop to achieve a safe state as quickly as the machine will allow.

Programming the outputs

This is almost always the last step in the programming process. When we have our operating sequence and our fault handling in place and tested, we can add the output code and run the plant, correct operation first time out.

Each step in our sequence should have an internal coil or flag set so we know which step is in operation at any given time.

By using these internal step flags to turn on our outputs on the appropriate steps, we can easily complete out program, knowing it is correct before we ever start to run the actual plant.

Storing your program

When you are using a PC based programming package, as is most commonly the practice with our modern PLCs, you will have the option to save your ‘Off-Line’ program file to your computer hard disk, or to a memory stick.

Saving your work often as you are writing your code is a good idea. This will save you the trouble of having to re-write the entire code if your computer crashes.

It is always a good idea to write your completed code to CDROM and leave it with the PLC or at least with your customer so when you need to make adjustments in the future, you will have the off- line files that contain all your documentation.

If the program is not too long, it is often a good idea to print it out to allow maintenance personnel to use the I/O lists and so on for plant fault-finding.

Answers to review questions

1. TRUE or FALSE

a. Compared to relay and hard-wired logic; PLCs can have many times more contacts for each ‘soft’ relay. TRUE

b. The two major types of I/O devices are Digital and Analogue. TRUE

c. PLCs typically scan the logic left to right from top to bottom like reading a book. TRUE.

d. PLCs normally operate in a fixed sequence or scan. TRUE

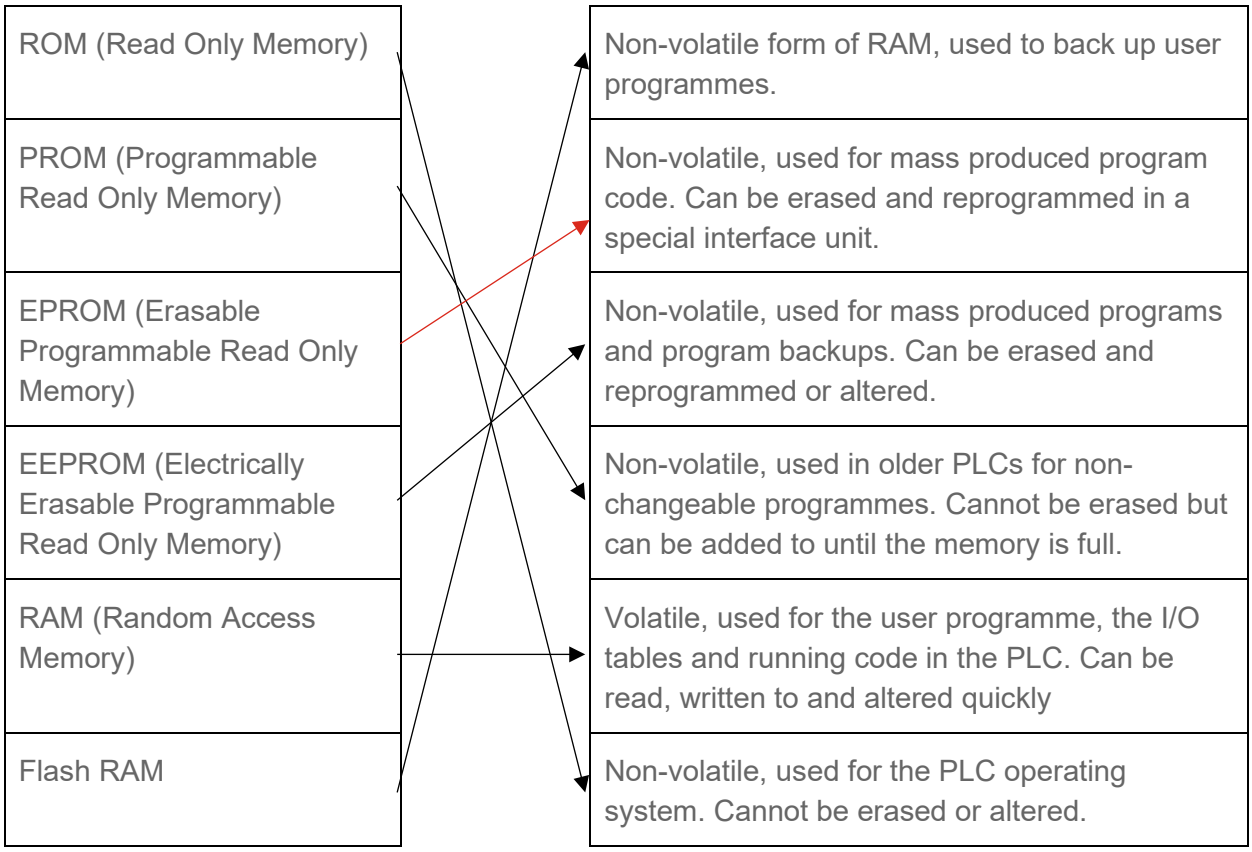
e. The Programmer is the device you use to enter your user program into the PLC Memory. TRUE

f. All PLC elements must have specific addresses to identify their memory locations for processing. TRUE

g. ALL Emergency Stop circuits should use normally closed Emergency Stop buttons, which you program with normally open contacts, and should self-latch so the equipment cannot restart just by releasing the Emergency Stop button. TRUE

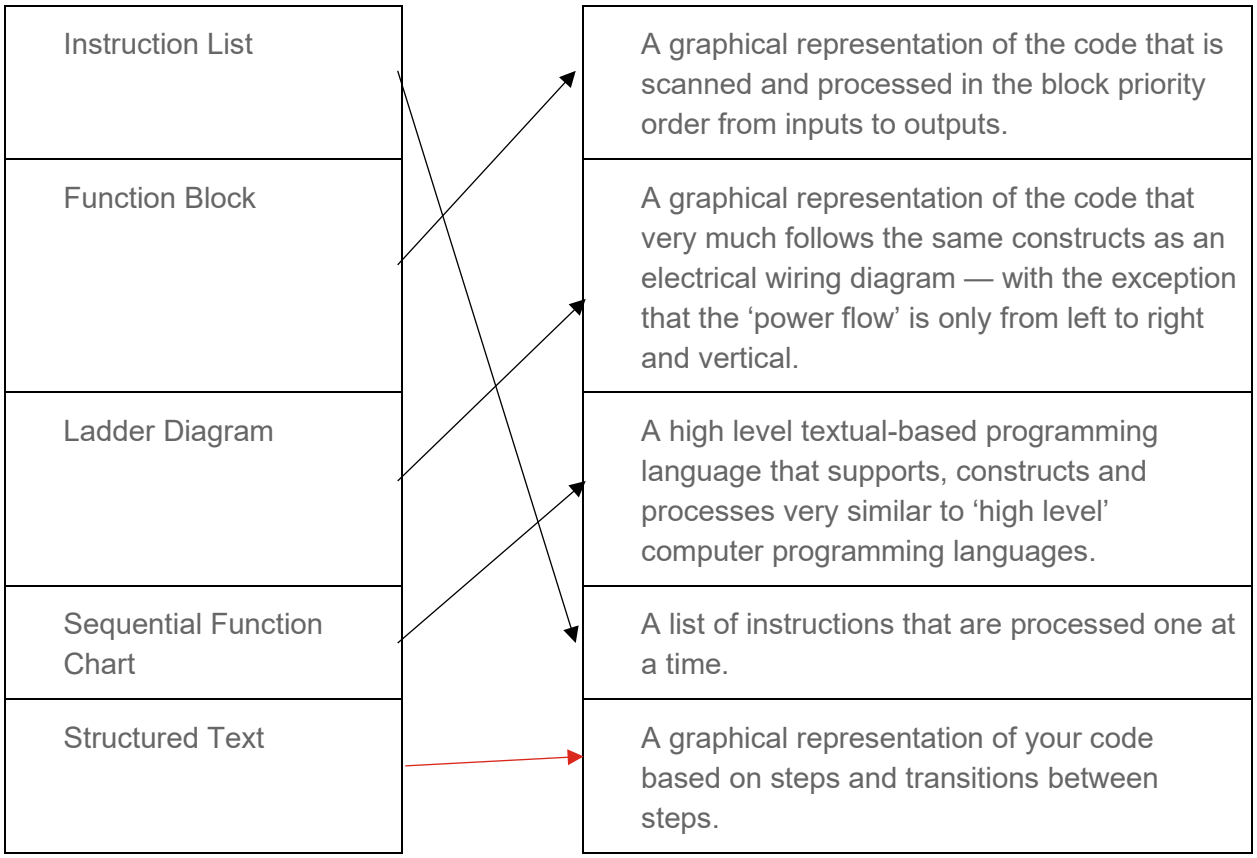
2. Match the columns:

a. All about memory



ROM (Read Only Memory)	Non-volatile form of RAM, used to back up user programmes.
PROM (Programmable Read Only Memory)	Non-volatile, used for mass produced program code. Can be erased and reprogrammed in a special interface unit.
EPROM (Erasable Programmable Read Only Memory)	Non-volatile, used for mass produced programs and program backups. Can be erased and reprogrammed or altered.
EEPROM (Electrically Erasable Programmable Read Only Memory)	Non-volatile, used in older PLCs for non-changeable programmes. Cannot be erased but can be added to until the memory is full.
RAM (Random Access Memory)	Volatile, used for the user programme, the I/O tables and running code in the PLC. Can be read, written to and altered quickly
Flash RAM	Non-volatile, used for the PLC operating system. Cannot be erased or altered.

b. Programming methods



Te Pūkenga

earnlearn-tepukenga.ac.nz

0800 327 648 (0800 EARN IT)