NEAR EAST UNIVERSITY

Faculty of Engineering

Department of Electrical and Electronic Engineering

MECHATRONIC COIN COUNTER

Graduation Project
EE- 400

Student: Amar Adam (20001384)

Supervisor: Asst. Professor
Kadri Bürüncük

Lefkoşa - 2005

AKNOWLEDGMENTS

First, I want to thank Assis.Professor.Dr.Kadri Bürüncük to be my advisor. Under his guidance, I successfully overcome many difficulties and learn a lot about Mechatronics. In each discussion, he explained my questions patiently, and I felt my quick progress from his advises. He always helps me a lot either in this project or other courses. He answers all of my questions quickly and in detail.

Thanks to the lab assistants: Mr.Kamil and Mr.Burhan for their interesting advises in my project and thanks to Faculty of Engineering for having such a good staff in the laboratories, that reflects their high observance for recruitment.

Thanks to my colleagues and friends, especially, Mohamed El-amin and Amir Joda, They really helped a lot in preparation of this project. Thanks to everybody tried to help, my appreciation for them one by one, especially, Omer Ahmed.

Finally, I want to thank my family, especially my parents. Without their endless support and love for me, I would never achieve my current position. And I promise not to stop on this. My goal is their satisfaction.

ABSTRACT

In the last century, we witnessed the revolution of mechanics, industries, and electronics. And while the importance of each one of them is increasing by the needs of human everywhere all over the world, they had to be improved more and more, to have more applications and studies applied to them.

The study that cared about the development of mechanic and electric disciplines was Mechatronics. It was quite useful and made marvelous positive changes in industries and every field depending on mechanics and electronics.

Then, the PLC programming comes to add different influences in the whole field. PLC is a three-letter abbreviation for (Programmable Logic Controller). From its name, we can realize its essence. The PLC is a device which a program can be filled in it and then, it can be connected to an order-receivable mechanical system to control its functions.

The coin counter is one of the many applications that the PLC can be involved in its structure. In this project, the design will include it as an elemental stuff, and it will play basic steps in counting the coins and classifying them.

PLC was the best choice for the system because it is reducing the error-appearing probability. The other equipments can not have the same efficiency of it so; it was the right choice for the design.

TABLE OF CONTENTS

1.1 What is Mechatronics?

Mechatronics is a natural stage in the evolutionary process of modern engineering design. The development of the computer, and then the microcomputer, embedded computers, and associated information technologies and software advances, made mechatronics an imperative in the latter part of the twentieth century. Standing at the threshold of the twenty-first century, with expected advances in integrated bioelectro-mechanical systems, and other unforeseen developments, the future of mechatronics is full of potential and bright possibilities.

1.1.1 Basic Definitions

The definition of mechatronics has evolved since the original definition by the Yasakawa Electric Company. In trademark application documents, Yasakawa defined mechatronics in this way:

The word, mechatronics, is composed of "mecha" from mechanism and the "tronics" from electronics. In other words, technologies and developed products will be incorporating electronics more and more into mechanisms, intimately and organically, and making it impossible to tell where one ends and the other begins.

The definition of mechanics continued to evolve after Yasakawa suggested the original definition. One quoted definition of mechatronics was presented by Harashima, Tomizuka, and Fukada in 1996, in their words mechatronics is defined as:

The synergistic integration of mechanical engineering, with electronics and intelligent computer control in the design and manufacturing of industrial products and processes.

That same year another definition was suggested by Auslander and Kempf:

Mechatronics is the application of complex decision making to the operation of physical systems.

Yet another definition due to Shetty and Kolk appeared in k1997:

Mechatronics is a methodology used for the optimal design of electomechanical products.

A mechatronic system is not just a marriage of electrical and mechanical systems and is more than just a control system; it is a complete integration of all of them.

All of these definitions and statements about mechatronics are accurate and informative, yet each one in and of itself fails to capture the totality of mechatronics. Despite continuing efforts to define mechatronics, to classify mechatronic products, and to develop a standard mechatronics curriculum, a consensus opinion on an all-encompassing description of "what is mechatronics" eludes us. This lack of consensus is a healthy sign. It says that the field is alive, that it is a youthful subject. Even without an unarguably definitive description of mechatronics, engineers understand from the definitions given above and from their own personal experiences the essence of the philosophy of mechatronics.

For many practicing engineers on the front line of engineering design, mechatronics is nothing new. Many engineering products of the last 25 years integrated mechanical, electrical, and computer systems yet were designed by engineers that were never formally trained in mechatronics per se, it appears tat modern concurrent engineering design practices, now formally viewed as part of the mechatronics specialty, are natural design processes. What is evident is that the study of mechatronics provides a mechanism for scholars interested in understanding and explaining the engineering design process to define, classify, organize, and integrate many aspects of product design into a coherent package, as the historical divisions between mechanical, electrical, aerospace, chemical, civil, and computer engineering become less clearly defined, we should take comfort in the existence of mechatronics as a field of study in academia. The mechatronics specialty provides an educational path, that is, a roadmap, for engineering students studying within the traditional structure of most engineering colleges. Mechatronics is generally recognized worldwide as a vibrant area of study. Undergraduate and graduate programs in mechatronic engineering art now offered in many universities. Refereed journals are being published and dedicated conferences are being organized and are generally highly attended.

It should be understood that mechatronics is not jest a convenient structure for investigative studies by academicians; it is a way of life in modern engineering practice. The introduction of the microprocessor in the early 1980s and the ever increasing desired performance to cost ratio revolutionized the paradigm of engineering design. The number of new products being developed at the intersection of traditional disciplines of

engineering, computer science, and the natural sciences is ever increasing. New developments in these traditional disciplines are being absorbed into mechatronics design at an ever increasing pace. The ongoing information technology revolution, advances in wireless communication, smart sensors design (enabled by MEMS technology), and embedded systems engineering ensures that the engineering design paradigm will continue to evolve in the early twenty-first century.

The figure shown below illustrates a mechatronic system:

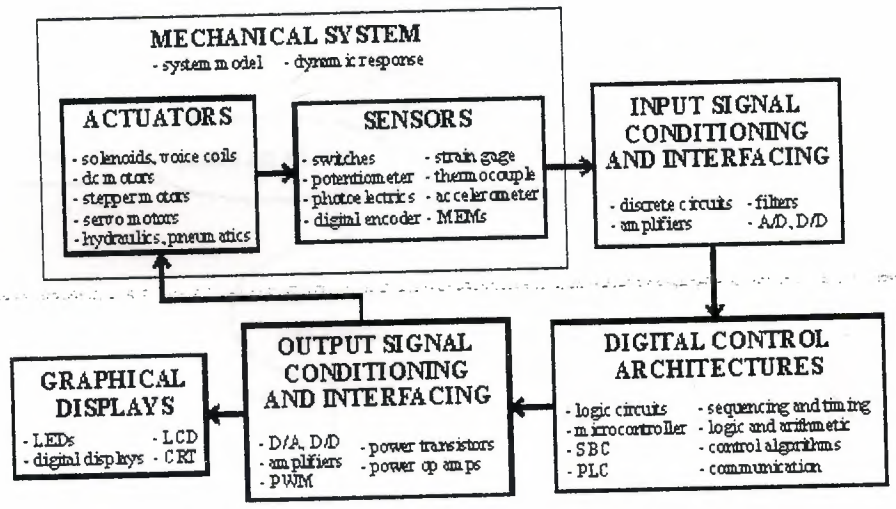


Figure 1.1 a mechatronic system

1.1.2 Key elements of Mechatronics:

The study of mechatronics can be divided into the following areas specialty:

1. Physical systems modeling
2. Sensors and Actuators
3. Signals and Systems
4. Computers and Logic Systems
5. Software and Data Acquisition

The key elements of mechatronics are illustrated in Figure 1.2 As the field of mechatronics continues to mature; the list of relevant topics associated with the area will most certainly expand and evolve.

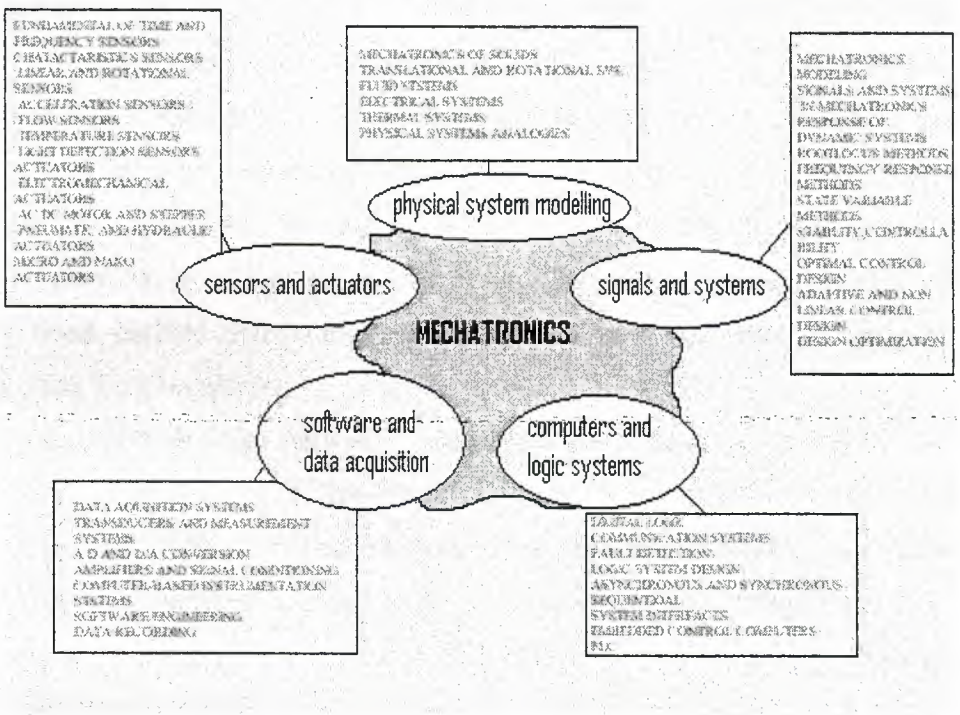


Figure 1.2 the key element of mechatronics

1.1.3 Historical Perspective

Attempts to construct automated mechanical systems has an interesting history. Actually, the term "automation" was not popularized until the 1940s when it was coined by the Ford Motor Company to denote a process in which a machine transferred a sub-assembly item from one station to another and then positioned the item precisely for additional assembly operations. But successful development of automated mechanical systems occurred long before then. For example early applications of automatic control systems appeared in Greece from 300 to 1 B.C. with the development of float regulator mechanisms. Tow important examples include the water clock of Ktesibios that used a float regulator, and an oil lamp devised by Philon, which also used a float regulator to maintain a constant level of fuel oil. Later, in the first century, Heron of Alexandria published a book entitled Pneumatica that described different types of water-level mechanisms using float regulators.

In Europe and Russia, between seventeenth and nineteenth centuries, many important devices were invented that would eventually contribute to mechatronics. Cornelis Drebble (1572-1633) of Holland devised the temperature regulator representing one of the feedback systems of that area. Subsequently, Dennis Papin (1647-1712) invented a pressure safety regulator for steam boilers in 1681. Papin's pressure regulator is similar to a modern-day pressure-cooker valve. The first mechanical calculating machine was invented by Pascal in 1642. The first historical feedback system claimed by Russia was developed by polzunov in 1765. Polzunov's water-level float regulator employs a float that rises and lowers in relation to the water level, thereby controlling the valve that water inlet in the boiler.

Further evolution in automation was enabled by advancements in control theory traced back to the Watt flyball governor was used to control the speed of a steam engine. Employing a measurement of the speed of the output shaft and utilizing the motion of the flyball to control the valve, the amount of steam entering the engine is controlled. As the speed of the engine increases, the metal spheres on the governor apparatus rise and extend away from the shaft axis, thereby closing the valve. This is an example of a feed back

control system where the feedback signal and the control actuation are completely coupled in the mechanical hardware.

These early successful automation developments were achieved through intuition, application of practical skills, and persistence. The next step in the evolution of automation required a theory of automatic control. The precursor to the numerically controlled (NC) machines for automated manufacturing (to be developed in the 1950s and 60s at MIT) appeared in the early 1800s with the invention of feed-forward control of weaving looms by Jacquard of France. In the late 1800s, the subject now known as control theory was initiated by J.C. Maxwell through analysis of the set of differential equations describing the flyball governor. Maxwell investigated the effect. Various system parameters had on the system performance. At about the same time, Vyshnegradskii formulated a mathematical theory of regulators. In the 1830s, Michael Faraday described the law of induction that would form the basic of the electric motor and the electric dynamo. Subsequently, in the late 1880, Nikola Tesla invented the alternating-current induction motor. The basic idea of controlling a mechanical system automatically was firmly established by the end of 1800s. The evolution of automation would accelerate significantly in the twentieth century.

The development of pneumatic control elements in the 1930s matured to a point of finding applications in the process industries. However, prior to 1940, the design of control systems remained an art generally characterized by trial-and-error methods. During the 1940s, continued advances in mathematical and analytical methods solidified the notion of control engineering as an independent engineering discipline. In the United States, the development of the telephone system and electronic and electronic feedback amplifiers spurred the use of feedback by Bode, Nyquist, and Black at Bell Telephone Laboratories. The operation of the feedback amplifiers was described in the frequency domain and the ensuing design and analysis practices are now generally classified as "classical control." During the same time period, control theory was also developing in Russia and Eastern Europe. Mathematicians and applied mechanicians in the former Soviet Union dominated the field of control and concentrated on time domain formulations and differential equation models of systems. Further developments of time domain formulations using state variable

system representation occurred in the 1960s and led to design and analysis practices now generally classified as "modern control."

The World War II war effort led to further advances in the theory and practice of automatic control in an effort to design and construct automatic airplane pilots, gun-positioning systems, radar antenna control systems, and other military systems. The complexity and expected performance of these military systems necessitated an extension of the available control techniques and fostered interest in control systems and the development of new insights and methods. Frequency domain techniques continued to dominate the field of controls following World War II, with the increased use of the Laplace transform, and the use of the so-called s-plane methods, such as designing control systems using root locus.

On the commercial side, driven by cost savings achieved through mass production, automation of the production process was a high priority beginning in 1940s. During the 1950s, the invention of the cam linkages, and chain drives became the major enabling technologies for the invention of new products and high-speed precision manufacturing and assembly. Examples include textile and printing machines, paper converting machinery, and sewing machines. High-volume precision manufacturing became a reality during this period. The automated paperboard container-manufacturing machine employs a sheet-fed process wherein the paperboard is cut into a fan shape to form the tapered sidewall, and wrapped around a mandrel. The seam is then heat sealed and held until cured. Another sheet-fed source of paperboard is used to cut out the plate to form the bottom of the paperboard container, formed into a shallow dish through scoring and creasing operations I a die, and assembled to the cup shell. The lower edge of the cup shell is bent inwards over the edge of the bottom plate sidewall, and heat-sealed under high pressure to provide a ring-on-shell structure to provide the stiffness needed for its functionality. All of these operations are carried out while the work piece undergoes a precision transfer from one turret to another and is then ejected. The production rate of a typical machine averages over 200 cups per minute. The automated paperboard container manufacturing did not involve any non0mechanical system except an electric motor for driving the line shaft. These machines are typical of paper converting and textile machinery and represent automated systems significantly more complex than their predecessors.

The development of the microprocessor in the late 1960s led to early forms of computer control in process and product design. Examples include numerically controlled (NC) machines and air craft control systems. Yet the manufacturing processes were still entirely mechanical in nature and the automation and control systems were implemented only as an afterthought. The launch of Sputnik and the advent of the space age provided yet another impetus to the continued development of controlled mechanical systems. Missiles and space probes necessitated the development of complex, highly accurate control systems. Furthermore, the need to minimize satellite mass (that is, to minimize the amount of fuel required for the mission) while providing accurate control encouraged advancements in the important field of optimal control. Time domain methods developed by Liapunov, Minorsky, and others, as well as the theories of optimal control developed by L.S. Pontryagin in the former Soviet Union and R.Bellman in the United States, were well matched with the increasing availability of high0speed computers and new programming languages for scientific use.

Advancements in semiconductor and integrated circuits manufacturing led to the development of a new class of products that incorporated mechanical and electronics in the system and required the two together for their functionality. The term mechatronics was introduced by Yasakawa Electric in 1969 to represent such systems. Yasakawa was granted a trademark in 1972, but after widespread usage of the term, released its trademark rights in 1982. Initially, mechatronics referred to systems with only mechanical systems and electrical components—no computation was involved. Examples of such systems include the automatic sliding door, vending machines, and garage door openers.

In the late 1970s, the Japan Society for the Promotion of Machine Industry (JSPMI) classified mechatronics products into four categories:

1. Class I: Primarily mechanical products with electronics incorporated to enhance functionality. Examples include numerically controlled machine tools and variable speed drives in manufacturing machines.
2. Class II: Traditional mechanical systems with significantly updated internal devices incorporating electronics. The external user interfaces are

unaltered. Examples include the modern sewing machine and automated manufacturing systems.

3. Class III: Systems that retain the functionality of the traditional mechanical system, but the internal mechanisms are replaced by electronics. An example is the digital watch.
4. Class IV: Products designed with mechanical and electronic technologies through synergistic integration. Examples include photocopiers, intelligent washers and dryers, rice cookers, and automatic ovens.

The enabling technologies for each mechatronic product class illustrate the progression of electromechanical products in stride with developments in control theory, computation technologies, and microprocessors. Class I products were enabled by servo technology, power electronics, and control theory. Class II products were enabled by the availability of early computational and memory devices and custom circuit design capabilities. Class III products relied heavily on the microprocessor and integrated circuits to replace mechanical systems. Finally, Class IV products marked the beginning of true mechatronic system, through integration of mechanical systems and electronics. It was not until the 1970s with the development of the microprocessor by the Intel Corporation that integration of computational systems with mechanical systems became practical.

The divide between classical control and modern control was significantly reduced in the 1980s with the advent of "robust control" theory. It is now generally accepted that control engineering must consider both the time domain and the frequency domain approaches simultaneously in the analysis and design of control systems. Also, during the 1980s, the utilization of digital computers as integral components of control system became routine. There are literally hundreds of thousands of digital process control computers installed worldwide. Whatever definition of mechatronics one chooses to adopt, it is evident that modern mechatronics involves computation as the central element. In fact, the incorporation of the microprocessor to precisely modulate mechanical power and to adapt to changes in environment are the essence of modern mechatronics and smart products.

In the future, growth in mechatronic systems will be fueled by the growth in the constituent areas. Advancement in traditional disciplines fuel the growth of mechatronics systems by providing "enabling technologies." For example, the invention of the microprocessor had a profound effect on the redesign of mechanical systems and design of new mechatronics systems. We should expect continued advancements in cost-effective microprocessors and microcontrollers, sensor and actuator development enabled by advancements in applications of MEMS, adaptive control methodologies and real-time programming methods, networking and wireless technologies, mature CAE technologies for advanced system modeling, virtual prototyping, and testing. The continued rapid development in these areas will only accelerate the pace of smart product development. The Internet is a technology that, when utilized in combination with wireless technology, may also lead to new mechatronic products. While developments in automotives provide vivid examples of mechatronics development, there are numerous examples of intelligent systems in all walks of life, including smart home appliances such as dishwashers, vacuum cleaners, microwaves, and wireless network enabled devices. In the area of "human-friendly machines" (a term used by H.Kobayashi), we can expect advances in robot-assisted surgery, and implantable sensors and actuators. Other areas that will benefit from mechatronic advances may include robotics, manufacturing, space technology, and transportation. The future of mechatronics is wide open.

1.2 Sensors

One type of feedback frequently needed by industrial-control systems is the position of one or more components of the operation being controlled. Sensors are devices used to provide information on the presence or absence of an object.

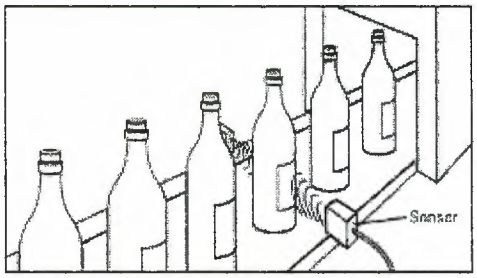


Figure 1.3 Application for sensors

1.2.1 Siemens Sensors

Siemens sensors include limit switches, photoelectric, inductive, capacitive, and ultrasonic sensors. These products are packaged in various configurations to meet virtually any requirement found in commercial and industrial applications.
Each type of sensor will be discussed in detail. At the end of the course an application guide is provided to help determine the right sensor for a given application.

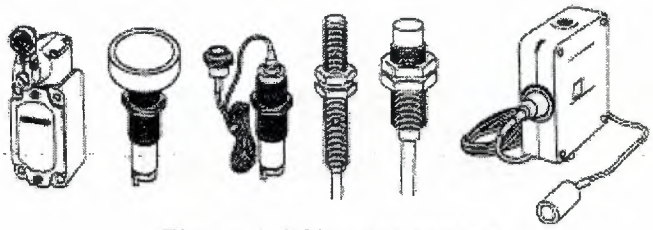


Figure 1.4 Siemens sensors

1.2.2 Technologies

Limit switches use a mechanical actuator input, requiring the sensor to change its output when an object is physically touching the switch. Sensors, such as photoelectric, inductive, capacitive, and ultrasonic, change their output when an object is present, but not touching the sensor. In addition to the advantages and disadvantages of each of these sensor types, different sensor technologies are better suited for certain applications. The following table lists the sensor technologies that will be discussed in this course.

1.2.3 Contact Arrangement

Contacts are available in several configurations. They may be normally open (NO), normally closed (NC), or a combination of normally open and normally closed contacts. Circuit symbols are used to indicate an open or closed path of current flow. Contacts are shown as normally open (NO) or normally closed (NC). The standard method of showing a contact is by indicating the circuit condition it produces when the contact-actuating device is in the reenergized or no operated state. For the purpose of explanation in this text a contact or device shown in a state opposite of its normal state will be highlighted. Highlighted symbols used to indicate the opposite state of a contact or device is not legitimate symbols. They are used here for illustrative purposes only.

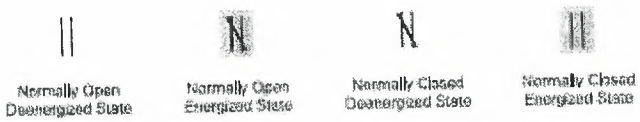


Figure 1.5 contacts configurations

Mechanical limit switches, which will be covered in the next section, use a different set of symbols. Highlighted symbols are used for illustrative purposes only.

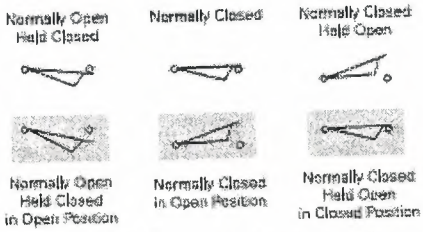


Figure 1.6 Positions of contacts

1.2.4 Circuit Example

In figure 1.7, a mechanical limit switch (LS1) has been placed in series with a Run/Stop contact and the "M" contactor coil. The Run/Stop contact is in the Run condition and the motor is running a process. This could be a conveyor or some other device. Note that the "M" contacts and the "Run/ Stop" are shown highlighted, indicating they are normally open contacts in the closed position. LS1 is a normally closed contact of the mechanical limit switch.

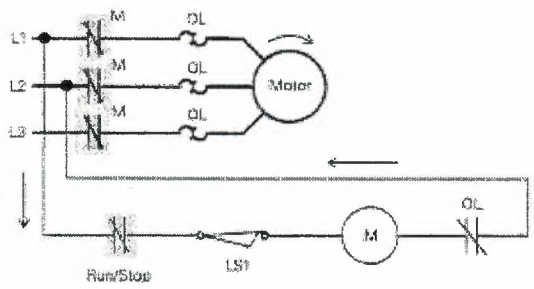


Figure 1.7 LS1 in short circuit position

When an object makes contact with the mechanical limit switch the LS1 contacts will change state. In this example the normally closed contacts of LS1 open. The mechanical limit switch symbol is highlighted. The "M" contactor coil is reenergized, returning the normally open contacts of the "M" contactor to their normal position, stopping the motor and the process.

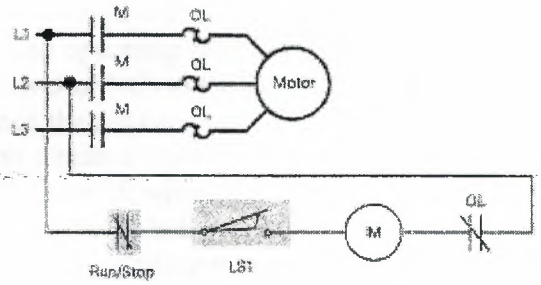


Figure 1.8 LS1 in open circuit position

1.2.5 Limit Switches

A typical limit switch consists of a switch body and an operating head. The switch body includes electrical contacts to energize and reenergize a circuit. The operating head incorporates some type of lever arm or plunger, referred to as an actuator. The standard limit switch is a mechanical device that uses physical contact to detect the presence of an object (target). When the target comes in contact with the actuator, the actuator is rotated from its normal position to the operating position. This mechanical operation activates contacts within the switch body.

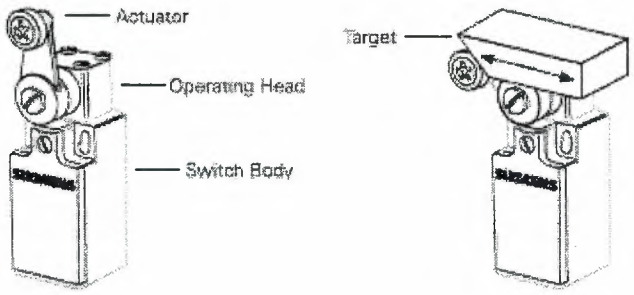


Figure 1.9 Limit switch

1.2.6 Principle of Operation

A number of terms must be understood to understand how a mechanical limit switch operates. The free position is the position of the actuator when no external force is applied. Prêt ravel is the distance or angle traveled in moving the actuator from the free position to the operating position. The operating position is where contacts in the limit switch change from their normal state (NO or NC) to their operated state.
Over travel is the distance the actuator can travel safely beyond the operating point.
Differential travel is the distance traveled between the operating position and the release position. The release position is where the contacts change from their operated state to their normal state. Release travel is the distance traveled from the release position to the free position. Figure 1.10 is illustrating the operation principle.

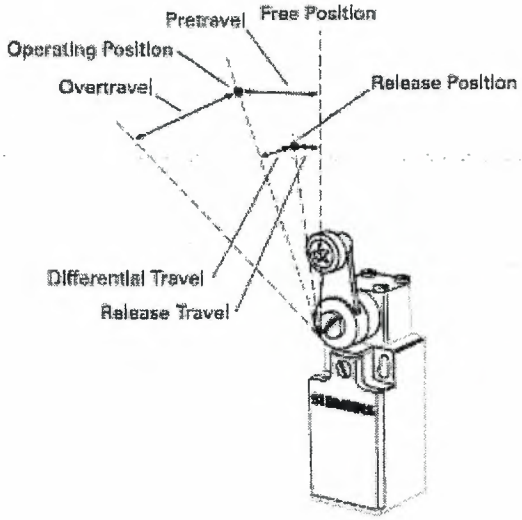


Figure 1.10 Operation principle of limit switch

2.1 Hard-Wired Control

Prior to PLCs, many of these control tasks were solved with contactor or relay controls. This is often referred to as hardwired control. Circuit diagrams had to be designed, electrical components specified and installed, and wiring lists created. Electricians would then wire the components necessary to perform a specific task. If an error was made, the wires had to be reconnected correctly. A change in function or system expansion required extensive component changes and rewiring.

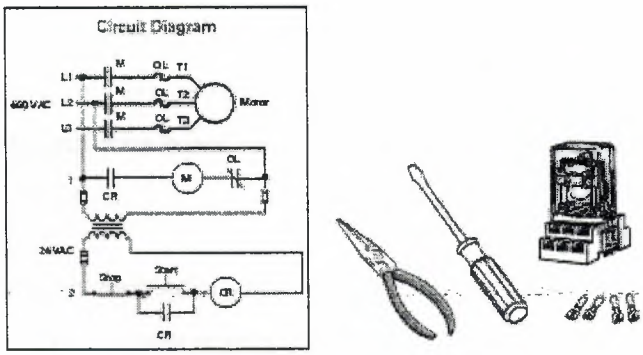


Figure 2.1 Hard wired controls

The programmable controller was detailed on New Year's Day, 1968, and from hence till now, a slow steady growth has allowed the manufacturing and process control industries to take advantage of applications-oriented software.

The early days however, were not as straightforward or as simple.

For example, the programmable controller, according to the system architecture specification, did not need to go fast. The initial machine, which was never delivered, only had 125 words of memory, and speed was not a criteria as mentioned earlier. You can imagine what happened!

During the last ten to fifteen years a wide range of different programming techniques has been used to write programs for industrial control applications and for Programmable Logic Controllers (PLCs). Control applications have been developed in BASIC, FORTH, C, Structured English, Instruction List and numerous other proprietary languages including various dialects of Ladder programming. Unfortunately, the only thing that can be said of all of these programming languages is that they are all different. For people involved with such systems from technicians, maintenance personnel, system designers to plant managers, this results in inefficient use of time and money. There is clearly a waste of human resources involved in training staff in skills in so many different control languages. Programmable Logic Controllers (PLCs) also referred to as programmable controllers are in the computer family. They are used in commercial and industrial applications. A PLC monitors inputs, makes decisions based on its program, and controls outputs to automate a process or machine.

Fortunately the international industrial community recognized that a new standard for programmable logic controllers was required. A working group within the International Electro-technical Commission (IEC) was set up to look at the complete design of programmable logic controllers, including the hardware design, installation, testing, documentation, programming and communications. The IEC as a sister organization to the International Standardization Organization (ISO) based in Geneva, has committees and working groups formed from representatives put forward by standardization bodies of most industrial countries of the world.

During the early 1990s, the IEC published various parts of the IEC61131 standard that covers the complete life cycle of the Programmable Logic Controllers (PLCs), these are:

1. General information Definition of basic terminology and concepts.
2. Equipment requirements and tests Electronic and mechanical construction and verification tests.
3. Programmable languages PLC software structure, languages and program execution.
4. User guidelines Guidance on selection, installation, maintenance of PLCs. 5.Messaging service specification Software facilities to communicate with other devices using

communications based on MAP Manufacturing Messaging Services.

6. Communications via field bus Software facilities of PLC communications using IEC field bus -Awaiting completion of field bus standards.

7. Fuzzy control programming Software facilities, including standard function blocks for handling fuzzy logic within PLCs.

8. Programmable controllers Application guidelines for the implementation of IEC61131-3 languages.

The IEC recognize that industrial instrumentation and control systems need an open systems approach to build large systems using equipment from different manufacturers. Until the IEC 61131-3 standard was published in March 1993, there was no suitable standard that defined the way control systems such as PLCs could be programmed. Ladder Programming has become as one of the most popular graphical languages for programming PLCs but unfortunately had a number of problems.

2.2 Basic PLC Operation

PLCs consist of input modules or points, a Central Processing Unit (CPU), and output modules or points. An input accepts a variety of digital or analog signals from various field devices (sensors) and converts them into a logic signal that can be used by the CPU. The CPU makes decisions and executes control instructions based on program instructions in memory. Output modules convert control instructions from the CPU into a digital or analog signal that can be used to control various field devices (actuators). A programming device is used to input the desired instructions. These instructions determine what the PLC will do or a specific input. An operator interface device allows process information to be displayed and new control parameters to be entered.

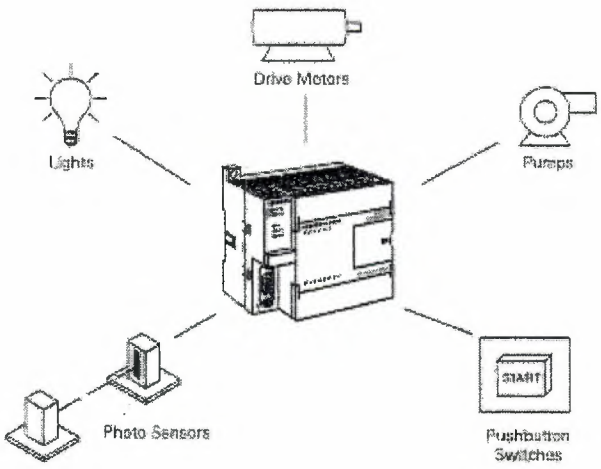


Figure 2.2 Basic PLC operation

2.3 Advantages of PLCs

The same, as well as more complex tasks can be done with a PLC. Wiring between devices and relay contacts is done in the PLC program. Hard wiring, though still required to connect field devices, is less intensive. Modifying the application and correcting errors are easier to handle. It is easier to create and change a program in a PLC than it is to wire and rewire a circuit Following are just a few of the advantages of PLCs:

- Smaller physical size.
- Faster and easier.
- PLCs have integrated diagnostics and override functions.
- Diagnostics are centrally available.
- Applications can be immediately documented.
- Applications can be duplicated faster and less expensively.

2.4 Terminology

The language of PLCs consists of a commonly used set of terms; many of which are unique to PLCs. In order to understand the ideas and concepts of PLCs, an understanding of these terms is necessary.

2.5 Sensors

A sensor is a device that converts a physical condition into an electrical signal for use by the PLC. Sensors are connected to the input of a PLC. A pushbutton is one example of a sensor that is connected to the PLC input. An electrical signal is sent from the pushbutton to the PLC indicating the condition (open/ closed) of the pushbutton contacts.

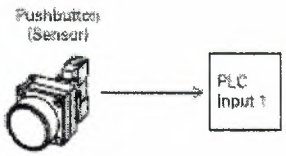


Figure 2.3 Sensor with a pushbutton

2.6 Actuators

Actuators convert an electrical signal from the PLC into a physical condition. Actuators are connected to the PLC output. A motor starter is one example of an actuator that is connected to the PLC output. Depending on the output PLC signal the motor starter will either start or stop the motor.

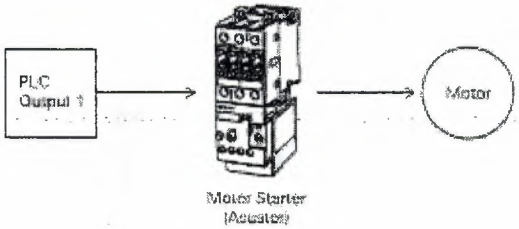


Figure 2.4 Actuator position in a system

2.7 Discrete Inputs

A discrete input also referred to as a digital input, is an input that is either in an ON or OFF condition. Pushbuttons, toggle switches, limit switches, proximity switches, and contact closures are examples of discrete sensors, which are connected to the PLCs discrete or digital inputs. In the ON condition a discrete input may be referred to as logic 1 or logic high. In the OFF condition a discrete input may be referred to as logic 0 or logic low.

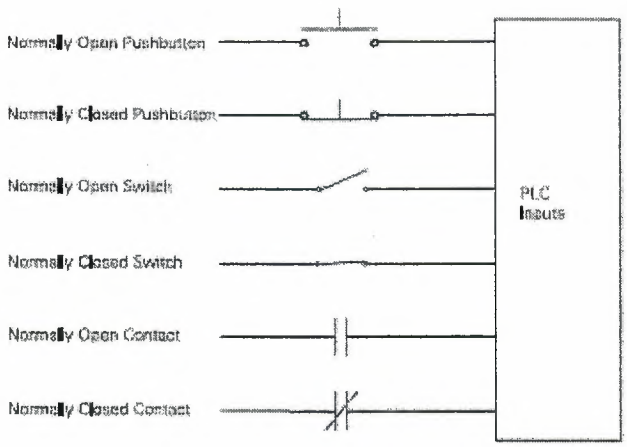


Figure 2.5 PLC inputs

2.8 Analog Inputs

An analog input is an input signal that has a continuous signal. Typical analog inputs may vary from 0 to 20 milliamps, 4 to 20 milliamps, or 0 to 10 volts e.g. level transmitter monitors, see figure 2.6.

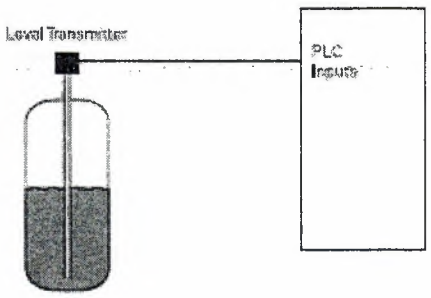


Figure 2.6 Analog input

2.9 Discrete Outputs

A discrete output is an output that is either in an ON or OFF condition. Solenoids, contactor coils, and lamps are examples of actuator devices connected to discrete outputs.

Discrete outputs may also be referred to as digital outputs e.g. a lamp can be turned on or off by the PLC output it is connected to.

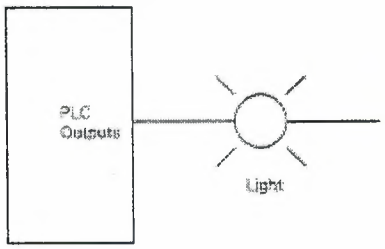


Figure 2.7 A discrete output

2.10 Analog Outputs

An analog output is an output signal that has a continuous signal. The output may be as simple as a 0-10 VDC level that drives an analog meter. Examples of analog meter outputs are speed, weight, and temperature. The output signal may also be used on more complex applications such as a current to pneumatic transducer that controls an air-operated flow-control valve.

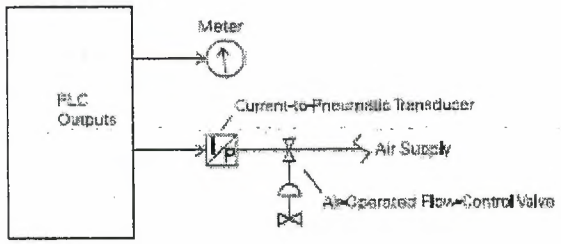


Figure 2.8 Analog outputs

2.11 CPU

The central processor unit (CPU) is a microprocessor system that contains the system memory and is the PLC decision-making nit. The CPU monitors the inputs and makes decisions asked on instructions held in the program memory. The PU performs relay, counting, timing, data comparison, and sequential operations.

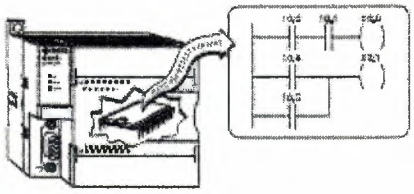


Figure 2.9 CPU in PLC

2.12 Programming

A program consists of one or more instructions that accomplish task. Programming a PLC is simply constructing a set of instructions. There are several ways to look at a program such s ladder logic, statement lists, or function block diagrams.

2.13 Ladder Logic

Ladder logic (LAD) is one programming language used in PLCs. Ladder logic uses components that resemble elements used in a line diagram format to describe hard-wired control.

PLC had to be maintainable by technicians and electrical personnel. To support the programming language of Ladder Logic was developed. Ladder Logic is based on the relay and contact symbols technicians were used to through wiring diagrams of electrical control panels.

Until recently there has been no formal programming standard for PLC's. The introduction of the IEC 61131 Standard in 1998 provides a more formal approach to coding. PLC Manufacturers have so far been slow on the uptake of the standard with partial implementation.

The documentation for early PLC Programs was either non existent or very poor, just providing simple addressing or basic comments, making large programs difficult to follow. This has been greatly improved with the development of PLC Programming Packages.

2.14 Ladder Logic Diagram

The left vertical line of a ladder logic diagram represents the power or energized conductor. The output element or instruction represents the neutral or return path of the circuit. The right vertical line, which represents the return path on a hard-wired control line diagram, is omitted. Ladder logic diagrams are read from left-to-right, top-to-bottom. Rungs are sometimes referred to as networks. A network may have several control elements, but only one output coil.

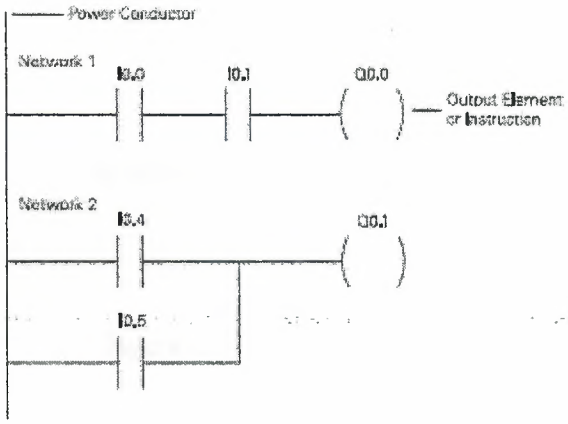


Figure 2.10 Ladder Logic Diagram

In the example program shown example I0.0, I0.1 and Q0.0 represent the first instruction combination. If inputs I0.0 and I0.1 are energized, output relay Q0.0 energizes. The inputs could be switches, pushbuttons, or contact closures. I0.4, I0.5, and
Q1.1 represent the second instruction combination. If either input I0.4 or I0.5 are energized, output relay Q0.1 energizes.

2.15 Statements list

A statement list (STL) provides another view of a set of instructions. The operation, what is to be done, is shown on the left. The operand, the item to be operated on by the operation, is shown on the right. A comparison between the statements list shown below, and the ladder logic shown on the previous page, reveals a similar structure. The set of instructions in this statement list perform the same task as the ladder diagram.

```
NETWORK 1
        LD      I0.0
        A       I0.1
        =       Q0.0
NETWORK 2
        LD      I0.4
        O       I0.5
        =       Q0.1
```

Figure 2.11 Statements list

2.16 Function Block Diagrams

Function Block Diagrams (FBD) provides another view of a set of instructions. Each function has a name to designate its specific task. Functions are indicated by a rectangle. Inputs are shown on the left-hand side of the rectangle and outputs are shown on the right-hand side. The function block diagram shown below performs the same function as shown by the ladder diagram and statement list.

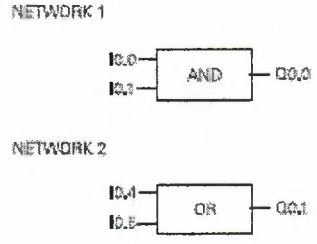


Figure 2.12 Function Block Diagrams

2.17 PLC Scan

The PLC program is executed as part of a repetitive process referred to as a scan. A PLC scan starts with the CPU reading the status of inputs. The application program is executed using the status of the inputs. Once the program is completed, the CPU performs internal diagnostics and communication tasks. The scan cycle ends by updating the outputs, and then starts over. The cycle time depends on the size of the program, the number of I/Os, and the amount of communication required.

Figure 2.133 PLC Scan

2.18 Software

Software is any information in a form that a computer or PLC can use. Software includes the instructions or programs that direct hardware, see figure 2.14.

Figure 2.14 Example for software

2.19 Hardware

Hardware is the actual equipment. The PLC, the programming device, and the connecting cable are examples of hardware, see figure 2.15.

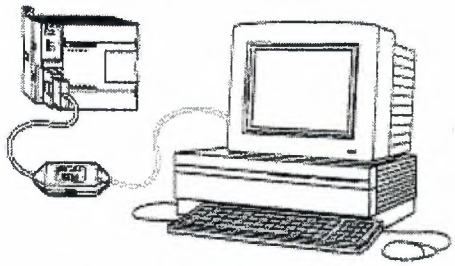


Figure 2.15 Hardware

2.20 Memory Size

Kilo, abbreviated K, normally refers to 1000 units. When talking about computer or PLC memory, however, 1K means 1024. This is because of the binary number system (2_{10}=1024). This can be 1024 bits, 1024 bytes, or 1024 words, depending on memory type.

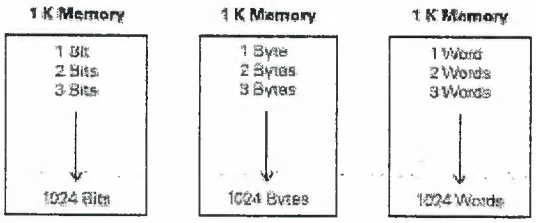


Figure 2.16 Kilo=1024 units

2.21 RAM

Random Access Memory (RAM) is memory where data can be directly accessed at any address. Data can be written to and read from RAM. RAM is used as a temporary storage area. RAM is volatile, meaning that the data stored in RAM will be lost if power is lost. A battery backup is required to avoid losing data in the event of a power loss.

2.22 ROM

Read Only Memory (ROM) is a type of memory that data can be read from but not written to. This type of memory is used to protect data or programs from accidental erasure. ROM memory is nonvolatile. This means a user program will not lose data during a loss of electrical power. ROM is normally used to store the programs that define the capabilities of the PLC.

2.23 EPROM

Erasable Programmable Read Only Memory (EPROM) provides some level of security against unauthorized or unwanted changes in a program. EPROMs are designed so that data stored in them can be read, but not easily altered. Changing EPROM data requires a special effort. UVEPROMs (ultraviolet erasable programmable read only memory) can only be erased with an ultraviolet light. EEPROM (electronically erasable programmable read only memory), can only be erased electronically.

2.24 Firmware

Firmware is user or application specific software burned into EPROM and delivered as part of the hardware. Firmware gives the PLC its basic functionality.

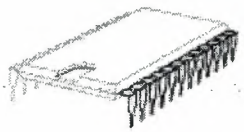


Figure 2.17 Firmware

2.25 Putting it together

The memory of siemens S7-200 is divided into three areas:

2.25.1 Program space

Program space stores the ladder logic (LAD) or statement list (STL) program instructions. This area of memory controls the way data space and I/O points are used. LAD or STL instructions are written using a programming device such as a PC, then loaded into program memory of the PLC.

2.25.2 Data space

Data space is used as a working area, and includes memory locations for calculations, temporary storage of intermediate results and constants. Data space includes memory

Locations for devices such as timers, counters, high-speed counters, and analog inputs and outputs. Data space can be accessed under program control.

2.25.3 Configurable parameter space

Configurable parameter space, or memory, stores either the default or modified configuration parameters.

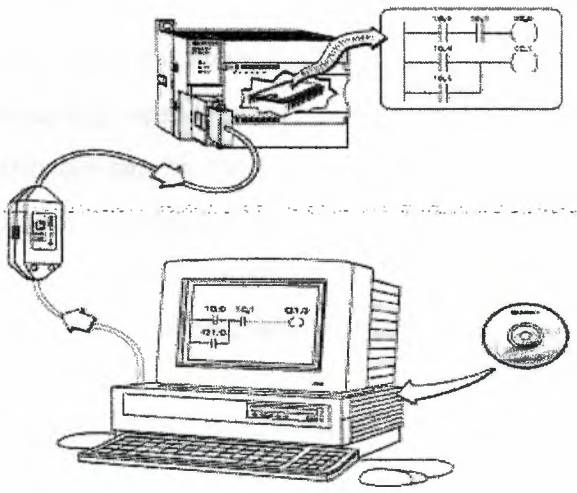


Figure 2.18 Configuration of parameter space

2.26 Programming A PLC

A number of instructions that must be arranged in a logical order to obtain the desired PLC operation. These instructions are divided into three groups: standard instructions, special instructions, and high-speed instructions.

2.26.1 Standard Instructions

Standard instructions consist of instructions that are found in most programs. Standard instructions include; timer, counter, math, logical, increment/decrement/invert, move, and block instructions.

2.26.2 Special Instructions

Special instructions are used to manipulate data. Special instructions include shift, table, find, conversion, for/next, and real-time instructions.

2.26.3 High-Speed Instructions

High-speed instructions allow for events and interrupts to occur independent of the PLC scan time. These include high-speed counters, interrupts, output, and transmit instructions. The programming software can be run Off-line or On-line. Offline programming allows the user to edit the ladder diagram and perform a number of maintenance tasks. The PLC does not need to be connected to the programming device in this mode. On-line programming requires the PLC to be connected to the programming device. In this mode program changes are downloaded to the PLC. In addition, status of the input/output elements can be monitored. The CPU can be started, stopped, or reset.

2.26.4 Symbols

In order to understand the instructions a PLC is to carry out, an understanding of the language is necessary. The language of PLC ladder logic consists of commonly used set symbols that represent control components and instructions.

2.26.5 Contacts

One of the most confusing aspects of PLC programming for first-time users is the relationship between the device that controls a status bit and the programming function that uses a status bit. Two of the most common programming functions are the normally open (NO) contact and the normally closed (NC) contact. Symbolically, power flows through these contacts when they are closed. The normally open contact (NO) is true (closed) when

the input or output status bit controlling the contact is 1. The normally closed contact (NC) is true (closed) when the input or output status bit controlling the contact is 0.

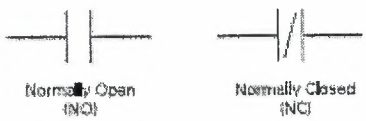


Figure 2.19 Contacts

2.26.6 Coils

Coils represent relays that are energized when power flows to them. When a coil is energized, it causes a corresponding output to turn on by changing the state of the status bit controlling that output to 1. That same output status bit may be used to control normally open and normally closed contacts elsewhere in the program, see figure 2.20.

Figure 2.20

2.26.7 Boxes

Boxes represent various instructions or functions that are executed when power flows to the box. Typical box functions are timers, counters, and math operations.

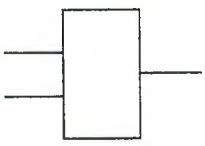


Figure 2.21 A box

2.26.8 Entering Elements

Control elements are entered in the ladder diagram by positioning the cursor and selecting the element from lists. In the following example the cursor has been placed in the position

to the right of I0.2. A coil was selected from a pull down list and inserted in this position, see figure 2.22.

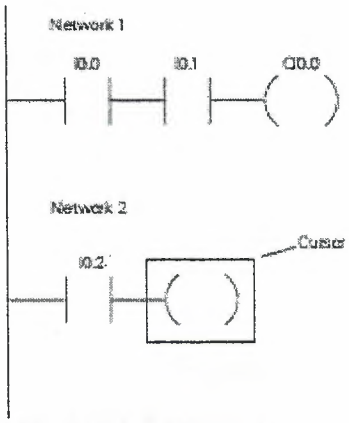


Figure 2.22

2.26.9 An AND Operation

Each rung or network on a ladder represents a logic operation. The following programming example demonstrates an AND operation. Two contact closures and one output coil are placed on network 1. They were assigned addresses I0.0, I0.1, and
Q0.0. Note that in the statement list a new logic operation always begins with a load instruction (LD). In this example I0.0 (input 1) and (A in the statement list) I0.1 (input 2) must be true in order for output Q0.0 (output 1) to be true. It can also be seen That I0.0 and I0.1 must be true for Q0.0 to be true by looking at the function block diagram representation.

Ladder Diagram Representation

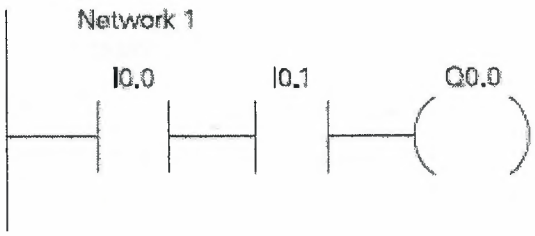


Figure 2.23 AND Operation

Statement List Representation

```
Network 1
LD    I0.0
A     I0.1
=     Q0.0
```

Function Block Diagram Representation, see the next figure:

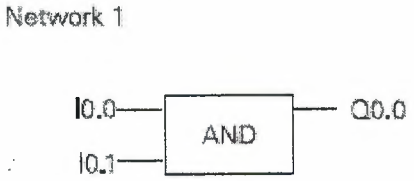
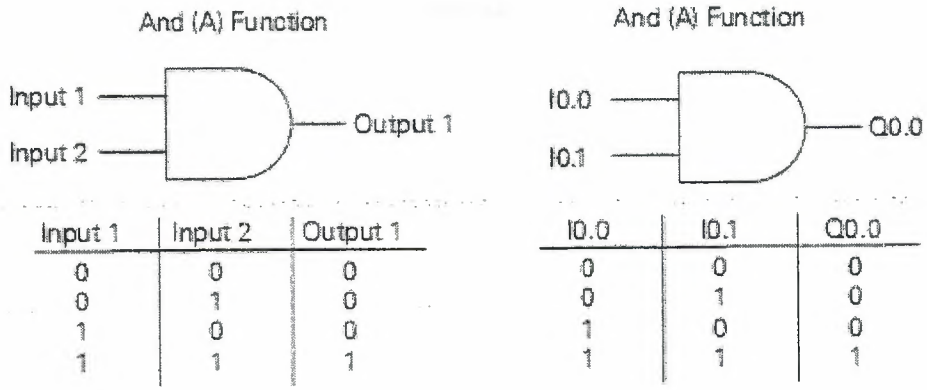


Figure 2.24

Another way to see how an AND function works is with a Boolean logic diagram. In Boolean logic an AND gate is represented by a number of inputs on the left side. In this case there are two inputs. The output is represented on the right side. It can be seen from the table that both inputs must be logic 1 in order for the output to be logic 1.



Input 1	Input 2	Output 1
0	0	0
0	1	0
1	0	0
1	1	1

I0.0	I0.1	Q0.0
0	0	0
0	1	0
1	0	0
1	1	1

Figure 2.25 AND (A) Function

2.26.10 An OR Operation

In this example an OR operation is used in network 1. It can be seen that if either input I0.2 (input 3) or (O in the statement list) input I0.3 (input 4), or both are true, then output Q0.1 (output 2) will be true.

Ladder Diagram Representation

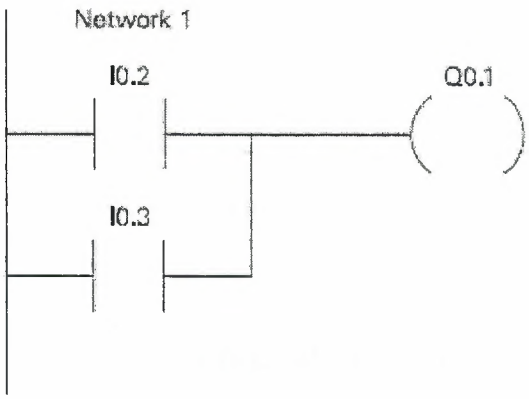


Figure 2.26 OR Operation

Statement List Representation

```
Network 1
LD    I0.2
O     I0.3
=     Q0.1
```

Function Block Diagram Representation, see the figure below:

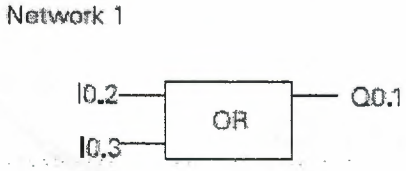
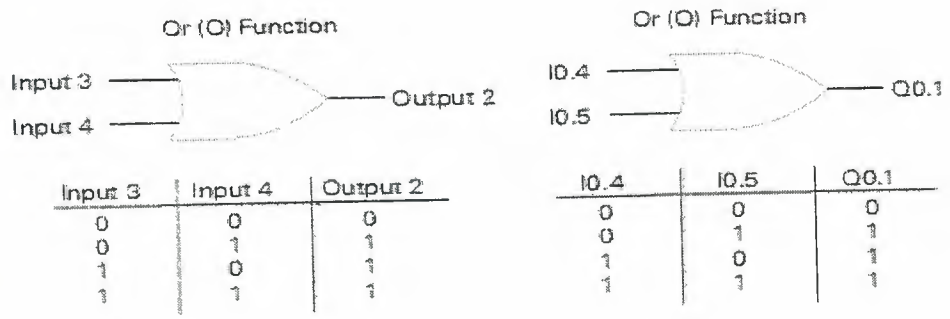


Figure 2.27

Another way to see how an OR function works is with a Boolean logic diagram. The symbol differs slightly from an AND function. The OR function is represented by a number of inputs on the left side. In this case there are two inputs. The output is represented on the right side. It can be seen from the table that any input can be logic 1 in order for the output to be logic 1.



Input 3	Input 4	Output 2
0	0	0
0	1	1
1	0	1
1	1	1

I0.4	I0.5	Q0.1
0	0	0
0	1	1
1	0	1
1	1	1

Figure 2.28 OR (O) Function

2.27 Testing a Program

Once a program has been written it needs to be tested and debugged. One way this can be done is to simulate the field inputs with an input simulator, such as the one made for the S7-200. The program is first downloaded from the programming device to the CPU. The selector switch is placed in the RUN position. The simulator switches are operated and the resulting indication is observed on the output status indicator lamps.

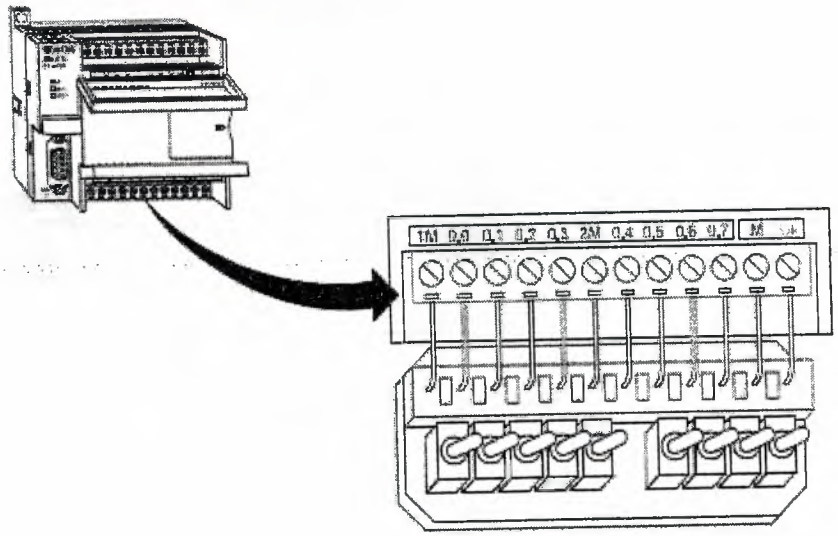


Figure 2.29 Testing Program Machine

2.28 Status Functions

After a program has been loaded and is running in the PLC, the actual status of ladder elements can be monitored. The standard method of showing a ladder element is by

indicating the circuit condition. It produces when the device is in the reenergized or non-operated state. In the following illustration (figure 2.30), input 1 (I0.0) is programmed as a normally open (NO) contact. In this condition, power will not flow through the contacts to the output (Q0.0).

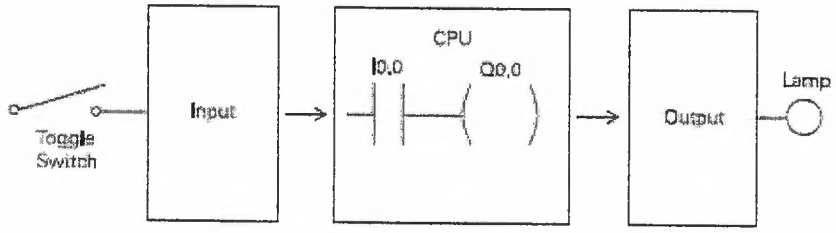


Figure 2.30

When viewing the ladder diagram in the status mode, control elements that are active, or true (logic 1), are highlighted. In the example shown the toggle switch connected to input 1 has been closed. Power can now flow through the control element associated with input 1 (I0.0) and activate the output (Q0.0). The lamp will illuminate, see the figure 2.31.

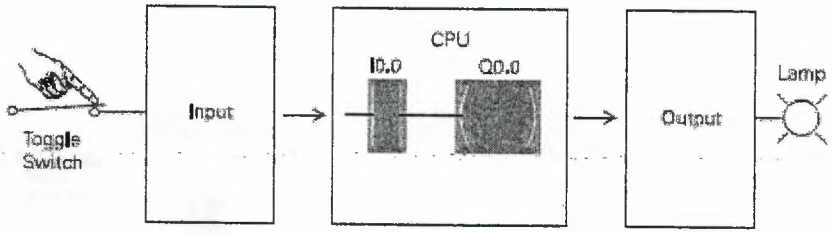


Figure 2.31

2.29 Forcing

Forcing is another useful tool in the commissioning of an application. It can be used to temporarily override the input or output status of the application in order to test and debug the program. The force function can also be used to override discrete output points. The force function can be used to skip portions of a program by enabling a jump instruction with a forced memory bit. Under normal circumstances the toggle switch, shown in the

illustration below (figure 2.32), would have to be closed to enable input 1 (I0.0) and turn on the output light. Forcing enables input 1 even though the input toggle switch is open. With input 1 forced high the output light will illuminate. When a function is forced the control bit identifier is highlighted. The element is also highlighted because it is on.

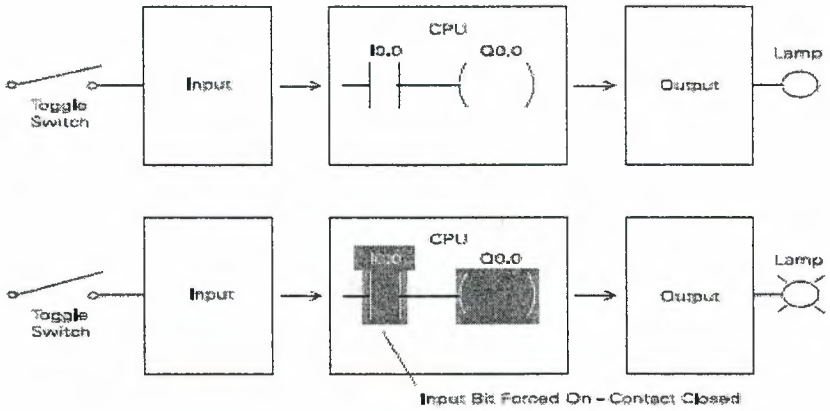


Figure 2.32

The next figure (figure 2.33) shows the appearance of ladder elements in the off, forced, and on condition.

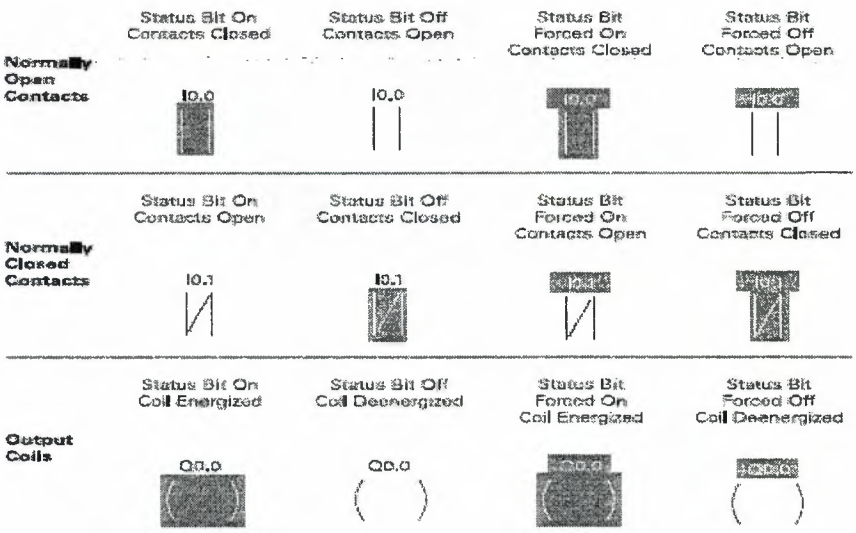


Figure 2.33

2.30 Discrete Inputs/Outputs

To understand discrete control of a programmable controller the same simple lamp circuit illustrated in (figure 2.34) with forcing will be used. This is only for instructional purposes, as a circuit this simple would not require a programmable controller. In this example the lamp is off when the switch is open and on when the switch is closed.

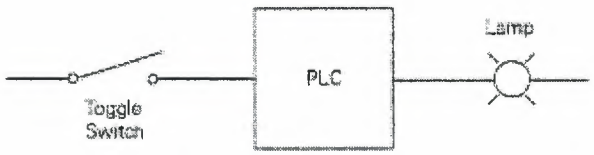


Figure 2.34

2.31 Wiring

To accomplish this task, a switch is wired to the input of the PLC and an indicator light is wired to output terminal.

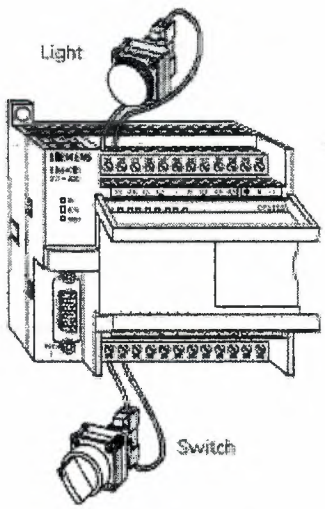


Figure 2.35 Wiring

The following drawing (figure 2.36) illustrates the sequence of events. A switch is wired to the input module of the PLC. A lamp is wired to the output module. The program is in the CPU. The CPU scans the inputs. When it finds the switch open I0.0 receives a binary 0. This instructs Q0.0 to send a binary 0 to the output module. The lamp is off. When it finds the switch closed I0.0 receives a binary 1. This instructs Q0.0 to send a binary 1 to the output module, turning on the lamp.

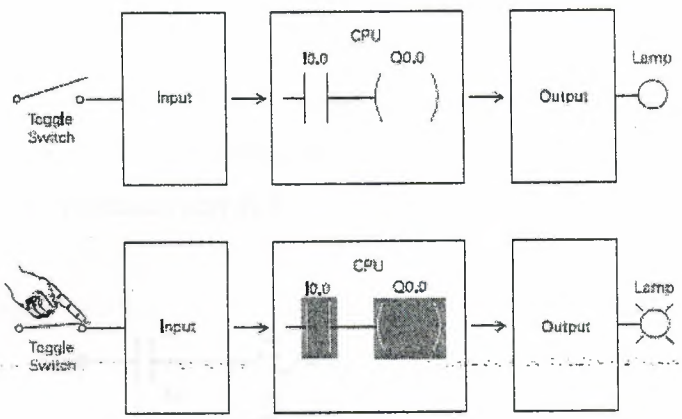


Figure 2.36

2.32 Program Instruction

When the switch is open the CPU receives logic 0 from input I0.0. The CPU sends logic 0 to output Q0.0 and the light is off, the ladder representation is illustrated in (figure 2.37).

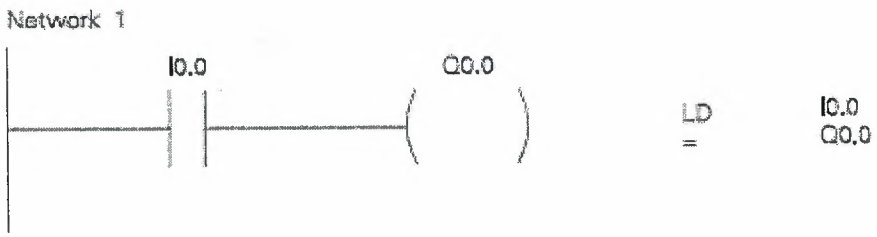


Figure 2.37

When the switch is closed the CPU receives logic 1 from input I0.0. The CPU sends logic 1 to output Q0.0, thus activating Q0.0. The light turns on, se the next figure (2.38).

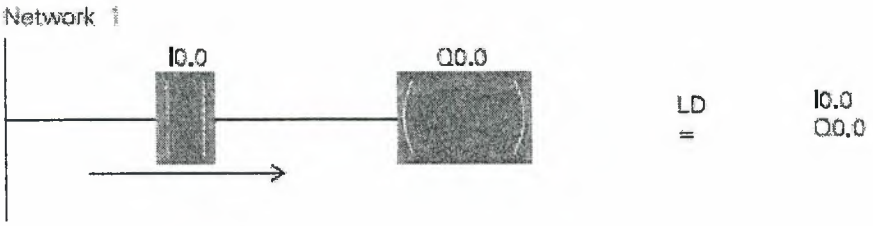


Figure 2.38

2.33 Motor Starter Example

The following example involves a motor start and stop circuit. The line diagram illustrates how a normally open and a normally closed pushbutton might be used in a control circuit. In this example a motor started (M) is wired in series with a normally open momentary pushbutton (Start), a normally closed momentary pushbutton (Stop), and the normally closed contacts of an overload relay (OL).

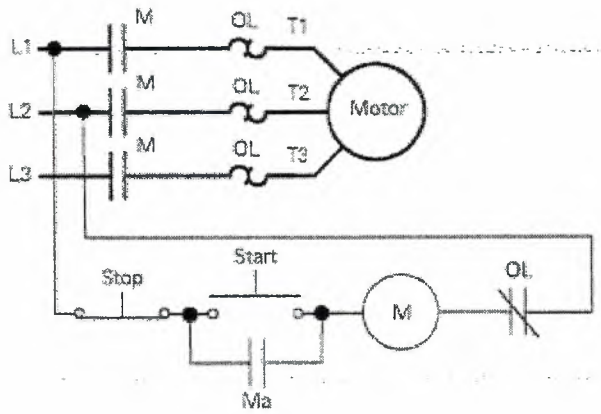


Figure 2.39 An example of motor starter

Momentarily depressing the Start pushbutton completes the path of current flow and energizes the motor starter (M), see the figure 2.40.

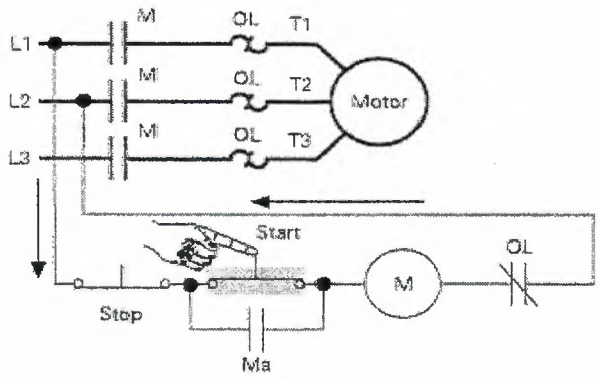


Figure 2.40

This closes the associated M and Ma (auxiliary contact located in the motor starter) contacts. When the Start button is released a holding circuit exists to the M contactor through the auxiliary contacts Ma. The motor will run until the normally closed Stop button is depressed, (see the next figure 2.41) or the overload relay opens the OL contacts, breaking the path of current flow to the motor starter and opening the associated M and Ma contacts.

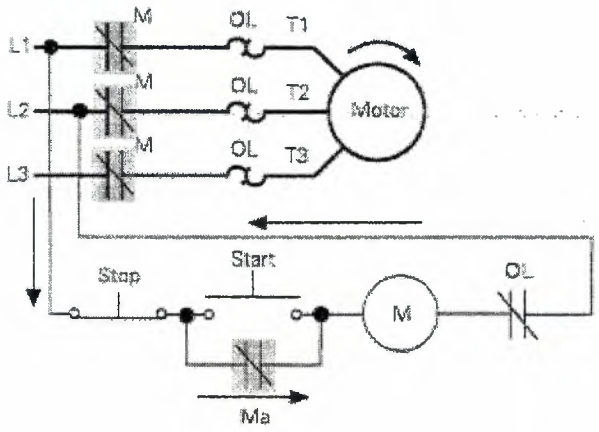


Figure 2.41

The control task illustrated in (figure 2.42) also can be accomplished with PLC.

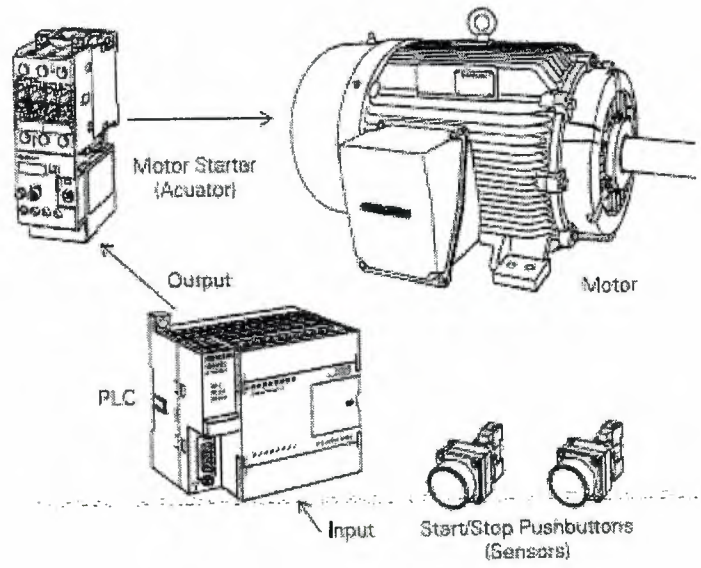


Figure 2.42

2.34 Program Instruction for Motor

A normally open Start pushbutton is wired to the first input (I0.0), a normally closed Stop pushbutton is wired to the second input (I0.1), and normally closed overload relay contacts (part of the motor starter) are connected to the third input (I0.2).

The first input (I0.0), second input (I0.1), and third input (I0.2) form an AND circuit and are used to control normally open programming function contacts on Network 1. I0.1 status bit is logic 1 because the normally closed (NC) Stop Pushbutton is closed. I0.2 status bit is logic 1 because the normally closed (NC) overload relay (OL) contacts are closed. Output Q0.0 is also programmed on Network 1. In addition, a normally open set of

contacts associated with Q0.0 is programmed on Network 1 to form an OR circuit. A motor starter is connected to output Q0.0, see the next figure (figure 2.43).

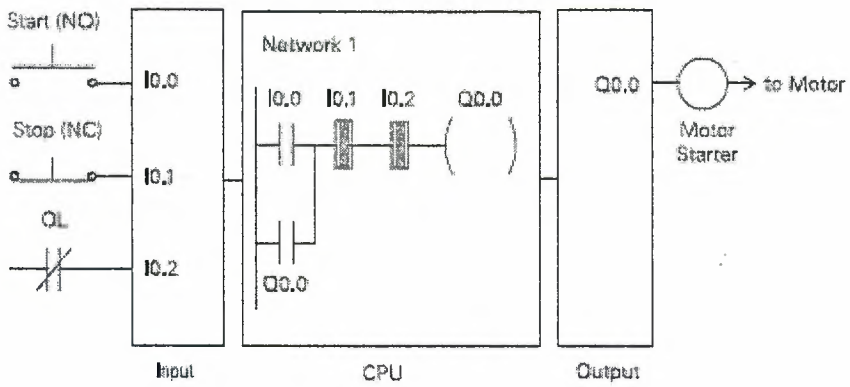


Figure 2.43

When the Start pushbutton is depressed the CPU receives logic 1 from input I0.0. This causes the I0.0 contact to close. All three inputs are now logic 1. The CPU sends logic 1 to output Q0.0. The motor starter is energized and the motor starts, as shown in (figure 2.44).

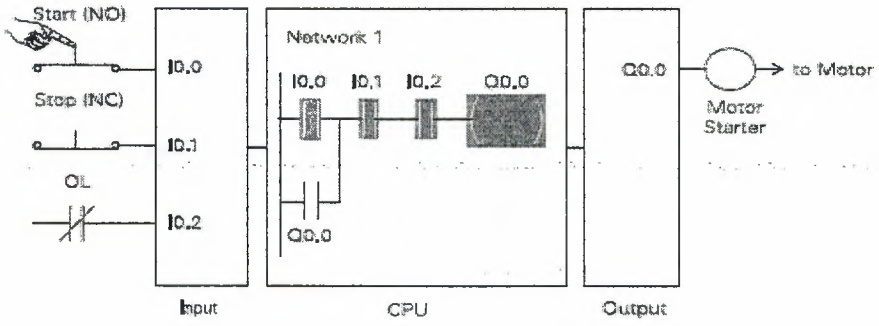


Figure 2.44

When the Start pushbutton is pressed, output Q0.0 is now true and on the next scan, when normally open contact Q0.0 is solved, the contact will close and output Q0.0 will stay on even if the Start pushbutton has been released, as shown in (figure 2.45).

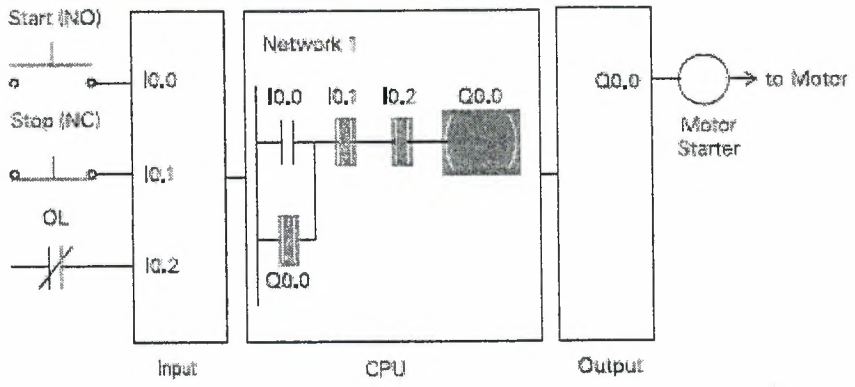


Figure 2.45

The motor will continue to run until the Stop pushbutton is depressed. Input I0.1 will now be logic 0 (false). The CPU will send a binary 0 to output Q0.0. As in (figure 2.46), the motor will turn off.

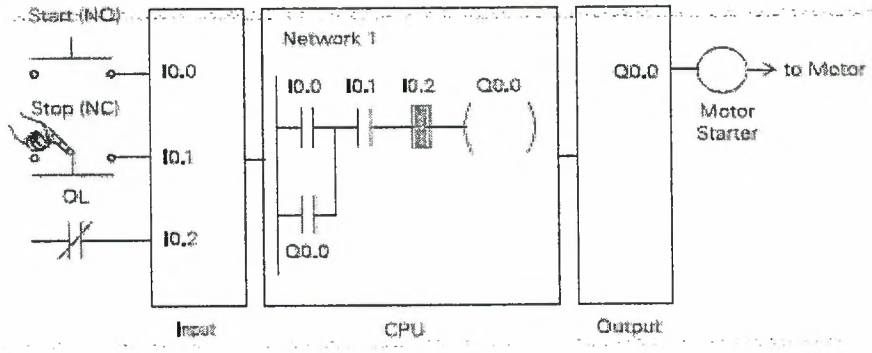


Figure 2.46

When the Stop pushbutton is released I0.1 logic function will again be true and the program ready for the next time the Start pushbutton is pressed, see (figure 2.27).

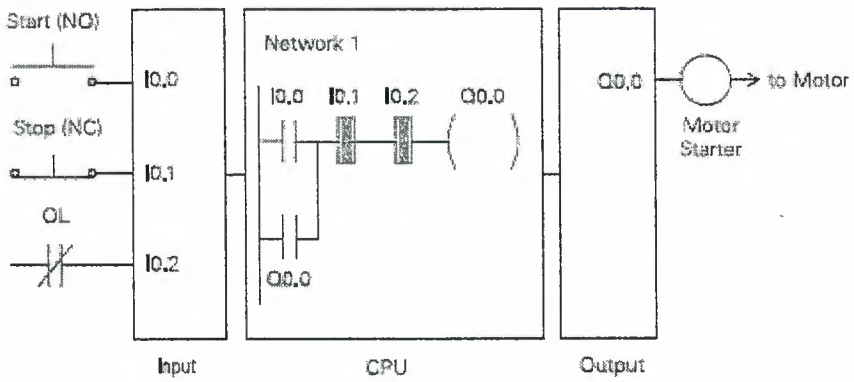


Figure 2.47

2.35 Expanding the Application

The application can be easily expanded to include indicator lights for RUN and STOP conditions. In this example, a RUN indicator light is connected to output Q0.1 and a STOP indicator light is connected to output Q0.2.

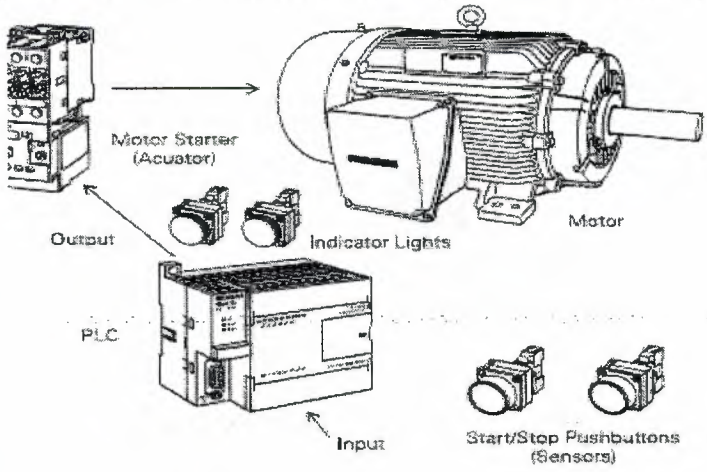


Figure 2.48 Expanding Machine

It can be seen from the ladder logic that a normally open output Q0.0 is connected on Network 2 to output Q0.1 and a normally closed Q0.0 contact is connected to output Q0.2 on network 3. In a stopped condition output Q0.0 is off. The normally open Q0.0 contacts on Network 2 are open and the RUN indicator, connected to output Q0.1 light is off. The normally closed Q0.1 on Network 3 lights are closed and the STOP indicator light, connected to output Q0.2 is on, as illustrated in (figure 2.49).

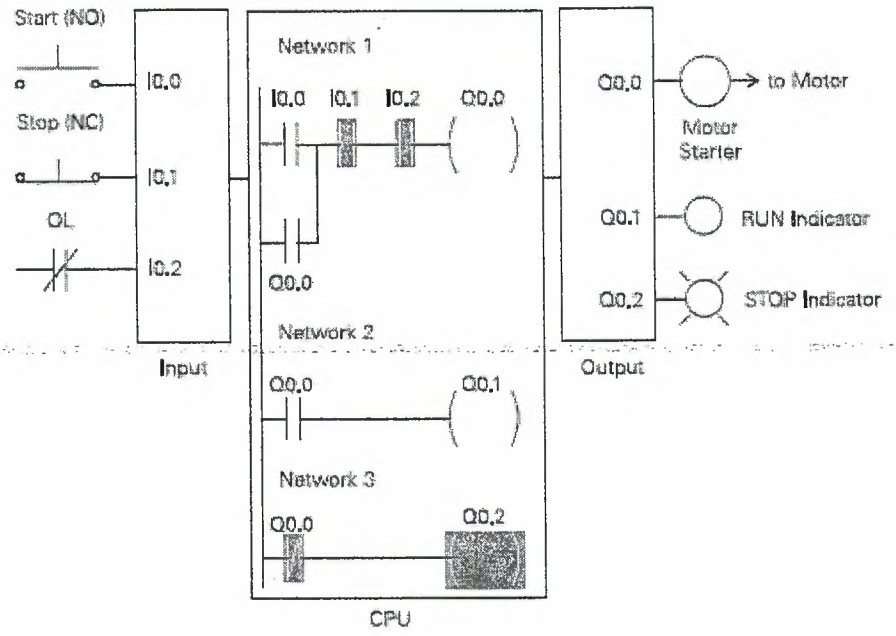


Figure 2.49

When the PLC starts the motor output Q0.0 is now logic high (On). The normally open Q0.0 contacts on Network 2 now switch to a logic 1 (closed) and output Q0.1 turns the RUN indicator on. The normally closed Q0.0 contacts on Network 3 switch to a logic 0 (open) and the STOP indicator light connected to output Q0.2 is now off, like illustrated in (figure 2.50).

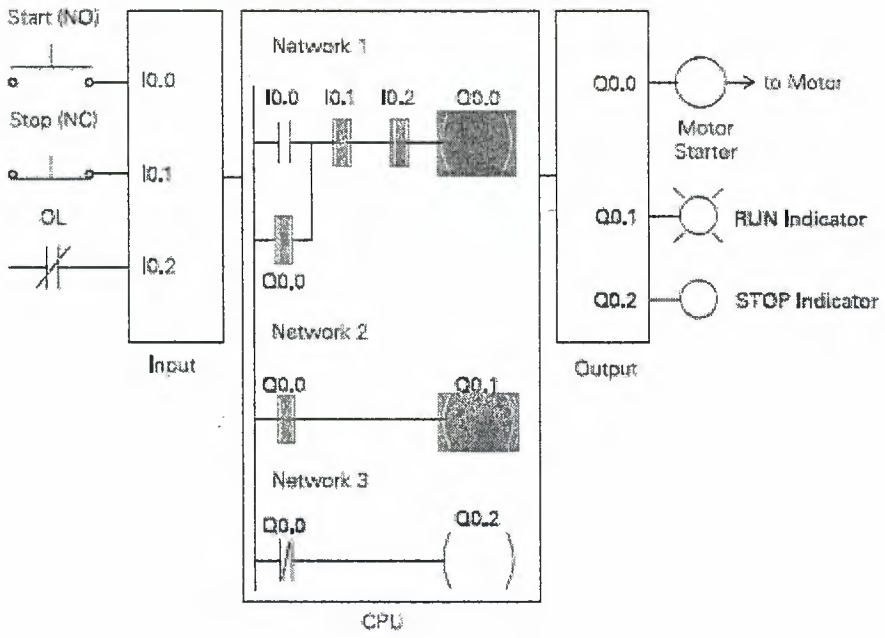


Figure 2.50

2.36 Adding a Limit Switch

Adding a limit switch with normally open contacts to input I0.3 can further expand the application.

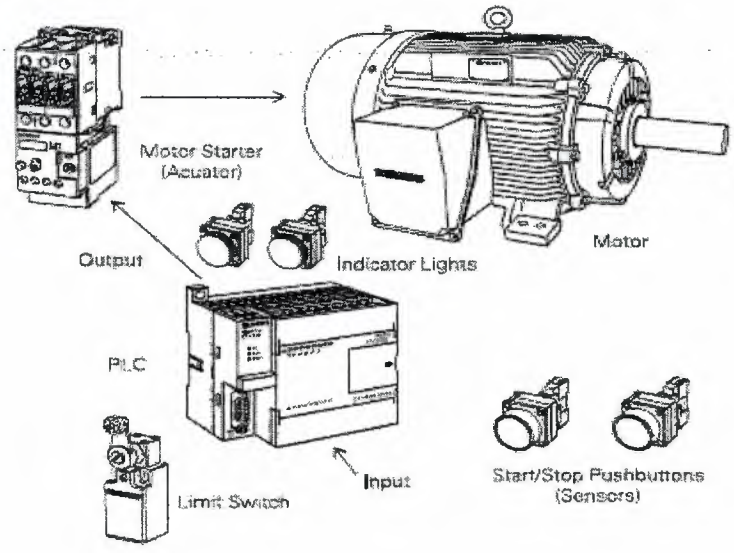


Figure 2.51 Adding a limit switch

A limit switch could be used to stop the motor or prevent the motor from being started. An access door to the motor, or its associated equipment, is one example of a limit switch's use. If the access door is open, the normally open contacts of LS1 connected to input I0.3 are open and the motor will not start, as illustrated in (figure 2.52).

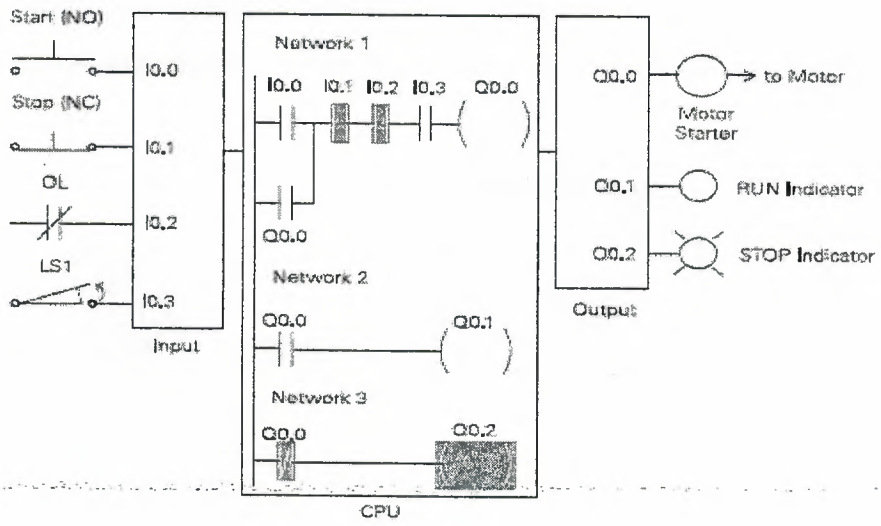


Figure 2.52

When the access door is closed, the normally open contacts on the limit switch (LS1) are closed. Input I0.3 is now on (logic 1), and the motor will start when the Start pushbutton is pressed, see (figure 2.53).

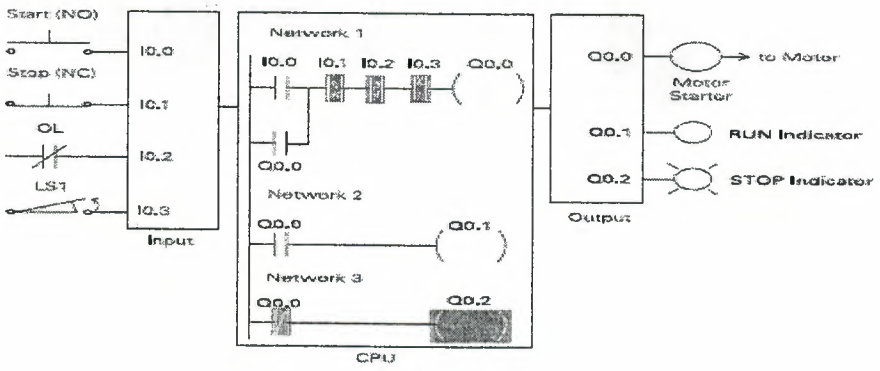


Figure 2.53

-Expansion The PLC program can be expanded to accommodate many commercial and industrial applications. Additional Start/Stop pushbuttons and indicator lights can be added for remote operation, or control of a second motor starter and motor. Over travel limit switches can be added along with proximity switches for sensing object position. In addition, expansion modules can be added to further increase the I/O capability. The number of I/Os and amount of memory available on the PLC only limits the applications, see (figure 2.54).

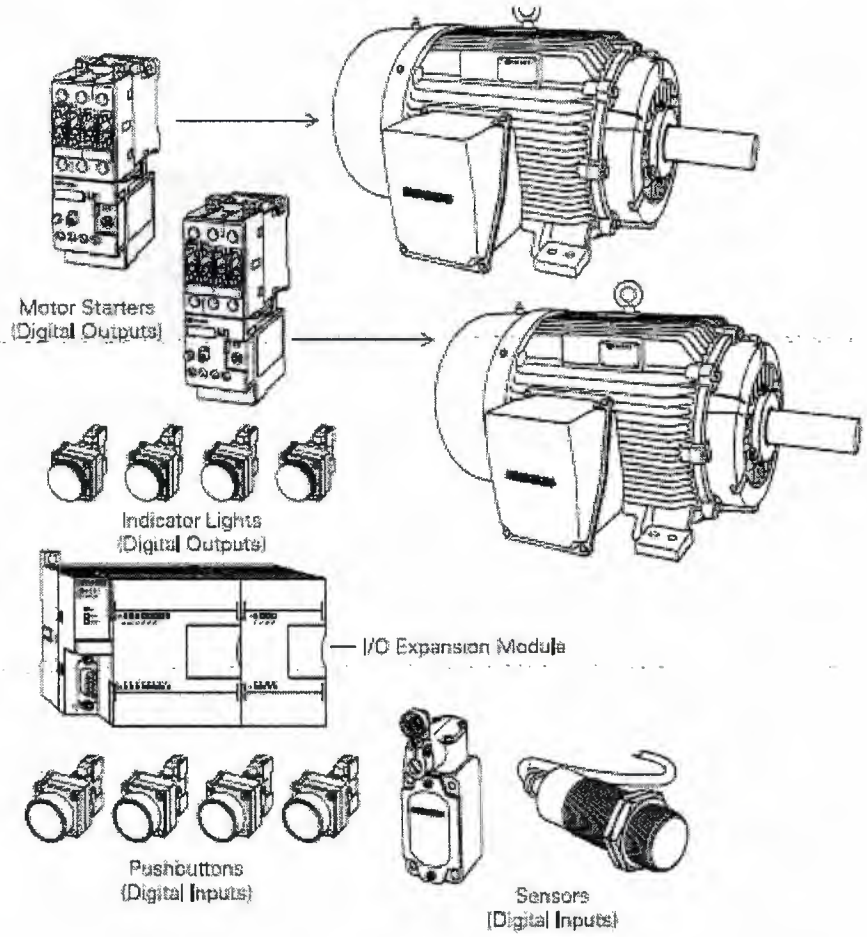


Figure 2.54

2.37 Analog Inputs and Outputs

PLCs must also work with continuous or analog signals. Typical analog signals are 0 - 10 VDC or 4 - 20 mA. Analog signals are used to represent changing values such as speed, temperature, weight, and level. A PLC cannot process these signals in an analog form. The PLC must convert the analog signal into a digital representation. An expansion module, capable of converting the analog signal, must be used. The S7-200 analog modules convert standard voltage and current analog values into a 12-bit digital representation. The digital values are transferred to the PLC for use in register or word locations. In addition, analog modules are available for use with thermocouple and RTD type sensors used in to achieve a high level of accuracy in temperature measurement.

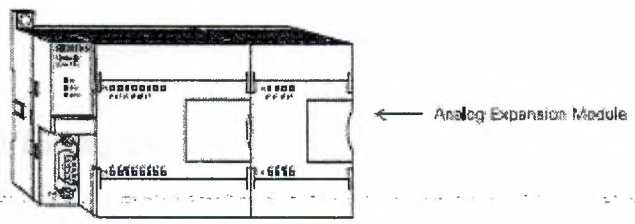


Figure 2.55 Analog Expansion Module

2.38 Timers

Timers are devices that count increments of time. Traffic lights are one example where timers are used. In this example timers are used to control the length of time between signal changes.

Boxes in ladder logic represent timers. When a timer receives an enable, the timer starts to time. The timer compares its current time with the preset time. The output of the timer is logic 0 as long as the current time is less than the preset time. When the current time is greater than the preset time the timer output is logic 1. S7-200 uses three types of timers: On- Delay (TON), Retentive On-Delay (TONR), and Off-Delay (TOF).

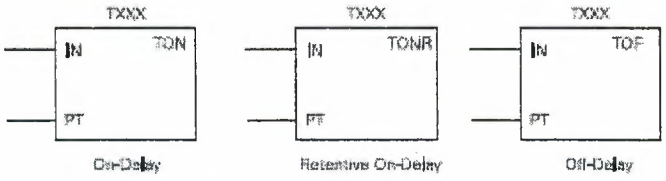


Figure 2.56 Timers

2.38.1 S7-200 Timers

S7-200 timers are provided with resolutions of 1 millisecond, 10 milliseconds, and 100 milliseconds. The maximum value of these timers is 32.767 seconds, 327.67 seconds, and 3276.7 seconds, respectively. By adding program elements, logic can be programmed for much greater time intervals.

2.38.2 Hard-Wired Timing Circuit

Timers used with PLCs can be compared to timing circuits used in hard-wired control line diagrams. In the following example, a normally open (NO) switch (S1) is used with a timer (TR1). For this example the timer has been set for 5 seconds. When S1 is closed, TR1 begins timing. When 5 seconds have elapsed, TR1 will close its associated normally open TR1 contacts, illuminating pilot light PL1. When S1 is open, reenergizing TR1, the TR1 contacts open, immediately extinguishing PL1. This type of timer is referred to as ON delay. ON delay indicates that once a timer receives an enable signal, a predetermined amount of time (set by the timer) must pass before the timer's contacts change state.

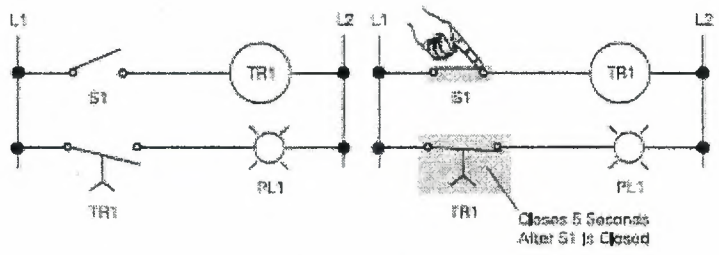


Figure 2.57 Hard wired timing circuit

2.38.3 On-Delay (TON)

When the On-Delay timer (TON) receives an enable (logic 1) at its input (IN), a predetermined amount of time (preset time - PT) passes before the timer bit (T-bit) turns on, see (figure 2.58). The T-bit is a logic function internal to the timer and is not shown on the symbol.

The timer resets to the starting time when the enabling input goes to logic 0.

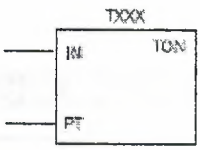


Figure 2.58

2.38.4 Retentive On-Delay (TONR)

The Retentive On-Delay timer (TONR) functions in a similar manner to the On-Delay timer (TON). There is one difference. The Retentive On-Delay timer times as long as the enabling input is on, but does not reset when the input goes off. The timer must be reset with a RESET (R) instruction, see (figure 2.59).

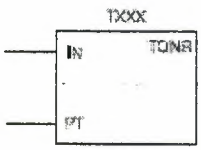


Figure 2.59

2.38.5 Off-Delay (TOF)

The Off-Delay timer is used to delay an output off for a fixed period of time after the input turns off. When the enabling bit turns on the timer bit turns on immediately and the value is set to 0. When the input turns off, the timer counts until the preset time has elapsed before the timer bit turns off.

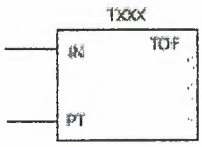


Figure 2.60 Off-Delay (TOF)

The S7-200s have 256 timers. The specific T number chosen for the timer determines its time base and whether it is TON, TONR, or TOF, check (Table 2.1).

Timer Type	Resolution	Maximum Value	Timer Number
TONR	1 ms	32.767 seconds	T0, T64
	10 ms	327.67 seconds	T1-T4, T65-T68
	100 ms	3276.7 seconds	T5-T31, T69-T95
TON, TOF	1 ms	32.767 seconds	T32, T96
	10 ms	327.67 seconds	T33-T36, T97-T100
	100 ms	3276.7 seconds	T37-T63, T101-T255

Table 2.1

2.39 Counters

Counters used in PLCs serve the same function as mechanical counters. Counters compare an accumulated value to a preset value to control circuit functions. Control applications that commonly use counters include the following:

- Count to a preset value and cause an event to occur
- Cause an event to occur until the count reaches a preset value

A bottling machine, for example, may use a counter to count bottles into groups of six for packaging.

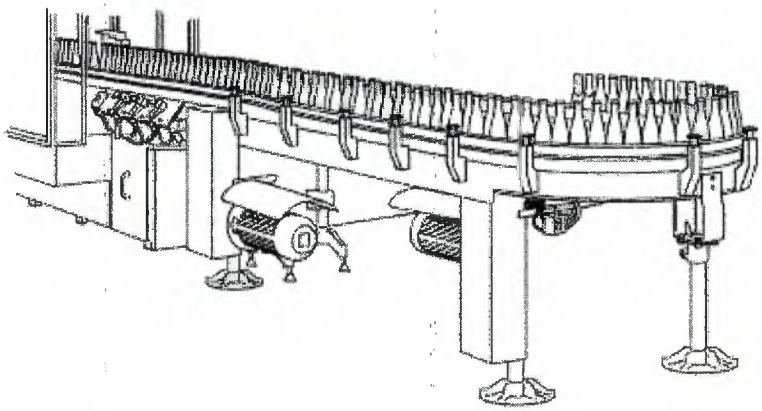


Figure 2.62 Counter's Application

Boxes in ladder logic represent counters. Counters increment/decrement one count each time the input transitions from off (logic 0) to on (logic 1). The counters are reset when a RESET instruction is executed. S7-200 uses three types of counters: up counter (CTU), down counter (CTD), and up/down counter (CTUD), see (figure 2.63).

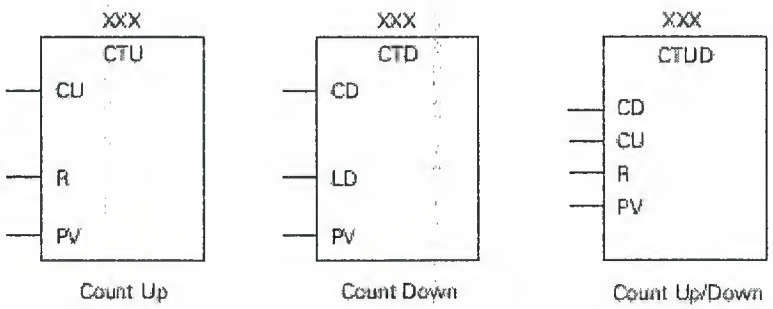


Figure 2.63

2.39.1S7-200 Counters

There are 256 counters in the S7-200, numbered C0 through C255. The same number cannot be assigned to more than one counter. For example, if an up counter is assigned number 45, a down counter cannot also be assigned number 45. The maximum count value of a counter is ±32,767.

2.39.2 Up Counter

The up counter counts up from a current value to a preset value (PV). Input CU is the count input. Each time CU transitions from logic 0 to logic 1 the counter increments by a count of 1. Input R is the reset. A preset count value is stored in PV input. If the current count is equal to or greater than the preset value stored in PV, the output bit (Q) turns on (not shown).

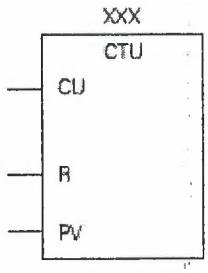


Figure 2.64 Up counter

2.39.3 Down Counter

The down counter counts down from the preset value (PV) each time CD transitions from a logic 0 to a logic 1. When the current value is equal to zero the counter output bit (Q) turns on (not shown). The counter resets and loads the current value with e
preset value (PV) when the load input (LD) is enabled.

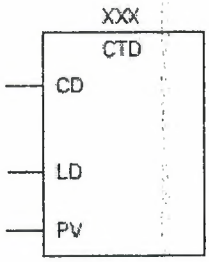


Figure 2.65 Down counter

2.39.4 Up/Down Counter

The up/down counter counts up or down from the preset value each time either CD or CU transitions from logic 0 to logic 1. When the current value is equal to the preset value, the output QU turns on. When the current value (CV) is equal to zero, the output QD turns on. The counter loads the current value (CV) with the preset value (PV) when the load input

(LD) is enabled. Similarly, the counter resets and loads the current value (CV) with zero when the reset (R) is enabled. The counter stops counting when it reaches preset or zero.

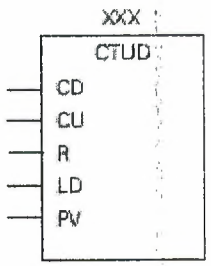


Figure 2.66 Up/Down counter

A mechatronic design of a coin counter:

For this problem, we need three sensors to classify the coins been put in the system. The sensors will classify the coins according to their diameter. The process of classification will be as shown in the figure 3.1 shown below:

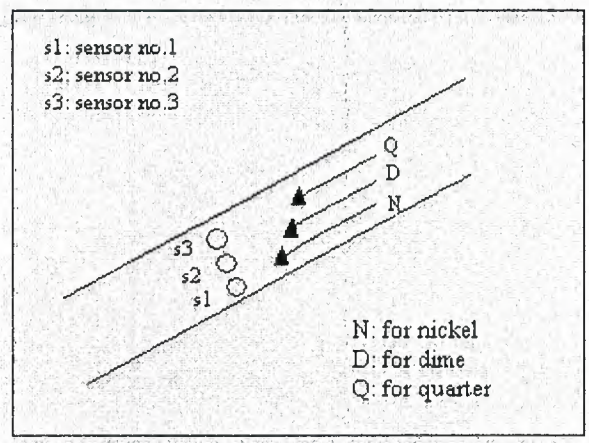


Figure 3.1 Classification processes of coins

In the figure3.1, it's clear that the diameter of (for example) the nickel will be N, if it passed, the sensor1 will give a signal to the plc, and the plc then will decide (by some program) that the coin passed was a nickel. And if the dime passed through the corridor, the sensors 1 and 2 will sense it, then, both of them will send signals to the plc and it will decide (by the same program) that the coin passed was a dime. The same process will be applied to the quarter coin.

Now, we insured that the coins will be classified by their diameters and the mechanical part of the system did its job successfully. Here, we have to look after the electronic part of the system which is represented by the plc program, how will it be successful and how will it deal with our mechanical part. Before starting our program, we have, first, to put some guide lines to be followed by the programmer to achieve the goals of the experiment.

The program is expected to have errors in counting part and confusion in receiving the signals, and to avoid those expected errors, we don't have to rush in putting the steps of the program.

Difficulties faced in building the PLC program:

While I was building the plc program, I found some difficulties which were not expected. The following program was the first one that I had put:

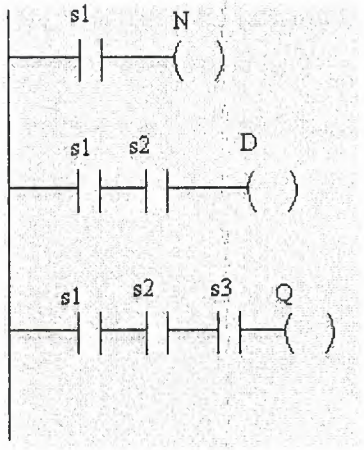


Figure 3.2

This was made in order to receive the signals from the mechanical part and classify them by deciding that the signal is for nickel or dime or quarter, the decision will be referred to the received signal as follows, if the sensor1, only, sent a signal, that means a nickel has passed through it, and that's right, but don't forget that when you have a dime, the sensor1 will sense it too, then, we will have an error. The system will behave like there is a nickel; it will add one to the counter responsible of the nickel while it had to add one to the counter of dime. Also, we will have the same error when a quarter passes through the corridor. That is the sensors 1 and 2 will send signals to the program and it will behave like there were both (nickel and dime). It is not important that we succeed sensing the quarter while we have an error in the system. So, how can we avoid such an error?

Solution:

To avoid the previous problem, we have to notice the point which caused it. The point can be explained in few words: while we are using just three sensors to sense the

three types of coins, and one type only needs one sensor to sense it, the others need more than one sensor, of the same previous sensors. When we have a signal from the sensor3, that makes it sure that the quarter passed through the corridor, the diameters of the other types are not enough for them to reach the point which the sensor3 is placed at. So, we can replace network (3) with another containing just the sensor3 as a stipulation to activate Q (or it is not important). But the exact aim of the correction is to prevent D to activate when sensor1 and sensor give signals caused from passing of a quarter, also preventing N to activate when sensor1 gives a signal caused from passing of a quarter or a dime. For the 1st case, we have to add another gate to the second network to insure that the sensors (1 and 2) are not activating by a quarter, and for the 2nd case, we have to add two gates to the first network to insure that the sensor1 is not activating by a quarter or a dime, as shown in the figure 3.3:

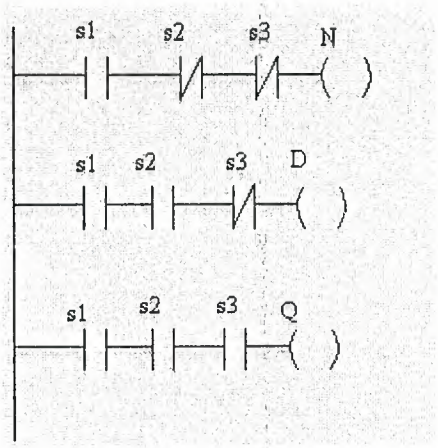


Figure 3.3

Now, after we did this program, when we have a quarter passing through the corridor, all of the sensors will be sensing it, but neither the D nor the N will be active with it, that's will happen because we did not program them to activate any time the s1 gives signal. The program is stipulating for the N to activate that the sensors (2 and3) are not sensing to insure that is for nickel. Also for the dime we are making a condition to it to activate that sensor3 is not sensing, because if all of the sensors are sensing that means we are having a quarter passing through the corridor not a dime. So we have solved the first error in the system we are building.

The second problem I faced was the counting process, how to translate the gates Q, D, and N into existing gates in the PLC programming to count the quarters, dimes, and nickels. I had, as a first try, added counters instead of the Q, D, and N symbols as coming in the figure 3.4. Then, the counter (C0) represents N and counts down from M to 0 to reach the maximum possible number of nickels can be counted by the system. So, the counting process will be dependent, as same as N, on the sensors s1, s2, and s3, sensor one should be sensing and both 2 and 3 should be off in order to let the network1 have the output of activating the counter (C0). Also the other networks are replaced with (C1) and (C2) to count from M to 0 to reach the maximum possible number of nickels and quarters can be counted by the system.

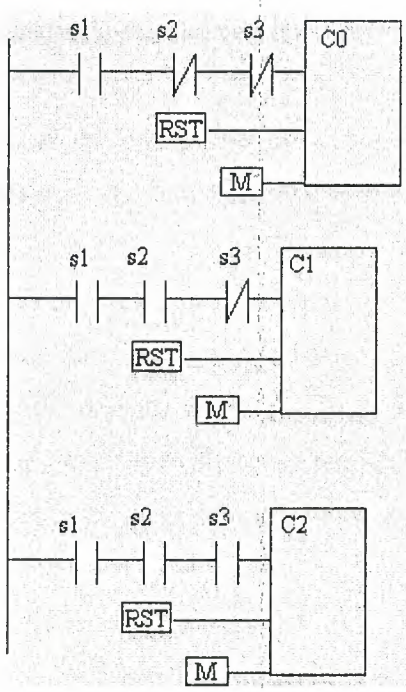


Figure 3.4

Conclusion

The coin counter design was modeled using a mechatronic system, containing, of course, a mechanical part and an electronic part. The electronic part was the PLC program, it controlled the counting process completely, then, a logic circuit had to be put in the last step of the system, to receive the pulses been produced by the PLC and count them one by one.

Also, we can use a display unit in order to see the final result of the counter. A perfect choice can be the dual-7-seg-display unit, to allow the system to count till 99 coins.

We can add other circuits to the system to improve its properties, such as, multipliers, adders, and modern counters and display units.

The multipliers can be added to multiply the number of the coins been inserted to the system by their unit values in order to perform the total value of them and display it beside their number.

The adders can improve the system too, they can add the values together to perform the final quantity of money been inserted, and this is useful in the banks and the supermarkets and such applications, it is ready to be developed. We can, also, develop the mechanical system, and distribute the responsibilities between the mechanical part and the electronic part.

Finally, the mechatronic system is depending on the designer's model that will be put. The model should satisfy the conditions of the using field, and the design in this project is just one of hundreds can be built.

REFERENCES:

- Introduction to mechatronics and measurement systems second edition by David G. Alciatore and Michael B. Histand Second Edition (2003; available July, 2002)
- Siemens STEP 2000 siries Basics of PLCs
- Siemens STEP 2000 deries Basics of sensors
- A CAPSTONE DESIGN PROJECT FOR ENGINEERING TECHNOLOGY STUDENTS Sohail Anwarl, Ph.D., Damian Marchetti2, 30th ASEE/IEEE Frontiers in Education Conference.