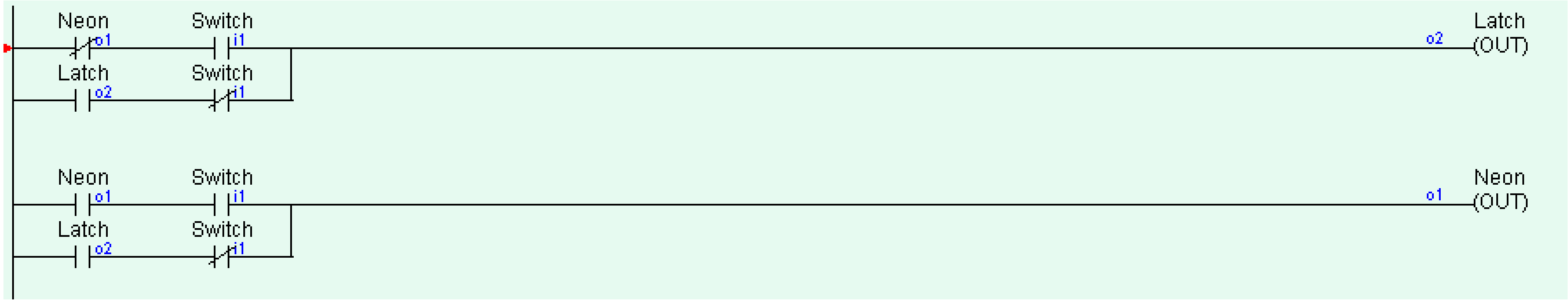
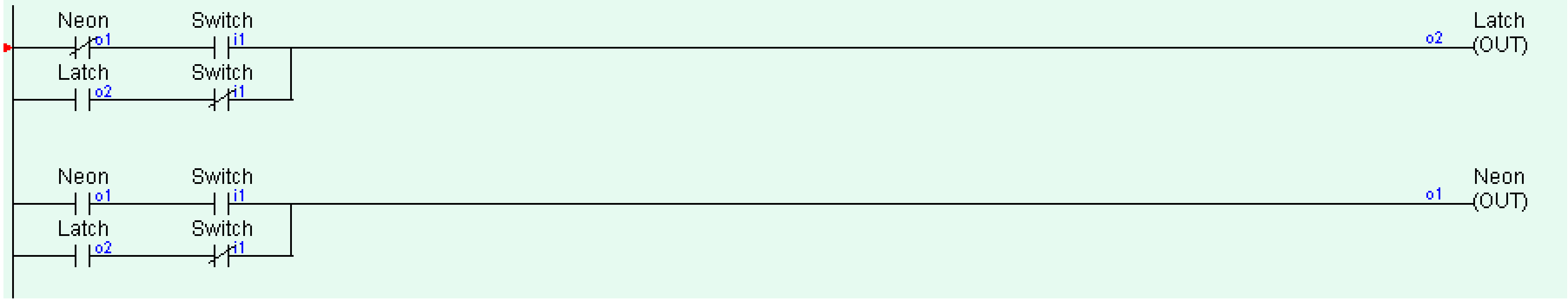
PLC

- A bit more convoluted: toggling Hello World with a single button.



- Remember - Switch state: ON OFF ON OFF
- Figure it out



input:

S	1	0		1	0
Sinv	0	1		0	1
L	1	1		0	0
N	0	1		1	0
Ninv	1	0		0	1

Latch = (Switch AND notNeon) OR (Latch AND notSwitch)

Neon = (Latch AND notSwitch) OR (Neon AND Switch)

The logic

SW	1	0	1	0
SWInv	0	1	0	1
Neon	0	1	1	0
NeonInv	1	0	0	1
Latch	1	1	0	0
A	1	0	0	0
B	0	1	0	0
C	0	0	1	0

A = SW and NeonInv
B = SWInv and Latch
C = SW and Neon
Latch = A or B
Neon = B or C

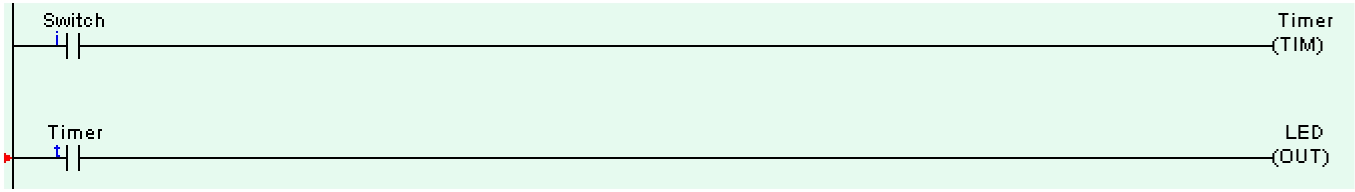
PLC

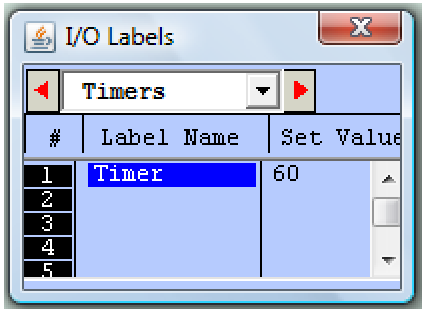
- PLCs also support a range of special functions: timers, counters, sequencers, memory instructions, etc…

- Beyond the basics, they are non-standard and manufacturer-specific.

Timers and Counters

- Timer: When enabled it counts down, then enables a rung when it reaches zero

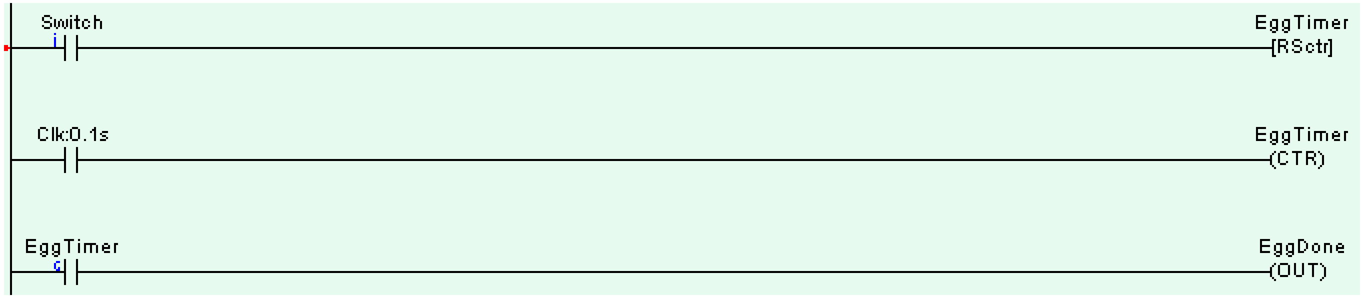


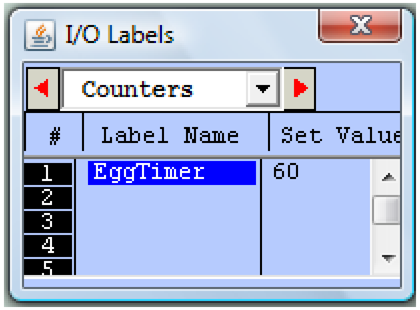


Timers

#	Label Name	Set Value
1	Timer	60
2		
3		
4		
5		

- Counter: counts down when pulsed, then enables a rung when it reaches zero





Counters

#	Label Name	Set Value
1	EggTimer	60
2		
3		
4		
5		

Exercises

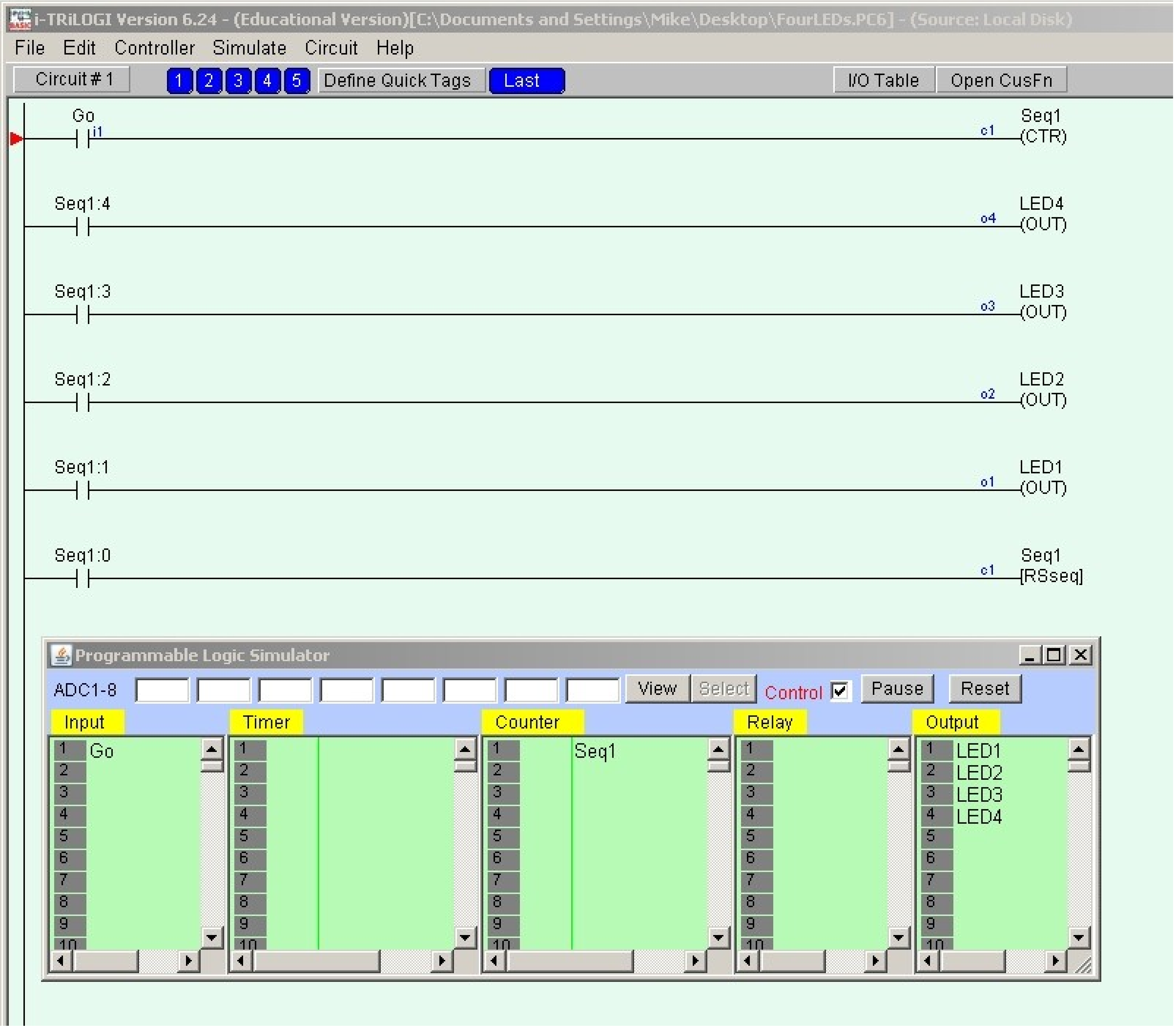
- Turn 8 LEDs on and off in sequence, repeat.

- 1 minute Egg Timer.

- Turn a pump on for 60 seconds, then off for 40 seconds, then repeat. Use a switch to start it off.

LEDs: Using a sequencer

- Define a Counter output – call it Seq1
- Give it a set value 1 greater than your desired sequences:
  - e.g. if 4, then 0 thru 4 = a set value of 5
- Use special bits as inputs
- Use Seq1:0 to reset the sequencer by defining an output FUNC


i-TRiLOGI Version 6.24 - (Educational Version)[C:\Documents and Settings\Mike\Desktop\FourLEDs.PC6] - (Source: Local Disk)
File
Edit
Controller
Simulate
Circuit
Help
Circuit # 1
1
2
3
4
5
Define Quick Tags
Last
I/O Table
Open CusFn
Go
i1
c1
Seq1
(CTR)
Seq1:4
o4
LED4
(OUT)
Seq1:3
o3
LED3
(OUT)
Seq1:2
o2
LED2
(OUT)
Seq1:1
o1
LED1
(OUT)
Seq1:0
c1
Seq1
[RSseq]
Programmable Logic Simulator
ADC1-8
View
Select
Control ☑
Pause
Reset
Input
Timer
Counter
Relay
Output
1 Go
1 Seq1
1 LED1
2 LED2
3 LED3
4 LED4
1
2
3
4
5
6
7
8
9
10

Egg Timer

IO Table -

- Inputs: 1/s Clock
- Outputs: EggDone
- Counters: Count down from 60

Steps:

- 1/sec used to trigger counter 60 times
- When counter at 0, energize EggDone

Problem: Use a switch to reset counter

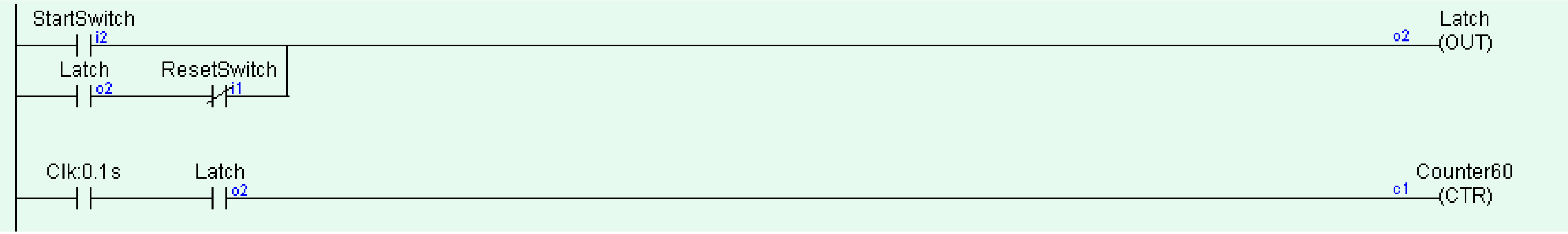
Problem: Add a Start Switch

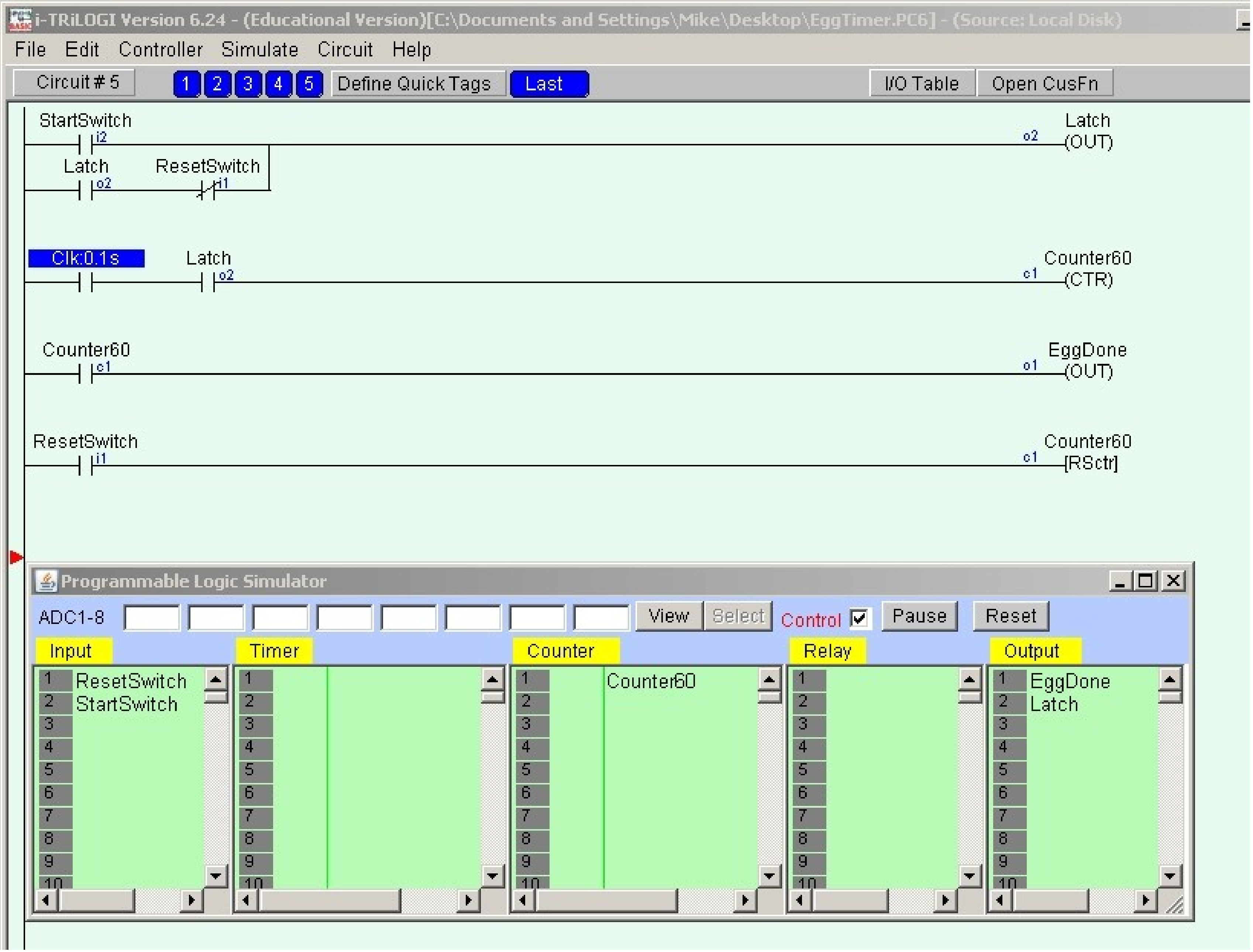
Add a start switch

Use a Latch

Enable counter with Latch

Add ~Reset to Latch Circuit





i-TRiLOGI Version 6.24 - (Educational Version)[C:\Documents and Settings\Mike\Desktop\EggTimer.PC6] - (Source: Local Disk)

Input	
1	ResetSwitch
2	StartSwitch
3	
4	
5	
6	
7	
8	
9	
10	

Timer		
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		

Counter		
1		Counter60
2		
3		
4		
5		
6		
7		
8		
9		
10		

Relay	
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	

Output	
1	EggDone
2	Latch
3	
4	
5	
6	
7	
8	
9	
10	

60 sec ON/ 40 sec OFF

Inputs: None

Outputs: None

Timers: 60 sec, 40 sec.

Steps -

Count down the 60 sec Timer ONLY WHEN the 40 sec Timer is OFF

Count down the 40 sec Timer ONLY WHEN the 60 sec Timer is ON

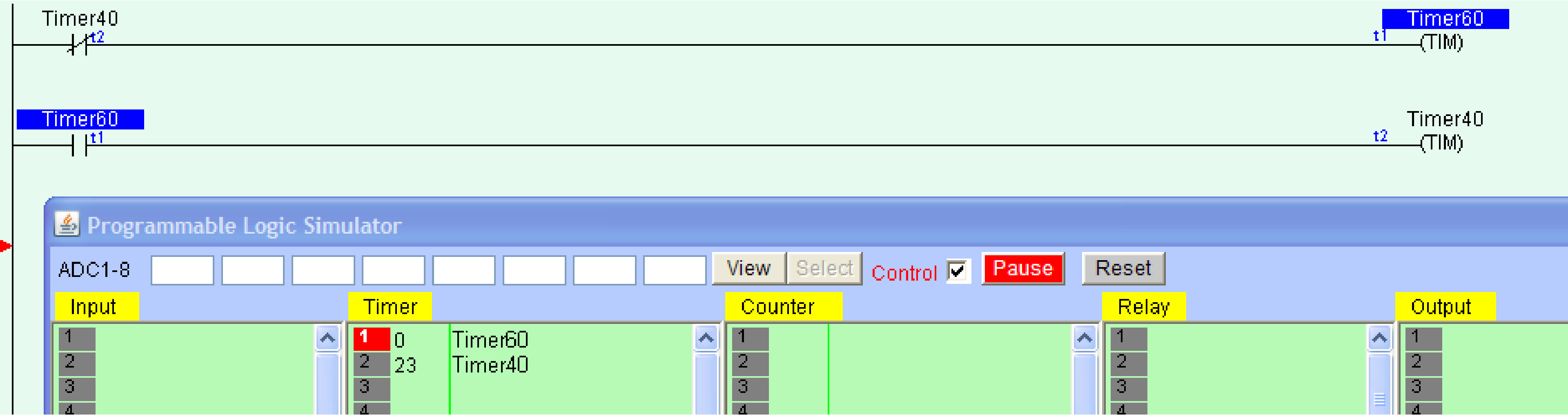
- A Timer is ON when it is at zero, not when it is counting.
- A Timer counts when it is being energized by its circuit.
- **A Timer is RESET when its circuit is shut off. This is important**

State1: at startup, Timer40 is OFF so Timer60 starts counting.

State2: Timer60 is counting, so Timer60 is OFF, so Timer 40 is not counting

State3: Timer60 is zero, therefore Timer40 is counting, therefore -Timer40 keeps Timer60 from counting

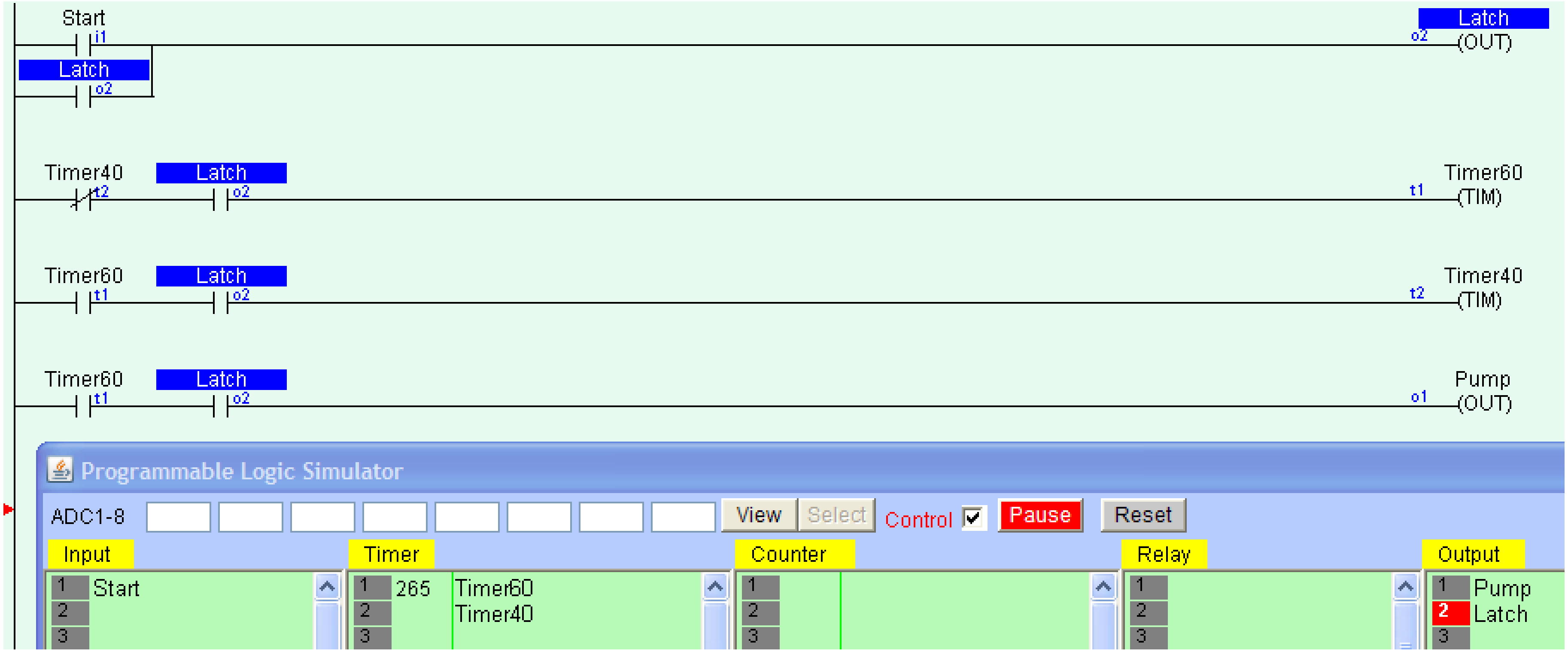
State4: When Timer40 reaches zero, Timer40 is ON, -Timer40 is OFF, Timer60 gets reset, therefore OFF, then Timer 40 is reset, therefore OFF, and Timer60 starts counting again.



1	0	Timer60
2	23	Timer40
3		
4		

Extend the problem

- Add a pump, on for 40, off for 60
  - The pump should be on when Timer40 is counting
  - Timer40 counts when Timer60 is at zero
  - Add the pump output with Timer60 Input
- Add a Start Switch
  - Just put a Latch everywhere



Programmable Logic Simulator

ADC1-8

Control ☑

Input
1 Start
2
3

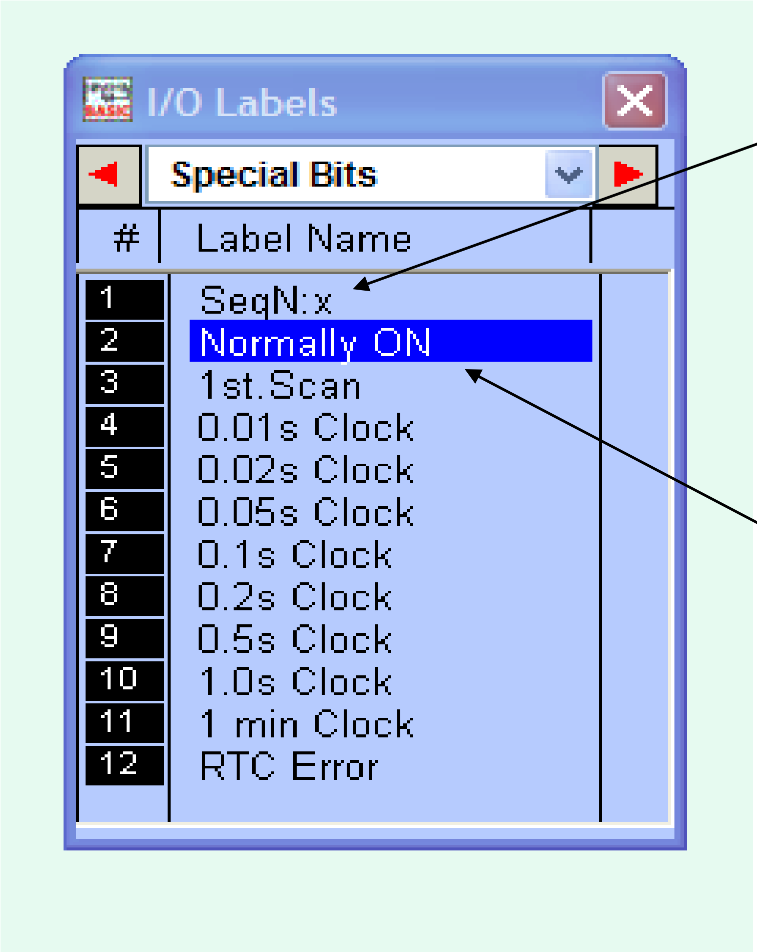
Timer
1 265 Timer60
2 Timer40
3

Counter
1
2
3

Relay
1
2
3

Output
1 Pump
2 Latch
3

Special Bits



I/O Labels

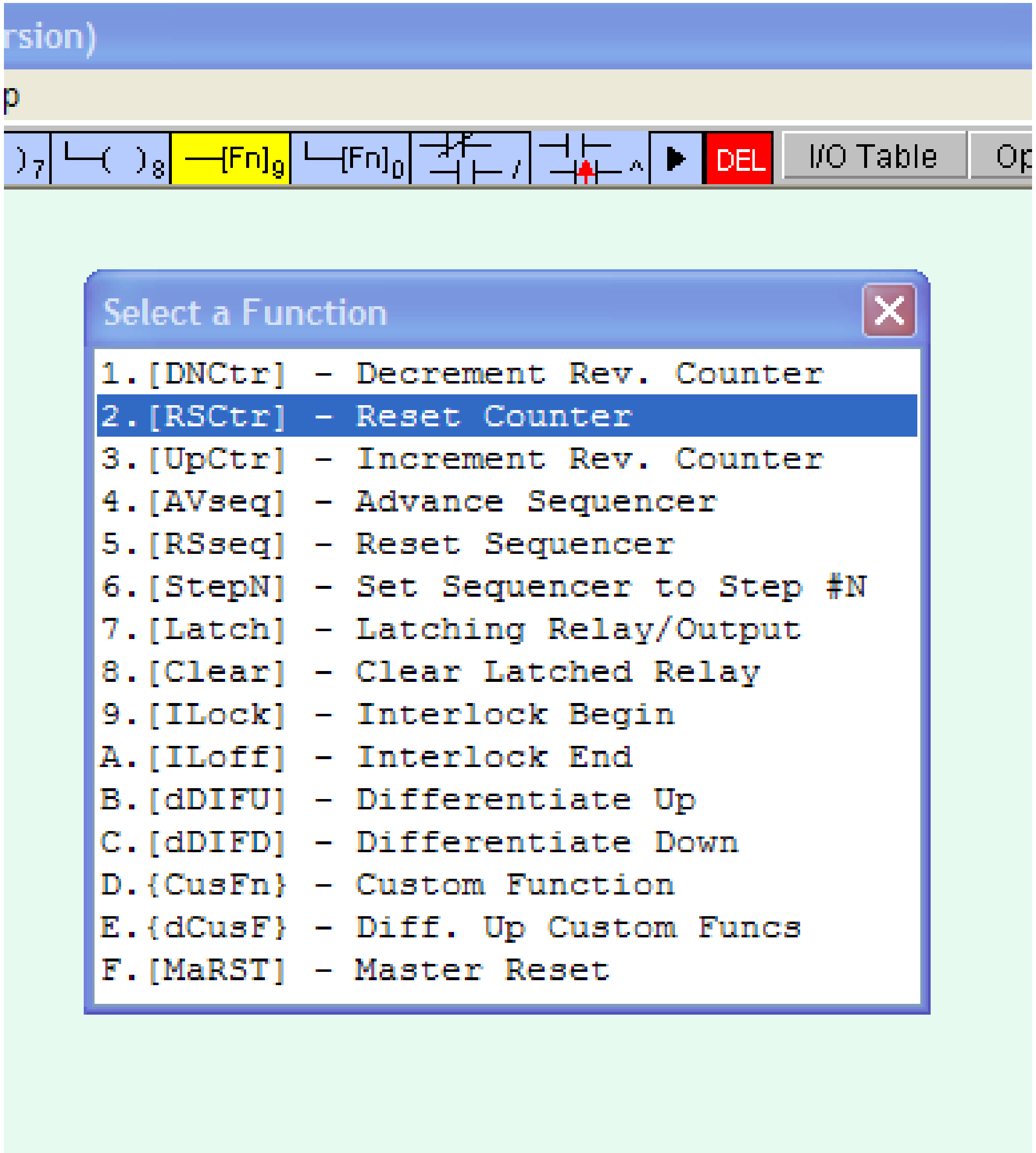
Special Bits

#	Label Name
1	SeqN:x
2	Normally ON
3	1st.Scan
4	0.01s Clock
5	0.02s Clock
6	0.05s Clock
7	0.1s Clock
8	0.2s Clock
9	0.5s Clock
10	1.0s Clock
11	1 min Clock
12	RTC Error

Use with Counters named “SeqN”

Unchangeable SWITCH
(always ON)

Special FUNCTION Inputs



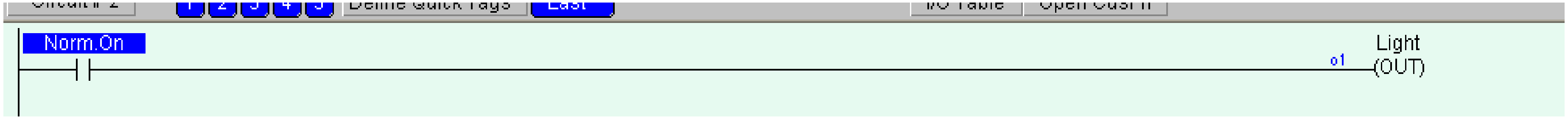
Select a Function

```
1.[DNCtr] - Decrement Rev. Counter
2.[RSCtr] - Reset Counter
3.[UpCtr] - Increment Rev. Counter
4.[AVseq] - Advance Sequencer
5.[RSseq] - Reset Sequencer
6.[StepN] - Set Sequencer to Step #N
7.[Latch] - Latching Relay/Output
8.[Clear] - Clear Latched Relay
9.[ILock] - Interlock Begin
A.[ILoff] - Interlock End
B.[dDIFU] - Differentiate Up
C.[dDIFD] - Differentiate Down
D.{CusFn} - Custom Function
E.{dCusF} - Diff. Up Custom Funcs
F.[MaRST] - Master Reset
```

Special Bits

Normally ON Flag - Norm.ON

- You can make use of this flag if you need to keep something permanently ON regardless of any input conditions. This is because a coil or a special function is not allowed to connect directly to the power line (the vertical line on the left end of the ladder diagram). If you need to permanently enable a coil, consider using the "Normally-ON" bit from the "Special Bits" menu, as follows:


Norm.On
Light
o1
(OUT)