The PLC Learner
An Integrated Approach towards Smart Simulation

Sandip Boral

MECON Limited
50, Chowringhee Road, Kolkata-700071, India

Abstract

The goal of this project is to promote the interest of Engineering Students and the Technical Personnels in the field of Industrial Automation with use of Programmable Logic Controller in a broad range. The objective is not great depth but the presentation is thorough enough to give the user the basic theory with different types of live examples.

'PLC Learner' is a MS DOS based Simulation Software, which supports the linear programming of SIEMENS PLC, STEP5 with STL representation. The simulator has an Integrated Development Environment having different features like Smart Editor, Compiler, Run Time Simulator and User Friendly Help System etc. It offers 128 bytes of Input & 128 bytes of Output modules, 128 bytes of Timers (including Pulse, Extended, On-Delay, Stored On-Delay and Off Delay mode), 128 bytes of Counters (including Up, Down and Up-Down mode) and 256 bytes of Internal Flash Memory.

Keywords: Introduction, Functional Units, Working Principle, Special Features, Programming, Development and Conclusions

1: Introduction

Programmable Logic Controller [PLC] plays an important role in modern 'Industrial Automation'. In the late 1960 PLC was first introduced. The primary reason for designing such a device was eliminating the substantial cost involved in replacing the complicated relay based machine control systems. There are so many companies who manufacture PLC. Some of the market giants are as follows: Allen Bradley, Siemens, ABB, Rockwell, Messung, Alstom, Control Microsystems, Festo, Hitachi, Idec, Honeywell, Mitsubishi, GE-Fanuc, AEG, Omron, Reliance Electric, Toshiba etc.

Now, these controllers are more or less same in physical design or hardware, but they vary in their software. Each brand has developed their own software and they maintain an international standard (IEC 1121-3).

An *Advanced Graphics User Interface* is designed to interface between the programmer and the simulated I/O modules via Mouse and Keyboard. There is an *On-screen Switchboard* like the simulation panel where 128 bytes of Inputs and 128 bytes of Outputs are available. The user can directly interact with these input bits by the switch On/Off. And the output bits are seen in the modules either On/Off according to the program.

2: Functional Units

Operating System: The OS contains the system programs that determine how the user program is executed, how inputs and outputs are managed, how the memory is divided, OS is fixed and cannot be changed.

Program Memory: In order to safely store the control program the controller needs the EEPROM memory sub module. Programs that are available in the EEPROM are copied into internal program memory i.e. RAM.

Process Image Table: Signal states of input and output modules are stored in the 'Process Image Tables'. It is the reserved area in the RAM. It is classified into two types:
Process image input table (PII)
Process image output table (PIQ)

Timers, Counters and Flags: The controller has timers, counters and flags available internally that the control program can use. The timer and counter parameters are stored in the reserved areas of the RAM.

Arithmetic Unit: The arithmetic units (ALU) consist of two accumulators, ACCU1 and ACCU2. the accumulators can process byte and word operations.

Processor: According to the program, the processor executes the instructions. It processes the inputs, outputs, flags, timers and counters.

Serial Interface: The programmers (PG), operator panels and monitors can be connected with the controller. The serial port is used for doing the same.

External I/O Bus: The I/O bus is the electrical connection for all signals that are exchanged between the CPU and the controller module.

3: Working Principle

PLC works by continually scanning the entire system. Mainly there are three steps that are as follows:

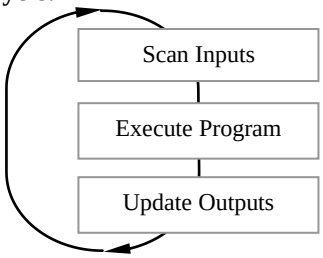
Check the Input Status: First the PLC takes a look at each input to determine if it is on/off. It records this data into its memory to be used during the next step.

Execute the Program according to the Inputs: The PLC executes the user program one instruction at a time. Since it is known from the previous step that which inputs are on and which are off, so it will be able to decide about the output status according to the program.

Change or update the Output Status: Finally, the PLC updates the status of the outputs. It updates the outputs based on which inputs were on during first step and the results of executing the programs during second step.

After the third step is over the PLC goes back to the first step.

PLC Program Cycle:



The Response time = Input Response Time + Program Execution Time + Output Response Time
Generally, the PLC Program cycle should be less than 150 ms.

4: Special Features

The simulator has a 'Program manager', has specially four important features i.e. Smart Editor, Compiler, Run Time Simulator and User Friendly Help System. They are briefly discussed below:

4.1: Smart Editor

The various techniques are used in the 'Smart Editor'**.** It automatically formats the PLC statements in a systematic way. There is the color code for different parts of the statements. If any statement is properly written, immediately the color of the different operand, operand ID & operand parameter indicate that it is written correctly. Else there will be 'light gray' color that indicates the statement is written wrongly. It also erases the extra parameters written in the programs, which is unwanted. There is an option for writing comments in the editor. The comments can be written both at the top of the program as well as at the right hand side of the statements. The special character '/' is allotted for separating comments from the statements. The color code 'cyan' denotes the comments.

Color code of the statements
Light Green denotes the Operation
Yellow denotes the Operand ID
Intense White denotes the Operand Parameter

There are two different scroll bars, one for scrolling the editor contents and another for scrolling the segments. The editor color is blue where the statements are displayed prominently.

4.2: Compiler

This is the compiler of this software, which is an important tool in the program manager. After writing the PLC programs the 'Compiler' can easily compile the programs written in different segments.

At the time of compilation all the segments in the file are checked thoroughly and the message box shows the number of errors if found and jumps to the particular line of the particular segment where the errors found.
If there is any syntax error or the operand parameter range exceeds the limit or the program is not written in proper way or there is an abnormal use of brackets then the compiler shows the errors also the color is changed in the particular line.
At the time of compilation, those segments, which are disabled, are not compiled. After the error check if no error found then from all the segments in the file the PLC program code is interpreted to an intermediate object code for executing the PLC program.

4.3: Run Time Simulator

This is the most attracting tool in this simulation software. Here the simulation is executed from the object code.
Here the Input and Output modules are found. An 'Advanced Graphics User Interface' is designed to interface between the programmer and the simulated I/O modules via Mouse and Keyboard. There is an on-screen switchboard like the simulation panel where 128 numbers of Input bytes and 128 numbers of Output bytes are available. The user can directly interact with these input bits by the switch On/Off. And the output bits are seen in the modules either On/Off according to the program.

4.4: User Friendly Help System

There is a 'User Friendly Help System' with this simulator. By pressing right button of the mouse the help is activated. The help system displays the detailed descriptions of each and every operation of PLC.
This is another important tool in the program manager where the help system is available. There are the list of help items in the help system selecting which a help editor displays the contents of the particular item. The special feature is that the user can create, modify and delete the help files. In MS-DOS editor the help files can be written and those files should be saved in the help folder "…\PLC\HELP*.HLP". Automatically that files can be accessed from the help system in the program manager.

5: Programming

A control program specifies a series of operations that tell the controller how it has to control the system.
Method of Representation: Statement List (STL) represents the program as a sequence of operation mnemonics. A statement has the following format:

Operation	Operand	
	Operand ID	Operand Parameter

	Range	Operands	Descriptions
I	0.0 to 127.7	**Inputs**	Interfaces from the process to controller
Q	0.0 to 127.7	**Outputs**	Interfaces from the controller to process
F	0.0 to 255.7	**Flags**	Memory for storing intermediate results
T	0 to 127	**Timers**	Memory for implementing timers
C	0 to 127	**Counters**	Memory for implementing counters

Operand Areas: This simulator has the following operand areas.

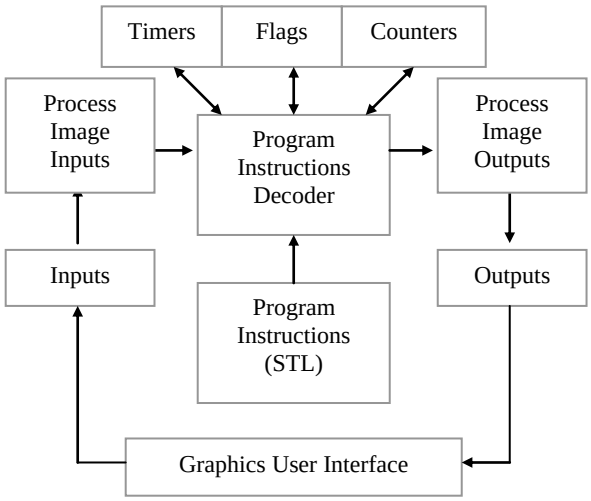
6: Development

6.1: Requirement Analysis

In origin of most software systems is the demand. The software system itself is created by the developer and finally, the completed system will be used by the end users. Thus there are three major parties interested in a new system, the clients, the users and the developer. The problem is that the developer usually does not understand the client's problem and application area. This causes a communication gap between the demand and supply. A basic purpose of 'Software Requirement Specification' is to bridge this gap.
Now, in this simulator the developer had the demand of simulation software by which he can practice the PLC programming. As he has developed this software so there is no communication gap between the demand of user and the supply of developer.
A high quality SRS is a prerequisite of high quality software. Hence, the quality of the SRS impacts customers and developer satisfaction.

6.2: Planning

Software development is a highly labor intensive activity. It has been seen that project management activities can be viewed as having three major phases: project planning, project monitoring and control.
The input to the planning is the SRS. During planning all the requirements are planned and during project control the plan is executed and updated. The basic goal of planning is to look into the future, identify the activities that need to be done to complete the project successfully. Ideally, all the future activities should be planned.
To ensure the final product of high quality, some quality control activities must be performed throughout the development.
The following planning was done:

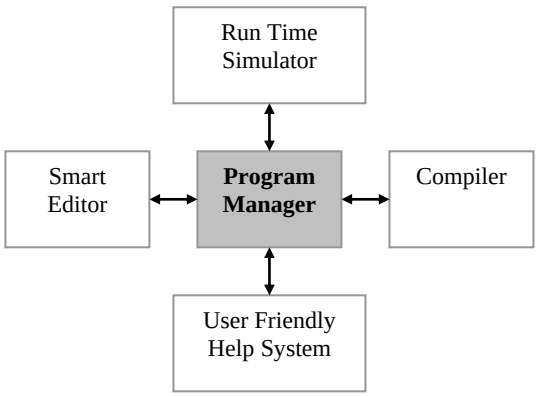


6.3: Designing

The design of a system is nothing but the blueprint of a plan for a solution of the system. The design process often has two levels. The first level the focus is on deciding which modules are needed for the system and how the module should be interconnected. This is called 'Top level design'. In the second level, the internal design of the modules is designed to satisfy the requirement. It is called 'Detailed design'.
An efficient system is one that consumes less processor time and requires less memory. Simplicity is the most important factor in design of a software system. The design of the system is one of the most important factors affecting the maintainability of a system. Creating a simple and effective design of a large system can be extremely complex task. As designing is a creative activity, it cannot be reduced to a series of steps that can be simply followed, though guidelines can be provided.
The principles that can be used in design are the same as those used in problem analysis. Though it seems that

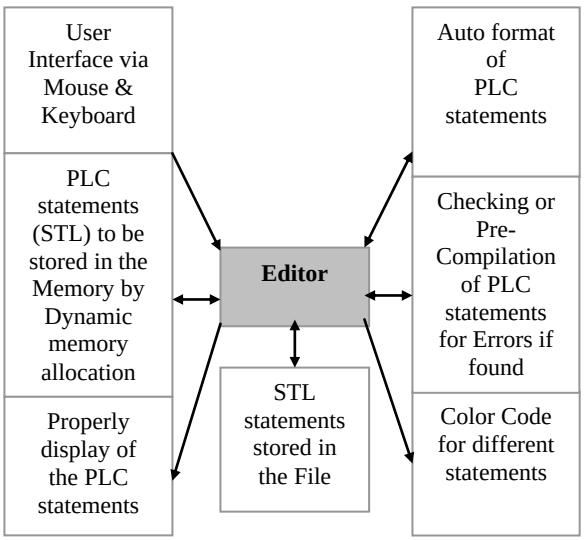
analysis and design are similar, but there is some major differences. The analysis is in the problem domain but the design is in the solution domain.
For software design, the goal is to divide the problem into manageably small pieces that can be solved separately. However, the different pieces cannot be independent for each other, as they together build up the system.
A design methodology is a systematic approach to create a design by applying a set of techniques and guidelines. A design can be object oriented or function oriented.
In this simulation software there is a function-oriented approach. In this type of approach, the design consists of module definitions, with each module supporting a functional abstraction.

Top Level Design:

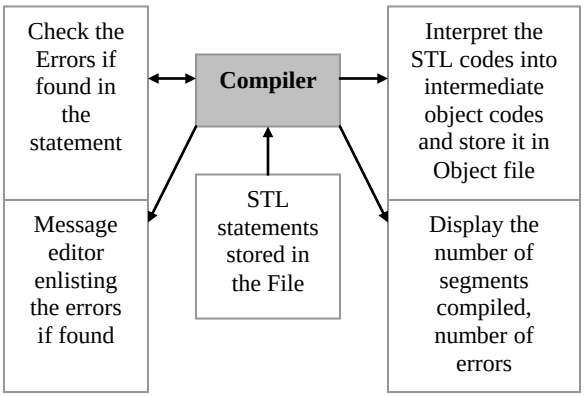


Detailed Design:

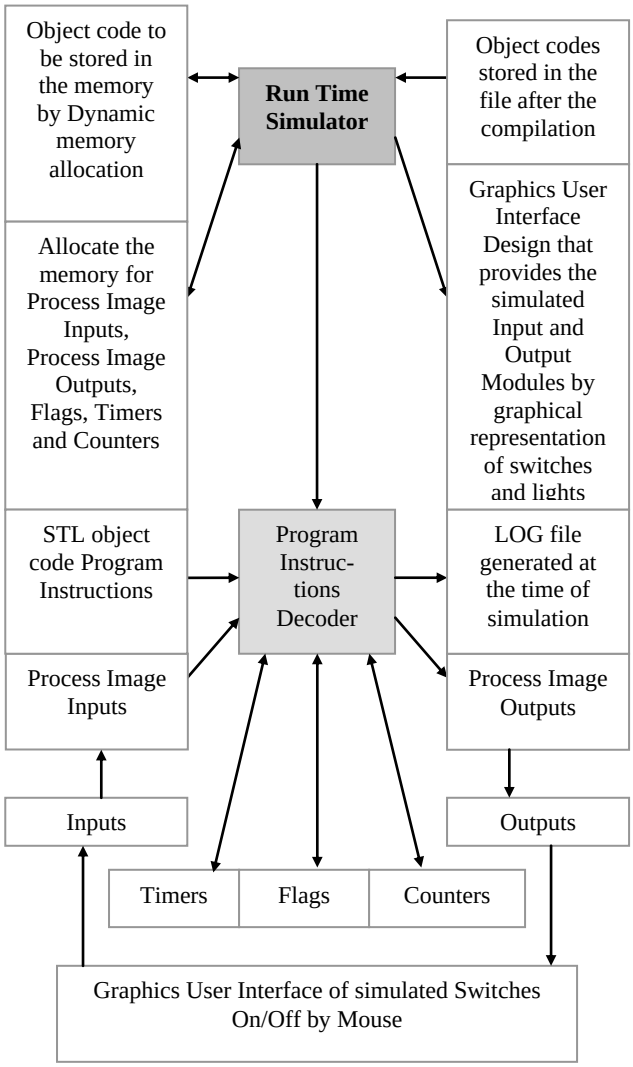
6.3.1: Smart Editor



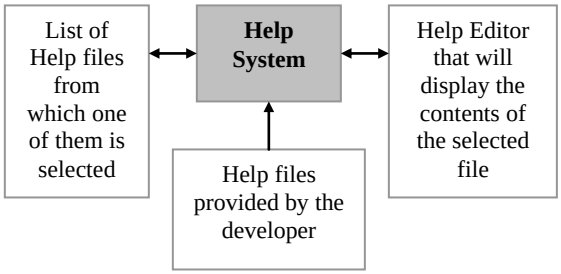
6.3.2: Compiler



6.3.3: Run Time Simulator



6.3.4: User Friendly Help System



6.4: Coding

The goal of the coding is to translate the design of the system produced during the design. For a given design, the aim is to implement the design in the best possible manner. During implementation, it should be kept in mind that the programs should not be constructed so that they are easy to write, but so that they are easy to read and understand.
There are many different criteria for judging a program, including readability, size of the program, execution time and required memory.
In a top down implementation the coding starts from the top and proceeds to the lower levels. First the main module is implemented, then its subordinates and their subordinates and so on. In a bottom up implementation, the process is reverse. The developments start from a module at the bottom level and then proceed towards the top.
In this case for developing this simulator, combination of two approaches is used during coding. The top module of the system contains the user interface for the user to 'look and feel' OK. In this case the procedure is top down. On the other hand, the bottom level modules typically form the 'service routine' that provide the basic operations used by higher-level modules. This is the bottom up approach.
It is impossible to provide an exhaustive list of what to do and what not to do to produce the code. The algorithms must be generated first to produce a good code. Here the following mentioned items are considered to write a good code i.e.

Variable names, User defined data types, Information hiding, Nested commands, Module size, Module interface, Program layout, Side effects, Robustness etc.

6.4.1: Programming Language

As the PLC handles Simulated Timers, Simulated Counters as well as bit locations of registers, so **C Language** is the most appropriate for coding.
Also, for excellent user interface the ROM-BIOS functions for mouse, keyboard and graphics adapter must be used. **Assembly Language** is the most efficient language for hardware interaction via software.
Beside this, the PLC Program cycle should have very low time period. Because, PLC samples input data and it has timers that can count time (10 ms - 9990 s) for discrete control and so, the sampling frequency must be very high and sampling time must be very low (μs – ms).

6.4.2: Compiler

Turbo C ++, Version 3.0
Copyright © 1990, 1992 by Borland International, Inc.
(Here any assembler is not used, as Turbo C++ compiler supports the Inline Assembly)

7: Conclusions

The software is available with Installation Disk in IDE format. The user interface both in text mode and graphics mode is very strongly designed, so user can easily handle the software. Also the robustness is in higher degree making it more reliable. Generally those who are the beginners in PLC programming can easily use this software. In training and learning purpose this simulator helps the beginners a lot.

8: References

[1] SIEMENS. SIMATIC S5 100U Programmable Controller, "*System Manual Siemens Ltd*."

[2] Brey, *"Intel Microprocessors, Architecture, Programming and Interfacing"*

[3] Pankaj Jalote, *"An Integrated Approach to Software Engineering"*