**A Project/Problem** **Report**

**On**

Speed controlling a Brushless DC Motor (BLDCM) using a Programmable Logic Controller (PLC)

##### A PROJECT REPORT

***Submitted by***

Bhatiya Lalabhai Govabhai NO 080220111003

Gajjar Tejaskumar Rameshbhai NO 080220111007

Patel Hardikkumar Jayantilal NO 080220111032

***In partial fulfillment for the award of the degree***

***Of***

**Bachelor of Electronics and Communication Engineering**



**Department of Electronics and Communication Engineering**

Government Engineering College, Patan

Gujarat Technological University, Ahmedabad

ABSTRACT

* The aim of this Problem is to investigate controlling a Brushless DC Motor (BLDCM) using a Programmable Logic Controller (PLC) instead of micro-controllers chips. As PLCs are now involved in most industrial processes, therefore, developing a program to handle the control of BLDCMs will save electronic components used in the drive circuit. Further, one PLC may control more than one motor via programming extra inputs and outputs already implemented in the PLC or simply by attaching additional input/ output cards. The speed is varied through the PWM technique. The PLC performed well with speeds up to 1550 rpm. The motor did not run faster due to the switching delay (scanning cycle time and hardware) of the PLC.

Chepter I

1.1 INTRODUCTION

* The project is about to assemble a controling a controlling a Brushless DC Motor (BLDCM) using a Programmable Logic Controller (PLC) as its various methods. Brushed DC motors have been used in industry due to their linear characteristics and the ease of adjusting their speed through a simple power electronic circuit. The commutator used in brushed DC motors is the main drawback for such systems which has motivated the researchers to direct their studies toward AC systems, induction and synchronous motors. Since the last decade, induction motors have dominated the industry due to the availability of the induction motors variable speed drives. However, these motors operate at low efficiency specially those of low power ratings. The solution for the drawbacks of brushed DC motors and AC motors can be found in BLDCMs, with superior performance. BLDCMs have similar characteristics to the separately excited DC machines but their control is similar to the AC machines control. This paper investigates the possibility of using a PLC (Programmable Logic controllers) to control such motors rather than microcontrollers.

1.2 PROJECT OVERVIEW

* Brush DC motors have been utilized in industry because of their linear typical and the easiness of correcting their speed throughout a easy of power circuit electronic. The commutator utilized in brushed DC motors is the major disadvantage from such systems which has inspired the researchers to straight their studies on the road to AC systems, synchronous and induction motors. As the last decade, the motors of induction have led the industry because of the accessibility of the variable speed drives of induction motors. Nevertheless, these motors run at medium efficiency particularly those of little power ratings. The drawbacks brushed solution motors of AC and DC can be established onto BLDCMs, with better performance. BLDCM have alike features to the independently stimulated DC machines but their control is similar to the machines control of AC. This article investigates the likelihood of using a PLC to manage thus motors than microcontrollers.
* Brushless DC motors can be separated into categories, BEMF Sinusoidal and BEMF Trapezoidal. The current learn arrangements with a motor BLDC with a BEMF trapezoidal. BEMF of BLDCM has a wave form of trapezoidal and the winding stator is supplied by a current of rectangular to generate constant torque. The three stage windings are put on the rotor and the stator (with magnets) is free to shift. There are always two stages on at any one time to offer continuous torque.
* The whole scheme for a BLDCM which demonstrates the cycle followed to take such systems. As it is known, the controller passes the time the signals provided by the sensors position. Then, the controller gets an achievement biasing the power transistors to change them on or off so that the motor windings are powered based on the position of rotor. The motor will operate constantly as the system makes an action loop.
* The PLC of Toshiba T1-16s was utilized. The PLC contains phototransistors, 2 transistors and 6 relay outputs, 8 inputs, and 8 outputs which are mostly used to supply PWM or clock signal. The majority of the results are outputs of relay that are not quick enough to give quick exchanging and consist little life span. To conquer this, a card of transistor output, type sink, is added to the PLC, so that all the signals of output are acquired from these transistors. The circuit of driving is realized to make six controlling of power MOSFETs the BLDCM.

1.3 OBJECTIVES

* The objectives of this project are to design a motor controller using PLC. After design process is done, the process are about to assemble a DC motor driver. To make this motor work automatically PLC Program required controlling the motor speed.

1.4 SCOPE OF WORK

* In the beginning of the project development, several factors were considered to determine the limitation or final outcome of the product. For this project, the aim is to develop a PLC circuit in order to complete the hardware part and PLC program part, these three parts are put together on a same board after it is fully function on each session and part.

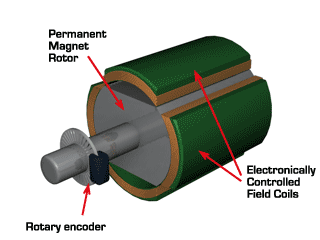
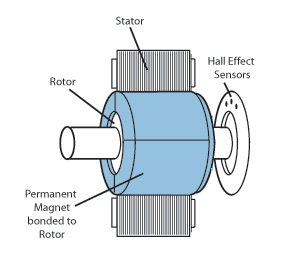
1.5 THESIS OUTLINE

1. **Chapter I**
   * Chapter I will focus on brief introduction of the project carried. The important overview or description including the problem statement, project objectives and project scopes are well emphasized in this part.
2. **Chapter II**
   * Chapter II will be based on the literature review of the project, It is mainly focused on the introduction about brushless DC motor and its componment. This is including DC motor, the driver system. On the starting part, readers will be told about all the randomly selected technology that available on market. For an example the DC motor type that available. For the finishing part, a comparison will be made and the most suitable method will be chosen.
3. **Chapter III**
   * Chapter III mainly focused on the introduction of PLC, its feature.Introduction of PLC programming Language and ladder logic diagram.
4. **Chapter IV**
   * Chapter III mainly focused on Experimental setup and Control system of BLDCM.
5. **Chapter V**
   * Chapter IV mainly focused on results and analysis done using the device, all testing and verification result are attached with the aid of figure, table and statistic related to the project.

Chapter II

2.1 LITERATURE REVIEW AND CONCEPTS

* DC motor is a device that works according to current, magnetic field, voltage, torque, load and power. Each changes of the element will make a change in motor speed. For an example, the speed of a DC motor will increase if we supply more voltage to its supply. The load will reduce the motor speed and overload can make the motor stop immediately. DC motor can be classified in 3 group which is separately excited, shunt, and series. At early stages, this part will explain about the DC motor itself and then move to the controller parts. DC motor controller is simpler than an AC motor. It doesn’t need an expensive inverter. The driver is cheaper and the cost of DC motor itself are cheaper than AC motor. However there are some advantages and disadvantages in the performance. Readers will be explained clearly in this part about the advantages and disadvantages of the motor and the driver. In this part I will then continue to explain briefly about the technique that available in controlling the motor speed such as chopper control, H-bridge, resistant and phase control. The best selected controller will be discussed in the next part (methodology).



***Figure 1: Brushless DC Motor***

* Brushless DC motors can be divided into types, Sinusoidal BEMF and Trapezoidal BEMF. The present study deals with a BLDC motor with a trapezoidal BEMF . BLDCMs’ BEMF has a trapezoidal wave form and the stator winding is fed by a rectangular current to produce constant torque. The three phase windings are placed on the stator and the rotor (with magnets) is free to move. There are always two phases on at any one time to provide continuous torque, as illustrated in Fig.1. As Given below:



***Figure 2: Waveform***

*In This Waveform:*

1. **Flux linkage wave form,**
2. **Back-EMF,**
3. **Current wave form,**
4. **Torque from coil-1,**
5. **Torque from coil-2,**
6. **Torque from coil-3,**
7. **Total developed torque**

2.2 BLDCM Drive System

* The complete scheme for a BLDCM is shown in Fig. 3 which illustrates the sequence followed to drive such systems. As it is seen, the controller waits the signals supplied by the position sensors.
* Then, the controller takes an action biasing the power transistors to switch them on or off so that the windings in the motor are powered according to the rotor position. The motor will run continuously as the system constructs a loop action.



***Figure 3-BLDCM drive system***

2.3 Advantages of BLDCM

1. There is no sparking & less electrical noise.
2. Smoother operation
3. Acceleration control
4. Different operating speed for each process recipe
5. Compensate for changing process variables
6. Allow slow operation for setup purposes
7. Adjust the rate of production
8. High speed can be achieved
9. Allow accurate positioning
10. Control torque or tension
11. As there are no brushes it requires no servicing.
12. It has longer life.

2.4 Application of BLDCM

|  |  |
| --- | --- |
|  |  |
| GMRT ANTENNA | SERVO MOTOR (at elevation) AT GMRT ANTENNA |
|  |  |
| Tape drives | Winders |
|  |  |
| Artificial heart | Microscopes |
| BOTTEL.jpg | economy-conveyor-belt-frontpage.jpg |
| Automatic botel filling system | Conveyor belt |

Chapter III

3.1 Introduction to Programmable Logic Controller

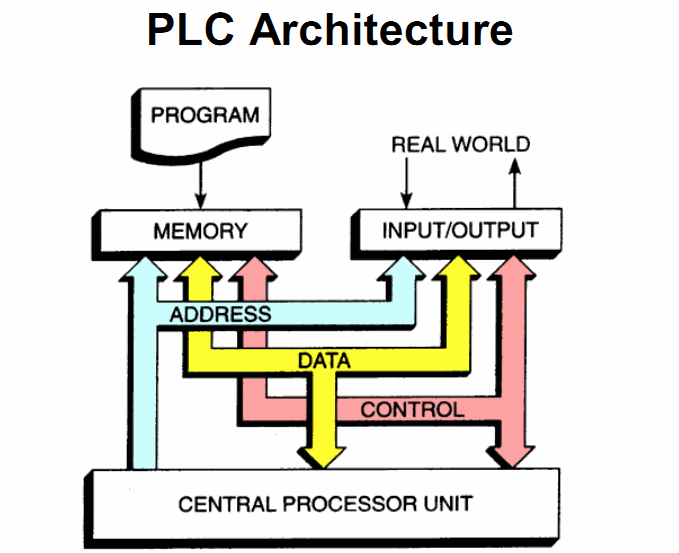
* A programmable logic controller (PLC) or programmable controller is a digital computer used for automation of electromechanical processes, such as control of machinery on factory assembly lines, amusement rides, or light fixtures. PLCs are used in many industries and machines. Unlike general-purpose computers, the PLC is designed for multiple inputs and output arrangements, extended temperature ranges, immunity to electrical noise, and resistance to vibration and impact. Programs to control machine operation are typically stored in battery-backed or non-volatile memory. A PLC is an example of a hard real time system since output results must be produced in response to input conditions within a bounded time, otherwise unintended operation will result.
* In this course module, the PLC hardware is discussed. After studying this module, the student will know the major components of a PLC and the functions of these components.

3.2 The PLC System

A programmable logic controller consists of the following components:

1. Central Processing Unit (CPU)
2. Memory
3. Input modules
4. Output modules and
5. Power supply.

* A PLC hardware block diagram is shown in Figure 1. The programming terminal in the diagram is not a part of the PLC, but it is essential to have a terminal for programming or monitoring a PLC. In the diagram, the arrows between blocks indicate the information and power flowing directions.

***Figure 1: PLC Hardware Block Diagram***

3.3 CPU

Like other computerized devices, there is a Central Processing Unit (CPU) in a PLC. The CPU, which is the “brain” of a PLC, does the following operations:

* Updating inputs and outputs. This function allows a PLC to read the status of its input terminals and energize or deenergize its output terminals.
* Performing logic and arithmetic operations. A CPU conducts all the mathematic and logic operations involved in a PLC.
* Communicating with memory. The PLC’s programs and data are stored in memory. When a PLC is operating, its CPU may read or change the contents of memory locations.
* Scanning application programs. An application program, which is called a ladder logic program, is a set of instructions written by a PLC programmer. The scanning function allows the PLC to execute the application program as specified by the programmer.
* Communicating with a programming terminal. The CPU transfers program and data between itself and the programming terminal.
* A PLC’s CPU is controlled by operating system software. The operating system software is a group of supervisory programs that are loaded and stored permanently in the PLC’s memory by the PLC manufacturer.

3.4 Memory

* Memory is the component that stores information, programs, and data in a PLC. The process of putting new information into a memory location is called *writing*. The process of retrieving information from a memory location is called *reading*.
* The common types of memory used in PLCs are Read Only Memory (ROM) and Random Access Memory (RAM). A ROM location can be read, but not written. ROM is used to store programs and data that should not be altered. For example, the PLC’s operating programs are stored in ROM.
* A RAM location can be read or written. This means the information stored in a RAM location can be retrieved and/or altered. Ladder logic programs are stored in RAM. When a new ladder logic program is loaded into a PLC’s memory, the old program that was stored in the same locations is over-written and essentially erased.
* The memory capacities of PLCs vary. Memory capacities are often expressed in terms of *kilo-bytes* (K). One byte is a group of 8 bits. One bit is a memory location that may store one binary number that has the value of either 1 or 0. (Binary numbers are addressed in Module 2). 1K memory means that there are 1024 bytes of RAM. 16K memory means there are 16 x 1024 =16384 bytes of RAM.

3.5 Input modules and output modules

* A PLC is a control device. It takes information from inputs and makes decisions to energize or de-energize outputs. The decisions are made based on the statuses of inputs and outputs and the ladder logic program that is being executed.
* The input devices used with a PLC include pushbuttons, limit switches, relay contacts, photo sensors, proximity switches, temperature sensors, and the like. These input devices can be AC (alternating current) or DC (direct current). The input voltages can be high or low. The input signals can be digital or analog. Differing inputs require different input modules. An input module provides an interface between input devices and a PLC’s CPU, which uses only a low DC voltage. The input module’s function is to convert the input signals to DC voltages that are acceptable to the CPU. Standard discrete input modules include 24 V AC, 48 V AC, 120 V AC, 220 V AC, 24 V DC, 48 V DC, 120 V DC, 220 V DC, and transistor-transistor logic (TTL) level.
* The devices controlled by a PLC include relays, alarms, solenoids, fans, lights, and motor starters. These devices may require different levels of AC or DC voltages. Since the signals processed in a PLC are low DC voltages, it is the function of the output module to convert PLC control signals to the voltages required by the controlled circuits or devices. Standard discrete output modules include 24 V AC, 48 V AC, 120 V AC, 220 V AC, 24 V DC, 48 V DC, 120 V DC, 220 V DC, and TTL level.

3.6 Power Supply

* PLCs are powered by standard commercial AC power lines. However, many PLC components, such as the CPU and memory, utilize 5 volts or another level of DC power. The PLC power supply converts AC power into DC power to support those components of the PLC.

3.7 Programming Terminal

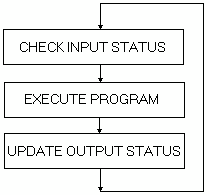
* A PLC requires a programming terminal and programming software for operation. The programming terminal can be a dedicated terminal or a generic computer purchased anywhere. The programming terminal is used for programming the PLC and monitoring the PLC’s operation. It may also download a ladder logic program (the sending of a program from the programming terminal to the PLC) or upload a ladder logic program (the sending of a program from the PLC to the programming terminal). The terminal uses programming software for programming and “talking” to a PLC.

3.8 Programmable Logic Controller Features

* The main difference from other computers is that PLCs are armored for severe conditions (such as dust, moisture, heat, cold) and have the facility for extensive input/output (I/O) arrangements. These connect the PLC to sensors and actuators. PLCs read limit switches, analog process variables (such as temperature and pressure), and the positions of complex positioning systems. Some use machine vision. On the actuator side, PLCs operate electric motors, pneumatic or hydraulic cylinders, magnetic relays, solenoids, or analog outputs. The input/output arrangements may be built into a simple PLC, or the PLC may have external I/O modules attached to a computer network that plugs into the PLC.

3.9 Scan Time of PLC

* A PLC program is generally executed repeatedly as long as the controlled system is running. The status of physical input points is copied to an area of memory accessible to the processor, sometimes called the "I/O Image Table". The program is then run from its first instruction run down to the last rung. It takes some time for the processor of the PLC to evaluate all the ladders and update the I/O image table with the status of outputs. This scan time may be a few milliseconds for a small program or on a fast processor, but older PLCs running very large programs could take much longer (say, up to 100 ms) to execute the program. If the scan time was too long, the response of the PLC to process conditions would be too slow to be useful.
* As PLCs became more advanced, methods were developed to change the sequence of ladder execution, and subroutines were implemented. This simplified programming and could also be used to save scan time for high-speed processes; parts of the program used, for example, only for setting up the machine could be segregated from those parts required to operate at higher speed.
* Special-purpose I/O modules, such as timer modules or counter modules, could be used where the scan time of the processor was too long to reliably pick up, for example, counting pulses from a shaft encoder. The relatively slow PLC could still interpret the counted values to control a machine, but the accumulation of pulses was done by a dedicated module that was unaffected by the speed of the program execution.



***Figure 2: The three steps of a scan cycle***

1. Step 1-**CHECK INPUT STATUS**-First the PLC takes a look at each input to determine if it is on or off. In other words, is the sensor connected to the first input on? How about the second input? How about the third... It records this data into its memory to be used during the next step.
2. Step 2-**EXECUTE PROGRAM**-Next the PLC executes your program one instruction at a time. Maybe your program said that if the first input was on then it should turn on the first output. Since it already knows which inputs are on/off from the previous step it will be able to decide whether the first output should be turned on based on the state of the first input. It will store the execution results for use later during the next step.
3. Step 3-**UPDATE OUTPUT STATUS**-Finally the PLC updates the status of the outputs. It updates the outputs based on which inputs were on during the first step and the results of executing your program during the second step. Based on the example in step 2 it would now turn on the first output because the first input was on and your program said to turn on the first output when this condition is true.

3.10 Programming of PLC

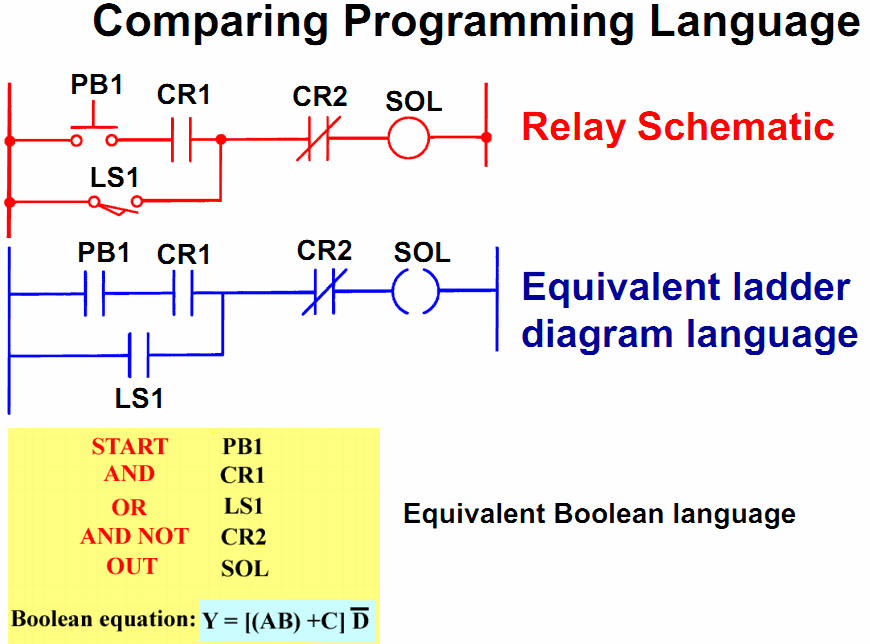
* PLC programs are typically written in a special application on a personal computer, then downloaded by a direct-connection cable or over a network to the PLC. The program is stored in the PLC either in battery-backed-up RAM or some other non-volatile flash memory. Often, a single PLC can be programmed to replace thousands of relays.
* Under the IEC 61131-3 standard, PLCs can be programmed using standards-based programming languages. A graphical programming notation called Sequential Function Charts is available on certain programmable controllers. Initially most PLCs utilized Ladder Logic Diagram Programming, a model which emulated electromechanical control panel devices (such as the contact and coils of relays) which PLCs replaced. This model remains common today.
* IEC 61131-3 currently defines five programming languages for programmable control systems: function block diagram (FBD), ladder diagram (LD), structured text (ST; similar to the Pascal programming language), instruction list (IL; similar to assembly language) and sequential function chart (SFC). These techniques emphasize logical organization of operations.

3.11 Programming Language of PLC

* While the fundamental concepts of PLC programming are common to all manufacturers, differences in I/O addressing, memory organization and instruction sets mean that PLC programs are never perfectly interchangeable between different makers. Even within the same product line of a single manufacturer, different models may not be directly compatible.

There are three types of PLC programming Language:

1. Ladder Diagram Language
2. Boolean Langauge
3. Functional Chart



3.12 Ladder Diagram Language Instruction

* The ladder diagram language is basically a symbolic set of instructions used to create the controller program. These ladder instructions symbols are arranged to obtain the desired control logic.

**Different Types of Instruction in Ladder Logic:**

**1. XIC**

* The XIC (eXamine the input to see If it’s physically Closed) instruction is a normally open contact. Sometimes it is also called load (LD) instruction. The symbol for an XIC instruction is shown below



***Figure 3: An XIC (normally open 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.

**2. XIO**

* The XIO (eXamine the input to see If its physically Opened) instruction is a normally closed contact. Sometimes it is also called LoaDNot (LDN) instruction. The symbol for an XIO instruction is shown below



***Figure 4: An XIO (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.

**3. OTE or OUT**

* The OTE (OuTputEnergize) instruction is sometimes also called Out instruction. The output instruction is like a relay coil. Its symbol looks as shown below.

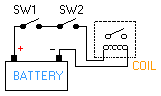


***Figure 5: An OTE (coil) symbol***

* 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.

3.13 Ladder Logic Diagrams

* The first PLCs were programmed with a technique known as Ladder Logic Diagrams. This was based on relay logic wiring schematics which eliminated the need to teach the electricians, technicians and engineers how to program a computer - but, this method has stuck and it is the most common technique for programming PLCs today. The method is explained via the following example:
* Let's consider a real world physically connected relay circuit (Fig. 6).



***Figure 6: A Simple circuit***

* A relay is a simple device that uses a magnetic field to control a switch, as pictured in Fig. 5. When a voltage is applied to the input coil, the resulting current creates a magnetic field. The magnetic field pulls a metal switch towards it and the contacts touch, closing the switch. The contact that closes when the coil is energized is called normally open. The normally closed contacts touch when the input coil is not energized. Relays are normally drawn in schematic form using a circle to represent the input coil.
* 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).



***Figure 7: Circuit converted to ladder diagram***

* Notice in this simple one rung ladder diagram we have recreated the external circuit above with a ladder diagram. Here we used the XIC and OTE instructions. Some manufacturers require that every ladder diagram include an END instruction on the last rung.
* Generally, to interpret these diagrams imagine that the power is on the vertical line on the left hand side which we call the hot rail. On the right hand side is the neutral rail. On each rung, we may have combinations of inputs (two vertical lines) and outputs (circles). If the inputs are opened or closed in the right combination the power can flow from the hot rail, through the inputs, to power the outputs, and finally to the neutral rail. An input can come from a sensor, switch, or any other type of sensor. An output will be some device outside the PLC that is switched on or off, such as lights or motors.

3.14 Experimental Work Using LogixPro

**The PLC simulator LogixPro**

* **LogixPro** is actually three distinct programs combined into a single package. First, **LogixPro** contains a PLC Ladder Logic **editor** that allows users to create and edit PLC programs using a series of PLC instructions. Secondly, **LogixPro** **emulates** the scanning sequence of a PLC. When placed into the "RUN" mode, the users program is scanned and the appropriate I/O is updated just as would occur in an actual PLC. Thirdly, **LogixPro** contains a number of **animated simulations** which respond accurately to the inputs, and outputs of the emulated PLC.

**Getting Started:**

**RSLogix Relay Logic Instructions**

* This exercise is designed to familiarize you with the operation of LogixPro and to step you through the process of creating, editing and testing simple PLC programs utilizing the Relay Logic Instructions supported by RSLogix.

logbar-250x34

***Figure 8***

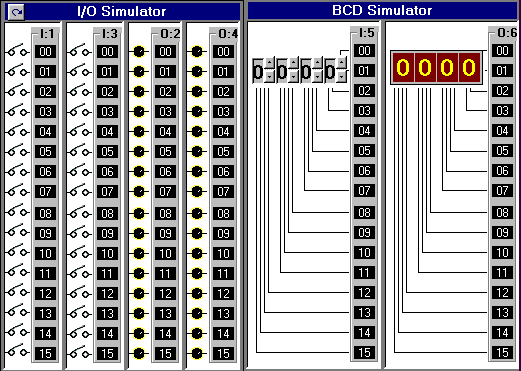
* From the Simulations Menu at the top of the screen, Select the I/O Simulation and ensure that the User Instruction Bar shown above is visible.

log-1-486x40

***Figure 9***

* The program editing window should contain a single rung. This is the End of Program rung and is always the last rung in any program. If this is the only rung visible then your program is currently empty.
* If your program is not empty, then click on the File menu entry at the top of the screen and select "New" from the drop-down list. A dialog box will appear asking for you to select a Processor Type. Just click on "OK" to accept the default TLP LogixPro selection.Now maximize





***Figure : 10 The ProSim-II Simulation Window***

**The I/O Simulator**

* The simulator screen shown above, should now be in view. For this exercise we will be using the I/O simulator section, which consists of 32 switches and lights. Two groups of 16 toggle switches are shown connected to 2 Input cards of our simulated PLC. Likewise two groups of 16 Lights are connected to two output cards of our PLC. The two input cards are addressed as "I:1" and "I:3" while the output cards are addressed "O:2" and "O:4".
* Use your mouse to click on the various switches and note the change in the status color of the terminal that the switch is connected to. Move your mouse slowly over a switch, and the mouse cursor should change to a hand symbol, indicating that the state of switch can be altered by clicking at this location. When you pass the mouse over a switch, a "tool-tip" text box also appears and informs you to "Right Click to Toggle Switch Type". Click your right mouse button on a switch, and note how the switch type may be readily changed.

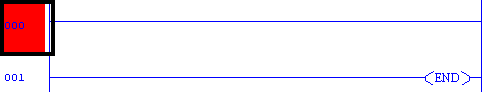
**RSLogix Program Creation**

* Collapse the I/O simulation screen back to its normal size by clicking on the same (center) button you used to maximize the simulation's window. You should now be able to see both the simulation and program windows again. If you wish, you can adjust the relative size of these windows by dragging the bar that divides them with your mouse.
* I want you to now enter the following single run program which consists of a single Input instruction (XIC - Examine If Closed) and a single Output instruction (OTE - Output Energize). There's more than one way to accomplish this task, but for now I will outline what I consider to be the most commonly used approach.

log-2-486x32

***Figure 11***

* log-rg-27x24First click on the "New Rung" button in the User Instruction Bar. It's the first button on the very left end of the Bar. If you hold the mouse pointer over any of these buttons for a second or two, you should see a short "ToolTip" which describes the function or name of the instruction that the button represents.



***Figure 12***

* You should now see a new Rung added to your program as shown above, and the Rung number at the left side of the new rung should be highlighted. Note that the new Rung was inserted above the existing (END) End Of Program Rung. Alternatively you could have dragged (left mouse button held down) the Rung button into the program window and dropped it onto one of the locating boxes that would have appeared.
* log-xic-27x24Now click on the XIC instruction with your left mouse button (Left Click) and it will be added to the right of your highlighted selection. Note that the new XIC instruction is now selected (highlighted). Once again, you could have alternatively dragged and dropped the instruction into the program window.
* **If you accidentally add an instruction which you wish to remove, just Left Click on the instruction to select it, and then press the "Del" key on your keyboard. Alternatively, you may right click on the instruction and then select "Cut" from the drop-down menu that appears.**
* log-ote-27x24Left Click on the OTE output instruction and it will be. Added to the right of your current selection.

log-4-486x36

***Figure 13***

* Double Click (2 quick left mousebutton clicks) on the XIC instruction and a textbox should appear which will allow you to enter the address (I:1/0) of the switch we wish to monitor. Use the Backspace key to get rid of the "?" currently in the textbox. Once you type in the address, click anywhere else on the instruction (other than the textbox) and the box should close.
* Right Click on the XIC instruction and select "Edit Symbol" from the drop-down menu that appears. Another textbox will appear where you can type in a name (Switch-0) to associate with this address. As before, a click anywhere else will close the box.

log-5-486x48

***Figure 14***

* Enter the address and symbol for the OTE instruction and your first RSLogix program will now be complete. Before continuing however, Double check that the addresses of your instructions are correct.

**Testing your Program**

* It's now time to "Download" your program to the PLC. First click on the "Toggle" button at the top right corner of the Edit Panel which will bring the PLC Panel into view.

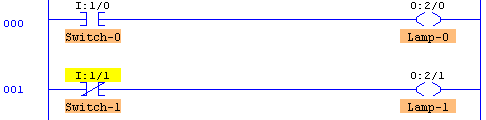
log-6-296x36

***Figure 15***

* Click on the "DownLoad" button to initiate the downloading of your program to the PLC. Once complete, click inside the "RUN" option selection circle to start the PLC scanning.
* Enlarge the Simulation window so that you can see both the Switches and Lamps, by dragging the bar that separates the Simulation and Program windows to the right with your mouse. Now click on Switch I:1/00 in the simulator and if all is well, Lamp O:2/00 should illuminate.
* Toggle the Switch On and Off a number of times and note the change in value indicated in the PLC Panel's status boxes which are being updated constantly as the PLC Scans. Try placing the PLC back into the "PGM" mode and then toggle the simulator's Switch a few times and note the result. Place the PLC back into the "Run" mode and the Scan should resume.
* We are usually told to think of the XIC instruction as an electrical contact that allows electrical flow to pass when an external switch is closed. We are then told that the OTE will energize if the flow is allowed to get through to it. In actual fact the XIC is a conditional instruction which tests any bit that we address for Truth or a 1.

**Editing your Program**

* Click on the "Toggle" button of the PLC Panel which will put the PLC into the PGM mode and bring the Edit Panel back into view.
* Now add a second rung to your program as shown below. This time instead of entering the addresses as you did before, try dragging the appropriate address which is displayed in the I/O simulation and dropping it onto the instruction.
* **Note** that the XIO instruction which Tests for Zero or False has it's address highlighted in yellow. This indicates that the instruction is True, which in the case of an XIO, means that the bit addressed is currently a Zero or False.



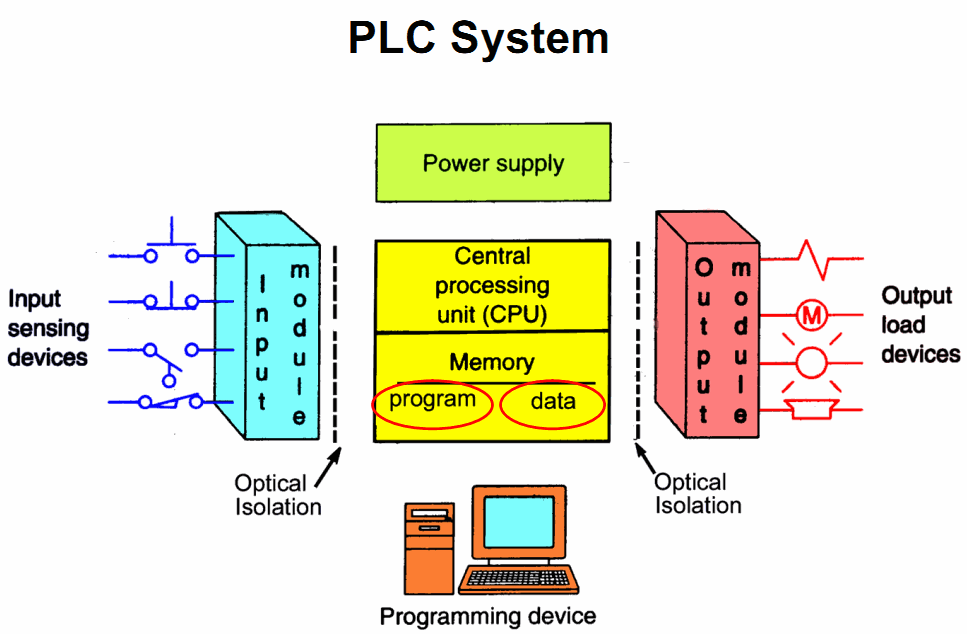
***Figure 16***

* This is probably a good time to practice your dragNdrop skills. Try moving instructions from rung to rung by holding the left mouse button down while over an instruction, and then while keeping the mouse button down, move the mouse (and instruction) to a new location. Try doing the same with complete rungs by dragging the box at the left end of the rung and dropping it in a new location.
* Once you feel comfortable with dragNdrop, ensure that your program once again looks like the one pictured above, Now download your program to the PLC and place the PLC into the Run Mode. Toggle both Switch-0 and Switch-1 on and off a number of times and observe the effects this has on the lamps. Ensure that you are satisfied with the operation of your program before proceeding further.

3.15 PLC as a System Controller

* Programmable logic controllers are modular, industrially hardened computers which perform control functions through modular input and output (I/O) modules. The modularity of PLC allows the user to combine generic I/O modules with a suitable controller to form a control system specific to his is most simply understood needs. The operation of a controller by envisioning that it repeatedly performs three steps:

1. Reads inputs from input modules
2. Solves preprogrammed control logic
3. Generates outputs to output modules based on the control logic solutions. Input devices and output devices of the process are connected to the PLC and the control program is entered into the PLC memory (Fig17).



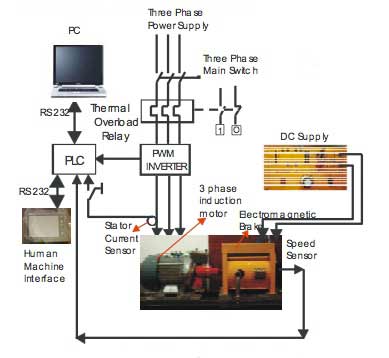
***Figure 17: Control Action of a PLC***

* In our application, it controls through analog and digital inputs and outputs the varying load-constant speed operation of an induction motor.
* Also, the PLC continuously monitors the inputs and activates the outputs according to the control program. This PLC system is of modular type composed of specific hardware building blocks (modules), which plug directly into a proprietary bus: a central processor unit (CPU), a power supply unit, input-output modules I/O and a program terminal.
* Such a modular approach has the advantage that the initial configuration can be expanded for other future applications such as multi machine systems or computer linking.

Cheptor IV

4.1 Control System of BLDCM

* The software models generated in the Software Requirements Analysis phase of the development project are refined and embellished in the design phase of the project. This phase involves making implementation decisions such as the interfacing between different software modules, the breakdown of software across multiple processors, assigning inputs and outputs to I/O cards, etc. PLC software, once written must be easy and intuitive to follow.
* PLCs are an integrated part of the domain system, advances in the technology of the system will effect the requirements of the PLC software. PLC software must therefore be maintainable and extensible.

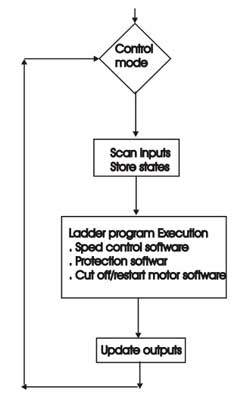


***Figure 18 : Experimental setup***

* In Fig. 18, the block diagram of the experimental system is illustrated. The following configurations can be obtained from this setup.

1. A closed-loop control system for constant speed operation, configured with speed feedback. The induction motor drives a variable load, is fed by an inverter and the PLC controls the inverter output.
2. An open-loop control system for variable speed and variable frequency operation. The induction motor drives a variable load and is fed by control mode. The PLC is an inverter in constant in activated.
3. The standard variable speed operation. The induction motor drives a variable load and is fed by a constant voltage-constant frequency standard three-phase supply.

* The open-loop configuration (2.) can be obtained from the closed-loop configuration (1.) by removing the speed feedback. On the other hand, operation c) results if the entire control system is bypassed. PLC’s programming is based on the logic demands of input devices and the programs implemented are predominantly logical rather than numerical computational algorithms. Most of the programmed operations work on a straightforward two-state “on or off” basis and these alternate possibilities correspond to “true or false” (logical form) and“ 1 or 0” (binary form), respectively. Thus, PLCs offer a flexible programmable alternative to electrical circuit relay-based control systems built using analog devices.
* The programming method used is the ladder diagram method. The PLC system provides a design environment in the form of software tools running on a host computer terminal which allows ladder diagrams to be developed, verified, tested, and diagnosed. First, the high-level program is written in ladder diagrams. Then, the ladder diagram is converted into binary instruction codes so that they can be stored in random-access memory (RAM) or erasable programmable read-only memory (EPROM). Each successive instruction is decoded and executed by the CPU. The function of the CPU is to control the operation of memory and I/O devices and to process data according to the program. Each input and output connection point on a PLC has an address used to identify the I/O bit. The method for the direct representation of data associated with the inputs, outputs, and memory is based on the fact that the PLC memory is organized into three regions: input image memory (I), output image memory (Q), and internal memory (M).

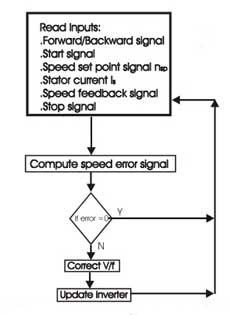


***Figure 19: Flowchart of the main program***

* Any memory location is referenced directly using %I, %Q, and %M (Table III). The PLC program uses a cyclic scan in the main program loop such that periodic checks are made to the input variables (Fig.3). The program loop starts by scanning the inputs to the system and storing their states in fixed memory locations (input image memory I).
* The ladder program is then executed rung-by-rung. Scanning the program and solving the logic of the various ladder rungs determine the output states. The updated output states are stored in fixed memory locations (output image memory Q). The output values held in memory are then used to set and reset the physical outputs of the PLC simultaneously at the end of the program scan. For the given PLC, the time taken to complete one cycle or the scan time is 0, 18 ms/K (for 1000 steps) and with a maximum Program capacity of 1000 steps.
* The development system comprises a host computer (PC) connected via an RS232 port to the target PLC. The host computer provides the software environment to perform file editing, storage, printing, and program operation monitoring.
* The process of developing the program to run on the PLC consists of: using an editor to draw the source ladder program, converting the source program to binary object code which will run on the PLC’ s microprocessor and downloading the object code from the PC to the PLC system via the serial communication port.
* The PLC system is online when it is in active control of the machine and monitors any data to check for correct operation.

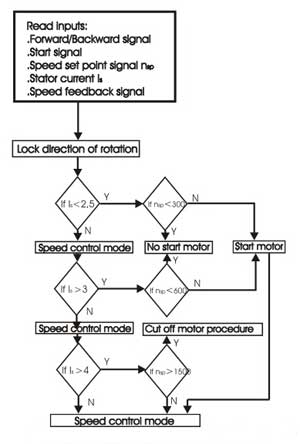
**As a microprocessor-based system, the PLC system hardware is designed and built up with the following modules :**

1. Central processor unit (CPU)
2. Discrete output module (DOM)
3. Discrete input module (DIM)
4. Analog outputs module (AOM)
5. Analog inputs module (AIM)
6. Power supply



***Figure 20: Flowchart of speed control software***

* In Fig.7, the flowchart of the speed control software is illustrated. The software regulates the speed and monitors the constant speed control regardless of torque variation. The inverter being the power supply for the motor executes this while, at the same time, it is controlled by PLC’ s software. The inverter alone cannot keep the speed constant without the control loop with feedback and PLC. From the control panel, the operator selects the speed set point nsp and forward/backward direction of rotation. Then, by pushing the manual start pushbutton, the motor begins the rotation. If the stop button is pushed, then the rotation stops.
* The corresponding input signals are interfaced to the DIM and the output signals to the DOM. The AIM receives the trip signal from the stator current sensor, the speed feedback signal from the tacho-generator, and the signal from the control panel. In this way, the PLC reads the requested speed and the actual speed of the motor. The difference between the requested speed by the operator and the actual speed of the motor gives the error signal. If the error signal is not zero, but positive or negative, then the PLC according to the computations carried out by the CPU decreases or increases the V/f of the inverter and, as a result, the speed of the motor is corrected.
* The implemented control is of proportional and integral (PI) type (i.e., the error signal is multiplied by gain Kp, integrated and added to the requested speed). As a result, the control signal is sent to the DOM and connected to the digital input of the inverter to control V/f variations. At the beginning, the operator selects the gain Kp, by using a rotary resistor mounted on the control panel (gain adjust) and the AIM receives its voltage drop as controller gain signal (0–10 V). The requested speed nsp is selected using a rotary resistor and the AIM reads this signal. Its value is sent to the AOM and displayed at the control panel (speed set point display).

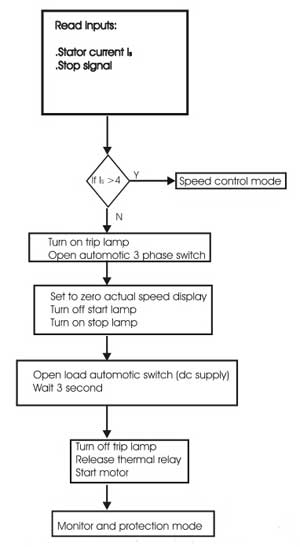


***Figure 21: Flowchart of monitor and protection Software***

* Another display of the control panel shows the actual speed computed from the speed feedback signal. In Fig. 21, the flowchart of this software is shown.
* During motor operation, it is not possible to reverse its direction of rotation by changing the switch position. Before direction reversal, the stop button must be pushed. For motor protection against overloading currents during starting and loading, the following commands were programmed into the software.

1. Forward/backward signal is input to DIM
2. Speed set point signal nsp, the stator current IS and the speed feedback signal are input to AIM.
3. At no load, Is ≤ 2,5 A, if the speed set point is lower than 20% or nsp< 300 r/min, the motor will not start.
4. At an increased load over 3,2 Nm (40% of rated torque), Is ≥ ,3 A and a speed set point lower than 40% or nsp< 600 r/min, the motor will not start.
5. If the load is increased more than 8,0Nm (rated torque) Is ≥ 4 A and if the speed set point exceeds 100% or nsp≥ 1500 r/min, the motor enters the cut off procedure.
6. In all other situations, the motor enters in the speed control mode and the speed control software is executed as described in Subsection A.

* In Fig. 22, the flowchart of this software is shown.



***Figure 22: Flowchart of cutoff/restart motor software***

1. In overloading situations, the motor is cut off and the trip lamp (yellow) is lit. The operator must release the thermal relays and then must turn off the trip lamp by pushing trip or stop button. The thermal relays are set to the motor rated current 4 A. Following this, the motor can be started again.
2. The motor can be cut off by the operator pushing the stop button: the display of the actual speed is set to zero, the start lamp (green) turns off, and the stop lamp (red) turns on and remains lit for 3 s.
3. The load must be disconnected immediately after the motor cuts off and before the drive system is restarted. The motor will not start before 3 s after cutoff even if the start button is pushed.

Conclusion

* It is clear that PLCs are fully equipped control systems. PLC, is used the maximum speed was 1550 rpm at average voltage of 15V. The on and off delays from the driving transistor of the PLC made the operation to consume more current and reduces the speed because of the on/off delay. Faster PLCs are recommended for better performance when a BLDCM needs to be controlled such as DVP ES/EX Series from Delta Corporation. Nevertheless the T116-S can be fairly used also for speed operations up to 1500 rpm with switching frequency of 2 KHz for insensitive applications.