YEDITEPE UNIVERSITY
ENGINEERING & ARCHITECTURE FACULTY
INDUSTRIAL ELECTRONICS LABORATORY

EE 432 – INDUSTRIAL ELECTRONICS

EXPERIMENT 2
TRAFFIC LIGHT CONTROL SYSTEM FOR AN INTERSECTION USING S7-300 PLC

Introduction:

The main objective in this experiment is to:

- Understanding the programmable logic controller and its peripherals.
- Programming the PLC with the STEP 7 software.
- Applying the PLC to control the operation of a demand-actuated traffic light system in an intersection.

Equipments:

Table 1. List of Equipments

728 740	Traffic Light Crossing
730 800	PLC Basic Unit
3SB14 00-0A	Normally Open (NO) Push-button switches (x3)
	Siemens SIMATIC S7-300

General Information:

A Programmable Logic Controller (*PLC*) is a device that replaces the necessary sequential relay circuits for machine control. The *PLC* works by looking at its inputs and depending upon their state, turning on/off its outputs as depicted in Figure 1. Today, *PLC*s are used in many "real world" applications such as machining, packaging, material handling, automated assembly or countless other industrial processes.

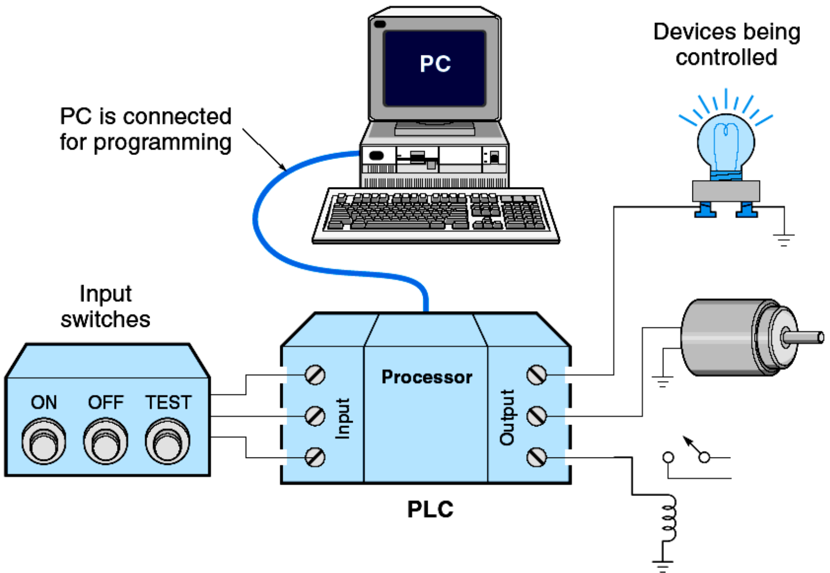


Figure 1. Operation principle of a PLC [1].

SIMATIC S7-300 *PLC* will be used to control the operation of a demand-actuated traffic light system in an intersection.. The general features of S7-300 are; it has a modular *PLC* system developed for medium performance projects, it features different modules for different automation problems, it is easy to re-program the device when an improvement is needed for the process, it is possible to be connected to communication networks such as MPI, Profibus and Industrial Ethernet, it has a vast amount of command set for programming.

The S7-300 mounted on a rack is shown in Figure 2. It includes the following components:

- **Power Supply (PS):** It is required to supply the modules connected to the *PLC*. There are three different models for 2 A, 5 A, and 10 A max.
- **CPU:** The central processing unit.
- **Interface Module (SM):**

  1. Digital Input Modules: 24 V DC transistor,
  2. Digital Output Modules: 24 V DC transistor,
  3. Analog Input Module: Voltage, Current, Resistor and Thermocouple,
  4. Analog Output Module: Voltage and Current.

- **Communication Module (CP)**

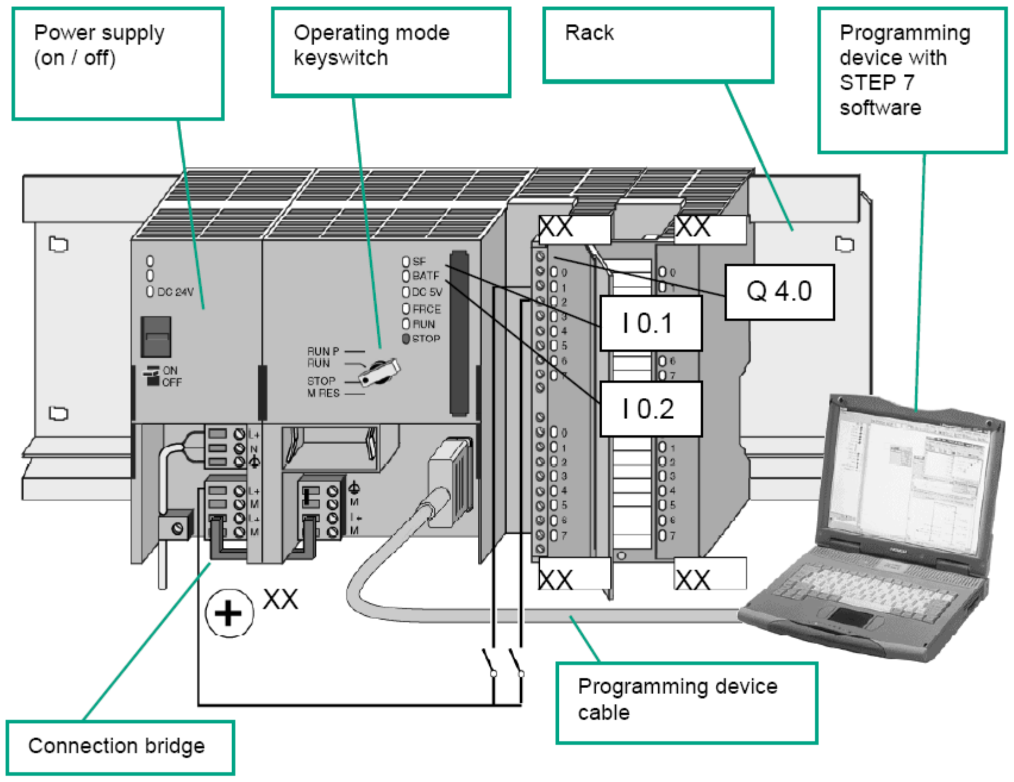


Figure 2. Main parts of a S7-300 PLC

Programming with STEP 7:

STEP 7 software is used to program the *PLC*. There are three different programming languages in the STEP 7: statement list (STL) for users who prefer programming in a language similar to machine code, Ladder logic (LAD) for users who are accustomed to working with circuit diagrams, and Function Block Diagram (FBD) for users who are familiar with the logic boxes of Boolean algebra. The S7 provides a vast variety of components into its component library which in turn is useful for variable applications from conventional control applications to applications with counters, timers, flip-flops, mathematical functions for industrial feed-back control processes such as *PID* controller design.

In STEP 7, we work with addresses (absolute addressing) such as I/O signals, bit memory, counters, timers, data blocks, and function blocks. An address is the abbreviation for a particular memory area and memory location, so it is important to correctly assign the input/output (I/O) address areas in the hardware configuration and specify them in the program (STEP 7). An absolute address comprises an address identifier

- I (input bit),
- IB (input byte-8 bits),
- IW (input word-16 bits),
- ID (input double word-32 bits),
- Q (output bit), QB (output byte),
- QW (output word),
- QD (output double),
- M (memory bit),
- MB (memory byte),
- MW (memory word),
- MD (memory double word),
- T (timer),
- C (counter)] and
- a memory location (for example, Q 4.0, I 1.1, M 2.0, FB 21).

Figure 3 shows the structure of an address for the input interface of a PLC.

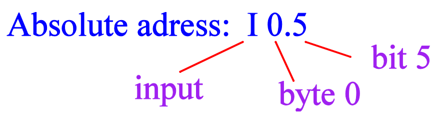


Figure 3. Address structure of a *PLC*.

Due to the hardware structure, every input and output is assigned a default absolute address. On the other hand, the absolute address can be replaced by a freely selectable (symbolic) name (e.g. Q 4.2: Automatic mode). Symbols are assigned independent of the programming language, that is, LAD, FBD or STL.

Programming Example

The circuit shown in Figure 4 is used to control the direction of rotation of a three-phase induction motor. There are three buttons that control the operation: two start buttons – one for clockwise (CW), one for counterclockwise (CCW) and a single stop button. Based on these inputs PLC controls the operation of two contactors by activating or deactivating relay coils. The direction of rotation in an induction motor can be reversed by exchanging any two wires of three-phases that are connected to the voltage source. Two contactors must not be on at the some time as this would definitely short circuit the three-phase supply.

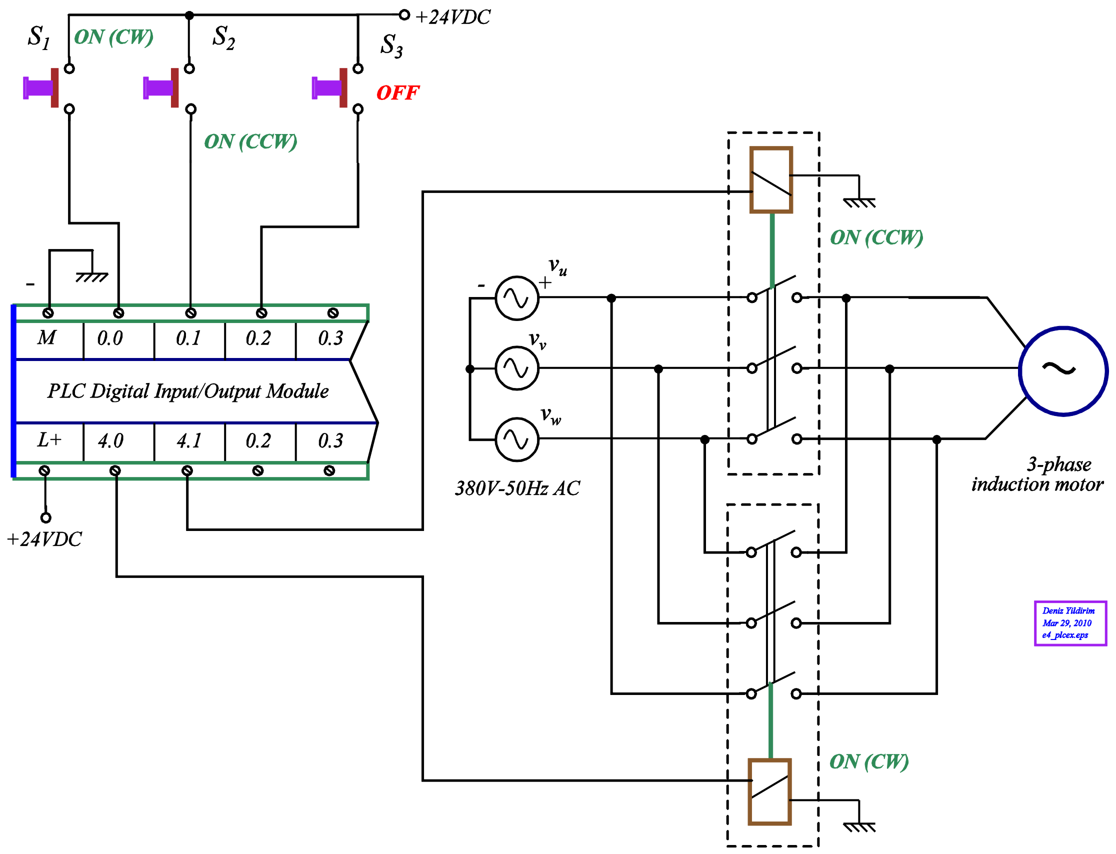
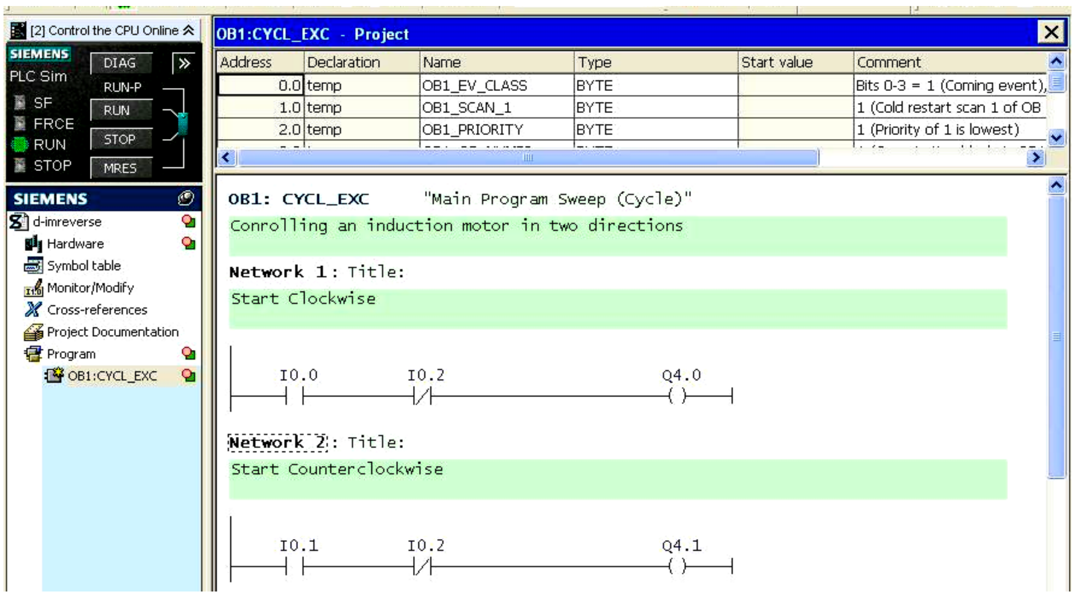


Figure 4. Controlling the direction of rotation employing a PLC.

The circuit shown in Figure 4 is used to control the direction of rotation of a three-phase induction motor. There are three buttons that control the operation: two start buttons and one stop button. When S_1 is pushed momentarily motor rotates in CW direction and pushing S_2 will rotate the motor in CCW direction. Pushing S_3 button at any time will stop the motor.

Figure 5 shows the ladder diagram for the operation of motor given in Figure 4. *I0.0*, *I0.1*, and *I0.2* are corresponding switches of S_1, S_2, and S_3, respectively. Figure 5a shows the simplest ladder diagram where continuous pushing of switches is required to operate the motor. In order to allow momentary pushing of buttons, a locking mechanism must be employed as depicted in Figure 5b. Since two contactors cannot be on at the some time, a protection mechanism must be employed to overcome this situation. The resulting ladder diagram is shown in Figure 5c where when one contactor is on, it automatically disables the other input.

Symbol names may also be assigned in the ladder diagram instead of using input/output addresses. Figure 5d shows the symbol table for assigning names to each input/output. Once names are assigned, ladder diagram is automatically updated as shown in Figure 5e.



OB1:CYCL_EXC - Project

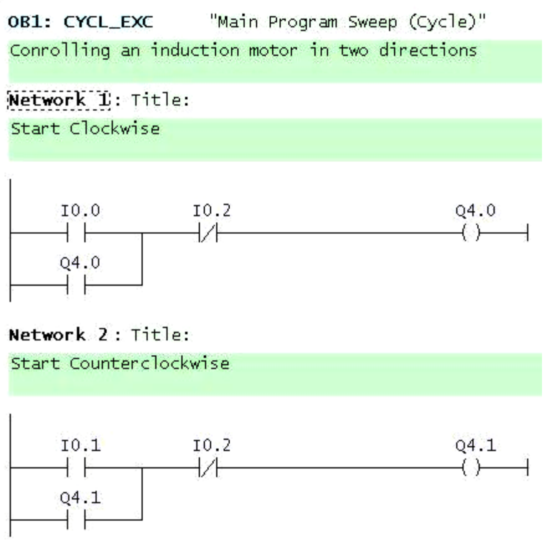
Address	Declaration	Name	Type	Start value	Comment
0.0	temp	OB1_EV_CLASS	BYTE		Bits 0-3 = 1 (Coming event),
1.0	temp	OB1_SCAN_1	BYTE		1 (Cold restart scan 1 of OB
2.0	temp	OB1_PRIORITY	BYTE		1 (Priority of 1 is lowest)

OB1: CYCL_EXC "Main Program Sweep (Cycle)"
Conrolling an induction motor in two directions

Network 1: Title:
Start Clockwise

Network 2: Title:
Start Counterclockwise

(a)

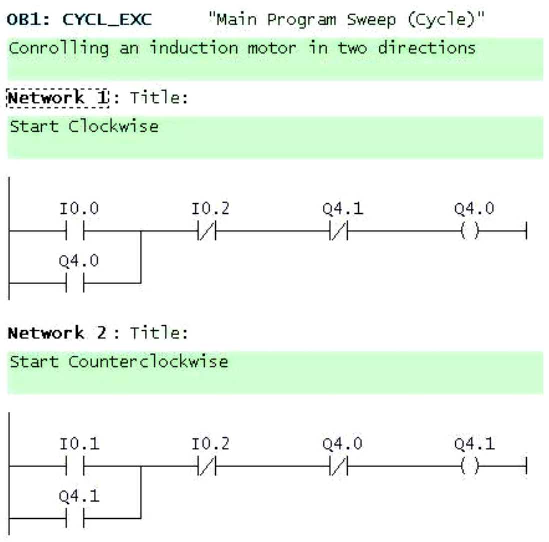


OB1: CYCL_EXC "Main Program Sweep (Cycle)"
Conrolling an induction motor in two directions

Network 1: Title:
Start Clockwise

Network 2: Title:
Start Counterclockwise

(b)

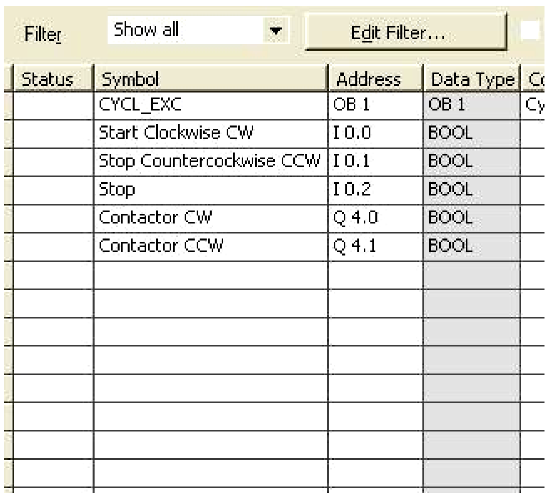


OB1: CYCL_EXC "Main Program Sweep (Cycle)"
Conrolling an induction motor in two directions

Network 1: Title:
Start Clockwise

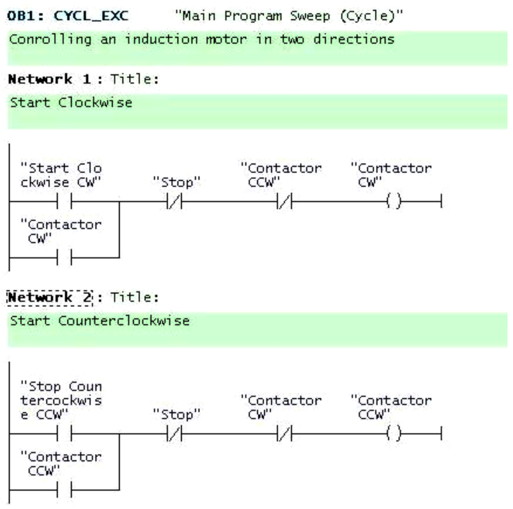
Network 2: Title:
Start Counterclockwise

(c)



Status	Symbol	Address	Data Type	Co
	CYCL_EXC	OB 1	OB 1	Cy
	Start Clockwise CW	I 0.0	BOOL	
	Stop Countercockwise CCW	I 0.1	BOOL	
	Stop	I 0.2	BOOL	
	Contactor CW	Q 4.0	BOOL	
	Contactor CCW	Q 4.1	BOOL	

(d)



OB1: CYCL_EXC "Main Program Sweep (Cycle)"
Conrolling an induction motor in two directions

Network 1: Title:
Start Clockwise

Network 2: Title:
Start Counterclockwise

(e)

Figure 5. Programming with LAD language for the circuit in Figure 4: (a) simplest ladder diagram (b) diagram with locking pushbuttons (c) complete ladder diagram (d) assigning names to symbols using symbol table (e) diagram with symbol names.

TIMER TYPES

Timers have an area reserved for them in the memory of your CPU. This memory area reserves one 16-bit word for each timer address. The ladder logic instruction set supports 256 timers. The timer value is written as:

Timer format: S5T# aH_bbM_ccS_ddMS

- where a = hours, bb = minutes, cc = seconds, and dd = milliseconds
- The time base is selected automatically, and the value is rounded to the next lower number with that time base.

For example: 500ms = S5T#500MS,   1min 10sec = S5T#1M_10S

There are five different timer types in Simatic Ladder programming as depicted in Figure 6 where each timer starts with an input signal.

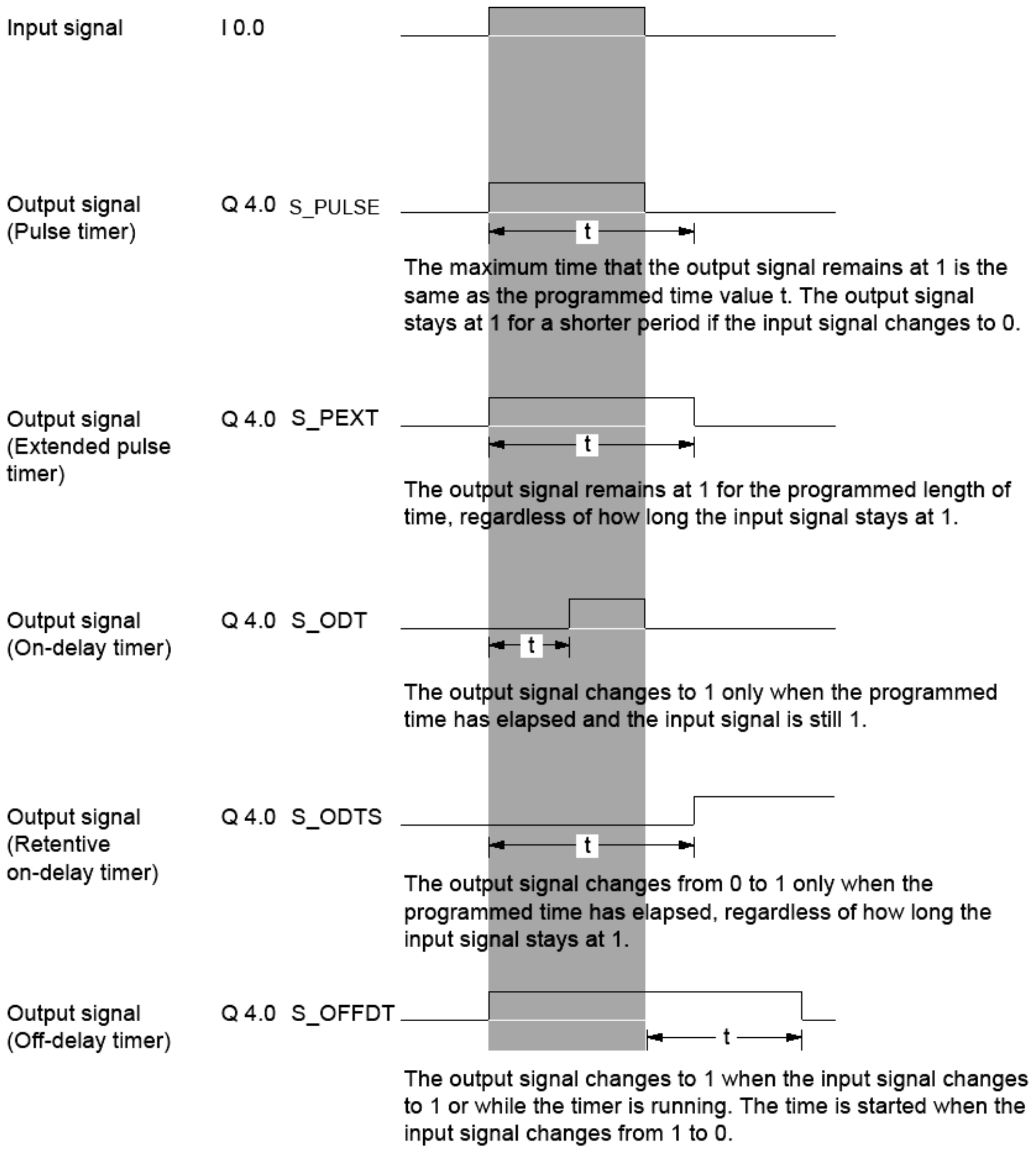
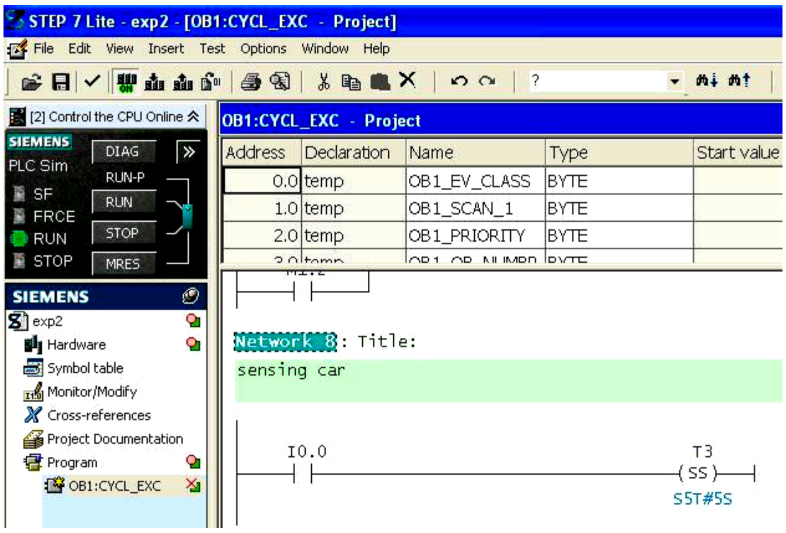

Input signal
I 0.0
Output signal (Pulse timer)
Q 4.0 S_PULSE
t
The maximum time that the output signal remains at 1 is the same as the programmed time value t. The output signal stays at 1 for a shorter period if the input signal changes to 0.
Output signal (Extended pulse timer)
Q 4.0 S_PEXT
t
The output signal remains at 1 for the programmed length of time, regardless of how long the input signal stays at 1.
Output signal (On-delay timer)
Q 4.0 S_ODT
t
The output signal changes to 1 only when the programmed time has elapsed and the input signal is still 1.
Output signal (Retentive on-delay timer)
Q 4.0 S_ODTS
t
The output signal changes from 0 to 1 only when the programmed time has elapsed, regardless of how long the input signal stays at 1.
Output signal (Off-delay timer)
Q 4.0 S_OFFDT
t
The output signal changes to 1 when the input signal changes to 1 or while the timer is running. The time is started when the input signal changes from 1 to 0.

Figure 6. Timer types

A sample use of a retentive on delay timer coil is illustrated in Figure 7 where timer *T3* value is set to 5seconds. This timer starts when input *I0.0* becomes high, i.e., if *I0.0* is a push-button switch, pressing this switch will turn on timer *T3*. This timer *T3* value will be high after 5 seconds.



Address	Declaration	Name	Type	Start value
0.0	temp	OB1_EV_CLASS	BYTE	
1.0	temp	OB1_SCAN_1	BYTE	
2.0	temp	OB1_PRIORITY	BYTE	

```
Network 8: Title:
sensing car

   I0.0                                        T3
---| |-----------------------------------------(SS)---
                                              S5T#5S
```

Figure 7. An example use for a retentive on-delay timer.

TRAFFIC LIGHT CROSSING

Figure 8 shows the traffic light crossing simulator in this experiment where it employs two inputs, named *S0* and *S1* (normally-open momentary type push-button switches), and six outputs (0 to 5). This board must be supplied with 24VDC voltage where ground must be connected to *0V* and 24V must be connected to *DC 24V* terminals. When any input switch is pushed on, the input terminal that the switch is connected becomes 24VDC. Six output terminals control the operation of six Light Emitting Diodes (LEDs). When a particular output terminal is connected to 24VDC, the LED with that output number turns on.

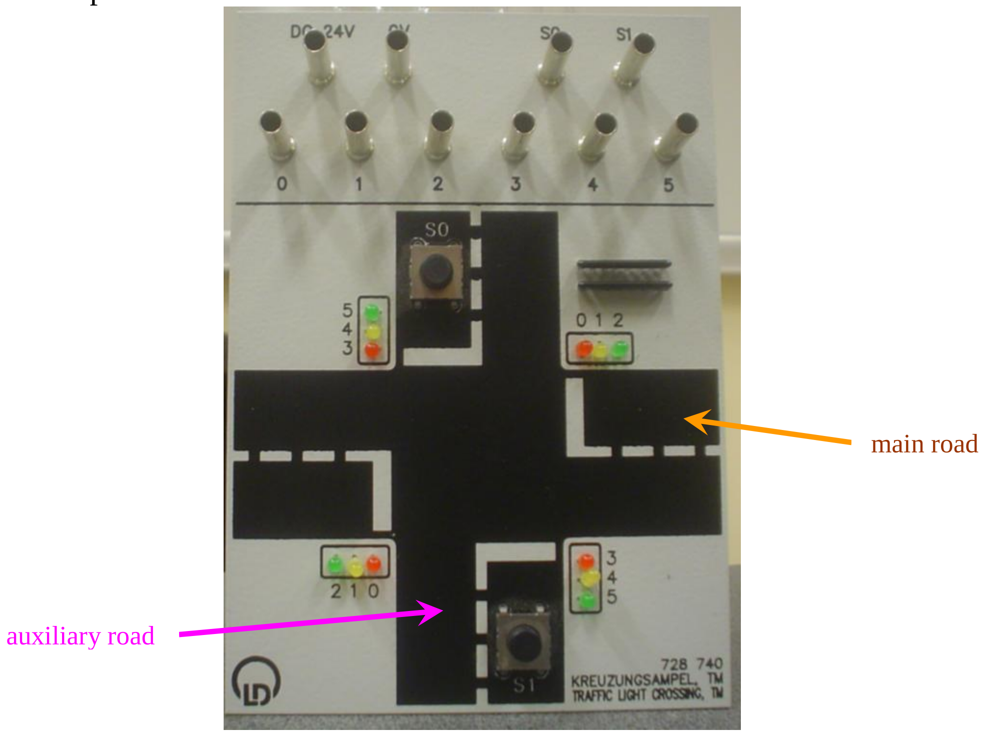


Figure 8. Traffic Light Crossing.

PUSHBUTTON SWITCH

Pushbutton switches are momentary type normally open (NO) or normally closed (NC) type switches where some pressure must be maintained to keep the switch activated. Some switches may employ both NO and NC type contacts as illustrated in Figure 9 where contacts numbered ***1*** and ***2*** are for NC and those marked ***3*** and ***4*** are for NO terminals. These switches are used to send signals into the PLC as input commands. For example, a green pushbutton can be used to start a motor while a red one is used for stopping a motor.

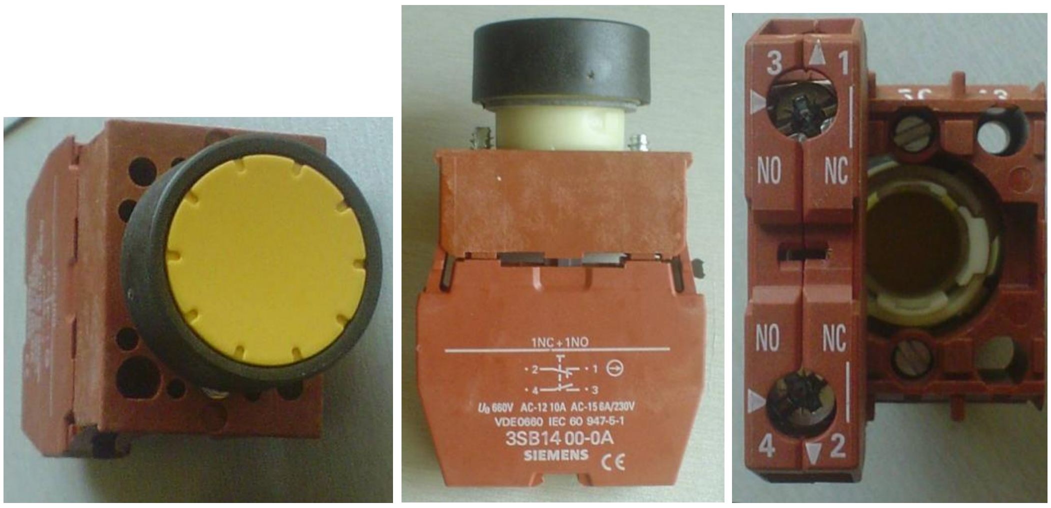


Figure 9. Pushbutton switch, normally-open and normally-closed type (see Appendix A for data sheet)

Procedure of Experiment:

Circuit Set-up: Assemble the circuit shown in Figure 10.

- Please make sure that you follow the same input and output terminal numbers given in Figure 10 when you make connections (i.e., S_1 must be connected to *I0.1* input terminal of PLC).
- Test the correctness of your connection by turning on output *Q4.3* using the corresponding switch on PLC. This will light up red LEDs on auxiliary road of traffic light crossing 728 740. You can try to turn on other outputs as you desire.

Open the STEP 7 software to program the PLC:

- Make an online connection between the PLC and the software by running the command from "*File/Connect Online*".
- Make sure that the switch on PLC is set to RUN-P mode.
- Create a new project. Then, in "*Program/OB1*" (double click *OB1*) write your code in Ladder diagram (make sure to select "*View/LAD*").
- Right Click on *OB1* and choose "*Download to CPU*" command.
- Test your program for correct operation.

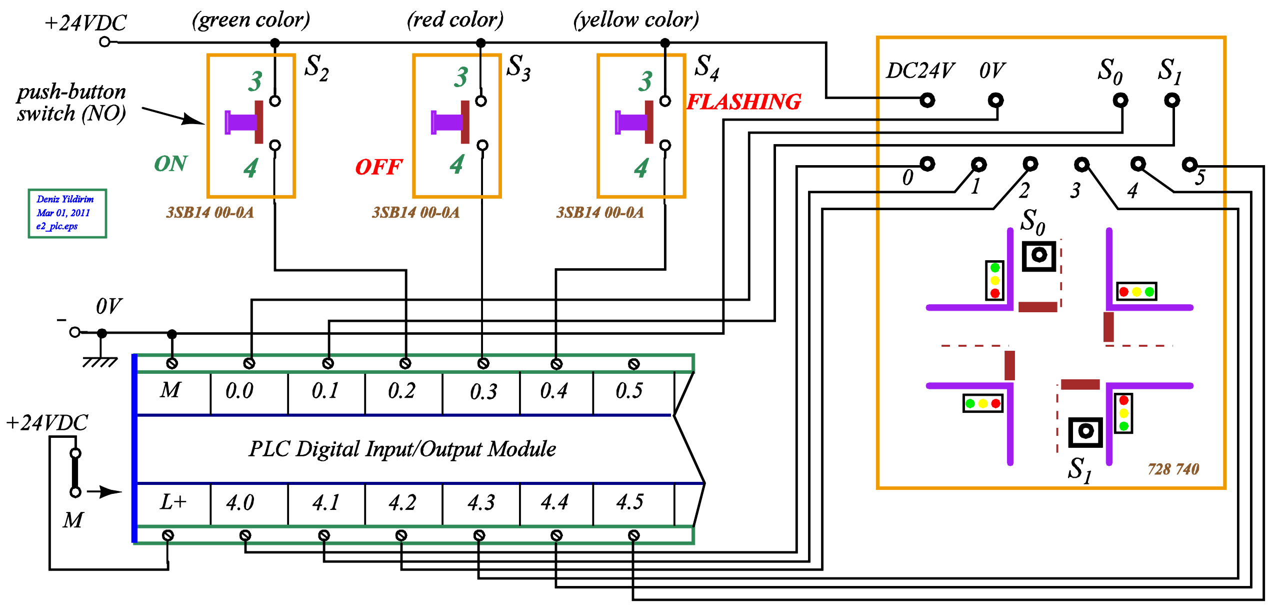


Figure 10. PLC control of traffic lights in an intersection.

Demand-actuated traffic light operation in an intersection:

This simple demand-actuated traffic light control system consists of an intersection where heavy traffic flows on main road and very light traffic is present on auxiliary road. Therefore, the lights on main road are always set to green whereas those on auxiliary road are set to red. When a car approaches on auxiliary road, sensors (either S_0 or S_1) detect that car, and traffic lights on main road turns from green-to-yellow and then yellow-to-red after a predetermined time. Following red light on main road, the lights on auxiliary road turn from red-to-green (no yellow light). After a fixed time delay, the lights on auxiliary road turn from green-to-red and those on main road turn from red-to-green. The timing sequence can be summarized as follows:

1. Initially all traffic lights must be off.

2. The traffic light system will start when either switch S_2 (green button) or S_4 (yellow button) is momentarily pushed on.

3. Pushing *ON* switch S_2 will turn on green lights for main road and red lights for auxiliary road.

4. Pushing *FLASHING* switch S_4 will activate flashing operation where yellow lights on main road and red lights on auxiliary road are flashing at a rate of 1s with 0.5s on followed by 0.5s off. Please note that flashing light on each road must be flashing alternately – not at the same time, i.e., when yellow lights on main street are on for 0.5s red lights on auxiliary street must be off and when yellow lights turn off for 0.5s red lights must turn on for 0.5s. Flashing operation is usually activated during very low traffic hours such as after midnight until dawn.

5. At any time, pushing *OFF* switch S_3 will turn off all lights.

6. When traffic light system is operating with *ON* mode, i.e., green lights on main road are on while red lights on auxiliary road are red, pushing either S_0 or S_1 will initiate traffic light change after a 10s delay.

7. After pressing either S_0 or S_1 the system will wait for 10s, then green lights on main road will change to yellow lights (duration of yellow lights is 3s) and to red lights. At this instant there must be an overlap time where all lights are set to red for time duration of 2s (this is required for safety reasons).

8. After this 2s overlap delay red lights on auxiliary road are turned to green (no yellow light).

9. The yellow lights on auxiliary road will stay on for 15s to allow vehicles cross into main road.

10. After 15s, the green lights on auxiliary road will turn to red and those on main road will switch from red to green.

11. Please note that under no circumstances, all green lights are turned on at the same time as this definitely results in a vehicle collision.

Conclusion:

Figure 11 shows the construction of a device which fills 3 different bins with individually prescribed mixture of balls having 3 different colors which are given in mixed form in a reservoir. A color sensor is used to recognize the color of the ball. If there is a red ball in front of the sensor, then “ R “ output of the sensor becomes 1. “G” output becomes 1, if the ball is green and “B” becomes 1, when the ball is blue. Four valves are used to run the process properly. Directions and input signals of the valves are shown in Figure 6.

a) Describe the operation of the system.

b) Write a PLC program in ladder diagram form to run this device properly:

- Determine the I/O addresses of the PLC which you use for this device.
- Determine the number of networks and function of each network.
- Logic gates, flip-flops, delay timers, counters, comparators, etc. can be used in your design.

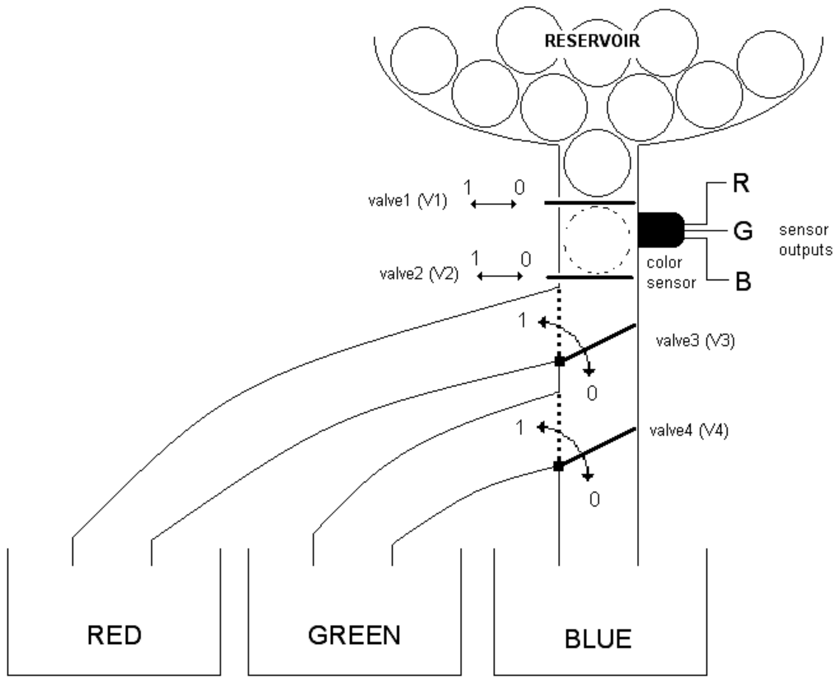


Figure 11. Ball sorting process.

References:

[1] A. Kilian, *Modern Control Technology: Components and Systems*, 2nd ed., Delmar Thomson Learning, 2007.

[2] T. E. Kissell, "*Industrial Electronics: Applications for Programmable Controllers, Instrumentation and Process Control and Electrical Machines and Motor Controls*", Prentice Hall, 3rd edition, 2002.

[3] T. J. Maloney, *Modern Industrial Electronics*, 5th Ed., Prentice Hall, 2001.

[4] T. Bartelt, *Industrial Control Electronics*, 3rd ed., Thomson Delmar Learning, 2006.

EXPERIMENT RESULT SHEET

This form must be filled in using a PEN. Use of <u>PENCIL IS NOT ALLOWED</u>

<u>EXPERIMENT 4</u>: TRAFFIC LIGHT CONTROL SYSTEM FOR AN INTERSECTION USING S7-300 PLC

	STUDENT NO	STUDENT NAME	SIGNATURE	**DATE**
1				
2				**INSTRUCTOR APPROVAL**
3				
4				

Please include flowchart of your program in here indicating inputs and outputs (*ladder diagram code must be included in your report*).