A MAJOR PROJECT REPORT

ON

**PLC HYBRID SYSTEM FOR SCADA**

Submitted for the partial fulfillment of degree

***M.E. IN CONTROL & INSTRUMENTATION***

****

Under the guidance of

***PROF. PARMOD KUMAR***

Department of Electrical Engineering

Delhi College of Engineering

By:

Karan Singh

**Rollno.15/C&I/2K8**

**CERTIFICATE**

This is to certify that the project work that is being presented in this dissertation entitled **“PLC HYBRID SYSTEM FOR SCADA”** has been carried out by “KARAN SINGH” **University Roll No 13963 a** student of M. E. (C & I), Delhi College of Engineering, University of Delhi. This work has been completed and carried out by him under my supervision and forms a part of M. E. (Control & Instrumentation) program. I convey my best wishes to him.

**PROFESSOR PARMOD KUMAR**

Department of Electrical Engineering,

Faculty of Technology

Delhi University

**ACKNOWLEDGMENT**

My heartfelt thanks to my mentor and guide **Professor Parmod Kumar,** Department of Electrical Engineering, Faculty of Technology, Delhi University , who provided me an opportunity to work under his able guidance . His keen interest and suggestions were the source of inspiration for the successful completion of the project. I also convey my sincere thank to all the faculty members of electrical engineering department and administration of erstwhile DCE /DTU for providing me opportunity to pursued this program.

I would like to avail this opportunity to humbly thank and appreciate the contribution and support by my family and all staff members, for the successful completion of this dissertation, specially to Mr. Bharat Kashyap , Department of Electrical Engineering.

**KARAN SINGH**

**Roll No. : 15/C&I/08**

**University Roll No. : 13963**

**M. E. (control & instrumentation),**

**EED, DCE, University of Delhi**

**CONTENTS**

**CHAPTER-I: - INTRODUCTION**

1.1 Introduction

1.2 PLCs versus Relay types of Control

1.3 PLC-features

1.4 Distributed system

1.5 Plc Hybrid System

1.6 Dissection of the thesis

1.7 Conclusion

**CHAPTER-II:-LITERATURE REVIEW**

*2.1* Introduction

2.2 Programmable Logic Controller

2.3 The conceptual design of the PLC

2.4 Advanced Programmable Controllers

2.5 Lab-VIEW:

2.6 Data Acquisition

2.7 Conclusion

**CHAPTER-III:-FUNCTIONs AND MODELLING OF PLC**

3.1 Introduction

3.2 PLC Functions

3.3 Control Functions

3.4 Sequence Functions

3.5 Instantaneous Functions

3.6 Memory Functions

3.7Ttypes of PLCs

3.8 Selection of PLCs

3.9 Input/output Considerations

3.10 Distributed Control

3.11 Memory Considerations

3.12 Software Considerations

3.13 Peripherals

3.14 Physical and Environmental

3.15 PLC Reliability

3.16 Modeling of PLC

3.17 Conclusions

**CHAPTER-IV:-PLC PROGRAMMING AND IMPLIMENTATION**

**GUIDE LINES**

4.1 Introduction

4.2 PLC language

4.3 Scan Evaluation

4.4 PLC Timers and Counters.

4.5 Timer Instructions

4.6 counter instructions

4.7 Boolean Mnemonics

4.8 Implementation guidelines

4.9 Languages and Instructions

4.10 Programming Language Notation

4.11 Sub-Programs

4.12 Types of step action

4.13 IEC 1131-3 Software Systems

4.14 PLC Languages similar to the IEC 1131-3

PLC Programming

SECTION

3

CHAPTER

10

TheIEC1131Standardand

Programming Language

CHAPTER

10

TheIEC1131Standardand

Programming Language

PLC Programming

SECTION

3

4.15 Conclusion

**CHAPTER-V:-PLC – HYBRIDE APPLICATIONS**

5.1 Introduction

5.2 Linear Variable Differential Transformer

5.3 Flash Light

5.4 Setup for Programmable Logic Controller (vplct-03):

5.5 PCLD 8710 screw terminal board

5.6 Wiring cable PCL-10168

5.7 PCI-1711 multi function card

5.8 PLC-Hybrid System for SCADA

5.9 Development of ladder program

5.10 Conclusion

**CHAPTER-VI:-DISCUTIONS, CONCLUSITIONS, AND**

**FURTHER SCOPE OF WORK.**

6.1 Discussion

6.2 Conclusion

6.3 Further Scope of Work

6.4 Global Database

6.5 Knowledge Database

6.6 Knowledge Representation

6.7 Rule-Based Knowledge Representation

6.8 Fuzzy Logic and PLCs

**References:**

**CHAPTER- I**

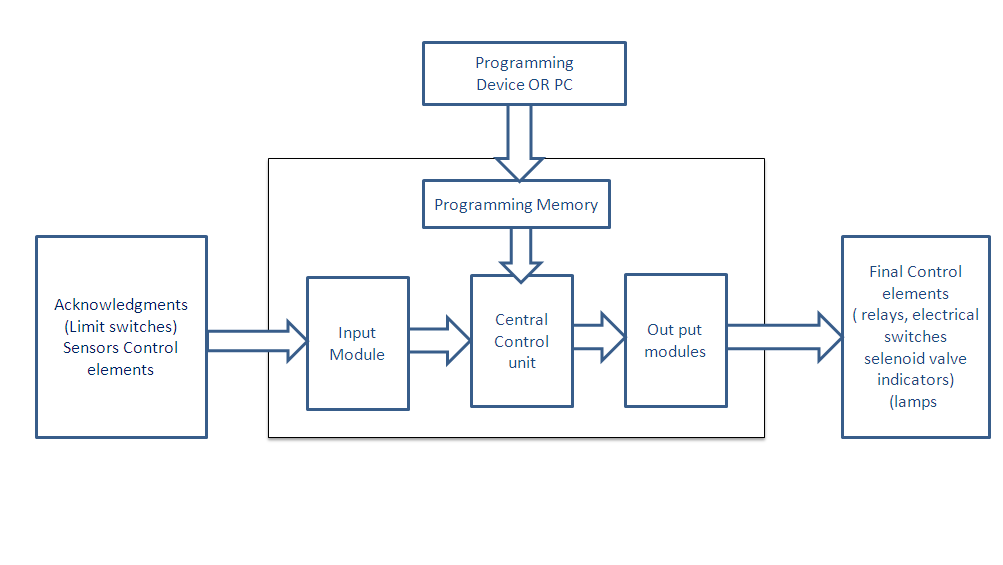
**INTRODUCTION**

**INTRODUCTION:**

With the price reduction of hardware, advancement in technological software and communication buses, the distributed SCADA system has evolved around PLC and microcontrollers. These are intelligent devices and provide a greater reliability for redundancy and fail-safe-operation to the plant and system. These devices also have the networking capabilities as well as more user friendly programming. Thus, the PLC-microcontroller-computer hybrid system forms a distributed processing system and extends all the features and functions of a distributed processing system. A simplified block diagram of PLC hybrid system for SCADA is presented in figure 1.1.Such hybrid systems are commonly useful to the process industries such as paper mills, textile mills, steel plants and cement plants, etc. It can continuously monitor the quality of product, energy consumption etc. and optimize the energy consumption as well as improve the product quality by operating the process plant, drives and equipment at critical or desired operating point.

The capabilities of programmable logic controllers (PLC) and distributed control system (DCS) have changed to the extent that today many applications that used to be the exclusive province of one or the other can be handled by both. PLCs were developed by manufacturers who had been making relays for logic and interlock applications, while DCS system were developed by process control manufacturers, having substantial experience in PID-type analog control. Therefore, in the past it made good sense to use each type of controller in its area of superior experience. If the bulk of the I/O was digital, the logical choice was to use a PLC, whereas if the I/O was mostly analog, a DCS system was selected. This logic, while still valid to some extent, is no longer universally true, and personal preference and end user familiarity has become a decisive factor in system selection.

Programmable logic controller’s I/O is likely to be more rugged and PLCs are likely to handle discreet logic faster than DCS systems. PLCs are also likely to be more desirable because their languages, such as ladder logic, are usually more familiar to plant personnel and, therefore, there is less resistance to using them. On the other hand, ladder logic type language can be undesirable in some situations because they are not well suited to analog process control. Some users have overcome the limitations of PLCs by coupling them to personal computers (PCs) using custom-code programming. The disadvantage of this approach is that such a non-standardized system is usually understood fully only by its designer, and when that person leaves the company, the system can be ruined. When it comes to communication redundancy and data security, the DCS systems are superior. The DCS systems are also superior in their programming library, in advanced or optimizing control, in self-tuning algorithms, and, particularly , in their total plant architecture and information management capabilities.



**Figure 1.1 Typical PLC- Hybrid system**

**1.2 PLCS VERSUS RELAY TYPES OF CONTROL:**

At one time, much of a system engineer’s time was spent trying to determine the cost-effectiveness of a PLC over relay control. Even today, many control system designers still think that they are faced with this decision. Today’s demand for high quality and productivity can hardly be fulfilled economically without electronic control equipment. With rapid technology developments and increasing competition, the cost of programmable controls has been driven down to the point where a PLC-versus-relay cost study is no longer necessary or valid. Programmable controller application scan now be evaluated on their own merits. If system requirements call for flexibility or future growth, a programmable controller brings returns that outweigh any initial cost advantage of a relay control system. Even in a case where no flexibility or future expansion is required, a large system can benefit tremendously from the trouble shooting and maintenance aids provided by a PLC. The extremely short cycle (scan) time of a PLC allows the productivity of machines that were previously under electromechanical control to increase considerably. Also, although relay control may costless initially, this advantage is lost if production down time due to failures is high.

**1.3 PLC FEATURES:**

It is a programmable micro-electronics device works on the principle of relays, control panels, however, the PLC provide the facilities of timer, counter besides the programming facility, also it can be re-programmed provided greater reliability and can communicate with microcontroller, computer and FPGA etc.. A typical PLC has:

* Processor,
* Memory,
* Power Supply,
* Programmable Device and
* Input /Output modules

**1.4 DISTRIBUTED SYSTEM:**

Increasingly computing addresses collaboration, data sharing, cycle sharing, and other modes of interaction that involve distributed resources form a distributed system. This trend results in an increased focus on the inter connection of systems both within and across enterprises. In addition, system designers have realized that they can achieve significant cost savings by such design. These evolutionary technologies improve the reliability for distributed application development and deployment. The continuing decentralization and distribution of software, hardware, and human resources make it essential that the desired **quality of service** (QOS) on resources assembled dynamically from process plant, system designer, and energy optimization is met despite the diversity.

Some of the properties of distributed system are as following:

1 Grid technologies enter the mainstream,

2 The evolution of enterprise computing,

3 Service providers and business-to-business computing,

1. Open grid services architecture,
2. Service orientation,
3. Virtualization,
4. Service semantics,
5. The grid service,
6. Standard interfaces,
7. Dynamic service creation,
8. Life time management,
9. Notification,
10. Manageability*.*

**1.5 PLC HYBRID SYSTEM:**

To design a PLC hybrid System there are basically two approaches model-based and non-model-based. Model-based methods are synthesis and verification. A combination of both is possible. The synthesis approach is the one most closely related to Control Theory. With a model of the PLC System. The combination of these System results in a System of the controlled process. Using these System formal specifications can be verified mathematically. Non-model-based approaches stem from the area of Software Engineering. They measure or verify certain properties of a piece of software under minimal or no assumptions about its environment. This approach is well suited for logic controllers because the realization of a logic controller includes hard- and software. But under the assumption of standard hardware with well-defined functionality, the application is realized by the program of the control algorithm i.e. software. Hence the quality and functionality of the controller depends mainly on the quality and functionality of that software. Non-model-based analysis methods for purely discreet logic control algorithms based on Signal Interpreted Petri Nets (SIPN). They include correctness analysis and the measurement of transparency.

**1.6** **DISSECTION OF THE THESIS:**

Chapter -II presents a brief literature review on the PLC, Ethernet bus and related buses, and other related areas. Modeling and functions of PLC are discussed in the chapter- III. Chapter -IV describes the PLC programming and implementation guide lines, language, instructions , timers and counters, programming languages and , their notation, and instructions, scan evaluation instructions, divergences and convergences, Boolean mnemonics, sub-programs, types of step action, IEC 1131-3 software systems and PLCs languages similar to the IEC 1131-3.

Chapter-V presents the PLC hybrid system design and criteria to be considered, and also gives the detail of PLC – hybrid applications implementation. In the present work two applications have considered (i) Linear Variable Differential Transformer for distance measurement, and (ii) Flash Light for voltage control. Chapter-VI Presents discussions, conclusions and further scope of work.

**1.7 CONCLUSIONS:**

This chapter brings out the objective of the project work, introduction to PLC and Distributed system. Also, some important features of PLC hybrid system is presented in this chapter. An abstract of subsequent chapters has been also presented.

**CHAPTER-II**

**LITERATURE REVIEW**

**INTRODUCTION:**

In this chapter a brief literature review on PLC, distributed system and associated areas have been presented without exhaustive explanation and discussions. PLC is a sequential logic controller whereas distributed control is general purpose controller. Each has some advantages as-well-as disadvantages. By integrating and fusion of their functions the advantages of two can be achieved. Such system is named PLC hybrid system.

The control of machines and plants was built with relay circuits. In this process the contacts were connected in series or parallel to generate the desired control. Mainly binary and Boolean signals were processed. The basic component of logic control system gates where as for sequencing operation timer is an essential component. Such relay based control and sequencing is a hardware logic and needs redesigning of relays based system when the process plant configuration changes and also a shutdown0 of the plant is required. This reduces the reliability and productivity of the plant. A programmable controller which provides the flexibility and communication between the plant and the operator has become the needful requirement in alternative to relay based control system.

The term "Programmable control" was developed first in the late 1960s. The primary region of designing such a device was to eliminate the large cost and is replacing the complicated relay based machine control systems for US car manufacturers. Some company proposed scream based on DEC-PDP-8 minicomputer at the time. DICK MORLAY’s Company Bedford developed the first PLC car, a modular digital controller (MODICON).

In the year 1968 General Electric Motors came up with the idea of programmable logic controllers (AHK) the PLC. The company looked for other possibilities than the hard-wired programmed connected relay, contactor and electronic controls. The control should be reprogrammable for changes very quickly, for example, if the machines are to be rebuilt. The costs for developing a control system should be more competitive than those of the other known types of control.

In mid 1970s the dominant PLC technology was sequencer state machine and the bit & slice based CPU. The conventional microprocessor lacks the power to quickly solve the PLC logic. In 1973 the communication abilities were in corporative and the first such system is known as MODICON MODBUS. The PLC could now communicate to other PLCs and given the birth of distributed PLC system. In 1980s the communication protocol was standardized known as Automation protocol. This facilitated to reduce the size of PLC and making them software programmable through symbolic program on PC instated of dedicated programming terminal.

The 1990 has seen a gradual reduction on developing the new protocol and modernization of physical layers. The latest standard IEC-1131-3 has standardized the PLC programming languages. Also, the control should work more reliable. The requirements of the "logic controllers" could be met by the advancing technology. After they had first worked primarily with binary signals, the producers developed methods for times, numbers or floating-point processing. Thus, it was possible to work with analogue data.

**2.2 PROGRAMABLE LOGIC CONTROLLER (PLC):**

The Programmable Logic Controller (PLC) is a solid state device designed to perform logic function accomplished by electromechanical relays used in industrial equipment. PLC differs in programming as well as in control. The sequence control can be divided in two categories:

(i) Logic control and

(ii) Sequencing control

Both types operate on variables that can take either of two values (e.g. on and off) which are referred to as switching systems in the sense that they are switching their output values on and off during operation. The logic control system, also referred to in the literature as a combinational system is a switching system whose output at any moment is determined by the values of the inputs. A logic control system has no memory and does not consider any previous values of input signals in determining the output signal. Neither does it have any operating characteristics which perform as a function of time. Suppose a robot is programmed to pick up a work part from a point along a conveyer and place it into a forging die. Three conditions must be satisfied to initiate the loading cycle. First, the work part must be at the stopping point; second, forge press must have completed the process on the previous part; third, the previous part must be removed from the die. The condition can be indicated by means of a simple limit switch that senses the presence of the part at the conveyor stop, and transmits an on signal to the robot controller. Second condition can be indicated by the forge press. This sends an on signal after it has completed the previous cycle. The third condition must be determined by a photo detector designed to sense the presence or absence of the part in the forging die. When the finished part is removed from the die, an on signal is transmitted by the photocell. All three of these on signals must be received by the robot controller in order to initiate the work cycle. Although we have referred to these in coming signals in robotics as input interlocks, they also illustrate logic control. When these signals have been received by the controller, the robot loading cycle is switched on. No previous conditions are needed.

A sequencing system is one that uses internal timing devices to determine when to initiate changes in output variables. Washing machine, dryers, dishwashers and similar appliances use sequencing systems to time the start and stop of cycle elements. There are many industrial application of sequencing systems. Suppose an induction heating coil is used to heat a part to the desired temperature in our previous example of a robotics forging application. An induction heating system is uses a high energy source focused on an object to heat it. Rather than use a temperature sensor that might be damaged by the induction coil, the heat cycle could be timed so that enough energy was provided to heat the work part to the desired temperature. The heating process is sufficiently predictable that certain duration of time in the induction coil will consistently heat the part to a certain temperature on a minimum variation. The first programmable controller in 1968 was developed by the General Motor Corporation. It replaced the relay controlled systems. Such a control system would reduce machine down time and provide expandability for the future. Some of the initial specifications included by the General Motor are:

• The new control system had to be price competitive with the use of relay systems.

• The system had to be capable of sustaining an industrial environment.

• The controller had to be designed in modular form; so that -subassemblies could be removed easily for replacement or repair.

• The control system needed the capability to pass data collection to a central system.

• The method used to program the controller had to be simple, so that it could be easily understood by plant personnel.

The product implementation to satisfy Hydramatic’s specifications was underway in 1968; and by 1969, the programmable controller had its first product off spring. These early controllers met the original specifications and opened the door to the development of a new control technology. The first PLCs offered relay functionality, thus replacing the original hardwired **relay logic**, which used electrically operated devices to mechanically switch electrical circuits. They met the requirements of modularity, expandability, programmability, and ease of use in an industrial environment. These controllers were easily installed, used less space, and were reusable. The controller programming, although a little tedious, had a recognizable plant standard: the ladder diagram format. In a short period, programmable controller use started to spread to other industries. By1971, PLCs were being used to provide relay replacement as the first steps toward control automation in other industries, such as food and beverage, metals, manufacturing, and pulp and paper.

**2.3 THE CONCEPTUAL DESIGN OF THE PLC:**

The first programmable controllers were more or less just relay replacers. Their primary function was to perform the sequential operations that were previously implemented with relays. These operations included ON/OFF control of machines and processes that required repetitive operations, such as transfer lines and grinding and boring machines. However, these programmable controllers were a vast improvement to the relays. They were easily installed, used considerably less space and energy, had diagnostic indicators that aided troubleshooting, and unlike relays, were reusable if a project was scrapped. Programmable controllers can be considered new comers when they are compared to their elder predecessors in traditional control equipment technology, such as old hardwired relay systems, analog instrumentation, and other types of early solid-state logic. Although PLC functions, such as speed of operation, types of interfaces, and data-processing capabilities, have improved throughout the years, their specifications still hold to the designers’ original intentions they are simple to use and maintain.

Many technological advances in the programmable controller industry continue today. These advances not only affect programmable controller design, but also the philosophical approach to control system architecture. Changes include both **hardware** (physical components) and **software** (control program) upgrades. The recent PLC hardware enhancements faster scan times are being achieved using new, advanced micro- processor and electronic technology. Small, low-cost PLCs, which can replace four to ten relays, now have more power than their predecessor, the simple relay replacer. High density input/output (I/O) systems provide space-efficient interfaces at low cost. Intelligent, microprocessor-based I/O interfaces have expanded distributed processing. Typical interfaces include PID (proportional-integral-derivative), network, can bus, field bus, ASCII communication, positioning, host computer, and language module. Mechanical design improvements have included drugged input/output enclosures and input/output systems that have made the terminal an integral unit. Special interfaces have allowed certain devices to be connected directly to the controller. Typical interfaces include thermocouples, strain gauges, and fast-response inputs. Peripheral equipment has improved operator interface techniques, and system documentation is now a standard part of the system. All of these hardware enhancements have led to the development of programmable controller families. These families consist of a product line that ranges from very small “microcontrollers,” with as few as 10 I/O points, to very large and sophisticated PLCs, with as many as 8,000I/ Opointsand128,000 words of memory. These family members, using common I/O systems and programming peripherals, can interface to a local communication network. The family concept is an important cost-saving development for users. Like hardware advances, software advances, such as the ones listed below, have led to more powerful PLCs. PLCs have in corporate object-oriented programming tools and multiple languages based on the IEC 1131-3 standard. Small PLCs have been provided with powerful instructions, which extend the area of application for these small controllers. High-level languages, such as BASIC and C, have been implemented in some controllers’ modules to provide greater programming flexibility when communicating with peripheral devices and manipulating data. Advanced functional block instructions have been implemented for ladder diagram instruction sets to provide enhanced software capability using simple programming commands. Diagnostics and fault detection have been expanded from simple system diagnostics, which diagnose controller malfunctions, to include machine diagnostics, which diagnose failures or malfunctions of the controlled machine or process. Floating-point math has made it possible to perform complex calculations in control applications that require gauging, balancing, and statistical computation. Data handling and manipulation instructions have been improved and simplified to accommodate complex control and data acquisition applications that involve storage, tracking, and retrieval of large amounts of data. Programmable controllers are now offering many more capabilities than were ever anticipated. They are capable of communicating with other control systems, providing production reports, scheduling production, and diagnosing their own failures and those of the machine or process. These enhancements have made programmable controllers important contributors in meeting today’s demands for higher quality and productivity. Despite the fact that programmable controller shave become much more sophisticated, they still retain the simplicity and ease of operation that was intended in their original design.

**2.4 ADVANCED PROGRAMMABLE CONTROLLERS:**

The advanced programmable controller relies not only on the continuation of new product developments, but also on the integration of PLCs with other control and factory management equipment. PLCs are being incorporated, through networks, into computer-integrated manufacturing (CIM) systems, combining their power and resources with numerical controls, robots, CAD/CAM systems, personal computers, management information systems, and hierarchical computer-based systems. New advances in PLC technology include features such as better operator interfaces, graphic user interfaces (GUIs), and more human-oriented man/machine interface (such as voice modules). They also include the development of interfaces that allow communication with equipment, hardware, and software that supports artificial intelligence, such as fuzzy logic I/O systems. Software advances provide better connections between different types of equipment, using communication standards through widely used networks. New PLC instructions are developed out of the need to add intelligence to a controller. Knowledge-based and process learning type instructions may be introduced to enhance the capabilities of a system.

The user’s concept of the flexible manufacturing system (FMS) will determine the control philosophy of the future. The future will almost certainly continue to cast programmable controllers as an important player in the factory. Control strategies will be distributed with “intelligence” instead of being centralized. Super PLCs will be used in applications requiring complex calculations, network communication, and supervision of smaller PLCs and machine controllers.

**Programmable logic controllers**, also called *programmable controllers* or *PLCs*, are **solid-state** member soft the computer family, using integrated circuits instead of electromechanical devices to implement control functions. They are capable of storing instructions, such as sequencing, timing, counting, arithmetic, data manipulation, and communication, to control industrial machines and processes. Figure 2.1 illustrates a conceptual diagram of a PLC application.

Process

Or

Machine

Measure Control

Programmable

Field Controller Field

Inputs outputs

**Figure 2.1 PLC conceptual application diagram**.

Programmable logic controllers are mature industrial controllers with their design roots based on the principles of simplicity and practical application. These components are enclosed in a cabinet of industrial standard which takes into account the temperature, vibration and dusted workshop/ process plant into consideration. As far as the PLC users are concerned, the program steps defined by the ladder logic diagram are executed simultaneously and continuously. In real, a certain amount of time is required for the PLC processor to step through the program and execute any change in the outputs. The inputs to the PLC are sampled by the processor and the contents are stored in the memory. Then the control program is executed. The input values stored in memory are used in the control logic calculations to determine the values of the outputs.

Therefore, the outputs are updated to agree with the calculated values. This cycle, consisting of reading the inputs, executing the control program, and revising the outputs, is referred to as a scan. The time to perform the scan is called the scan time, and this depends on the number and complexity of control functions to be performed each cycle. Another way, Stating this scan time depends on the number of rungs in the ladder diagram and the complexity of the logic operations to be carried out on each rung. These times typically vary from 1ms to 100 ms. One of the potential problems that can occur during the scan cycle is that the value of an input can change immediately after it has been sampled. Since the program uses the input value that is stored in memory, any output value that is dependent on that input is determined incorrectly. There is a potential risk involved in this mode of operation. However, the risk is minimized because the time between updates is so short that is unlikely that the output value being incorrect for such a short duration will have a serious effect on the process operation. The risk becomes most significant in process in which the response times are very fast, and where hazards can occur during the scan time. Some PLCs have special features for making “immediate” updates of output signals when input variables are known to cycle back and forth at frequencies faster than the scan time. The logic control and sequencing control elements are perhaps the principal control operations that are accomplished by the PLC. The logic control and sequencing control were the functions for which the programmable controller was originally designed. However, the PLC has evolved to include several capabilities in addition to logic control and sequencing. Some of the important capabilities available on commercial PLCs are:

**2.5 Lab-VIEW:**

Conventional methods of controlling a physical variable use hardwired controllers for the purpose, such as op-amps, resistors, pneumatic relays, hydraulic controllers etc. These components were used to make controllers, which makes the work of the controller designer cumbersome, as one has to interface the controller and then to develop the same through different components either by electrical means or pneumatic or hydraulic means depending upon the power requirements and the surrounding environment also. Other than that the traditional instruments are vendor defined and are susceptible to the environmental disturbances such as temperature variation and humidity variation which can cause to mal-function the controller and the output of the controller and hence the plant can be erroneous. For example the output of an op-amp varies with the input offset current which is a function of the temperature. Lab-view is a graphical programming technique, in which the designer only has to focus and use his gray cells to develop the new techniques of controlling and enhance the efficiency of the system and also to provide much more freedom and ease to use for the end user, which an ordinary hardwired controller can't provide. Moreover, the plant operator has various options and he can monitor the variable remotely on the computer screen i.e. on the front panel of the controller. Other than that there is no effect of the surroundings on the virtual instruments as there are no hardware components or very less hardware components are used. In the present project the operator can change the set point; monitor the level of the prototype model and in emergency situation can raise an alarm and stop the functioning of the plant, the functions which greatly increase the reliability and efficiency of the plant.

1. **Hardware:** Hardware consists of the signal conditioner, MIC (condenser) ; PCLD-8710 screw terminal board; wiring cable PCL -10168 and DAQ card/PCI-1711 multifunction board.
2. **Software:** A graphical programming based software LABVIEW is used in this project, DLL drivers for the DAQ card.

**2.6 DATA ACQUISITION:**

**Data Acquisition Device:**

DAQ hardware is what usually interfaces between the signal and a PC. It could be in the form of modules that can be connected to the computer’s ports( i.e. Parallel ports, serial port, USB port, etc.) or cards connected to slots (S-100 bus, Apple Bus , ISA, Micro channel architecture, PCI, PCI-E , etc.) in the Mother board. Usually the space on the back of a PCI card is too small for all the connections needed, so an external breakout box is required. The cable between this box and the PC can be expensive due to the many wires, and the required shielding. **Data Acquisition (**abbreviated **DAQ**) is the process of sampling of real world physical conditions and conversion s and conversion of the resulting samples into digital numeric values that can be manipulated by a computer. Data acquisition and data acquisition system (abbreviated as **DAS**) typically involves the conversion of analog waveforms into digital value for processing. The components of data acquisition system include:

* Signal conditioning circuitry to convert input signals into a form that can be acceptable to ADC card.
* Analog – to – digital converters, which convert conditioned input signals to digital values.

Main component of DAQ card is ADC (analog to digital convertor) which samples the analog signal on a regular basis and converts the amplitude at each sample time to a digital value with finite resolution. These are termed discrete time functions that are defined only at times specified by the sample rate and may have values determined by the resolution of ADC. In other words, when an analog signal is digitized, an approximation has to be made based on the signal and the specifications used for data analysis. Resolution also plays a very important role here. The additional amplitude and temporal resolution are the two important factors that are to be discussed, but they may be expensive. To overcome this, there should be a clear idea on sampling. The process of sampling is based on Nyquist Shannon’s sampling theorem.

Here in this case the electrical signal corresponding to the level in the tank is obtained with the help of the transducer which converts the level in the tank to the equivalent electrical signal. This electrical signal is fed to the computer through PCI-1711 Multi-function ADVANTECH DAQ card through PCLD-8710 screw terminal board (used when there is no sufficient space available for connections in multifunction card) and then the acquired signal is depicted on the front panel of virtual instrument in PC.

**2.7 CONCLUSIONS:**

A brief history of programmable logic controller development has been presented the literature review of plc started from the relay logic, the conceptual design and technological development also has been presented in the chapter. The Lab-VIEW brief and DATA Acquisition /SCADA have also briefly described.

**CHAPTER-III**

**FUNCTIONING AND MODELLING OF PLC**

**INTRODUCTION:**

The programmable logic controllers are available in all shapes and sizes, covering a wide spectrum of capabilities. On the low end are “relay replacers,” with minimum I/O and memory capability. At the high end are large supervisory controllers, which play an important role in hierarchical systems by performing a variety of control and data acquisition functions. In between these two extremes are multifunctional controllers with both communication capabilities, which allow integration with various peripherals, and expansion capabilities, which allow the product to grow as the application requirements change. Deciding a right controller for a given application has become increasingly more difficult. With the explosion of new products, including general and special purpose programmable controllers, system selection now places an even greater demand on the designer to take a system approach to selecting the best product for each task. Programmable controller selection affects many factors, so the designer must determine which characteristics are desirable in the control system and which controller best fits the present and future needs of the application. Prior to evaluating the system requirements, the designer should understand the different ranges of programmable controller products and the typical features found within each range. This understanding will enable the designer to quickly identify the type of product that comes closest to matching the application’s requirements.

**3.2 PLC FUNCTIONS:**

**ARITHMETIC FUNCTIONS**:

These are addition, subtraction, multiplication, and division functions. Use of these functions permits more complex control algorithms to be developed than what is possible with conventional logic and sequencing elements.

**MATRIX FUNCTIONS:**

Some of PLCs have the capability to perform matrix operations on stored values in memory. This capability can be used to compare the actual values of a set of inputs and outputs with the values stored in the PLC memory to determine if some error has occurred.

**ANALOG FUNCTIONS:**

The proportional-integral-derivative (PID) control is a available on some programmable logic controllers. These control algorithms have traditionally been implemented on analog controllers. Today the analog control schemes are approximated using the digital computer. Either with a PLC or a computer process controller. The approximation of PID control on a digital computer is called **direct digital control** **(DDC).**

**3.3 CONTROL FUNCTIONS:**

**C**ontrolling an automatic flow process is a typical problem, owing to the sheer Number of sequential steps that must be carried out. There are three main functions that are utilized to control the operation of an automatic flow system, which are given bellow:

(i) The operational requirement,

(ii) The safety requirement, and

(iii) Dedicated to improving quality.

**3.4 SEQUENCE FUNCTIONS:**

The purpose of this function is to coordinate the sequence of actions of the transfer system and its workstations. The various activities of the automated flow line must be carried out with split-second timing and accuracy. On a metal machining transfer line, for example the work parts must be transported, located, and clamped in place before the work-heads can begin to feed. Sequence control is basic to the operation of the flow line.

**SAFETY MONITORING:**

This Function ensures that the transfer system does not operate in an unsafe or hazardous condition. Sensing devices may be added to make certain that the cutting tool status is satisfactory to continue to process the work part in the case of a machining-type transfer line. Other checks might include monitoring certain critical steps in the sequence control function to make sure that these steps have all been performed and in the correct order. Hydraulic or air pressure might also be checked if these are crucial to the operation of automated flow lines.

**QUALITY MONITORING:**

The third control function is to monitor certain quality attributes of the work parts. Its purpose is to identify and possibly reject defective work parts and assemblies. The inspection devices required to perform quality monitoring are sometimes incorporated into existing processing stations. In other cases, separate stations are included in the line for the sole purpose of inspecting the work part.

It is possible to extend the notion of quality monitoring and to incorporate a control loop into the flow line. An inspection station would be used to monitor certain quality characteristics of the part and to feed back information to the preceding workstations so that adjustments in the process could be made.

The traditional means of controlling the sequence of steps on the transfer system has been to use electromechanical relays. Relays are employed to maintain the proper order of activating the work-heads, transfer mechanism, and other peripheral devices on the line. However, owing to their comparatively large size and relative unreliability, relays have lost ground to other control devices, such as programmable controllers and computers. These more modern components offer opportunities for a higher level of control over the flow line, particularly in the areas of safety monitoring and quality monitoring.

Conventional thinking on the control of the line has been to stop operation when a malfunction occurred. While there are certain malfunctions representing unsafe conditions that demand shutdown of the line, there are other situations where stoppage of the line is not required and perhaps not even desirable. For example, take the case of a feed mechanism on an automatic assembly machine that fails to feed its component. Assuming that the failures are random and infrequent, it may be better to continue to operate the machine and lock out the affected assembly from further operations. If the assembly machine were stopped, production would be lost at all other stations while the machine is down. Deciding whether it is better to stay in operation or stop the line must be based on the probabilities and economics of the particular case. The point is that there are alternative control strategies to choose between, instantaneous control and memory control.

**3.5 INSTANTANEOUS FUNCTIONS:**

This mode of control stops the operations of the flow line immediately when a malfunction is detected. It is relatively simple, inexpensive, and trouble-free. Diagnostic features are often added to the system to aid in identifying the location and cause of the trouble to the operator so that repairs can be quickly made. However, stopping the machine results in loss of production from the entire line, and this is the system’s biggest drawback.

**3.6 MEMORY FUNCTIONS:**

In contrast to instantaneous control, the memory system is designed to keep the machine operating. It works to control quality and/or protect the machine by preventing subsequent station from processing the particular work-part and by segregating the part as defective at the end of the line. The premise upon which memory-type control is based is that the failures which occur at the stations will be random and infrequent. If, however, the station Failures result from cause (a work head that has gone out of alignment, for example) and tend to repeat; the memory system will not improve production but, rather, degrade it. The flow line will continue to operate, with the consequence that bad parts will continue to be produced. For this reason, a counter is sometimes used so that if a failure occurs at the same stations for two or three consecutive cycles, the memory logic will cause the machine to stop for repairs.

**3.7 TYPES OF PLCS:**

PLC product ranges divided into four major areas with overlapping boundaries. The basis for this product segmentation is the number of possible inputs and outputs the system can accommodate (I/O count), the amount of memory available for the application program, and the system’s general hardware and software structure. As the I/O count increases, the complexity and cost of the system also increase. Similarly, as the system complexity increases, the memory capacity, variety of I/O modules, and capabilities of the instruction set increase as well.

**MICRO PLCS:**

Micro PLCs are used in applications that require the control of a few discrete I/O devices, such as small conveyor controls. Some micro PLCs can perform limited analog I/O monitoring functions (e.g., monitoring a temperature set point or activating an output). Figure 3.2 shows a typical microcontroller Table 2.1 lists the standard features of micro PLCs.



**Figure 3.1. micro PLC DL105**.

**Micro PLCs**

• Up to 32 I/O

• 16-bit processor

• Relay replacer

• Memory up to 1K

• Digital I/O

• Built-in I/Os in a compact unit

• Master control relays

• Timers and counters

• Programmed with handheld programmer

**Table 3.1. Standard features of micro PLCs**.

**SMALL PLCS:**

Small controllers are mostly used in applications that require ON/OFF control for logic sequencing and timing functions. These PLCs, along with microcontrollers, are widely used for the individual control of small machines. Often, these products are single-board controllers. Table 3.2 lists the standard features of small PLCs.

**Small PLCs**

• Up to 128 I/O

• 16-bit processor

• Relay replacer

• Memory up to 2K

• Digital I/O

• Local I/O only

• Ladder or Boolean language only

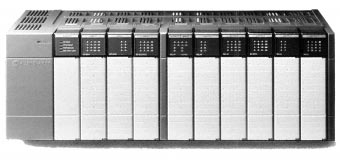
• Master control relays

• Timers/counters/shift registers

• Drum timers or sequencers

• Programmed with handheld programmer

**Table 3.2. Standard features of small PLCs.**

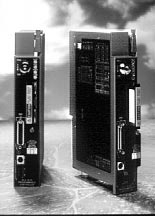
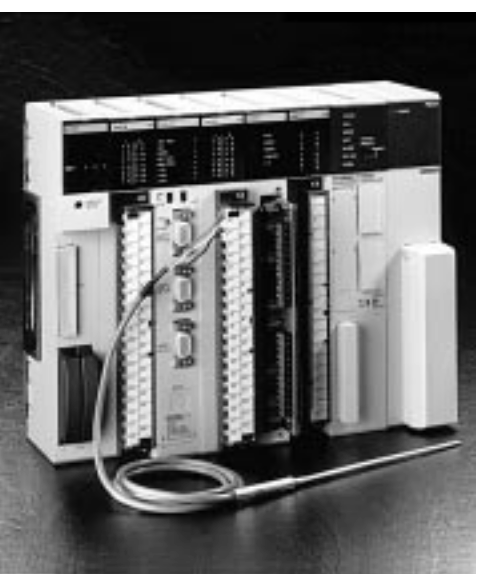


Courtesy of Allen-Bradley, Highland Heights, OH

**Figure 3.2 controller capable of handling up to 72 discrete and 4 analog I/O.**

**MEDIUM PLCS:**

Medium PLCs Figure 3.3 are used in applications that require more than 128 I/O, as well as analog control, data manipulation, and arithmetic capabilities. In general, the controllers in segment 3 have more flexible hardware and software features than the controllers previously mentioned. Table 3.3 lists these features.

**Figure 3.3 Medium PLCs controller**

**Medium PLCs**

• Up to 1024 I/O Math capabilities

• 16- or 32-bit processor Addition

• Relay replacer and analog control Subtraction

• Memory up to 4K words Multiplication

• Expandable to 16K Division

• Digital I/O Limited data handling

• Analog I/O Compare

• Local and remote I/O Data conversion

• Special function I/O modules

• Ladder or Boolean language

• Function block/high-level language

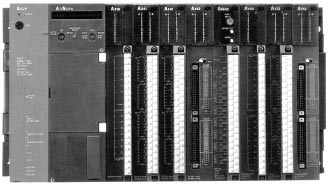
• Jump

• RS-232 communication port

• Timers/counters/shift registers

**Table 3.3. Standard features of medium PLCs.**

**LARGE PLCS:**

Large controllers in Figure 3.6 are used for more complicated control tasks, which require extensive data manipulation, data acquisition, and reporting. Further software enhancements allow these products to perform complex numerical computations. Table 3.4 summarizes the standard features of large PLC

**Figure 3.4 Large Mitsubishi A3NCPU controller with 2048 I/O capacity.**

|  |
| --- |
| **Large PLCs** |
| • Up to 4096 I/O  • 16- or 32-bit processor  • Relay replacer and analog control  • Memory up to 12K words  • Expandable to 128K  • Digital I/O  • Analog I/O  • Local and remote I/O  • Ladder or Boolean language  • Function block/high-level language  • Master control relays  • Timers/counters/shift registers  • Drum timers and sequencers  • Jump  • Subroutines, interrupts  • PID modules or system software PID  • One or more RS-232 communication ports  • Host computer communication modules |

**Table 3.4. Standard features of large PLCs.**

**3.8 SELECTION OF PLCS:**

PROCESS CONTROL SYSTEM D

Selecting the right programmable controller for a machine or process involves evaluating not only current needs, but future requirements as well. If present and future goals are not properly evaluated, the control system may quickly become inadequate and obsolete.

Keeping the future in mind when choosing a programmable controller will minimize the costs of changes and additions to the system. For example, with proper planning, future memory expansion may only require the installation of a memory module; furthermore, the addition of a peripheral may be as easy as connecting the device to the communication port. A local area network can also ease the future integration of programmable controllers into a plant wide communication scheme.

Once the basic control application has been defined, the user should begin evaluating the controller requirements, including:

**3.9 INPUT/OUTPUT CONSIDERATIONS:**

Determining the amount of I/O required is typically the first step in selecting a controller. Once the decision has been made to automate a machine or process, determining the amount of I/O is simply a matter of counting the discrete and/or analog devices that will be monitored or controlled. This count will help to identify the minimum size constraints for the controller. Remember that the controller should allow for future expansion and spares (typically10% to 20% spares), although spares do not affect the choice of PLC size.

**Discrete Inputs/Outputs:**

Input/output interfaces with standard ratings are available for accepting signals from sensors and switches (e.g., push buttons, limit switches, etc.), as well as ON/OFF control devices (e.g., pilot lights, alarms, motor starters, etc.). If these input/output devices receive power from separate sources, then the discrete interface circuits must have isolated commons (return lines). Typical discrete AC inputs/outputs range from 24 to240 V, and typical DC inputs/outputs range from 5 to 240 V. Input circuits vary from one manufacturer to another. Nevertheless, characteristics like de-bouncing circuitry, which protects against false signals, and surge protection, which guards against large transients, are desirable in any input circuit. Another good input circuit quality is optical or transformer isolation between the high-power input and the interface’s control logic circuitry. When evaluating discrete outputs, the following are key characteristics: fuses, transient surge protection, and isolation between the power and logic circuits. Fused circuits cost more initially, but they usually cost less than having a fuse installed externally. These circuits should also have easily accessible fuses, so that replacing fuses does not require shutting down several other devices for a long period of time. Moreover, fused output circuits should have blown fuse indicators, as well as an output current rating and a specified operating temperature (typically 60 F) that fits the application’s requirements.

**Analog Inputs/Outputs:**

Analog input/output interfaces sense signals generated by transducers. These interfaces measure quantity values, such as flow, temperature, and pressure, and are used to control voltage or current output devices. Typical interface ratings include –10 to +10 V, 0 to +10 V, 4 to 20 mA, and 10 to 50 mA. Some manufacturers provide special analog interfaces that accept low-level signals (e.g., RTD, thermocouple). Typically, these interface modules accept a mix of thermocouple or RTD signals on a single module. Users should consult the vendor concerning specific requirements.

**Special Function Inputs/Outputs:**

Sometimes an application requires a special type of I/O conditioning (e.g., positioning, fast input, frequency, etc.) that may be impossible to implement using standard I/O modules. Special function I/O modules and smart modules, a type of special interface, can perform this task. Typically, these interfaces process all of the field data within the module itself, thus relieving the CPU from performing this time- consuming duty. For example, PID, three-axis positioning, and stepper motor modules are special function I/O modules that make control implementation much easier. These modules reduce programming and implementation time.

**Remote Inputs/Outputs:**

Remote I/O modules are convenient, cost effective processing devices, especially when used in large systems. Remote I/O subsystems, which are located away from the CPU and connected to it by twisted-pair cables, can dramatically reduce wiring costs, both from a labor and a material standpoint. Another advantage of remote I/O subsystems is that inputs and outputs can be strategically grouped to control separate machines or sections of a machine or process. This grouping provides easy maintenance and allows start-up without involving the entire system. Most controllers that have remote I/O have remote digital I/O. However, users who require remote analog I/O should check to see if this feature is available in the products being considered.

**I/O Bus Networks:**

I/O bus networks, which include device bus and process bus networks, should be considered in applications requiring decentralized control within the PLC system. I/O bus networks provide a topology that allows the direct connection of field devices to a bus network, thereby simplifying wiring. At the same time, these networks let the PLC directly receive I/O field device information about the status of the device. However, the system’s I/O field devices must be compatible with the I/O bus network to take advantage of these enhanced communications features.

**3.10 DISTRIBUTED CONTROL:**

The need to have several main PLCs communicating with each other has brought about **distributed control**. For manufacturers’ LAN systems can be difficult. Therefore, the user should properly define the process application’s functional requirements from the beginning.

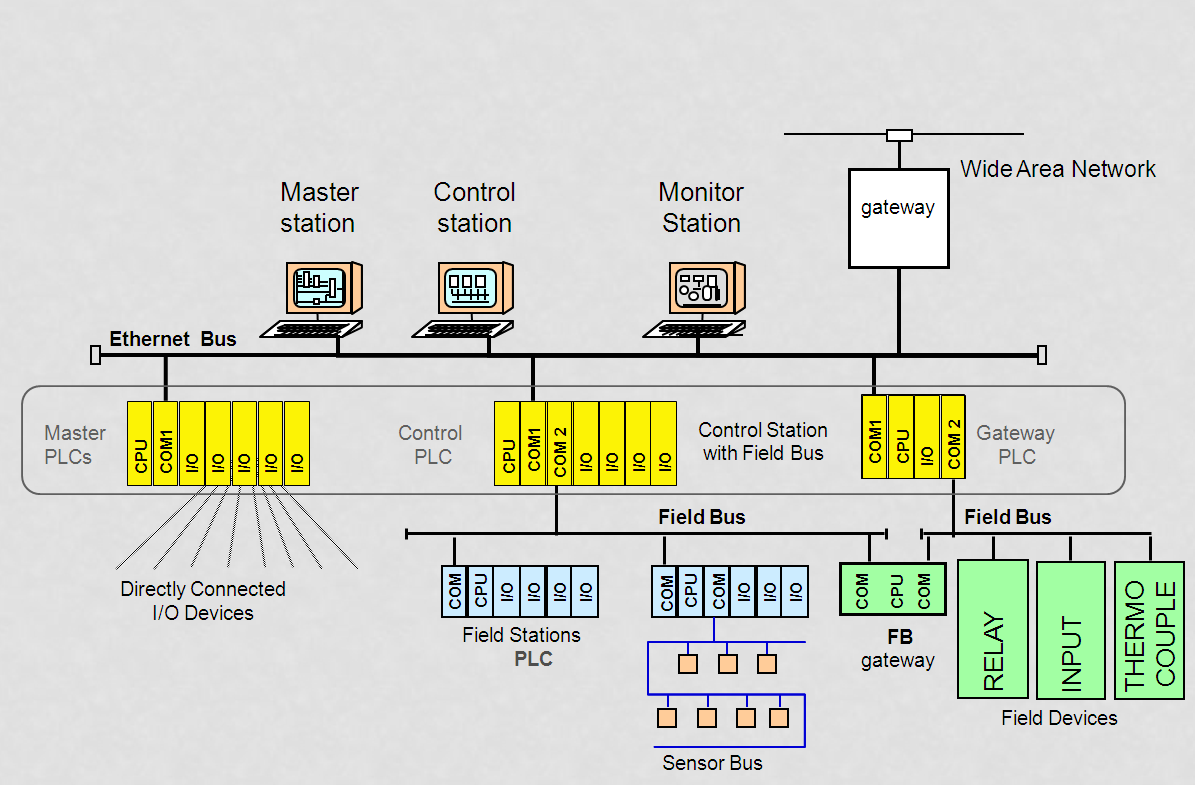


Figure 3.5 **distributed control**

This type of control employs local area networks (LANs), which allow several PLCs to control different stages or processes locally while constantly interchanging in- formation about the process. Communication among PLCs occurs at very high speeds (up to 1 mega baud) through single coaxial or fiber-optic cables. Despite this powerful configuration, communication between two different

**(b)** Centralized control

**(c)** Distributed control

**3.11 MEMORY CONSIDERATIONS:**

The two main factors to consider when choosing memory are the type and the amount. An application may require two types of memory: nonvolatile memory and volatile memory with a battery backup. A nonvolatile memory, such as EPROM, can provide a reliable, permanent storage medium once the program has been created and debugged. If the application will require on-line changes, then it should probably be stored in read/write memory supported by a battery. Some controllers offer both of these options, which can be used individually or in conjunction with each other.

Small PLCs normally have a fixed (non-expandable) memory capacity of1/2K to 2K words. Therefore, the amount of memory is not a major concern when selecting a small controller. In medium and large controllers, however, memory expands incrementally in units of 1K, 2K, 4K, etc. Although there are no fixed rules for determining the amount of memory required, certain guidelines can provide an estimate of memory needs the amount of memory required for a given application is a function of the total number of inputs and outputs to be controlled and the complexity of the control program. The complexity refers to the amount and type of arithmetic and data manipulation functions that the PLC will perform. For each of their products, manufacturers have a rule-of-thumb formula that helps to approximate the memory requirement. This formula involves multiplying the total number of I/O by a constant (usually a number between 3 and 8). If the program involves arithmetic or data manipulation, this memory approximation should be increased by 25 to 50%.

Although memory requirement formulas do a good job of estimating memory needs, the best way to obtain memory requirement data is to create the program and count the number of words used. Knowledge of the number of words required to store each instruction will allow the user to determine exact memory requirements.

**3.12 SOFTWARE CONSIDERATIONS:**

During system implementation, the user must program the PLC. Because the programming is so important, the user should be aware of the software capabilities of the product they choose. Generally, the software capability of a system is tailored to handle the control hardware that is available with the controller. However, some applications require special software functions that are beyond the control of the hardware components. For instance, an application may involve special control or data acquisition functions that require complex numerical calculations and data-handling manipulations. The instruction set selected will determine the ease with which these software tasks can be implemented. It will also directly affect the time required to implement and execute the control program.

**3.13 PERIPHERALS:**

The programming device is the key peripheral in a PLC system. It is of primary importance because it must provide all of the capabilities necessary to accurately and easily enter the control program into the system. The two most common types of programming devices are handheld units and personal computers. Handheld units, which are small and low cost, are typically used to program relatively small control programs in small PLCs. The amount of information that can be displayed on a handheld unit is normally a single program element or, in some cases, a single program rung. Personal computers provide a better way to program a system if the control program is large. Many PLC manufacturers provide software that allows their PLCs to be programmed using a standard PC. However, expansion boards or special interfacing cables may be required to link the personal computer with the programmable controller. Using a PC as a programming device becomes even more advantageous when the same program development software can be used in same-model PLCs or those of the same family. Laptop PCs equipped with programming and documentation software pro- vide even more programming flexibility by joining the ease of PC programming with the transportability of handheld programming devices.

In addition to the programming device, a system may require other types of peripherals at certain control stations to provide an interface between the controller and the operator. The most common peripheral is the line printer, used for obtaining a hardcopy printout of the program and for sending report information about the process. Other peripherals include color displays and alphanumeric displays, which can be used to send messages or alarms about the process, as well as diskette drives, which can be used for storing hourly or monthly production reports on a floppy diskette. If a PC is used as a graphic interface to a PLC system, both systems must have compatible DDE (dynamic data exchange) drivers to properly interface with peripherals. Peripheral requirements should be evaluated along with the CPU, since the CPU will determine the type and number of peripherals that can be interfaced to the system. The CPU also influences the method of interfacing, as well as the distance that peripherals can be placed from the PLC.

**3.14 PHYSICAL AND ENVIRONMENTAL:**

The physical and environmental characteristics of the various controller components will significantly impact total system reliability and maintenance. Ambient conditions, such as temperature, humidity, dust level, and corrosion, can affect the controller’s ability to operate properly. The user should determine operating conditions (i.e., temperature, vibration, EMI/ RFI, etc.), and packaging requirements (i.e., dustproof, drip-proof, ruggedness, type of connections, etc.) before selecting the controller and its I/O system. Most programmable controller manufacturers provide products that have undergone certain environmental and physical tests (e.g., temperature, EMI/RFI, shock, etc.). Users should be aware of the tests performed and whether or not the results meet the demands of the operating environment.

**3.15 PLC RELIABILITY:**

The selection of the PLC down to one of a few possible candidates. More than likely, two or more products will meet all of the requirements of the preliminary system design, meaning that a final decision must still be made. At this point, the user should evaluate a few more factors, which can lead to the selection of the product that best fits the system specifications and the application requirements. The user should discuss these factors with the potential vendors.

The reliability of the controller plays an important role in overall system performance. Lack of reliability usually translates into downtime, poor quality products, and higher scrap levels. The user can investigate several factors to determine the proven reliability of a particular product. **Mean-time-between-failures (MTBF) studies** can be helpful if the user knows how to evaluate the data. These studies provide information about the average time between equipment malfunctions and how long the equipment will operate without a failure. Knowledge of a similar application in which the product has been successfully applied is also useful. A sales representative can provide this information and even, on occasion, arrange a site visit. Moreover, the user should ensure that the vendor can truly satisfy any unique or peculiar specifications (e.g., EMI and vibration requirements). Finally, the user should research the **burn-in procedures** for the product (e.g., the total system burn-in process or the parts burn-in process). The burn-in process involves operating the product at an elevated temperature to simulate extended operation in order to force an electronic board or part to fail. If a part passes the burn-in procedure, it will have an extremely high probability for proper operation. Usually, the vendor can provide MTBF and burn-in information upon request. For making the final decision on a PLC is the possibility of future plans to standardize machinery, that is, to use only products from a given manufacturer and product line. Many companies are adopting this practice for good reasons. Several vendors are creating complete product families of PLCs that cover the entire range of capabilities, thus making standardization more feasible. Another current trend by manufacturers is to build completely inter-compatible product families, with products ranging from very small to very large PLCs. These families share the same I/O structure, programming device, and elementary instruction set. They also have similar memory organization and structure. Because of their similarities, these product families can be linked in a network configuration. PLC families also provide the following important benefits:

•Training on a new PLC family member is a progression of current knowledge, rather than the development of a totally new set of skills.

•Standardized products can result in better plant maintenance in emergency situations.

•I/O spares can be used for all family products, resulting in a smaller spare inventory.

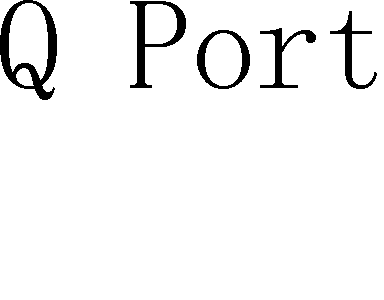
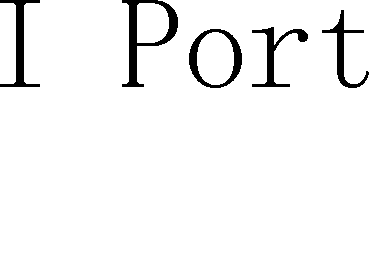
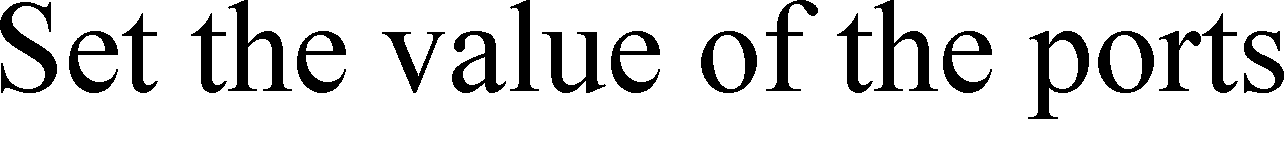
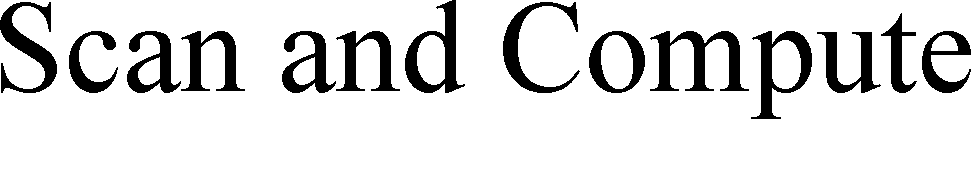
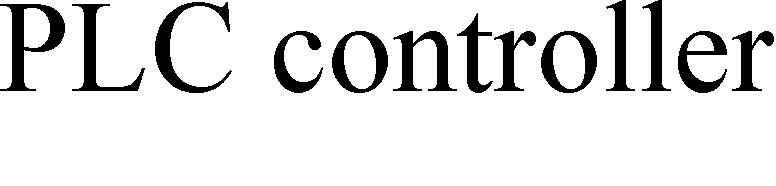
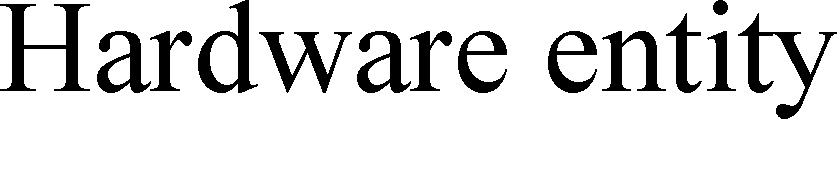
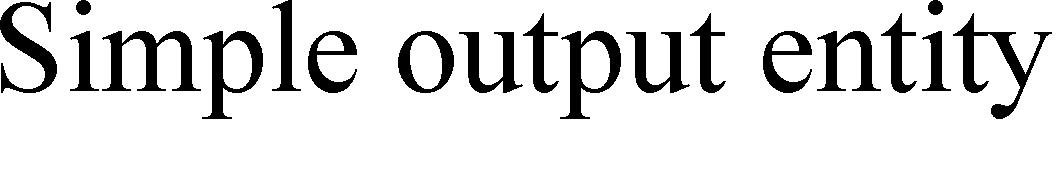
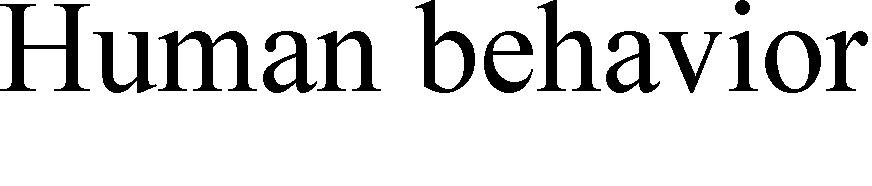
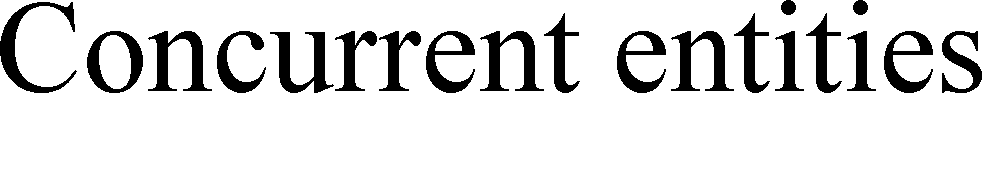
•An outgrown product can be replaced with the next larger product by simply removing the smaller CPU, installing the larger CPU, and reloading the old program.

**3.16** **MODELING OF PLC:**

The technique of PLC using programmable language to control large scale integrated circuit has been widely used in industry. Because of safety critical software can cause serious damage to life or property, verification of safety critical software has become an indispensable step required to assure software quality. The present verifying method for the PLC is still stuck by simulation and testing. However, they cannot cover all possible cases, especial whether the design model of PLC to meet the demand. Therefore, the model checking technology is introduced into the field of PLC. To give theoretical analysis of PLC design becomes important.

The primary step of PLC model checking is to the establishment of PLC model, such as establish a model. The PLC model focuses on the establishment of the time attributes. The state space of the model will be decreased compared to timed automata. PLC control program runs in real-time operating system

There are three steps of modeling, property description, and verification. The most important is how to build the system, PLC controller is not isolated, but has interaction with its working environment, driver and human. Therefore, these factors should also be modeling. The environment, human, and the PLC controller are independent and concurrent with each other in logic. PLC controller interacts with the concurrent entities through the symbols in image table. The symbols of PLC system include I (input port), Q (output port), and M (intermediate relay). Figure 1 is a diagram of PLC system model.



**Figure 3.6 MODELING OF PLC**

Time interval modeling strategy using the flag which specific the bit state of concurrent entities to represent the concurrent entities in the state. In the real model will be a subset of the built model Time interval model. The real PLC environment is complex, and includes a variety of hardware and human behavior. The following gives an analysis of different kinds of PLC environment concurrent entities.

1) Hardware entity

2) Simple output entity

Time interval modeling strategy can get an abstract PLC model; however, the “time interval model” has large deviation with the real model, and if the deviation is “time interval model” does not reflect the high-speed scanning characteristics of PLC and low-speed characteristics of concurrent entities. That is, all the changes in the environment should be scanned by the high-speed PLC, but the time interval model ignores the high-speed characteristics of PLC, which makes changes in the external environment may not be scanned. To address the issues, taking into account the external high speed scanning and low speed concurrent physical characteristics, time interval modeling strategy shall be improved by adding a notice waiting mechanism base on the time interval model, each concurrent state entity must be blocked and wait after the transfer took to concurrent entities to remove the block and go on working. Then the transfer finished. The process that concurrent entities work to complete this mechanism ensures every state change of concurrent entities can be scanned at least once by PLC controller. It is similar to prove proposition. After add the notice waiting mechanism the model still has a good nature. The PLC model can also be developed as:

* Logical Model,
* Reliability Model,
* Stochastic Model, and
* Lab-VIEW Model.

This has been discussed in later chapters.

**LOGICAL MODEL:**

The basic element of PLC logic models are the logic gates AND, OR, and NOT these logic gates are designed to provide a specified output value based on the values of the inputs. The fig. 3.7 to fig. 3.9 shows the elements of logic model with their truth table. The two states are represents the opening and closing the valve or machine status. The output can be either one or zero determent by the presence or absence of the signals.

A push button switch can be used for starting and stopping electric motors and other powered devices, in a Process plant as shown in fig.3.10. It is a combination of two buttons, one for start and second for stop. When start button is depressed momentarily by the operator, power is supplied to the motor and run it until the stop button is pressed.

L

S1 S2

|  |  |  |
| --- | --- | --- |
| S2 | S1 | L |
| 0 | 0 | 0 |
| 0  V | 1 | 0 |
| 1 | 0 | 0 |
| 1  S1  L | 1 | 1 |

S2

**(a) (b)**

**Fig.3.7 Logic AND gate: (a) Circuit illustrating its Model; (b) Truth table**

S1

L

|  |  |  |
| --- | --- | --- |
| S2 | S1 | L |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

S2

V

S1

L

S2

**(a) (b)**

**Fig.3.8 Logic OR gate: (a) Circuit illustrating its Model; (b) Truth table**

|  |  |
| --- | --- |
| S | L |
| 0 | 1 |
| 1 | 0 |

L

S

L

S

V

**(a) (b)**

**Fig.3.9 Logic NOT gate: (a) Circuit illustrating its Model; (b) Truth table**

|  |  |  |  |
| --- | --- | --- | --- |
| START | STOP | MOTOR | POWER TO MOTOR |
| 0  START | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0  STOP | 1 | 1 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 |

**(a) (b)**

STOP

POWER TO MOTOR

START

MOTOR

**( c)**

**Fig.3.10 (a) Pushbutton switch; (b) Truth table; and (c) Logic network Model**

START = 0 normally open contact

START = 1 when the START button is pressed to contact

STOP = 0 normally closed contact

STOP = 1 when the STOP button is pressed to break contact

MOTOR = 0 when off

MOTOR = 1 when on

POWER TO MOTOR is the output of the pushbutton switch

POWER TO MOTOR = 0 when the contact are opened

POWER TO MOTOR = 1 when the contact are closed

The truth table for the pushbutton is presented in fig. 3.10 (b). From initial motor is off (MOTOR = 0), the motor is started by depressing the start button (MOTOR = 1). If the stop button is in its normally closed condition (MOTOR = 0), power to be supplied to the motor (POWER TO MOTOR = 1). While the motor is running (MOTOR = 1), it can be stopped by depressing the stop button (STOP=1). The corresponding network logic diagram is shown in fig. 3.10(c).

Various PLC diagram techniques have been developed to represent Logic Models are:

* Ladder Logic,
* Boolean Logic

**Reliability Model:**

Reliability Model is the operational relationship of is constitute components the PLC reliability model has a series of n basic parallel units as shown in fig.3.11 in n redundant components.

1 2 k

1

1

1

2

2

2

INPUT OUTPUT

3

3

3

: : :

: : :

: : :

n

n

n

**Fig.3.11 Reliability Series and parallel Model**

Let the model has k stages in series and jth stage has nj redundant PLCs. For the failure free operation of jth stage at least one of the nj PLCs.

**nj**

**i=1**

**k**

**J=1**

Rs = {1- (1-rji)} -------------- (1)

Here rji is the reliability of ith redundant PLC of jth stage . If all the redundant PLCs jth stage have same reliability, the system reliability

**k**

**J=1**

Rs = {1- (1-rj)nj }-----------------(2)

The model of having a series of 3 basic units with the parallel of some other units in series as shown in fig.3.11

Cm

X1 X2

C

Ym

Z1

Z2

Y

**Fig.3.12 Reliability Model using Ladder Logic diagram**

Reliability of **X1 & Z1**  = 0.95

Reliability of **C& Y** = 0.87

Reliability of **X2& Z2** = 0.88

Reliability of **Cm& Z1** = 0.90

Reliability (Rs) of X1 and C**=** 1-(1-X1)(1-C)

Rs = 1-(1-0.95)(1-0.87)

Rs = 1- 0.05\*0.13

Rs = 1- 0.0065

Rs = 0.9935

Reliability (Rs) of X1, C, X2, and Cm **=** Rs \*Rx2\*Rcm

= 0.9935\*0.88\*0.90

= 0.786852

Reliability (Rs) of complete model = 1-(1-0.786852)(1-0.786852)

Rs = 1-(1-0.786852)2

Rs = 1-(0.213148)2

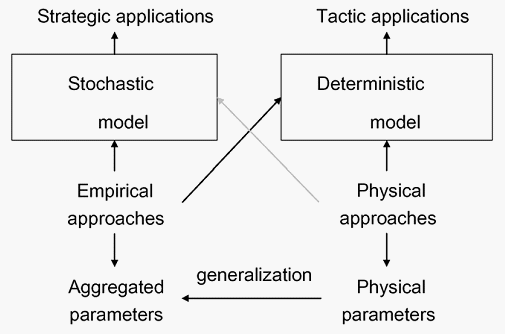
Rs = 1-0.04543207

Rs = 0.954568

Rs = 96%

**STOCHASTIC MODEL:**

A stochasticmodel is a collection of random variables; which is used to represent the evolution of some random value, or system, over time. This is the probabilistic counterpart to a deterministic process. Instead of describing a process which can only evolve in one way, in a stochastic process there is some indeterminacy even if the initial condition is known; there are several directions in which the process may evolve. In the simple case of discrete time, a stochastic model amounts to a sequence of random variables known as a time series. Another basic type of a stochastic model is a random field, whose domain is a region of space, in other words, a random function whose arguments are drawn from a range of continuously changing values. One approach to stochastic models treats them as functions of one or several deterministic arguments whose values (outputs) are random variables non-deterministic (single) quantities which have certain probability distributions. Random variables corresponding to various times may be completely different. The main requirement is that these different random quantities all have the same type. Type refers to the co domain of the function. Although the random values of a stochastic model at different times may be independent random variables, in most commonly considered situations they exhibit complicated statistical correlations.



**LAB-VIEW MODEL:**

Lab-VIEW Model is a presentation of the Real-Time Module combines Lab-VIEW graphical programming with the power of a real-time operating system, enabling to build real-time applications and enhancements to the block diagram and related functionality. Lab-VIEW is a graphical icon based model. It is based on the virtual instrumentation concept and processes the information step-by step instructions with real time module. It can process the information/ data as per the logic designed and having time; counter etc. facility like PLC. Thus, Lab-VIEW can be model for the PLC functions. Also, the Lab-VIEW provides a compatibility with Matlab, Pspice, and standard DAS hardware. Lab-VIEW can be model as sequence style state machine. It executes the states in order until a false value is observed to the conditional terminal. Lab-VIEW also optimizes the performance of the systems and access information about real-time programming concepts, step-by-step instructions for using Lab-VIEW with the Real-Time Module. The DSC Module also provides tools for graphing historical or real-time trends, enhancing the security of front panels, and writing custom I/O servers.

**3.17 CONCLUSIONS:**

In this chapter detailed description of PLCs type, their selection, consideration how can use input and output connection to a PLC. This chapter provides brief consideration of Control, memory, software and its peripherals. For a PLC which types of environment and physical conditions required. The reliability of PLC and its modeling also mentioned.

**CHAPTER-IV**

**PLC PROGRAMMING AND IMPLIMENTATION GUIDE LINES**

**INTRODUCTION:**

The programming languages used in programmable controllers have been evolving since the inception of the PLC in the late 1960s. There are three types of languages used in PLCs today—ladder, Boolean, and Grafcet. These instructions expand programming possibilities in areas such as data manipulation, network communication, data transfer, and program/flow controls, just to name a very few. This programming language standard holds powerful capabilities for the future of PLC programming.

As PLCs have developed and expanded, programming languages have developed with them. Programming languages allow the user to enter a control program into a PLC using an established syntax. Today’s advanced languages have new, more versatile instructions, which initiate control program actions. These new instructions provide more computing power for single operations performed by the instruction itself. For instance, PLCs can now transfer blocks of data from one memory location to another while, at the same time, performing a logic or arithmetic operation on another block. As a result of these new, expanded instructions, control programs can now handle data more easily.

In addition to new programming instructions, the development of powerful I/O modules has also changed existing instructions. These changes include the ability to send data to and obtain data from modules by addressing the modules’ locations. For example, PLCs can now read and write data to and from analog modules.

**4.2 PLC LANGUAGE:**

All of these advances, in conjunction with projected industry needs, have created a demand for more powerful instructions that allow easier, more compact, function-oriented PLC programs. The three types of programming languages used in PLCs are:

•Ladder

•Boolean

•Grafcet

The ladder and Boolean languages essentially implement operations in the same way, but they differ in the way their instructions are represented and how they are entered into the PLC. The Grafcet language implements control instructions in a different manner, based on steps and actions in a graphic oriented program.

**LADDER LANGUAGE:**

The programmable controller was developed for ease of programming using existing relay ladder symbols and expressions to represent the program logic needed to control the machine or process. The resulting programming language, which used these original basic relay ladder symbols, was given the name **ladder language**. Figure 4-1 illustrates a relay ladder logic circuit and the PLC ladder language representation of the same circuit.

L1 L2

PB\*

LS PL

FS

(a) Hardwired Ladder Circuit

LS PL

FS

(b) PLC Ladder Circuit

**\*Note: The PLC will know the elements PB, LS, FS, and PL by their addresses once the address assignment has been performed.**

**Figure 4.1. Hardwired logic circuit and its PLC ladder language implementation**.

The evolution of the original ladder language has turned ladder programming into a more powerful instruction set. New functions have been added to the basic relay, timing, and counting operations. The term functionis used to describe instructions that, as the name implies, perform a function on data that is, handle and transfer data within the programmable controller. These instructions are still based on the simple principles of basic relay logic, although they allow complex operations to be implemented and performed.

New additions to the basic ladder logic also include function blocks, which use a set of instructions to operate on a block of data. The use of function blocks increases the power of the basic ladder language, forming what is known as **enhanced ladder language**. Figure 4.2 shows enhanced functions driven by basic relay ladder instructions. As shown in the figure, a block or a functional instruction between two contact symbols represents an enhanced functional block.

Function Block

MOVE Register-to-Table

Enable

A B

Reset

Functional Instruction Output

A B

MOVE register to table

**Figure 4.2. Enhanced functional block format**.

The format representation of an enhanced ladder function depends on the programmable controller manufacturer; however, regardless of their format, all similar enhanced and basic ladder functions operate the same way. Throughout this chapter, we will refer to enhanced ladder instructions as block format instructions.

As indicated earlier, the ladder languages available in PLCs can be divided into two groups:

•basic ladder language

•enhanced ladder language

Each of these groups consists of many PLC instructions that form the language. The classification of which instructions fall into which categories differs among manufacturers and users, since a definite classification does not exist. However, a de facto standard has been created throughout the years that sorts the instructions into either the basic or enhanced ladder language. Sometimes, basic ladder instructions are referred to as *low-level language*, while enhanced ladder functions are referred to as *high-level language*. The line that defines the grouping of PLC ladder instructions, however, is usually drawn between functional instruction categories. These instruction categories include:

•Ladder relay

•Timing

•Counting

•Program/flow control

•Arithmetic

•Data manipulation

•Data transfer

•Special function (sequencers)

•network communication

Although these categories are straightforward, the classification of them is subjective. For example, some people believe that basic ladder instructions include ladder relay, timing, counting, program/flow control, arithmetic, and some data manipulation. Others believe that only ladder relay, timing, and counting categories should be considered basic ladder instructions. Regardless of classification, the effects of instruction categories are simple the more instruction categories a PLC have, the more powerful its control capability becomes. Usually, small PLCs have only basic instructions with, perhaps, some enhanced instructions. Larger PLCs usually have more advanced instruction sets. However, recent advances in software development and I/O hardware have increased the computational power of small PLCs through advanced instructions. This new trend has made small PLCs very desirable in single, as well as distributed control, applications.

**BOOLEAN LANGUAGE**

Some PLC manufacturers use **Boolean language** also called *Boolean mnemonics*, to program a controller. The Boolean language uses Boolean algebra syntax (see Chapter 3) to enter and explain the control logic. That is, it uses the AND, OR, and NOT logic functions to implement the control circuits in the control program. The Boolean language is primarily just a way of entering the control program into a PLC, rather than an actual instruction-oriented language. When displayed on the programming monitor, the Boolean language is usually viewed as a ladder circuit instead of as the Boolean commands that define the instruction. We will discuss Boolean programming, along with its instruction set, at the end of this chapter.

**GRAFCET LANGUAGE**

**Grafcet** (Graphe Fonctionnel de Commande Étape Transition) is a symbolic, graphic language, which originated in France that represents the control program as steps or stages in the machine or process. In fact, the English translation of Grafcet means “step transition function charts.” Note that Grafcet charts provide a flowchart like representation of the events that take place in each stage of the control program. These charts use three components steps, transitions, and actions—to represent events. The IEC 1131 standard’s SFCs also use these components; however, the instructions inside the actions can be programmed using one or more possible languages, including ladder diagrams.

Few programmable controllers may be directly programmed using Grafcet. However, several Grafcet software manufacturers provide off-line Grafcet programming using a personal computer. Once programmed in the PC, the Grafcet instructions can be transferred to a PLC via a translator or driver that translates the Grafcet program into a ladder diagram or Boolean language program. Using this method, a Grafcet software manufacturer can provide different PLCs that use the same “language.”

-5 LADDER RELAY PROGRAMMING

**4.3 SCAN EVALUATION:**

Scan evaluation is an important concept, since it defines the order in which the processor executes a ladder diagram. The processor starts solving a ladder program after it has read the status of all inputs and stored this information in the input table. The solution starts at the top of the ladder program, beginning with the first rung and proceeding one rung at a time. As the processor solves the control program, it examines the reference address of each programmed instruction, so that it can assess logic continuity for the rung being solved. Even if the output conditions in the rung being solved affect previous rungs, the processor will not return to the previous rung to resolve it.

**4.4 PLC TIMERS AND COUNTERS:**9- TIMERS AND COUNTERS

PLC timers and counters are internal instructions that provide the same functions as hardware timers and counters. They activate or deactivate a device after a time interval has expired or a count has reached a preset value. Timer and counter instructions are generally considered internal outputs. Like relay type instructions, timer and counter instructions are fundamental to the ladder diagram instruction set.

**4.5 TIMER INSTRUCTIONS:**

**Timer instructions** may have one or more ***time bases***(TB) which they use to time an event. The time base is the resolution, or accuracy, of the timer. For instance, if a timer must time a 10 second event, the user must choose the number of times the time base must be counted to get to 10 seconds. Therefore, if the timer has a time base of 1 second, then the timer must count ten times before it activates its output. This number of counts is referred to as *ticks*. The most common time bases are 0.01 sec, 0.1 sec, and 1 sec. Timers are used in applications to add a specific amount of delay to an output in the program. Applications of PLC timers are innumerable, since they have completely replaced hardware timers in automated control systems.

**4.6 COUNTER INSTRUCTIONS:**

**Counter instructions** are used to count events, such as parts passing on a conveyor, the number of times a solenoid is turned ON, etc. Counters, along with timers, must have two values, a **preset value**and an **accumulated value**. These values are stored in register or word locations in the data table. The pre-set value is the target number of ticks or counting numbers that must be achieved before the timer or counter turns its output ON. The accumulated value is the current number of ticks (timer) or counts (counter) that have elapsed during the timer or counter operation. The pre-set value is stored in a pre-set register, while the accumulated value is kept in an accumulated register.

**UP COUNTER:**

An ***up counter***(CTU) output instruction adds a count, in increments of one, every time its referenced event occurs. In a control application, this counter turns a device ON or OFF after reaching a certain count (i.e., the pre-set value in the pre-set register). Also, this counter can keep track of the number of parts that pass a certain point. An up counter increases its accumulated value each time the up count event makes an OFF-to-ON transition. When the accumulated value reaches the pre-set value, the counter turns ON the output, finishes the count, and closes the contact associated with the referenced output. After the counter reaches the pre-set value, it either resets its accumulated register to zero or continues its count for each OFF-to-ON transition, depending on the controller.

**DOWN COUNTER:**

A ***down counter***(CTD) output instruction decreases the count value in its accumulated register by one every time a certain event occurs. In practical use, a down counter is used in conjunction with an up counter to form an *up/ down counter*, given that both counters have the same reference registers. In an up/down counter, the down counter provides a way to correct data that is input by the up counter. For example, while an up counter counts the number of filled bottles that pass a certain point, a down counter with the same reference address can subtract one from the accumulated count value every time it senses an empty or improperly filled bottle. Depending on the programmable controller, the down counter will either stop counting down at zero or at a specified maximum negative value. In a block format instruction, a down count occurs every time the down input of the counter transitions from OFF to ON.

**COUNTER RESET**

A *counter reset* (CTR) output instruction resets up counter and down counter accumulated values to zero. When programmed, a counter reset coil has the same reference address as the corresponding up/down counter coils. If the counter reset rung condition is TRUE, the reset instruction will clear the referenced address. The reset line in a block format counter instruction sets the accumulated count to zero (accumulated register = 0).

**4.7 BOOLEAN MNEMONICS:**

Boolean mnemonics is a PLC language based primarily on the Boolean operators AND, OR, and NOT. A complete Boolean instruction set consists of the Boolean operators and other mnemonic instructions, which implement all of the functions of the basic ladder diagram instruction set. A mnemonic instruction is written in an abbreviated form, using three or four letters that imply the operation of the instruction. The Boolean language is used to enter logic into a PLC’s memory. However, a PLC may display the entered Boolean information as a ladder diagram on the programming terminal. Enhanced Boolean output operators, which perform additional control functions, are a result of further enhancements to the Boolean instruction set.

**4.8 IMPLIMENTATION GUIDELINES:**

Programming a PLC can be a difficult task due to increased interlocking requirements in the control program as it becomes larger and more complicated. Additionally, each PLC manufacturer offers a different set of instructions within its PLC family. Many of these instruction sets are not applicable to other PLCs, and there is no easy way to translate a written PLC program to another brand of PLC’s programming format.

We will explain the languages used with the **International Electro-technical Commission (IEC)** standard control program using sequential function charts to ease interlocking. The International Electro-technical Commission (IEC) SC65B-WG7 committee developed the **IEC 1131 standard** in an effort to standardize programmable controllers consists of five parts:

• General information

• Equipment and test requirements

• Programming languages

• User guidelines

• Messaging services (communications)

Although there are five parts in the IEC 1131 standard, the third part programming languages provides all of the information about instructions and programming standards. The other four sections describe the different guidelines to be used for the testing and communication of language instructions, as well as the methodology that must be employed by the programmable controller user.

**4.9 LANGUAGES AND INSTRUCTIONS:**

The IEC 1131-3 standard defines two graphical languages and two text- based languages for use in PLC programming. The graphical languages use symbols to program control instructions, while the text-based languages use character strings to program instructions.

**Graphical languages**

•ladder diagrams (LD)

•function block diagram (FBD)

**Text-based languages**

•instruction list (IL)

•structured text (ST)

By the IEC 1131-3 standard includes an object oriented programming framework called **sequential function charts (SFCs)**. The IEC 1131-3 standard is a graphic/object-oriented block programming method, which increases the programming and troubleshooting flexibility of its programmable controllers. It allows sections of a program to be individually grouped as tasks, which can then be easily interlocked with the rest of the program. Thus, a complete IEC 1131-3 program may be formed by many small task programs represented inside SFC graphic blocks. The IEC 1131-3 uses a wide variety of standard data functions and function blocks, which operate on a large number of data variable types. G LANGUAGES

**4.10 PROGRAMMING LANGUAGE NOTATION:**

As we have noted, sequential function charts can provide the infrastructure for a control program, which is then built using one or more of the four IEC1131-3 programming languages. In the next section, we will further explain how SFCs can be used implement a control program. However, let’s first review the similarities between programming notations in the ladder diagram (LD), function block diagram (FBD), structured text (ST), and instruction list (IL) languages.

A simple ladder diagram and its FBD, ST, and IL language equivalents. Note that the ST language uses two operators, AND &, to denote the AND function. Symbol is used in an ST program to assign an output variable to a logic expression. In instruction list the first instruction (instruction LD) loads the status of variable Limit\_S\_1 to the accumulator register, which IL calls the *result register*. The second instruction (instruction AND) ANDs the status of Limit\_S\_1 with the variable Start Cycle and stores the outcome back in the result register. The third instruction (instruction ST) stores the contents of the result register as the output variable, Valve\_3. This process is similar to Boolean programming language.

As demonstrated, the instructions used to implement control sequences in each programming language are very similar in their construction, as well as their visual representation. Depending on the PLC application, an SFC may use one or more of these languages to program instructions inside its actions. To differentiate between languages, some software manufacturers include starting and ending commands that define the language being used. Other manufacturers allow the mixing of languages without any differentiation between them.

**4.11 SUB-PROGRAMS:**

As a process may have several main program charts executing different main control tasks within the PLC system. Depending on the IEC 1131-3 software system, these main programs may utilize one or more **subprograms** to implement specialized control sequences with the ability to have a main program with one or more subprograms organized in a “father-child” relationship. A father program can “call” (i.e., jump to) any of the child programs in a process, but a child program can only have one father program.

S

**4.12 TYPES OF STEP ACTION:**

An action in a sequential function chart is executed when its corresponding step is active. When the step becomes active, the software control instructions contained in the action will be executed and scanned until the token is transitioned to the next step in the chart. A step action can take several forms, depending on the desired operation and result. These types of actions are:

• Boolean actions

• Pulse actions

• Normal actions

• SFC actions

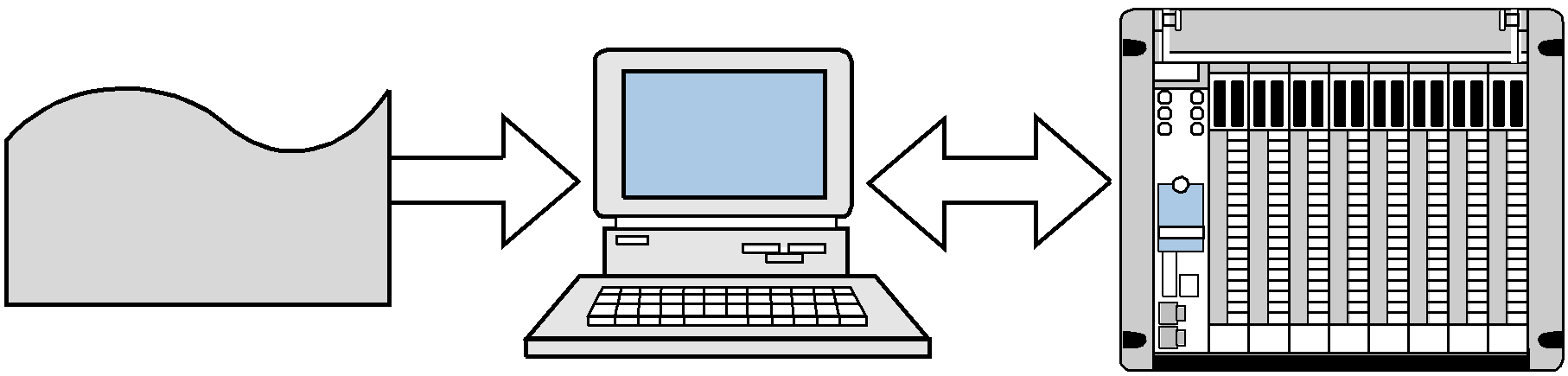
**4.13 IEC 1131-3 SOFTWARE SYSTEMS:**

In addition to the implementation of the IEC 1131-3 in PLCs, many manufacturers of software systems provide the IEC 1131-3 standard in different hardware platforms and operating systems, such as Windows and UNIX. These software systems emulate the operation of a programmable controller (i.e., they are software PLCs or “soft PLCs”) in the hardware platform being used (e.g., a PC). They support either a third-party I/O system or one or more of a PLC manufacturer’s I/O through the use of built-in drivers that communicate with an I/O rack as shown in Figure

PC

(“Soft PLC”) I/O Devices

IEC 1131-3



Software System

**Figure. 4.3** A software PLC interfaced with I/O devices.

**4.14 PLC LANGUAGES SIMILAR TO THE IEC 1131-3:**

PLC Programming

SECTION

3

CHAPTER

10

The IEC 1131 Standard and

Programming Language

CHAPTER

10

The IEC 1131 Standard and

Programming Language

PLC Programming

SECTION

3

PLC manufacturers may adapt their programmable controller languages to embrace some of the qualities of the IEC 1131-3 standard. These qualities usually reflect the ease of programming found when using sequential function charts to encapsulate parts of a ladder program into an action. This added software versatility enhances a programmable controller system tremendously by speeding up program development, minimizing interlocking sequences, and reducing system troubleshooting time.

For instance, PLC Direct, a PLC manufacturer, offers programmable controllers with both standard ladder programming language instructions (RLL—relay ladder logic) and RLL Plus, which is their software language that incorporates some of the features of sequential function charts. In fact, the RLL Plus language closely follows the activation of a horizontal flowchart. The sequence chart in the RLL Plus program is much easier to understand and trouble shoot

The RLL Plus programming language, like sequential function When the control program starts, the initial stage is activated. Jump instructions, driven by the ladder diagram contacts that form the transition logic, pass the token from stage to stage. The last rung in the active stage performs the transition logic. The RLL Plus software also supports divergences and convergences, along with the use of timers and counters in the implementation of transitions. Subroutine implementations are also available through the use of call instructions in the stage programming

The one-shot circuits used in the LVDT application prevent the system from moving the motor forward or backward until the part is at exactly the virtual position in counts. Analog count signals may jump one or two counts in either direction (up or down). This can result in instability, causing the forward and reverse signals to clash. The logic that is employed in this subroutine will detect, once the part crosses the virtual position whether the part is coming from a reverse motor or forward motor operation. Once the part is detected (i.e., when the one-shot is triggered), a minor jump in analog counts will not affect the operation, since the program has already determined that the part has just passed the virtual position. After the part stops at the virtual position, both the forward and reverse motor commands from the subroutine are inhibited.

**4.15 CONCLUSION:**

A brief introduction of PLC language, scan evaluation, timer and counter. The use of timer instructions, counter instruction and Boolean mnemonics in detail. The implementation guidelines, languages, instructions, programming language and notation of PLCs. The steps of action taken for a program and IEC 1131-3 Software Systems and PLC Languages similar to the IEC 1131-3 given in brief.

**CHAPTER-V**

**IMPLEMENTATION OF PLC-HYBRIDE SYSTEM FOR SCADA**

**INTRODUCTION:**

In order to show the utility of “PLC Hybrid System for SCADA“, Fonic PLC has been integrated with core-2 duo PC having windows XP-OS. The lab prototype model uses of linear, Variable Differential transformer and voltage source as processes LabVIEW for the data capturing and PLC as pre process. The pre-processed data are further passed to operator consol using Ethernet bus.

**5.2 LINEAR VARIABLE DIFFERENTIAL TRANSFORMER (LVDT):**

Linear Variable Differential Transformers (LVDTs) operate on the principle of a movable core inside a wound coil, which generates voltage changes depending on the position of the core. Therefore, the attachment of a rod or a similar element to the core provides a way to measure linear displacement, force and strain etc..

It can also provide position feed- back for the moving mechanism of a machine. Figure 5.1 illustrates a block diagram of an LVDT application. The LVDT has arranged of 10cms from its null position; therefore, the effective total range is 20 cm from a zero reference. The LVDT provides a10VDCsignal disconnected to an analog input module, which transforms the –10 to +10 VDC voltage swing into counts ranging from –4095 to +4095.

When the start push button starts the machine, the moving piece must move to the virtual starting position (V.P.) defined by these to digit TWS. The TWS settings range from 00.00 to 20.00; the decimal point will be implemented in the controller. When the machine finishes its cycle, the moving piece must return to the virtual position. The machine cycle may end at either side of the virtual starting position.

Subroutines are used to implement the flowchart, to facilitate interlocking and programming. Latch instructions enable the subroutines, allowing the program to go to a subroutine until its operation has been performed. Once a subroutine finishes its function, it sends an unlatch signal signifying the end of the subroutine. This unlatch signal triggers the execution of the next sub-routine. The Figures 5.4, 5.5, and 5.6 present the subroutine codes. In Figure 5.4 (check for 0-cms position), the compare instruction checks for the LVDT count to be less than or equal to the compare constant –4090, rather than strictly equal to the value –4095. If the instruction checked for the value to be strictly equal to –4095, then fluctuations inherent in the LVDT’s count output could cause the PLC to not latch this value. So, once the LVDT passes –4090 counts, it latches this value and assumes that position is at 0 cm.

LVDT

LVDT Attachment

Motor

0inches

Virtual Position (V.P.)

20inches

Virtual Position

5 8 6 7

TWS

Start PB( ) Reset PB( ) Stop PB( )

Counts

+4095

10 20

Displacement (inches)

**Figure 5.1 LVDT analog position reading system**

–4095

START

Read LVDT analog input continously.

Is No

StartPB1

ON?

Yes

GOSUBtoensure thatpositionisat

0inches.

Onceat0inches, thengotoV.P. ReadTWSand converttocounts.

AfterV.P.isread, startmachinecycle. Issueendofcycle.

GobacktoV.P. afterendof machinecycle.

Ifstopispushed, stopallmachine activity.

Ifresetispushed, stopactivityandgo backto0"position.

END

**Figure 5.2** Flowchart of the LVDT reading and virtual position calculations.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **I/O Address** | | |  |
| **Module**  **Type** | **Rack** | **Group** | **Terminal** | **Description** |
| Input | 0  0  0  0 | 0  0  0  0 | 0  1  2  3 | StartPB1—tovirtual position StopPB2—stopmachine(NC) Reset PB3—resetto0"position |
| Register Input (high byte) | 0  0  0  0  0  0  0  0 | 1  1  1  1  1  1  1  1 | 0  1  2  3  4  5  6  7 | Mostsignificanttwodigitsof TWSchannel1(virtual position indecimalpoints) |
| Register  Input  (lowbyte) | 0  0  0  0  0  0  0  0 | 2  2  2  2  2  2  2  2 | 0  1  2  3  4  5  6  7 | Leastsignificanttwodigitsof TWSchannel1(virtual position indecimalpoints) |
| Output | 0  0  0  0 | 3  3  3  3 | 0  1  2  3 | Forwardcommand  Reversecommand |
| Analog  Input | 0  0  0  0 | 7  7  7  7 | 0  1  2  3 | Channel1LVDTanaloginput  Channel2spare Channel3spare Channel4spare |

**Table 5.1 I/O address assignment**.

**Register**

**Description**

4000

4001

4002

4003

4100

TWSvalue inBCD;virtual position TWSvalue inbinary afterconversion Subtractionof4095(–4095)

Virtualpositionincounts (equation) LVDTanalogvalue incounts

**Table 5.2 Register assignment**.

**Device**

**Internal**

**Description**

|  |  |  |
| --- | --- | --- |
| — | 1000 | Startmachinecommand |
| — | 1001 | LVDTanaloginputestablished(LVDTRead) |
| — | 1100 | Latchforenabletogotosubroutine |
| — | 1150 | CompareLVDTpositionwith0inches |
| — | 1151 | Position reached—0"position |
| — | 1152 | Energizereversemotor commandfromthissub |
| — | 1153 | One-shotposition0"found |
| — | 1200 | Latchtoenabletogotosubroutine(TWSRead) |
| — | 1250 | Read TWSblock enable(TWSRead Sub) |
| — | 1251 | ConvertoutputfromBCDtobinary (decimal) |
| — | 1252 | Multiply(accordingtoequation)enable |
| — | 1253 | Subtract enabled |
| — | 1254 | Compareenabled |
| — | 1255 | V.P.found—1254ON(Pos V.P.) |
| — | 1256 | Energizeforwardmotor fromthissub |
| — | 1257 | One-shotpositionV.P.found |
| — | 1300 | Latchtoenabletogotosubroutinetoreturn toV.P. |
| — | 1350 | CompareLVDTwithV.P. ()—ReturntoV.P. sub  (Pos>V.P.) |
| — | 1351 | Position V.P.—reversemotor (Ahead ofV.P.) |
| — | 1352 | CompareLVDTwithV.P.()(Pos<V.P.) |
| — | 1353 | Position V.P.—forwardmotor (BehindV.P.) |
| — | 1354 | LatchfoundV.P.fromPos V.P.(Reverseaheadof  V.P.) |
| — | 1355 | OneshotfoundV.P.fromreverse |
| — | 1356 | LatchfoundV.P. fromPos V.P. (ForwardbehindV.P.) |
| — | 1357 | OneshotfoundV.P.fromforward |
| — | 1360 | Reversemotor fromthissub |
| — | 1361 | Forwardmotor fromthissub |
| — | 1362 | OneshotfoundV.P.from orfrom aftercycle |
| — | 1400 | Latcharesettogotosubfor0"positionafterreset |
| — | 1700 | Latchtogotomachinecycle |
| — | 1750 | Gosubmachinecycle |
| — | 1777 | Endofcycle signal (CycleDone) |

**Table 5.3 Internal output assignment**.

Start

000

StartMach

1000

Stop

001

Reset

002

StartMach

1000

Startmachine–nostop,noreset.

CycleDone

1777

Stop

001

DisableCycle

1700

U

XFERIN LVDTRead

1001

Rack0

Slot7

Reg4100

Readanalogpositioncontinu- ously.

Reset

002

StartMach

1000

Length1

Enable

GOSUB1100

L

CycleDone

1777

BacktoV.P.

1300

L

Oncetheendof cycleis completed, return toV.P.for next cycle. WhenV.P. is achieved(1362ON),proceed

Enable

GOSUB1100

0"Found

CMPLVDT=0"

GOSUB1150

After start, go to subroutine and make sure at0"tostart. Onceat 0" position(1153ON), gotoV.P.

BacktoV.P.

1300

FoundV.P.AfterCycle

SubReturntoV.P. tonextcycle.

GOSUB1350

DisableSub

1153 1100

U Stop

001

Reset

002

Deactivate subroutineto look for0".Systemis stoppedor reset.

1362

Stop

001

Reset

002

1300

U

Disablesubroutineto findthe V.P. aftercycle.Can be stopped because stop of resetcommands.

0"Found

1153

EnableReadTWSSub

1200

Reset

002

Goto0"afterreset

1400

Ifresetispushed,returnto0"

EnableReadTWSSub

1200

L

ReadTWSSub

GOSUB1250

Read TWSand look for V.P. (sub1250).

Goto0"afterreset

1400

L

CMPLVDT=0" GOSUB1150

positionandwaitforstart.

AtV.P.Ready

Stop

001

Reset

002

AtVPReady

1257

CycleReady

1700

Deactivate

U

CycleReady

1700

L

CycleMachine

GOSUB1750

look forV.P. becauseit is

cycleof themachinein sub- routine 1750. Theprogram ofsubroutine1750(not included) performsthe machiningtask. Deactivation ofmachine cycle canalsooccurduetoa stopor resetcommand. Oncecycleis finished,the subroutinemust issue anOS 1777 to signify theendof cyclefromthesub- routine.Whenit returns,the machinecycleisdeactivated.

0"Found

GoFwd1

1256

GoFwd2

1361

GoRev1

1152

GoRev2

1360

REV

031

FWD

030

UFWDMotor

030

REV

031

tionafterreset.

Forward Motor from com- cle)orGoFwd2(aftercycle).

Reversemotor.

**Figure 5.3 PLC implementation of the LVDT analog position reading example**.

XFERIN ReadTWSSub

1250

Rack0

Slot1

Reg4000

Length1

ReadTWSvaluein inches. Theformathas twodecimal points(10–2).

BCD-BIN Reg4000

Reg4001

Length1

MUL Reg4001

x

RegK4095

=

Reg4002

Scale–3

SUB

BCD-BINDone

1251

MULDone

1252

ConvertfromBCDtobinary

(decimal).

Multiply decimal value multiplier(x10–2 becauseof two decimals) with409.5 (4095x 10–1).Storein register4002 (counts). Scaleto 10–3due to both multipliers.

SUBDone

1253

Reg4002

–

RegK4095

=

Reg4003

Subtract4095accordingto thelinearizationequation.

CMP 1254

Reg4100

Compare valueof analog inputincountswithV.P.in

V.P.Found

1255

V.P.Found

1255



Reg4003 

V.P.Found

1255

GoFwd1

1256

AtV.P.Ready

1257

OS RET

counts.Ifgreaterorequal,

indicate1255(ON).

If V.P.not found,start motor.Move forwarduntil V.P.isreached.

Readyfor machinecycle.V.P.reached.Proceedwith

nextoperation.

**Figure 5.4Subroutine 1250 moves the part to the virtual position**.

Reg4100



Reg4003

Reg4100



Reg4001

Return to V.P.

1350

Ahead of V.P.

1351



CompareDone

1352

Behind V.P.

1353



ComparevalueofLVDTpo- sition with V.P. If greater or equal, indicate 1351. If

1351 =ON, V.P. is notfound from aposVP.

Compare value ofLVDT postionwithV.P.Ifless or equal, indicate 1353. If

1353 =ON,V.P.is not found from apos< V.P.

Ahead of V.P.

1351

Ahead of V.P.

1351

OS Found V.P. Rev

1355

Behind V.P.

1353

Behind V.P.

1353

OS Found V.P. Fwd

1357

RevAhead of V.P.

1354

Fwd Behind V.P.

1356

OS Fwd

1357

OS Rev

1355

RevAhead of PV

1354

L

OS Found V.P. Rev

1355

OS

Stop RevMotor

1354

U

Fwd Behind V.P.

1356

L

OS Found V.P. Fwd

1357

OS

Stop Fwd Motor

1356

U

Go Rev2

1360

Go Fwd2

1361

Not back at V.P. froma position greater than V.P. Therefore, reversemotor.

FoundV.P.(latchsignal) via reversing motorcommand. Issuea foundcommand (1355ON).

OSstopsreverse motor commandGo Rev2(1360).

FoundV.P.(latchsignal) via forwarding motor command. Issuea foundcommand (1357ON).

OS stops forward motor commandGo Fwd 2(1361).

Reversemotor untilV.P. isfound.

Forward motor until V.P. is found.

OS Found V.P. Rev

1355

OS Found V.P. Fwd

1357

Found V.P. After Cycle

1362

RET

Issuecommandbacktomain reachedafter machinecycle.

**Figure 5.5Subroutine 1350 returns the part to the virtual position at the end of cycle.**

**5.3 FLASH LIGHT:**

Here a voltage source is used for the input to the PLC. When the voltage exceed the limit, the LED starts blinking. A flasher circuit (see Figure 5.7) toggles an output ON and OFF continually.

TMR2 TMR1

TMR1

TMR2

.Alarm

Cond. 1

Alarm

Cond. 2

Alarm

Cond. n

n

TMR1

TMR1

TMR1

Alarm 1

Output

Alarm 2

Output

Alarm n

n

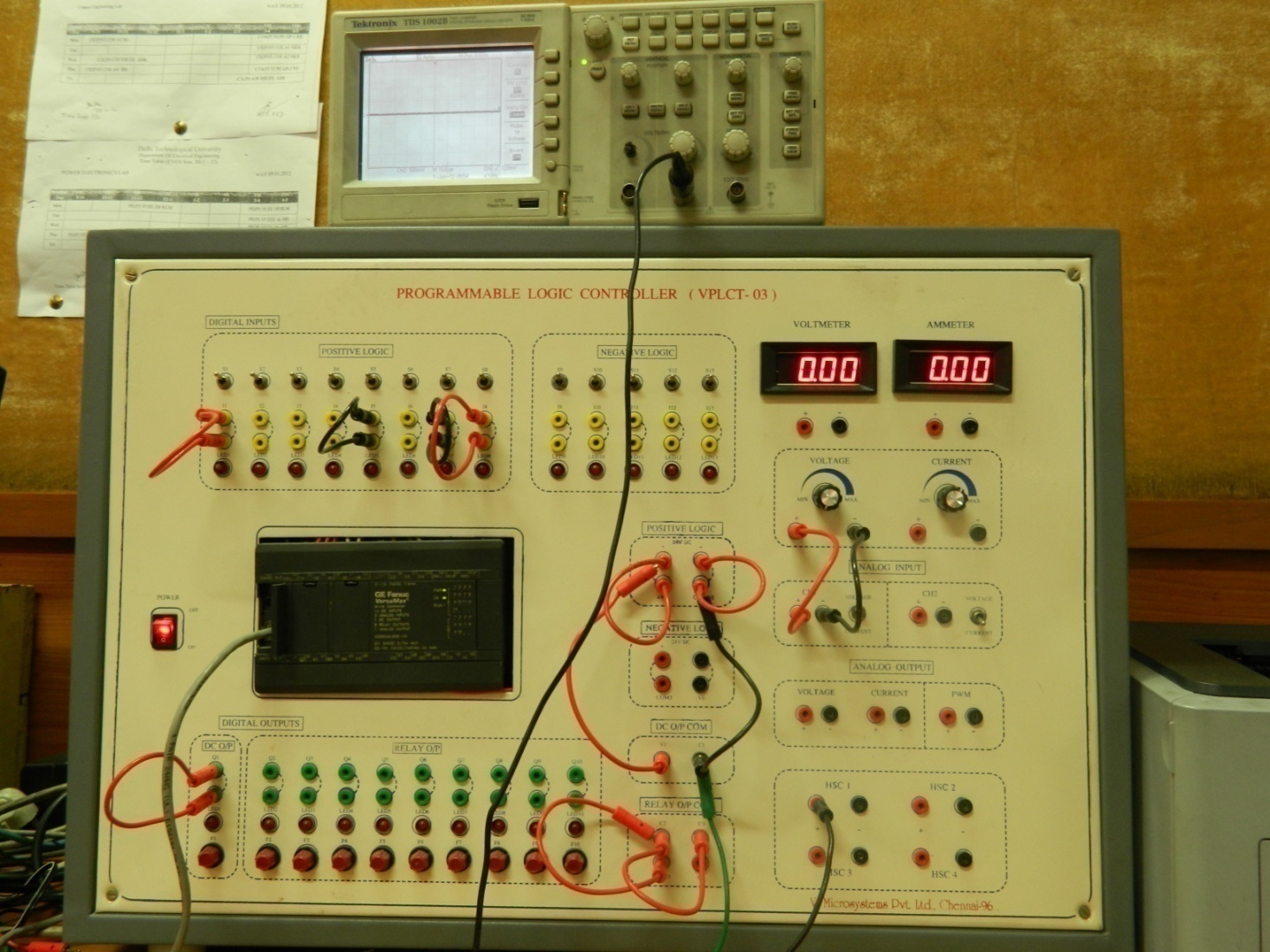
Output

**Figure 5.6** Enunciator flasher circuit.

In this circuit, an oscillator circuit output (TMR1) is programmed in series with an alarm condition. As long as the alarm condition is TRUE, the enunciator output will flash. The output in this case is a pilot light; however, this same logic could be used in conjunction with a horn, which would pulse during the alarm condition. Any number of alarm conditions can be programmed using the same flasher circuit.

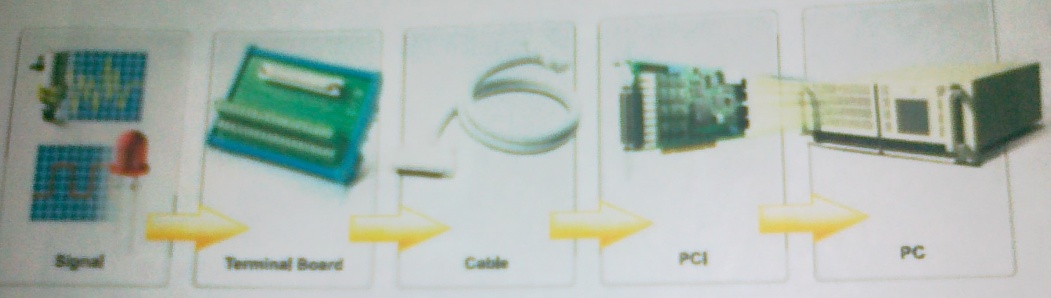
**5.4 SETUP FOR PROGRAMMABLE LOGIC CONTROLLER (VPLCT-03):**

VPLCT-03 is a PLC setup used in this project in which a **MICRO PLC –IC200UAL006** is used for operation. It is a 23 point PLC having 10relay

****

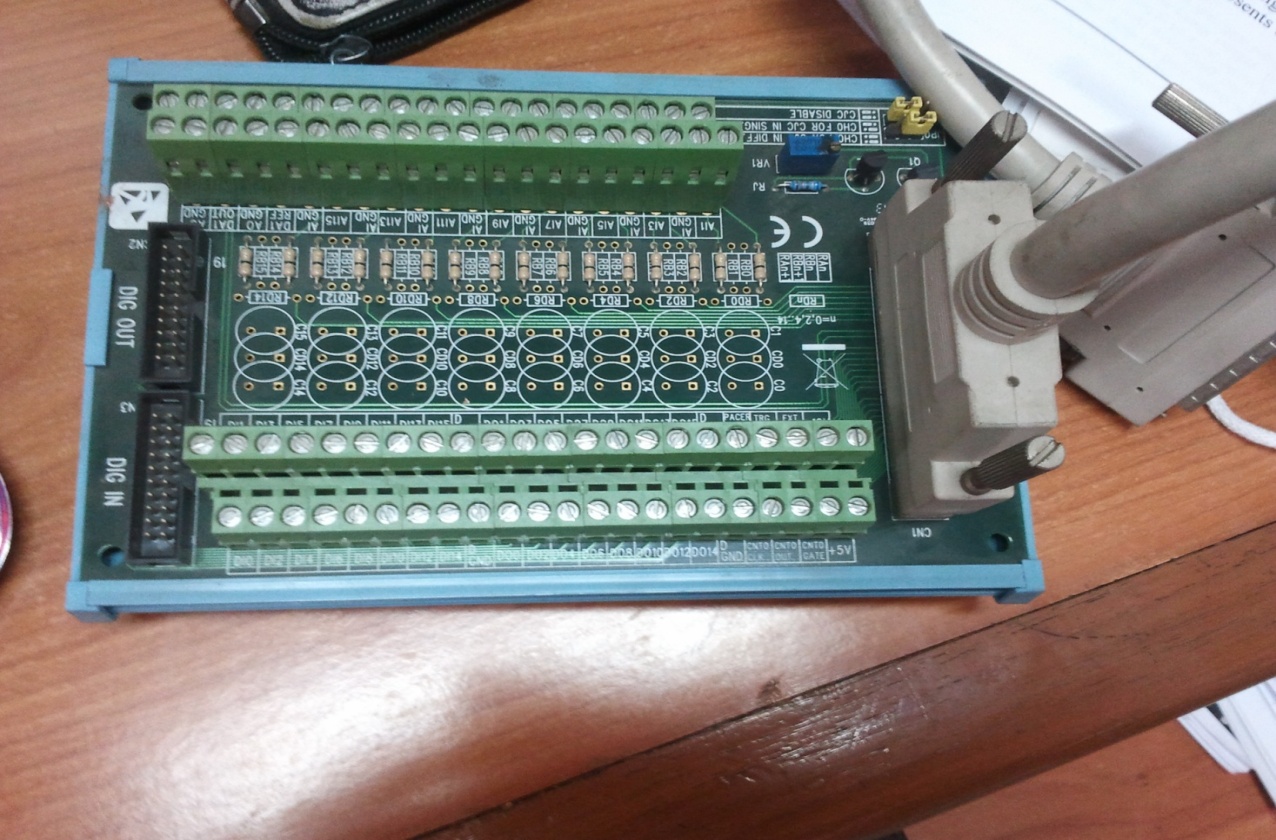
**Figure 5.7 Setup for Programmable Logic Controller (VPLCT-03)**

outputs, 8 positive logic inputs and 5 negative logic inputs are available. This system has also high speed counter ( HSC) function with 4 channel setting and also provision for analog input and PWM output. This system is stand alone for system designing using PLC architecture.



**Figure 5.8 Setup accessary for Programmable Logic Controller (VPLCT-03**

**5.5 PCLD 8710 SCREW TERMINAL BOARD:**



**Figure 5.9 PCLD-8710 WIRING TERMINAL BOARD REV. A1 01-3**

PCLD-8710 is designed to match multifunction card with 68-pin SCSI-II connectors, such as **PCI-1710**/1710L/1710HG/1710HGL/1711/1711L/1716/1716L cards. This screw – terminal board also include cold junction sensing circuitry that allows direct measurement from LVDT, voltage sources including thermocouple transducer. Together with software compensation and linearization, signal – conditioning circuits, can easily be construct as a low – pass filter, attenuator or current shunt converter by adding resistors and capacitors onto the board circuit pads.

**5.6 WIRING CABLE PCL-10168:**

PCL -10168: shielded cable which is specially designed for PCI-1711 card to provide high resistance to noise.



**Figure 5.10 PCLD-8710 WIRING TERMINAL BOARD REV. A1 01-3 WITH DATA CABLE**

To achieve better signal quality, the signal wires are twisted in such a way as to form a “twisted – pair cable “, reducing cross – talk and noise from other signal sources. Furthermore, its analog and digital lines are separately sheathed and shielded to neutralize EMI/EMC problems. Hence the communication between PCI-1711 card and PCLD -8710 is through wiring cable. PCL-10168 which is as shown in below fig. But practically PCI-1711 multi-function card will be inserted in PC therefore it can't be viewed.

**5.7 PCI-1711 MULTI FUNCTION CARD:**

 PCI-1711 is a powerful, but low – cost multifunction card for PCI bus. PCI-1711 comes with 2 analog output channels.

**Figure 5.10 PCI-1711 MULTI FUNCTION CARD**

**Features**

* 16- Channel single ended analog input.
* 12 bit A/D converter, with up to 100 KHz sampling rate.
* Programmable gain
* Automatic channel/ gain scanning.
* On board FIFO memory (1,024 samples)
* Two 12 bit analog output channels
* 16 bit digital input and 16 bit digital output
* On board programmable counter

**5.8 PLC-HYBRID SYSTEM FOR SCADA:**

Integrating DCS with PLC and/0f computers is a vertical communication for information exchange only. It also includes horizontal linkages between the DCS and PLC AND/OR computer sharing control responsibility as peers in the system. Thus PLC-HYBRID SYSTEM FOR SCADA provides common variable names addressing and functions, all sharing a unified operation interface with data information in the system accessible through. Therefore the content of this section are like menu system integration. Some of the key regions to integrate and design the **PLC-HYBRID SYSTEM FOR SCADA** are:

To obtain a superior men-machine interface, sequencing, supervisory control and improvement in performance and higher productivity figure below 5.11 shows a PLC HYBRID SYSTEM.

The LVDT and voltage source devices/Transducers used as SCADA function. The serial data links are far more common between DCSs and computers, especially where very high speed communication is not essential to proper performance. Some PLC-to-SLC communication is also well supported by serial links. Where the PLC serves only to provide information to the DCS for purposes of archival storage or where minimal HMI- related data is transferred, serial link is the appropriate choice. A serial link transfers the data as a string of pulses. Typical transmission speeds are from 1200 bits per second to 19,200 bits per second. With a 70% data-to-overhead ratio, 840 data bits per second to 13640 per second.

The operator interface must present information to the operator quickly and in a form permitting rapid decision making and efficient manipulation of needed controls. Continuous monitoring of key operating variables must be provided, with alarming for malfunction of process equipment or control system elements. The alarm must be detected, with prompt access provided to the display and control of pertinent data during upset conditions to ensure proper implementation of emergency procedures.

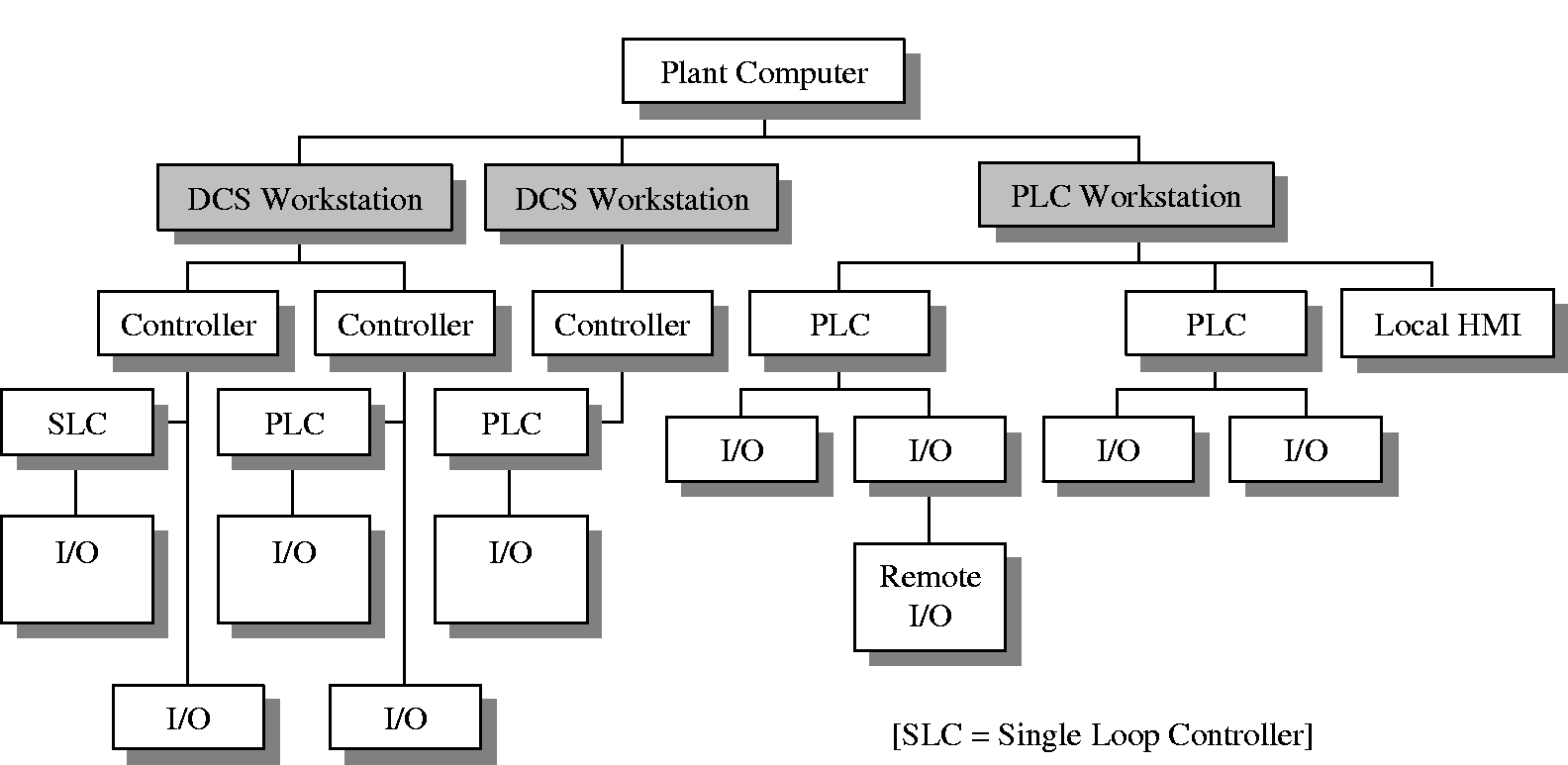


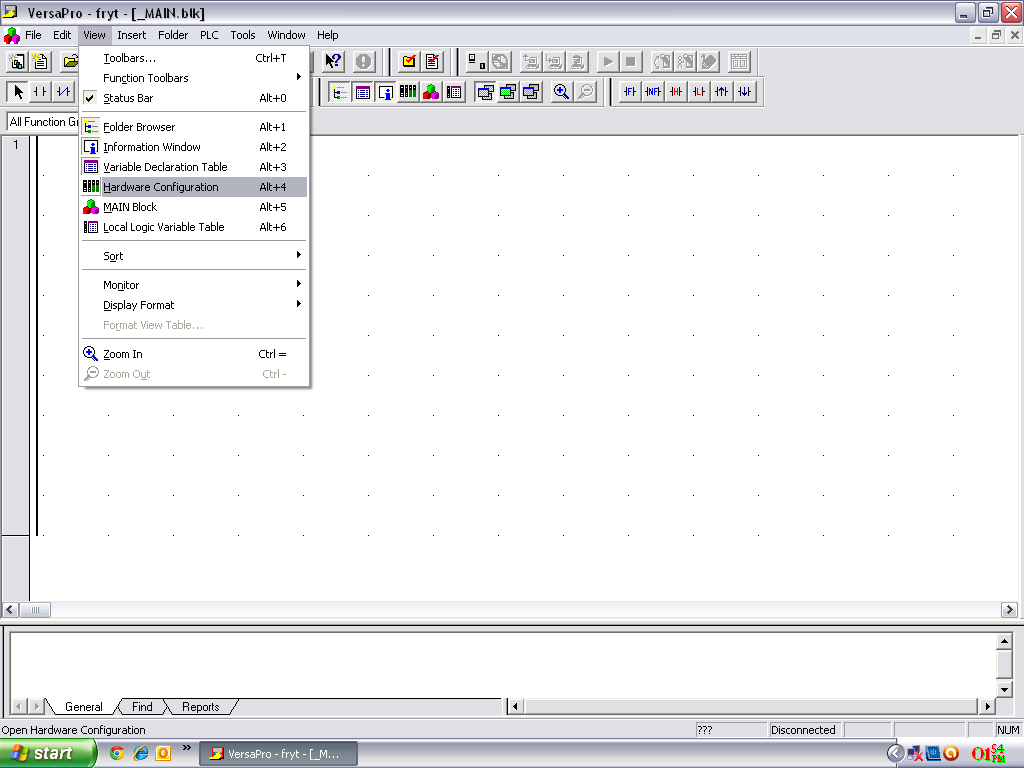
Figure 5.11 **PLC HYBRID SYSTEMS FOR SCADA**

These needs have resulted in various strategies for the design of operator interfaces that are now becoming standardized with regard to ergonomic philosophy. In general, operator input is performed in serial sequences by using a single interface/keyboard/CRT providing all functions at one location within easy reach of a seated operator

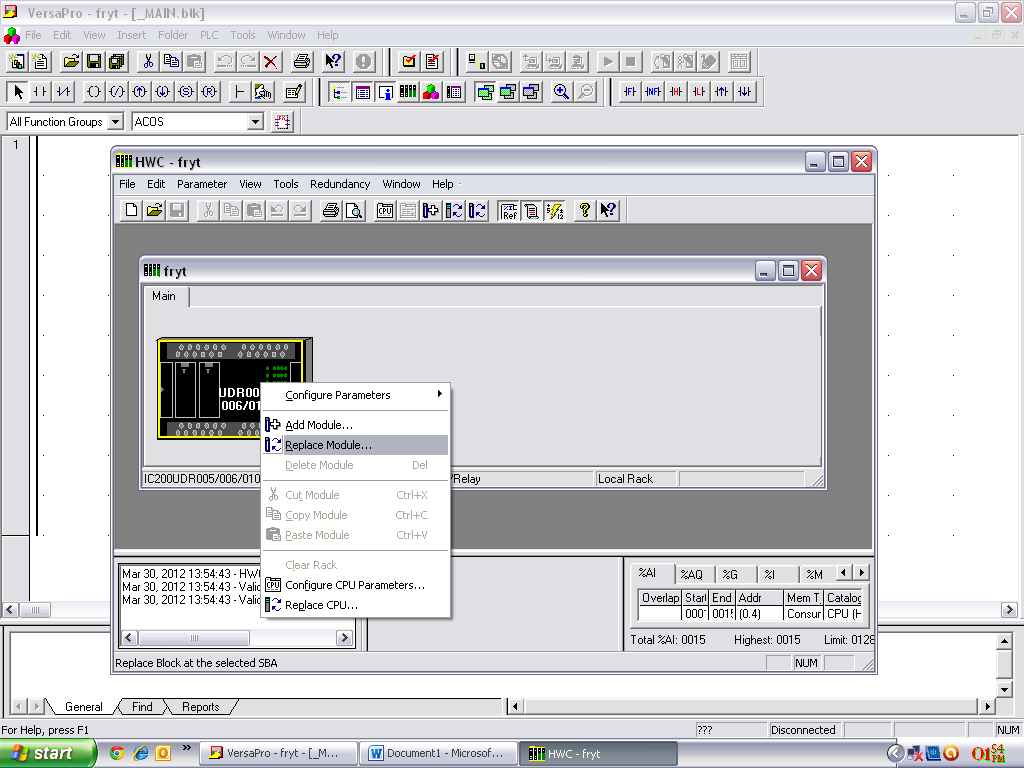
**5.9 DEVELOPMENT OF LADDER PROGRAM:**

The ladder program for PLC-Hybrid system was developed by and presented in anexture-1, the pre-processing on the DATA capture using Lab-VIEW was made using the ladder program developed. This pre-process data then communicated to operator console for further processing and control signal generation.

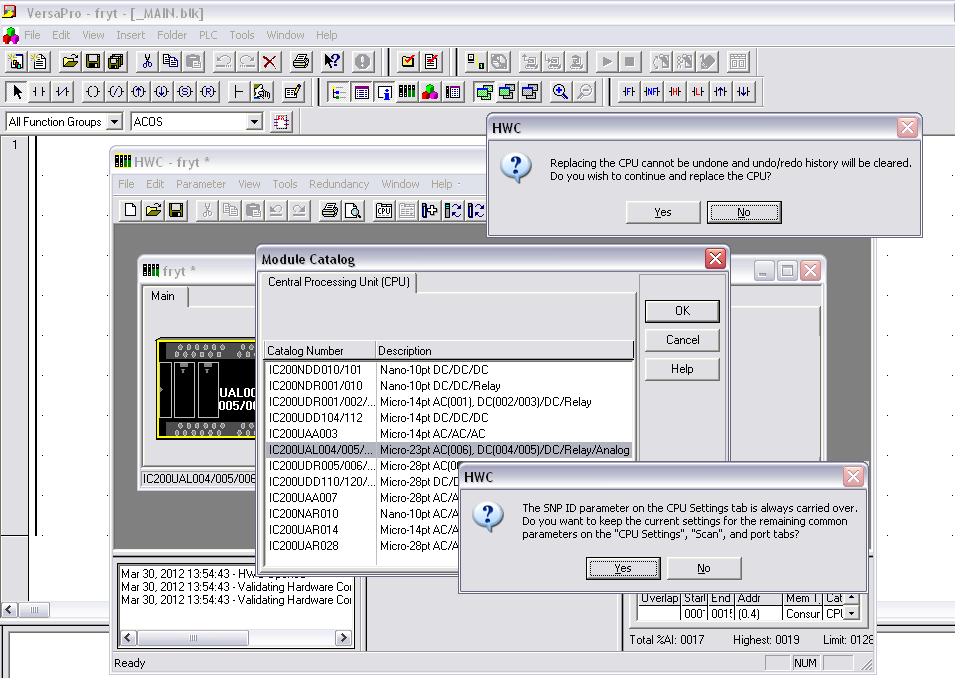
**Step 1: In programming window select view then hardware configuration.**



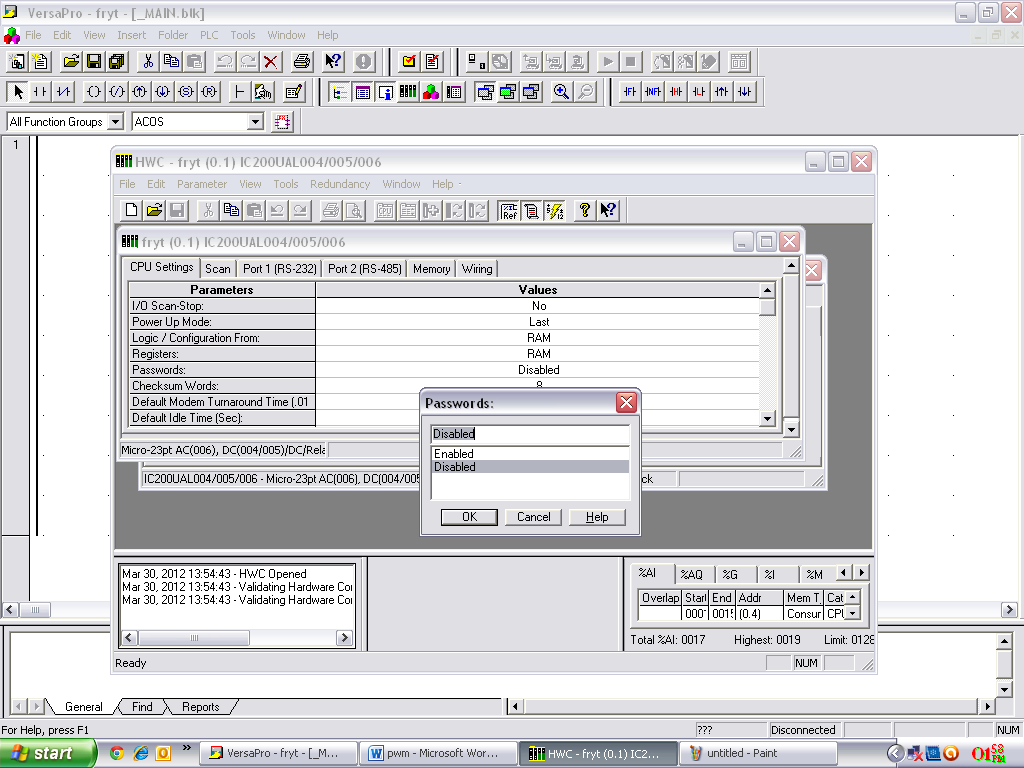
**Step 2: In PLC diagram right click and select replace module.**



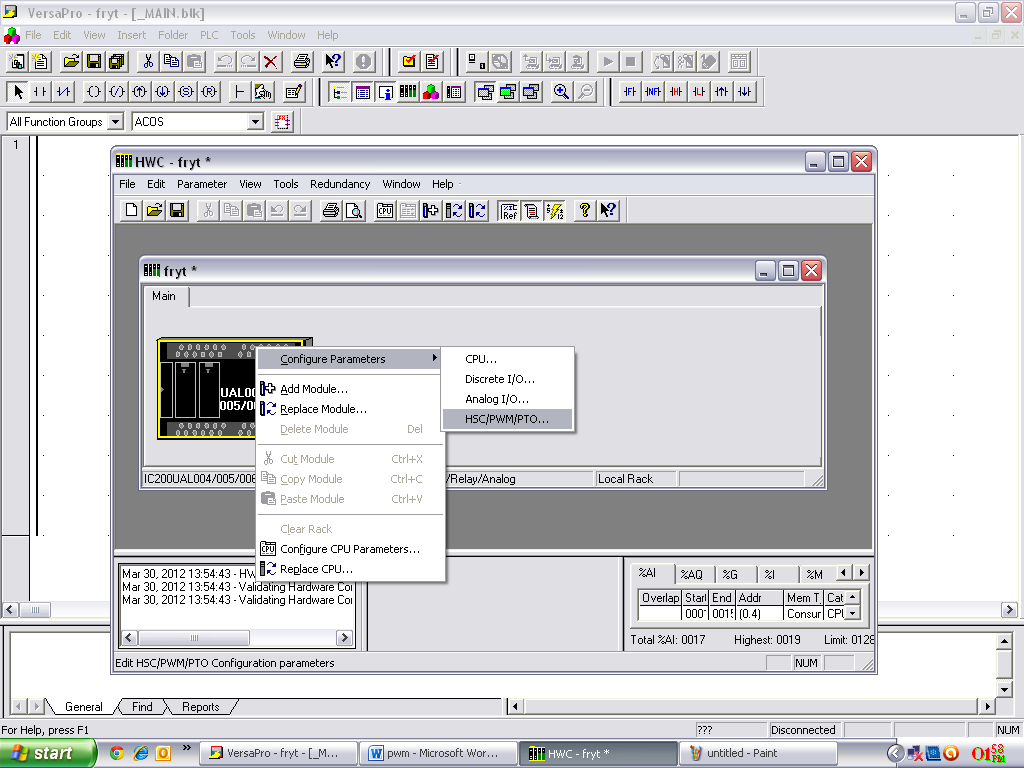
**Step 3: Select the PLC module Number, then select ok then yes, and again yes.**



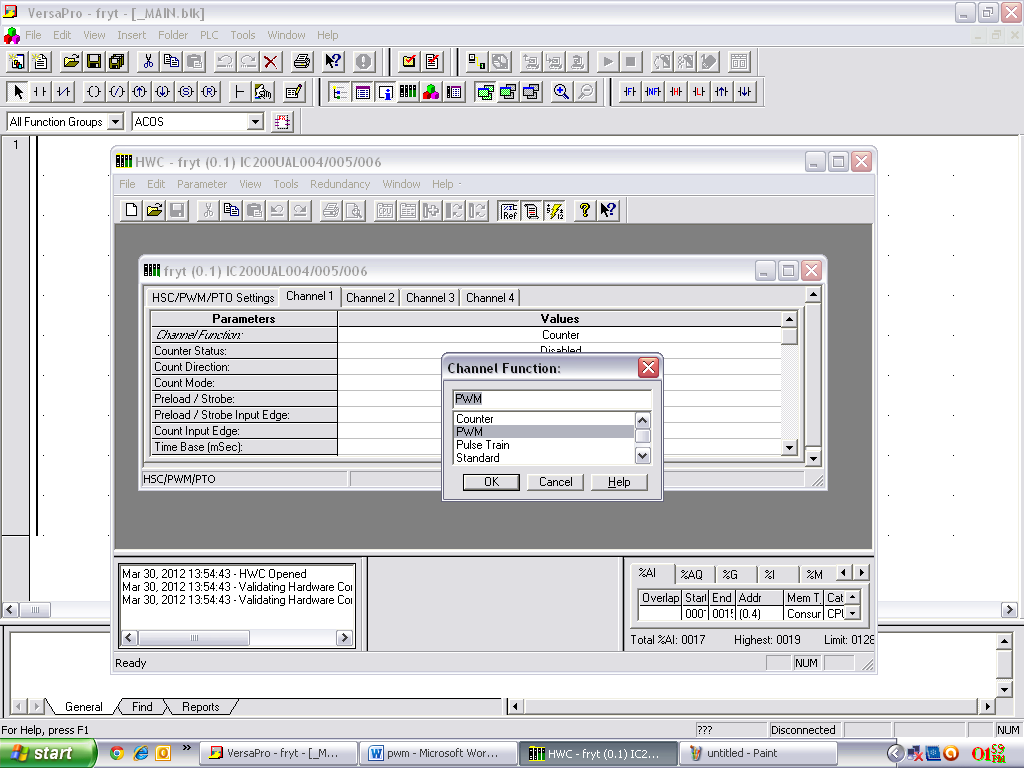
**Step 4: In parameters window DISABLED the passwords.**



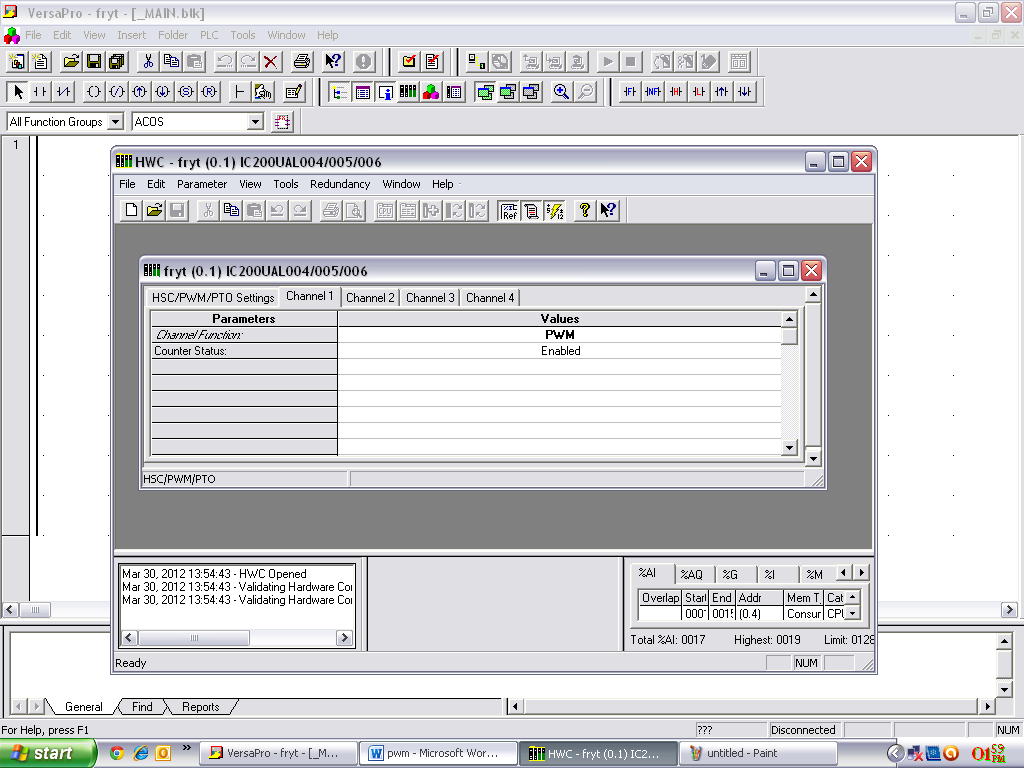
**Step 5: Right click on PLC Diagram select Configure parameters then HSC/PWM.**



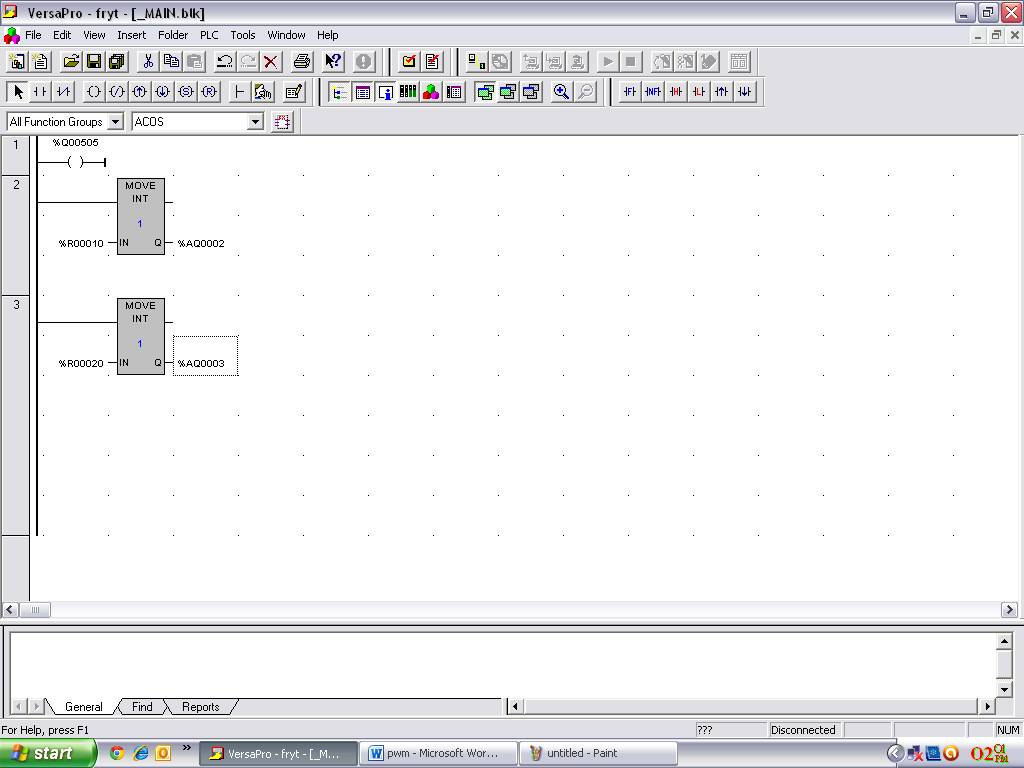
**Step 6: Select channel 1 and select channel function as PWM.**



**Step 7: Now PWM channel enabled and close this Hardware configuration window and save the Changes.**



**Step 8: create the program displayed below. Coil 505q for PWM channel enable, and %aq0002 for Frequency address and %aq0003 for Duty cycle. After creating the program download it. Enter the frequency value in %r000010 (5 – 15 KHZ) and duty cycle in %r00020 (0-10000).**



**5.10 CONCLUSIONS:**

A detail description of Linear Variable Differential Transformer and Flash Light and its implementation guidelines, languages, instructions, programming language and notation using PLCs has been presented. The steps used for PLC Hybrid System for SCADA and development of ladder program is attempted..

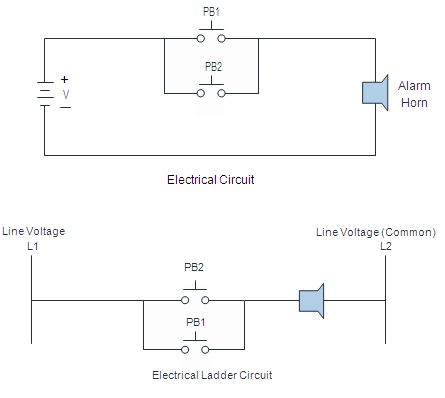
**CHAPTER-VI**

**DISCUSSIONS, CONCLUSIONS, AND**

**FURTHER SCOPE OF WORK**

**DISCUSSIONS:**

The common languages used in Programmable Logic Control instruction is Ladder diagram Language and Boolean Language. In this project I have used only the ladder diagram language. This language is basically a symbolic set of instructions used to creating the controller diagram. These ladder instructions are arranged to obtain the desired control logic that is entered in the memory of the PLC. The main symbols that are used in PLCs programming are shown in table 6.1. In PLC ladder programming continuous path is required for logic continuity, and to energies the output. A ladder diagram can be represented by the circuit representation for an alarm horn that will sound if either of its inputs, push button PB1 or PB2, is ON or depressed as shown in figure 6.1.



**figure 6.1Electrical circuit as a Ladder circuit**

The main function of the ladder logic diagram is to control outputs based on input conditions. This control is accomplished through the use of what is preferred to as a ladder rung. For an output to be activated or energized at least one path left-to-right path of the contact must be closed. A complete closed path is preferred to as having logic continuity. When logic continuity exists in at least one path, the rung condition is said to be true. The rung condition is false if no path has continuity.

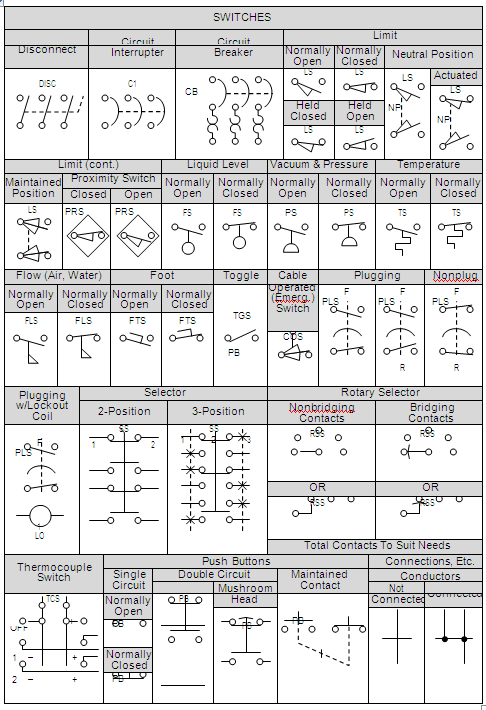


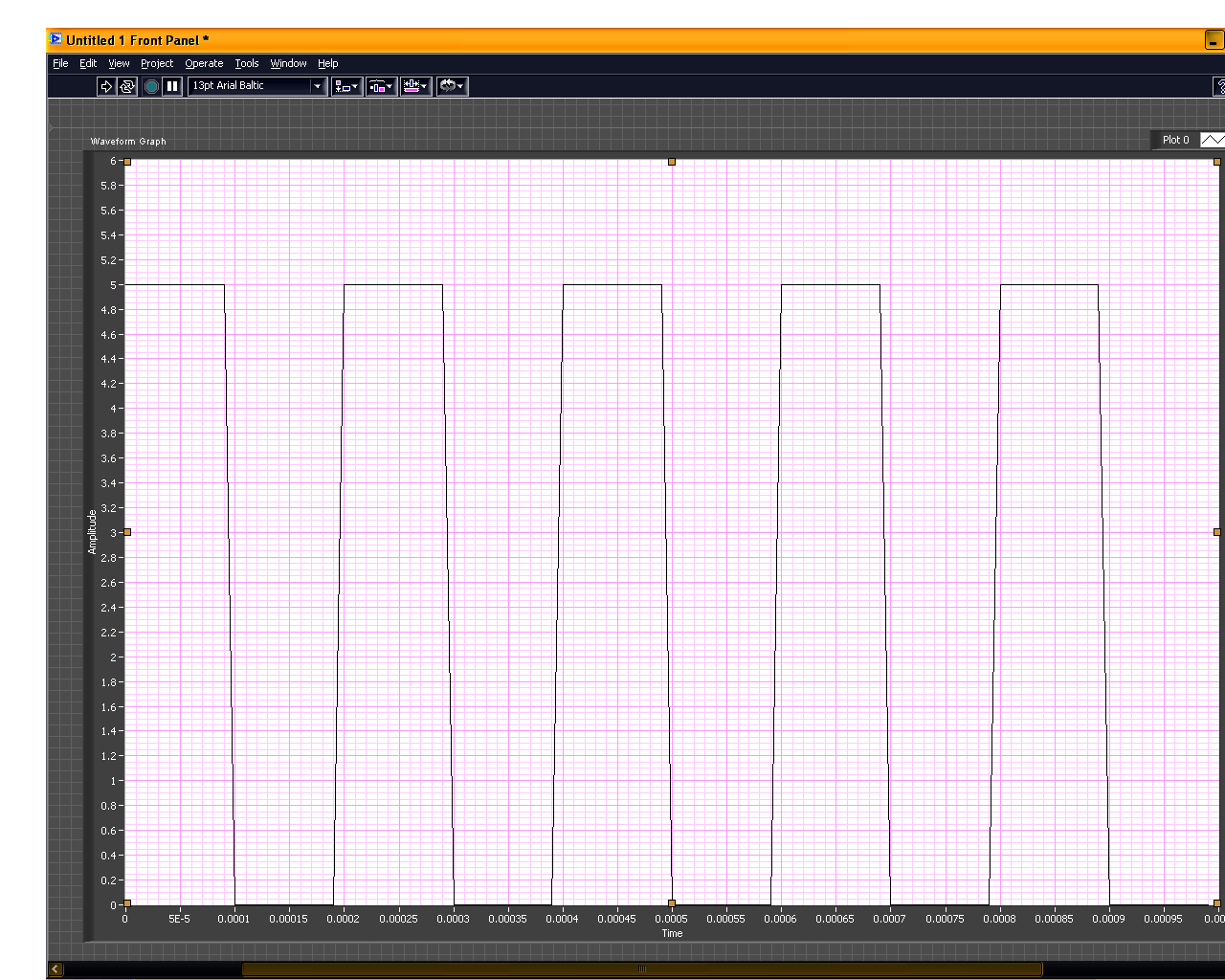
Table 6.1 PLC programming symbols

**6.2 CONCLUSION:**

The use of a Digital Signal Processor (DSP) to evaluate the actual value of an analogue process output and then compute a correction signal has many advantages. DSP does not suffer from the long term drift effect that analogue circuits do. Changes to constants can easily be made without the actual physical change to the circuitry and simply modifying the loaded program or loading a new one can radically alter the mode of control. Realizing PWM techniques for the vast power points of digital controllers. Actually, in most cases DSPs are designed to replace the ON line analogue ones. This explains the continuous approaches to implement digitally the traditional analogue control modes such as PWM actions. A PLC of **GE Fanuc IC 200 UAL 005 of Versa Max is** utilized as a DSP. Nowadays most modern sequential control systems are based on PLCs, which are in fact specialized industrial computers. Thus, one of the targets of this work is to design a PLC program for PWM control algorithm and to develop it to get at the PLC output a PWM signal proportional to the value at the output of the controller. In this case there is no need for a DAC IC nor for a specific power amplifier stage. Here the PWM signal coming from PLC goes to the **PCI-1711 MULTI FUNCTION CARD** by the **PCLD 8710 SCREW TERMINAL BOARD** to the Lab-VIEW which shows the graphical form to another PC, by using the designed interface board, one can exclude the implementation of a high cost proprietary power interface module, this simplifies the circuit and reduces its cost.

PLC manufacturers very often provide the option of analogue I/O and support instant PWM functions for extra cost. Such functions can be used directly by entering the control parameters and constants.

Nevertheless, this feature is only usable if the analogue I/O module is installed. Therefore PWM algorithm has been designed instead of using a ready one. This, also, provides more flexibility to use the program with those PLCs. Designed I/O board satisfies our demands and costs of the I/O module.

****

**Figure 6.2 Pulse graph of output**

**6.3 FURTHER SCOPE OF WORK:**

The simple and complex PLC applications present an area of PLC applications that goes one step beyond these artificial intelligence are of the three as:

Diagnostic,

Knowledge, and

Expert work. 1 INTRODUCTION TO AI SYSTEMS

**Artificial intelligence (AI)** is an area of computer science that has been around for some time. In fact, the conceptual design of AI was first developed in the early 1960s. The definition of artificial intelligence varies among people in the computer industry, making the concept somewhat difficult to perceive and understand. In general, AI can be defined as the subfield of computer science that encompasses the creation of computer programs to solve tasks requiring extensive knowledge. The software programs that form an AI system are developed using the knowledge of an expert person (or persons) in the field where the system will be applied. An exact classification of the types of artificial intelligence systems is very difficult to obtain because of the varying definitions of AI applications. Each of these types of AI systems have similar characteristics, and in fact, the systems evolve sequentially. As the systems become more sophisticated, the size of the database grows and the extent of how the process data is compiled and interpreted increases.

**DIAGNOSTIC SYSTEMS:**

**Diagnostic AI systems** are the lowest level of artificial intelligence implementation. These systems primarily detect faults within an application, but they do not try to solve them. For example, a diagnostic system can diagnose a pump fault by detecting a loss of tank pressure or by reading flow meter values. A diagnostic system reaches a fault conclusion through inferring techniques based on known facts (knowledge) introduced into its detection system. This type of AI is used in applications that have a small knowledge and database structure. Diagnostic systems typically make GO or NO GO decisions and sometimes provides information about the fault’s probable cause.

**KNOWLEDGE SYSTEMS:**

A **knowledge AI system** is, in reality, an enhanced diagnostic system. Knowledge systems not only detect faults and process behaviors based on resident knowledge, but also make decisions about the process and/or the probable cause of a fault.

**EXPERT SYSTEMS:**

An **expert AI system** is the top of the line in AI type applications; it has all of the capabilities of a knowledge system and more. An expert system provides an additional capability for examining process data using statistical alysis. The use of statistical data analysis lets the system predict outcomes based on current process assessments. The outcome prediction may be a decision to continue a process in spite of fault detection. The knowledge introduced into an expert system is more complex than in the other types of AI systems; therefore, expert systems generate more data verification (feedback information). The decisions made by expert systems also require more sophisticated software programming, since their decision trees involve more options and attributes. The implementation of an expert AI system requires not only extra programming effort but also more hardware capability. The total system will need more transducers to check other transducers and field devices. Moreover, the PLC will require the use of two or more processors to implement the control and intelligence programs. The speed of the system must also be fast so that it can operate in real time. A typical artificial intelligence system consists of three primary elements:

• A global database

• A knowledge database

• An inference engine

Figure 6-1 shows a block diagram of an AI system’s architecture. As the figure illustrates, the AI system must receive its knowledge from a person who thoroughly understands the process or machine being controlled. This individual, called the ***expert***, must communicate all information about system maintenance, fault causes, etc. to the ***knowledge engineer***, the person responsible for system implementation. The process of gathering data from the expert and transmitting it to the knowledge engineer is known as ***knowledge acquisition***.

**6.4 GLOBAL DATABASE:**

The **global database** section of an AI system contains all of the available information about the system being controlled. This information mainly deals with the input and output data flow from the process. The global database resembles a storage area where information about the process is stored and updated. The AI system can access the data in this area at any time to perform statistical analysis on historical process control data, which in turn can be used to implement AI decisions. The global database resides in the memory of the control system implementing the artificial intelligence. If a PLC is used to implement a diagnostic AI system, the global database will most likely be located in the storage area of the PLC’s data table. If a PLC is used in conjunction with a computer or computer module to implement an AI system, then the global database will probably be located in the computer, the computer module’s memory, or a hard disk storage subsystem.

**6.5 KNOWLEDGE DATABASE:**

The **knowledge database** section of an AI system stores the information extracted from the expert. Like the global database, this database includes information about the process; however, it also stores information about faults, along with their probable causes and possible solutions. Moreover, the knowledge database stores all of the rules governing the AI decisions to be made. The more involved the AI system, the larger the knowledge database. Accordingly, the knowledge database of a diagnostic system is less complex than that of a knowledge system; likewise, the knowledge database of a knowledge system is less sophisticated than that of an expert system. The knowledge database is stored in the section of the system memory that implements the AI techniques.

**INFERENCE ENGINE**

An AI system’s **inference engine** is the place where all decisions are made. This section uses the information stored in the knowledge database to arrive at a decision and then execute all applicable rules and decisions about the process. The inference engine also constantly interacts with the global database to examine and test real-time and historical data about the process. The inference engine usually resides in the main CPU (i.e., the one that performs the AI computations). However, in a PLC-based system, the inference engine may or may not be stored in the main CPU, depending upon the system’s complexity (i.e., diagnostic, knowledge, or expert).

**6.6 KNOWLEDGE REPRESENTATION:** 16-4 KN

**Knowledge representation** is the way the complete artificial intelligence system strategy is organized that is, how the knowledge engineer represents the expert’s input. This representation is stored in the knowledge database of the AI system. In rule-based knowledge representation, the expert’s knowledge is transformed into IF and THEN/ELSE statements, which facilitate actions and decisions. All control systems that implement artificial intelligence, whether diagnostic, knowledge, or expert, execute the control strategy (via the software control program) in the inference engine. Whenever a decision must be made due to a fault or another situation, the inference engine refers to the knowledge representation to obtain a decision about the probable cause. This decision is the result of a group of software subroutines. Once the knowledge database reaches an AI decision, the inference engine will determine the appropriate course of action. Depending on the control strategy formulation (main program), the inference engine may, at this time, refer to the global database to verify data or obtain more information.

**6.7 RULE-BASED KNOWLEDGE REPRESENTATION:**

**Rule-based knowledge representation** defines how the expert’s knowledge is used to make a decision. The rules used are either ***antecedent***(IF something happens) or ***consequent***(THEN take this action). For example, to the question, What causes the volume in the tank to drop?, the expert may respond with the answer, a malfunctioning tank system. The knowledge engineer may implement this information as the following rule: IF the volume is less than the set point, THEN annunciate a system malfunction due to a loss of volume. Rules can be as long and complex as needed for the process, and they usually define the involvement of the AI system.

**6.8 FUZZY LOGIC AND PLCS:**

In further investigated to arrive at a complete formal rule. The inference engine can use the consequents derived in the knowledge representation to obtain a better definition of the problem’s cause. Knowledge and expert AI systems use this process to provide advanced decision making capabilities.

Fuzzy logic provides PLCs with the ability to make “reasoned” decisions about a process. In this chapter, we will introduce you to the basics of fuzzy logic, including fundamental concepts and historical origins.DTION TO FUZZY LOG**Fuzzy logic** is a branch of artificial intelligence that deals with reasoning algorithms used to emulate human thinking and decision making in machines. These algorithms are used in applications where process data cannot be represented in binary form. For example, the statements “the air feels cool” and “he is young” are not discrete statements. They do not provide concrete data about the air temperature or the person’s age (i.e., the air is at 65 F or the boy is 12 years old). Fuzzy logic interprets vague statements like these so that they make logical sense.

**REFERENCES**

1. Instrument Engineers hand book for Process Control, CRC publication, third edition, LIPTAK.
2. American Institute of Chemical Engineers. Draft “Guidelines for safe automation of chemical processes”. Center for Chemical process Safety, process control safety committee, 1990.
3. B.W. Balls and p.Gruhn. “Design considerations for high-risk safety systems “ISA Transactions, vol.30, no. 4 1991, pp. 71-77.
4. H.Berger. Automatization with S5 -115U. Siemens 1987.
5. Comity European de Normalization. “Safety of machinery. Safety related parts of control systems. Part1: General principles for design”. Final Draft preens 954-1.Annex 3, April 1995.
6. Ph.charpentier. “conception d’un dispositive a microprocessors sure de fonctionnement: Regales ET methods”. Institute National de Recherché ET de Securities, Project de publication dams electronique, 1991.
7. DIN. “Principles for computers in safety related computer systems”. Standard 0801, January 1990.
8. L.Dombrowski, W.johnson and R.McCall. “Separation of interlocking and regulatory control in programmable electronic systems.” ISA Transactions, vol.30, no. 4, 1991, pp. 19-25.
9. J.Hill. “Fail-safe circuits”. Digital Computer Laboratory, University of Illinois, Urbana. Report 120, June 1962.
10. IEC. Draft “Functional safety of programmable electronic systems: generic aspects”. Technical Committee no .65, 1990.
11. IEEE. “Standard for software verification& validation”.1986.
12. ISA. Draft “Programmable electronic systems for use in safety applications”.SP84 Committee, 1990.
13. B.W.Johnson Design and analysis of fault tolerant digital systems.Addeson-wesley, 1989.
14. T.E. Kassel. Understanding and using programmable controller’s prentice-Hall, 1986.
15. E.J. Mccluskey.”Hardware fault tolerance. “Center for Reliable computing, Stanford University, 1986.
16. G. Michel programmable Logic Controllers. Marcombo1990.
17. National Fire protection Association, 85 D. Burnet management system logic. April 1988.
18. J-j.paques. “The elements of safety for using programmable controllers. ISA Transactions, vol.30, no. 4, 1991, pp 67-70.
19. M.Ranzini. “Los PLC en la automatizacion basic del process”. Automatizacion integrate &revisit de robotics, no. 70, July 1992.
20. D.A. Reynolds and Gamete. “Fault detection capabilities of alternating logic.” IEEE Trans. Computers, vol. 27. No...12, December 1978, pp.1093-1098.
21. A. Schweitzer and J.P.Gerardin. “Methods permitting d, ameliorer le novae de security des systems a logique programmed. Logique programmed et security: Enounce du problem”. Electroniques ET industries, no.12, November 1984, pp 51-56.
22. A. Schweitzer and jip. Gerard in. “Methods permitting d’ ameliorer leniveau de security des systems a logique programmed. Logique programmed ET security. Les solutions”. Electronique, Techniques ET industries, no 13, December 1984, pp.39-49.
23. Y. Tourna. “Realization of fail-safe sequential machines using flip-flops”. In proceedings of Kyoto international conference on circuit and system Theory, September 1970.
24. Y. Thomas, Y.Ohyama, and R. Sakai. “Realization of fail-safe sequential machines by using a k-out-of-n code”.IEEE Transactions on Computers, vol. 20, no. 11, November 1971, pp. 1270-1275.
25. J.P. Vaulting and M. Collier. “Les functions dynamism, autocontroble et security des systems logiques”. Electronique industrielle, no. 60, November 1983 pp. 89-98.
26. Library of Congress Cataloging-in-Publication Data Bryan, L.A. ,Programmable controllers: theory and implementation/L.A. Bryan, E.A. Bryan.,2nd ed.p. cm. Includes index. ISBN 0-944107-32-X,Title.TJ223.P76B795 1997