Solution to Homework #3: ECE 461 / 661

Analog Inputs, Flow Control - Due Monday, September 10th

You may work in groups of 1-3 if you like.

Homework can be turned in in class, in my office, or emailed to jacob_glower@yahoo.com

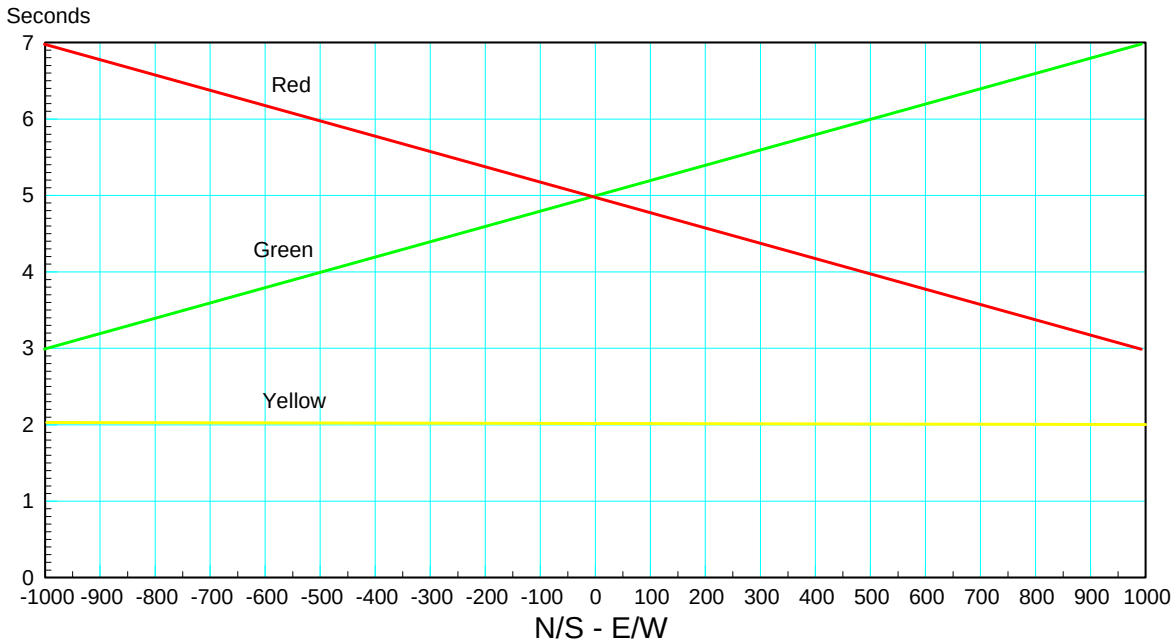
Problem 1: Write a ladder logic program for a traffic light where the Red / Green times depend upon traffic. Assume sensors detect the traffic each direction (0 = no traffic, 1000 = 10V = heavy traffic)

- Analog Input 03 = E/W traffic (0 to 1000)
- Analog Input 04 = N/S traffic (0 to 1000)

The green times should be related to the traffic:

- if NS - EW = 1000,
  - Green = 7 seconds
  - Yellow = 2 seconds (fixed)
  - Red = 3 seconds
- If (NS - EW) = 0
  - Green = 5 seconds
  - Yellow = 2 seconds
  - Red = 5 seconds
- If (NS - EW) = -1000,
  - Green = 3 seconds
  - Yellow = 2 seconds
  - Red = 7 seconds

Problem 2: Demonstate your program (in person or with a video)



Solution:

Rung 1..6: Convert the analog inputs into the green time and red times.

Rung 1: Analog inputs are unsigned integers. Convert these to DINT variables so that negative numbers are permitted.

Rung 2 - 3: Implement the computations:

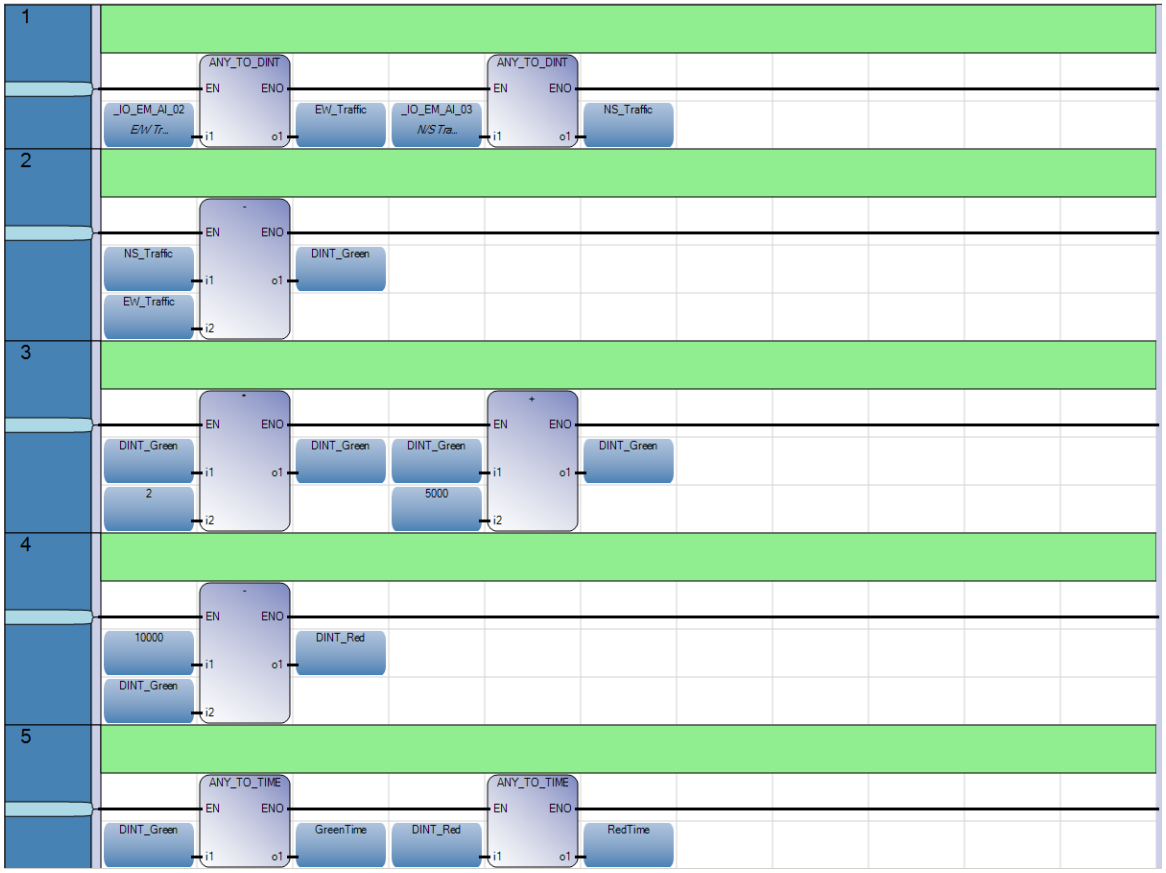
$$Green = 2 \cdot (N/S - E/W) + 5000$$

so that the green time goes from 3000 (NS - EW = -1000) to 7000 (NS - EW = +1000)

Rung 4: Compute the red time

$$Red = 10000 - Green$$

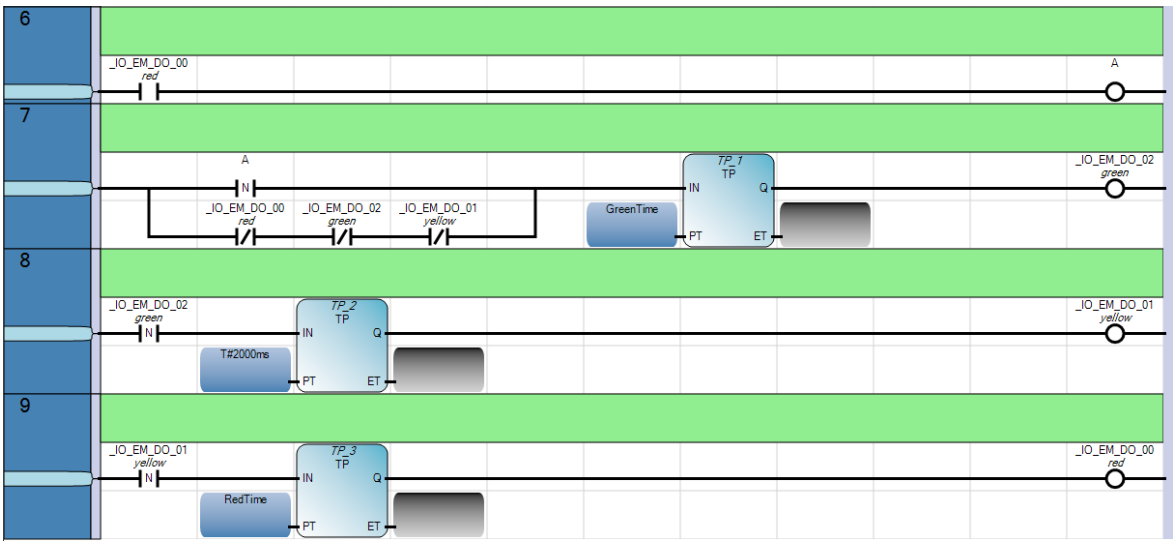
Rung 5: Convert the green and red times to type TIME.



Rung 6-9: Implement the stoplight with

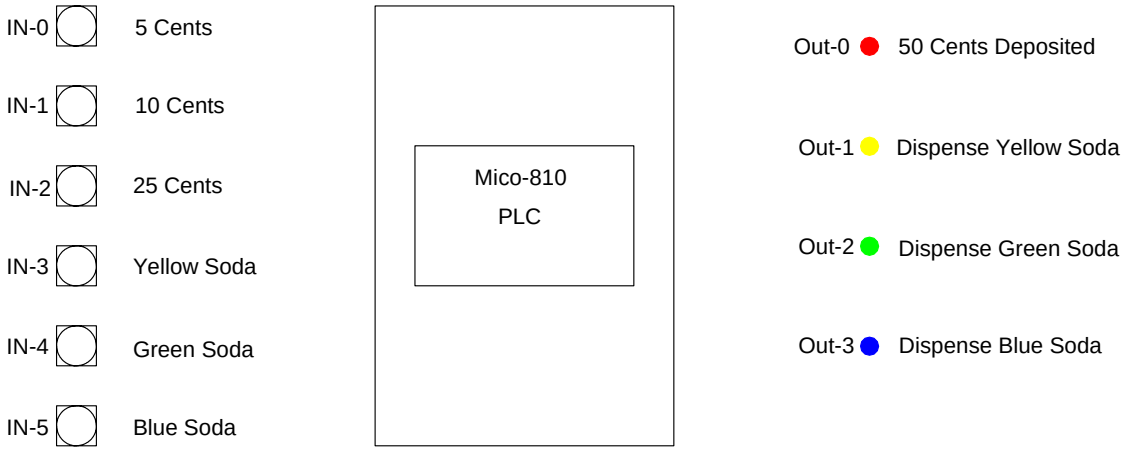
- GreenTime setting the green time (3000ms to 7000ms)
- YellowTime = 2000ms
- RedTime going from 7000ms to 3000ms

Repeat over and over


6
_IO_EM_DO_00
red
A
7
A
N
_IO_EM_DO_00
red
_IO_EM_DO_02
green
_IO_EM_DO_01
yellow
TP_1
TP
IN
Q
GreenTime
PT
ET
_IO_EM_DO_02
green
8
_IO_EM_DO_02
green
N
TP_2
TP
IN
Q
T#2000ms
PT
ET
_IO_EM_DO_01
yellow
9
_IO_EM_DO_01
yellow
N
TP_3
TP
IN
Q
RedTime
PT
ET
_IO_EM_DO_00
red

A PLC is to control a pop machine. The coin slot can accept either nickles, dimes, or quarters.

- When you add 50 cents or more, the red light turns on indicating that you have enough money to buy a pop.
- If there is more than 50 cents in the machine when you press a select button (yellow, green, blue), then
  - The corresponding light turns on for 2 seconds then turns off, indicating that a pop has been dispensed, and
  - If you added more than 50 cents, change is given. The red light is turned off for 2 seconds.
  - If change is due, the red light turns back on for 2 seconds indicating that a nickle was returned
  - Change continues to be given until you have no change due.

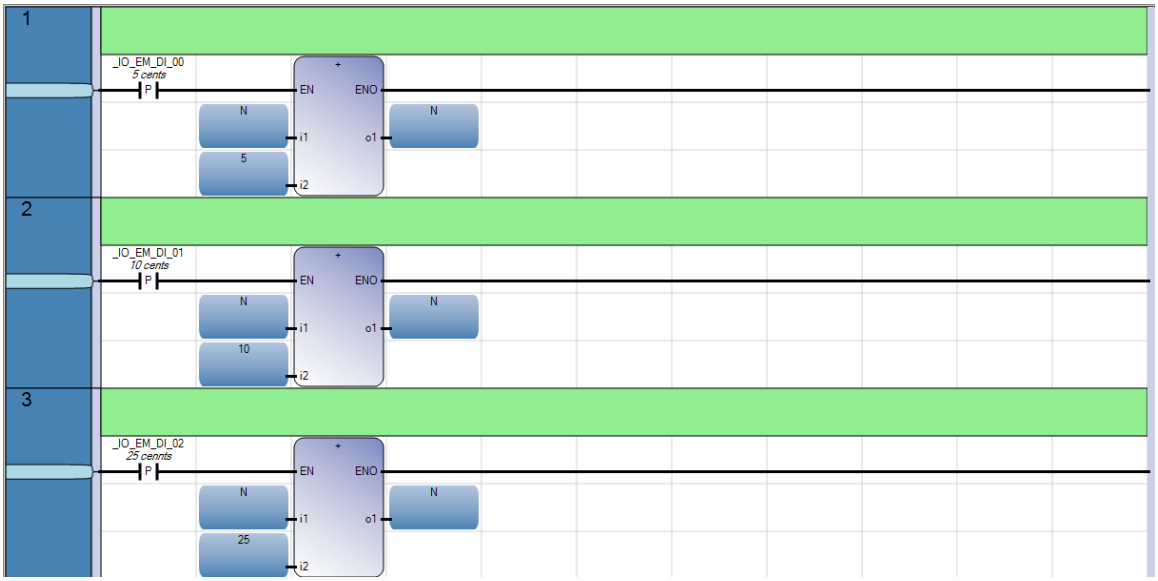


Problem 3) Wite a ladder-logic program to implement the soda pop controller that gives change.

Problem 4) Demonstrate that your program works (video or in-person demo is OK)

Soda Pop: Ladder Logic Solution

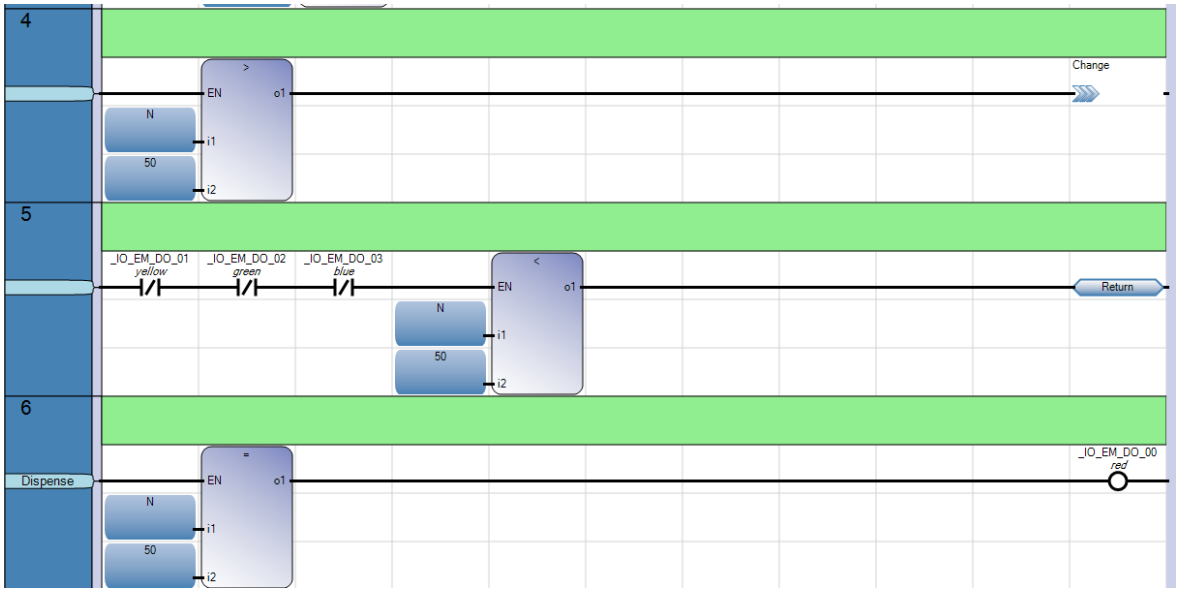
Rung 1-3: Constantly monitor the buttons and add to the money input into the soda pop machine (N) each pass.



Rung 4: If you input more than 50 cents, jump to the Change routine to return a nickel at a time until you get to 50 cents.

Rung 5: If you have less than 50 cents and you're not dispensing a pop at present, exit

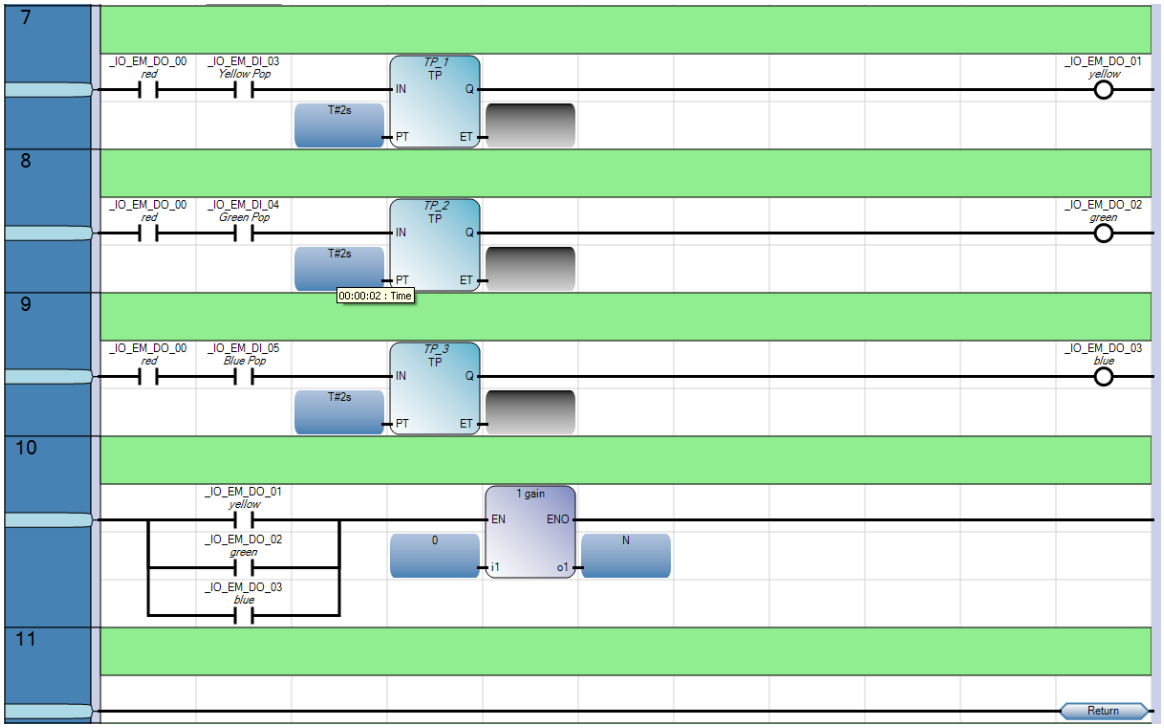
Rung 6: If there's 50 cents in the machine, turn the red light on, indicating that you're ready to dispense a pop.



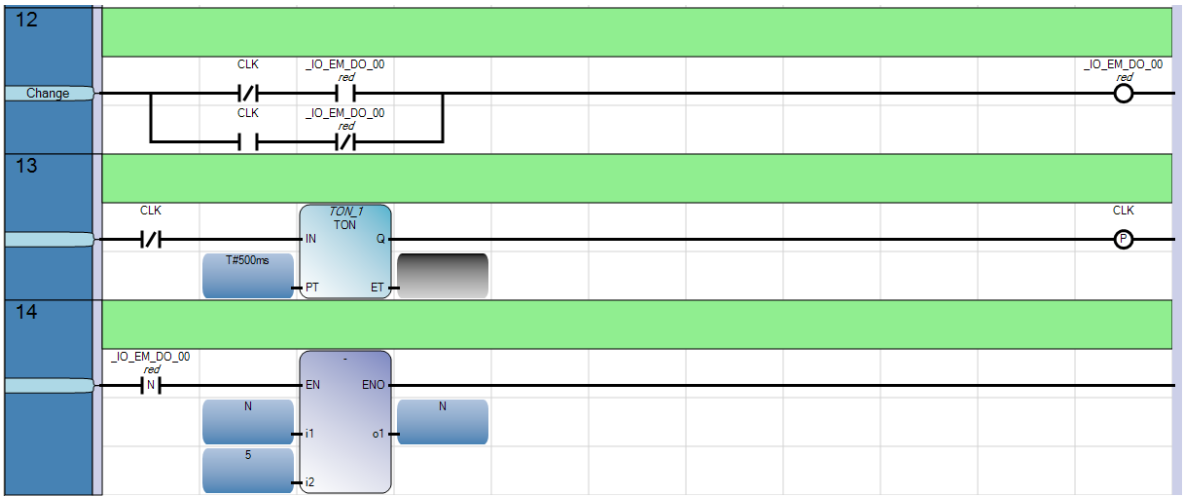
Rung 7-11: Dispense a Pop routine.

If the red light is on (50 cents is in the machine), monitor the Pop buttons. If one is pressed, turn on the corresponding light for 2 seconds.

Rung 10: If any button is pressed, immediately clear N, preventing you from getting another pop for free (this turns off the red light in the next pass in rung #6)



Rung 12-15: Change routine. Toggle the red light every 500ms. On the falling edge of the red light, decrement the coins by 5 cents.



Soda Pop: Structured Text Solution

```
;  Pascal doesn't have a rising edge input, so create it.  If the previous
;  input was false and the present input is true, you see a rising edge.
;  On the rising edges, increment how much money you have

IF( NOT(Old_DI00) and _IO_EM_DI_00) THEN
   Money := Money + 5;
   END_IF;

IF( NOT(Old_DI01) and _IO_EM_DI_01) THEN
   Money := Money + 10;
   END_IF;

IF( NOT(Old_DI02) and _IO_EM_DI_02) THEN
   Money := Money + 25;
   END_IF;

Old_DI00 := _IO_EM_DI_00;
Old_DI01 := _IO_EM_DI_01;
Old_DI02 := _IO_EM_DI_02;


; Change Routine.  If you have added more than 50 cents, toggle the red
; LED every 500ms.  On the falling edge, also decrement the amound of
; money by 5 cents

IF(Money > 50) THEN
   TON_1(not(TON_1.Q), T#500ms);

   IF(TON_1.Q) THEN
      _IO_EM_DO_00 := not(_IO_EM_DO_00);
      IF(_IO_EM_DO_00) THEN
         Money := Money - 5;
         END_IF;
      END_IF;
   END_IF;


; Dispense Pop Routine.  If you have 50 cents, then turn on the red light.
; Check the buttons to see which type of pop you want
;   Pop = 0:   no button was pressed
;   Pop = 1:   yellow pop
;   Pop = 2:   green pop
;   Pop = 3:   blue pop

IF(Money = 50) THEN
   _IO_EM_DO_00 := TRUE;
   Pop := 0;
   IF(_IO_EM_DI_03) THEN
      Pop := 1;
      Money := 0;
      END_IF;
   IF(_IO_EM_DI_04) THEN
      Pop := 2;
      Money := 0;
      END_IF;
   IF(_IO_EM_DI_05) THEN
      Pop := 3;
      Money := 0;
      END_IF;
   END_IF;
```

```
; Clear the red LED when you have less than 50 cents.
; This prevents you from getting 2 pops

   IF(Money < 50) THEN
      _IO_EM_DO_00 := FALSE;
      END_IF;

; Dispense Pop routine.  (This is always executed, which keeps the
; timers working).  If you select a pop, turn on the corresponding light
; for 2 seconds.

TP_1(Pop = 1, T#2s);
TP_2(Pop = 2, T#2s);
TP_3(Pop = 3, T#2s);

_IO_EM_DO_01 := TP_1.Q;
_IO_EM_DO_02 := TP_2.Q;
_IO_EM_DO_03 := TP_3.Q;



;  end of Soda Pop routine
```