Project Based Training on PLC

At

Center for Electronics Test Engineering (CETE)

Presented by SAIKAT ADAK

College: Netaji Subhash Engineering College

Training Date: 19th June to 14th July, 2006

Group Members:

SAIKAT ADAK

Arobindo Chanda

Kartik Samanta

Signed by -

➲ About CETE

To bridge the gap between the training provided by academic institutions and the real needs of the industries, CETE *(Center for Electronics Test Engineering)*; an Indo-German Institution was set up in 1994 to address these requirements in the field of electronics and quality assurance.

Organizationally, CETE, Kolkata right from its inception in August 1994 belongs to SETE *(Society for Electronics Test Engineering)*, an autonomous society of the STQC *(Standardization, Testing, Quality Certification)* Directorate of Ministry of Information Technology, Government of India. SETE was set up jointly by STQC Directorate and GTZ *(Deutsche Gesellschaft Fur Technische Zusammenarbeit - Technical Cooperation Federal Republic of Germany)*. GTZ is a privately managed organization implementing 2000 projects more than 130 countries on behalf of the German Government.

CETE, Kolkata, right from its inception in August 1994 has been regularly conducting short term course in specialized areas of electronics, quality assurance and automation for professionals from industry and service organization as well as trainers from training and educational institutions. A large number of reputed industries have availed of the courses offered by CETE, Kolkata.

The courses presently being offered by CETE are in the broad areas of:

- Electronic Manufacturing Technology.
- Test and Calibration.
- Repair and Maintenance of Electronic Equipment.
- Industrial Automation.
- Other areas like Quality, Energy and Environmental Management, Laboratory Management, NDT.

CETE, Kolkata continuously updates its programs depending on changing industrial demand.

➲ Acknowledgements

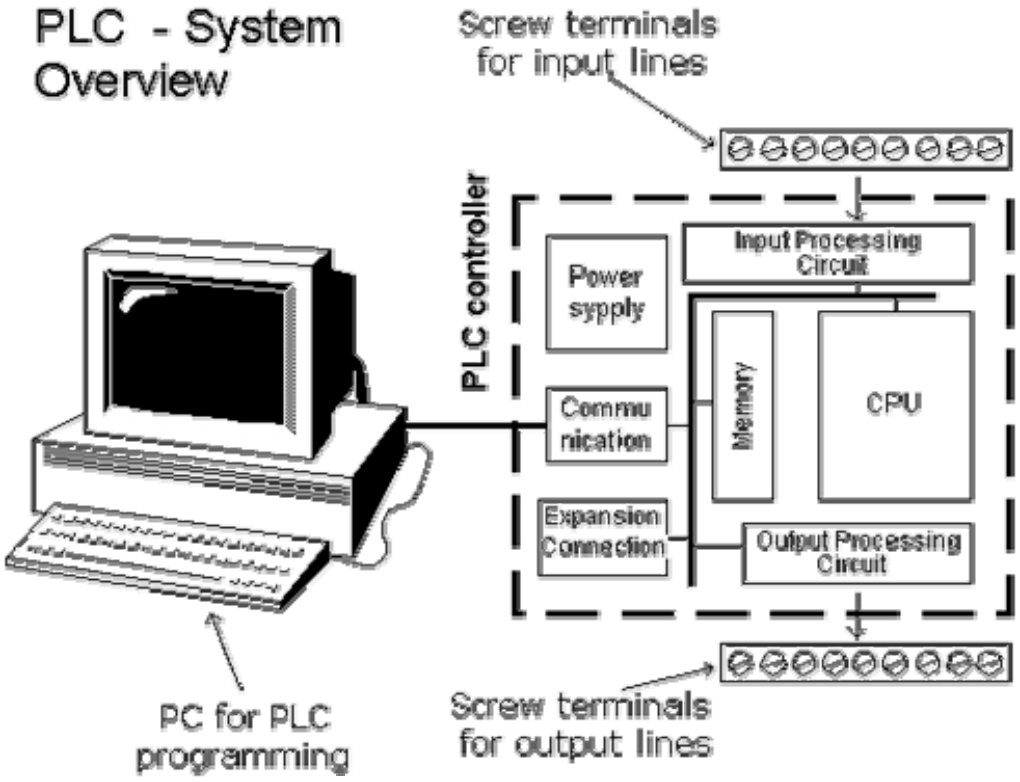
I acknowledge my indebtedness and convey my sincere thanks to my College Authority to send us to this esteemed institution (CETE) for our Summer Vacation Training (Final Year). I also convey my thanks to the Faculties of CETE who helped us wholeheartedly during the entire session. I would specially remember and convey my gratitude to Mr. Abhijit Dasgupta, Faculty of CETE, who sincerely helped us throughout the session to guide us. I would also convey my thanks to Mr. S. Mukherjee, who is our Project Guide.

➲ Content

➲ PLC – Introduction

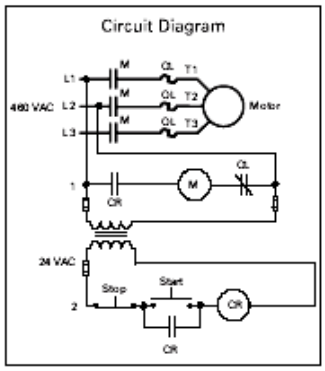
A Programmable Logic Controller (PLC) is an industrial computer control system that continuously monitors the state of input devices and makes decisions based upon a custom program, to control the state of devices connected as outputs. Almost any production line, machine function or process can be automated using a PLC. The speed and accuracy of the operation can be greatly enhanced using this type of control system. But the biggest benefit in using a PLC is the ability to change and replicate the operation or process while collecting and communicating vital information.

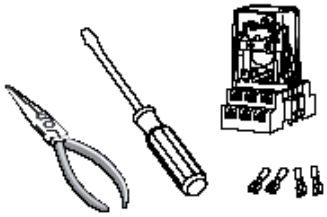
Typical PLCs



➲ Hard-Wired Control

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





➲ Advantage of PLC

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

Following are just a few of the advantages of PLCs:

- ☑ Smaller physical size than hard-wire solutions.
- ☑ Easier and faster to make changes.
- ☑ PLCs have integrated diagnostics and override functions.
- ☑ Diagnostics are centrally available.
- ☑ Applications can be immediately documented.
- ☑ Applications can be duplicated faster and less expensively.

➲ PLC Operations

1. INPUT SCAN: Scans the state of the Inputs (Sensing Devices, Switches and Pushbuttons, Proximity Sensors, Pressure Switches etc.).
2. PROGRAM SCAN: Executes the program logic.
3. OUTPUT SCAN: Energize/de-energize the outputs (Valves, Solenoids, Motor, Actuators, Pumps).
4. HOUSEKEEPING: Communication checking with the software and perform other requests according to their preference.

➲ PLC Terminology

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

Ladder Logic:

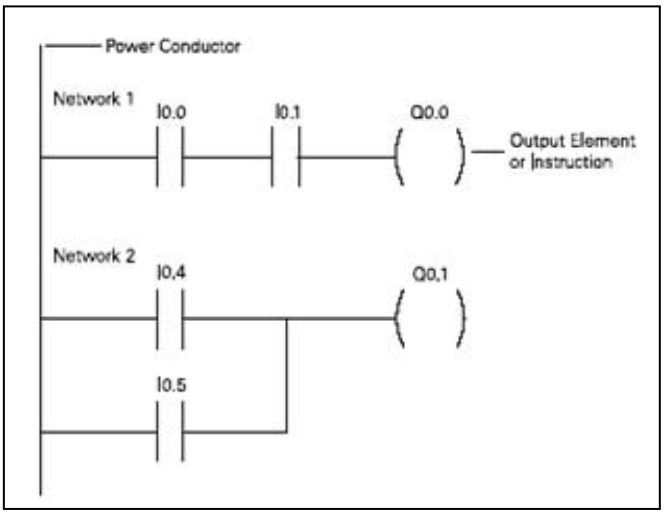
Ladder logic (LAD) is one programming language used with PLCs. Ladder logic uses components that resemble elements used in a line diagram format to describe hard-wired control.

The LAD editor displays the program as a graphical representation similar to electrical wiring diagrams. Ladder programs allow the program to emulate the flow of electric current from a power source through a series of logical input conditions that in turn enable logical output conditions. A LAD program includes a left power rail that is energized. Contacts that are closed allow energy to flow through them to the next element, and contacts that are open block that energy flow.

Ladder Logic Diagram (LAD):

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

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



The important features of LAD Editor:

- Ladder logic is easy for beginning programmers to use.
- Graphical representation is easy to understand and is popular around the world.
- The LAD editor can be used with both the SIMATIC and IEC 1131–3 instruction sets.
- You can always use the STL editor to display a program created with the SIMATIC LAD editor.

Statement list (STL):

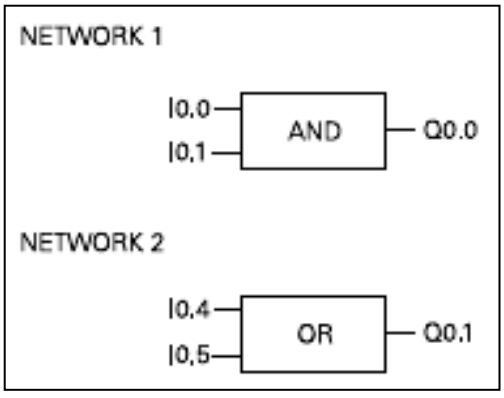
A statement list (STL) provides another view of a set of instructions. The operation, what is to be done, is shown on the left. The operand, the item to be operated on by the operation, is shown on the right. A comparison between the statement list shown below, and the ladder logic shown on the previous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

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

The important features of STL Editor:

- STL is most appropriate for experienced programmers.
- STL sometimes allows you to solve problems that you cannot solve very easily with the LAD or FBD editor.
- You can only use the STL editor with the SIMATIC instruction set.
- While you can always use the STL editor to view or edit a program that was created with the LAD or FBD editors, the reverse is not always true. You cannot always use the LAD or FBD editors to display a program that was written with the STL editor.

Function Block Diagrams (FBD):



Function Block Diagrams (FBD) provides another view of a set of instructions. Each function has a name to designate its specific task. Functions are indicated by a rectangle. Inputs are shown on the left-hand side of the rectangle and outputs are shown on the right-

hand side. The function block diagram shown here performs the same function as shown by the ladder diagram and statement list.

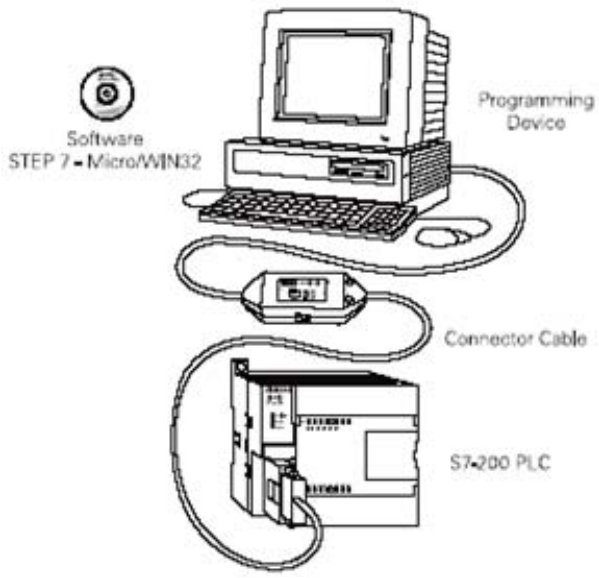
The important features of FBD Editor:

- The graphical logic gate style of representation is good for following program flow.
- The FBD editor can be used with both the SIMATIC and IEC 1131–3 instruction sets.
- You can always use the STL editor to display a program created with the SIMATIC FBD editor.

➲ Basic Requirements

In PLC programming in order to create or change a program, the following items are needed:

- PLC
- Programming Device
- Programming Software
- Connector Cable



Throughout our training we used the S7-200 (Siemens) because of its ease of use.

For the setup of Siemens the above items are:

- PLC: (S7-200)
- Programming Device :(Personal Computer)
- Programming Software: (Step 7 - MicroWIN 32)
- Connector Cable: (PC/PPI Cable) [PPI: Point to Point Interface]

[**NOTE**: Connector cables are required to transfer data from the programming device to the PLC. Communication can only take place when the two devices speak the same language or protocol. Communication between a Siemens programming device and the S7-200 is referred

to as PPI protocol (point to point interface). An appropriate cable is required for a programming device such as a PG 720 or PG 740. The S7-200 uses a 9-pin, D-connector. This is a straight-through serial device that is compatible with Siemens programming devices (MPI port) and is a standard connector for other serial interfaces.

A special cable is needed when a personal computer is used as a programming device. Two versions of this cable are available. One version, called an RS-232/PPI Multi-Master Cable, connects a personal computer's RS-232 interface to the PLC's RS-485 connector. The other version, called a USB/PPI Multi-Master Cable, connects a personal computer's USB interface to the PLC's RS-485 connector.]

➲ S7-200 Micro PLCs

The S7-200 Micro PLC is the smallest member of the SIMATIC S7 family of programmable controllers. The central processing unit (CPU) is internal to the PLC. Inputs and outputs (I/O) are the system control points. Inputs monitor field devices, such as switches and sensors. Outputs control other devices, such as motors and pumps. The programming port is the connection to the programming device.

[**PLC used throughout the training**: S7-200 CPU 214, CPU 224 AC/DC/RLY (EM 235) and S7-300]

S7 – 200 Models:

There are five S7-200 CPU types: CPU 221, CPU 222, CPU 224, CPU 224XP, and CPU 226 and two power supply configurations for each type.

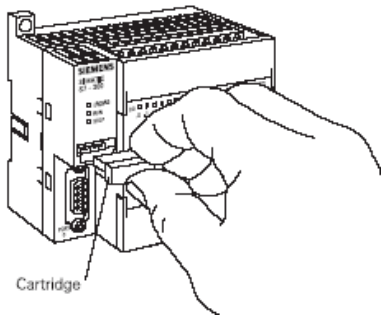
Model Description	Power Supply	Input Types	Output Types
221 DC/DC/DC	20.4-28.8 VDC	6 DC	4 DC
221 AC/DC/Relay	85-264 VAC, 47-63 Hz	6 DC	4 Relay
222 DC/DC/DC	20.4-28.8 VDC	8 DC	6 DC
222 AC/DC/Relay	85-264 VAC, 47-63 Hz	8 DC	6 Relay
224 DC/DC/DC	20.4-28.8 VDC	14 DC	10 DC

224 AC/DC/Relay	85-264 VAC, 47-63 Hz	14 DC	10 Relay
224XP DC/DC/DC	20.4-28.8 VDC	14 DC, 2 Analog	10 DC, 1 Analog
224XP AC/DC/Relay	85-264 VAC, 47-63 Hz	14 DC, 2 Analog	10 Relay, 1 Analog
226 DC/DC/DC	20.4-28.8 VDC	24 DC	16 DC
226 AC/DC/Relay	85-264 VAC, 47-63 Hz	24 DC	16 Relay

The model description indicates the type of CPU, the power supply, the type of input, and the type of output.

➲ Optional Cartridge

The S7-200 supports an optional memory cartridge that provides a portable EEPROM storage for your program. The cartridge can be used to copy a program from one S7-200 PLC to a like S7-200 PLC.



In addition, two other cartridges are available. A real-time clock with battery is available for use on the CPU 221 and CPU 222. The battery provides up to 200 days of data retention time in the event of a power loss. The CPU 224, CPU 224XP and CPU 226 has a real-time clock built in. Another cartridge is available with a battery only.

➲ I/O Numbering

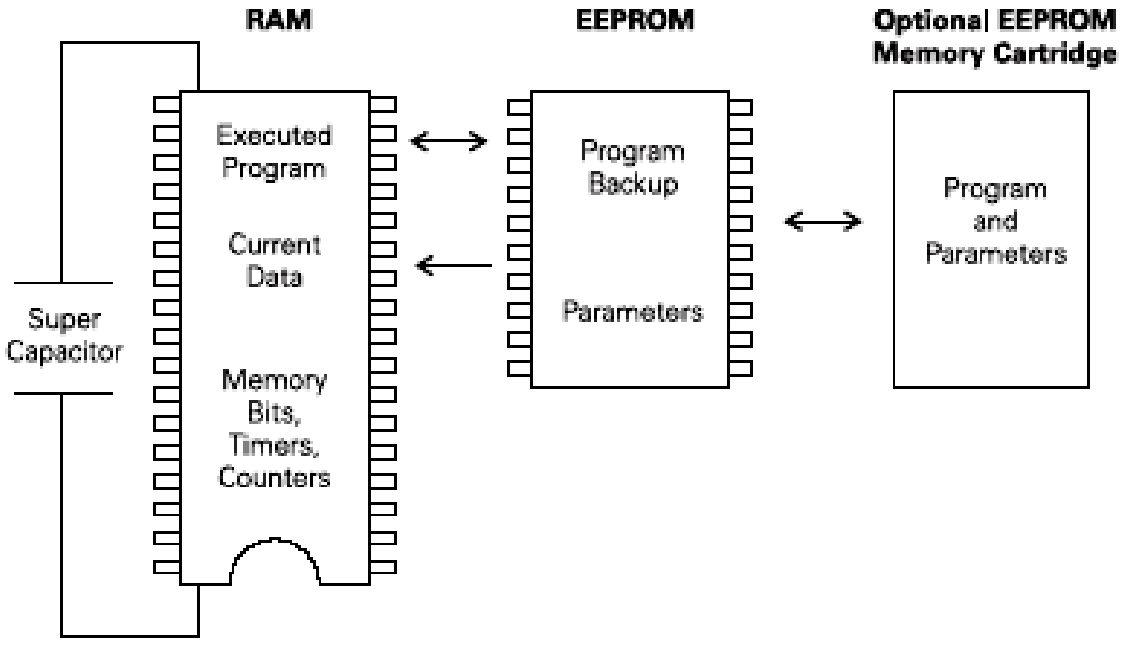
I designate a discrete input and Q designates a discrete output. The first

Process Image Input (PII) [Memory Matrix]								
	Bit 7	*Bit 6*	*Bit 5*	*Bit 4*	*Bit 3*	*Bit 2*	*Bit 1*	*Bit 0*
Byte 1	I0.7	I0.6	I0.5	I0.4	I0.3	I0.2	I0.1	I0.0
Byte 2	I1.7	I1.6	I1.5	I1.4	I1.3	I1.2	I1.1	I1.0
Byte 3	I2.7	I2.6	-	-	-	-	-	I2.0
-	-							
-	I4.7							

number identifies the byte; the second number identifies the bit. Input I0.0, for example, is byte 0, bit 0. The following diagram depicts the concept of memory matrix of PII. Similarly PIQ (Process Image Output) can be drawn in same fashion. Number of bytes of both PII and PIQ depends on the type of CPU used.

➲ Super Capacitor

A super capacitor, so named because of its ability to maintain a charge for a



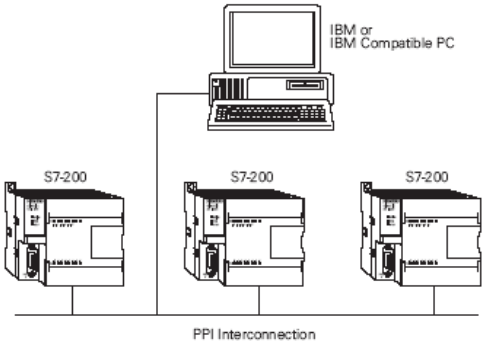
long period of time, protects data stored in RAM in the event of a power loss.

The RAM memory is typically backed up on the CPU 221 and CPU 222 for 50 hours and on the CPU 224, CPU 224 XP, and CPU 226 for 100 hours.

➲ Free Port Mode

The programming port has a mode called Freeport Mode. Freeport mode allows connectivity to various intelligent sensing devices such as a bar code reader.

➲ Interconnection

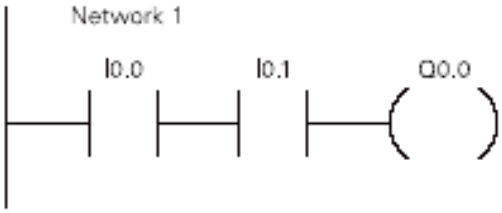


It is possible to use one programming device to address multiple S7-200 devices on the same communication cable. A total of 31 units can be interconnected without a repeater.

➲ An AND Operation

Each rung or network on a ladder represents a logic operation. The following programming example demonstrates an AND operation. Two contact closures and one output coil are placed on network 1. They were assigned addresses I0.0, I0.1, and
Q0.0. Note that in the statement list a new logic operation always begins with a load instruction (LD). In this example I0.0 (input 1)

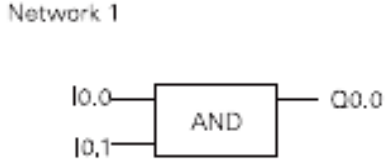
Ladder Diagram Representation



Statement List Representation

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

Function Block Diagram Representation



and (A in the statement list) I0.1 (input 2) must be true in order for output Q0.0 (output 1) to be true. It can also be seen that I0.0 and I0.1 must be true for Q0.0 to be true by looking at the function block diagram representation.

➲ An OR Operation

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

➲ Timer

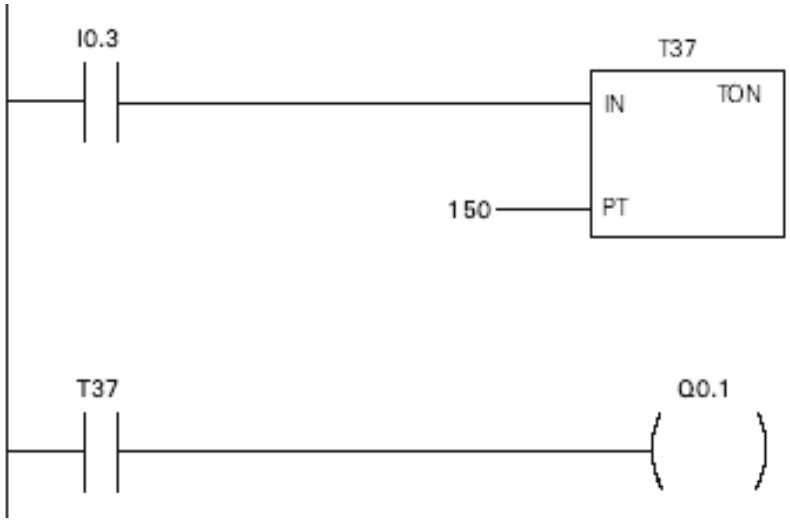
Timers are devices that count increments of time. Timers are represented by boxes in ladder logic. When a timer receives an enable, the timer starts to time. The timer compares its current time with the preset time. The output of the timer is logic 0 as long as the current time is less than the preset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

➲ S7-200 Timers

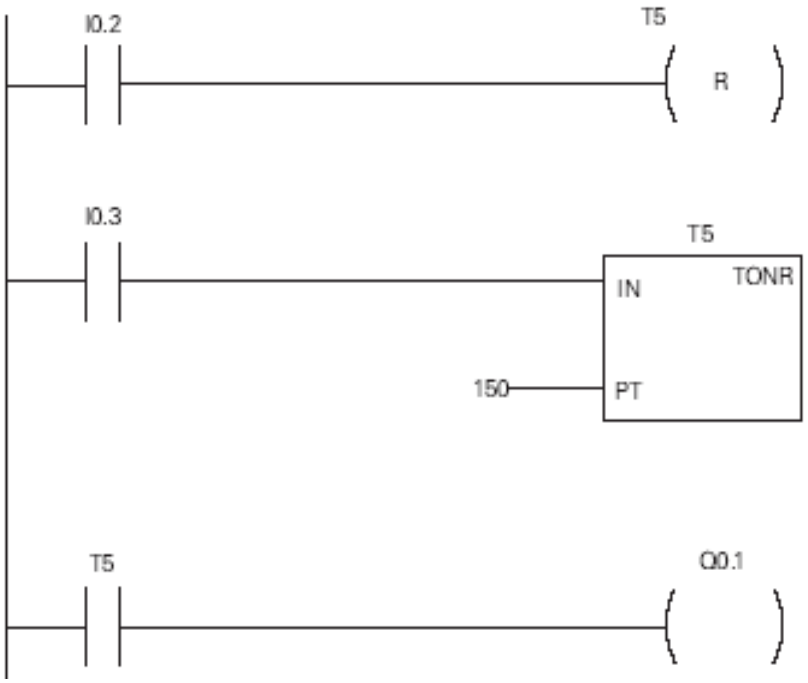
S7-200 timers are provided with resolutions of 1 millisecond, 10 milliseconds, and 100 milliseconds. The maximum value of these timers is 32.767 seconds, 327.67 seconds, and 3276.7 seconds, respectively. By adding program elements, logic can be programmed for much greater time intervals.

On-Delay Timer (TON):

When the On-Delay timer (TON) receives an enable (logic 1) at its input (IN), a predetermined amount of time (preset time - PT) passes before the timer bit (T-bit) turns on. The T-bit is a logic function internal to the timer and is not shown on the symbol. The timer resets to the starting time when the enabling input goes to logic 0.



In the following simple timer example, a switch is connected to input I0.3, and a light is connected to output Q0.1. When the switch is closed input 4 becomes a logic 1, which is loaded into timer T37. T37 has a time base of 100 ms (.100 seconds). The preset time (PT) value has been set to 150. This is equivalent to 15 seconds (.100 x 150). The light will turn on 15 seconds after the input switch is closed. If the switch were opened before 15 seconds had passed, then re-closed, the timer would again begin timing at 0.



Retentive On-Delay (TONR):

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

Off-Delay (TOF):

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

➲ Counter

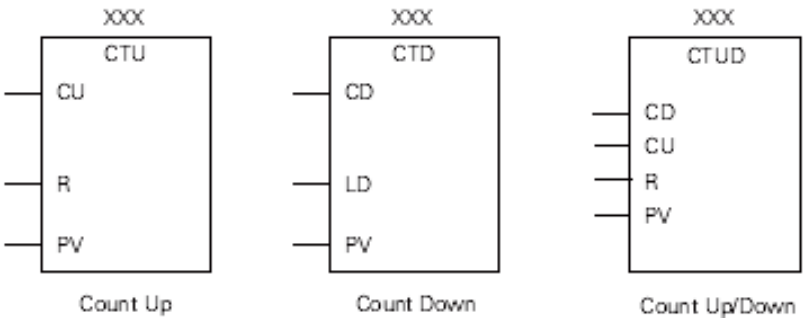
Counters used in PLCs serve the same function as mechanical counters. Counters compare an accumulated value to a preset value to control circuit functions. Control applications that commonly use counters include the following:

• Count to a preset value and cause an event to occur

• Cause an event to occur until the count reaches a preset value

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

Counters are represented by boxes in ladder logic. Counters increment/decrement



one count each time the input transitions from off (logic 0) to on (logic 1). The counters are reset when a RESET instruction is executed. S7-200 uses three types of counters: up counter (CTU), down counter (CTD), and up/down counter (CTUD). There are 256 counters in the S7-200, numbered C0 through C255. The same number cannot be assigned to more than one counter. For example, if an up counter is assigned number 45, a down counter cannot also be assigned number 45. The maximum count value of a counter is ±32,767.

Up Counter:

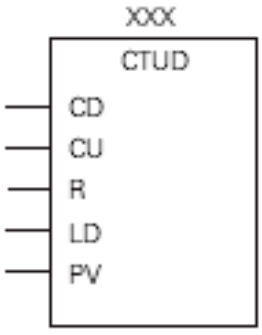
The up counter counts up from a current value to a preset value (PV). Input CU is the count input. Each time CU transitions from a logic 0 to a logic 1 the counter increments by a count of 1. Input R is the reset. A preset count value is stored in PV input. If the current count is equal to or greater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

Down Counter:

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

Up-Down Counter:

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



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

➲ Latching Contact / Hold On Button *(Task Performed)*

A Hold On Button is a type of push button which once pressed will latch its last state and then another switch will be needed to disconnect the corresponding circuit.

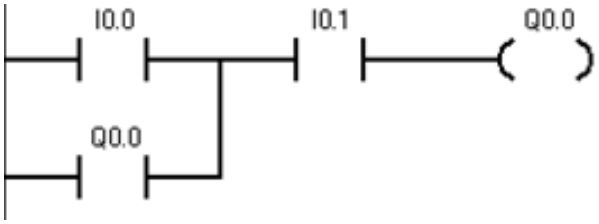
PROBLEM:

Make the logic flow diagram of Hold-On Button or Latching Contact in ladder representation using only NO contacts (XIC) and a simple coil. (Given: A NO push button for ON switch and an NC push button for OFF switch)

SOLUTION:

The ladder representation follows…

Here, I0.0 => ON switch (NOPB)
I0.1 => OFF switch (NCPB)
Q0.0 => Output coil



EXPLANATION:

Once the ON switch (I0.0) is pressed, the logic (or, power) will flow from left to right and the bulb will glow. Now that o/p is taken as feedback and connected in parallel with the network. So that the light may glow until it is turn off separately by another switch. That switch is I0.1 i.e. OFF switch.

➲ Push-To-On, Push-To-Off Button *(Task Performed)*

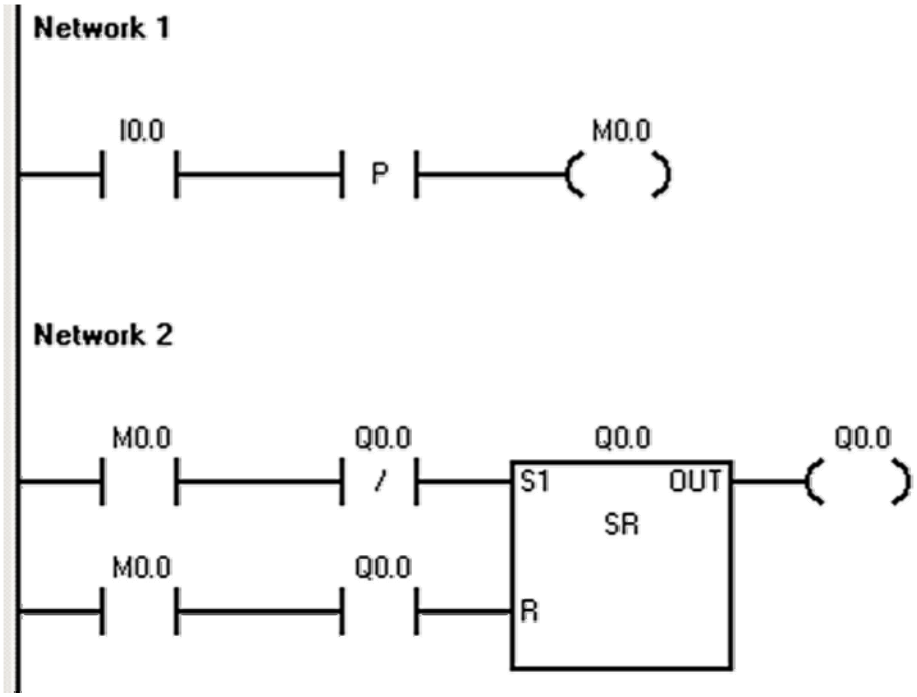
PROBLEM:

Design a Push Button which will be ON while pushing first time and OFF for second time.

SOLUTION:

The ladder representation follows…

Here, I0.0 => Push-To-On, Push-To-Off Button (NOPB)
M0.0 => Temporary memory
Q0.0 => Output to display the on / off state of the switch

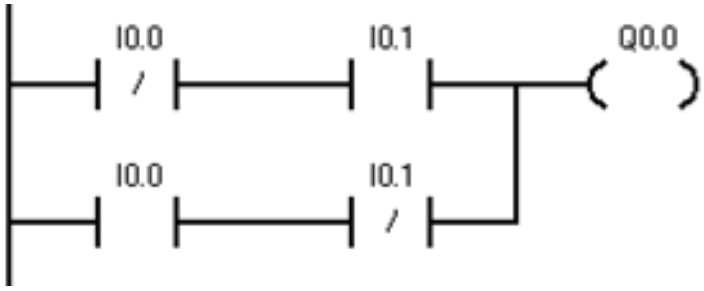


➲ Stair Case Lighting Arrangement *(Task Performed)*

PROBLEM:

There is a bulb which has two switches one on 1st and the other on 2nd floor. To make the bulb glow both the switches should be in opposite state. Make the ladder diagram using only NO (Normally Open) and NC (Normally Closed) contacts and a simple coil.

SOLUTION:

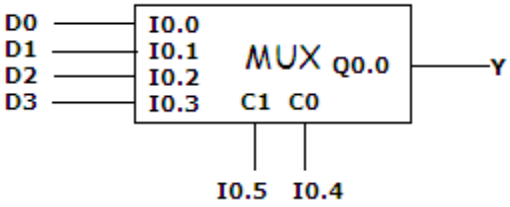


We assume that there are two switches I0.0 (switch n 1st floor) and I0.1 (switch n 2nd floor) and the bulb is assumed to have the address Q0.0. Now the ladder logic in the picture represents the Stair Case Switch circuit.

➲ Designing Multiplexer *(Task Performed)*

PROBLEM:

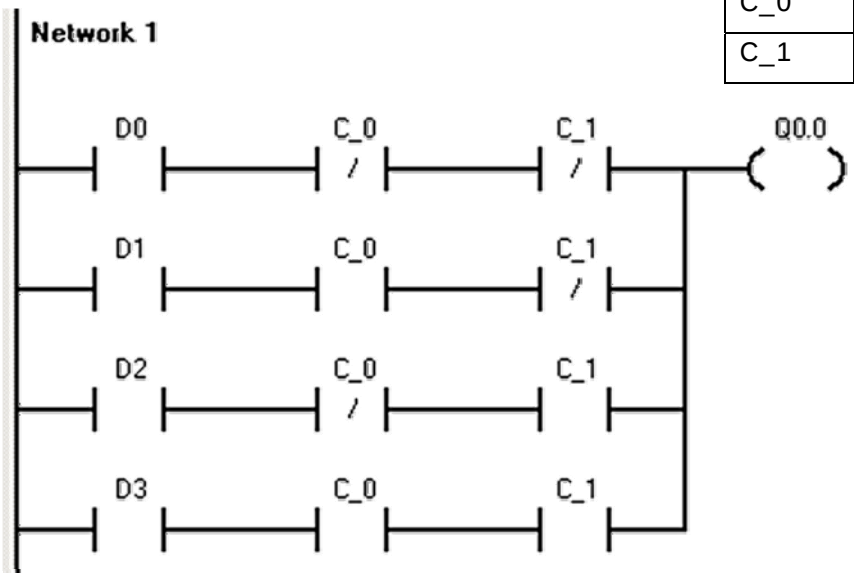
Design a MULTIPLEXER according to the figure beside …



Symbol	Address	Comment
D0	I0.0	1st data I/P
D1	I0.1	2nd data I/P
D2	I0.2	3rd data I/P
D3	I0.3	4th data I/P
C_0	I0.4	1st control line
C_1	I0.5	2nd control line

SOLUTION:

The ladder representation and symbol table follows …



➲ Interlocking Arrangement *(Task Performed)*

PROBLEM:

There is a shutter door which has an UP, DOWN (normally open) and OFF (normally closed) pushbutton. Now design an interlocking system between UP and DOWN button so that the shutter can't be opened when it is being closed. *[The OFF button is an emergency switch which will stop the shutter irrespective of its position or movement.]*

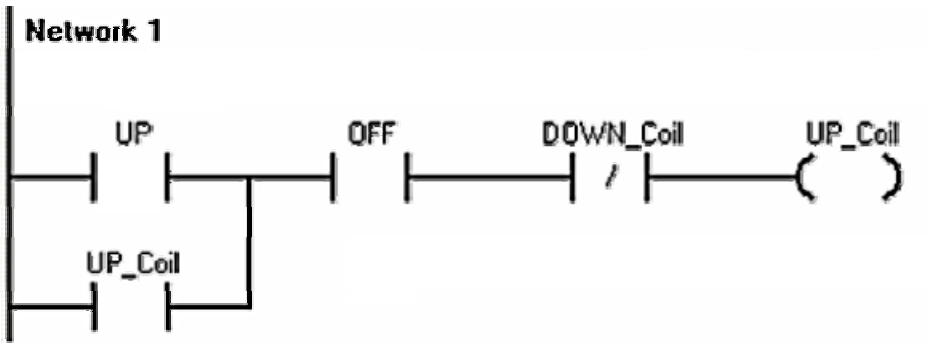
SOLUTION:

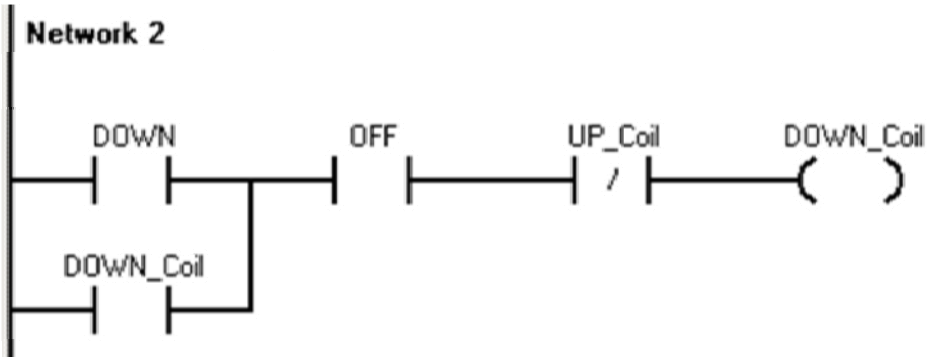
We will use two contactor coils (UP and DOWN) for UP and DOWN movement.

Used Symbol Table:

Symbol	Address	Comment
UP	I0.0	UP Button (Normally open push button)
DOWN	I0.1	DOWN Button (Normally open push button)
OFF	I0.2	OFF Button (Normally closed push button)
UP_Coil	Q0.0	Contractor coil K1 for upward direction
DOWN_Coil	Q0.1	Contractor coil K2 for downward direction

The ladder diagram of the above problem follows –




Network 2
DOWN
OFF
UP_Coil
DOWN_Coil
DOWN_Coil

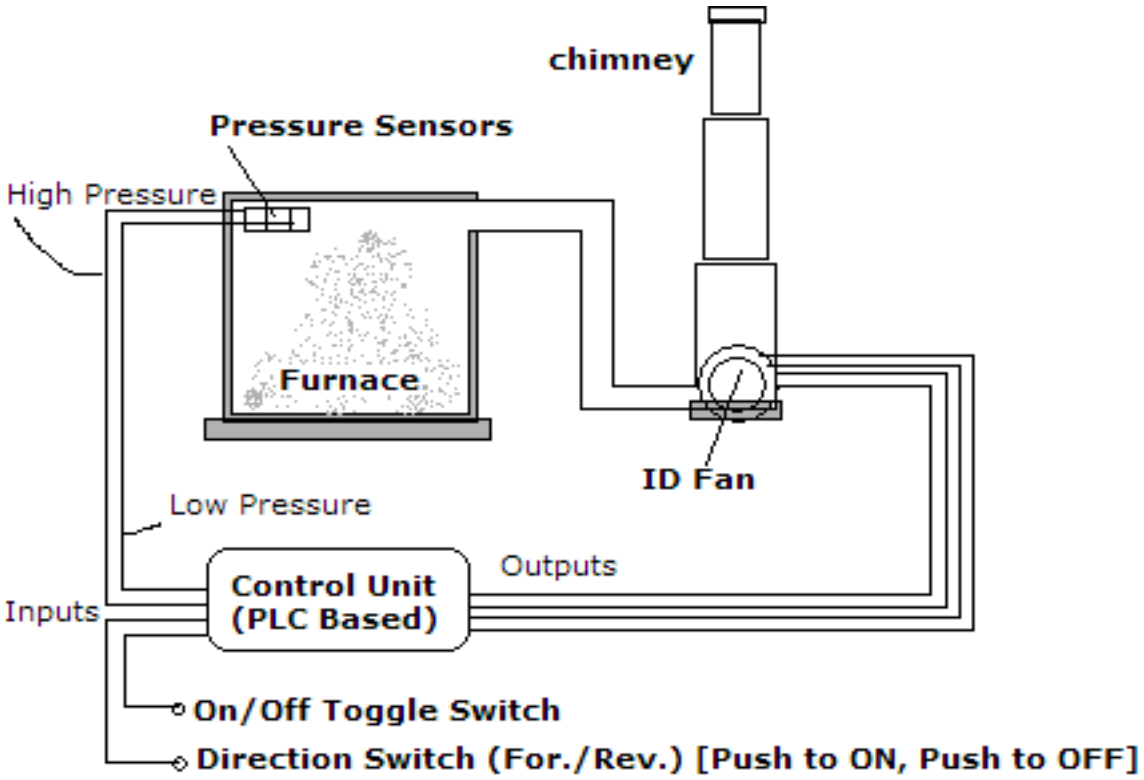
➲ Given Project on PLC

Speed Control Of Induced Draught Fan In A Furnace

PTO

⌘ OBJECT:

There is a furnace with which a chimney is attached to reduce the flue gas created by the reaction taking place inside the chimney. Also two sensors are there to sense high and low pressure. When pressure of flue gas inside the



furnace crosses a predefined value then one of these sensors gives a logic high signal to the controlling unit; similarly other gives a logic high signal when pressure falls below a certain value. Two switches are available to the operator; first one is On/Off switch of the furnace and second one is direction control switch for forward and reverse direction. This switch is push-to-on – push-to-off switch. Our main object is to design a PLC base automation system which will serve this purpose. There are some conditions also which are very important to keep in mind while making the system.

Conditions:

1. The reverse operation (known as Back Purging) of ID Fan is done for clearing, removal of waste and repairing purpose. So the ID Fan should not run in reverse direction when the furnace is on; otherwise it may cause a severe accident by increasing pressure of flue gas very quickly.
2. If by any fault two sensors (high and low) give signal at a time then there may be an unexpected error. So there should be an interlocking between two outputs corresponding to two input signals from pressure sensors.

⌘ SOLUTION:

The ladder logic diagram made in Siemens MicroWIN32 software follows in the next section with some other related headings:

Inputs		Outputs	
Address	**Comments**	**Address**	**Comments**
I0.0	Start/Stop switch (Toggle Switch) ON means Start and OFF means Stop.	Q0.0	High means Furnace will be Turn On, otherwise Turned Off.
I0.1	Forward/Reverse switch. (Push to ON - Push to OFF)	Q0.1	First time pushing means forward and second time pushing means reverse.
I0.2	Speed Up (NOPB)	Q0.2	High – Speed up, otherwise retain the last RPM
I0.3	Speed Down (NOPB)	Q0.3	High – Speed down, , otherwise retain the last RPM

⌘ LADDER LOGIC DIAGRAM:

⌘ EQUIPMENT USED:

i) PLC Simulator: (Simatic S7-200, Siemens)

CPU 214 EM 235 [AI3X12 Bit, AQ1X12 Bit]	AND	CPU 235 (AC/DC/RLY) EM 235 [AI4 /AQ1X12 Bit]

ii) Software: MicroWIN32 (STEP 7), made by Siemens

iii) AC Drive: Micro Master and Midi Master [It is used as Induced Draught Fan in the furnace]

⌘ EXTRA FEATURES ADDED:

With the given task regarding the project we added some extra features which we think are the important operations regarding speed control of ID fan in any furnace. These are:

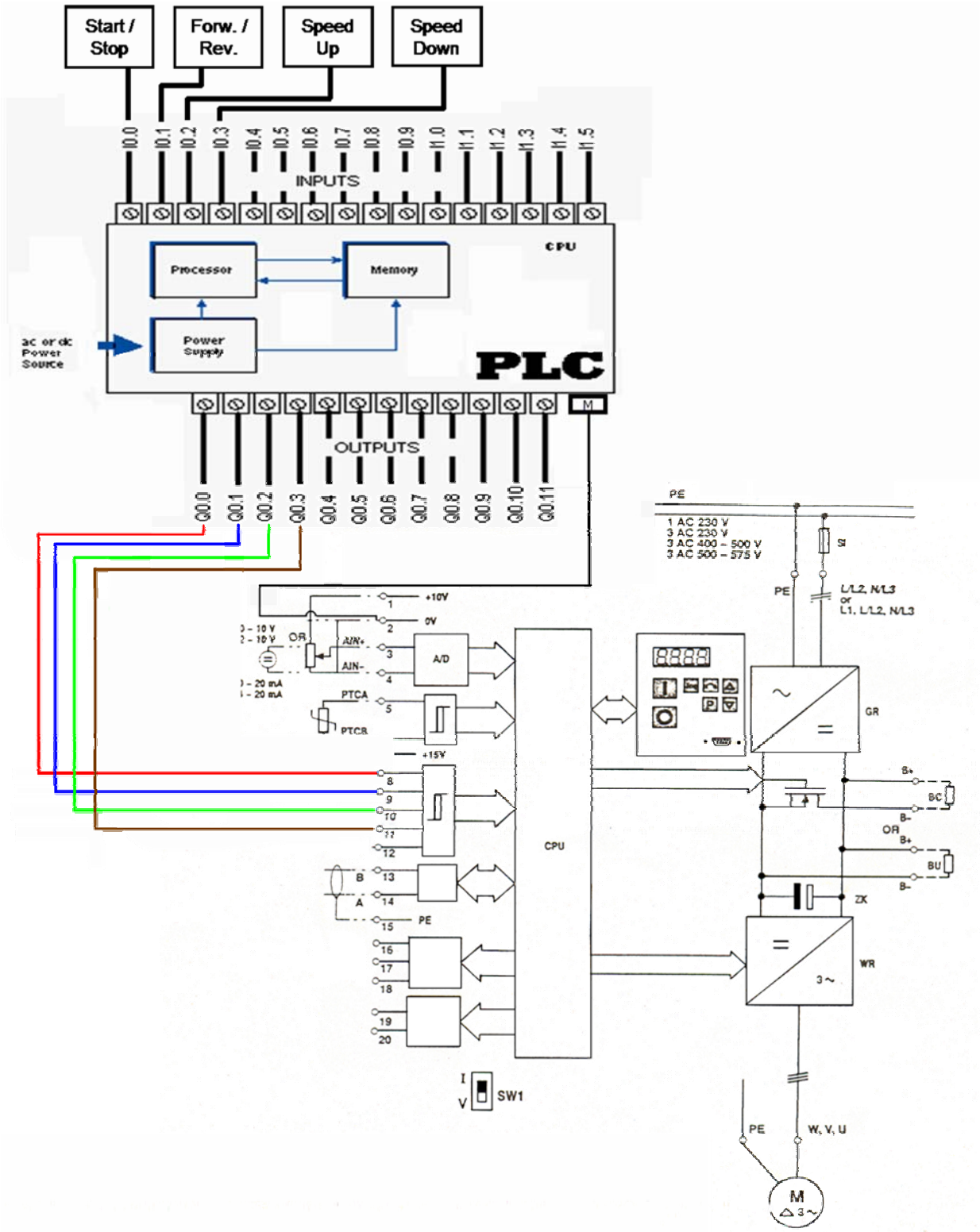
- Operating time of the furnace.
- Keeping emergency records of tripping time of the furnace.
- Volume of flue gas emitted by the ID fan.

To perform the last job (Volume of flue gas) we need to take feedback connection (RPM) from the AC Drive and need to connect it in the first Analog I/P Channel.

At any time we can see the volume of gas (emitted by the ID Fan), tripping time and operating time (of furnace) at the memory location as specified in the Symbol Table. A snapshot of the Symbol Table follows –

Sl.	Symbols	Address	Comments
22	TripTime_Mem	VB0	Place to store Tripping Time
24	Opertng_Min_Mem	VW10	Place to store Minutes of operating time.
25	Opertng_Hr_Mem	VW12	Place to store Hours of operating time.
26	Opertng_Day_Mem	VW14	Place to store Days of operating time.
27	Opertng_Sec_Mem	VW16	Place to store seconds of operating time.
31	Flue_Val_Mem	VW20	Value of FLUE emitted (Volume).

⌘ WIRING DIAGRAM:



MICRO MASTER/MIDI MASTER BLOCK DIAGRAM USING PLC

Abbreviation And Symbols Used:

AD	Analogue to Digital Converter
BC	Break Chopper (MICRO MASTER)
BU	Breaking Unit (MIDI MASTER)
CPU	Microprocessor
GR	Rectifier
M	Motor
RS485	Serial Interface
SI	Mains Fuses
WR	Inverter
ZK	DC Link Capacitor

⌘ SYMBOL TABLE:

	Symbols	Address	Comments
1	Strt_Stp_Switch	I0.0	Furnace Start / Stop Switch [Toggle Switch]
2	For_Rev_Switch	I0.1	Fan For/Rev Switch [Push to ON / Push to OFF Switch]
3	Press_High	I0.2	[Push Button]
4	Press_Low	I0.3	[Push Button]
5	RTC_Set	I0.4	[Push Button]
6			
7	Fur_Strt_Stp	Q0.0	Q0.0 is High --> Furnace Start , Q0.0 is Low--> Furnace Stop
8	For_Rev	Q0.1	Q0.1 is High --> Forward , Q0.1 is Low --> Reverse
9	Speed_Up	Q0.2	Q0.2 is High --> Speed Up , Q0.2 is Low --> No Change
10	Speed_Down	Q0.3	Q0.3 is High --> Speed Down , Q0.3 is Low --> No Change
11			
12	Temp_Mem	M0.0	Used for Push to ON / Push to OFF operation
13	Sec_Count_Temp	M0.1	Used to create pulse of 0.5 sec delay.
14			
15	Reset_Timer	T37	Used for Reset operation in second counting.
16	Set_Timer	T38	Used for Set operation in second counting.
17			
18	Minute_Counter	C0	Minute Counter

19	Hour_Counter	C1	Hour Counter
20	Day_Counter	C2	Day Counter
21			
22	TripTime_Mem	VB0	Place to store Tripping Time
23			
24	Opertng_Min_Mem	VW10	Place to store Minutes of operating time.
25	Opertng_Hr_Mem	VW12	Place to store Hours of operating time.
26	Opertng_Day_Mem	VW14	Place to store Days of operating time.
27	Opertng_Sec_Mem	VW16	Place to store seconds of operating time.
28	Mul_Factor	VW18	Multiplying Factor (F) : Assuming F unit reduction of flue gas is done by 1 rps
29	Temp_VW_Mem	VW22	Temporary Memory used for calculation of volume of flue emitted.
30			
31	Flue_Val_Mem	VW20	Value of FLUE emitted (Volume).
32			
33	RPM_Feedback	AIW0	RPS feedback can be taken from analog input channel AIW0.

⌘ DATA BLOCK:

```
//DATA PAGE COMMENTS

//Current Value of year, month etc. to Set the Real Time Clock
//All values are in BCD (Hexadecimal) format as per the requirement of SET_RTC block

VB30   16#06          //Year
VB31   16#07          //Month
VB32   16#02          //Day
VB33   16#05          //Hour
VB34   16#04          //Minute
VB35   16#02          //Second
VB36   0              //Always Set to ZERO
VB37   16#1           //Day of Week

Flue_Val_Mem 0        //For the first time VW20 is set to zero.

Mul_Factor 1                //Loading VW18 with 1, assuming that 1 unit of gas is
                reduced when the fan moves with 1rps speed.
```

While downloading the program to the PLC the data bock should also be downloaded.

⌘ USING SCADA:

SCADA is a software (also known as Human Machine Interface, HMI), which helps to visualize the state of the process input-outputs in a single monitor and communicate with PLC to track the memory content and others. It can also draw different types of graphs and animation depending on the application.
In our project we can use SCADA to check the memory where the Tripping Time, Operating Time and Volume of gas emitted is stored.

➲ Bibliography

The following resources help me to write the report to a great extent…

i) Internet (tutorials, different websites and free resources),
ii) Manual of Simatik STEP-7 PLC (Siemens),
iii) Study material from CETE.

Also the faculties helped us very much to write this report.