NATIONAL UNIVERSITY OF SINGAPORE

EE3302/EE3302E Industrial Control Systems

E2: PLC Programming for Sequence Control

1. Objectives

The experiment is designed to provide experience in programming a modern IEC-compliant PLC system for sequence control of typical industrial processes.

2. Introduction

A sequence control system is one that performs a set of operations in a prescribed manner. The automatic washing machine is a familiar example of sequence control: the control system performs the operations of filling the tub, washing the clothes, draining the tub, rinsing the clothes, and spin drying the clothes.

Before 1970, large relay panels were used to control sequential operations.In 1968, General Motors Corporation specified the design criteria for the first programmable logic controller (PLC). The purpose was to replace the inflexible relay panels with a computer-controlled soild-state system. The project succeeded beyond anyone's dreams. Programmable logic controllers have gone beyond replacement of relay panels to include PID modules for process control and communications interfaces that make it possible to link programmable controllers into an integrated manufacturing operations. PLCs have since become an indispensable portion of industrial automation and by the 80s, there were already a very wide range of PLC and programming languages in the market, many of which were proprietary. Having so many languages often result in misunderstandings which may have disastrous results. In 1993, a working group within the International Electro-technical Commission (IEC) was formed to look into setting up a PLC language standard, otherwise known as the IEC 1131-3. The major players in the market like Siemens, Allen-Bradley, Modicons, Omron etc. have since adopted the standard.

The experiment will be based on the OMRON IEC-compliant PLC. Students will be trained on various aspects of sequence control as in control analysis, logic operations and use of standard function blocks like the Timer and the Set/Reset Logic (SR).

3. Apparatus

The equipment set up required for this experiment consists of :

- Pentium workstation
- OMRON SYSMAC PLC with I/O modules
- CX-Programmer programming software running on Windows XP
- Technology simulators:
    - Reaction Vessel
    - Traffic Light Junction

Figure 1 shows the OMRON SYSMAC PLC which you will be working with in this experiment.

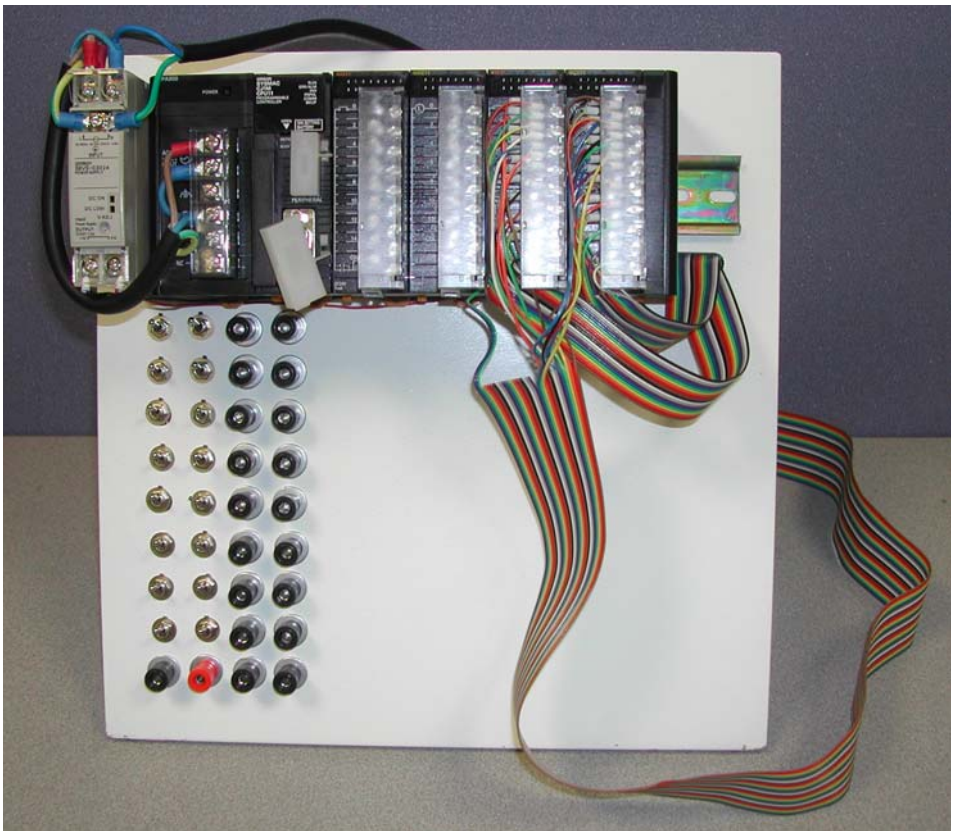

Figure 1: OMRON SYSMAC PLC

4. PLC Program Design for a Reaction Vessel

In the first portion of this experiment, step-by-step instructions are given to design and implement PLC program for a Reaction Vessel simulator. First, let us understand the basic operations of the reaction vessel.

4.1 Operations of the Reaction Vessel

The process schematic of the reaction vessel is shown in Figure 2. A chemical process is to take place in a reaction vessel at a specific temperature and at a specific pressure. The reaction vessel has a thermal detector for measuring the temperature and a pressure gauge for the pressure. Temperature and pressure are regulated via the following three actuators:

- Heater H
- Cooling-water inlet K and
- Safety valve S

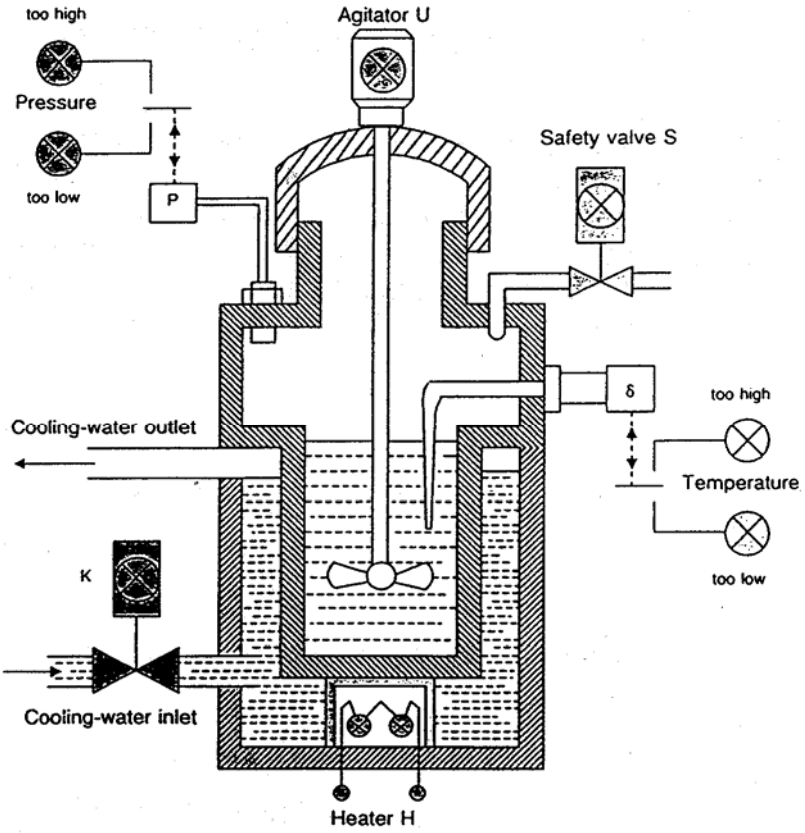

Figure 2: Process Schematic of Reaction Vessel

The enabling conditions for the actuators are given as follows:

Safety valve S is activated when

- pressure P is too high and
- temperature δ is too high or normal

Cooling-water inlet K is activated when

- temperature δ is too high and
- pressure P is too high or normal

Heater H is activated when

- temperature δ is too low and
- pressure P is not too high

or

- pressure P is too low and
- temperature δ is normal

If actuator K (cooling-water inlet) or H (heater) has been activated, agitator U must also be activated.

The reaction states must also be correctly shown on the indicator lamps as follows :

Starting state : Pressure P too low

Normal state : Pressure P is normal

Alarm : Pressure P too high

The abovementioned conditions are to be implemented using a PLC. Specifically, you will design the ladder program and implement it on the Omron PLC. To maximize the benefits from the experiment, it is advisable for you to do some preparatory homework and come to the laboratory with at least a rough PLC program for this simulator.

4.2 Create Project Structure

At the Windows desktop environment, double click the CX-Programmer icon. The project window for the CX-Programmer will then be opened.

Click on the menu command **File→New** in the CX-Programmer. In the "Device Name" field, type a project name of your choice, e.g., "reactor", "traffic", etc. Select the "Device Type" as CJ1M, which corresponds to the PLC processor used here. Keep the "Network Type" field as Toolbus. This sets the communication link between the PC and the PLC, which utilizes the COM1 serial port in this case. Click on **OK** to create your new project. You will see a main window similar to the screenshot in Figure 3.

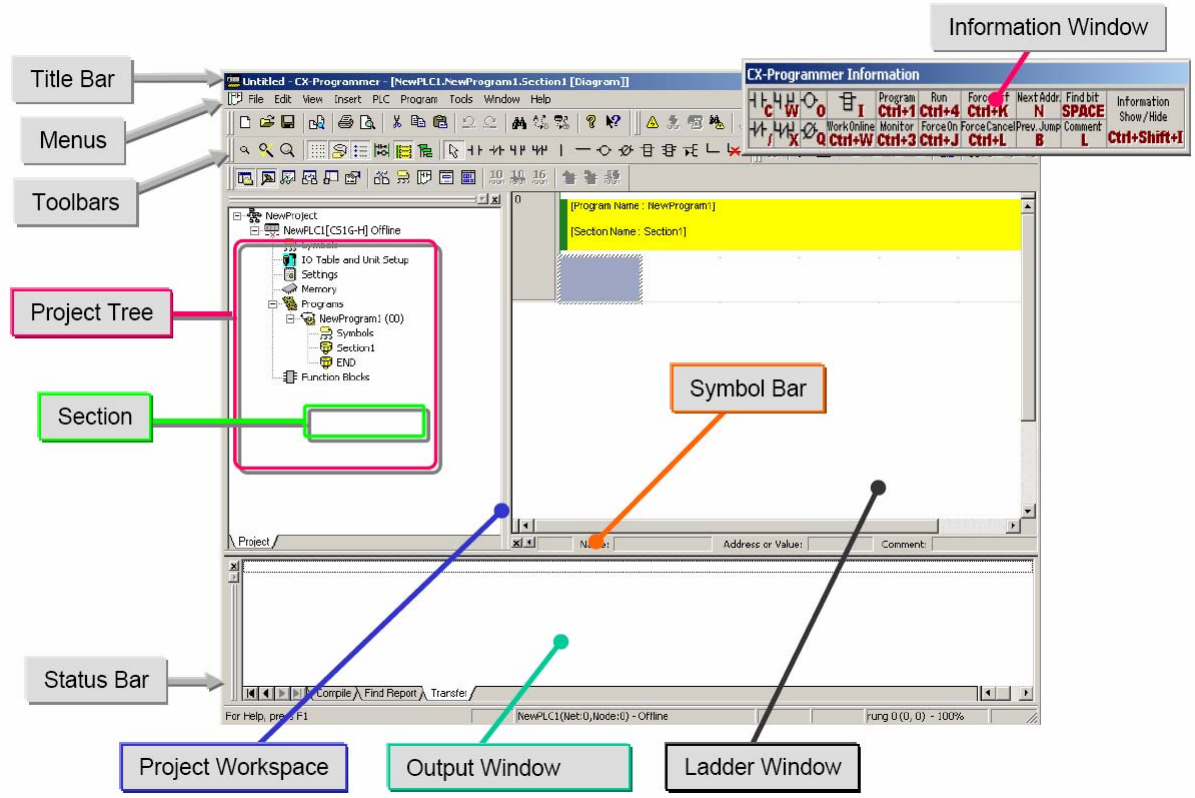

Figure 3: Screenshot of CX-Programmer Main Window

4.3 Programming the PLC

In this subsection, information on creation of a symbol table to match variable names to addresses will be given, as well as quick instructions to input the ladder program to the OMRON PLC.

4.3.1 Personalized Symbol Identification

Before the actual coding, we need to first create a symbol table. The symbol table may be viewed as a declaration of the match between the variables used in the program and the actual physical I/Os or temporary storage they represent. Basically, this table provides for the use of named addresses as discussed in the lectures. There are two sets of symbol table in the Project Workspace, a global symbol table, which include standard variables for declaration of all symbols used in the entire project; and a local symbol table which exists in each program submenu. Double click on the symbol icon just below your project name in the project workspace to view the various default symbols.

To begin, open the excel spreadsheet symbols.xls, copy and paste the entire symbol table in the first spreadsheet: “reaction vessel” to your program’s local symbol table. These symbols match the physical input and output addresses of the PLC to the names of the actual pin (I/Os) on the reaction vessel board. Note each symbol and its associated comments to identify the I/O it represents.

For all other variables, you may choose your own variable names. There is no need to define or allocate memory addresses for these variables as they will be automatically added to the local symbol table when you create a new variable in the program code. Note that the variable names are case sensitive! i.e., state1 and State1 are two different variables.

4.3.2 Entering the Ladder Program

Double click on “section1” under the Project Tree. You may now enter your ladder program through the ladder window to fulfill the logic requirements specified in Section 4.1.

4.4 Running the Ladder Program on the PLC

Once the program is ready, click on **Program→Compile** to compile the program. Any error which requires debugging will be displayed in the status bar. If there is no error, you may proceed to go online and communicate with your PLC. This is achieved by clicking on **PLC→Work Online**. Click **Yes** when prompted. The next step would be to transfer the program to the PLC. Click on **PLC→Transfer→To PLC** (select only to transfer the program), click **Yes** when prompted. All the other boxes must be left unchecked as depicted in the screenshot of Figure 4.

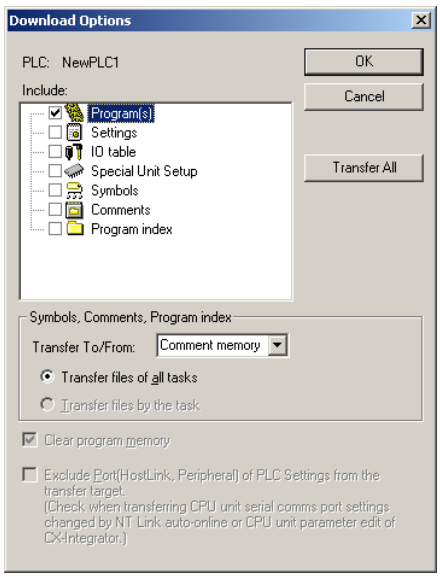

Figure 4: Transfer of Program to PLC

You should ask the lab demostrator to verify the results after your program is running correctly. You may proceed to the next section if everything is in order.

5. PLC Program Design for a Traffic Light Junction

In this portion, we shall design and implement PLC for a traffic light junction using the Omron PLC. A schematic of the traffic light junction is shown in Figure 5.

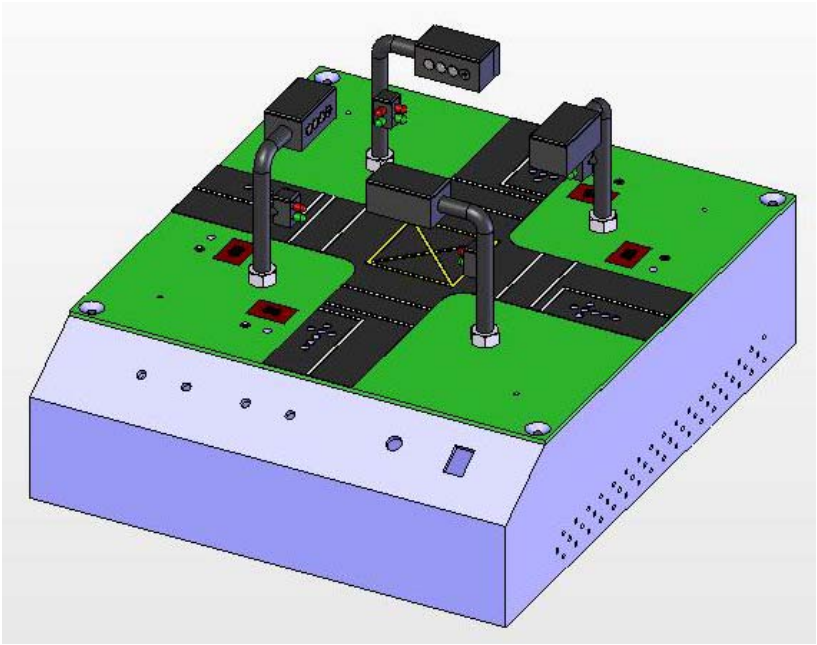

Figure 5: Traffic Light Junction

5.1 Operations of Traffic Light Junction

Due to laboratory time constraint, a basic PLC program for the traffic light control system has been designed and will be provided to you. Under the CX-Programmer environment, open the project “Traffic”. The laboratory demonstrator will discuss the general logic flow of this program with you. The logic/rules governing the basic traffic light junction control system are given below:

1. The traffic light system will be initiated with the traffic flowing along the vertical direction (Green light activitated along the vertical direction). The traffic in the horizontal direction should then be at standstill (Red light activated along the horizontal direction).
2. After 15 seconds, the vertical traffic light will turn amber for 2 seconds before turning red.
3. When the vertical direction traffic light turns red, the horizontal direction traffic shall begin flowing (Green light activated along the horizontal direction).
4. After another 15 seconds, the horizontal direction traffic light will turn amber for 2 seconds before it turns red.
5. The cycle is completed. The loop involving steps (1)-(4) will continue indefinitely as long as the traffic light system is on.

5.2 Additional Functions

Certain functionalities are not included in the traffic light junction control program provided to you. We shall now add in one of these functions. The laboratory demonstrator will assign one of these specific functions to you.

5.2.1 Pedestrian Crossing (Option 1)

The logic/rules governing the pedestrian crossing are as follow:

1. There will be an addition of four inputs to the system (two each for the vertical and horizontal directions) to indicate the presence of pedestrians needing to cross the junction along these directions.
2. If none of these inputs are activitated, the traffic light control system will function as per the basic program.
3. If either of the inputs for the vertical (horizontal) direction is activitated, when the next cycle of vertical (horizontal) traffic flow begins, the corresponding pedestrian crossing indicator must be activated.
4. The time duration for pedestrian crossing is set at 10 seconds.
5. After 5 seconds, the pedestrian crossing green indicator begins blinking.
6. After 10 seconds, the pedestrian crossing red indicator will turn on.

5.2.2 Right-turn Function (Option 2)

The logic/rules are as follow:

1. After the vertical (horizontal) direction traffic light turns amber followed by red, instead of switching traffic flow to the other direction immediately, the right turning arrow indicator will be activated.
2. The right turning arrow indicator will be activated for 6 seconds.
3. After 3 seconds, the indicator starts blinking.
4. At the end of 6 seconds, traffic flow switches to the other direction.

5.2.3 Intelligent Traffic Control (Option 3)

The inclusion of monitoring devices such as electromagnetic proximity sensors will give us a rough indication of the traffic conditions, i.e., whether there is a high volume of traffic awaiting to cross at a particular junction. With this information, we will be able to fine tune the traffic control system to change the traffic light timing to adapt to the traffic conditions. When the vehicular traffic is low, the traffic light can change more frequently to minimize waiting time. When the vehicular traffic is high, the traffic light can remain green for a longer period of time. Hence, the following set of logic/rules can be used:

1. There will be two additional sets of inputs here, representing the proximity sensors for traffic volume detection (i.e, the sensors will indicate high or low traffic along the vertical and horizontal directions)
2. When the vertical (horizontal) direction traffic is heavy, the green light for the vertical (horizontal) direction will be activated for 20 seconds instead of the usual 15 seconds.
3. However, when the vertical (horizontal) direction traffic is low, the green light for the vertical (horizontal) direction will be activated for 10 seconds instead of the usual 15 seconds.

5.3 Modular Programming

The additional functions can be incorporated in the basic program through modular programming. In modular programming, individual modules (programs) with its own functionalities are independently created as function blocks. Subsequently, they may be re-used and integrated into the overall main system. The followings are some guidelines which may assist you in your programming:

1. First, identify the necessary I/O points and “symbolize” them in the local symbol table.
2. Next, create your own modular ladder program based on the functional requirements.
3. Subsequently, identify the places in the overall traffic control system ladder program, where the necessary modifications are required to activate your modular ladder program as well as the modifications required for your modular ladder program to influence the overall flow.
4. Finally, you have the complete ladder program to provide the basic and the additional function.

6. Report

You should ask the laboratory assistant to verify the results after your program is running correctly. The program and results of each stage of the experiment should be logged and adequate comments and explanation on the logic of the program should be provided where necessary.