Project Report

Of

PLC Based Elevator Controller

Presented By

SAIKAT ADAK

E.E., B-Tech

U-Tech. Roll No. – 10916031040

Netaji Subhash Engineering College

Project Duration: 7th & 8th Semester

Under The Guidance of

Dr. U. Kar and Mr. T. Nag

PLC Based Elevator Controller

A Project Report

Submitted In Partial Fulfillment Of The

Requirement for the Degree Of

Bachelor of Technology

In

Electrical Engineering

Of

WBUT

By

	Name	WBUT Roll No.
1.	Saikat Adak	10916031040
2.	Arabindo Chandra	
3.	Aniruddha Mitra	
4.	Subhasis Dhar	
5.	Debnil Chakraborty	
6.	Dibyendu Mukherjee	

Under The Guidance Of

Mr.T.Nag

And

Dr. U. Kar

At

Netaji Subhash Engineering College

Technocity, Panchpota, Garia

Kolkata - 152

Certificate of Approval

Netaji Subhash Engineering College

This is to certify that the project entitled "**PLC Based Elevator Controller**" submitted in partial fulfillment of the requirement for the award of *Bachelor of Technology* in *Electrical Engineering* under *WBUT* is faithful record of the bona fide project work, carried out by the following candidates under my guidance and supervision.

Name	WBUT Roll No.
1. Saikat Adak	10916031040
2. Arabindo Chandra	
3. Aniruddha Mitra	
4. Subhasis Dhar	
5. Debnil Chakraborty	
6. Dibyendu Mukherjee	

(Dr. M. K. Ghosh)
Head of the Dept.
EE, NSEC

(Dr. U. Kar)
Professor,
EE, NSEC

(Mr. T. Nag)
Lecturer,
EE, NSEC

Acknowledgement

I acknowledge my indebtedness and convey my sincere thanks to our project guides Dr. U. Kar and Mr. T. Nag, faculty of Dept. of E.E., NSEC, who sincerely helped us by giving inspiration, new ideas and infrastructure throughout the semesters. I would also like to thank Electrical Department of Netaji Subhash Engineering College to give us such platform for making this project successful. I also convey my thanks to the faculties of my college and my group members who helped me wholeheartedly during the entire session.
At last I again convey my special thanks to the project guides who decided to show the model of this project at the workshop arranged at that time.

Abstract

Though practically elevators are not controlled by PLC, still we employed it, because elevator is an appropriate system where we can explore a lot of features of the PLC. As it is a mere model only, while shifting to practical elevator some module of our model need to be replaced, viz. DC motor drive need to be replaced by an Induction motor drive, a weight counter-balancing technique should be employed. But as our target of doing this project is mainly PLC oriented, we mainly focused in PLC ladder logic and how to connect an external hardware/system with the PLC to control that hardware.

Content

Chapter - 1: Basics of PLC

Chapter - 2: Project - PLC Based Elevator Controller

Chapter - 3

Basics of PLC

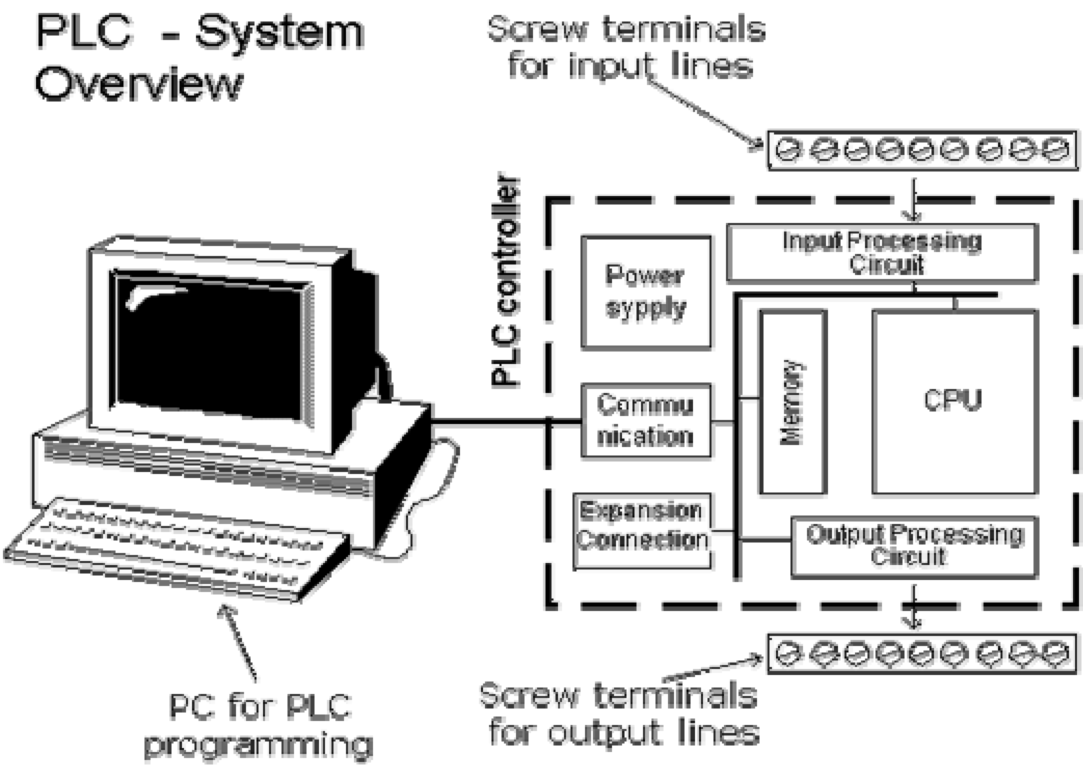
Chapter – 1

Introduction

PLC is actually an industrial microcontroller system (in more recent times we meet processors instead of microcontrollers) where we have hardware and software specifically adapted to industrial environment. Blocks came with typical components, which PLC consist of, is found in the following picture. Special attention needs to be given to input and output, because in these blocks you find protection needed in isolating a CPU blocks from damaging influences that industrial environment can bring to a CPU via input lines. Program unit is usually a computer used for writing a program (often in ladder diagram).

Typical PLCs

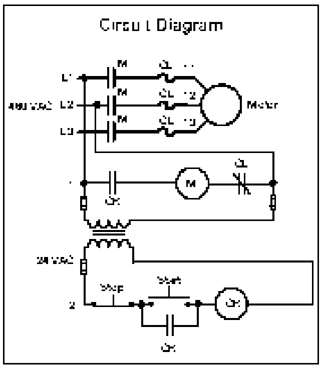
Central processing unit is the brain of a PLC controller. CPU itself is usually one of the microcontrollers. CPU also takes care of communication, interconnectedness among other parts of PLC controller, program execution, memory operation, overseeing input and setting up of an output. PLC controllers have complex routines for memory check up in order to ensure that PLC memory was not

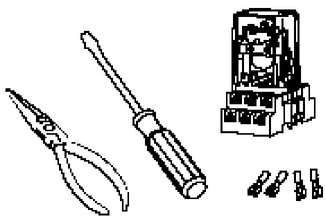


damaged (memory check up is done for safety reasons).

PLC controller can be reprogrammed through a computer, but also through manual programs (console). This practically means that each PLC Controller can be programmed through a computer if you have the software needed for programming. Today's transmission computers are ideal for reprogramming of PLC comptroller in factory itself. This is of great importance in industry. Once the system is corrected, it is also important to read the right program into a PLC again. It is also good to check from time to time whether program in a PLC has not changed. This helps to avoid hazardous situations in factory rooms.

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 to perform 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.





Architecture

The programmable logic controller is basically computer-based and therefore, their architecture is very similar to computer architecture. The memory contains operating system stored in fixed memory like ROM, rather than disk in case of computers. The application program is stored in Read-Write portion of memory.

All programmable controllers contain a Central processing Unit (CPU), Memory, Power Supply, Input/Output (I/O) modules and programming device.

The operating system is the main workhorse of the system. It is necessary to distinguish between the instructions used by operating system to command the microprocessor and the instruction used by the programmable controller to handle the specific control problem. The operating system performs the following tasks:

- Execution of application program.
- Memory management.
- Communication between programmable controller and other units.
- I/O interfaces handling.
- Diagnostics.
- Resource sharing.

The CPU, upon receiving instructions from the memory together with feedback on the status of the I/P-O/P devices, generates commands to the outputs by means of the o/p modules these commands control the o/p elements on a machine or process device such as relay coils, solenoid valves, indicator lamps and motor starters are typical loads to be controlled.

During program execution the processor reads all the inputs, takes these values and according to control application program, energizes or de-energizes the outputs, thus solving the ladder network.

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

I. INPUT SCAN: Scans the state of the Inputs (Sensing Devices, Switches and Pushbuttons, Proximity Sensors, Pressure Switches etc.).
II. PROGRAM SCAN: Executes the program logic.
III. OUTPUT SCAN: Energize/de-energize the outputs (Valves, Solenoids, Motor, Actuators, Pumps).
IV. 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:

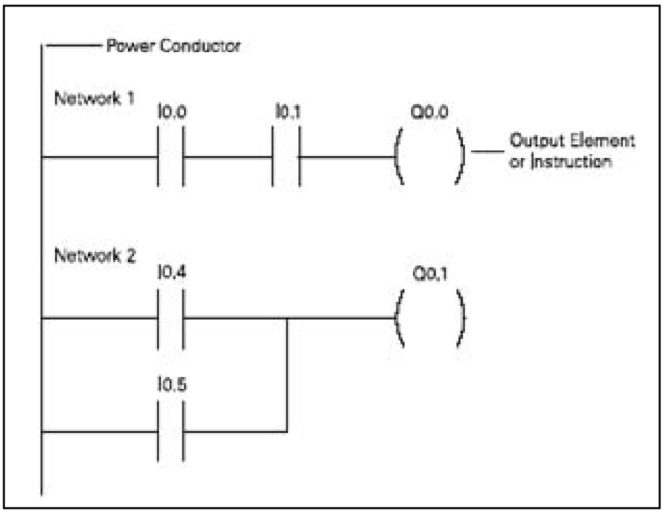
Ladder logic (LAD) is one programming language used with PLCs. Ladder logic uses components that resemble elements used in a line diagram format to describe hard-wired control.

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):

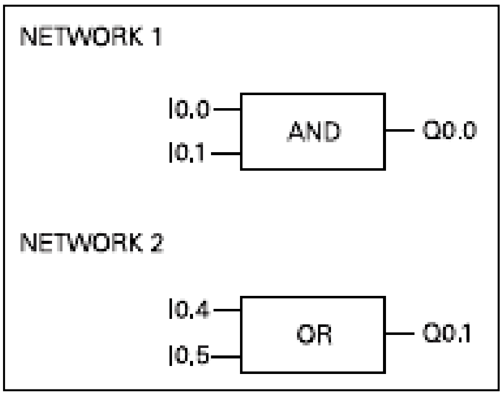
A statement list (STL) provides another view of a set of instructions. The operation, what is to be done, is shown on the left. The operand, the item to be operated on by the operation, is shown on the right. A comparison between the statement list shown below, and the ladder logic shown on the previous 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 we cannot solve very easily with the LAD or FBD editor.
- We can only use the STL editor with the SIMATIC instruction set.
- While we 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. We 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.
- We 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:

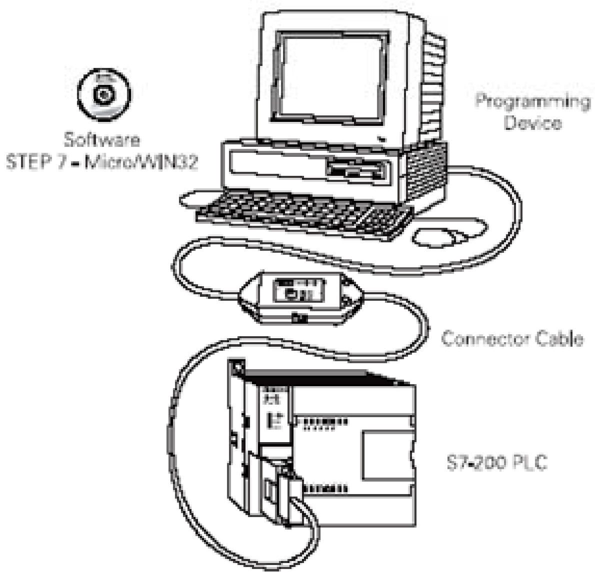
- PLC
- Programming Device
- Programming Software
- Connector Cable

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

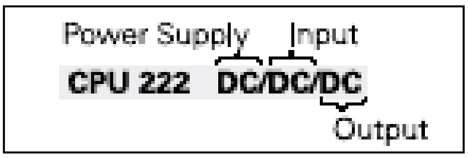
The S7-200 Micro PLC is the smallest member of the SIMATIC S7 family of programmable controllers. The central processing unit (CPU) is internal to the PLC. Inputs and outputs (I/O) are the system control points. Inputs monitor field devices, such as switches and sensors. Outputs control other devices,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

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.

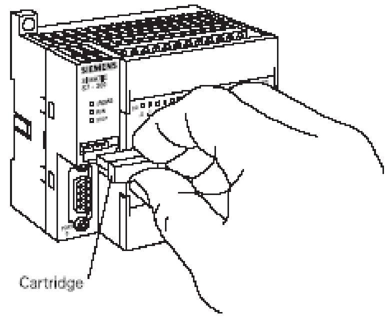
Model Description	Power Supply	Input Types	Output Types
221 DC/DC/DC	20.4-28.8 VDC	6 DC	4 DC
221 AC/DC/Relay	85-264 VAC, 47-63 Hz	6 DC	4 Relay
222 DC/DC/DC	20.4-28.8 VDC	8 DC	6 DC
222 AC/DC/Relay	85-264 VAC, 47-63 Hz	8 DC	6 Relay
224 DC/DC/DC	20.4-28.8 VDC	14 DC	10 DC
224 AC/DC/Relay	85-264 VAC, 47-63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

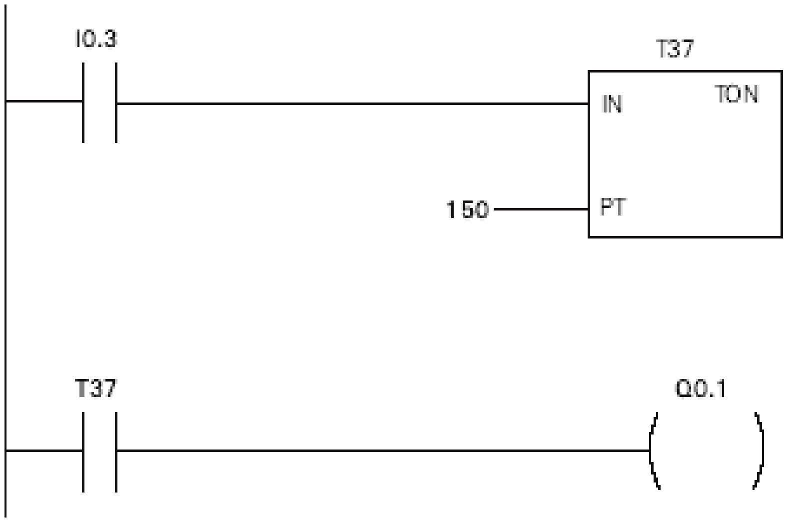
Process Image Input (PII) [Memory Matrix]								
	Bit 7	*Bit 6*	*Bit 5*	*Bit 4*	*Bit 3*	*Bit 2*	*Bit 1*	*Bit 0*
Byte 1	I0.7	I0.6	I0.5	I0.4	I0.3	I0.2	I0.1	I0.0
Byte 2	I1.7	I1.6	I1.5	I1.4	I1.3	I1.2	I1.1	I1.0
Byte 3	I2.7	I2.6	-	-	-	-	-	I2.0
-	-							
-	I4.7							

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).

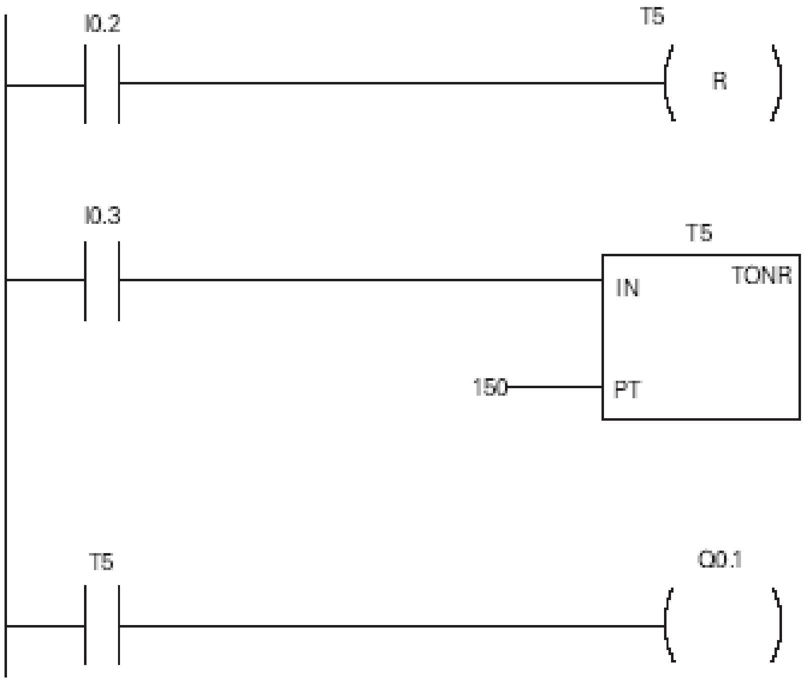
S7-200 Timers

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):

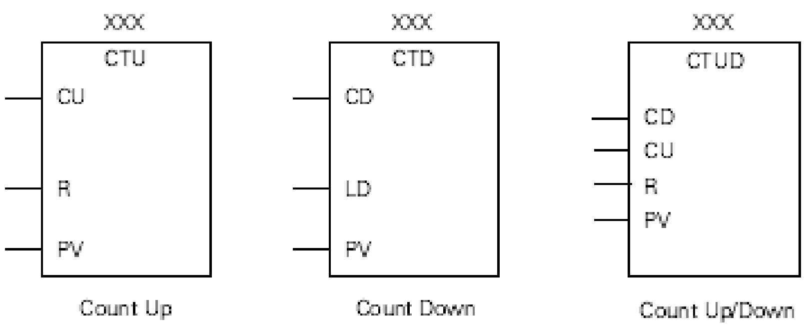
The Off-Delay timer is used to delay an output off for a fixed period of time after the input turns off. When the enabling bit turns on the timer bit turns on immediately and the value is set to 0. When the input turns off, the timer counts until the preset time has elapsed before the timer bit turns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

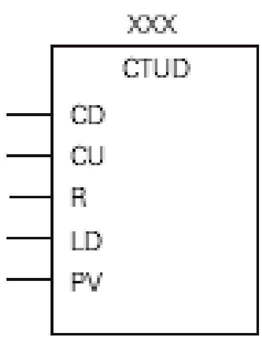
The up counter counts up from a current value to a preset value (PV). Input CU is the count input. Each time CU transitions from a logic 0 to a logic 1 the counter increments by a count of 1. Input R is the reset. A preset count value is stored in PV input. If the current count is equal to or greater 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.



Description of Model

Chapter – 2

PROJECT

A Model

Of

PLC Based Elevator Controller

(Applicable for any number of floor)

Objective

A controlling module/unit is to be installed to control elevators of any multi storied building/shopping mall etc. The problem concerns the logic required to move elevators between floors according to the following constraints:

- Each floor has a button to request upward or downward movement.
- The door of the elevator will be programmed to open and close automatically.
- When the elevator has no request, it remains at its current floor with its doors opened.
- When the elevator gets multiple requests from different floors it will serve them according to first-come-first-serve basis. Also the ladder logic should be that much of flexible so that the serving technique can be changed according to the requirement (like nearest-floor-first, floor-having more-people-first etc).
- The current floor number will be shown within the lift by a small display.

Description

Here are the description of few key terms and symbols which are used in this project. Also the description of the model will be covered shortly in the next few paragraphs.

Symbol / Component	**Description**
LiftPosMem	Flag to store the current position of the lift.
TempMem	Flag to store the next request pending collected from QUEUE.
Touch Sensors	Used to detect the lift position.
Push Buttons	Normally open push buttons are used to take request from different floors and from lift.
K1, K2	Contactor coil connected with the motor for upward & downward movement of the lift. [K1 – Up, K2 - Down]
QUEUE	Used to store all requests from lift and different floors. While storing the current request, some arrangement are done according to predefined scheme like FCFS, Nearest-Floor-First, Floor having more people first etc.
Timers	It is used to open and close the door of the lift after/for a certain time. Also it is used to stay for a certain time in a floor.

Flowchart

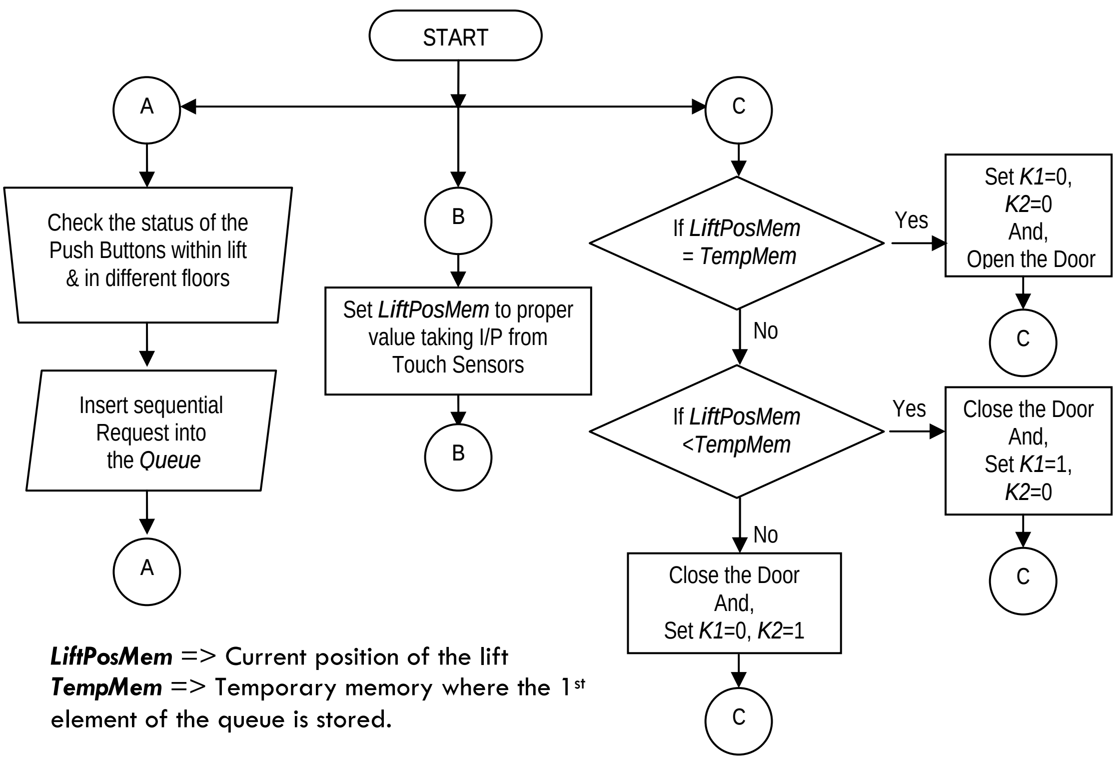
Here is the flowchart of the ladder logic which consists of three threads (A, B, and C) operating simultaneously.

<u>Thread A</u>: Continuously checks the status of push buttons placed in different floors and within the lift. It also inserts the requests into a queue which is defined in the memory of PLC.

<u>Thread B</u>: Always checking the current lift position using touch sensors placed in different floors.

<u>Thread C</u>: Takes care of upward or downward movement of the lift and also responsible for the door close & open operation.

These threads are placed in a loop to continuously perform their tasks. The parameters used in this flowchart are tabulated in the previous heading.



LiftPosMem => Current position of the lift
TempMem => Temporary memory where the 1st element of the queue is stored.

Ladder Description

The ladder diagram has been designed in Step-7 MicroWIN 32 software (made by Siemens). It has been designed for multi level or multi storied building. The ladder has been designed in such a way, so that it can be easily applied to a building having any number of floors. The ladder has mainly three parts (or, threads) which are running simultaneously. The first thread is running to check the status of push buttons placed in different floors and store them in a queue. The second one is used to track the current position of the lift and store them in a temporary memory. And the third one is used to serve the requests stored in the queue as they were stored (i.e. first-in-first-out). To perform this kind of operation an inbuilt data structure and some readymade blocks have been used.

Siemens supports some inbuilt data structure like FIFO, LIFO Table etc. Here FIFO Table is used to implement the First-Come-First-Serve technique. Also some other important blocks are used to perform the Queuing and Erase operation in Queue.

The Symbol Table, Ladder Diagram, Data Block etc. are listed below…

Symbol Table

All the inputs, outputs, memory blocks used in this program are tabulated below. Also the symbols used in reference with those memory addresses are mapped according to this table-

	Symbol	Address	Comment
1	MaxEntries_Qu	VW0	Maximum no. of entries in the queue / Starting address of Table
2	EC_Qu	VW2	Entry count of the queue
3			
4	LiftPosMEM	VW50	Memory to store current Lift Position
5	TempMem	VW52	Next floor waiting for service
6			
7	Req_Gnd_Flor	I0.0	Request coming from Ground floor
8	Req_1st_Flor	I0.1	Request coming from 1st floor
9	Req_2nd_Flor	I0.2	
10	Req_3rd_Flor	I0.3	
11	Req_4th_Flor	I0.4	
12			
13	TSensr_Gnd_Lift	I1.0	Touch sensor o/p - Lift is in ground floor
14	TSensr_1st_Lift	I1.1	Touch sensor o/p - Lift is in 1st floor
15	TSensr_2nd_Lift	I1.2	Touch sensor o/p - Lift is in 2nd floor
16	TSensr_3rd_Lift	I1.3	
17	TSensr_4th_Lift	I1.4	
18			
19	Indcatr_Gnd	Q0.0	Indicator of Ground floor request
20	Indcatr_1st	Q0.1	Indicator of 1st floor request
21	Indcatr_2nd	Q0.2	Indicator of 2nd floor request
22	Indcatr_3rd	Q0.3	Indicator of 3rd floor request
23	Indcatr_4th	Q0.4	Indicator of 4th floor request
24			
25	K1	Q1.0	Contactor 1 (Upword direction)
26	K2	Q1.1	Contactor 2 (Downword direction)
27	DoorOpen	Q1.2	
28	DoorClose	Q1.3	
29			
30	Serv_Status	VB60	if 1 then Served, 0 means not served (here VW52 [TempMEM] is to be serve)
31			
32	Door_Status	Q1.6	1 means OPEN, 0 means CLOSE
33			
34	ResetQueue	I2.7	Clear the content & EntryCount of the queue (Table).
35			
36	DoorClose_Timer	T62	Door will be closed within 30 seconds
37	DoorOpen_Timer	T63	Door will be opened within 30 seconds

Data Block

```
//
//Data Block
//
//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
//Some variables, which are to set with some
//initial values & to put them in PLC memory,
//before running the programme.
//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~



//Initializing Queue (FIFO Table)

MaxEntries_Qu 10            //VW0
EC_Qu  0                    //VW2

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

//VB60 = Serv_Status [1 means served, 0 means request pending]
Serv_Status 2#1             //VB60


//Next floor to serve. [Lift will be launched in the Ground (0th) floor]
TempMem 0                   //VW52


//Lift position will be stored in this memory
LiftPosMEM 0                //VW50, now it is replaced by a counter
```

Ladder Diagram

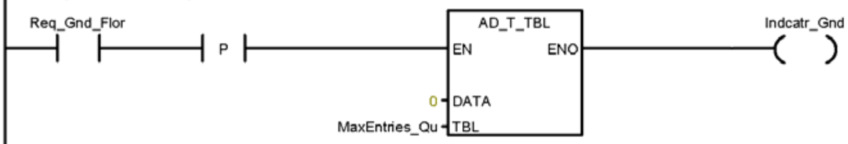
The ladder logic employed here consists of 19 networks. Among them first five are responsible for status checking of different push buttons and the next five are employed to keep a track of the current position of the lift. And then some condition checking is there to compare the current position of lift and the pending request to serve. Here is a snapshot of these first ten networks/ladders…

>>> PLC Based Elevator Controlling Unit <<<
~~~~~~~~~~~~~~~~~~~~~~~~~~

This ladder is applicable for any building (due to limitation of the PLC, not more than Seven Storied building; however by simply adding some ladders it can be used for any number of floors.)

Network 1

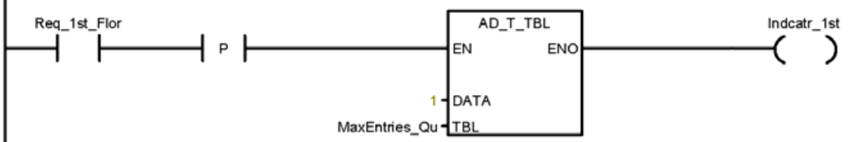
Tracking the request from ground floor & Lift



Symbol	Address	Comment
Indcatr_Gnd	Q0.0	Indicator of Ground floor request
MaxEntries_Qu	VW0	Maximum no. of entries in the queue / Starting address of Table
Req_Gnd_Flor	I0.0	Request coming from Ground floor

Network 2

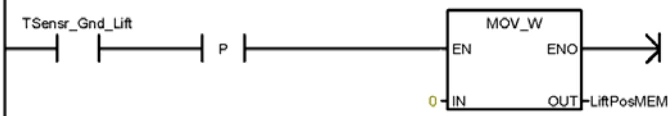
Tracking the request from 1st floor & Lift



Symbol	Address	Comment
Indcatr_1st	Q0.1	Indicator of 1st floor request
MaxEntries_Qu	VW0	Maximum no. of entries in the queue / Starting address of Table
Req_1st_Flor	I0.1	Request coming from 1st floor

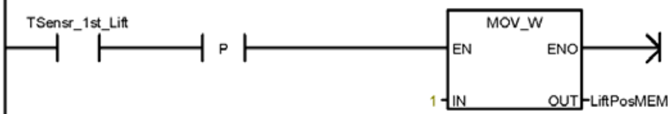
This was some of the ladders which are responsible for tracking the status of different push buttons. Now let's have a short look to the ladders tracking the touch sensors.

Network 6



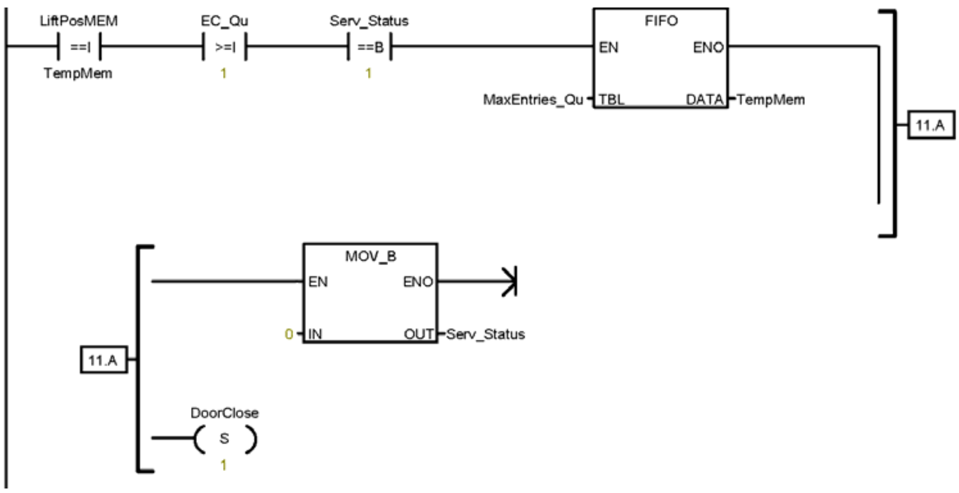
Symbol	Address	Comment
LiftPosMEM	VW50	Memory to store current Lift Position
TSensr_Gnd_Lift	I1.0	Touch sensor o/p - Lift is in ground floor

Network 7

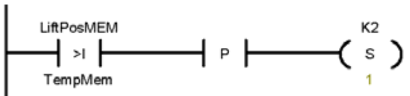


Symbol	Address	Comment
LiftPosMEM	VW50	Memory to store current Lift Position
TSensr_1st_Lift	I1.1	Touch sensor o/p - Lift is in 1st floor

Also there are ladders for checking all required conditions…
~~~~~~~~~~~~~~~~~~~~~~~~~~

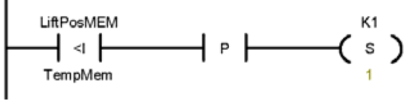


Based on the result of this condition checking the output (i.e. motor drive) is driven by the following ladder…

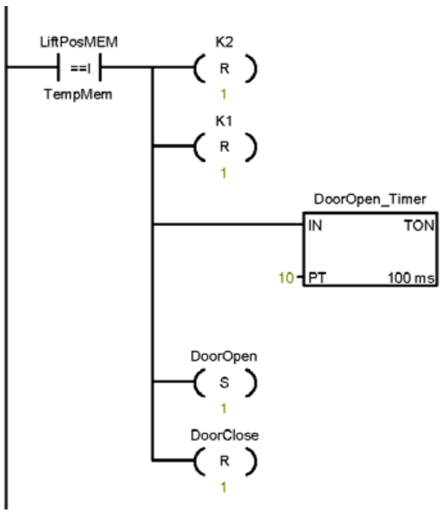


Symbol	Address	Comment
K2	Q1.1	Contactor 2 (Downword direction)
LiftPosMEM	VW50	Memory to store current Lift Position
TempMem	VW52	Next floor waiting for service

Network 13

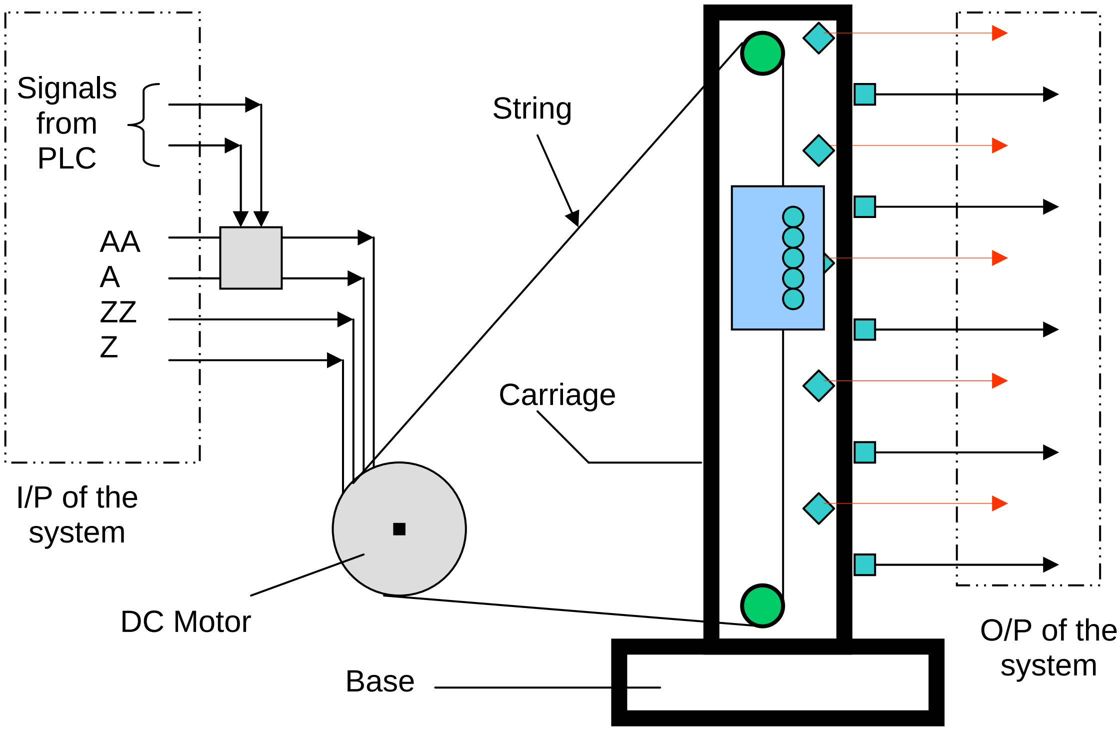


The door of the lift is also automated based on the movement of lift i.e. when the lift is moving the door of the lift will be closed and for rest of the time it will be opened. Following are the ladders to achieve that goal…



Schematic Of Model

In addition to the ladder logic a model has been also designed to have a proper visualization of the system. The schematic of that model is displayed here…



Model Description

The model consists of

- A small wooden box in place of original lift.
- A wooden carriage which helps the lift to up & down smoothly.
- A pulley system.
- A DC motor, driving the pulley to make the lift up & down.
- A hollow base to add weight to the system.

Components Attached With The Model

Some electronic components are attached with this model. Those are –

- Push button (in every floor)
- Push Button (within the lift)
- Touch sensor (in every floor)

Input Of The System

Inputs of this hardware are

- From PLC
  - ✓ Signal to drive the motor in clock wise direction, and
  - ✓ Signal to drive the motor in anti-clock wise direction.
- External voltage source to supply power to the DC motor.
- Five pushbuttons in every floor and another five within the lift, which takes input from the user.

Output Of The System

- Signal coming from touch sensors placed in different floors.
- Signal coming from push buttons placed in different floors & within lift.

Specification Of The PLC Used

- CPU Number: Simatic S7-200 (Siemens)
- No. of I/Ps available: 24
- Number of O/Ps available: 16
- Software Used: Step-7 MicroWin32 (Siemens)
- I/P Used: 10
- O/P Used: 4

Future scope of Improvement

This model can be improved further as described below-

- Implementing some techniques like Nearest-Floor-First or Floor-Having-More-People-First to save both time and consumed power.
- Adding weight sensor within the lift to set a maximum limit of weight the lift can carry.
- Also adding weight sensor to each floor to keep track that which floor has the maximum crowd.
- A weight counter balancing technique should be employed to operate it practically.
- More security (like ringing of an alarm when the weight of the lift crosses the preset maximum level) may be employed.

Conclusion & Bibliography

Chapter – 3

Conclusion

Before starting this project it was a challenge for us to develop proper ladder logic as we were beginner in the field of PLC programming. Gradually we managed to design the ladder by practicing different kinds of PLC programming. After designing the ladder I faced another challenge to interface the hardware system (i.e. model of the lift) with the PLC. By dividing the whole interfacing module into different parts I have also finished it successfully.

As it is a mere model, it may not match totally with the components used practically. But it can give a good visualization of the practical Elevation Operation. Also PLC is not used in Elevator generally. Still we have used PLC as elevation process controller because it is a good area to apply the full strength of PLC.

Bibliography

- www.plcs.net
- www.wikepedia.com
- www.google.com
- Materials supplied by college