NEAR EAST UNIVERSITY

Faculty of Engineering

Department of Electrical and Electronic Engineering

PROGRAMMABLE LOGIC CONTROLLER (PLC)

Graduation Project
EE- 400

Student: Husain Ahmed DAHALAN (991748)

Supervisor: Mr. Özgür C. ÖZERDEM

Nicosia – 2003

ACKNOWLEDGMENT

*"This project was not possible to be prepared without the guidance and the support of my supervisor **Mr. Özgür C. Özerdem**.*

I am indebted to him for his complete support and showing me the guidance throughout all the stages of the preparation, and providing his constructive comments.

*So I would like to take this opportunity to thank **Mr. Özgür** for every single help and support, not just throughout this project but also through the courses which he provides to the students in the Department of Electrical and Electronic Engineering, because through these courses have gained a lot of knowledge, which helped me in the preparation of this project.*

*I thank my family specially my brother **Ashraf** for their constant encouragement and support during the preparation of this project.*

*I would like also to thank my friends specially **Amjad Hammouda** , **Mahmoud Almassri** for their advice and support."*

TABLE OF CONTENTS

INTRODUCTION

A programmable logic controller **(PLC)** is a device that was invented to replace the necessary sequential relay circuits for machine control. The **PLC** works by looking at its inputs and depending upon their state, turning on/of its outputs. The user enters a program, usually via software, that gives the desired results.

P1C's are used in many real word applications. If there is industry present, chances are good that there is a **PLC** present. If you are involved in a machining, packaging, material handling, automated assembly or countless other industries you are probably already using them. If you are not, you are wasting money and time.

Almost any application that needs some type of electrical control has for a **PLC**.

For example, let's assume that when a switch turns on we want turn a solenoid on for 5 seconds and then turn it off regardless of how long the switch is on for.

We can do this with a simple external timer. But what if the process included 10 switches and solenoids? We would need 10 external timers. What if the process also needed to count how may time the switches individually turned on? We need a lot of external counters.

As you can see the bigger the process the more of a need eve have for a PLC.

We can simple program the PLC to count its inputs and turn the solenoids on for the specified time.

This site gives enough formation to be able to write programs far more complicated than the simply one above. We will take a look at what is considered to be the 'top 20' PLC instructions. It can safely estimated that with a firm understanding of these instructions one can solve more than 80% of the applications insistence.

1. Background of PLC

1.1. History of PLC

In the late 1960's large, complex panels of electromechanical relays controlled assembly lines in the major automotive plants. These relay panels controlled the sequence of events that was required to assemble a particular vehicle. At the end of each model year the relay panels had to be extensively rewired requiring the plants to be shut down for up to a month. Often it was more economical to scrap the old panels and replace them with new ones. General Motors was looking for a way to save money on this costly, time-consuming process. In 1968 its Hydra-Matic Division designed the first Programmable Logic Controller (PLC).

The PLC replaced the relay panel with a computer. Rewiring the relay panel was replaced with reprogramming the computer. The model changeover that once took weeks was reduced to a matter of days. By the year 2000 the control of the assembly line had evolved to the point where a plant like GM's Lords town Assembly Plant was able to assemble Cavaliers and have 2 door and 4 door vehicles, vehicles in a variety of trims and with a variety of options in many different colors, even vehicles with the steering wheel on the right hand side, roll off the same assembly line one after the other.

Although the PLC was a great advance in controlling complex processes, it took a while to become widely adopted. Early PLCs had a reputation for being unreliable and required highly trained programmers to make any changes to the program. By today's standards they were also expensive and very limited in terms of their capabilities. Over the years manufacturers 'hardened' their PLCs to withstand the industrial environments in which they were installed. As computers advanced so did PLCs becoming more powerful and less expensive. PLC programming also became more user friendly and more easily understood by a much wider segment of the workforce.

1.2. What is a PLC?

PLCs are often defined as miniature industrial computers that contain hardware and software that is used to perform control functions. A PLC consists of two basic sections: the central processing unit (CPU) and the input/output interface system. The CPU, which controls all PLC activity, can further be broken down into the processor and memory system. The input/output system is physically connected to field devices (e.g., switches, sensors, etc.) and provides the interface between the CPU and the information providers (inputs) and controllable devices (outputs). To operate, the CPU "reads" input data from connected field devices through the use of its input interfaces, and then "executes", or performs the control program that has been stored in its memory system. Programs are typically created in ladder logic, a language that closely resembles a relay-based wiring schematic, and are entered into the CPU's memory prior to operation. Finally, based on the program, the PLC "writes", or updates output devices via the output interfaces. This process, also known as scanning, continues in the same sequence without interruption, and changes only when a change is made to the control program.

A PLC (i.e. Programmable Logic Controller) is a device that was invented to replace the necessary sequential relay circuits for machine control. The PLC works by looking at its inputs and depending upon their state, turning on/off its outputs. The user enters a program, usually via software, that gives the desired results.

PLCs are used in many "real world" applications. If there is industry present, chances are good that there is a plc present. If you are involved in machining, packaging, material handling, automated assembly or countless other industries you are probably already using them. If you are not, you are wasting money and time. Almost any application that needs some type of electrical control has a need for a plc.

For example, let's assume that when a switch turns on we want to turn a solenoid on for 5 seconds and then turn it off regardless of how long the switch is on for. We can do this with a simple external timer. But what if the process included 10 switches and solenoids? We would need 10 external timers.

What if the process also needed to count how many times the switches individually turned on? We need a lot of external counters.

As you can see the bigger the process the more of a need we have for a PLC. We can simply program the PLC to count its inputs and turn the solenoids on for the specified time.

This site gives you enough information to be able to write programs far more complicated than the simple one above. We will take a look at what is considered to be the "top 20" plc instructions. It can be safely estimated that with a firm understanding of these instructions one can solve more than 80% of the applications in existence.

1.3. Today's PLC

As PLC technology has advanced, so have programming languages and communications capabilities, along with many other important features. Today's PLCs offer faster scan times, space efficient high-density input/output systems, and special interfaces to allow non-traditional devices to be attached directly to the PLC. Not only can they communicate with other control systems, they can also perform reporting functions and diagnose their own failures, as well as the failure of a machine or process.

Size is typically used to categorize today's PLC, and is often an indication of the features and types of applications it will accommodate. Small, non-modular PLCs (also known as fixed I/O PLCs) generally have less memory and accommodate a small number of inputs and outputs in fixed configurations. Modular PLCs have bases or racks that allow installation of multiple I/O modules, and will accommodate more complex applications. When you consider all of the advances PLCs have made and all the benefits they offer, it's easy to see how they've become a standard in the industry, and why they will most likely continue their success in the future.

2. Architecture and Operations of PLC

2.1. Ladder Logic

Ladder logic, ladder diagrams, ladder logic diagrams, elementary diagrams, and line diagrams are all terms referring to a very popular graphical method of describing event driven or time/event driven sequential processes. Ladder diagrams were originally developed to represent non-electronic control circuits consisting of switches, relays, solenoids, indicators, and other components used to control industrial machinery. They earned the name ladder diagrams because they bear some resemblance to a ladder with a number of circuits comprising the rungs of the ladder running between two vertical lines or rails.

Since the vertical lines represent power lines to the controller, every complete path between the two lines must contain exactly one relay coil, solenoid, indicator, or other load. Like the lights in your house, the loads may be placed in parallel but not in series combinations. Any path between the two lines that does not contain a load is a potential short circuit. In addition to these basic rules other common practices for the format of ladder diagrams include:

- All relay coils, solenoids, indicators, or other loads are on the right.
- Switches, contacts, or any other devices that make or break electrical contact are on the left.
- Switches, contacts, or any other devices that make or break electrical contact may be multiple contacts in series, parallel, or series-parallel combinations.

2.1.1. A Recipe for Creating a Ladder Logic Diagram

The following is as close as I have been able to come to a 'recipe' that guarantees success when creating ladder logic diagram. The ladder diagrams that result from this recipe usually have many more control relays and may at first look more complicated than ladder diagrams obtained by other methods, but the logic is clear and all the rungs (In each portion of the diagram - the control portion and the output portion) are very similar to each other.

The example we will use for our recipe is very simple. Suppose we have a tank where we wish to mix two liquids for a specified amount of time. When we press a start button two solenoid operated valves (SOLA and SOLB) will open and the two liquids will begin filling the tank. The tank is equipped with two float switches, one near the bottom of the tank to tell us if the tank is empty and one near the top of the tank to tell us if the tank is full. We will say that the switches are designed so that when the tank is empty float switch 1, FS1 (near the bottom of the tank) will be closed (true) otherwise it will be open (false). Float switch 2, FS2 (near the top of the tank) is designed so that when the tank is full it is closed (true) otherwise it will be open (false). When the tank becomes full we want the two-solenoid valves to close and the mixer motor (M1) to start.

The mixer motor should run for some predetermined amount of time, then stop, and a third solenoid valve (SOLC) should open to drain the tank. When the tank is empty, SOLC should close but the refilling process should not begin until the start button is once again pressed.

- **Define the Process**

The first step is to define the process (figure 2.1). This often involves creating some type of drawing or diagram of the process.

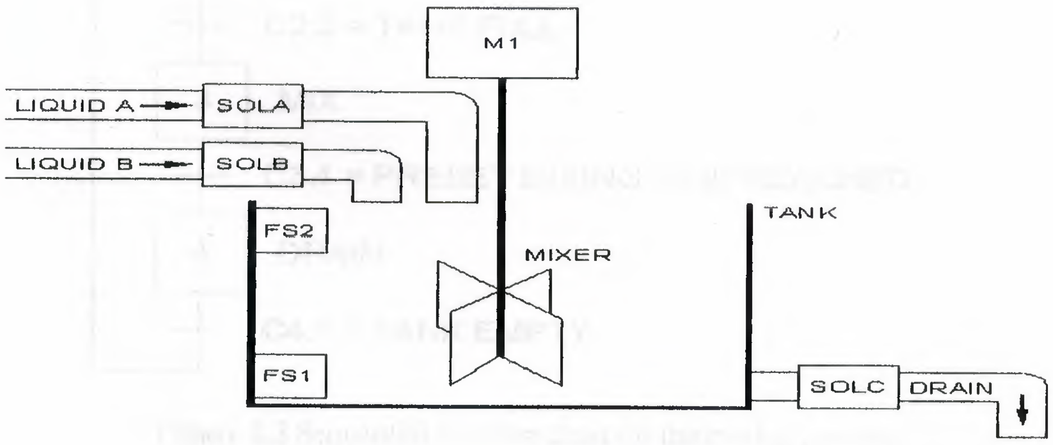


Figure 2.1. Diagram of the mixing process

- **Define the Steps or States**

The second step is to define the steps or states of the process (figure 2.2). Two different tools are used to assist us in defining the steps, the Sequential Function Chart and the State Chart. A Sequential Function Chart for the mixing process is shown below. Each step or state in the process has been numbered and is represented by a box. The flow is from top to bottom in order of the numbers of the steps except after step number four the process can repeat beginning at step one again. Since the flow is usually quite obvious, arrows are generally not drawn on Sequential Function Charts. The condition for advancing from one step to the next is written beside a horizontal line that crosses the transfer path between the two steps. The transfer conditions are given labels of the form Ci.j, where i is the state that the transfer is from and j is the state that the transfer is to.

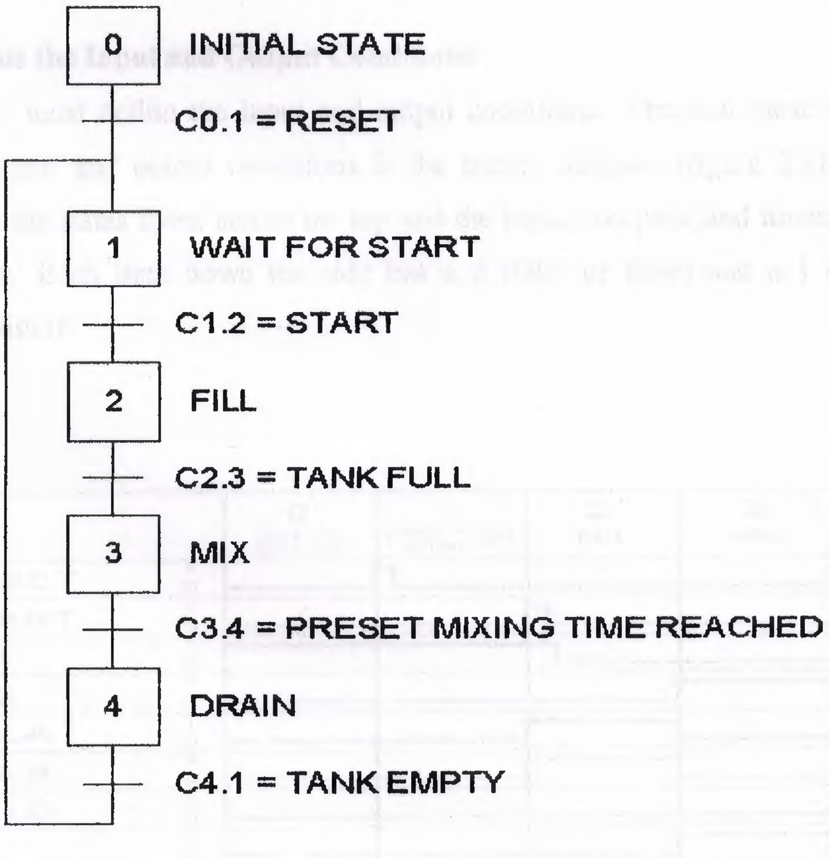


Figure 2.2 Sequential function chart for the mixing process

A state chart is simply a truth table for the outputs from our controller. The steps are listed in the first column of the table and the outputs in the first row. An X indicates that an output is ON while no mark indicates that it is off.

	SOLA	SOLB	SOLC	M1
1				
2	X	X		
3				X
4			X	

Table 2.1. State chart for the mixing process

- **Define the Input and Output Conditions**

Next, we must define the input and output conditions. The tool most often used to define the input and output conditions is the timing diagram (figure 2.3). The timing diagram has the states listed across the top and the inputs, outputs, and timers listed down the left side. Each item down the side has a 0 (OFF or false) and a 1 (ON or true) associated with it.

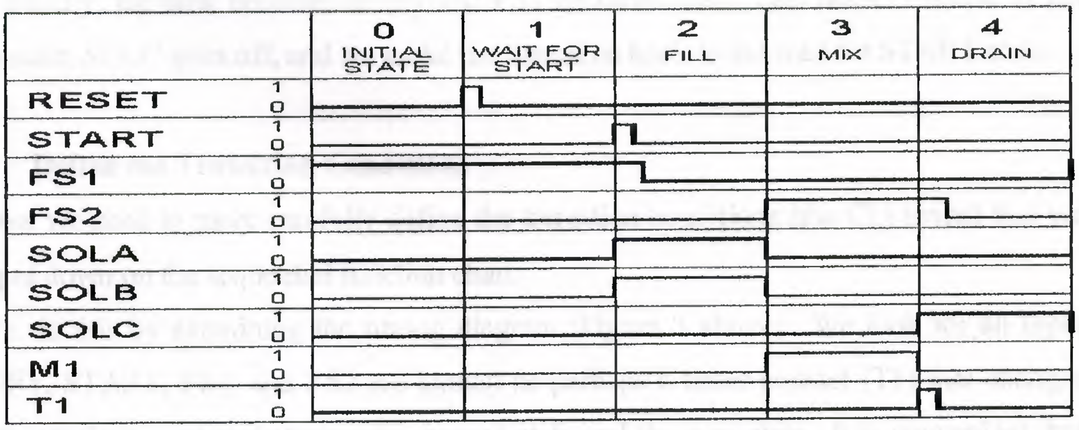


Figure 2.3. Timing diagram for the mixing process.

Some explanation is in order. Both the RESET and START are momentary contact push button switches. The RESET really just provides us with a method to begin. We start in the initial state and when the RESET button is pressed (RESET goes from 0 or OFF to 1 or ON) we transition to state 1, waiting for the START button to be pressed. Notice that since the RESET button is momentary contact, it does not remain in the true (1 or ON) condition for long but returns to false (0 or OFF) when released a short time after it was pressed. We don't really care how long the RESET button remains pressed as long as it is not through the entire step 1.

The START button behaves similarly; however, when the START button is pressed we also want SOLA and SOLB to activate allowing liquids A and B to flow into the tank. A short time after the tank begins to fill FS1 becomes false

(i.e. the tank is no longer empty) and FS1 goes from 1 to 0. SOLA and SOLB remain on until FS2 goes true (the tank becomes full) at which point they go off. When the tank becomes full M1 must also go on. Not shown but also occurring at the transition between states 2 and 3 is activation of the timer coil. This is when the timer begins to time. The length of time that the timer is preset for determines the length of the mix. When the timer times out T1 goes true and we transition from state 3 to state 4. This marks the end of the mix state and the beginning of the drain state so M1 goes off and SOLC goes on. A short time after the tank begins to drain FS2 becomes false. In reality FS2 may have transitioned between true and false many times as the liquid was mixed in the tank. We really don't care as long as it makes the initial transition from false to true when the tank first becomes full. Finally, the tank becomes empty and FS1 becomes true. This marks the end of the drain state, SOLC goes off, and we make the transition back to the wait for START state.

- **Define the Transition Conditions**

Now we need to more carefully define the transition conditions (the Ci.j terms) that we first put down on the sequential function chart.

We do this by examining the timing diagram (Figure 3 above). We look for an input (RESET, START, FS1, and FS2 are inputs) or perhaps a timer contact (T1) that changes exactly on the boundary between the present state and the next state. It is convenient, but certainly not necessary, that the input change from 0 to 1 at the boundary.

Often only one input changes and the choice is therefore obvious but sometimes more than one input will change at the boundary and we must make an arbitrary choice.
From the timing diagram we can make the following determination of the transition conditions for the mixing process:

C0.1 = RESET
C1.2 = START
C2.3 = T1
C4.1 = FS1

- **Define the Output Functions**

The output functions can be defined from either the state chart (probably the easiest) or the timing diagram. Simply list each of the outputs. The output function for any output is the logical OR of all the steps for which it is energized. For the mixing process the output functions are:

SOLA= STEP2
SOLB= STEP2
SOLC= STEP4
M1 = STEP3

Since all of the outputs in this example are only active for a single step we have no OR operators in the output functions. Just to show you what that would look like suppose SOLA needed to be on for both STEP 2 and STEP 3. In that case the first output function would become SOLA = STEP 2 OR STEP 3.

- **Define the Timer Functions**

We must determine when the timer coil is to be energized, i.e. which step(s) we are timing.

T1 = STEP 3.

Construct the Controller Ladder Diagram - We are now ready to actually begin to construct the ladder diagram. Our ladder diagram will have two distinct parts, the controller ladder diagram, and the output ladder diagram. We first work on the controller ladder diagram section. The rungs in our controller ladder diagram will all be very similar.

Each one will be shown in figure 2.4:

```
|
|    CRk        CRi      ........                   CRj
j---|/|-------| |----. Ci.j. ---------------------(RLY)|
|         |   CRj     ........|
|         +---| |------------+
```

Figure 2.4. Of Controller Ladder Diagram

Here i is the previous rung in the diagram, j is this rung, and k is the next rung. The Ci.j is just the transition condition defined above. One last thing before we construct the controller portion of our ladder diagram. By looking at the sequential function chart we can see that there are really two ways to get to STEP 1, either from STEP 0 by pressing the RESET or simply recycling back after completing STEP 4. The OR function can be implemented by placing contacts in parallel. The circuit will be completed (true) if one contact OR the other (or both) are true.

(The AND function can be implemented by placing contacts in series).

The controller portion of the ladder diagram is finally shown in figure 2.5

```
|    CR2        RESET                                      CR1
1---|/|-------| |----------------------------------(RLY) |
|             |   CR4          FS1      |
|             |---| |--------| |---     |
|             |   CR1                   |
|             +---| |-------------------+
|
|
|    CR3          CR1         START                          CR2
2---|/|-------|  |--------|  |----------------------(RLY) |
|             |   CR2                   |
|             +---|  |------------------+
|
|
|    CR4          CR2          FS2                           CR3
3---|/|-------|  |--------|  |----------------------(RLY) |
|             |   CR3                   |
|             +---|  |------------------+
|
|
|    CR1          CR3          T1                            CR4
4---|/|-------|  |--------|  |----------------------(RLY) |
|             |   CR4                   |
|             +---|  |------------------+
```

Figure 2.5. The controller portion of the ladder diagram

2.1.2. Controller Ladder Diagram-LogixPro

The output section of our ladder diagram is very simple. We construct it by inspecting the output definitions and timer definitions above. Note that ladder rung(s) relating to the timer would normally be considered part of the controller section of the ladder diagram but we place it here in the output section because it looks like the other output rungs. (See figure 2.6)

```
|      CR2                                        SOLA
5---|   |-----------------------------------------(OUT)|
|
|
|
|      CR2                                        SOLB
6---|   |-----------------------------------------(OUT)|
|
|
|
|      CR4                                        SOLC
7---|   |-----------------------------------------(OUT)|
|
|
|
|      CR3                                          M1
8---|   |-----------------------------------------(OUT)|
|
|
|
|      CR3                                          T1
9---|   |-----------------------------------------(TIM)|
|
```

Figure 2.6. Controller Ladder Diagram-LogixProzx

2.2. Counters

A counter is a simple device intended to do one simple thing - count. Using them, however, can sometimes be a challenge because every manufacturer (for whatever reason) seems to use them a different way. Rest assured that the following information will let you simply and easily program anybody's counters.

Kinds of counters up-counters (they only count up 1,2,3...). These are called CTU, (count up) CNT, C, or CTR. There are down counters (they only count down 9,8,7...). These are typically called CTD (count down) when they are a separate instruction. There are also up-down counters (they count up and/or down 1,2,3,4,3,2,3,4,5...) these are typically called UDC (up-down counter) when they are separate instructions.

Many manufacturers have only one or two types of counters but they can be used to count up, down or both. The theory is all the same regardless of what the manufacturers call them. A counter is a counter is a counter.

To further confuse the issue, most manufacturers also include a limited number of high-speed counters. These are commonly called HSC (high-speed counter), CTH (Counter-High speed).

Typically a high-speed counter is a "hardware" device. The normal counters listed above are typically "software" counters. In other words they don't physically exist in the PLCs but rather they are simulated in software. Hardware counters do exist in the PLC and they are not depending about time scan.

A good rule of thumb is simply to always use the normal (software) counters unless the pulses you are counting will arrive faster than 2X the scan time. (i.e. if the scan time is 2ms and pulses will be arriving for counting every 4ms or longer then use a software counter. If they arrive faster than every 4ms (3ms for example) then use the hardware (high-speed) counters. (2xscan time = 2x2ms= 4ms).

- To use them we must know 3 things:

Where the pulses that we want to count are coming from. Typically this is from one of the inputs. (A sensor connected to input 0000 for example).

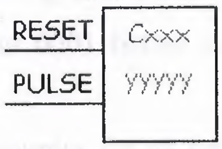
How many pulses we want to count before we react. Let's count 5 widgets before we box them, for example.

When/how we will reset the counter so it can count again. After we count 5 widgets lets reset the counter, for example.

When the program is running on the plc the program typically displays the current or "accumulated" value for us so we can see the current count value.

Typically counters can count from 0 to 9999, -32,768 to +32,767 or 0 to 65535. Why the weird numbers? Because most PLCs have 16-bit counters. We'll get into what this means in a later chapter but for now suffice it to say that 0-9999 is 16-bit BCD (binary coded decimal) and that -32,768 to 32767 and 0 to 65535 is 16-bit binary.

Here are some of the instruction symbols we will encounter (depending on which manufacturer we choose) and how to use them. Remember that while they may look different they are all used basically the same way. If we can setup one we can setup any of them.



In this counter we need 2 inputs:

- One goes before the reset line. When this input turns on the current (accumulated) count value will return to zero.
- The second input is the address where the pulses we are counting are coming from.

For example, if we are counting how many widgets pass in front of the sensor that is physically connected to input 0001 then we would put normally open contacts with the address 0001 in front of the pulse line.

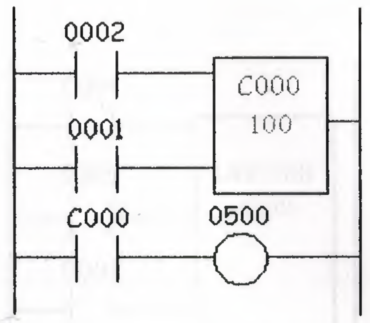
Cxxx is the name of the counter.

If we want to call it counter 000 then we would put "C000" here.

yyyyy is the number of pulses we want to count before doing something.

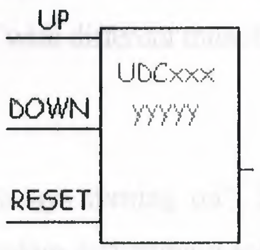
If we want to count 5 widgets before turning on a physical output to box them we would put 5 here. If we wanted to count 100 widgets then we would put 100 here, etc. When the counter is finished (i.e. we counted yyyyy widgets) it will turn on a separate set of contacts that we also label Cxxx.

Note that the counter accumulated value ONLY changes at the off to on transition of the pulse input.



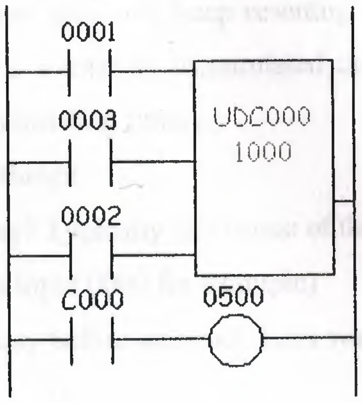
Here's the symbol on a ladder showing how we set up a counter (we'll name it counter 000) to count 100 widgets from input 0001 before turning on output 500. Sensor 0002 resets the counter.

Below is one symbol we may encounter for an up-down counter. We'll use the same abbreviation as we did for the example above. (i.e. UDCxxx and yyyyy)



In this up-down counter we need to assign 3 inputs. The reset input has the same function as above. However, instead of having only one input for the pulse counting we now have 2. One is for counting up and the other is for counting down. In this example we will call the counter UDC000 and we will give it a preset value of 1000. (We'll count 1000 total pulses) For inputs we'll use a sensor that will turn on input 0001 when it sees a target and another sensor at input 0003 will also turn on when it sees a target. When input 0001 turns on we count up and when input 0003 turns on we count down. When we reach 1000 pulses we will turn on output 500.

The ladder diagram is shown below.



2.3. Timers

A timer it is an instruction that waits a set amount of time before doing something. Sounds simple doesn't it.

When we look at the different kinds of timers available the fun begins. As always, different types of timers are available with different manufacturers. Here are most of them:

2.3.1. On-Delay timer

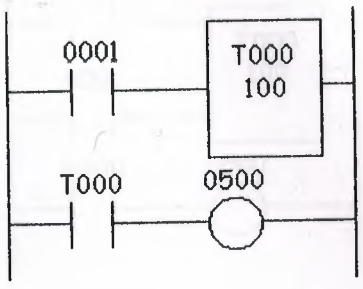
This type of timer simply "delays turning on". In other words, after our sensor (input) turns on we wait x-seconds before activating a solenoid valve (output). This is the most common timer. It is often called TON (timer on-delay), TIM (timer) or TMR (timer).

2.3.2. Off-Delay timer

This type of timer is the opposite of the on-delay timer listed above. This timer simply "delays turning off". After our sensor (input) sees a target we turn on a solenoid (output). When the sensor no longer sees the target we hold the solenoid on for x-seconds before turning it off. It is called a TOF (timer off-delay) and is less common than the on-delay type listed above.

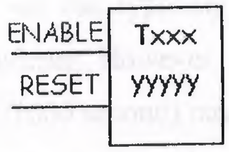
Retentive or Accumulating timer- this type of timer needs 2 inputs. One input starts the timing event (i.e. the clock starts ticking) and the other resets it. The on/off delay timers above would be reset if the input sensor wasn't on/off for the complete timer duration. This timer however holds or retains the current elapsed time when the sensor turns off in mid-

Below is the symbol shown on a ladder diagram:



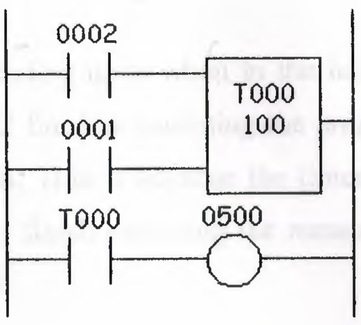
In this diagram we wait for input 0001 to turn on. When it does, timer T000 (a 100ms increment timer) starts ticking. It will tick 100 times. Each tick (increment) is 100ms so the timer will be a 10000ms (i.e. 10 second) timer. 100ticks X 100ms = 10,000ms. When 10 seconds have elapsed, the T000 contacts close and 500 turns on. When input 0001 turns off (false) the timer T000 will reset back to 0 causing its contacts to turn off (become false) thereby making output 500 turn back off.

An accumulating timer would look similar to this:



This timer is named Txxx. When the enable input is on the timer starts to tick. When it ticks yyyyy (the preset value) times, it will turn on its contacts that we will use later in the program. Remember that the duration of a tick (increment) varies with the vendor and the time base used. (i.e. a tick might be 1ms or 1 second or...) If however, the enable input turns off before the timer has completed, the current value will be retained. When the input turns back on, the timer will continue from where it left off. The only way to force the timer back to its preset value to start again is to turn on the reset input.

The symbol is shown in the ladder diagram below.



In this diagram we wait for input 0002 to turn on. When it does timer T000 (a 10ms increment timer) starts ticking. It will tick 100 times. Each tick (increment) is 10ms so the timer will be a 1000ms (i.e. 1 second) timer. 100ticks X 10ms = 1,000ms. When 1 second has elapsed, the T000 contacts close and 500 turns on. If input 0002 turns back off the current elapsed time will be retained. When 0002 turns back on the timer will continue where it left off. When input 0001 turns on (true) the timer T000 will reset back to 0 causing its contacts to turn off (become false) thereby making output 500 turn back off.

2.3.3. Timer Accuracy

When we are creating a timer, we can typically not be very concerned about their precision because it's usually insignificant. However, when we're creating timers that have duration in the millisecond (1ms= 1/1000 second) range we must be concerned about their precision. There are general two types of errors when using a timer. The first is called an input error. The other is called an output error. The total error is the sum of both the input and output errors.

2.3.4. Input error

An error occurs depending upon when the timer input turns on during the scan cycle. When the input turns on immediately after the PLC looks at the status of the inputs during the scan cycle, the input error will be at its largest. (i.e. more than 1 full scan time!).

This is because, as you will recall, the inputs are looked at once during a scan. If it wasn't on when the PLC looked and turns on later in the scan we obviously have an error. Further we have to wait until the timer instruction is executed during the program execution part of the scan. If the timer instruction is the last instruction on the rung it could be quite a big error!

2.3.5. Output error

An another error occurs depending upon when in the ladder the timer actually "times out" (expires) and when the PLC finishes executing the program to get to the part of the scan when it updates the outputs. This is because the timer finishes during the program execution but the PLC must first finish executing the remainder of the program before it can turn on the appropriate output.

Below figure 2.7. is a diagram illustrating the worst possible input error. You will note from it that the worst possible input error would be 1 complete scan time + 1 program execution time. Remember that a program execution time varies from program to program. (Depends how many instructions are in the program)

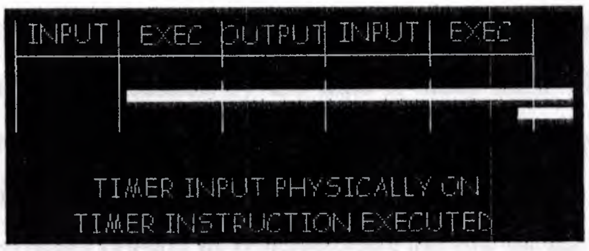


Figure 2.7. Illustrating the worst possible input error.

Figure2.8. Shown below is a diagram illustrating the worst possible output error. You can see from it that the worst possible output error would be 1 complete scan time.

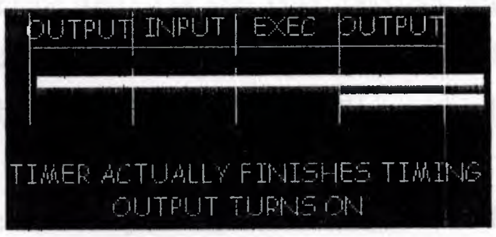


Figure 2.8. Illustrating the worst possible output error.

Based upon the above information we can now see that the total worst possible timer error = 1 scan time + 1 program execution time +1 scan time.
=2 scan times + 1 program execution time.
It means that even though most manufacturers currently have timers with 1ms increments they really shouldn't be used for durations less than a few milliseconds.

This assumes that your scan time is 1ms. If your scan time is 5ms you had better not use a timer with a duration less than about 15ms. The point is however, just so that we will know what errors we can expect. If we know what error to expect, we can then think about whether this amount of error is acceptable for our application. In most applications this error is insignificant but in some high speed or very precise applications this error can be very significant.

We should also note that the above errors are only the "software errors". There is also a hardware input error as well as a hardware output error.

The hardware input error is caused by the time it takes for the plc to actually realize that the input is on when it scans its inputs. Typically this duration is about 10ms. This is because many PLCs require that an input should be physically on for a few scans before it determines it's physically on. (To eliminate noise or "bouncing" inputs).

The hardware output error is caused by the time it takes from when the PLC tells its output to physically turn on until the moment it actually does. Typically a transistor takes about 0.5ms whereas a mechanical relay takes about 10ms.

The error keeps on growing doesn't it! If it becomes too big for the application, consider using an external "hardware" timer.

2.4. Boolean Math

Boolean math lets us do some vary basic functions with the bits in our registers. These basic functions typically include AND, OR and XOR functions. Each is described below.

AND- this function enables us to use the truth table below. Here, we can see that the AND function is very much related to multiplication. We see this because the only time the Result is true (i.e. 1) is when both operators A AND B are true (i.e. 1). The AND instruction is useful when your PLC doesn't have a masking function. Oh yeah, a masking function enables a bit in a register to be "left alone" when working on a bit level. This is simply because any bit that is ANDed with itself will remain the value it currently is. For example, if you wanted to clear (make them 0) only 12 bits in a 16-bit register you might AND the register with 0's everywhere except in the 4 bits you wanted to maintain the status of.

See the truth table 2.2 below to figure out what we mean. (1 AND 1 = 1, 0 AND 0= 0)

Result = A AND B		
A	B	Result
0	0	0
1	0	0
0	1	0
1	1	1

Table 2.2. AND- Truth table

OR- this functions based upon the truth table below (table 2.3). Here, we can see that the OR function is very much related to addition. We see this because the only time the Result is true (i.e. 1) is when operator A OR B is true (i.e. 1). Obviously, when they are both true the result is true. (If A OR B is true...)

Result = A OR B		
A	B	Result
0	0	0
1	0	1
0	1	1
1	1	1

Table 2.3. OR- Truth table

EXOR- this function enables us to use the truth table below (table 2.4). Here, we can see that the EXOR (XOR) function is not related to anything I can think of! An easy way to remember the results of this function is to think that A and B must be one or the other case, exclusively.

In other words, they must be opposites of each other. When they are both the same (i.e. A=B) the result is false (i.e. 0).

This is sometimes useful when you want to compare bits in 2 registers and highlight which bits are different. It's also needed when we calculate some checksums. A checksum is commonly used as error checking in some communications protocols.

Result = A XOR B		
A	B	Result
0	0	0
1	0	1
0	1	1
1	1	0

Table 2.4. XOR- Truth table

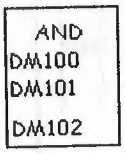
The ladder logic instructions are commonly called AND, ANDA, ANDW, OR, ORA, ORW, XOR, EORA XORW.

As we saw with the MOV instruction there does generally the majority of plc makers use two common methods. The first method includes a single instruction that asks us for a few key pieces of information. This method typically requires:

Source A- this is the address of the first piece of data we will use. In other words its the location in memory of where the A is.

Source B- this is the address of the second piece of data we will use. In other words it's the location in memory of where the B is.

Destination- this is the address where the result will be put. For example, if A AND B = 0 the result (0) would automatically be put into this destination memory location.

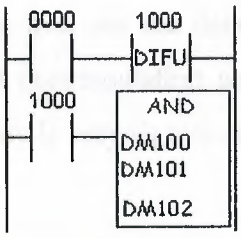


AND symbol

The instructions above typically have a symbol that looks like that shown here. Of course, the word AND would be replaced by OR or XOR. In this symbol, The source A is DM100, the source B is DM101 and the destination is DM102.

Therefore, we have simply created the equation DM100 AND DM101 = DM102. The result is automatically stored into DM102.

The Boolean functions on a ladder diagram are shown in fig

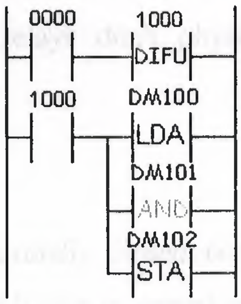


Please note that once again we are using a one-shot instruction. As we've seen before, this is because if we didn't use it, we would execute the instruction on every scan. Odds are good that we'd only want to execute the function one time when input 0000 becomes true.

AND symbol (dual instruction method)

The dual instruction method would use a symbol similar to that shown above. In this method, we give this symbol only the Source B location. The Source A location is given by the LDA instruction. The Destination would be included in the STA instruction.

Below is a ladder diagram showing what is meant:



The results are the same as the single instruction method shown above. It should be noted that although the symbol and ladder diagram above show the AND instruction, OR or EXOR can be used as well. Simply substitute the word "AND" within the instruction to be either "OR" or "EXOR". The results will be the same as shown in their respective truth tables.

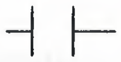
We should always remember that the theory is most important. If we can understand the theory of why things happen as they do, we can use anybody's PLC. If we refer to the manufacturers documentation we can find out the details for the particular PLC we are using. Try to find the theory in that documentation and you might come up short. The details are insignificant while the theory is very significant.

2.5. Basic Instructions

2.5.1. Load

The load (LD) instruction is a normally open contact. It is sometimes also called examine if on. (XIO) (As in examine the input to see if it's physically on).

The symbol for a load instruction is shown below.

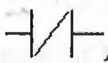

A Load (contact) symbol

This is used when an input signal is needed to be present for the symbol to turn on. When the physical input is on we can say that the instruction is True. We examine the input for an on signal. If the input is physically on then the symbol is on. An on condition is also referred to as a logic 1 state.

This symbol normally can be used for internal inputs, external inputs and external output contacts. Remember that internal relays don't physically exist. They are simulated (software) relays.

2.5.2. LoadBar

The LoaDBar instruction is a normally closed contact. It is sometimes also called LoaDNot or examine if closed. (XIC) (As in examine the input to see if its physically closed) The symbol for a LoaDBar instruction is shown below.

 A LoaDNot (normally closed contact) symbol

This is used when an input signal does not need to be present for the symbol to turn on. When the physical input is off we can say that the instruction is True. We examine the input for an off signal. If the input is physically off then the symbol is on. An off condition is also referred to as a logic 0 state.

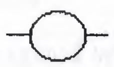
This symbol normally can be used for internal inputs, external inputs and sometimes, external output contacts. Remember again that internal relays don't physically exist. They are simulated (software) relays. It is the exact opposite of the Load instruction.
(See table2.5)

Logic State	Load	LoadBar
0	False	True
1	True	False

Table 2.5. NOT- gate

2.5.3.1. Out

The Out instruction is sometimes also called an Output Energize instruction. The output instruction is like a relay coil. Its symbol looks as shown below.



An OUT (coil) symbol

When there is a path of True instructions preceding this on the ladder rung, it will also be True. When the instruction is True it is physically On. We can think of this instruction as a normally open output. This instruction can be used for internal coils and external outputs.

2.5.3.2. Outbar

The Outbar instruction is sometimes also called an OutNot instruction. Some vendors don't have this instruction. The outbar instruction is like a normally closed relay coil. Its symbol looks like that shown below.

An OUTbar (normally closed coil) symbol

When there is a path of False instructions preceding this on the ladder rung, it will be True. When the instruction is True it is physically On.

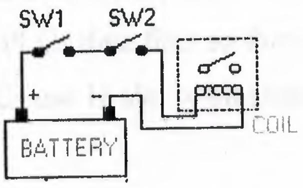
We can think of this instruction as a normally closed output. This instruction can be used for internal coils and external outputs.

It is the exact opposite of the Out instruction. (See table2.6)

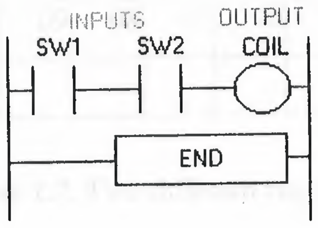
Logic State	Out	OutBar
0	False	True
1	True	False

Table 2.6. Out and Outbar table

Let's compare a simple ladder diagram with its real world external physically connected relay circuit and see the differences.



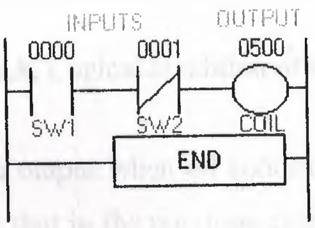
In the above circuit, the coil will be energized when there is a closed loop between the + and - terminals of the battery. We can simulate this same circuit with a ladder diagram. A ladder diagram consists of individual rungs just like on a real ladder. Each rung must contain one or more inputs and one or more outputs. The first instruction on a rung must always be an input instruction and the last instruction on a rung should always be an output (or its equivalent).



In this simple one rung ladder diagram we have recreated the external circuit above with a ladder diagram. Here we used the Load and Out instructions. Some manufacturers require that every ladder diagram include an END instruction on the last rung. Some PLCs also require an ENDH instruction on the rung after the END rung.

2.6. PLC Registers

Now take the previous example and change switch 2 (SW2) to a normally closed symbol (LoadBar instruction). SW1 will be physically OFF and SW2 will be physically ON initially. The ladder diagram now looks like this:



Notice also that we now gave each symbol (or instruction) an address. This address sets aside a certain storage area in the PLCs data files so that the status of the instruction (i.e. true/false) can be stored. Many PLCs use 16 slot or bit storage locations. (See table2.7)

In the example above we are using two different storage locations or registers.

REGISTER 00															
15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
														1	0
REGISTER 05															
15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
															0

Table 2.7. Two different registers.

In the tables above we can see that in register 00, bit 00 (i.e. input 0000) was a logic 0 and bit 01 (i.e. input 0001) was a logic 1. Register 05 shows that bit 00 (i.e. output 0500) was logic 0. (See table2.8)

The logic 0 or 1 indicates whether an instruction is False or True.

LOGICAL CONDITION OF SYMBOL			
LOGIC BITS	LD	LDB	OUT
Logic 0	False	True	False
Logic 1	True	False	True

Table 2.8. Logical condition of symbols

The PLC will only energize an output when all conditions on the rung are TRUE. So, looking at the table above, we see that in the previous example SW1 has to be logic 1 and SW2 must be logic 0. Then and ONLY then will the coil be true (i.e. energized). If any of the instructions on the rung before the output (coil) are false then the output (coil) will be false (not energized).

Let's now look at a truth table 2.9 of our previous program to further illustrate this important point. Our truth table will show ALL possible combinations of the status of the two inputs.

Inputs		Outputs	Register Logic Bits		
SW1(LD)	SW2(LDB)	COIL(OUT)	SW1(LD)	SW2(LDB)	COIL(OUT)
False	True	False	0	0	0
False	False	False	0	1	0
True	True	True	1	0	1
True	False	False	1	1	0

Table 2.9. Possible combinations of the status of the two inputs.

Notice from the chart that as the inputs change their states over time, so will the outputs. The output is only true (energized) when all preceding instructions on the rung are true.

2.6.1. A Level Application

We've seen how registers work; let's process a program like PLCs do to enhance our understanding of how the program gets scanned.

Let's consider the following application:

We are controlling lubricating oil being dispensed from a tank. This is possible by using two sensors.

We put one near the bottom and one near the top, as shown in the figure 2.9.

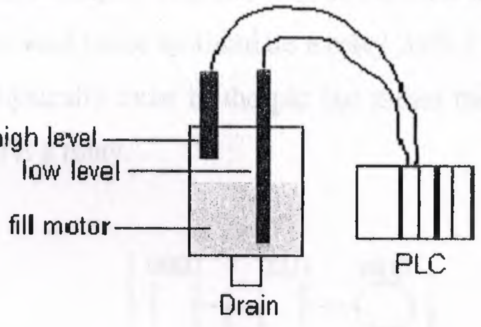


Figure 2.9. Dispensing oil from a tank

Here, we want the fill motor to pump lubricating oil into the tank until the high level sensor turns on. At that point we want to turn off the motor until the level falls below the low level sensor. Then we should turn on the fill motor and repeat the process.

Here we have a need for 3 I/O (i.e. Inputs/Outputs). 2 are inputs (the sensors) and 1 is an output (the fill motor). Both of our inputs will be NC (normally closed) fiber-optic level sensors. When they are NOT immersed in liquid they will be ON. When they are immersed in liquid they will be OFF.

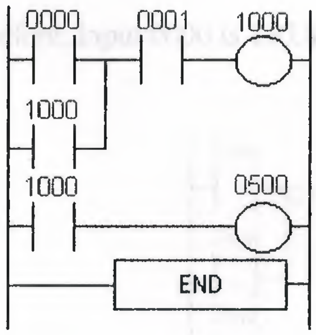
We will give each input and output device an address. This lets the PLC know where they are physically connected.

The addresses are shown in the table2.10.

Inputs	Address	Output	Address	Internal Utility Relay
Low	0000	Motor	0500	1000
High	0001			

Table 2.10. Addresses of inputs an outputs

Below is what the ladder diagram will actually look like. Notice that we are using an internal utility relay in this example. You can use the contacts of these relays as many times as required. Here they are used twice to simulate a relay with 2 sets of contacts. Remember, these relays DO NOT physically exist in the plc but rather they are bits in a register that you can use to SIMULATE a relay.

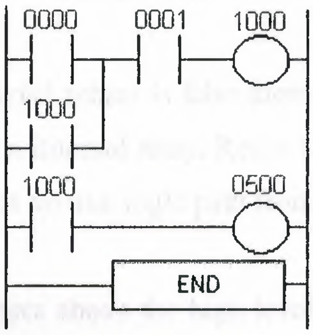


We should always remember that the most common reason for using PLCs in our applications is for replacing real-world relays. The internal utility relays make this action possible. It's impossible to indicate how many internal relays are included with each brand of PLC. Some include 100's while other include 1000's while still others include 10's of 1000's! Typically, PLC size (not physical size but rather I/O size) is the deciding factor. If we are using a micro-PLC with a few I/O we don't need many internal relays. If however, we are using a large PLC with 100's or 1000's of I/O we'll certainly need many more internal relays.

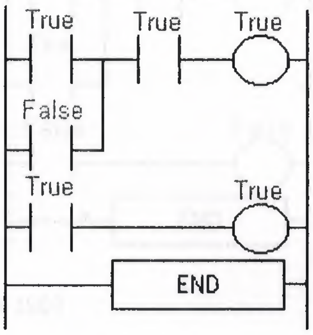
If ever there is a question as to whether or not the manufacturer supplies enough internal relays, consult their specification sheets. In all but the largest of large applications, the supplied amount should be MORE than enough.

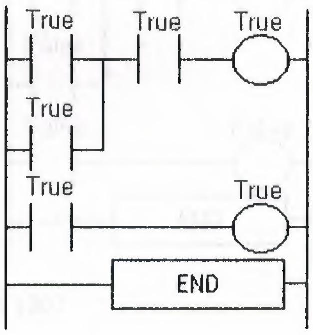
2.6.2. The Program Scan

Let's watch what happens in this program scan by scan.



Initially the tank is empty. Therefore, input 0000 is TRUE and input 0001 is also TRUE.

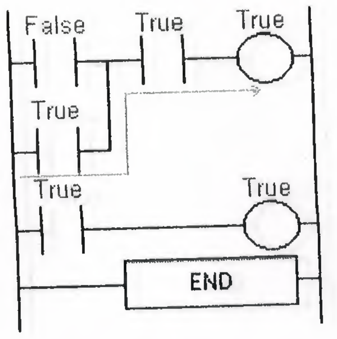




Scan 1 Scan 2-100

Gradually the tank fills because 500(fill motor) are on.

After 100 scans the oil level rises above the low level sensor and it becomes open. (i.e. FALSE)

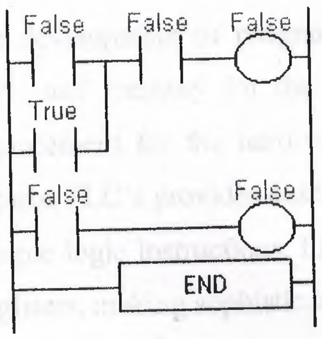


Scan 101-1000

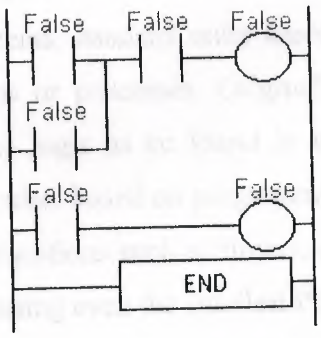
Notice that even when the low level sensor is false there is still a path of true logic from left to right. This is why we used an internal relay. Relay 1000 is latching the output (500) on. It will stay this way until there is no true logic path from left to right.

(i.e. when 0001 becomes false)

After 1000 scans the oil level rises above the high level sensor at it also becomes open (i.e. false)



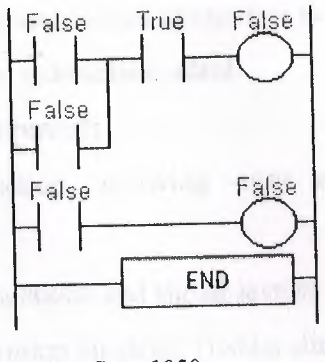
Scan 1001



Scan 1002

Since there is no more true logic path, output 500 is no longer energized (true) and therefore the motor turns off.

After 1050 scans the oil level falls below the high level sensor and it will become true again.



Scan 1050

After 2000 scans the oil level falls below the low level sensor and it will also become true again. At this point the logic will appear the same as SCAN 1 above and the logic will repeat as illustrated above.

2.7. PROGRAMMABLE CONTROLLER PLC's

2.7.1. Introduction

The need for low cost, versatile and easily commissioned controllers has resulted in the development of programmable-control systems standard units based on a hardware CPU and memory for the control of machines or processes. Originally designed as a replacement for the hard-wired relay and timer logic to be found in traditional control panels, PLC's provides ease and flexibility of control based on programming and executing simple logic instructions. PLC's have internal functions such as timers, counters and shift registers, making sophisticated control possible using even the smallest PLC.

A programmable control operates by examining the input signals from a process and carrying out logic instructions on these input signals, producing output signal to drive process equipment or machinery. Standard interfaces build into PLC's allow them to be directly connected to process actuators and transducers (pumps and valves) without the need for intermediate circuitry or relays.

Through using PLC's it became possible to modify a control system without having the disconnect or re-route a signal wire. It was necessary to change only the control program using a keypad or VDU terminal. Programmable controllers also require shorter installation and commissioning times than do hardwired systems.

Although PLC's are similar to conventional computers in terms of hardware technology, they have specific features suited to industrial control:

- Rugged, noise immune equipment;
- Modular plug-in construction, allowing easy replacement\addition of units (input\output);
- Standard input\output connections and signal levels;
- Easily understood programming language (ladder diagram and function chart),

ease of programming and reprogramming in-plant.

These features make programmable controllers highly desirable in a wide variety of industrial-plant and process-control situations.

2.7.2. Background

The programmable controller was initially conceived by a group of engineers from General Motors in 1968, where an initial specification was provided: the controller must be:

Easily programmed and reprogrammed, preferably in-plant to alter its sequence of operations.

Easily maintained and repaired- preferably using plug-in modules.

(a)-More reliable in plant environment.

(b)-Smaller than it is relay equivalent.

Cost competitive, with solid-state and relay panels than in use.

This provoked a keen interest from engineers of all disciplines in how to PLC could be used for industrial control. With this came demands for additional PLC capabilities and facilities, which were rapidly implemented as the technology became available.

The instruction sets quickly moved from simple logic instructions to include counters, timers and shift registers, than onto more advanced mathematical functions on the machines. Developments hardware were also occurring, with larger memory and greater numbers of input / output points featuring on new models. In 1976 became possible to control remote I / O racks, where large numbers of distant I / O points were monitored updated via a communications link, often several hundred meters from the main PLC. The Allan-Bradley Corporation in America introduced a microprocessor-based PLC in 1977.

It was based on an 8080 microprocessor but used an extra processor to handle bit logic instruction at high speed.

The increased rate of application of programmable controllers within industry has encouraged manufacturers to develop whole families of microprocessor-based systems having various levels of performance.

The range of available PLC's now extends from small self-contained units with 20 digital I / O points and 500 program steps, up to modular systems with add-on function modules:

-Analogue I/O;

-PID control (proportional, integral and derivative terms);

-Communications;

-Graphics display;

-Additional I/O;

-Additional memory.

This modular approach allows the expansion or upgrading of a control system with minimum cost and disturbance.

Programmable controllers are developing at a virtually the same pace as microcomputers, with particular emphasis on small controllers, positioning\numeric control and communication networks. The market for small controllers has grown rapidly since the early 1980's when a number of Japanese companies introduced very small, low cost units that were much cheaper than others available at that time.

This brought programmable controllers within the budget of many potential users in the manufacturing and process industries, and this trend continues with PLC's offering ever-increasing performance at ever-decreasing cost.

The Mitsubishi F40 PLC is a typical example of a modern small PLC, providing 40 I/O points, 16 timers and counters, plus other functions. The controller uses a microprocessor and has 890 RAM locations fur user programs.

The 24-input channels of the F40 operate at 24 V d.c. Whilst 16 outputs may be 24 V d.c. Or 240 V a.c. to provide easy interfacing to industrial equipment.

2.7.3. Terminology-PC or PLC

There are several different terms used to describe programmable controllers, most referring to the functional operation of the machine in question:

PC programmable controller

PLC programmable logic controller

PBS programmable binary system

By their nature these terms tend to describe controllers that normally work in a binary environment. Since all but the smallest programmable controllers can now be equipped to process analogue inputs and outputs these labels are not representative of their capabilities. For these reason the overall term programmable controller has been widely adopted to describe the family of freely programmable controllers. However, to avoid confusion with the personal computer PC, this text uses the abbreviation PLC for programmable (logic) controller.

2.7.4. PLC Hardware Design

Programmable controllers are purpose-built computers consisting of three functional areas:

-Processing

-Memory

-Input / output

Input conditions to the PLC are sensed and than stored in the memory, where the PLC performs the programmed logic instructions on these input states. Output conditions are then generated to drive associated equipment. The action taken depends totally on the control program held in memory.

In smaller PLC these functions are performed by individual printed circuit cards within a single compact unit, whilst larger PLC's are constructed on a modular basis with function modules slotted in to the backplane connectors of the mounting rack.

This allows simple expansion of the system when necessary. In both these cases the individual circuit board are easily removed and replaced, facilitating rapid repair of the system should faults develop.

In addition a programming unit is necessary to download control programs to the PLC memory.

2.7.5. Input output / units

Most PLC'S operate internally at between 5 and 15 V d.c. (Common TTL and CMOS voltages), whilst process signals much greater, typically 24 V d.c. to 240 V a.c. at several amperes.

The I / O units form the interface between the microelectronics of the programmable controller and real world outside, and must therefore provide all, necessary signal conditioning and isolation functions. This often allows a PLC to be directly connected to process actuators and transducers (pumps and valves) without the need for intermediate circuitry and relays.

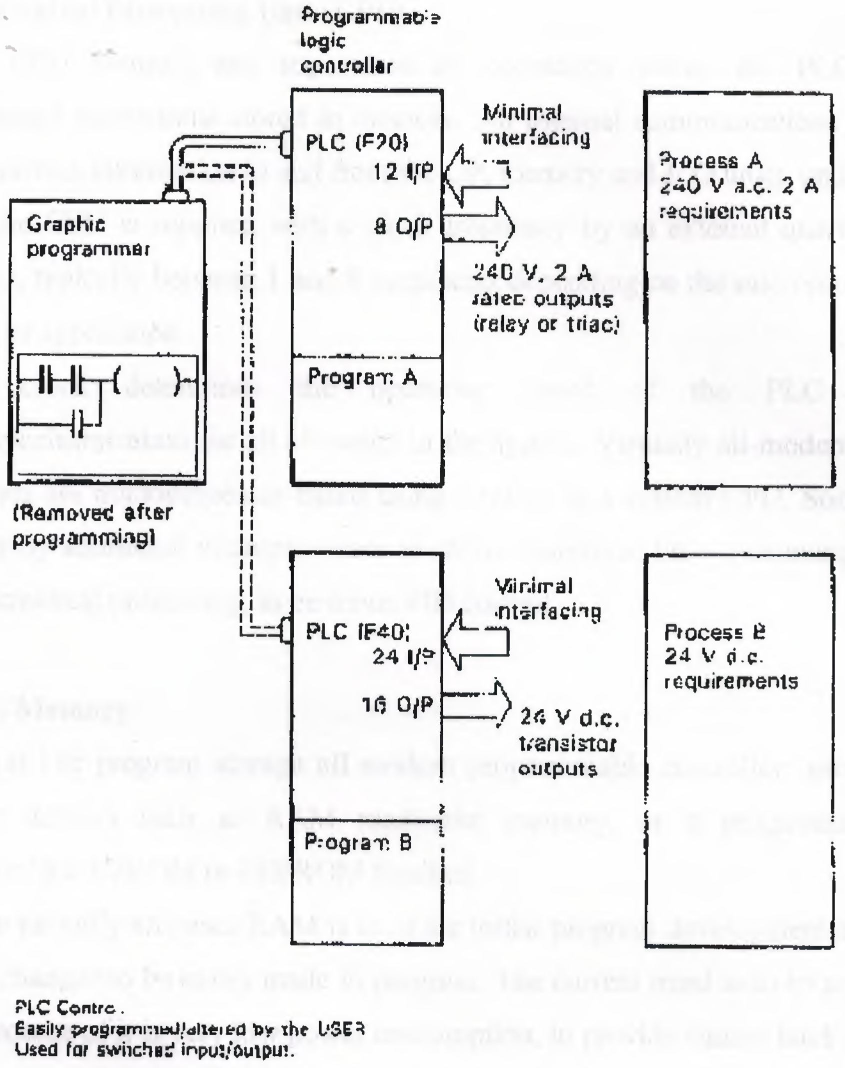
To provide this signal conversion programmable controllers are available with a choice of input / output units to suit different requirements.

For example;

Inputs	5 V (TTL level) switched I/ P
	24 V switched I/ P
	110 V switched I/ P
	240 v switched I/ P
Outputs	24 V 100 mA switched O/ P
	110 V 1mA
	240 V 1 A a.c. (triac)
	240 V 2 A a.c. (relay)

It is standard practice for all I/O channels to electrically isolated from the controlled process, using opto-isolator circuits on the I/O modules. An opto-isolator circuit consists of a light emitting diode and a phototransistor, forming an opto coupled pair that allows small signals to pass through, but will clamp any high voltage spikes or surges down to the same small level. This provides protection against switching transients and power-supply surges, normally up to 1500V.

In small self contained PLC's in which all I/ O points are physically located on the one casing, all inputs will be of one type (e.g. 24 V) and the same for outputs (e.g. all 240 V triac). This is because manufacturers supply on the standard function boards for economic reasons. Modular PLC's have greater flexibility of I/ O, however, since the user can select from several different types and combinations of input and output modules.
In all cases the input/output units are designed with the aim of simplifying the connections of process transducers and actuators to the programmable controller. For these purpose all PLC'S are equipped with standard screw terminals or plugs on every I\O point, allowing the rapid and simple removal and replacement of a faulty I/ O card. Every input\output point has a unique address or channel number, which is using during program development to specify to monitoring of an input or the activating of a particular output within the program. Indication of the status of input\output channels is provided by light-emitting diode (LED's) on the PLC or I/ O unit, making it simple to check the operation of process inputs and outputs from the PLC itself. (See figure2.10).


Programmable logic controller
Minimal interfacing
PLC (F20)
12 I/P
8 O/P
Process A
240 V a.c. 2 A requirements
Graphic programmer
240 V, 2 A rated outputs (relay or triac)
Program A
(Removed after programming)
Minimal interfacing
PLC (F40)
24 I/P
16 O/P
Process B
24 V d.c. requirements
24 V d.c. transistor outputs
Program B
PLC Control.
Easily programmed/altered by the USER
Used for switched input/output.

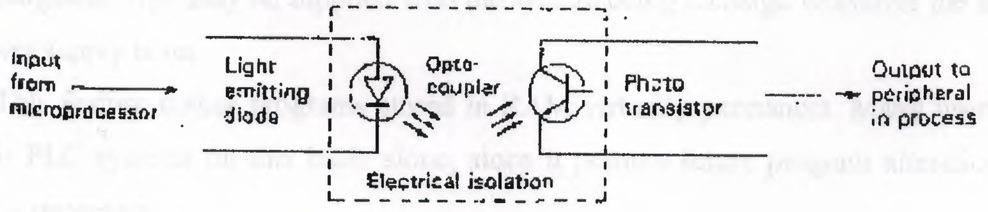


Figure 2.10. Opto-isolator circuit

2.7.6. Central Processing Unit (CPU)

The CPU controls and supervises all operations within the PLC, carrying out programmed instructions stored in memory. An internal communications highway or bus system carries information to and from the CP, memory and I/ O units, under control of the CPU. The CPU is supplied with a clock frequency by an external quartz crystal or RC oscillator, typically between 1 and 8 megahertz depending on the microprocessor used and the area of application.

The clock determines the operating speed of the PLC and provides timing\synchronization for all elements in the system. Virtually all-modern programmable controllers are microprocessor based using a micro as a system CPU. Some larger PLC's also employ additional microprocessor to control complex, time-consuming functions such as mathematical processing, three terms PID control.

2.7.6.1. Memory

(a) For program storage all modern programmable controllers use semiconductor memory devices such as RAM read\write memory, or a programmable read-only memory of the EPROM or EEPROM families.

In the virtually all cases RAM is used for initial program development and testing, as it follows changes to be easily made in program. The current trend is to be providing CMOS RAM because of it is very low power consumption, to provide battery back-up to this RAM in order to maintain the contents when the power is removed from the PLC system. This battery has a lifespan of at least one year before replacement is necessary, or alternatively a rechargeable type may be supplied with the system being recharge whenever the main PLC power supply is on.

This feature makes programs stored in RAM virtually permanent. Many users operate their PLC systems on this basis alone, since it permits future program alterations if and when necessary.

After a program is fully developed and tested it may be loaded (blown) into a PROM or EPROM memory chip, which are normally cheaper than RAM devices.

PROM programming is usually carried out with a special purpose programming unit, although many programmable controllers now have this facility built-in, allowing programs in the PLC RAM to be down loaded into a PROM IC placed in a socket provided on the PLC itself.

(b) In addition to program storage, a programmable controller may require memory for other functions:

Temporary buffer store for input\output channels status- I/ O RAM

Temporary storage for status of internal function (timers, counters, marker relays)

Since these consist of changing data they require RAM read\write memory, which may be battery-backed in sections.

2.7.6.2. Memory size

Smaller programmable controllers normally have a fixed memory size, due in part to the physical dimensions of the unit. This varies in capacity between 300 and 1000 instructions depending on the manufacturer. This capacity may not appear large enough to be very useful, but it has been estimated that 90 % of all binary control tasks can be solved using less than 1000 instructions, so there is sufficient space to meet most users needs.

Larger PLC's utilize memory modules of between 1K and 64K in size, allowing the system to be expanded by fitting addition RAM or PROM memory cards to the PLC rack.

As integrated circuit memory costs continue to fall, the PLC manufacturers are providing larger program memories on all products.

2.8. Logic instruction set

The most common technique for programming small PLC's is to draw s ladder diagram of the logic to be used, and then convert this in to mnemonic instructions, which will be keyed in to programming panel attached to the programmable controller. These instructions are similar in appearance to assembly-type codes, but refer to physical inputs, outputs and functions within the PLC itself.

The instruction set consists of logic instructions (mnemonics) that represent the actions that may be performed within a given programmable controller. Instructions sets vary between PLC's from different manufacturers, but are similar in terms of the control actions performed.

Because the PLC logic instruction set tends to be small, it can be quickly mastered and used by control technicians and engineers.

Each program instructions are made up of two parts: a mnemonic operation component or opcode, and an address or operand component that identifies particular elements (E.g. outputs) within the PLC.

For example:

Opcode	Operand
OUT	Y430
Device symbol	Identifier

Here the instruction refers to output (Y) number 430.

2.9. Input\output numbering

These instructions are used the program logic control circuits that have been designed in ladder diagram form, by assigning all physical inputs and outputs with an operand suitable to the PLC being used. The numbering system used differs between manufacturers, but certain common terms exist. For example, Texas instrument and Mitsubishi use the symbol X to represent inputs, and Y to label outputs.

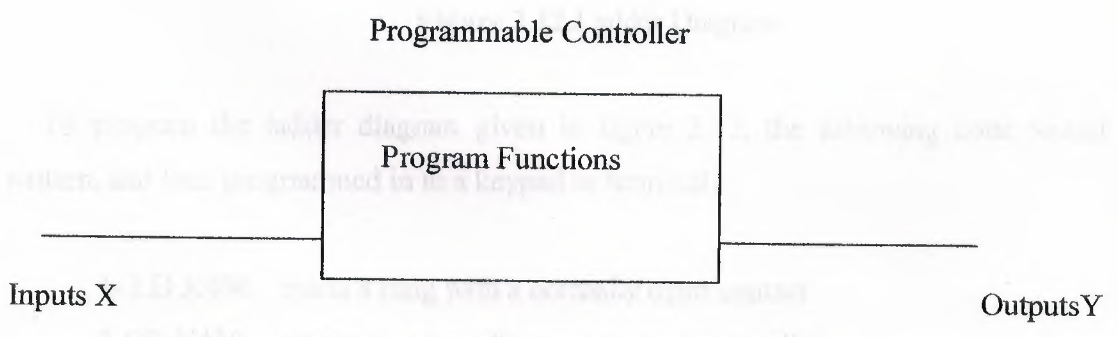


Figure 2.11. Programmable Controller

A range of addresses will be allocated to particular elements:

For example:

Mitsubishi F40 PLC:	24 inputs:	X400 – 407, 410 – 413
		X500 – 507, 510 – 513
	16 Outputs:	Y430 – 437
		Y530 – 537

Inspections of these numbers ranges will reveal that there is no overlap of addresses between functions; that is, 400 must be an input, 533 must be an output. Thus for these programmable controllers the symbol X or Y is redundant, being used purely for the benefit of the user, who is unlikely to remember what element 533 represents. However, for many PLC's both parts of the address are essential, since the I\O number ranges are identical. For example the Klockner-Moeller range of controllers (see figure 2.12)

Sucos PS 21 PLC: 8 inputs I0 to 7, etc. 8 Outputs Q0 to 7, etc

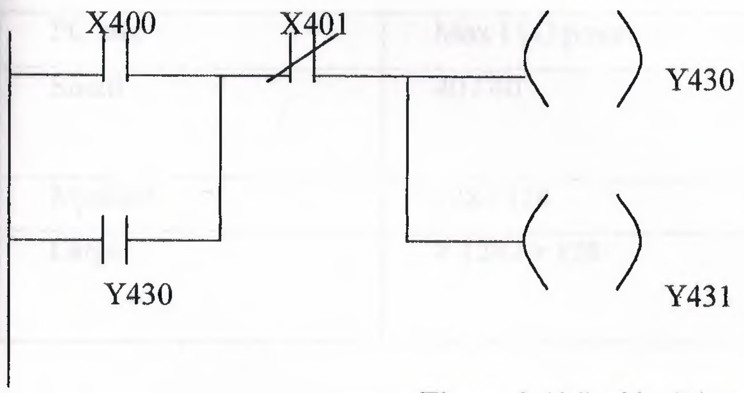


Figure 2.12. Ladder Diagram

To program the ladder diagram given in figure 2.12, the following code would be written, and then programmed in to a keypad or terminal.

1- LD X400	starts a rung with a normally open contact
2-OR Y430	connect a normally open contact in parallel
3-ANI X401	connect a normally closed contact in series
4- OUT Y430	drive an output channel
5- OUT Y431	drive another channel
6- END	end of program-return to start

Notice the contact Y430 that forms a latch across X400. The Y contact is not a physical element, but is simulated within the programmable controller and will operate in unison with the output point Y430. The programmer may create as many contacts associated with an output as necessary.

2.10. TYPES OF PLC

The increasing demand from industry for programmable controllers that can be applied to different forms and sizes of control tasks has resulted in most manufacturers producing a range of PLC's with various levels of performance and facilities.

Typical rough definitions of PLC size are given in terms of program memory size and the maximum number of input\output points the system can support. Table 2.11 gives an example of these categories.

PC size	Max I \ O points	Use memory size
Small	40 / 40	1K
Medium	128 / 128	4K
Large	> 128 / > 128	> 4K

Table 2.11. Categories of PLC

However, to evaluate properly any programmable controller we must consider many additional features such as its processor, cycle time language facilities, functions, and expansion capabilities.

A brief outline of the characteristics of small, medium of large programmable controller is given below, together with typical applications.

2.10.1. Small PLCs

In general, small and 'mini' PLC's are designed as robust, compact units, which can be mounted on or beside the equipment to be controlled. They are mainly used the replaced hard-wired logic relays, timers, counters. That control individual items of plant or machinery, but can also be used to coordinate several machines working in conjunction with each other.

Small programmable controllers can normally have their total I/ O expanded by adding one or two I/ O modules, but if any further developments are required this will often mean replacement of the complete unit. This end of the market is very much concerned with non-specialist and users; therefore ease of programming and a 'familiar' circuit format are desirable. Competition between manufacturers is extremely fierce in this field, as they vie to obtain a maximum share in this partially developed sector of the market.

A single processor is normally used, and programming facilities are kept a fairly basic level, including conventional sequencing controls and simple standard functions: e.g. timers and counters. Programming of small PLC's is by way of logic instruction list (mnemonics) or relay ladder diagrams.

Program storage is given by EPROM or battery-backed RAM. There is now a trend towards EEPROM memory with on-board programming facilities on several controllers. (See table 2.11).

Electrical:	240 V a.c. supply;
	24 V d.c. On-board for in requirements;
	12 input, 8 output points;
	LED indicators on all I/ O points;
	All I\O Opto-isolated

Choice of output:	Relay (240 V 2 a rated) Triac (240 V 1 A rated) Transistor (24 V d.c. 1 A) 320- step memory (CMOS battery-backed RAM)
Programming:	Ladder logic or instruction set using hand-held or graphic LCD programmer, with editor, test and monitor facilities;
Facilities:	8 counters, range 1-99 (can be cascaded) 8 timers, range 0.1-99s (can be cascaded) 64 markers\auxiliary relays; can be used individually or in blocks forming shift registers; Special function relays; Jump capability;

Table 2.11. Features of a typical small PLC – Mitsubishi F20

2.10.2. Medium-sized PLC'S

In this range modular construction predominates with plug-in modules based around the Euro card 19-inch rack format or another rack mounting system. This construction allows the simple upgrading or expansion of the system. This construction allows the simple upgrading or expansion of the system by fitting additional I/ O cards in to the cards into the rack, since most rack, systems have space for several extra function cards. Boards are usually 'rugged zed' to allow reliable operation over a range of environments.

In general this type of PLC is applied to logic control tasks that can not be met by small controllers due to insufficient I\O provision, or because the control task is likely to be extended in the future. This might require the replacement of a small PLC, where as a modular system can be expanded to a much greater extent, allowing for growth. A medium-sized PLC may therefore be financially more attractive in the long term.

Communications of a single and multi-bit processor are likely within the CPU. For programming, standard instructions or ladder and logic diagrams are available. Programming is normally carried out via a small keypad or a VDU terminal. If different sizes of PLC are purchased from a single manufacturer, it is likely that programs and programming panels will be compatible between the machines.

2.10.3. Large PLC

Where control of very large numbers of input and output points is necessary and complex control functions are required, a large programmable controller is the obvious choice. Large PLC's are designed for use in large plants or on large machines requiring continuous control. They are also employed as supervisory controllers to monitor and control several other PLC's or intelligent machines. e.g. CNC tools.

Modular construction in Euro card format is standard, with a wide range of function cards available including analogue input output modules.

There is a move towards 16-bit processor and also multi-processor usage in order to efficiently handle a large range of differing control tasks.

For example;

-16-bit processor as main processor for digital arithmetic and text handling.

-Single-bit processor as co-or parallel processor for fast counting, storage etc.

-Peripheral processor for handling additional tasks, which are time-dependent or time-critical, such as:

- Closed-loop (PID) control
- Position controls
-Floating-point numerical calculations
-Diagnostic and monitoring
-Communications for decentralized
-Remote input\output racks.

This multi-processor solution optimizes the performance of the overall system as regards versatility and processing speed, allowing to PLC to handle very large programs of 100 K instructions or more. Memory cards can now provide several megabytes of CMOS RAM or EPROM storage.

2.10.4. Remote input\output

When large numbers of input / output points are located a considerable distance away from the programmable controller, it is uneconomic to run connecting cables to every point. A solution to this problem is to site a remote I/ O unit near to the desired I/ O points. This acts as a concentrator to monitor all inputs and transmit their status over a single serial communications link to the programmable controller. Once output signals have been produced by the PLC they are feedback along the communications cable to the remote I/ O unit, which converts the serial data into the individual output signals to drive the process.

2.10.5. Programming large PLC's

Virtually any function can be programmed, using the familiar ladder symbols via a graphics terminal or personal computer. Parameters are passed to relevant modules either by incorporating constants in to the ladder, or via on screen menus for that module.

There may in addition be computer-oriented languages, which allow programming of function modules and subroutines.

There is progress towards standardization of programming languages; with programs becoming easier to over-view through improvement of text handling, hand improved documentation facilities. This is assisted by the application of personal computers as workstations.

2.11. Developments

Present trends include the integration of process data from a PLC into management databases, etc. This allows immediate presentation of information to those involved in scheduling, production and planning.

The need to pass process information between PC's and PLC sand other devices within a automated plants has resulted in the provision of a communications capability on all but the smallest controller. The development of local area networks (LAN) and in particular the recent MAP specification by General Motors (manufacturing automation protocol) provides the communication link to integrate all levels of control systems.

2.12. DC and AC Inputs

2.12.1. DC Inputs

This will give us a better understanding of how we should wire them up. Bad things can happen if we wire them up incorrectly!

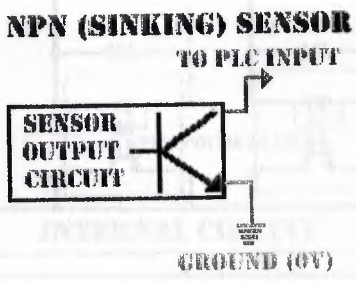
Typically, dc input modules are available that will work with 5, 12, 24, and 48 volts. Be sure to purchase the one that fits your needs based upon the input devices you will use.

We'll first look at how the dc inputs work. DC input modules allow us to connect either PNP (sourcing) or NPN (sinking) transistor type devices to them.

If we are using a regular switch (i.e. toggle or pushbutton, etc.) we typically don't have to worry about whether we wire it as NPN or PNP. We should note that most PLCs won't let us mix NPN and PNP devices on the same module. When we are using a sensor (photo-eye, prox, etc.) we do, however, have to worry about its output configuration. Always verify whether it's PNP or NPN. (Check with the manufacturer when unsure)

The difference between the two types is whether the load (in our case, the plc is the load) is switched to ground or positive voltage. An NPN type sensor has the load switched to ground whereas a PNP device has the load switched to positive voltage.

Below is what the outputs look like for NPN and PNP sensors.

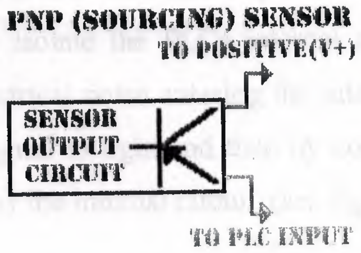


On the NPN sensor we connect one output to the PLCs input and the other output to the power supply ground.

If the sensor is not powered from the same supply as the PLC, we should connect both grounds together. NPN sensors are most commonly used in North America.

Many engineers will say that PNP is better (i.e. safer) because the load is switched to ground, but whatever works for you is best. Just remember to plan for the worst.

On the PNP sensor we connect one output to positive voltage and the other output to the PLCs input. If the sensor is not powered from the same supply as the PLC, we should connect both V+ together. PNP sensors are most commonly used in Europe.



Inside the sensor, the transistor is just acting as a switch. The sensors internal circuit tells the output transistor to turn on when a target is present. The transistor then closes the circuit between the 2 connections shown above. (V+ and PLC input).

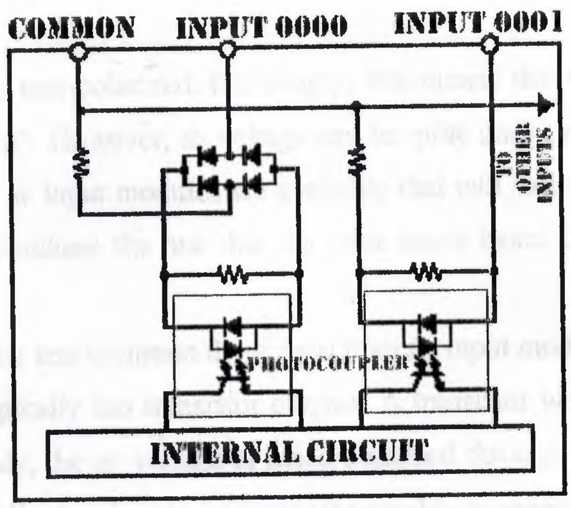


Figure 2.13. Transistor inside the sensor

The only things accessible to the user are the terminals labeled COMMON, INPUT 0000, INPUT 0001, INPUTxxxx... The common terminal either gets connected to V+ or ground. Where it's connected depends upon the type of sensor used. When using an NPN sensor this terminal is connected to V+. When using a PNP sensor this terminal is connected to 0V (ground).

A common switch (i.e. limit switch, pushbutton, toggle, etc.) would be connected to the inputs in a similar fashion. One side of the switch would be connected directly to V+. The other end goes to the PLC input terminal. This assumes the common terminal is connected to 0V (ground). If the common is connected to V+ then simply connect one end of the switch to 0V (ground) and the other end to the PLC input terminal.

The photo couplers are used to isolate the PLCs internal circuit from the inputs. This eliminates the chance of any electrical noise entering the internal circuitry. They work by converting the electrical input signal to light and then by converting the light back to an electrical signal to be processed by the internal circuit. (see figure2.13)

2.12.2. AC Inputs

An ac voltage is non-polarized. Put simply, this means that there is no positive or negative to "worry about". However, ac voltage can be quite dangerous to work with if we are careless. Typically, ac input modules are available that will work with 24, 48, 110, and 220 volts. Be sure to purchase the one that fits your needs based upon the input devices (voltage) you will use.

AC input modules are less common these days than dc input modules. The reason being that today's sensors typically has transistor outputs. A transistor will not work with an ac voltage. Most commonly, the ac voltage is being switched through a limit switch or other switch type. If your application is using a sensor it probably is operating on a dc voltage.

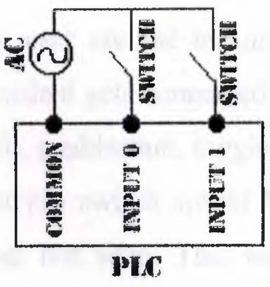


Figure 2.14. AC Input

We typically connect an ac device to our input module as shown in figure 2.14

Commonly the ac "hot" wire is connected to the switch while the "neutral" goes to the plc common. The ac ground (3rd wire where applicable) should be connected to the frame ground terminal of the plc. (not shown) As is true with dc, ac connections are typically color coded so that the individual wiring the device knows which wire is which.

This coding varies from country to country but in the US is commonly white (neutral), black (hot) and green (3rd wire ground when applicable). Outside the US it's commonly coded as brown (hot), blue (neutral) and green with a yellow stripe (3rd wire ground where applicable).

The PLC AC input module circuit shown in figure 2.15.

:

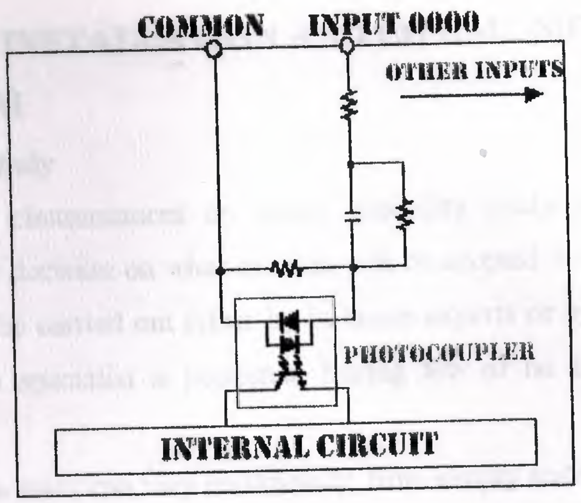


Figure 2.15. PLC AC input module circuit

The only things accessible to the user are the terminals labeled COMMON, INPUT 0000, INPUTxxxx... The common terminal gets connected to the neutral wire.

A common switch (i.e. limit switch, pushbutton, toggle, etc.) would be connected to the input terminals directly. One side of the switch would be connected directly to INPUT XXX. The other end goes to the ac hot wire. This assumes the common terminal is connected to neutral.

The photo couplers are used to isolate the PLCs internal circuit from the inputs. This eliminates the chance of any electrical noise entering the internal circuitry. They work by converting the electrical input signal to light and then by converting the light back to an electrical signal to be processed by the internal circuit.

One last note, typically an ac input takes longer than a dc input for the plc to see. In most cases it doesn't matter to the programmer because an ac input device is typically a mechanical switch and mechanical devices are slow. It's quite common for a plc to require that the input be on for 25 or more milliseconds before it's seen. This delay is required because of the filtering, which the plc internal circuit needs.

2.13. CHOOSING INSTALLATION AND COMMISSIONING OF PLC SYSTEM

2.13.1. Feasibility Study

Under certain circumstances an initial feasibility study may be suggested or warranted prior to any decision on what solution will be adopted for a particular task. The feasibility study may be carried out either by in house experts or by external consultants. Often an independent specialist is preferred, having few or no ties to specific vendor equipment.

The scope of such a study can vary enormously, from simply stating the feasibility of the proposal, through to a comprehensive case analysis with complete equipment recommendations.

Typically, though, a feasibility study of this nature encompasses several specific areas of investigation:

(a) Economic feasibility.

Consisting of the evaluation of possible installation and development costs weighed against the ultimate income or benefits resulting from a developed system.

(b) Technical feasibility.

Where the target process and equipment are studied in terms of function, performance and constraints that may relate to achieving an acceptable system.

(c) Alternatives

With an investigation and evaluation of alternative approaches to the development of the acceptable system.

Area (a), economic feasibility and worth, can only be addressed fully once the result of areas (b) and (c) are available, with estimated castings, and direct indirect benefits being considered. Area (b) is detailed in the following sections, with background information for area (a) usually being compiled through liaison with company personnel.

The achievement of a complete technical proposal requires us to know what the present and future company needs are in terms of plant automation and desired information systems.

Once the control function has been accurately defined, a suitable programmable control system has to be chosen from the wide range available. Following the identification of a suitable PLC, work can begin on aspects of electrical hardware design and software design.

2.13.2. Design Procedure for PLC System

Because the programmable controller is based on standard modules, the majority of hardware and software design and implementation can be carried out independently of, but concurrently with, each other.

Developing the hardware and software in parallel brings advantages both in terms of saving time and of maintaining the most flexible an adaptable position regarding the eventual system function. This allows changes in the actual control functions through software, until the final version is placed in the system memory and installed in the PLC.

An extremely important aspect of every design project is the documentation.

Accurate and up-to-date documentation of all phases of a project need to be fully documented and updated as the job progresses through to completion. This information will form part of the total system documentation, and can often be invaluable during later stages of commissioning and troubleshooting.

2.13.3. Choosing a programmable controller

There is a massive range of PLC Systems available today, with new additions or replacement continually being produced with enhanced features of one type or another. Manufacturers quickly adopt advances in technology in order to improve the performance and market status of their products. However, irrespective of make, the majority of PLCs in each size range are very similar in terms of their control facilities. Where significant differences are to be found is in the programming methods and languages, together with differing standards of manufacturer support and backup.

This latter point is often overlooked when choosing a suitable make of controller, but the value of good, reliable manufacturers assistance cannot be overstated, both for present and future control needs.

2.13.4. Size and type of PLC system

This may be decided in conjunction with the choice of manufacturer, on the basis that more than one make of machine can satisfy a particular application, but with the vast choice of equipment now available, the customer can usually obtain similar systems from several original equipment manufacturers (OEM's).

Where the specification requires certain types of function or input/output, it can result in one system from a single manufacturer standing out as far superior or cost-effective than the competition, but this is rarely the case. Once the stage of deciding actual size of the PLC system is reached, there are several topics to be considered:

- Necessary input/output capacity; types of I/ O required;
- Size of memory required;
- Speed and power required of the CPU and instruction set.

All this topics are to a large extent interdependent, with the memory size being directly tied to the amount of I/ O as well as program size. As the I/ O memory size rises, this takes longer to process and requires a more powerful, faster central processor if scan times arc remain acceptable.

2.13.4.1. I/0 requirements

The I/ O sections of a PLC system must be able to contain sufficient modules to connect all signal and control lines for the process. These modules must conform to the basic system specifications as regards voltage levels, loading, etc.,

- The number and type of I/ O points required per 'nodule;
- Isolation required between the controller and the target process;
- The need for high speed I/ O, or remote I/ O, or any other special facility;

•Future needs of the plant in terms of both expansions potential and installed spare I/ O points,

•Power supply requirements of I/ O points are an on board PSU needed to drive any transducer or actuators?

In certain cases there may be a need for signal conditioning modules to be included in the system, with obvious space demands on the main or remote racks. When the system is to he installed over a wide area, the use of a remote or decentralized form of I/ O working can give significant economies in cabling the sensors and actuators to the PLC.

2.13.4.2. Memory and programming requirements

Depending on the type of programmable control let being considered, the system memory may be implemented on the same card as the CPU, or alternatively on dedicated cards. This ladder method is the more adaptable, allowing memory size to be increased as necessary- up to the system maximum, without a reciprocal change in CPU card.

As stated in the previous section, memory size is normally related to the amount of I/ O points required in the system. The other factor that affects the amount of memory required is of course the control program that is to be installed.

The exact size of any program cannot be defined until of the software has been designed, encoded, installed and tested. However, it is possible to accurately estimate this size based on average program complexity A control program with complex, lengthy interlocking or sequencing routines obviously requires more memory than one for a simple process. Program size is also related to the number of I/ O points, since it must include instructions for reading from or writing to each point. Special functions are required for the control task may also require memory space in the unit PLC memory map to allow data transfer between cards. Finally additional space should be provided to allow for changes in the program, and for future expansion of the system.

There is often a choice of available memory type RAM or EPROM. The RAM form is the most common, allowing straightforward and rapid program alterations both before and after the system is installed. RAM contents are made semi-permanent by the provision of battery backing on their power supply. RAM must always be used for I/ O and data functions, as these involve dynamic data.

EPROM memory can be employed for program storage only, and requires the use of a special EPROM eraser / programmer to alter the stored code. The use of EPROMS is ideal where identical programmable controllers running the same control several machines.

However, until a program has been a hilly developed and tested, RAM storage should be used.

As mentioned in earlier chapters, microcomputers arc commonly used as program development stations. The large amounts of RAM and disk storage space provided in these machines allow the development and storage of many PLC programs, including related text and documentation. Programs can be transferred between the microcomputer and the target PLC for testing and alteration. EPROM programming can also often be carried out via the microcomputer.

2.13.4.3. Instruction set CPU

Whatever else is left undefined; any system to be considered must provide an instruction set that is adequate for the task. Regardless of size, all PLCs can handle logic control, sequencing, etc. Where differences start to emerge are in the areas of data handling, special functions and communications. Larger programmable controllers tend to have more powerful instructions than smaller ones in these areas, but careful scrutiny of small medium machines can often reveal the capability to perform specific functions at surprisingly good levels of performance.

In modular programmable controllers there may be a choice of CPU card, offering different levels of performance in terms of speed and functionality. As the number of I/ O and function cards increases, the demands on the CPU also increase, since there ate greater numbers of signals to process each cycle. This may require the use of a faster CPU card if scan time is not to suffer.

Following the selection of the precise units that will make up the programmable controller for a particular application, the software and hardware design functions can be carried out independently.

2.14. Installation

The hardware installation consists of building up to necessary racks and cubicles, then installing and connecting the cabling.

The cabinet that contains the programmable controller and associated sub-racks must be adequate for the intended environment, as regards security safety band protection from the elements:

Security in the form of a. robust, lockable cabinet; safety, by providing automatic cut off facilities alarms if the cabinet door is opened; protection from humid or corrosive atmospheres by installation of airtight seals on the cubicle. Further electrostatic shielding by earthing the cubicle body.

For maintenance purposes, there must be easy access to the PLC racks for card inspection, changing etc. Main on/ off and status indicators can be built in to the cabinet doors, and glass or Perspex windows fined to allow visual checking of card status or relay/ contactor operation.

2.15. Testing and Commissioning

Once the installation work is completed, the next step is to consider the testing and commissioning of the PLC system.

Commissioning comprises two basic stages:

1-Checking the cable connections between the PLC and the plant to be controlled.

2-Installing the completed control software and testing its operation on the target process.

The system interconnections must be thoroughly checked out to ensure all input' output devices are wired to the correct I/ O points. In a conventional control system buzzing out the connections with suitable continuity test instruments would do this. With a programmable, however, the programming panel may be used to monitor the status of inputs points directly this is long before the control software is installed which will only be done after all hardware testing is satisfactorily completed. Before any hardware testing is started, a thorough test of all mains voltages, earthing, etc. must be carried out.

With the programmer attached to the PLC, input points are monitored as the related transducer is operated, checking that the correct signal is received by the PLC. The same technique is used to test the various function cards installed in the system. For example, altering can check analog inputs the analog signal and observing a corresponding change in the data stored in the memory table.

In turn, the output devices can be forced by instructions from the programming panel. Checking their connection and operation.

The commissioning team must ensure that any operation or disoperation of plant actuators will not result in damage to plant or personnel.

Testing of some PLC functions at this stage is not always practical, such as for PID loops and certain communications channel. These require a significant amount of configuring by software before they can be operated, and are preferably tested once the control software has been installed.

Some programmable controllers contain in built diagnostic routines that can be used to check out the installed cards, giving error codes on a VDU or integral display screen. These diagnostic are run by commands from the programming panel, or from within a control program once the system is fully operational.

2.16. Software testing and simulation

The preceding sections have outlined the various stages in hardware design and implementation. Over the same period of time, the software to control the target process is developed, in parallel, for the chosen PLC system. These program modules should be tested and proved individually wherever possible, before being linked together to make up' the complete applications program. It is highly desirable that any faults or error be removed before the program is installed in the host controller.

The time required to rectify faults can be more than doubled once the software is running in the host PLC.

Virtually all-programmable controllers, irrespective of size, contain elementary software checking facilities. Typically these can scan through an installed program to check for incorrect labels. Double output coils etc, Listings of all I/ O points used, counter/ timer settings and other information is also provided. The resulting information is available on the programmer screen or as a printout. However, this form of testing is only of limited value, since there is no facility to check the operation of the resident program.

In terms of time and cost economies, an ideal method for testing program modules is to reproduce the control cycle by simulation; since this activity can be carried out in the design workshop without having the actually connect up to the physical process. Simulation of tile process is done in a number of ways, depending on the size of process involved.

When the system is relatively small with only a handful of I/ O channels it is often possible to adequately simulate the process by using. Sets of switches connected up to the PLC as inputs, with outputs represented by connecting arrays of small lambs or relays in the figure 10.4. This allows inputs to be offered to a test bed controller containing software under test, checking the action of the control program by noting the operation and sequence of the output lambs or relays. By operating the input switches in specific sequences, it is possible to test sequence routines within a program. Where fast response times are involved, the tester should use the programming panel to force larger time intervals into the timers concerned, allowing that part of the circuit to be tested by the manual switch method.

Most I/ O modules have LED indicators that show tile status of the channels. These can be used instead of additional test actuators where digital outputs arc concerned Analog inputs can be simulated in part by using potential dividers suitably connected to the input channel, and corresponding analog outputs connected either to variable devices such as small motors or to a moving coil meter configured to measure voltage or current. Standard sets of input switches and output actuators are normally available from PLC manufacturers.

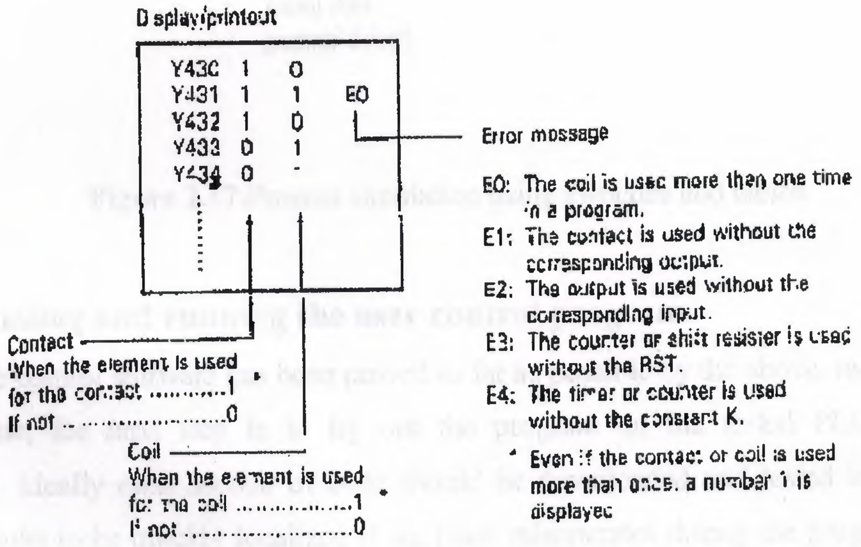
When the system is larger with input/output channels and longer, more complex programs, the simple form of simulation described above becomes inadequate. Many larger PLC Systems are fitted an integral simulation unit that reads and writes information directly into the I/ O memory, removing the need to connect external switches, etc. The simulator is controlled from an associated terminal, which can force changes in input status and record all changes in output status as the program runs, for later scrutiny by the test team.

The program monitoring facility provided with most programming terminals should be used in virtually all these proceedings, since it allows the dynamic checking of all elements in the program including preset and remaining values as the program cycles.

It is important to realize that the display on the programmer does not up date as rapidly as the control program is executing, due to the delays in transmitting the data across to the terminal.

Contacts and other elements that are operated for only a few scans are unlikely to affect the display, but since a human observer could not detect this fast a change this is not a significant disadvantage. To display all changes, the PLC should be run in single step mode.

The monitor display shows a select portion of the ladder program, using standard symbols to depict contacts, output and present functions. All elements within the display are dynamically monitored. (See figure 2.16 and 2.17).



D splay/printout
Y430 1 0
Y431 1 1 E0
Y432 1 0
Y433 0 1
Y434 0 -
Error message
E0: The coil is used more then one time in a program.
E1: The contact is used without the corresponding output.
E2: The output is used without the corresponding input.
E3: The counter or shift resister is used without the RST.
E4: The timer or counter is used without the constant K.
* Even if the contact or coil is used more than once, a number 1 is displayed
Contact
When the element is used for the contact ....1 *
If not 0
Coil
When the element is used for the coil ....1 *
If not 0

Figure 2.16.PLC printout of I/O static diagnostics information

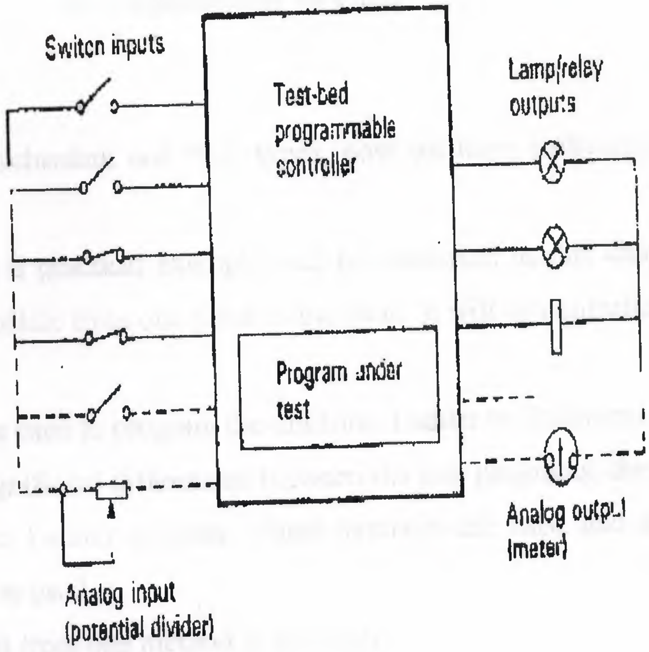


Figure 2.17.Process simulation using switches and lambs

2.17. Installing and running the user control program

Once the control software has been proved as far as possible by the above, methods on a test machine, the next step is to try out the program on the tested PLC hardware installation. Ideally each section of code should be downloaded and tested individually, allowing faults to be quickly localized if the plant misoperates during the program test. If this subdivided testing is not possible, another method is to include JUMP commands in the complete program to miss out all instructions except those in the section to be tested. As each section is proved, the program is amended to place the JUMP instructions so as to select the next section to be tested.

Where a programmable controller supports single step operation, this can be used the examine individual program steps for correct sequencing. Again the programming terminal should be utilized to monitor I/ O status or any other area of interest during these tests. With continuous printouts if this is possible.

3. Application of PLC

3.1. Introduction

After we explore the mechanism and PLC types, now we have sufficient idea about PLC.

For more understanding a practical example will be discussed in this chapter, it is a machine, which carries a particle from one place to the other, it will be controlled by PLC.

3.1.1. Process

Two methods can be used to program the machine, Ladder or Statement.

Actually there are no significant differences between the tow programs, they both carry the same algorithm, but in Ladder program visual symbols are used and in Statement program written syntaxes are used.

The program can convert from one method to the other.

3.1.2. How does it work?

The machine is used shown in figure 3.1.

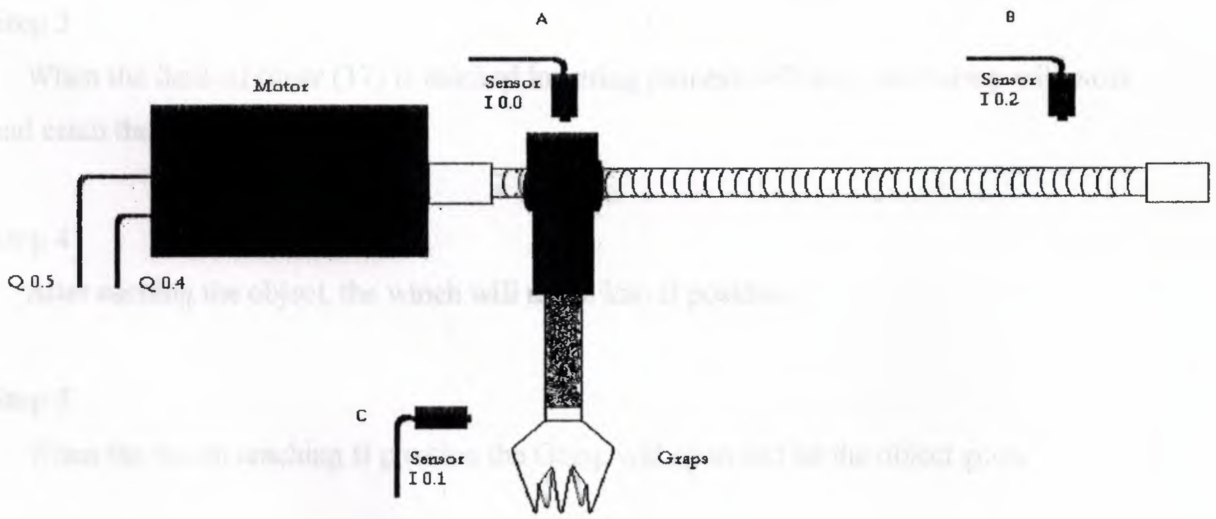


Figure 3.1. Shown the machine schematic

Where:

I0.0	Left Limit
I0.1	Upper Limit
I0.2	Right Limit
Q0.0	Lowering
Q0.1	Grasp Open
Q0.2	Grasp Close
Q0.3	Lifting
Q0.4	Turn Right
Q0.5	Turn Left

Step 1

Computer sends the program orders to the PLC device, which is connected to the machine by input and output wires.

An advantage of PLC device that has the ability to save the program in its memory, so existence of the computer is no more important.

Step 2

When the winch in A position, the Grasp is opened and the winch is lowering.

Step 3

When the limit of timer (37) is reached lowering process will stop, and Grasp will work and catch the object.

Step 4

After caching the object, the winch will move into B position.

Step 5

When the winch reaching B position the Grasp will open and let the object goes,

Step 6

Then the winch will go up until it cuts the beams of sensor (I0.1), then the sensor will turn off.

Step 7

When the sensor turn off the winch will go back to A position.

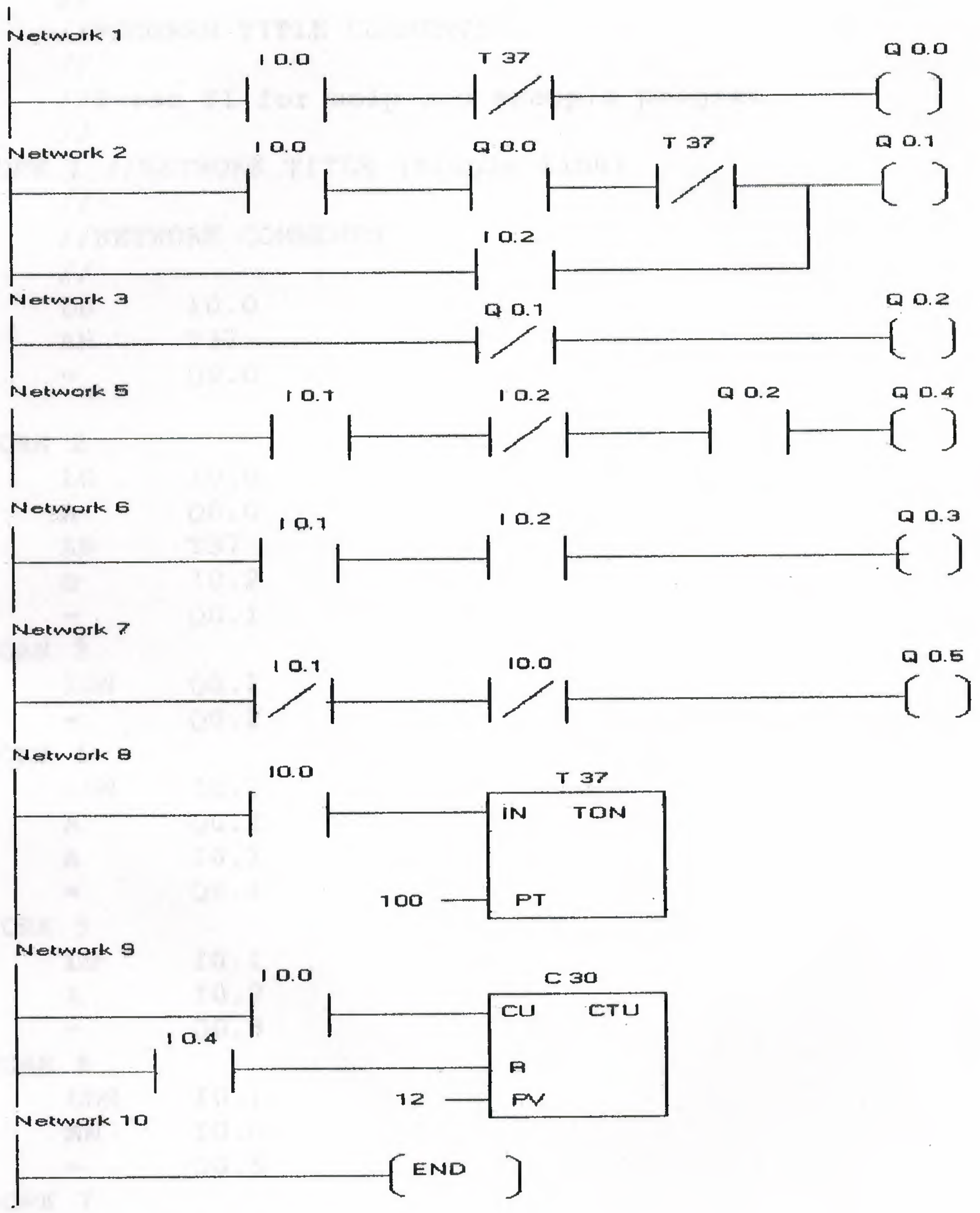
Step 8

Using a counter, which is found in the main program, controls repeating the process.

3.1.3. Software approach

In this section the used program for this project will be shown in tow method **ladder** and **Statement.**

3.1.3.1. Ladder Program



Block diagram 3.1. Ladder program

3.1.3.2. Statement program

```
        //
        //PROGRAM TITLE COMMENTS
        //
        //Press F1 for help and example program
        //
NETWORK 1 //NETWORK TITLE (single line)
        //
        //NETWORK COMMENTS
        //
        LD      I0.0
        AN      T37
        =       Q0.0
NETWORK 2
        LD      I0.0
        A       Q0.0
        AN      T37
        O       I0.2
        =       Q0.1
NETWORK 3
        LDN     Q0.1
        =       Q0.2
NETWORK 4
        LDN     I0.2
        A       Q0.2
        A       I0.1
        =       Q0.4
NETWORK 5
        LD      I0.1
        A       I0.2
        =       Q0.3
NETWORK 6
        LDN     I0.1
        AN      I0.0
        =       Q0.5
NETWORK 7
        LD      I0.0
        TON     T37, +100
NETWORK 8
        LD      I0.0
        LD      I0.4
        CTU     C30, +12
NETWORK 9
        MEND
```

Figure 3.1. Statement program

CONCLUSION

When developing this project we see that **PLC** is making the operation in industrial places, and that's the reasons, why it's gaining interest. (Notice of most of the companies.)

With the information observed from the lecturer and our researchers for this topic **PLC**, is a convenient tool wit a wide range of useful ways to be used. Such examples can be mentioned several machines cab be used at the same time, easy adjustments from the **PLC** program can be made within a few minutes by the keyboard, installed **PLC** programs can be controlled or checked before within the once laboratory, even the **PLC** program as for firm can be meet at the home.

Its very portative and safe for the workers which they protected the danger, communication programs of **PLCs** within each other or within operates can happen with the PLC; the developed lances have constructed the productivity, security establishment security fast productivity, quality and we can see that **PLC** is a very cheap program that can be fundamentally used.

REFERENCES

Books:

(1) Hugh Jack (June 1999). Programmable Logic controllers.

(2) Alan J.Crispin. Programmable Logic controller and their engineering applications.

(3) Ian G. Warnock. Programmable controllers operation and application.

Websites

1- www.plcs.com.

2- www.serch.kent.edu.com