PLC and its Programming

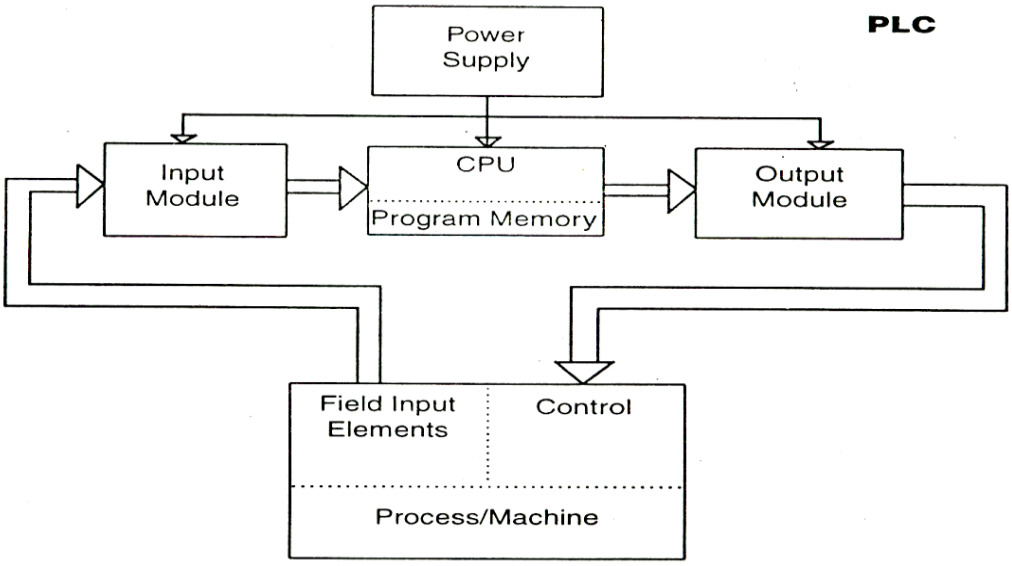
A programmable logic controller (PLC) is a digital computer used for automation of industrial processes, such as control of machinery on factory assembly lines. Unlike general-purpose computers, the PLC is designed for multiple inputs & output arrangements, extended temperature ranges, immunity to electrical noise, resistance to vibration and impact. Programs to control machine operation are typically stored in battery-backed or non-volatile memory. A PLC is an example of a real time system since output results must be produced in response to input conditions within a bounded time, otherwise unintended operation will result [9]. PLC has following specific features suited to industrial control…

- PLC is rugged and noise immune equipment.
- PLC have modular plug in construction, allowing easy replacement / editing of input-output units.
- Standard low-level and high-level programming is possible.
- Ease of programming and reprogramming in plant both in runing &/ editing mode.
- Communication port for interfacing with other PLCs and PCs.
- Ease of maintenance.
- Greater life and reliability.
- Energy saving.
- Shorter project time.
- Easier storage archiving and documentation.

As shown in figure 55, the PLC consists of following:

1. Input Modules
2. CPU with processor and program memory
3. Output Modules
4. Bus system
5. Power supply

Figure 55: Building blocks of PLC system



DCS is a system having the controller, HMI integrated with control / supervision capability, the network integration, and usually many choices of Asset Management and Historian software, often with ERP integration. Some DCSs use PLCs as their controllers, such as ControlLogix / FlexLogix / ProcessLogix etc. from Rockwell, and PCS7 from Siemens.
In this project the function of process controller and I/O is handled by RSLogix5000 PLC system of Rockwell. RSViewSE is used as the HMI software. RSLinx Professional software is used as interface between HMI and PLC system.

Software development wise we have two main objectives:
[1] Development of Ladder Logic to implement the control philosophy using RSLogix5000 software platform **(this chapter)**.
[2] Development of HMI displays, trends etc using RSViewSE software platform – (**Chapter 6**)

5.1 **Distributed Control System**

A **distributed control system** (DCS) refers to a *control system* usually of a *manufacturing system*, *process* or any kind of *dynamic system*, in which the controller elements are not central in location (like the brain) but are distributed throughout the system with each component sub-system being controlled by one or more controllers [10]. The entire system of controllers is connected by networks for communication and monitoring. Distributed Control Systems are dedicated systems used to

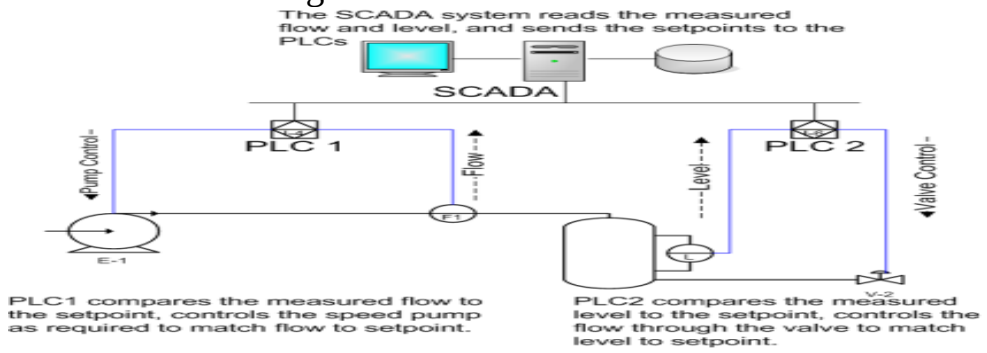
control manufacturing processes that are continuous or batch-oriented, petrochemicals, pharmaceuticals, food & beverage manufacturing, cement production, steelmaking, and papermaking etc.

DCSs are connected to sensors and actuators and use setpoint control to control the flow of material through the plant. The most common example is a setpoint *control loop* consisting of a pressure / flow sensor, controller, and *control valve*. Pressure or flow measurements are transmitted to the controller, usually through the aid of a signal conditioning Input/Output (I/O) device. When the measured variable reaches a certain point, the controller instructs a valve or actuation device to open or close until the fluidic flow process reaches the desired setpoint. Large oil refineries have many thousands of I/O points and employ very large DCSs. Processes are not limited to fluidic flow through pipes, however, and can also include things like paper machines and their associated variable speed drives and motor control centers, cement kilns, mining operations, ore processing facilities, and many others.

5.2 **Relationship between PLC & SCADA**

The term SCADA usually refers to centralized systems which monitor and control entire sites, or complexes of systems spread out over large areas (on the scale of kilometers or miles). Most site control is performed automatically by remote terminal units ("RTUs") or by programmable logic controller ("PLCs"). Host control functions are usually restricted to basic site overriding or *supervisory* level intervention [11]. For example, a PLC may control the flow of cooling water through part of an industrial process, but the SCADA system may allow operators to change the set points for the flow of cooling water, and enable alarm conditions for filling / pressurization / feeding / depressurization condition and high temperature at heat exchanger, to be displayed and recorded. The feedback control loop passes through the RTU or PLC, while the SCADA system monitors the overall performance of the loop as shown in figure 56.

Figure 56: SCADA using PLC



5.3 **Digital and Analog Signals**

When the argon tank level control scheme is implemented using only digital signals, the PLC has two digital inputs from float switches (tank empty and tank full). The PLC uses a digital output to open and close the inlet valve into the tank. When the argon level drops enough so that the tank empty float switch is off (down), the PLC will open the valve to let more argon in. Once the argon level rises enough so that the tank full switches is on (up), the PLC will shut the inlet to stop the argon from overflowing. While an analog system might use a argon pressure and level transmitter.

While an analog system might use a pressure and level transmitter and a software PID block and 4-20 mA analog output to position the level control valve incrementally.

5.3.1 Digital or discrete signals behave as binary switches, yielding simply an On or Off signal (1 or 0, True or False, respectively). Pushbuttons, limit switches, and photoelectric sensors are examples of devices providing a discrete signal. Discrete signals are sent using either voltage or current, where a specific range is designated as *On* and another as *Off*.

For example, a PLC might use 24 V DC I/O, with values above 22 V DC representing *On*, values below 2VDC representing *Off*, and intermediate values undefined. Initially, PLCs had only discrete I/O.

5.3.2 Analog signals are like volume controls, with a range of values between zero and full-scale. These are typically interpreted as integer values (counts) by the PLC, with various ranges of accuracy depending on the device and the number of bits available to store the data. As PLCs typically use 16-bit signed binary processors, the integer values are limited between -32,768 and +32,767. Pressure, temperature, flow, and level are represented by analog signals. Analog signals use *voltage or current* with a magnitude proportional to the value of the process signal. For example, an analog 4-20 mA or 0 - 10 V input would be converted into an integer value of 0 - 32767.

Note: Current inputs are less sensitive to electrical noise (i.e. from welders or electric motor starts) than voltage inputs.

5.4 **System scale**

A fixed PLC will have a fixed number of connections built in for inputs and outputs. Typically, expansion slots are available if the base model does not have enough I/O.

Modular PLCs have a chassis (also called a rack), into which is placed modules with different functions. The processor and selection of I/O modules is customised for the particular application. Several racks can be administered by a single processor, and may have thousands of inputs and outputs. A special high speed serial I/O link is used so that racks can be distributed away from the processor, reducing the wiring costs for large plants.

Another very important thing is need of opto isolation for modular PLC. While utility of opto isolation is to separate power supply from CPU. As normally I/P supply is made at 210-230V AC while O/P requirement at 5V DC. So opto isolation is required for safety of CPU.

PLCs used in larger I/O systems may have peer-to-peer (P2P) communication between processors. This allows separate parts of a complex process to have individual control while allowing the subsystems to co-ordinate over the communication link. These communication links are also often used for HMI (Human-Machine Interface) devices such as display units with keypads or PC-type workstations. Some of today's PLCs can communicate over a wide range of media including RS-485, Coaxial, and even Ethernet for I/O control at network speeds up to 100 Mbit/s.

Note: In this project Ethernet serves as the communication link between the PLC and the HMI station. Module 1756-ENBT on the PLC chasis serves as the ethernet interface of the PLC.

5.5 **Programming**

Recently, the International standard *IEC 61131-3* has become popular. *IEC 61131-3* currently defines five programming languages for programmable control systems:
[a]FBD (*Function block diagram*), graphical
[b]LD (*Ladder diagram*), graphical
[c]ST (*Structured text*, similar to the *Pascal programming language*), textual
[d]IL (*Instruction list*, similar to *assembly language*), textual and
[e]SFC (*Sequential function chart*), has elements to organize programs for sequential and parallel control processing.

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

From above mentioned language in this project, we followed by Ladder diagram approach using RSLogix5000 for programming part of controller.

5.6 **To Start with Ladder Logic**

Ladder logic is a philosophy of drawing electrical logic schematics. It is now a graphical language very popular for programming. It was originally invented to describe logic made from *relays*. The name is based on the observation that programs in this language resemble ladders, with two vertical "rails" and a series of horizontal "rungs" between them.

A program in ladder logic, also called a **ladder diagram**, is similar to a schematic for a set of *relay circuits*. An argument that aided the initial adoption of ladder logic was that a wide variety of engineers and technicians would be able to understand and use it without much additional training, because of the resemblance to familiar hardware systems.

Note: This argument has become less relevant given that most ladder logic programmers have a software background in more conventional *programming languages*, and in practice implementations of ladder logic has characteristics; such as sequential execution and support for control flow features that make the analogy to hardware somewhat imprecise.

Ladder logic is widely used to program PLCs, where sequential control of a process or manufacturing operation is required. Ladder logic is useful for simple but critical control systems, or for reworking old *hardwired relay circuits*. As programmable logic controllers became more sophisticated it has also been used in very complex automation systems.

Manufacturers of programmable logic controllers generally also provide associated ladder logic programming systems. Typically, the ladder logic languages from two manufacturers will not be completely compatible; ladder logic is better thought of as a set of closely related programming languages rather than one language (the *IEC 61131-3* standard has helped to reduce unnecessary differences, but translating programs between systems still requires significant work). Even different models of programmable controller within the same family may have

different ladder notation such that programs cannot be seamlessly interchanged between models.

Ladder logic can be thought of as a rule-based language, rather than a procedural language. A "rung" in the ladder represents a rule. When implemented with relays and other electromechanical devices, the various rules "execute" simultaneously and immediately. When implemented in a programmable logic controller, the rules are typically executed sequentially by software, in a continuous loop (scan). By executing the loop fast enough, typically many times per second, the effect of simultaneous and immediate execution is relatively achieved to within the tolerance of the time required to execute every rung in the "loop" (the "scan time"). It is somewhat similar to other rule-based languages, like *spreadsheets or SQL*. However, proper use of programmable controllers requires understanding the limitations of the execution order of rungs.

Ladder logic has "contacts" that "make" or "break" "circuits" to control "coils." Each coil or contact corresponds to the status of a single bit in the programmable controller's memory. Unlike electromechanical relays, a ladder program can refer any number of times to the status of a single bit, equivalent to a relay with an indefinitely large number of contacts.

So-called "contacts" may refer to physical ("hard") inputs to the programmable controller from physical devices such as pushbuttons and limit switches via an integrated or external input module, or may represent the status of internal storage bits which may be generated elsewhere in the program.

Each rung of ladder language typically has one coil at the far right. Some manufacturers may allow more than one output coil on a rung.

--( )-- a regular coil, true when its rung is true
--(\)-- a "not" coil, false when its rung is true
--[ ]-- A regular contact, true when its coil is true (normally false)
--[\]-- A "not" contact, false when its coil is true (normally true)

The "coil" (output of a rung) may represent a physical output which operates some device connected to the programmable controller, or may represent an internal storage bit for use elsewhere in the program.

5.7 **Limitations and Successor Languages**

Ladder notation is best suited to control problems where only binary variables are required and where interlocking and sequencing of binary is the primary control problem. Since execution of rungs is sequential within a program and may be undefined or obscure within a rung, some logic *race conditions* are possible which may produce

unexpected results; complex rungs are best broken into several simpler steps to avoid this problem. Analog quantities and arithmetical operations are clumsy to express in ladder logic and each manufacturer has different ways of extending the notation for these problems. There is usually limited support for arrays and loops, often resulting in duplication of code to express cases which in other languages would call for use of indexed variables.

As *microprocessors* have become more powerful, notations such as sequential function charts and function block diagrams can replace ladder logic for some limited applications. Very large programmable controllers may have all or part of the programming carried out in a dialect that resembles *BASIC* or *C* or other *programming language* with bindings appropriate for a real-time application environment.

5.8 **Instructions**

Major instructions are used in this project during ladder logic design as follows [12 - 13]…

- **CPT**: Compute comes under ‘Compute / Math Instructions’.
- **DIV**: Divide comes under ‘Compare Instructions’.
- **GEQ**: Greater than & Equal comes under ‘Compare Instructions’.
- **GSV**: Get System Value comes under ‘Input / Output instructions’.
- **JSR**: Jump Sub-Routine comes under ‘Program Control Instructions’.
- **LEQ**: Less than or Equal to comes under ‘Compare Instructions’.
- **LIM**: Limit comes under ‘Compare Instructions’.
- **MOV**: Move comes under ‘move / logical instructions’.
- **NEQ**: ‘Not Equal to’ comes under ‘Compare Instructions’.
- **PID**: proportional Integral Derivative comes under ‘Special Instructions’.
- **TON**: Timer On Delay comes under ‘Timer and Counter instructions’.
- **Holding contacts**

For any further details of Instructions used during Controller’s Ladder Logic design using ControlLogix ControlNet RSLogix5000 may refer “Appendix H”.

5.9 **Ladder Logic Design**

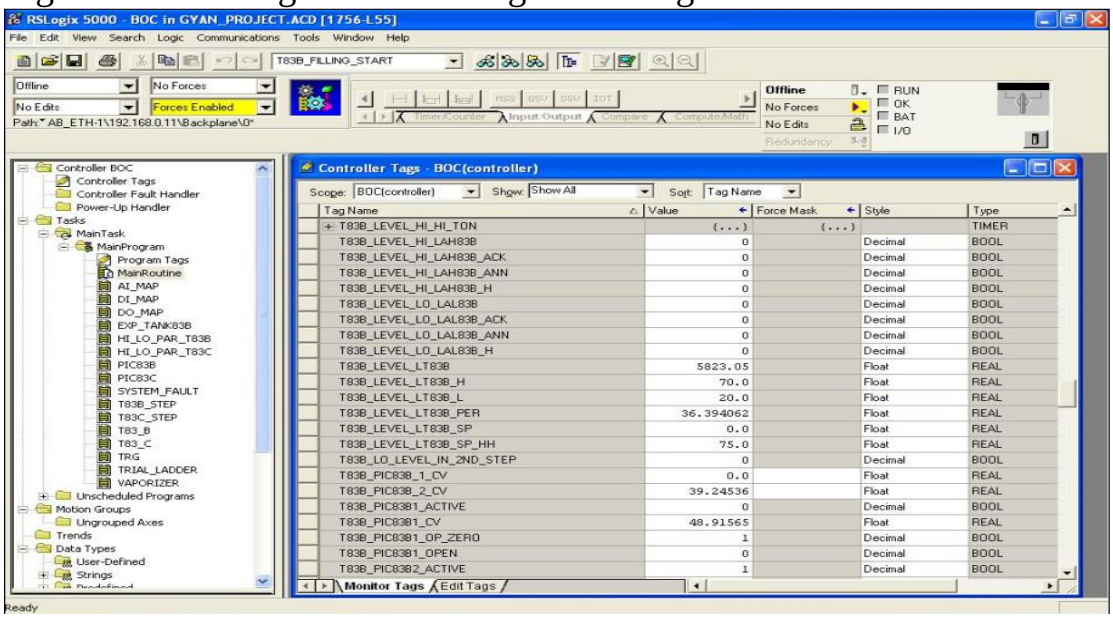
After identifying Logix5000 system, RSLogix5000 software components, creating, organizing and communicating a new RSLogix5000 project the configuration of Logix5000 controller has implemented as mentioned already in 'section 4.2'. In this project RSLogix5000 software is used for ladder logic based programming in OP-2, Bhilai Steel Plant. It is considered a high security area as a process unit from management's point of view. So each & every information regarding to Controller's point of view is not going to provide here. Procedure for creating, organizing, configuring, and programming of project for controller in detail might be referred from reference 3 or visited to RSLogix5000-BOC in GYAN_PROJECT.ACD [1756-L55] with due permission of Bhilai Steel Plant. There are four major and important steps from controller's ladder logic design point of view as shown in figure 57- 60.

In this project for PLC using RSLogix5000 the created tags are categorized as Controller Tags and Program Tags of two types…

- H/W Tags
- S/W Tags

For any further details of the tags are used Controller ladder logic using RSLogix5000 might be referred from "Appendix G – The Instrument Lists of PLC System".

Figure 57: Creating Controller Tags in RSLogix5000

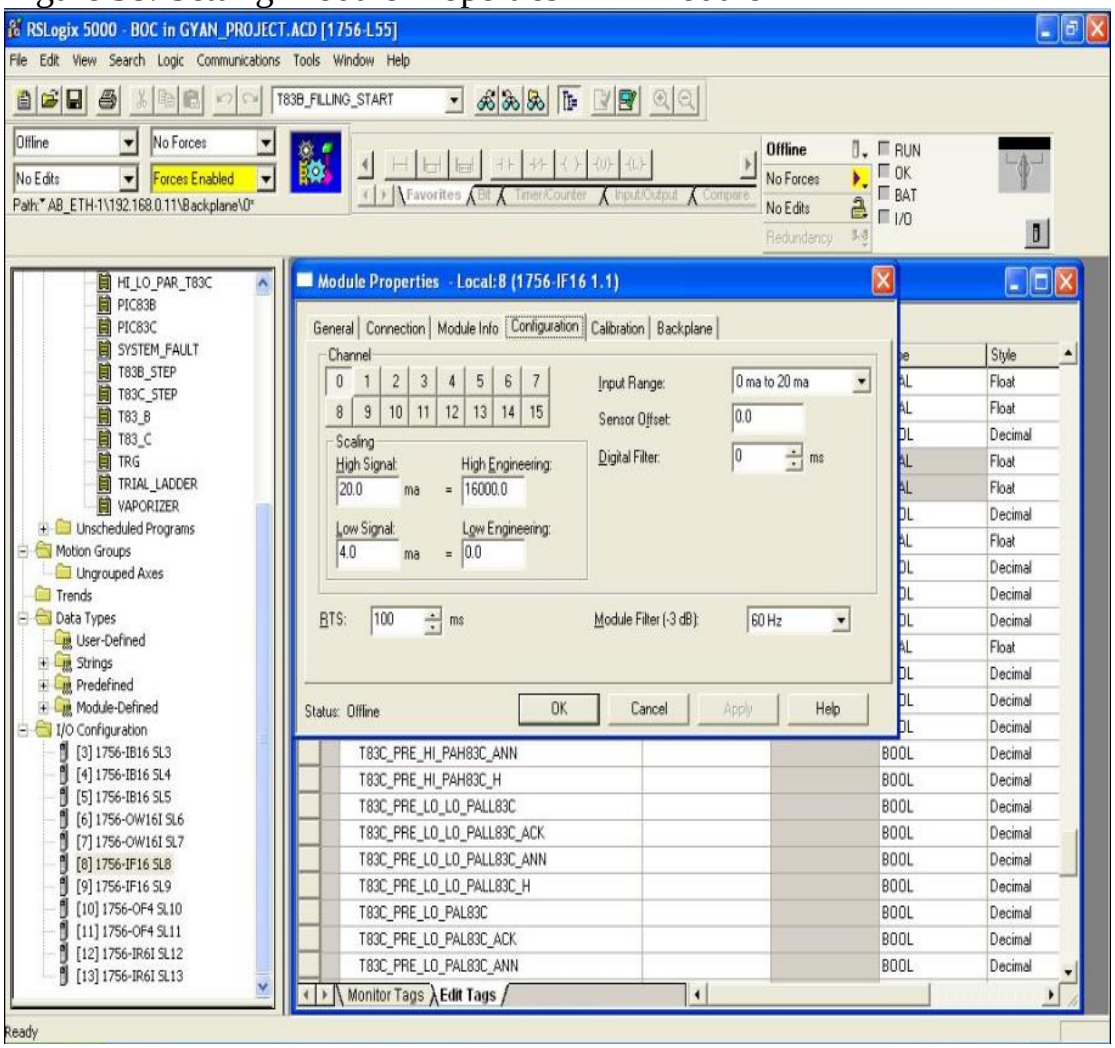


Once tags are created in PLC based controller using RSLogix5000 software, the next major job is to setting module properties for different analog or digital I/O modules. As mentioned in proposed system architecture there are four types of I/O modules categorized into…

- Analog Input Module (AI)
- Analog Output Module (AO)
- Digital Input Module (DI)
- Digital Output Module (DO)

For an example configuring of AI module might be referred from figure 58.

Figure 58: Setting Module Properties – AI Module



As mentioned in figure 57 and figure 58, once tags are created and setting individual module properties the RSLox5000 is ready for programming. The programming of individual modules and main task of controller is reflects as shown in figure 59-60.

Figure 59: Programming AI_Map – RSLogix5000

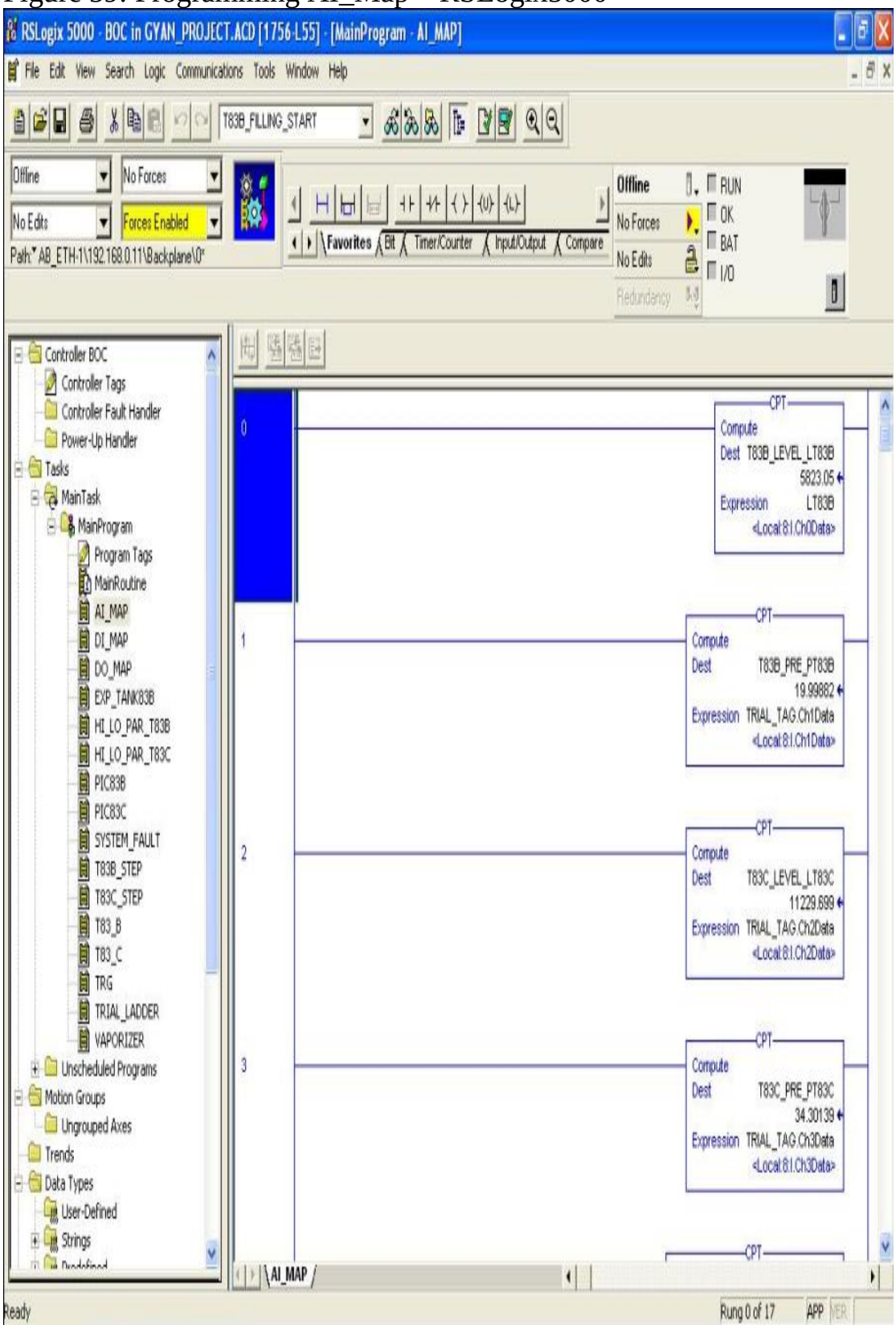
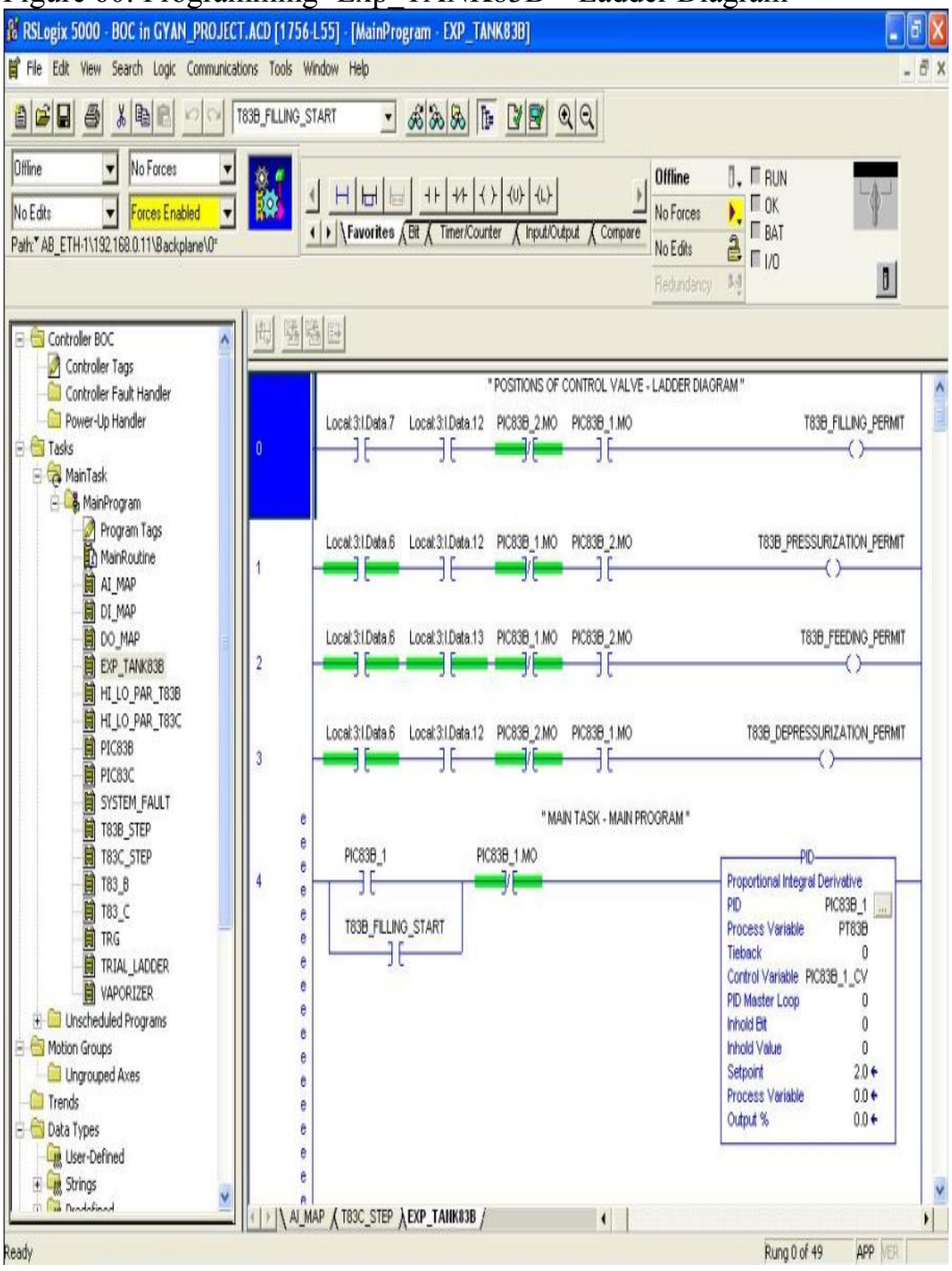


Figure 60: Programming 'Exp_TANK83B – Ladder Diagram'



Similar steps needs to follow for programming of stand by Tank T-83C.

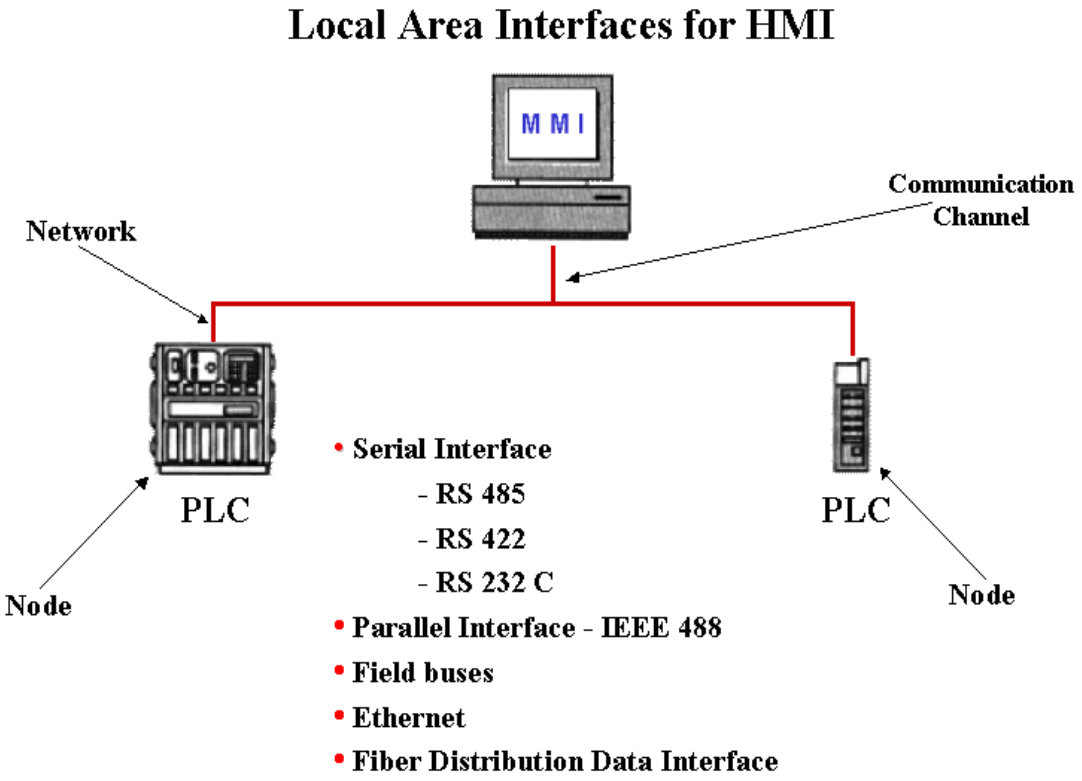
Note: This part of project is implemented in OP-2, Bhilai Steel Plant. For any further detail OP-2 of Bhilai Steel Plant might be visited with due permission of BSP management and can be observed directed by RSLogix5000 based controller to field sensors for augmentation of argon.

Chapter 6

Designing of HMI

As described in section 4.5 about Configuring RSViewSE, it is used to develop application software (designed by Rockwell automation) for interfacing HMI with Controller (using RSLogix5000). Which helps to establish communication interlink ONLINE with microprocessor L55M12. Generalized model of interface for HMI is as shown in figure 61 below.

Figure 61: HMI Interface with PLC



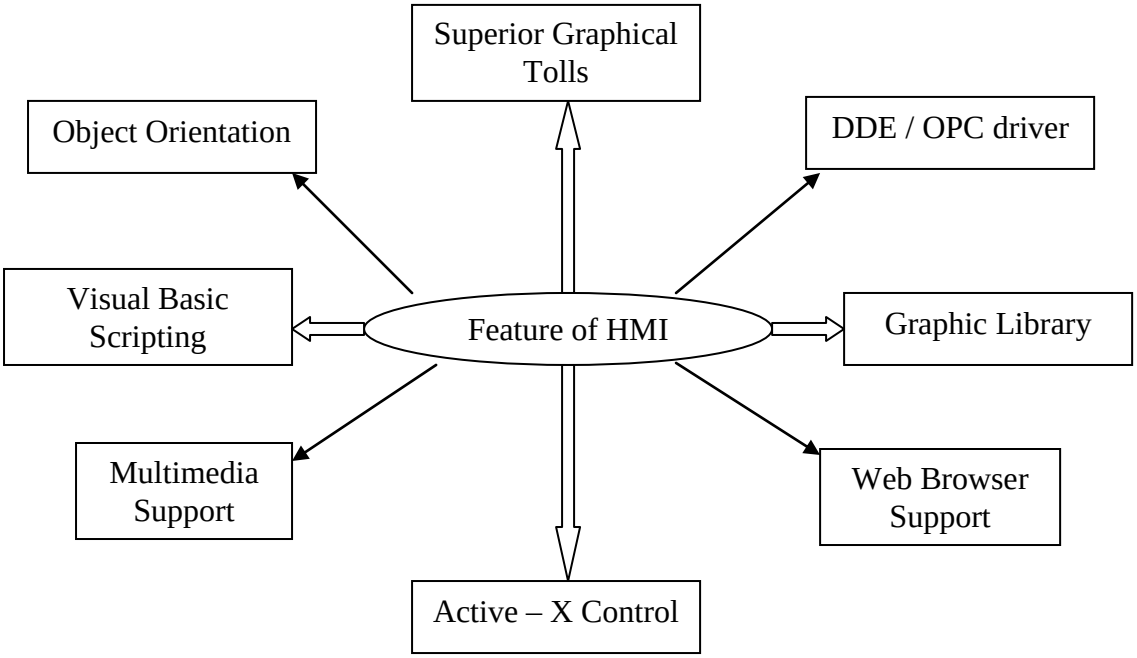
6.1 User Interface - HMI

PLCs may need to interact with people for the purpose of configuration, alarm reporting or everyday control. A *Human-Machine Interface* (HMI) is employed for this purpose. HMI's are also referred to as MMI's (Man Machine Interface) and GUI (Graphical User Interface).

A simple system may use buttons and lights to interact with the user. Text displays are available as well as graphical touch screens. Most modern PLCs can communicate over a network to some other system,

such as a computer running a *SCADA* (Supervisory Control And Data Acquisition) system, which is one of the system of automation refers to the combination of telemetry and data acquistion as shown in ‘figure 62’.

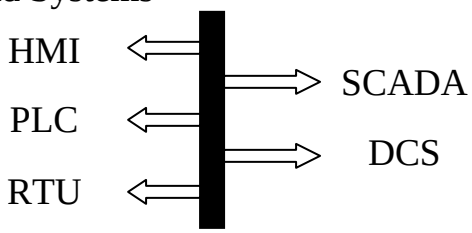
Figure 62: HMI interface



6.1.1 **Definition**

HMI is an interactive graphical system that enables the user to generate various control operations and control loops of the plant. It also facilitates continuous observation of the plant parameters as well as facility to give necessary commands required for plant operation [14]. The inter-relationship of different available ‘systems’ with ‘softwares’ is as shown in figure 63.

Figure 63: Softwares and Systems



6.1.2 **Purpose**

Automation with PLC provides facilities to sense the input conditions from the plant sensors / transducers and generate controlling output for plant operation. For effective process monitorong and control

efficient communication is necessary between process operator and the process to be automated and hence diffenent data terminals are incorporated into the system. The HMI is a user friendly link between plant operator and the automation system which finally controls the plant or process. The purpose of HMI is to help plant personnel monitor and control their industrial automation systems more effectively. The objectives are …

- Viewing graphical representations of plant devices and processes.
- Monitoring and controlling real time data.
- Receiving notification of alarm conditions.
- Trends of real-time / historical process vaules.
- Limiting access to critical proceses by implementing security measures.

6.1.3 **HMI with SCADA & DCS**

As SCADA is a recognized as a system, which is typically a combination of telemetry and data acquisition. It consist of collecting information, transferring it back to a central site (using CMS – Central Monitoring System), carrying out necessary analysis and control, and then displaying this data on one / number of operators screens as per requirement. It can help a small utility with just a single operator as well as a large utility with operators manning the process around the clock.

While the DCS system is one in which has several micro-computers can physically spread allover and each assigned with a special task and all manually linked through a data highway, might be co-axial / fiber optics. Where, each of these computers can perform its own task concurrently and independently of thr micro-computers in the system.

- Differences between SCADA and DCS as shown in Table 7

Table 7: SCADA and DCS

In Industrial Applicatons		
Based on	DCS	SCADA
Area	Confined	Large geographical
Communication Links	High speed network	Radio / Telemetry
Close loop system	Significant amount	Least priorities

6.2 **Communication Channels**

The Communication network refers to the communication equipment needed to transfer data to and from different sites. The medium can be either being cable, telephone or radio. Where,

- The use of cable doesn't suited more for large geographical areas because of huge cost of cable and less rudgeness.
- The use of telephone lines (i.e. leased or dial-up) is a cheaper solution for the system with large coverage. Dial-up lines can be used on systems requiring updates at regular intervals (hourly updates / within seconds), popular in case of PLC based applications.
- While, remote sites are usually not accessible by telephone lines. The use of radio offers an economical solution (particularly applications, where telephony media doesn't suits at its best by introducing radio modems), very popular in case of RTUs.

The entire communication channel is sub-classified into…

6.2.1 **Network Configurations**

Normally in process and automation industries 'Ethernet' is used to interconnect PLC based controller with HMI. While same controller is connected to field instruments and sensors via 'ControlNet' due to its balanced communication. PLCs usually have built in communications ports usually *9-Pin RS-232,* and optionally for *RS-485* and *Ethernet/IP or TCP/IP* and *DH or DH+ or DH-485 / Modbus or DF1* is usually included as one of the *communications protocols*. Others' options include various fieldbuses such as *DeviceNet* / *ControlNet / Profibus*.

6.2.2 **Communication Device**

The device that connects communication channels to the computer, can be Internet, such as 1784-KTIKTX or external devices connected through serial port using RS-232.

6.2.3 **Communication Driver**

The communication driver is the software that permits the computer to communicate with the communication device. For

communication with Allen-Bradley programmable controller, normally we use ‘RSLinx’ for Windows NT 3.54 and above.

6.2.4 **Node**

Sometimes communication nodes are reconazied as ‘Control Devices’. The node is a programmable controller attached to a highway or network. Once the RSViewSE station is set, it must be periodically update its value table. This is done by scanning its nodes.

6.2.5 **DDE**

DDE (Dynamic Data Exchange) is used to connect RSVIewSE with non Allen-Bradley devices via DDE server such as ‘Rockwell software RSServer™.