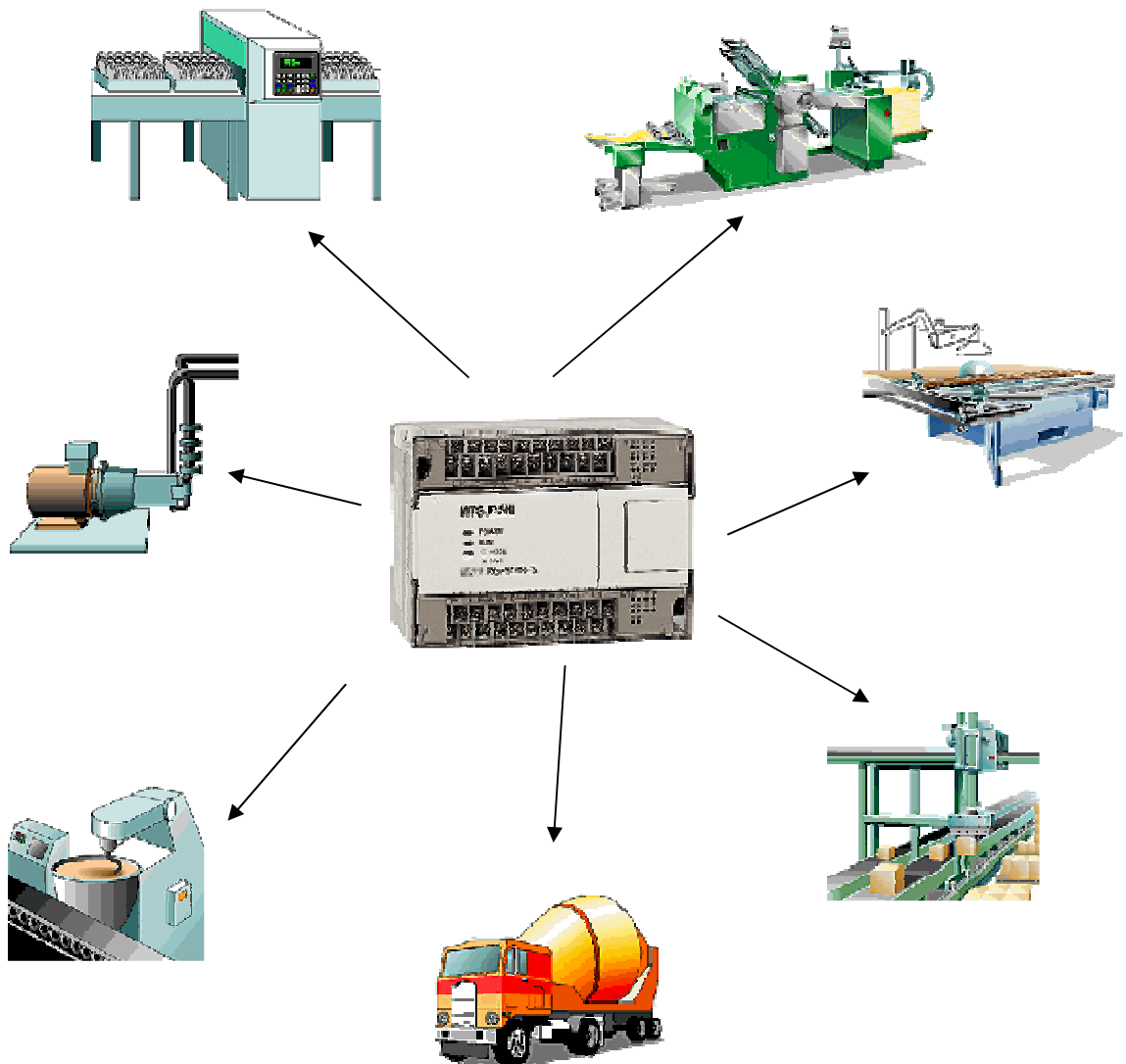


Demonstrate knowledge of Programmable Logic Controllers (PLC's)



Note: This short course workbook also meets the requirements of Unit Standard 5926 Version 3. By completing the Theory Test, Research assignment, and most of the PLC programming exercises students can gain a cross credit to US5926 version3 level 4 5 credits (Part of NCEE)

TABLE OF CONTENTS

| | |
|--|-----------|
| INTRODUCTION | 3 |
| PLC HISTORY | 6 |
| ELEMENT 1 – PLC PRINCIPLES | 12 |
| PLC'S COMPARED TO HARD WIRED, RELAY LOGIC. | 12 |
| HARDWARE MODULES | 17 |
| PLC OPERATING SEQUENCE | 25 |
| INPUT / OUTPUT (I/O) | 31 |
| INPUT | 31 |
| OUTPUT | 32 |
| PROGRAMMING METHODS | 35 |
| LADDER LOGIC | 43 |
| PLC PROGRAMMING SIMULATION SOFTWARE | 45 |
| PLC TERMS | 46 |
| EMERGENCY STOP PROGRAMMING CONVENTIONS | 49 |
| ELEMENT 2 – DESIGN, WRITE AND STORE PLC PROGRAMS | 51 |
| THE TASKS | 51 |
| TIPS & SUGGESTIONS | 52 |
| REFER APPENDIX B: GETTING STARTED WITH ZELIOSOFT2 SELF TEST QUESTIONS | 53 |
| SELF TEST QUESTIONS | 53 |
| APPENDIX A – RELAY PROBLEM (1ST NIGHTS TASKS) | 60 |
| APPENDIX B – GETTING STARTED WITH ZELIOSOFT2 | 64 |
| APPENDIX C - PLC PROGRAMMING EXERCISES | 69 |
| PROGRAM 1 – NORMALLY OPEN CONTACT | 69 |
| PROGRAM 2 – AND | 69 |
| PROGRAM 3 – OR | 69 |
| PROGRAM 4 – XOR | 69 |
| PROGRAM 5 – SEAL-IN CIRCUIT | 69 |
| PROGRAM 6 – DELAY ON | 69 |
| PROGRAM 7 – DELAY ON DELAY OFF | 69 |
| PROGRAM 8 – CONTINUOUS OPERATION | 70 |
| PROGRAM 9 - COUNTERS | 70 |
| PROGRAM 10 – STAR/DELTA STARTING CIRCUIT | 70 |
| PROGRAM 11 – AUTOMATED CAN SYSTEM | 71 |

Introduction

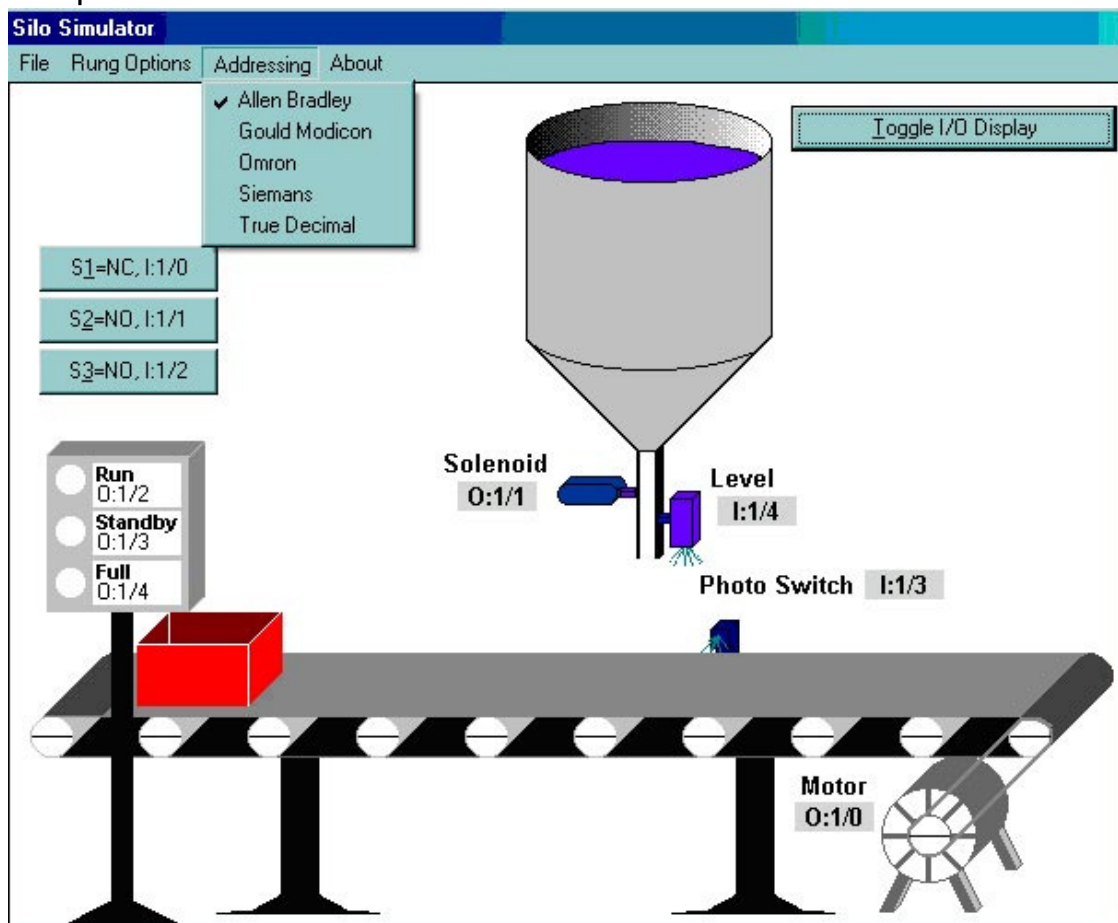
What is a PLC? (Programmable Logic Controller)

A PLC is a dedicated computer system. As such, it shares with all other computer systems much common ground.

The essential structure of a PLC comprises

- a Central Processing Unit (CPU),
- memory,
- Input modules / interface
- Output modules / interface
- Peripheral units (programming modules, HMI)

The CPU may be a stock microprocessor such as the 8086, 6800 series or a specially designed micro-controller. Memory will include Read-Only Memory (ROM) and Random Access Memory (RAM) similar to that found in any desktop home or office PC.



The main differences between a PLC and a personal computer are as follows:

Number of Inputs and Outputs (I / O)

A typical PLC application will call on several dozen physical inputs and/or outputs, and may have as many as several thousand.

PLC I/O will be designed to interface with industrial-level voltages and currents, and typically will be able to handle 230 or 110 VAC or 24 V DC, rather than the standard 5V logic levels of PC interface cards.

Memory allocation

There are three areas of memory in a PLC and a PC, which are related to the embedded software, the applications program, and data. PLC memory will not be as extensive as a PC can be (PC's these days 250+ Gbytes of HDD)

Generally, the embedded software will be in ROM form and is not user accessible. Non-volatile memory: When power goes off the information stays. Applications software is alterable by the user, to reflect the required changes in control system operation. Created on Ram (with battery backup) first and then often, at the end of the development phase, the applications software will be burnt into EPROM (Erasable PROM or UV PROM) and nowadays EEPROM (Electrically Erasable PROM).

Data memory will hold those items such as I/O states that can change from time to time as the system operates and are held in RAM. Volatile memory: power off and data goes. Memory may have a battery back-up system to hold vital data on RAM.

Programs

Programming will be done in a format that is easily developed from the industrial requirements of the package. Most are done these days on the PC and are then downloaded onto the PLC. The software packages can be used to simulate the operation of the PLC and the programmer can possibly pick up design faults before implementing. "Live" implementation is now using this method and changes can be quickly carried out with minimum intrusion into an automated system.

The PC's will have a high-level language (HLL), such as FBD (Functional Block Diagram) installed with multiple options and facilities available to be used.

The PLC program has little or no need for an operator interface once they are programmed. They can be created on a PC e.g. Zeliosoft2, simulated and tested on the PC before being downloaded to the PLC.

PLC Keyboards and displays are often very simple and added into the standard discrete I/O hardware.

The applications program consists of a (relatively) few steps repeated over and over again, rather than the far more complex operations in a general-purpose computer.

Before we get too far into this workbook there are a few things that it is assumed you will already know. Most of this is covered by Unit Standard 2780. If you have problems in these areas you should let your tutor know so that some remedial work can be set for you to catch up.

You should have a working knowledge of these areas,

How to start up and run computer programs

- Numbering systems and how to convert from one to another.
 - Binary
 - Binary Coded Decimal (BCD)
 - Decimal
 - Hexadecimal (Hex)
 - Octal
- Relay logic (how to read and interpret relay wiring diagrams & operation)
- Basic electrical safety rules
- Electrical wiring practice.

Remember

***There is no such thing as a dumb question,
Just some people too self conscious to ask!***

PLC HISTORY

A LITTLE HISTORY THAT SHOWS HOW PLC'S EVOLVED

When we first look at PLC's it is quite often as a part of an overall control system in a factory, or as the "black box" in the middle of a dedicated machine.

For years a select few "god like" engineers, electricians and instrument technicians have held power over this marvellous piece of equipment. They have talked about the PLC in a language all of their own, full of impressive sounding jargon, seemingly designed to confuse the issue.

While the PLC can be very complex, all PLC's follow a few simple conventions, which, when you have mastered them, allow anyone to follow what, where, when and how they work.

When electricity came into common usage in industry, there were very few ways of using it to control a process. The steam engine was replaced by electric motors but everything else was pretty much the same, manual or mechanical control.

As time progressed and we saw the introduction of mass production techniques, electric motors became a cheaper and more reliable option. With the introduction of large assembly lines it became necessary to provide some automation to get things started and stopped in the correct sequences.

Several companies had devised electromechanical devices (Solenoids, Contactors, relays etc.) to allow the switching of large currents by using a low current pilot (control) circuit. Your starting system of the car does this. However there were several limitations.

If you wanted to turn on 20 motors at once you would require several relays to energise the individual motor starters. To physically switch 20 contacts required too much mechanical force to try with one physical device. Hence the development of relay logic control panels.

Examples of early relays
Taken from "The New Electrical Encyclopedia" (circa 1950's)

RELAY: INSTRUMENTS FOR PROTECTION, STARTING & OTHER PURPOSES

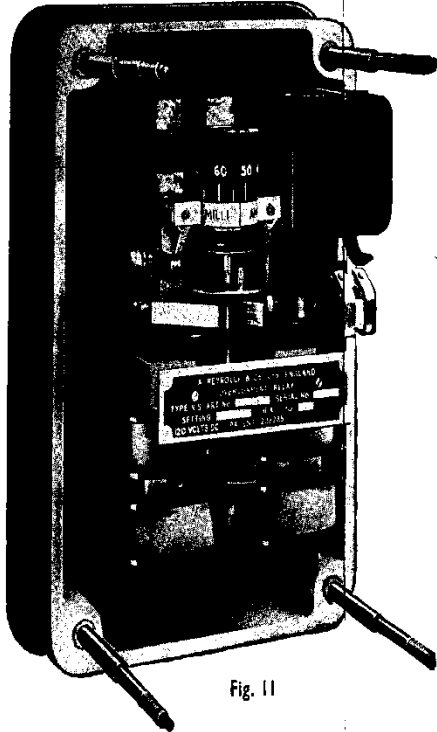


Fig. 11

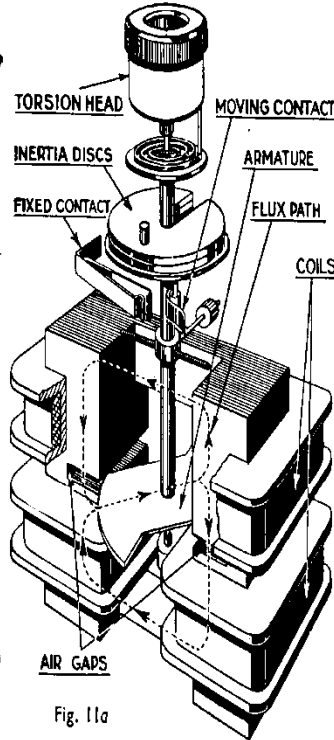


Fig. 11a

Fig. 11. Rotating-armature sensitive relay. Type RS. When operating coils are energized resulting flux draws armature into air gaps of electro-magnet, closing main contact and energizing the auxiliary tripping contactor, which completes the tripping circuit and operates a flag indicator. Relay is self-resetting but flag indicator has to be reset by hand. Three current settings; adjustable by calibrated torsion head, which adjusts tension of control spring. Line diagram (Fig. 11a) shows construction. Applications: split-pilot feed protection, sensitive earth-fault protection.

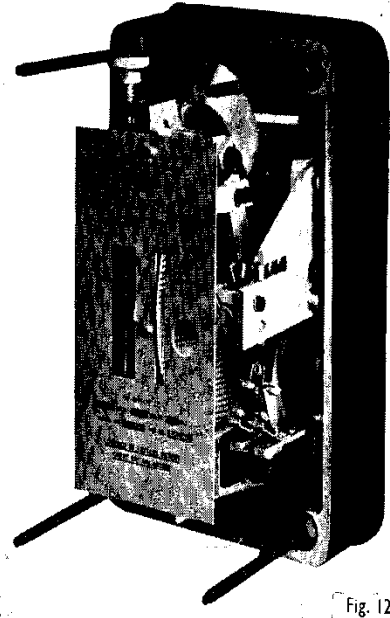


Fig. 12

Fig. 12. Definite-time-lag relay with gravity controlled magnetically braked revolving disc geared through clutch to contact-carrying quadrant. May be either time-lag operation and instantaneous reset or instantaneous operation, time-lag reset, adjustable. Application: residual-voltage protection. *Reynolds & Co., Ltd.*

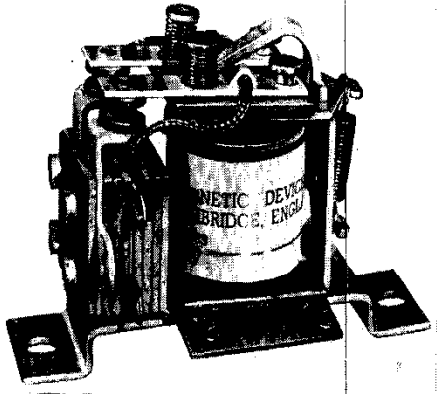


Fig. 13

Fig. 13. Heavy duty miniature relay for A.C. (up to 250V) or D.C. (up to 140V). Up to 15 amp. continuous rating, but will carry heavy overloads for short periods, e.g. for motor starting. May be arranged for double pole change-over or double pole single throw (normally open, or normally closed). *Magnetic Devices, Ltd.*

Fig. 15. Telephone type relay which also finds many uses in power engineering, e.g. supervisory control. Compact design permits close stacking of banks of relays. Particularly suitable for plug and socket mounting. *Automatic Telephone Co., Ltd.*

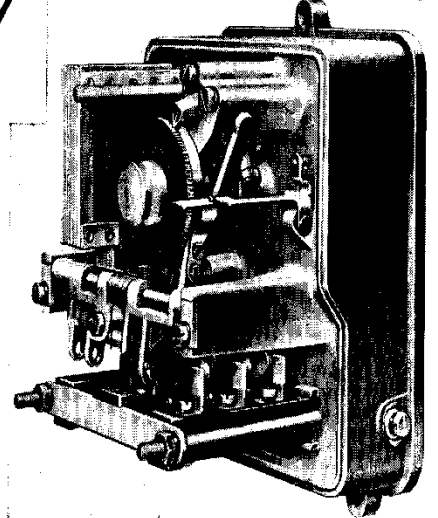
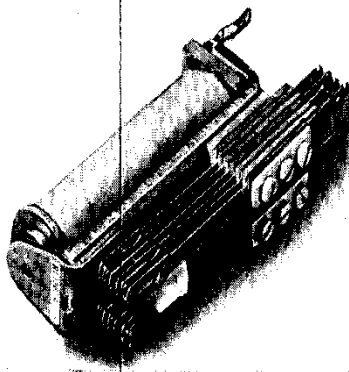


Fig. 14

Fig. 14. Industrial electric timer. For giving definite predetermined time interval between initial energization of timer operating circuits and operation of timer contacts. Relay operated by synchronous motor and timer energized by pilot switch or by push button. Three contacts for external use. *The B.T.H. Co., Ltd.*

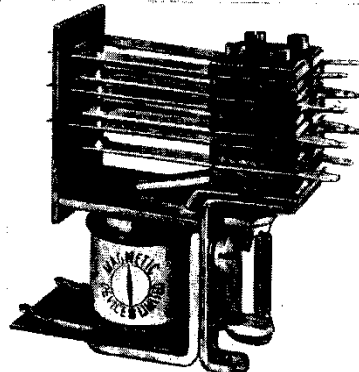


Fig. 16. Versatile miniature relay for A.C. and D.C. suitable for large number of contact combinations. It comprises two basic parts; coil assembly and contact assembly. Coils easily interchanged to suit required voltage (3.5V/250V A.C. and 1.5V to 110V D.C.). *Magnetic Devices, Ltd.*

By using “special relays”, Timers, Counters etc. it became possible to fully automate plant and machinery without the more complex mechanical systems that they followed.

This eventuated in a huge increase in automated manufacturing.

One industry that leapt into this technology was automobile manufacturing.

However as the industry developed the demand for cars meant that new models were produced yearly. This led to the assembly lines having to be rebuilt and the control systems being rewired annually, with the associated added costs and inherent delays with down-time and fault-finding.

In the late 1960's PLC's were first introduced. The primary reason for designing such a device was eliminating the large cost involved in replacing the complicated relay based machine control systems. Bedford Associates (Bedford, MA) proposed something called a Modular Digital Controller (MODICON) to a major US car manufacturer.

Other companies at the time proposed computer based schemes, one of which was based upon the PDP-8. The MODICON 084 brought the world's first PLC into commercial production.

When production requirements changed so did the control system. This becomes very expensive when the change is frequent. Since relays are mechanical devices they also have a limited lifetime which required strict adherence to maintenance schedules. Troubleshooting was also quite tedious when so many relays are involved. Now picture a machine control panel that included many, possibly hundreds or even thousands, of individual relays. The size could be mind boggling. How about the complicated initial wiring of so many individual devices! These relays would be individually wired together in a manner that would yield the desired outcome. Were there problems? You bet!

These "new controllers" also had to be easily programmed by maintenance and plant engineers. The lifetime had to be long and programming changes easily performed. They also had to survive the harsh industrial environment. That's a lot to ask! The answers were to use a programming technique most people were already familiar with and replace mechanical parts with solid-state ones.

In the mid70's the dominant PLC technologies were sequencer state-machines and the bit-slice based CPU. The AMD 2901 and 2903 were quite popular in Modicon and Allen Bradley PLC's. Conventional microprocessors lacked the power to quickly solve PLC logic in all but the smallest PLC's. As conventional microprocessors evolved, larger and larger PLC's were being based upon them. However, even today some are still based upon the 2903. (AB's PLC-3) Modicon has yet to build a faster PLC than their 984A/B/X which was based upon the 2901.

Communications abilities began to appear in approximately 1973. The first such system was Modicon's Modbus. The PLC could now talk to other PLC's and they could be far away from the actual machine they were controlling. They could also now be used to send and receive varying voltages to allow them to enter the analogue world. Unfortunately, the lack of standardisation coupled with continually changing technology has made PLC communications a nightmare of incompatible protocols and physical networks. Still, it was a great decade for the PLC!

The 80's saw an attempt to standardise communications with General Motor's manufacturing automation protocol (MAP). It was also a time for reducing the size of the PLC and making them software programmable through symbolic programming on personal computers instead of dedicated programming terminals or handheld programmers. Today the world's smallest PLC is about the size of a single control relay!

The 90's have seen a gradual reduction in the introduction of new protocols, and the modernisation of the physical layers of some of the more popular protocols that survived the 1980's. The latest standard (IEC 61131-3) has tried to merge five (5) PLC programming languages under one international standard. We now have PLC's that are programmable in Function Block Diagrams (FBD), Instruction Lists (IL), Ladder Diagrams (LD), Sequential Function Chart (SFC) and Structured Text (ST) all at the same time!

PC's are also being used to replace PLC's in some applications.

The original company who commissioned the MODICON 084 has actually switched to a PC based control system.

PC's still need to activate high voltage/high current external devices and will need some form of interface to achieve this.

PLC's have the advantage of being less sensitive to static damage and being able to working in noisy (electrically noisy) and dusty environments.

MODERN DEVELOPMENTS

Early PLCs were merely capable of monitoring on/off conditions & using Boolean logic principles to control simple on/off outputs. Simple timing functions were also usually available. This is well suited to the majority of industrial control requirements (e.g. transfer lines, grinding & boring machines which had previously been controlled by cam followers & relay logic.)

Modern PLCs however also incorporate at least some of the following features:

- real time programmable timers & programmable counters
- arithmetic, bit & data manipulation & processing
- PID capability (usually via a special module)
- expanded fault diagnostics capabilities
- common data highway or LAN communications interfacing between PLC modules & a main computer or control console.
- operator display/control interfacing (HMI)

With these features, PLCs have become the “worker bees” of modern process control & are capable of providing almost every conceivable function such as data logging, remote monitoring, process display, management advisory & inventory control. The main thrust of future development is likely to be in the area of providing better tools & methods for system programming & fault analysis by factory engineers.

A NOTE: Some manufacturers are now using the abbreviation "PC" to indicate a programmable controller. Since personal computers (PCs) are increasingly being used as programming consoles for logic controllers, this situation leads to even further confusion. We shall standardize on "PLC" to indicate an industrial controller.

IEC 61131-3 allows for multi-vendor compatibility and multi-language programming.

SFC is a graphical language that provides coordination of program sequences, supporting alternative sequence selections and parallel sequences.

FBD uses a broad function library to build complex procedures in a graphical format. Standard math and logic functions may be coordinated with customizable communication and interface functions.

LD is a graphic language for discrete control and interlocking logic. It is completely compatible with FBD for discrete function control.

ST is a text language used for complex mathematical procedures and calculations less well suited to graphical languages.

IL is a low-level language similar to assembly code. It is used in relatively simple logic instructions.

Relay Ladder Logic (RLL), or ladder diagrams, is the primary programming language for programmable logic controllers (PLCs). Ladder logic programming is a graphical representation of the program designed to look like relay logic.

Flow Chart is a graphical language that describes sequential operations in a controller sequence or application. It is used to build modular, reusable function libraries. C is a high level programming language suited to handle the most complex computation, sequential, and datalogging tasks. It is typically developed and debugged on a PC.

BASIC is a high level language used to handle mathematical, sequential, data capturing and interface functions.

Element 1 – PLC Principles

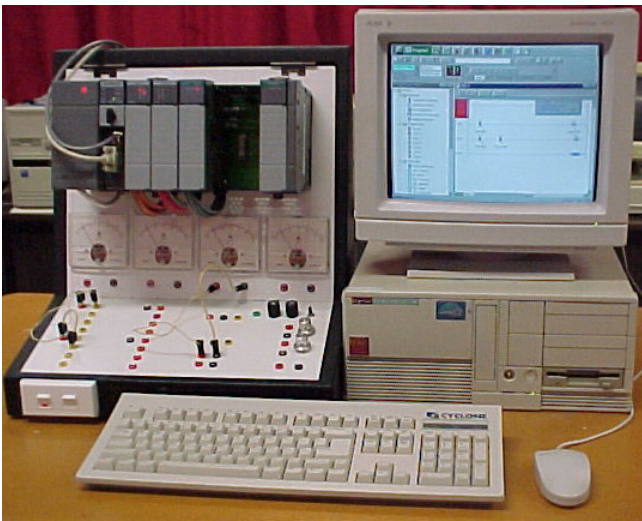
PLC'S COMPARED TO HARD WIRED, RELAY LOGIC.

MULTIPLE CONTACTS, VARIETY OF CONTROL TASKS, EASE OF ALTERATION AND DUPLICATION, TIME SAVINGS, ON-LINE DOCUMENTATION.

As you may have already discovered, a relay is an electro-mechanical or electronic device that allows a small level of current to switch a large current or several individual currents at the same time. A relay can have multiple contacts which can be either NC (Normally Closed and open on operation) or NO (Normally Open and close on operation), but there are mechanical limits as to how many contact pairs can physically be on a relay.

Individual relays may have special functions such as dedicated timing relays (timers) counting functions where the relay switches after an input signal is switched a pre-set number of times, (Counters) and many other special relay types.

To build automated plant from these individual relays requires a large investment in space and time. Then if you want to change the process it takes a similarly large investment in time etc. to re-build the system.



With a PLC system, it still requires all the “Field devices” (limit switches, Level sensors etc.) that you would require automating the process by using relays. However the savings in space and time start to be realised when you consider the process in completing the operation.

The Allen Bradley SLC500 training rig shown can hold 6 I/O modules without extra racks and can interface with 100's of field devices yet takes up little more space than a loaf of bread. Larger Chassis' and multiple chassis' can address 100's more I/O.

| <u>Relays</u> | <u>PLC's</u> |
|--|---|
| Process requirements are identified and documented. | |
| Field wiring is designed and checked against the process requirements | Field wiring is designed and checked against the process requirements |
| Field Wiring has to be run and connected to control panel. | Field Wiring has to be run and connected to PLC I/O racks. |
| Relay Logic (Hard wired) All control wiring has to be physically installed and connected in the workshop or on the plant floor. | PLC's (Soft wired) Write program on-line or off-line in the office. Automated documentation coincides with programming. |
| Commissioning involves checking all operations individually each relay and connection should be checked. (usually by isolating each area and trial running parts of the process) | Commissioning involves automatic software checks and I/O checks |
| Errors are identified (time-consuming) and the control system is redesigned, redrawn, documented and rewired. | Errors are identified and the program is edited. Documentation is completed on-line |
| Re-commission (as required) | Re-commission (as required) |
| Completed as built drawings | Print PLC listings |
| Then, what if the plant manager decides to alter the process? Start again at the beginning? And maybe days or weeks behind schedule? | Then, what if the plant manager decides to alter the process? Just Edit the program and maybe only hours behind schedule? |

From what we have already considered, we can see that the PLC has evolved from the requirements of simplifying the “old” relay logic approach to automation and increases in efficiency gained when plants need to be altered.

Relays usually are limited to about 4 sets of contacts due to the mechanical requirements of operation.

PLC “Soft Relays” or Internal relays (programmed in the software) can have as many contacts as you want in any configuration. And can be set as N/O N/C, Momentary etc.

Changing a hard-wired system requires shutting the plant down, observing safety requirements, physically changing relays and wiring which can take considerable time.

Changing a PLC based process can usually be performed on-line without shutting down the process. Alternatively the changes can be performed elsewhere, simulated on the PC and then downloaded to the PLC. (Saving considerable downtime and fault-finding)

Documenting a hard wired system requires tracing the existing system and or drawing the original designed system and editing or re-drawing each time a modification is done.

Documenting a PLC system is as simple as labelling the various elements of your program as you go and then printing a program listing when you are complete.

Fault finding a relay based system requires a good knowledge of the particular plant, along with reading the drawings and testing individual components.

Fault finding a PLC based system requires knowledge of the programming package and reading the on-line documentation as you remotely check the process.

It should be stated/noted that before any programming on a PC takes place a draft is created on paper first.

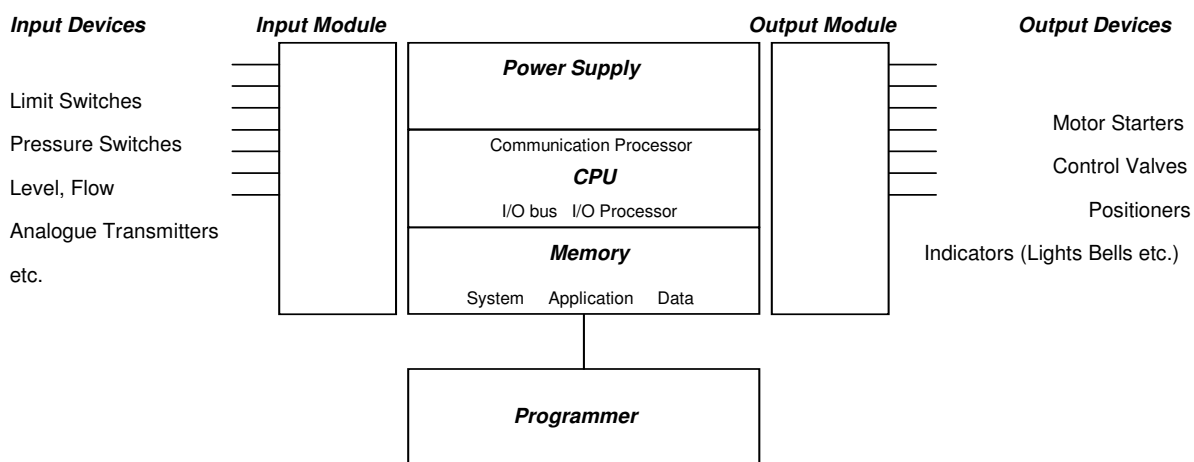
This can be checked and “paper-tested” for logical errors.

Programs, usually in a HLL, can be compiled and checked for syntax errors beforehand and simulated on the PC.

FUNCTIONAL COMPONENTS OF PLC'S

In general terms the PLC consists of a power supply CPU, memory areas, and appropriate circuits to receive or send input /output data and finally a device to write the program. We can actually consider the PLC to be a box full of hundreds or thousands of separate relays, counters, timers and data storage locations. They don't "physically" exist but rather they are simulated and can be considered software or internal counters, timers, etc.

Internal relays (Auxiliary relays), are simulated through “bit locations” in registers. (More on that later)



POWER SUPPLY – This is as the term says, provides the power voltage to the PLC. Most PLC's have a dedicated power supply that supplies power for the internal workings of the PLC itself as well as providing power that can be used to supply some field devices. When installing a PLC system it is important to allow a power supply with enough current capacity to supply all the needs of the installation.

CENTRAL PROCESSING UNIT (CPU) - The CPU controls 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 PLC module. Some machines (most modern PLC's) will have additional processors that look after the Inputs and Outputs and any PLC communications required.

INPUT CHANNELS (In the Input module) - These are connected to the input devices. They physically exist and receive voltage or current signals from switches, sensors, etc.

INTERNAL AUXILIARY RELAYS (contacts) - These do not receive signals from the outside world nor do they physically exist. They are simulated relays and are what enables 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-These also do not physically exist. They come in many varieties and increments. The most common type is an on-delay type. Others include off-delay and both retentive and non-retentive types. 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 some more advanced such as retentive timers that remember how far they have timed the last time they were activated.

OUTPUT CHANNELS (In the output module) - These are connected to the output devices. They physically exist and send on/off signals to solenoids, lights, etc. They can be transistors, relays, or triacs depending upon the model chosen.

DATA STORAGE-Typically there are registers assigned to simply store data. They are usually used as temporary storage for math or data manipulation. They can also be used to store data when power is removed from the PLC. Upon power-up they will still have the same contents as before power was removed. Very convenient and necessary!! These will have an internal battery source providing backup power.

Have a look at the Zelio class set materials and see what these modules can do.

HARDWARE MODULES

INPUT / OUTPUT DEVICES

These are the many and varied real world devices that are attached to the PLC.

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 probably more use to you to look at the types of signal that the PLC receives from these devices and how the PLC uses these signals.

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.

I/O MODULES

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.

In the “Modular” PLC the input and output modules plug into a backplane or chassis which provides the power supply and connection to the PLC’s “CPU” module.

There are several “Special” 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.

Others you may come across include the following,

ASCII input and output modules for receiving ASCII data from weighing machines, modems etc. and outputting to computers, displays printers and so on.

RTD (Resistance Temperature Detector) and Thermocouples input modules for directly reading temperatures.

PID Module has its own dedicated PID controller built in.

A **proportional–integral–derivative controller (PID controller)** is a generic control loop feedback mechanism widely used in industrial control systems. A PID controller attempts to correct the error between a measured process variable and a desired setpoint by calculating and then outputting a corrective action that can adjust the process accordingly. http://en.wikipedia.org/wiki/PID_controller

SERVO CONTROL MODULE This module has both outputs to control the servo motors and inputs to read the servo position.

COMMUNICATIONS MODULE This module type allows computers and other controllers etc. to communicate with the PLC directly. These can vary from bar code readers to smart instruments, Variable speed drives etc.

CO-PROCESSOR MODULES. These are generally available only of the larger more sophisticated PLC's such as the Allen Bradley PLC5 series and similar. The co-processor can be a dedicated computer running an Intel ® processor that runs totally independent software for data logging and includes its own disk drives. It can also be another PLC processor that is a backup to the main processor.

From these few examples you can see that there are a large number of “speciality” I/O Modules. If you come across any of these, tread with caution, as they each have their own particular requirements and if you are not careful you could inadvertently crash a module. In these cases, **always** RTFM. (Read The Flaming Manual)

I/O SIGNAL TYPES

If we ignore the communications modules which can send or receive serial data etc. most other I/O modules fall into TWO categories. These are either Analogue or Digital.

We will look at each in turn along with the variations in signal to and from each.

ANALOGUE I/O comes in various forms with the Input or output falling into three distinct types; Voltage, Current or Resistance.

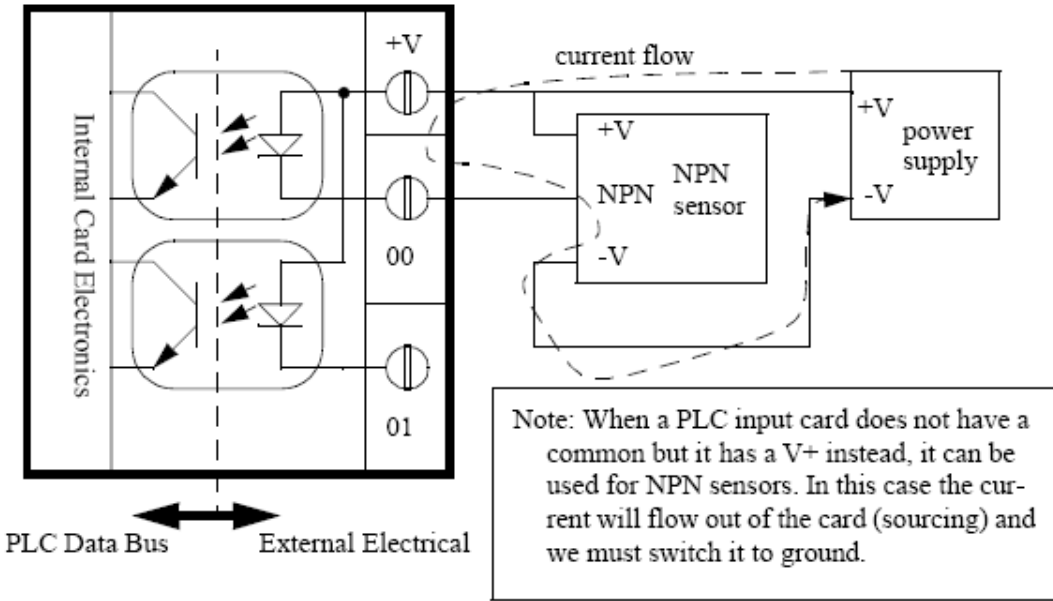
Ultimately the PLC doesn't care what the field signal is as the Input module must suit the input analogue signal type and this then converts the signal to TTL signal levels, 1 to 5 Volts that the PLC can handle.

This 1 to 5 volt signal is then fed into an analogue to digital converter (ADC) that outputs a binary number equating to the signal level. This numeric value is then stored in a register location in the PLC's memory dedicated to the particular field input.

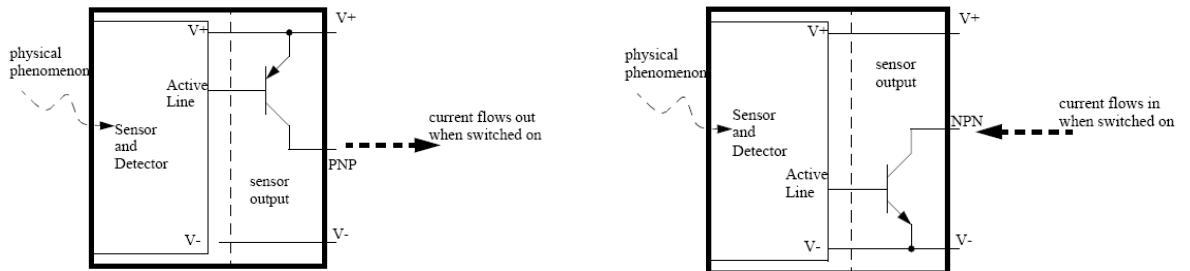
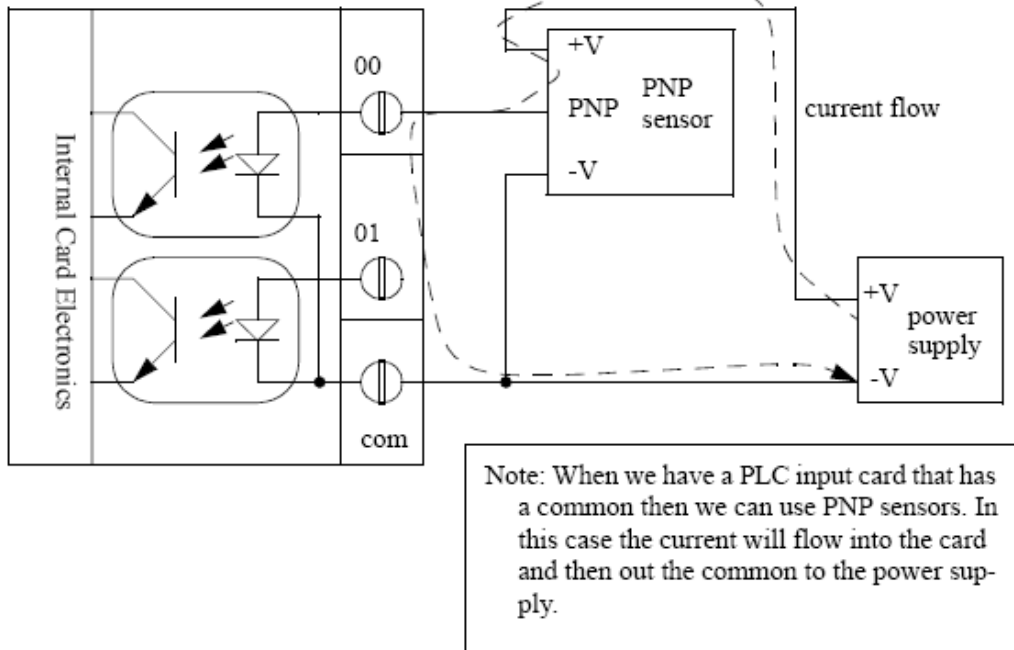
The binary number can be as small as 1 or 2 bits to 64 bits, or more.

The more bits then more memory required and the bigger the processor to process it.

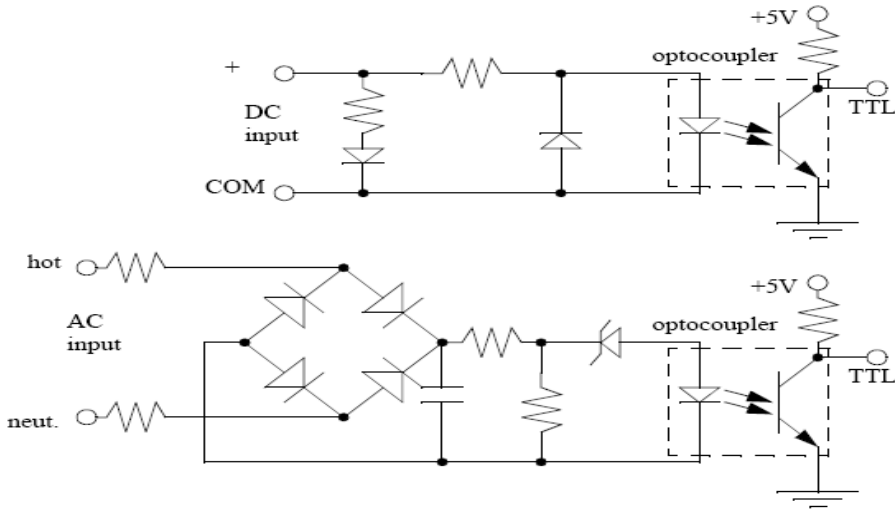
PLC Input Card for Sinking Sensors



PLC Input Card for Sourcing Sensors

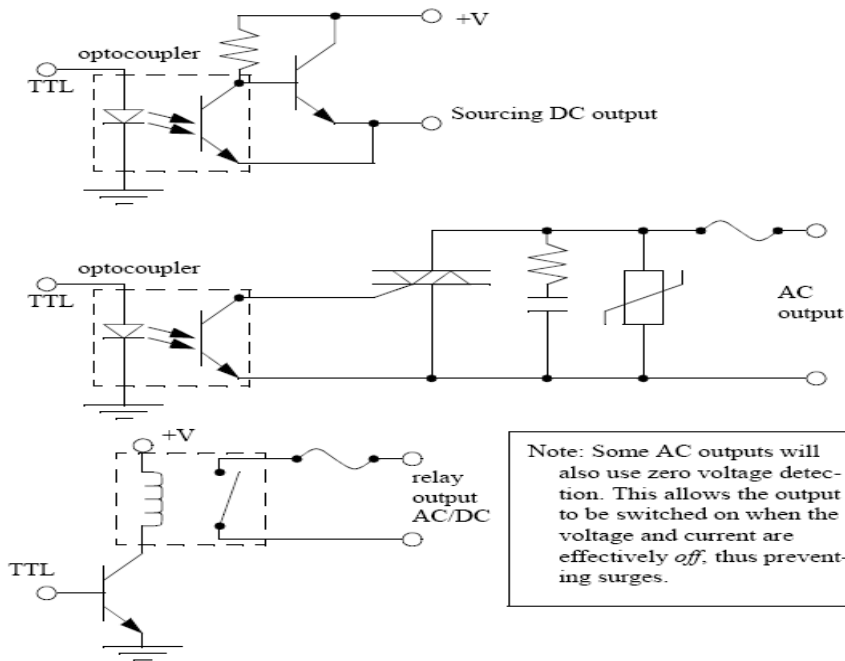


ASIDE: PLC inputs must convert a variety of logic levels to the 5Vdc logic levels used on the data bus. This can be done with circuits similar to those shown below. Basically the circuits condition the input to drive an optocoupler. This electrically isolates the external electrical circuitry from the internal circuitry. Other circuit components are used to guard against excess or reversed voltage polarity.



Opto-coupling provides great electrical isolation on the Input

ASIDE: PLC outputs must convert the 5Vdc logic levels on the PLC data bus to external voltage levels. This can be done with circuits similar to those shown below. Basically the circuits use an optocoupler to switch external circuitry. This electrically isolates the external electrical circuitry from the internal circuitry. Other circuit components are used to guard against excess or reversed voltage polarity.



Opto-coupling provides great electrical Isolation on the Output.

DIGITAL I/O as for analogue I/O comes in various forms with differing levels of input being accepted. While you may be looking at a contact closure, or Voltage input, the Digital I/O only sees a 1 or a 0 for each input location.

Each input forms one “bit” of a binary word in a particular memory location. Each digital output is energised bit by bit from an output word dedicated to the particular output module.

HIGH SPEED COUNTER MODULES

When we look at the operation of a particular PLC, we see that the time taken for each program scan cycle can take 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 either lost or missed.

Because of this most PLC’s 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 can operate an interrupt that halts the PLC scan cycle and updates the logic immediately regardless of the current position in the PLC scan.

This immediate interrupt can then operate any output devices etc. as soon as they need to be operated, prior to returning to the normal operational scan.

This ensures any high-speed process can still be effectively controlled without the need for expensive dedicated controllers.

HIGH SPEED TIMER MODULES

As for the high-speed counter module, the high-speed timer is a hardware device that operates much faster than the programmed “soft” timers you use in your program. These modules can time extremely accurately and for very short time periods. They can also use interrupts as for the high-speed counters.

POWER SUPPLY MODULES

PLC's all require "Power" to operate. That is they require a source of voltage / current to run their internal operations. They also require "power" to operate the input and output modules. The field devices (both input and output) also require "power" to operate. Most PLC power supply modules provide a low voltage 24VDC output for powering some field devices as well as the internal supply for the PLC and the I/O modules.

When we look at the simple "brick" PLC's, 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 PLC's, 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.

PROGRAM MEMORY TYPES

RAM

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 etc. 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 etc. most PLC's have a battery providing power to the RAM chips. Typically this battery will last anywhere from 2 to 5 years before requiring replacement. It is also charged when power is supplied to the PLC.

ROM

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 generally can not be altered in the field. Once written it cannot be changed. A new version of operating system requires replacement of the ROM chip.

PROM

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 the use of current pulses that melt fusible links within the chip itself. Very few PLC's would use PROM chips as any program alterations would require the PROM chips to be replaced.

EPROM

Erasable Programmable Read Only Memory is similar in operation to the PROM, with one major difference. The EPROM can be erased by exposing it to ultraviolet light through a window in the chip case. When erased, the chip can then be reprogrammed from scratch. A sticky cover is placed over the window to prevent stray UV erasing it.

EPROM chips are used a lot in a large range of devices, from dedicated controllers to personal computers to PLC's. EPROM's are programmed by use of a special programming device or interface unit that the chip is inserted into. They can be programmed, then installed into PLC's for mass produced machine control without the necessity of programming each individual PLC. They also provide a level of program security in that they are difficult to edit and therefore are difficult for others to interfere with. They are non-volatile and therefore do not require battery backup.



EEPROM

Electrically Erasable Programmable Read Only Memory. Once again this 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. 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 ROM.

Memories are required for:-

- Application programs (RAM or EEPROM)
- Diagnostics (ROM or EPROM)
- PLC overall operation (RAM)
- Data (RAM & ROM)
- Communications (RAM & ROM)

The application program is written by the user and stored in the “user memory”. It determines how the process is controlled, in accordance with the application program, the input and output data status.

The size of the application program determines the amount of memory required, and the memory capacity may vary from less than 1,000 words to more than 64,000 words (64 KB of words), depending on the PLC manufacturer.

Some PLC’s are able to have memory upgrades.

The user can also access the data files memory. Data files store data associated with the control program, such as I/O status bits, counter and timer preset and accumulated values, and other stored constants or variables.

Memory Characteristics.

| Characteristics | TYPE OF MEMORY | | | | | |
|---|----------------|-----|------|-------|--------|---------------------------------|
| | RAM | ROM | PROM | EPROM | EEPROM | *Flash Memory (digital cameras) |
| VOLATILE (erased at power off) | YES | NO | NO | NO | NO | NO |
| WRITEABLE | YES | NO | YES | YES | YES | YES |
| ERASEABLE | YES | NO | NO | YES | YES | YES |
| REWRITABLE | YES | NO | NO | YES | YES | YES |
| UV REQUIRED (Ultra Violet for erasing) | NO | - | - | YES | NO | NO |

PLC OPERATING SEQUENCE

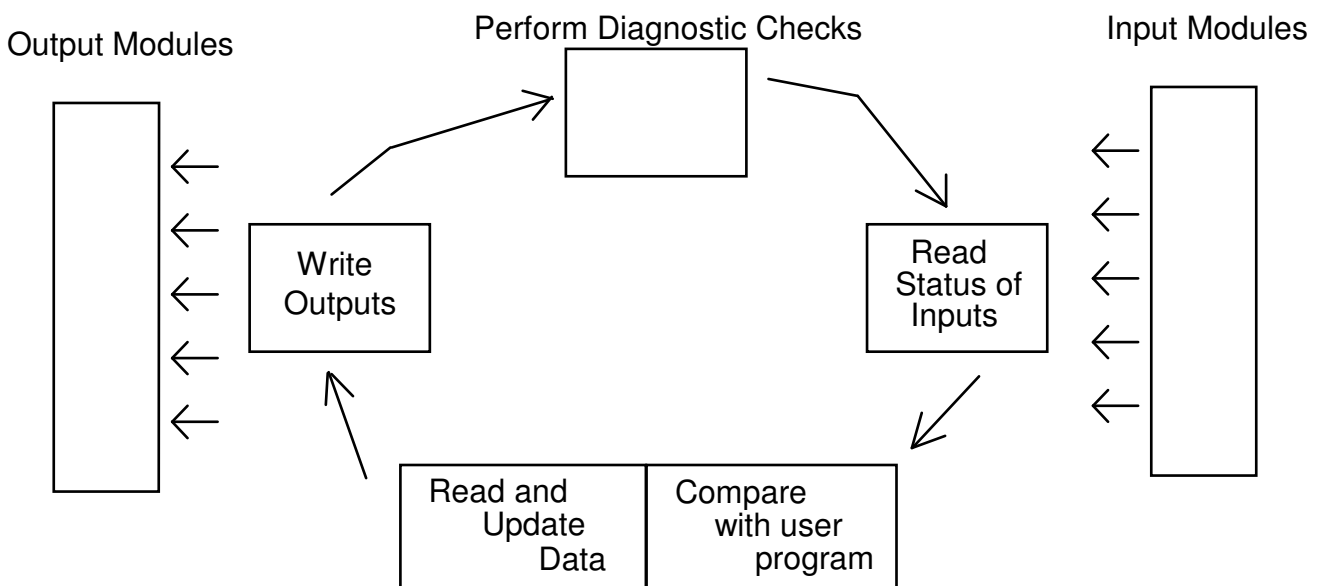
PROGRAM MODE

Where the PLC is not reading the inputs or operating the outputs, but is 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.

RUN MODE,

The PLC is actually 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.

This sequence is set and although may differ slightly from PLC to PLC, all follow this general sequence of operation.



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

The input modules whether fixed or modular measure a voltage or current signal from some real world 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 pigeon holes or post boxes that each can store a piece of information.

| | | | | | | | | | | | | | | |
|--|----------|----------|---------|----------|----------|----------|--|----------|----------|----------|--|----------|----------|----------|
| | | | | | | | | | | | | | | |
| | 10110010 | 0 | 1101010 | | | 1101010 | | | | 10101010 | | | | |
| | | 11010011 | 1101010 | | 11010100 | 11100101 | | | | 00010101 | | | 11000011 | 11100011 |
| | | | 1101010 | 11010001 | | | | 11010101 | 11111110 | | | 11010010 | | |
| | | | | | | | | | | | | 11010101 | | |

As with post office boxes each location has a unique address that the CPU can locate and you can use in your program to identify each particular piece of information that you may wish to use or change.

The actual addressing conventions differ from PLC manufacturer to PLC manufacturer and can also differ between PLC models from the same manufacturer. The particular addressing convention of each is usually covered in the documentation that comes with the PLC, or with newer machine it is covered in on-line help within the programming software.

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

Depending on the particular PLC this is updated by the CPU or a separate I/O processor within the CPU module.

(Note: some speciality I/O modules contain their own individual processors)

Each digital input module stores the status of each individual input as “bits” of an input word.

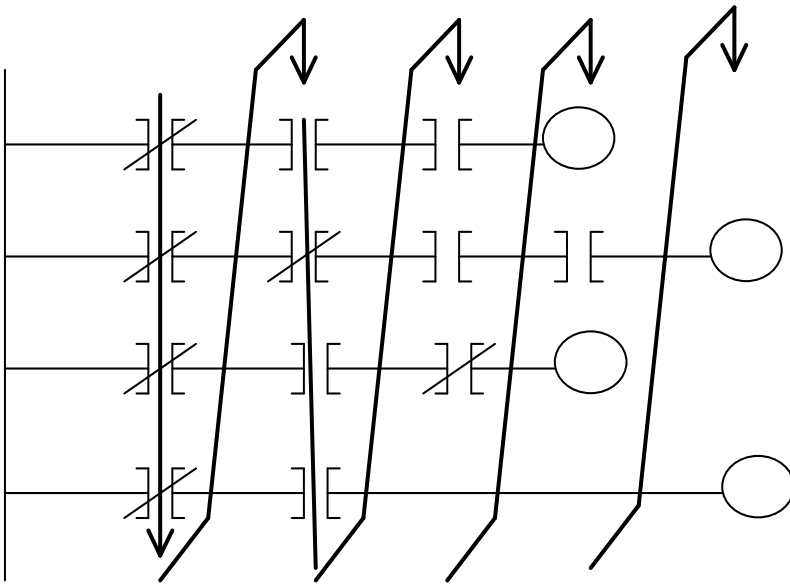
If you look back at our “PLC diagram” (page 15) you will see that we split the “memory” into three parts. Namely System, Application and Data.

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 this “data” area of memory.

The Application memory is where your program resides. It doesn't matter which programming system you use the result is converted to data that can be read by the PLC's processor.

When we write a Program and run it in the PLC there are a few conventions on how the processor actually reads the program. These can affect the way you actually need to put your program together.

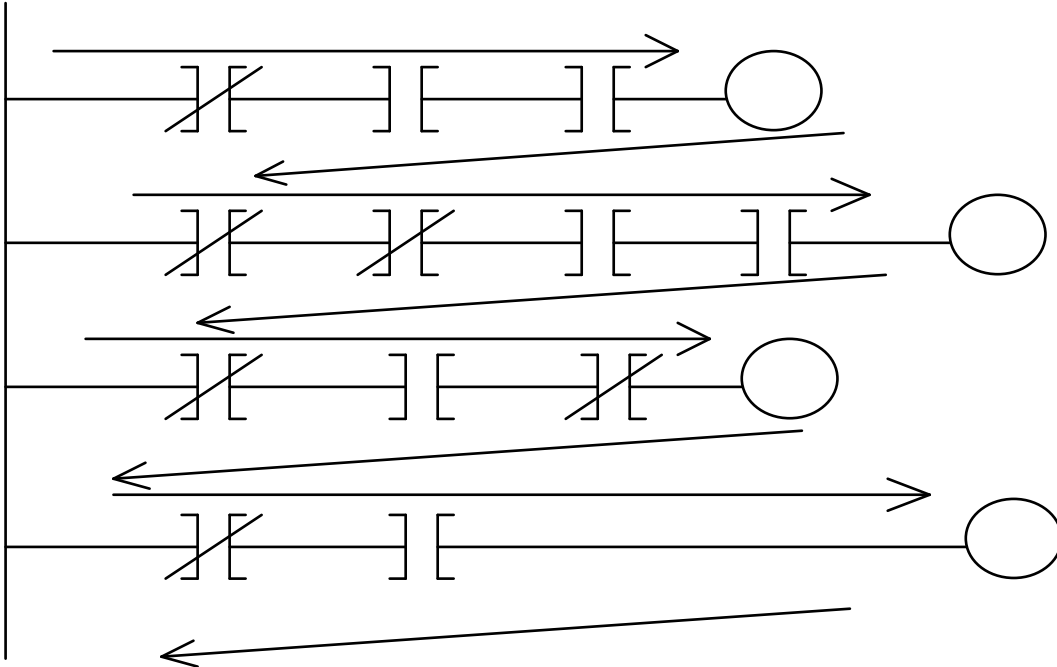
Modicon PLC's actually write the program one page at a time and scan the ladder logic column by column as shown in the diagram below.



The picture shows a Modicon Compact 984 (A120) and the PC based programming software (ModSoft) running online from the computer, with one page of the PLC program showing.

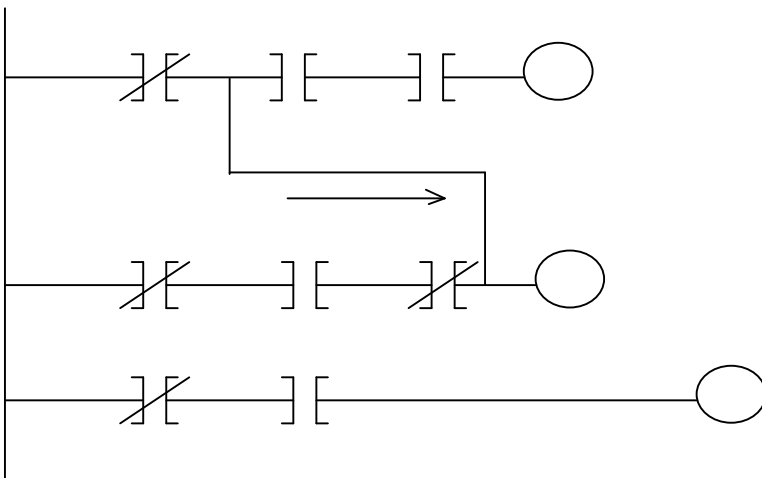


Most other PLC's scan the logic from left to right and top to bottom just like reading a book. **Zeliosoft2** does this convention and this will be the standard for the rest of the course.



Another very important point to note is:

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



Scan or Execution time

OK, getting back to the PLC scan, the processor CPU checks the status of the Inputs from the memory location that represents the inputs.

The application program is then scanned in the appropriate sequence and compared to the data in the various memory locations relating to the inputs, outputs and internal logic values.

From the results of this comparison, the CPU then updates the data tables to reflect what is required by the field inputs and the user program.

The CPU or separate I/O processor then updates the outputs to reflect these changes in the output data table (similar to the input data table).

The internal diagnostics of the PLC then checks what has happened in that particular PLC scan to make sure it 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 etc. 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.

Typically a scan can take from 50 to 500 milliseconds depending on the speed of the processor and the size of the machine and program.

Like all other micro-processor based equipment's the PLC has a strict sequential operational process, that is when it executes its programmes a series of operation are performed inside the PLC. These internal operations can be broadly classified into five categories:-

| Step | Action | Comment |
|------|--|---|
| 1 | Execution of initial power-up processes | Self diagnostic checks:- I/O modules, Internal relays, resets all timers, comms to peripheral devices, memories, hardware configuration, power supplies and user program Time taken for this activity is not critical |
| 2 | Execution of common processes | Resets the watch-dog timer, checks User Program memory and I/O bus – this is all part of self diagnostics. Time taken for this activity is fixed. For the C20 it is 1.07mS |
| 3 | Input and Output data execution | Reads status of:- Input channels and stores info in an Image Table or Memory , Writes Image Table or Memory info to all Output channels Time dependent upon number of I/O. For the C20 it is 1.04mS |
| 4 | Application program execution | Reads the user program step by step, making changes as needed due to I/O memories and Internal relay status. Time dependent upon number of steps in the program and the functions used |
| 5 | Servicing peripheral devices | Reads and writes information to peripheral devices Time taken for this activity is fixed. It is approx 1.1mS |

Notes

- 1 Step 1 is only carried out when power is first applied to the PLC.
- 2 Steps 2 to 5 repeat on a cyclic basis and form what is known as the **Scan Time** of the PLC, and in some applications where the external process has some very short timing operations, the scan time may be critical to the successful operation of the external process, it may therefore be necessary to calculate the scan time of the PLC.
- 3 Not considered in the above table is the time taken for the signal from an Input device to be recognised by the input module or the step of the cycle when this occurs. This too may be critical for the successful operation of the external process and would need to be investigated.
- 4 User program check is internal only, i.e. format and instruction usage only.

INPUT / OUTPUT (I/O)

INPUT/OUTPUT MODULES

- Provide interface & signal conditioning between external field devices & the μ P I/O data bus.
- Most manufacturers use opto-isolation in I/O modules to ensure that the μ P environment is protected against voltage spikes etc.
- 0-10V & 0-20mA.
- Typical inputs: push button, limit, proximity, float, pressure, or temperature switch types, & analogue signals.
- Typical outputs: solenoids, motor starters, alarms, visual indicators, or analogue signals.

Input and Output devices can be any of the following:

| INPUT DEVICES (many of these are known as transducers*) | |
|--|---------------------|
| STOP/START BUTTONS | THERMAL OVERLOAD |
| LIMIT SWITCH* | PRESSURE SWITCH* |
| FLOAT SWITCH* | TEMPERATURE SENSOR* |
| FLOW SWITCH* | LIGHT SENSOR* |
| LEVEL SWITCH* | PROXIMITY SWITCH* |
| Input devices can also be an analogue output from a current transformer (CT)* or perhaps a voltage generated by a tachometer* | |

| OUTPUT DEVICES | |
|--|------------------|
| CONTACTOR | SOLENOID (VALVE) |
| INDICATOR LAMP | ALARM |
| Output devices can also be an analogue output to an ammeter or some other device that requires an analogue input | |

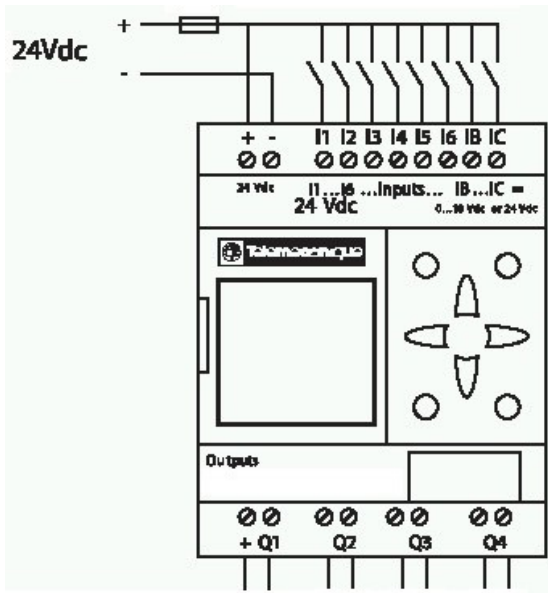
Typically a “shoebox” PLC will have terminals on one side dedicated to the connection of input devices, and terminals on the other side dedicated to output devices.

As is the case with the **Zelio-Logic Smart Relay SR1 B121BD** (8 inputs and 4 outputs). It is also not common for these smaller PLC’s to have either analogue Input or Output however the **SR1 B121BD** does have 2 analogue inputs.

INPUT

Typically input devices are connected between a common point at the PLC and a discrete input channel. This means that each input device (a limit switch etc) can be connected to the PLC with a 2-core cable.

The diagram below clearly shows the connection schematic (it does not clearly show the 2 core wiring system).



Input signals are referred to either as “digital” or “analogue”, and therefore at the PLC need to be connected to a compatible input channel. The diagram above shows 8 inputs, each with a unique alpha-numeric code, and in this case, “I1” to “IC”. The “I” stands for “input” with channels I1 to I6 being digital, while channels IB and IC being analogue inputs which can also be used as digital.

Input voltages can be a.c. or d.c. depending on the specifications of the PLC and are typically in the range of 5Vd.c. to 230Va.c. for digital and either 0v to 10v or 0 to 20mA for analogue inputs.

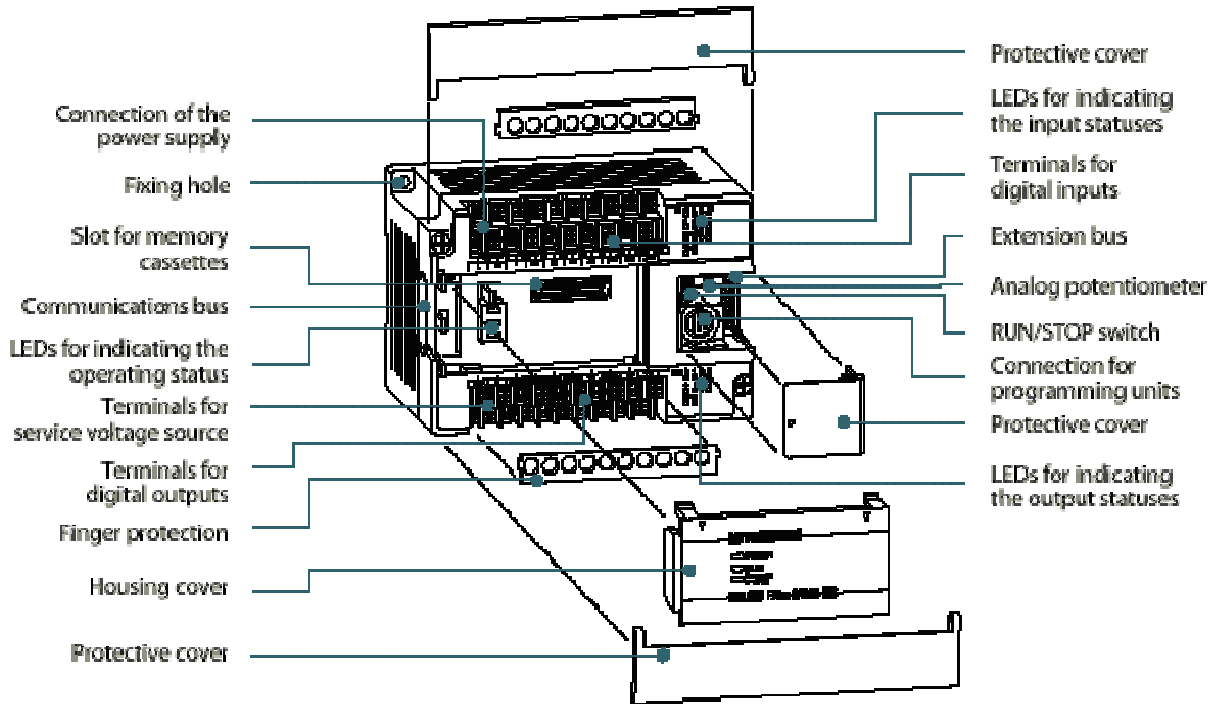
Input channels convert input signals to DC (where necessary) signals. These signals are then isolated from the PLC electronics by “opto-couplers”. These signals are then modified to suit the type of d.c. voltage which can be processed by the CPU (the microprocessor) – typically 5V.

OUTPUT

Outputs are typically designated by an alpha-numeric code also.

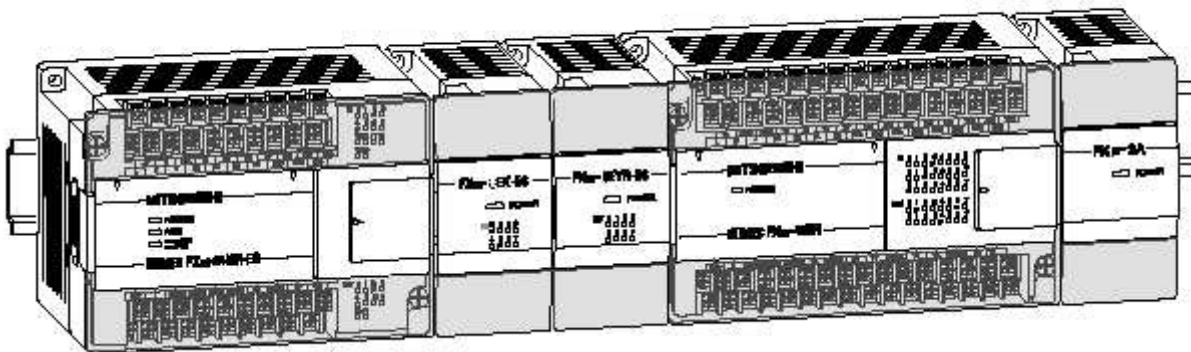
PLC’s are not capable of switching (or supplying) high currents to say a motor, but are typically quite adequate for supplying sufficient current to the coil of a contactor which controls a motor.

The Mitsubishi MELSEC FX_{ON} –24MR-ES has a maximum switching current of 2A (100W or 80VA) per output. It has 10 outputs.



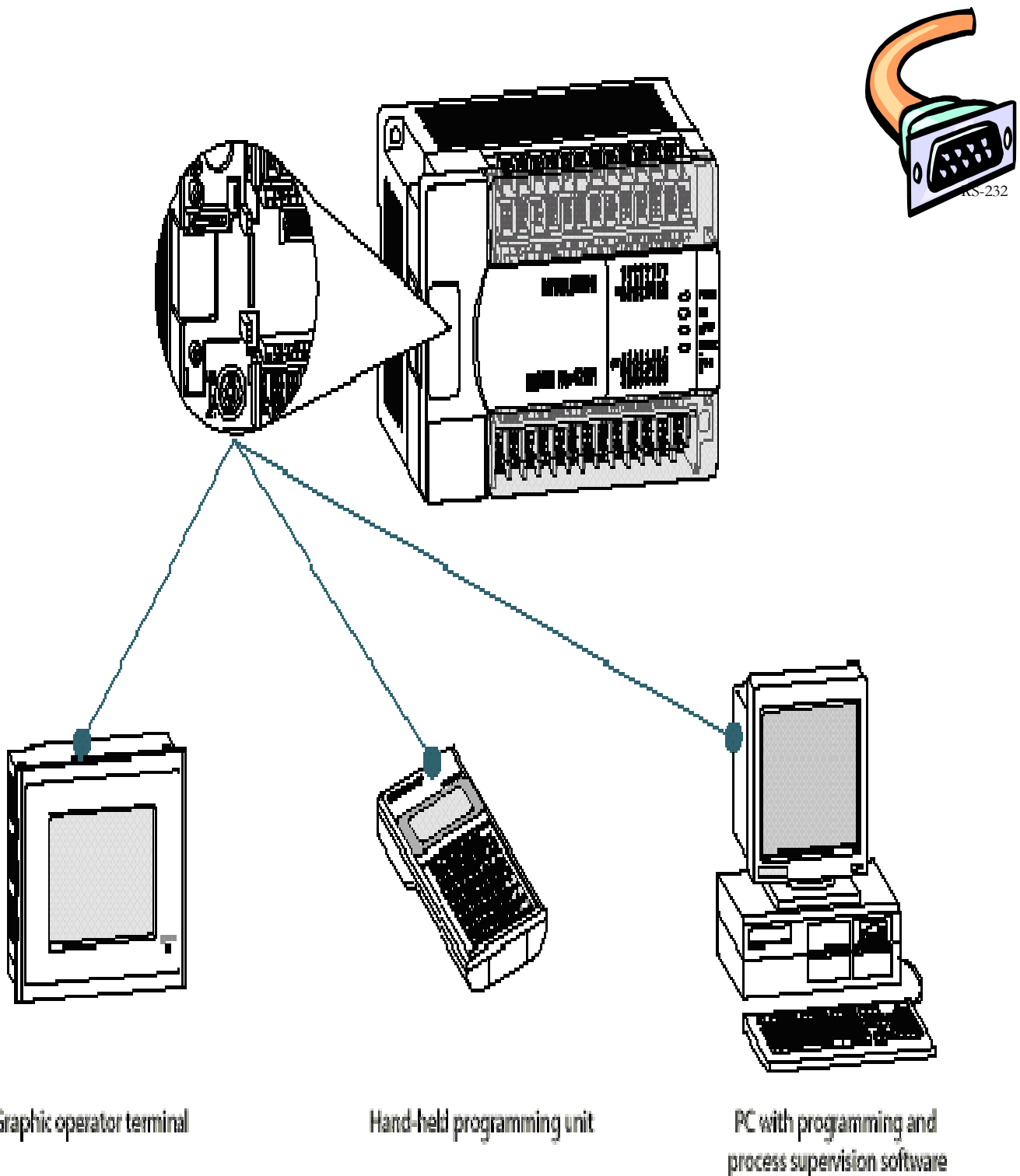
Additional input and output modules can be added to most PLC's when more I/O is required. These can include digital (standard inputs either ON or OFF), analogue (respond to varying voltage/current), and other specialised inputs such as communication, temperature sensing (PT100), and high-speed counters (positioning).

Configuration Example FX0N



The diagram above shows 2 PLC's, one auxiliary input module, one auxiliary output module and a special function input module on a DIN rail mount.

Next we'll look at programming methods



Graphic operator terminal

Hand-held programming unit

PC with programming and process supervision software

PROGRAMMING METHODS

PROGRAMMING DEVICES

DEDICATED HAND HELD PROGRAMMER

Early PLC's 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 of the location of PLC's, generally inside cabinets, the manufacturers developed their own versions of programming devices that could be attached via cables so the programmers could perform their programming tasks in relative comfort.

An example of this is the Allen Bradley hand held programmer shown at right. This unit could attach via cable to the AB SLC 100 and SLC 150 PLC's.



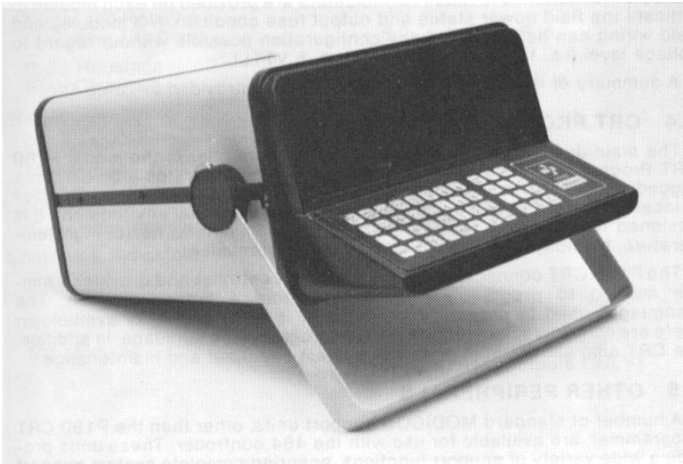
Over a period of time the hand-held programmer (device) has become more sophisticated with much more easily followed instruction sets and keyboards and can have LCD displays that show several rungs of logic at once. These devices however, can only access the actual PLC program stored in the machine itself. To back up your program you need another device. Some have jack points to attach a tape recorder, but these are probably the exception not the rule.



This "hand held" programmer 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.

DEDICATED PROGRAMMER – LOADER

As previously mentioned a lot of the early PLC's could only be programmed by the use of dedicated devices that were virtually a part of the PLC itself. With the increased popularity that followed most companies developed programming terminals that were dedicated to a particular PLC or range of PLC's. They also developed the capability of "Loading" the program onto tape from the PLC and loading the program from tape to program the PLC. This ability gave rise to the development of Off-line programming.



This development allowed engineers and others to write a program and only spend a few minutes downloading it to the PLC memory therefore saving hours of downtime that would have been required in programming the PLC directly.

Later dedicated programmers are in fact industrial versions of personal computers and generally have moved from tape to floppy disks for storing the programs etc.

These devices also included "centronics or serial" printer ports to allow printouts of the PLC user programs and PLC status, configuration etc.

PERSONAL COMPUTER

With the huge advances in Personal Computer technology and the development of Laptop and Notebook computers, the dedicated programmer met its match.

Initially the programs used on the PC looked and operated very much like the dedicated programmers.



The Allen Bradley SLC100 training rig shown can be programmed by the hand terminal pictured or by the Personal Computer as shown.

The newer software packages are generally far superior to most dedicated programmers in ease of operation and capability.

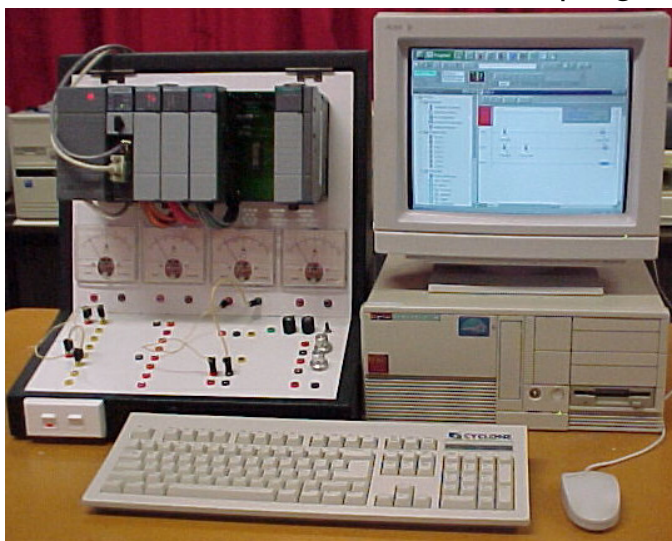


(<Left) Modicon's ModSoft DOS based software programming a Modicon 984 compact PLC.

(Below) Omron's Syswin Windows 3.x / 95 based software programming a CPM PLC training rig.

Also note that the prices of PC's have fallen dramatically, while the dedicated programmer price has remained relatively high.

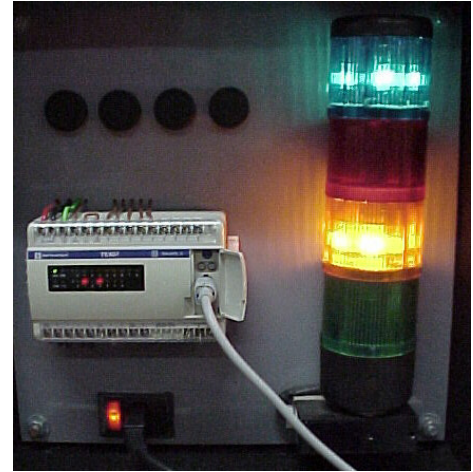
The advent of personal computers being used in large numbers has also allowed economies in scale of producing more advanced software for PLC programming.



(Left) Allen Bradley SLC500 Training rig, with RSLogix 500 Windows NT/95 programming software.

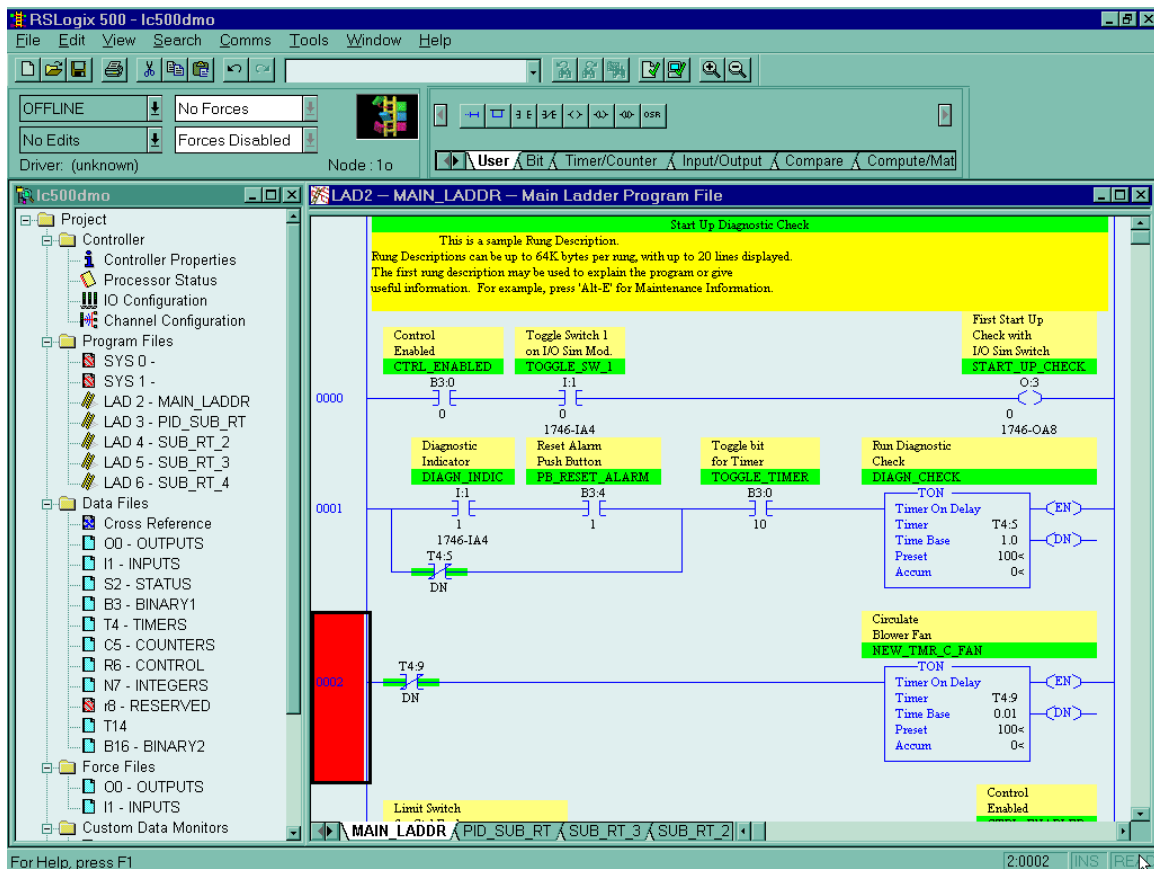


(Left) Omron SP16 mini PLC and (Below) a NANO PLC (Demo unit ex Schneider NZ).



These PLC's are a bit of an anomaly, in that although they are relatively recent (especially the NANO) the programming software is very basic and is still DOS based.

(Below) Screen shot from Rockwell Software's RSLogix 500 ladder programming software for micrologix and SLC 500 series PLC's.



PROGRAM / LANGUAGE - SYSTEMS

LADDER LOGIC

Ladder logic programming came from the PLC's origin, in that they were developed as replacements for relay logic.

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

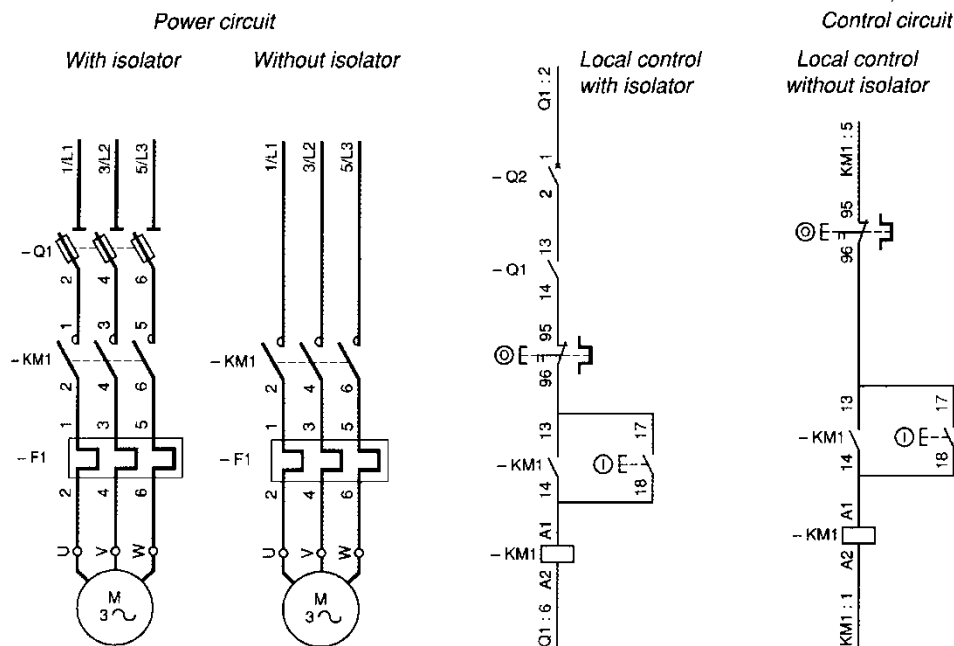


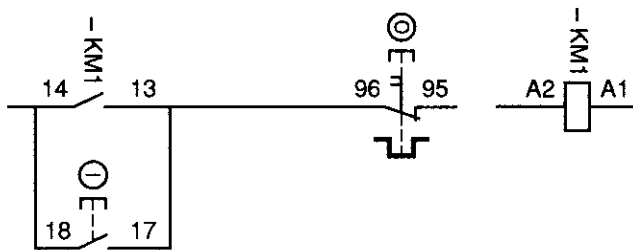
Diagram Courtesy of Groupe Schneider. From their Book "Practical aspects of industrial control technology" Telemecanique technical collection.

Now let us look at how the hard wired logic diagram can be converted to a PLC Ladder logic program

For ease of understanding the control section of the above diagram has been turned 90degrees and had the "output" coil shifted to the right hand end, to follow PLC convention

Each element is labelled to identify which device it actually 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.



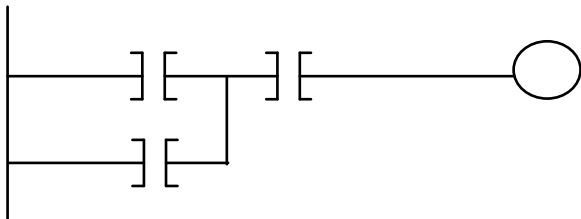
When converting this to a PLC ladder logic program you must look at things a little differently. As you will recall the Input modules on the PLC are monitored to see what the current state of the real world inputs actually are.

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. So it is in the PLC logic.

The N/O contact we refer to as XIC (examine if closed). When the input is live the “PLC input relay” is active (Closed) and all N/O contacts associated with it will be closed.

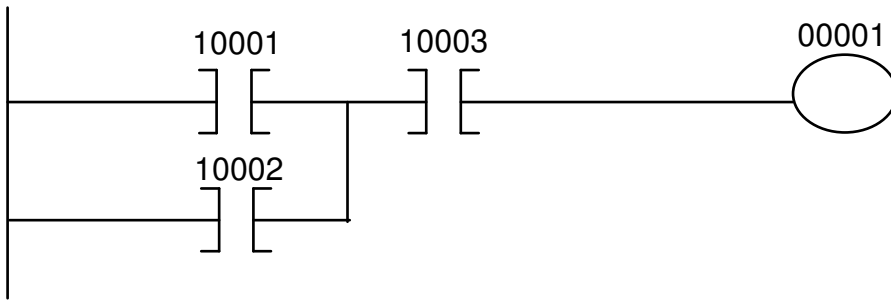
The N/C contact we refer to as XIO (examine if open). When the input is NOT live the “PLC input relay” is not active (Open) and all N/C contacts associated with it will be closed.

Subsequently the equivalent PLC logic for the above starter would be as follows.



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.

E.G.



The particular numbering convention used depends on the particular PLC.

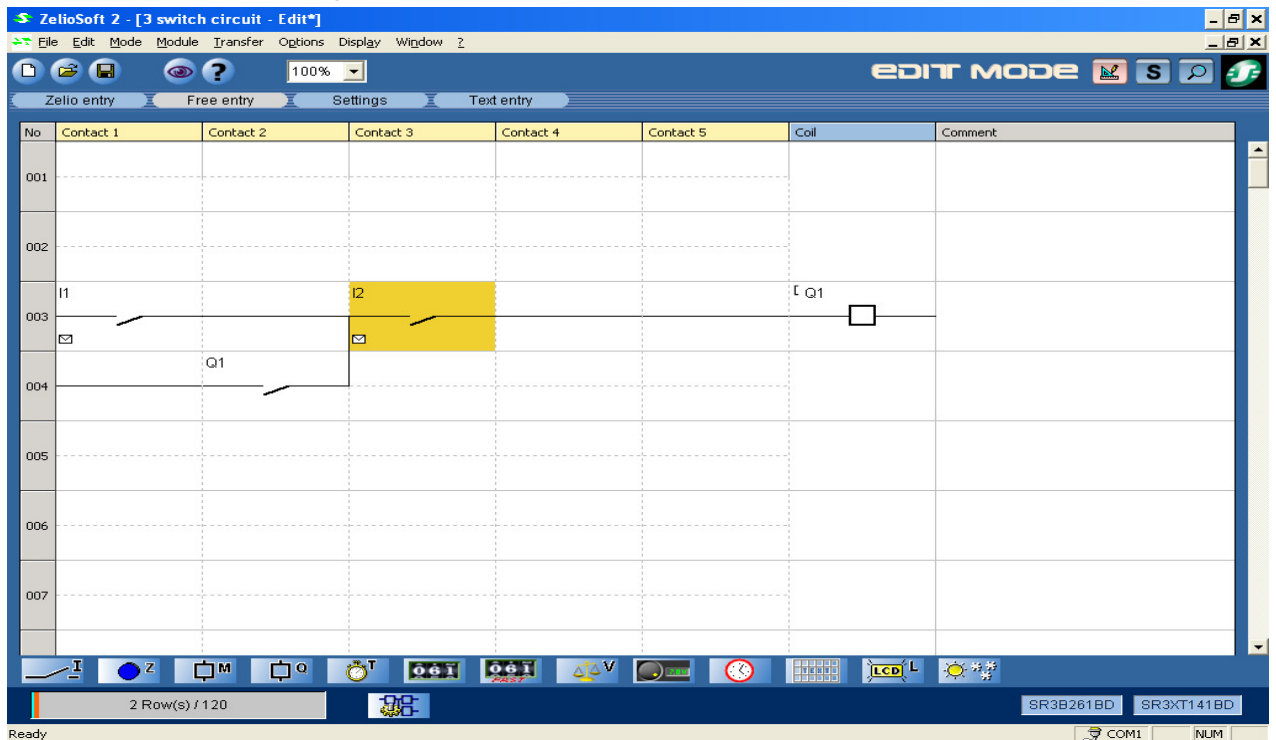
This addressing forms part of your as-built documentation and you can print the ladder listing whenever you wish.

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 |

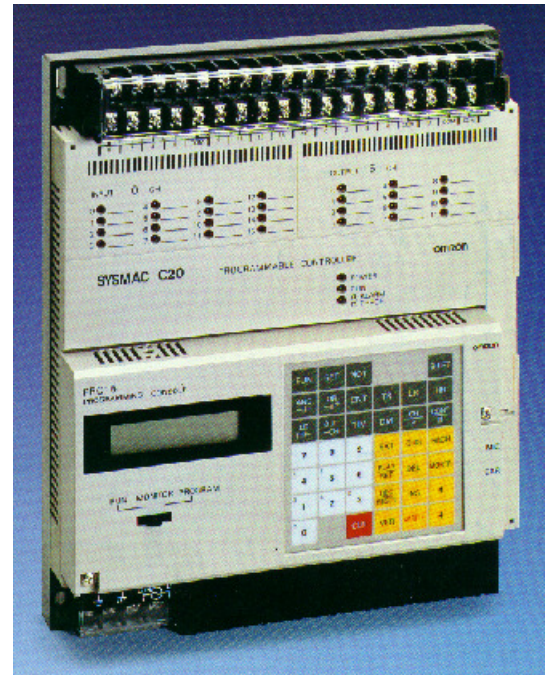
Here's a screen shot using Zeliosoft2 for the similar circuit



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 either a dedicated programmer/ loader or a keypad attached to the front of their controllers. A large number of companies were crying out for the technology to do their own programming with a simple transportable programming terminal. The dedicated programming devices that you could buy were very expensive and could not safely be used in many industrial environments.



By using sealed keypads, LED displays and more recently, LCD displays, small, dedicated programming devices were created.

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 here attached to a C20 PLC). Later versions attached via a short flexible cable.

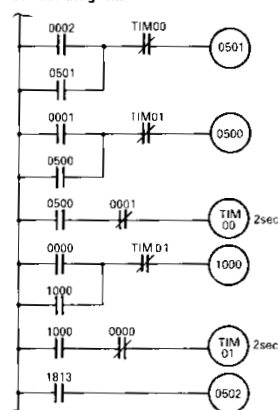
While there are many variations between manufacturers, the basic instructions were very much the same.

The following shows a program written in “Instruction List” and the equivalent “Ladder Logic.”

Coding chart

| Address | Instruction | Data |
|---------|-------------|-------|
| 0200 | LD | 0002 |
| 0201 | OR | 0501 |
| 0202 | AND-NOT | TIM00 |
| 0203 | OUT | 0501 |
| 0204 | LD | 0001 |
| 0205 | OR | 0500 |
| 0206 | AND-NOT | TIM01 |
| 0207 | OUT | 0500 |
| 0208 | LD | 0500 |
| 0209 | AND-NOT | 0001 |
| 0210 | TIM | 00 |
| | | #0020 |
| 0211 | LD | 0000 |
| 0212 | OR | 1000 |
| 0213 | AND-NOT | TIM01 |
| 0214 | OUT | 1000 |
| 0215 | LD | 1000 |
| 0216 | AND-NOT | 0000 |
| 0217 | TIM | 01 |
| | | #0020 |
| 0218 | LD | 1813 |
| 0219 | OUT | 0502 |

Ladder diagram



This example is taken from the Omron C20 programming manual, issued free with each C20 PLC.

Ladder Logic

LADDER LOGIC AND DIAGRAMS

In heavy electrical designs, it is more common to find ladder network diagrams used to describe logic sequencing than Boolean logic symbols & gates.

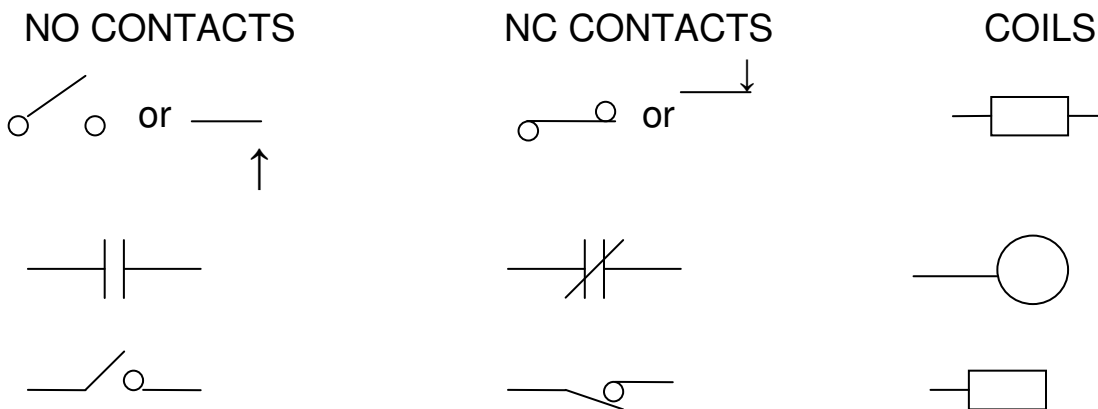
In mathematical terms however there is direct ladder logic equivalence for the AND, OR, NOT, NAND & NOR functions.

PLC ladder logic also frequently includes counter, timer, data shifting & other slightly more complex circuit blocks.

Relay ladder logic was (& is still) used by industrial electricians for plant installation, & servicing. The number of each contact normally appears alongside it, as does information regarding the output coils being activated.

PLC sequencing design can be done using mnemonic instruction logic (a little like a high level μ P assembly language) or by extended use of ladder logic diagrams. Generally ladder diagrams are eventually required by installation & maintenance electricians anyway, as mentioned above. In this case, contacts & relay outputs are drawn using the same symbols but a PLC channel number is shown alongside each.

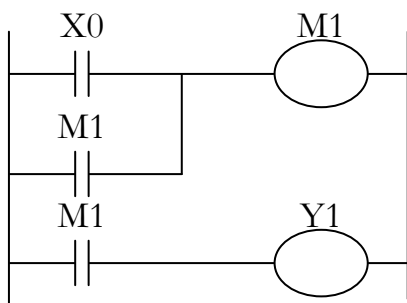
The most popular symbols are shown below. The IEC standard symbols are shown along the top row.



(Note: NO denotes normally open & NC denotes normally closed.)

The basic rules for ladder logic may be summarized as follows:

- A live common vertical bus is provided along the LHS of the diagram. All horizontal logic lines are derived from this.
- Each line starting from the left bus must end in a coil or several parallel coils.
- Output coils may have as many auxiliary contacts as required. Each auxiliary contact may be configured as normally open or normally closed.
- Each output coil can appear only once in a program.
- A designated contact must not be placed to the right of a coil.
- All contacts & coils are numbered to indicate which device (or PLC I/O channel) they are associated with.
- PLC output coils may be programmed as contacts, but contacts can not be programmed as output coils.
- With PLC ladder logic, output devices may be shown as some variation of a simple numbered circle. This can be used to indicate a counter, timer or other logical device.
- In PLC logic, temporary relays (TR) or internal memories (IM) are often required for branch connection of intermediate logic points into other logic arms.
- PLC output coils may be programmed as contacts, but contacts can not be programmed as output coils.
- With PLC ladder logic, output devices may be shown as some variation of a simple numbered circle. This can be used to indicate a counter, timer or other logical device.
- In PLC logic, temporary relays (TR) or internal memories (IM) are often required for branch connection of intermediate logic points into other logic arms.



In this example of ladder logic:
If the “normally open” input “X0” is closed, internal relay “M1” will be energised. Both contacts associated with M1 will close and hold in relay M1 and energise output “Y1”.

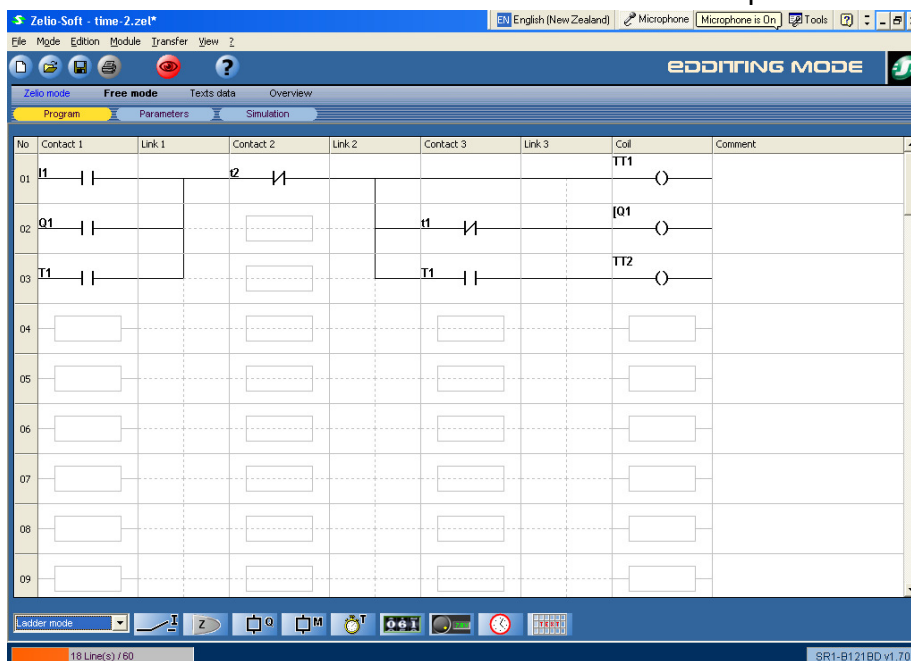
PLC Programming Simulation Software

In this Unit we will be using a Windows based software package called **ZelioSoft2** which has been provided to you by Schneider Electric (NZ) Ltd.

You can also download this from www.schneiderelectric.ie along with other Zelio resources.

We will be writing the program on a PC using “Ladder Logic” as illustrated by the screenshot below. After writing, we will simulate the operation and once this confirms the control strategy the program is ready to download to the PLC.

We will produce PLC programs from simple ON/OFF operations to programs including counters and timers. To find out about each element use the Help? function.



Study the above ladder diagram, and mark the ladder language elements you can find or where to find them such as:-

- | | |
|------------------------------------|-----------------------------|
| 1. Discrete Inputs | 12. Analog Comparators |
| 2. Discrete Outputs | 13. Texts |
| 3. Modbus Inputs/Outputs | 14. LCD Screen Backlighting |
| 4. Auxiliary Relays | |
| 5. Zx Keys | |
| 6. Counters | |
| 7. Counter Comparator | |
| 8. Fast Counter | |
| 9. Clocks | |
| 10. Change to Summer / Winter Time | |
| 11. Timers | |

PLC TERMS

The following are terms associated with PLC's in everyday use. The list is by no means exhaustive and there are several variants from different manufacturers.

While the PLC's we are generally referencing in this workbook are from 3 or 4 major manufacturers, the hundreds of different brands and models leave us with a vast array, of sometimes, conflicting terms.

In the end when in doubt, you just have to RTFM.


No one can hope to know all there is to know across all brands. The best you can be confident of achieving in PLC programming is to be conversant with a few brands and models.

The various industry experts tend to specialise in one or two brands, and need to read the manuals as they go when they step out of this "comfort zone". However when you are up to speed with the principles of programming, you can usually figure out how to cross platforms relatively easily by using the manufacturers manuals.

THE FOLLOWING TABLES LIST THE TERMS SPECIFICALLY MENTIONED FOR YOU TO BE ABLE TO EXPLAIN AS THE RANGE STATEMENT FOR ELEMENT 1 PC 6 FOR THIS UNIT (5926 V3).

Study them and **briefly** explain what each term means include any symbol. The first term "examine on" is done to show what is required.

BASIC INSTRUCTIONS

| | | | |
|----------------------------|---|--|--|
| Examine on / XIC Or N/O |  | Examine beyond the instruction if the associated input is on. NOTE: do not confuse with the Allen Bradley convention XIO (examine if open) this is actually an XIC (examine if closed) | |
| Examine off | | | |
| And | | | |
| Not | | | |
| Or | | | |

STANDARD INSTRUCTIONS

| | | | |
|---------------------------------|--|--|--|
| Set | | | |
| Reset | | | |
| Function Libraries | | | |
| Up Counter | | | |
| Down Counter | | | |
| Self Resetting | | | |
| Cycling Counter | | | |
| ON | | | |
| Off | | | |
| Accumulation | | | |
| Timers TON / TOF | | | |
| Retentive timers RTO | | | |

DATA MANIPULATION

| | | | |
|-------------------------------|--|--|--|
| Time Driven Sequencer | | | |
| Event Driven Sequencer | | | |
| Data Bits | | | |
| Control Bits | | | |

SPECIAL INSTRUCTIONS

| | | | |
|----------------------------------|--|--|--|
| Retentive Bits | | | |
| Power Down Retentive Bits | | | |
| Transitional Contacts | | | |
| Internal relays | | | |
| Coils | | | |
| Flags | | | |

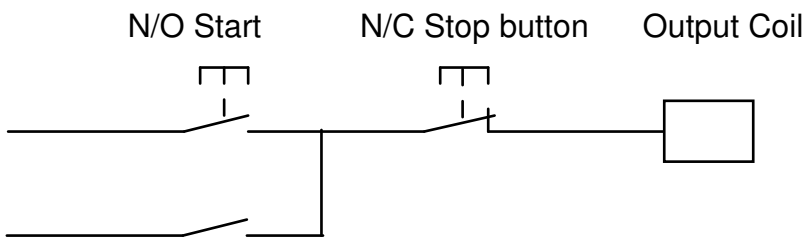
EMERGENCY STOP PROGRAMMING CONVENTIONS

When we hard-wire an Emergency Stop button into a control circuit, the normal convention is to wire a Normally Closed contact as the stop. This practice allows a stop condition as fail-safe. That is, if a wire falls off or the contacts corrode and no longer conduct, the system will see the fault (Fail) as a stop condition (Safe).

SPECIAL CONSIDERATIONS

When we wire this fail-safe stop button into the PLC it is effectively inverted in the logic. Compare the following diagrams.

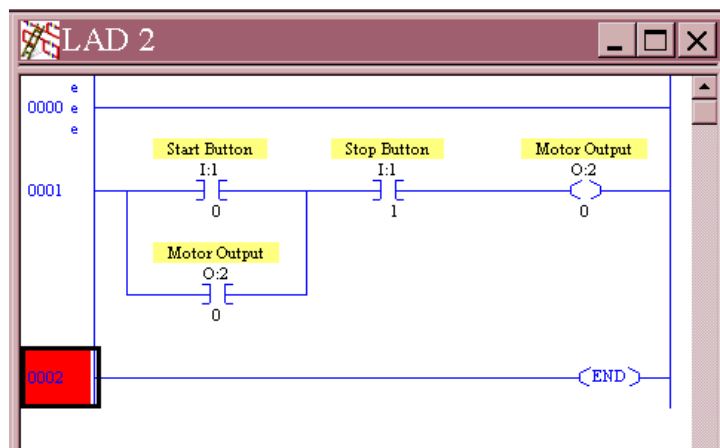
A part wiring diagram for a simple DOL motor starter, showing the use of a normally open start button and a normally closed stop button.



The following is the same part DOL starter as it would be programmed using PLC ladder logic.

This is taken from RS Logix 500 programming software

Note the Stop Button is programmed using an XIO instruction the same as the start button.



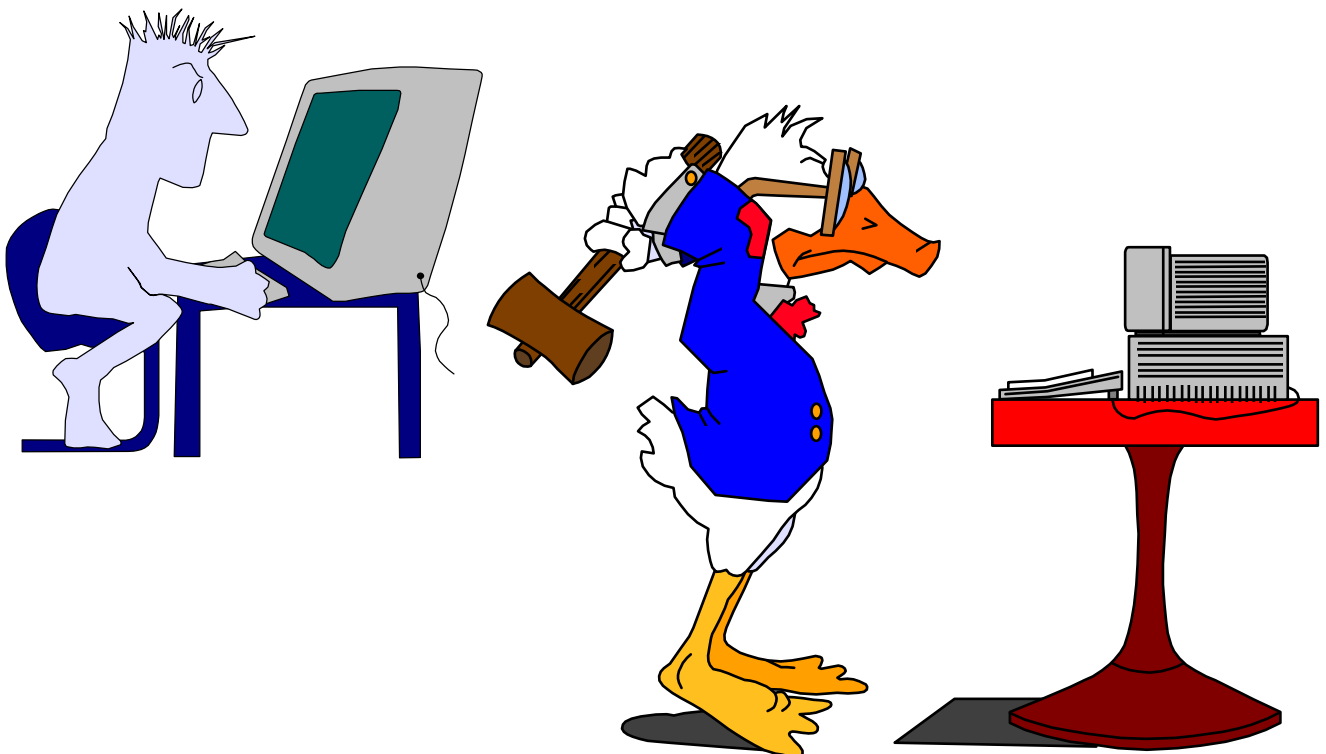
If you think about how an input is read into the input image table, you will recall that each input is treated as if it were the signal feeding the coil of a control relay. This means that each time it is shown in the logic we must remember that you are looking at the equivalent of a contact on the input "Software" relay. Unsure? Try programming the above circuit & test it.

OK You should now have a good understanding of the theory of PLC operation.

The next Section covers the requirements of your programming task. If you are unsure of anything we have covered to this point, study the section in the workbook and ask questions. Your tutor is there to help you.



You may attempt the programming task on any PLC or simulator package that you feel comfortable with.



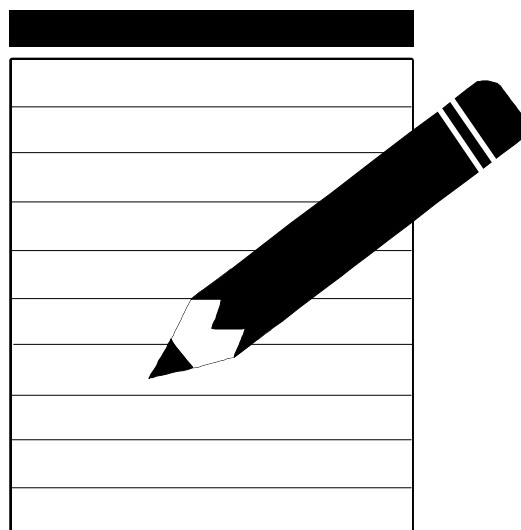
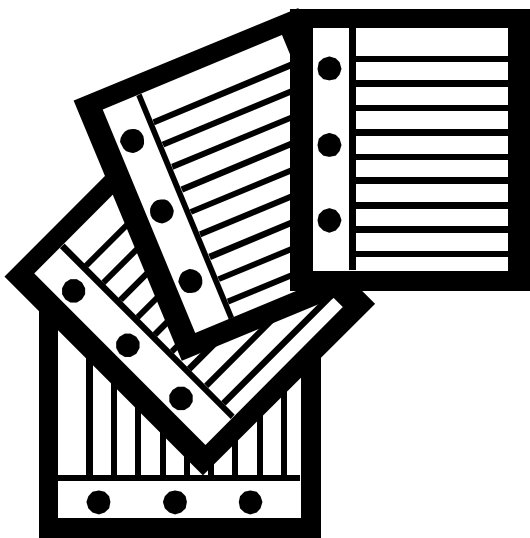
Element 2 – Design, Write and store PLC programs.

THE TASKS

These tasks are your assessment for element 2 of this unit. Carefully study the requirements and note the tips given over the next few pages.

- Design your programs using any method you are familiar with e.g. Flow chart. You may use the examples in this workbook and the New Zealand standard AS/NZS 4382:1996 handbook to assist you if required.
- Program must meet the specifications.
- Using the PLC Manufacturers manuals (Class sets and on-line Help? functions) to assist you, program the PLC or simulator to complete the automation task you have been given. We shall also program directly into the Smart relay using the front screen panel. (Sometimes we have to perform a minor change in the field but don't have a PC or laptop handy)
- Document your programs (On paper and Online: Flash card)
- Save your programs and documentation to disk and/or Flash card.

When complete check with your tutor for an evaluation. Competency is met when your tutor can follow your logic, and it performs the task requirements. Efficiency of programming is not assessed, however your tutor may comment on efficiency.



TIPS & SUGGESTIONS

PROGRAMMING LANGUAGE RULES

Remember that the flow of logic is from left to right and from top to bottom. The reverse does not work.

Output instructions must always be at the right end of the rung (ladder Diagrams).

Each rung must ultimately have an output instruction.

RTFM

PLC ADDRESSING / SYMBOL USAGE

All your programs should be documented On-line, so you must carefully chose appropriate symbols and labels for all your instructions. Don't forget rung comments etc. to enable others to follow your logic.

It can also be very embarrassing to come back to a program you have written years before and have to work out just what it is doing. Document everything!

PROGRAM ENTRY

Watch how you enter your logic, especially with FBD. Dragging and dropping of icons/Elements within WindowsXP varies depending on the programming package being used. You will need to experiment. If in doubt, RTFM.

PROGRAM VERIFICATION

Make sure you check that the program you produce actually meets the requirements of the task. It can handle all variables expected and not expected. What does the program do if you go outside there parameters? This is why a "paper test" can help solve problems.

SAVING / BACKING UP YOUR PROGRAM

The program you write must be saved. This may include a printout, but saving it to disk or flash card is best. The saved "software" program may be asked to be produced to meet the competency requirements for this element.

Refer Appendix B: Getting started with Zeliosoft2

Self test QUESTIONS

1.1 *Why are PLC's replacing relay logic in industry?*

(list comparisons of PLC and Relay logic panels, and advantages / disadvantages of PLC's in your answer)

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

1.2 *Describe what each of the following PLC components are and how they are used?*

Input device

.....

.....

.....

Output device

.....

.....

.....

Input Interface

.....

.....

.....

Central Processing Unit – CPU – Microprocessor

.....

.....

.....

Memory

.....

.....

.....

Programmer

.....

.....

.....

1.3 Describe what each of the following hardware components are and where they are used.

I/O Devices

.....

.....

.....

.....

.....

I/O Modules (State different types)

.....

.....

.....

.....

.....

I/O signal types (State examples)

.....

.....

.....

.....

.....

High Speed counter input

.....

.....

.....

.....

.....

High Speed Timer

.....

.....

.....

.....

.....

Power supply Module

.....

.....

.....

.....

.....

1.4 *Describe what each of the following Memory terms are, where they are used and whether they are volatile?*

RAM

.....
.....
.....

ROM

.....
.....
.....

PROM

.....
.....
.....

EPROM

.....
.....
.....

EEPROM

.....
.....
.....

1.5 Explain the operating sequence of a typical PLC with reference to the following.

Program Steps – Scan / Execution time – image table – register update – diagnostic checking.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

1.6 Choose two different programming devices and explain how they operate.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

1.7 ***Choose two different programming methods (types of logic) and explain how they operate.***

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

1.8 ***With the aid of a diagram, explain how N/C emergency stop inputs are normally programmed in a PLC.***

.....

.....

.....

.....

.....

.....

Research Assignment: Transducers

(Compulsory for students who wish to gain a cross credit to US5926)

There are a multitude of different transducer types which convert a physical quantity into an electrical value (Voltage or current) that is proportional, or represents this quantity.

The electrical value can then be easily converted into a digital signal (bits) using an ADC (Analogue to Digital Converter) for use by the CPU for processing.

Some physical quantities are: (Students please add the units.....)

Pressure (mBar)

Mass (Kg)

Heat

Light

Strain

Density

Temperature (C, K, F)

Humidity

Speed

Rotation

Velocity

Acceleration

Colour

Flow (gas or liquid)

Magnetic field strength

Length

Width

Height

Others?

Task 2: Taking any 3 different quantities locate typical transducers' information from the Internet: Name, Cost, Typical Use, Input/Output, size etc.

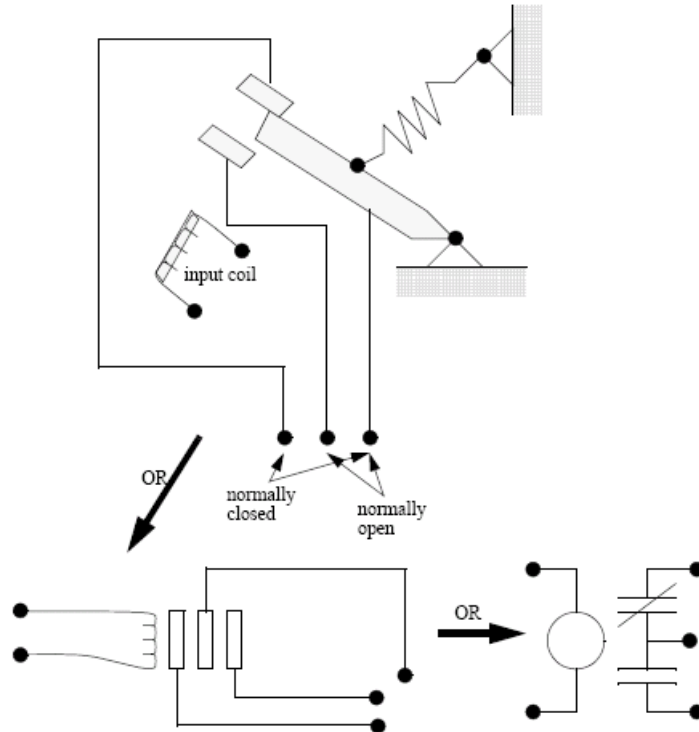
Number of Pages: 1 page cover sheet

1 page per Transducer

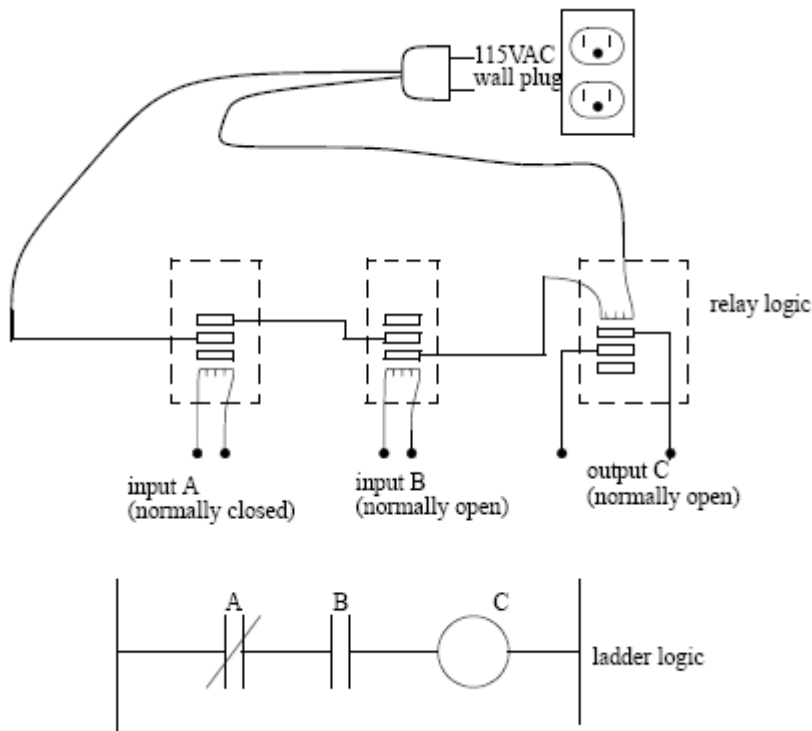
1 page Reference sheet (Include websites used. They will be checked)

Due date:.....

APPENDIX A – Relay problem (1st nights tasks)



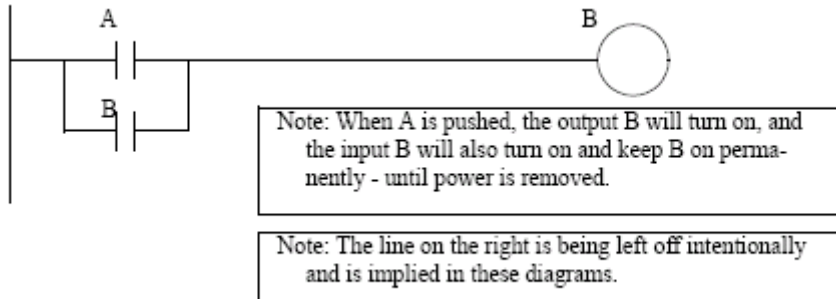
Relay model



Relays can be used as both input switches and as outputs to activate actuators

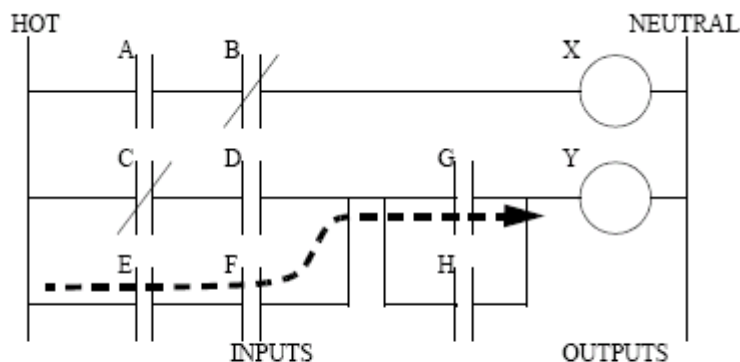
A Holding or Latch circuit

(Has the disadvantage that the Main power must be turned off to reset circuit)



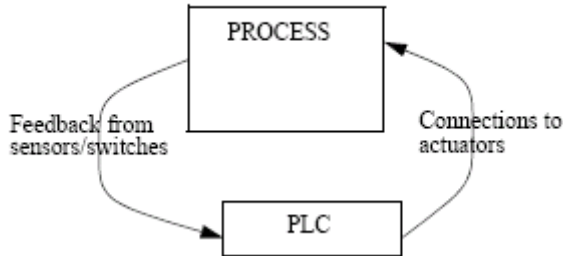
(Tutor will draw up the Seal-In circuit, which is a modified Latch circuit which does not need to be reset. Student to draw below.)

A circuit showing multiple Inputs, Outputs and alternate switching paths

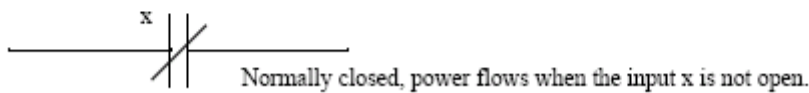
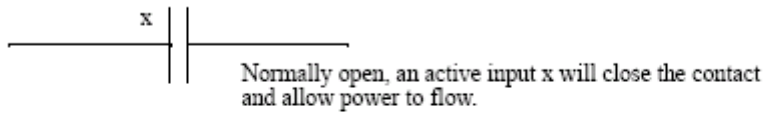


Note: Power needs to flow through some combination of the inputs (A,B,C,D,E,F,G,H) to turn on outputs (X,Y).

Order of operation of relays in ladder logic program



PLC operating sequence for Normally Open (NO) & Normally Closed (NC) relays



Note: When we get into the Zeliosoft2 elements we can also look at the SET and RESET type functions used for smart latching (also for resetting Timers and Counters)

Single Light Problem

Problem: Try to develop a relay based controller that will allow three switches in a room to control a single light. Three separate designs.

- A. The three switches must all be selected to make the light work.
(Series: AND Circuit)
- B. Any of the three switches will turn on the light.
(Parallel: OR Circuit)
- C. Only 1 switch will turn on the light, BUT not if another is on. (THREE INPUT EXCLUSIVE OR Circuit)
- D. From **ANY** previous state, a change of state of any of the switches will change the state of the light. (I.E. any switch will turn light on, any switch will turn the light off).

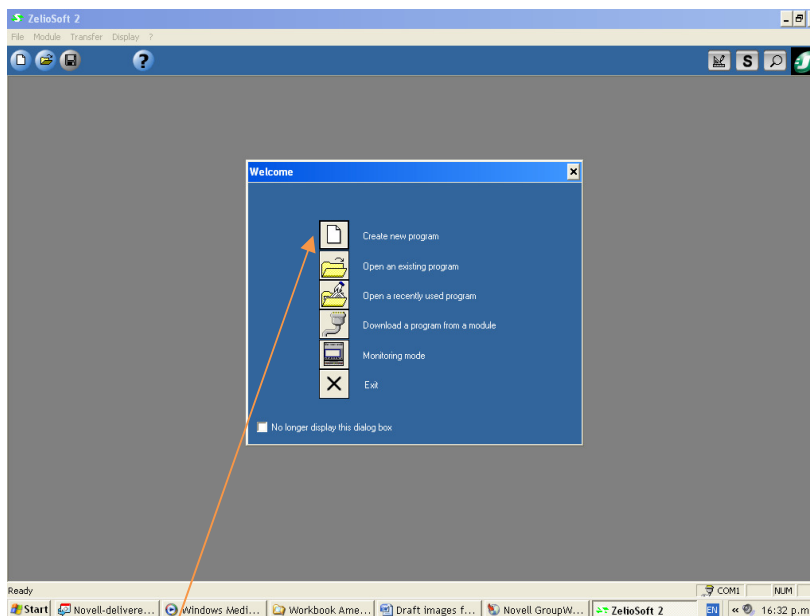
Okay. Now download Zeliosoft2 from www.schneiderelectric.ie (In Ireland) to your home PC.

It's under *Software Downloads* where you'll find this and other resources you can use.

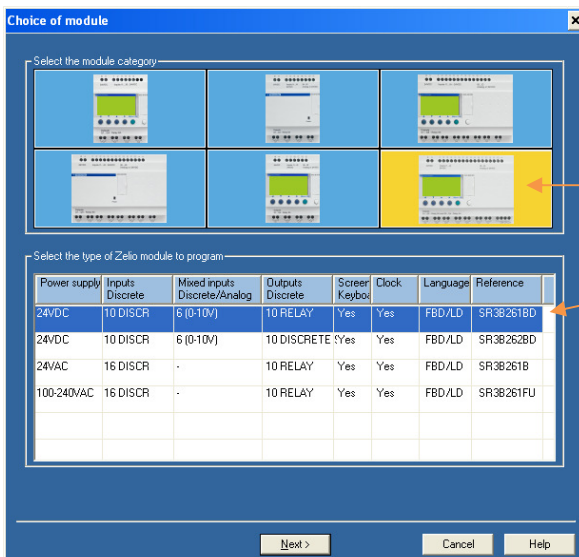
Now, see if you can do the same 3 circuits using Zeliosoft2. You may need to look at the Zelio Tutorial to figure out how to use it first.

APPENDIX B – Getting started with Zeliosoft2

Open Zeliosoft2 (Look for this Icon on your screen)

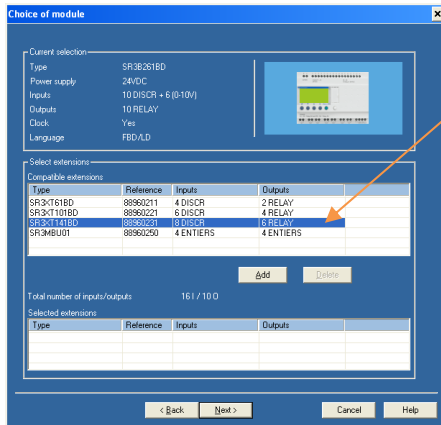


Select *Create new program* (First time user. Open an existing program otherwise)



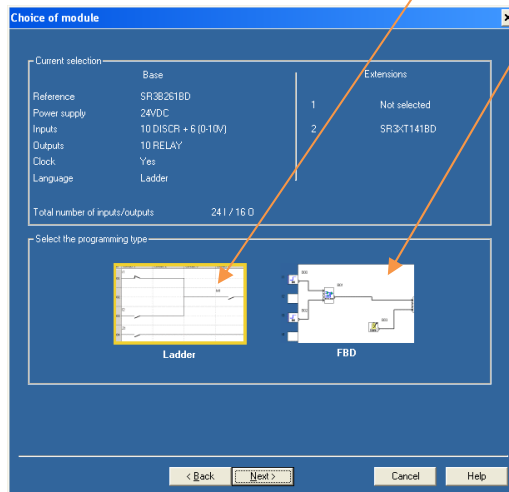
And Select this module, with this option. SR3B261BD, SELECT next>

Now select the Extension module: SR3XT141BD



SELECT add, then
SELECT next>

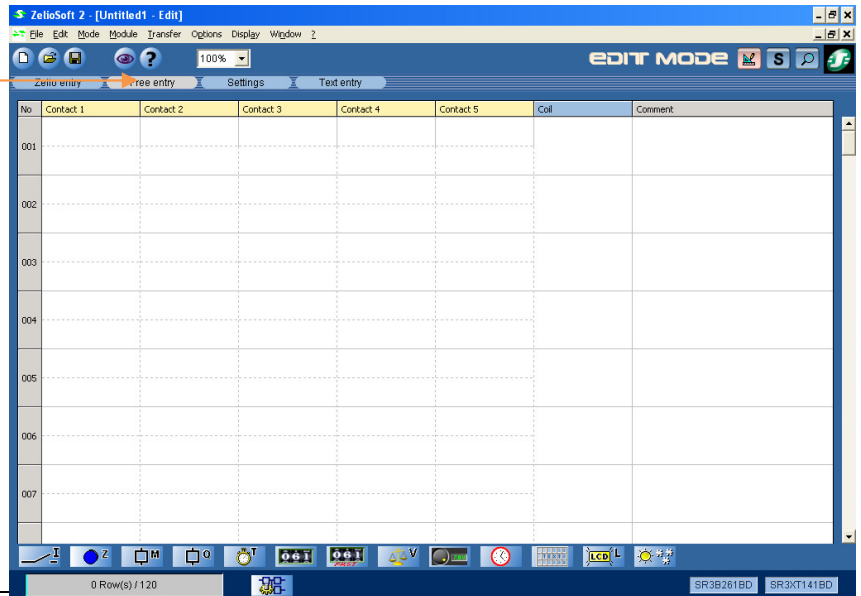
Select Programming type to be used: Ladder or FBD



SELECT next> and the main screen comes up, and you'll be in the **free entry** mode

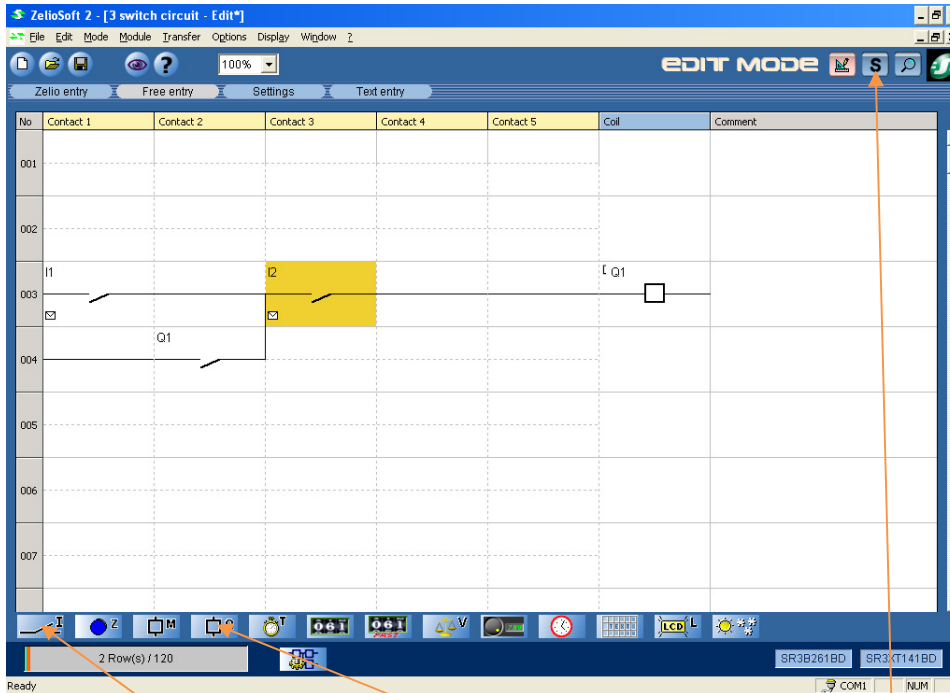
You're ready to create a programming masterpiece.

But, don't forget to design it on paper FIRST, test it for logical errors and then create it in this programme.



Document everything so you don't repeat mistakes, but DO repeat what works

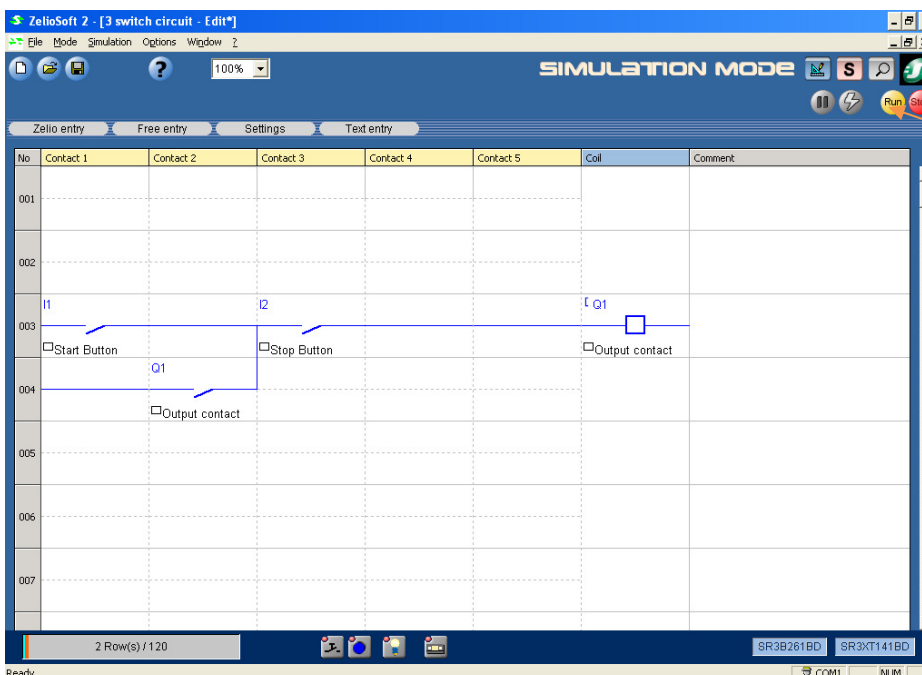
Here is the Seal-In circuit with I1 as the Start contact (NO), I2 is the Stop contact (NO, which closes when the PLC is first powered On)



Discrete Inputs come from here

Discrete outputs from here.

Simulate button here.



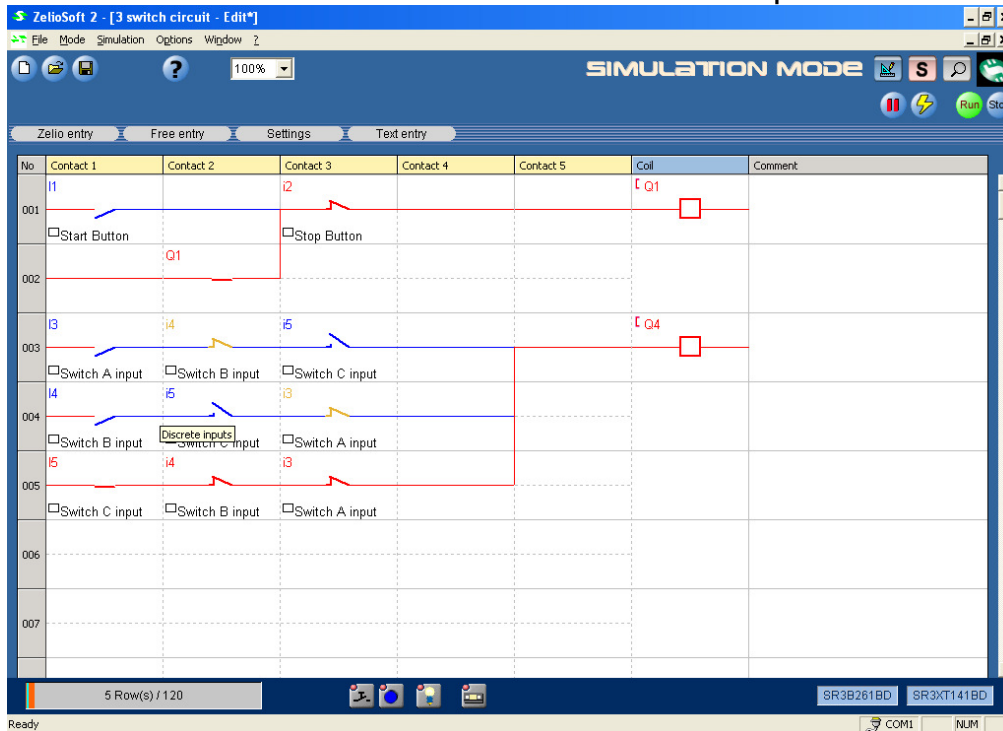
RUN it here

Use the mouse to simulate I2 switching being Normally Closed to opening.

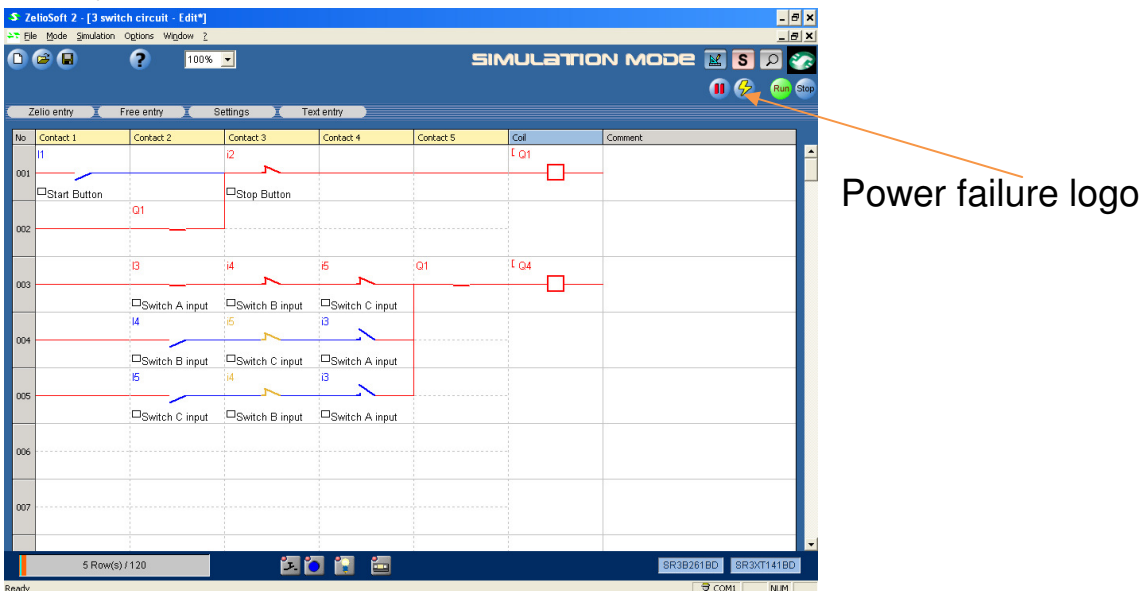
Here is the Seal-In circuit (I1, I2, and Q1)

I3, I4, and I5 are the 3 switches which are connected to produce an Exclusive OR (XOR) circuit to control the output Q1.

I5 is ON and the contacts I5 in the other lines are open.



Here's a further refinement. If you engage the power failure button/logo, if any switch input is still On (I3 or I4 or I5) Q4 will come back On again. By placing Q1 into the circuit this prevents an accidental turn On of Q4 (A safety feature)



The screenshot shows the ZelioSoft 2 software interface in Simulation Mode. The main window displays a ladder logic diagram with five rows of contacts and two coils. The first row (001) contains a normally open contact labeled 'I1' (Start Button), a normally closed contact labeled 'I2' (Stop Button), and a coil labeled 'Q1'. The second row (002) contains a coil labeled 'Q1'. The third row (003) contains three normally open contacts labeled 'I3' (Switch A input), 'I4' (Switch B input), and 'I5' (Switch C input), followed by a coil labeled 'Q4'. The fourth row (004) contains three normally open contacts labeled 'I4' (Switch B input), 'I5' (Switch C input), and 'I3' (Switch A input), followed by a coil labeled 'Q4'. The fifth row (005) contains three normally open contacts labeled 'I5' (Switch C input), 'I4' (Switch B input), and 'I3' (Switch A input), followed by a coil labeled 'Q4'. A 'Discrete outputs' panel is overlaid on the right side of the diagram, showing a grid of 21 output indicators labeled Q1 through QG. The Q1 indicator is currently lit, indicating that the output is active. The software interface includes a menu bar (File, Mode, Simulation, Options, Window), a toolbar with icons for file operations and simulation control, and a status bar at the bottom showing '5 Row(s) / 120' and hardware information (SR3B261BD, SR3XT141BD).

Have fun

Select this icon and you get lights to see what output you'll get.
Look at Inputs (Where do you find this selection?)

APPENDIX C - PLC Programming exercises

(Compulsory for students who wish to gain a cross credit to US5926)

| Program details | Tutor |
|---|-------|
| <p>PROGRAM 1 – NORMALLY OPEN CONTACT This program uses 1 No input and 1 Output ① Lamp is energised when NO input is closed ② Lamp stays on so long as NO input remains closed. Note: Try reversing this and use a NC input to make the lamp stay On.</p> | |
| <p>PROGRAM 2 – AND This program uses 3 NO inputs and 1 Output ① Lamp is energised if all 3 X NO inputs are closed ② Lamp stays on so long as all 3 NO input remains closed.</p> | |
| <p>PROGRAM 3 – OR This program uses 3 NO inputs and 1 Output ① Lamp is energised when any one of 3 X NO inputs are closed ② Lamp stays on so long as any one of NO input remains closed.</p> | |
| <p>PROGRAM 4 – XOR This program uses 2 NO Inputs and 1 Output ① Lamp is energised when only one of the 2 X NO inputs are closed. ② Lamp stays on as long as only one NO input remains closed. Note: If the second NO input is closed the Lamp should de-energise</p> | |
| <p>PROGRAM 5 – SEAL-IN CIRCUIT This program uses 1 NO and 1 nc inputs and 1 Output Build on program 1- 4 ① Lamp is energised when NO input is momentarily closed (i.e. closed and then opened) ②Lamp remains ON even if the NO input is open ③Lamp is de-energised when a second NC input is momentarily opened (i.e. opened then closed). Note1: As long as the NC input is kept open, the Lamp should not to energise even when the NO input is closed.</p> | |
| <p>PROGRAM 6 – DELAY ON This program uses 1 no and 1 nc input and 1 Output Build on program 5 ① Lamp is energised 3 seconds after NO input is momentarily closed (i.e. closed then opened) ②Lamp remains ON even when the NO input is opened ③Lamp de-energises when a second NC input is momentarily opened (i.e. opened then closed). Note1: As long as the NC input is kept open, the Lamp should not energise even when the NO input is closed.</p> | |
| <p>PROGRAM 7 – DELAY ON DELAY OFF This program uses 1 no and 1 nc input and 1 Output Build on program 6</p> | |

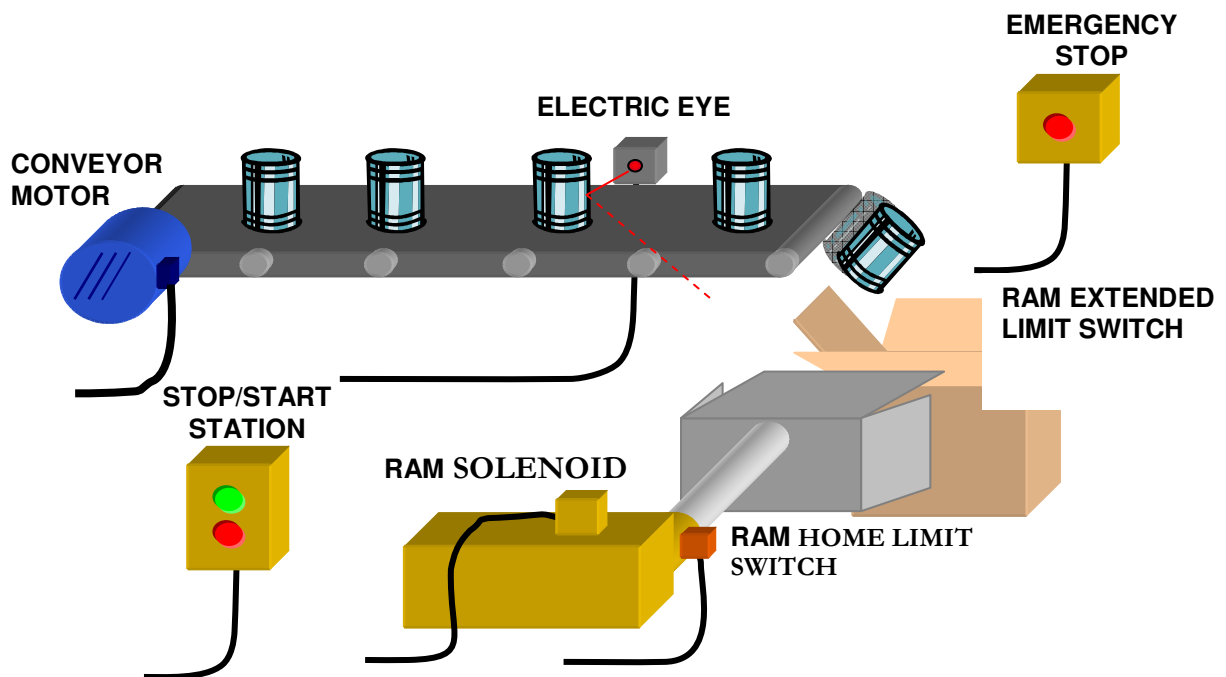
| | |
|---|--|
| <p>① Lamp is energised 3 seconds after NO input is momentarily closed (closed then opened) ② Lamp de-energises 4 seconds after it is energised or whenever the NC input is operated (opened then closed).</p> | |
| <p>PROGRAM 8 – CONTINUOUS OPERATION This program uses 1 no (Master Switch) and 1 Output Build on program 7 ① Lamp is energised 3 seconds after Master Switch is ON ② Lamp de-energises 4 seconds after it is energised ③ Step 1 and 2 repeat automatically until Master Switch is OFF NOTE: If Master Switch is opened after a sequence has started the lamp must be ON once in that sequence before coming to a stop and lamp is OFF</p> | |

| | |
|--|--|
| <p>PROGRAM 9 - COUNTERS Build on program 8 This program uses 2 NO inputs (One NO is a Master Switch and the other is a Start Push Button) and 1 Output ① When Master Switch is ON, the following process commence. ② Lamp (Output) is energised 3 seconds after Start PB is momentarily closed (i.e. closed then opened) ③ Lamp de-energises 4 seconds after it is energised ④ Step 1 and 2 repeat automatically until Master Switch is turned OFF (NO input is opened), OR when the cycle has repeated three times. NOTE: If Master Switch is opened after a sequence has started, the lamp must be ON only once in that sequence before the lamp is OFF</p> | |
| <p>PROGRAM 10 – STAR/DELTA STARTING CIRCUIT This program uses 3 NC (One NC input is Emergency Stop PB, the second is Thermal Overload Relay and the third is Stop PB) inputs and 1 NO (Start PB) input and 3 (Main, Star and Delta Contactors) Outputs</p> <p>①The Main contactor and Star contactors are switched ON by the Start PB if the Emergency Stop PB, Stop PB and Thermal Overload relay are not acted. ②The main contactor together with the star contactor are ON and holds on ③Then after 10 seconds the star contactor alone switches OFF and Delta contactor switches ON. ④During the transition from Star to Delta main contactor continues to be ON. ⑤Now the Delta and Main contactors are ON. ⑥All contactors are switched OFF for any opening operation of any one of the 3 NC inputs at any time. ⑦Sequence must double ensure that at any time star and delta contactors cannot be ON at the same time</p> | |

PROGRAM 11 – AUTOMATED CAN SYSTEM

This program uses 2 NO, 3 NC and 3 Outputs

- ① When the Start Button at the Stop/Start station is Momentarily closed, (i.e. closed and opened) the conveyor motor is energised by an Output
- ② Once the conveyor is moving, the cans on that conveyor travel past an electronic eye. The Electronic eye “NO” contact momentarily closes and opens for passing of each Can
- ③ After 10 cans have passed the electric eye, the conveyor runs for another 5 seconds and then stops
- ④ Then the extend coil a Ram is energised and the Ram starts to extend from its “home” position and first closes the card board carton and then push the carton away, to make place for another carton.
- ⑤ The Ram is now fully extended and opens the Ram Extended position limit switch. The Ram extend coil is de-energised.
- ⑥ Now the Ram retract Coil is energised and Ram starts to retract. When it is fully retracted, the Home position limit switch is opened and the Ram retract coil is de-energised
- ⑦ Now the whole process automatically starts again (Start button does not need to be pushed).
- ⑧ A Stop button must be Normally Closed (fail safe), and when operated must stop all actions, at any part of the cycle.
- ⑨ Again the Start button needs to be momentarily closed and the action is initiated again from step ①



Tip: Draw a time-line diagram, or flow chart to assist you here.

