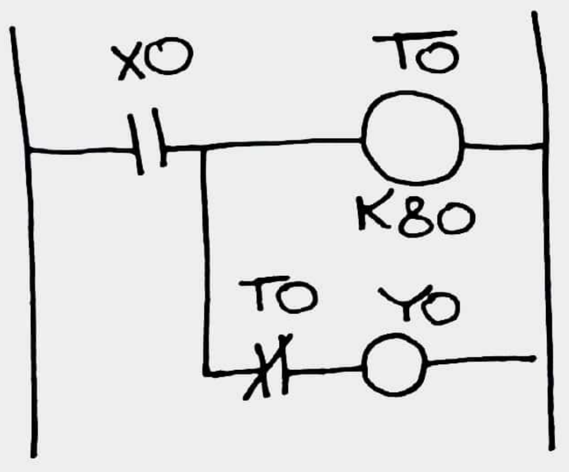
PLC Sheet 1

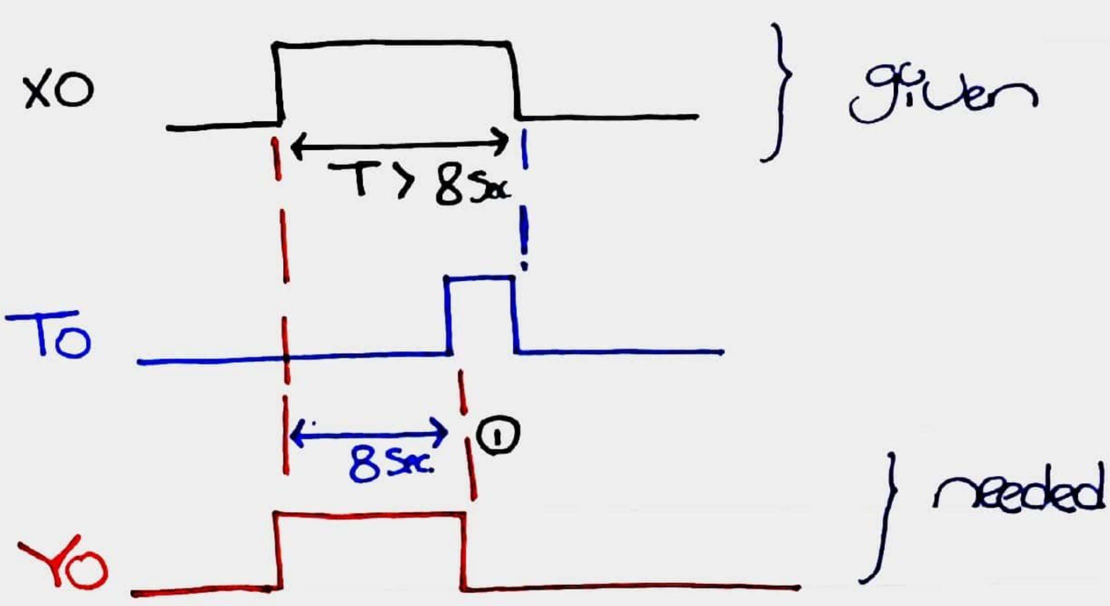
1)



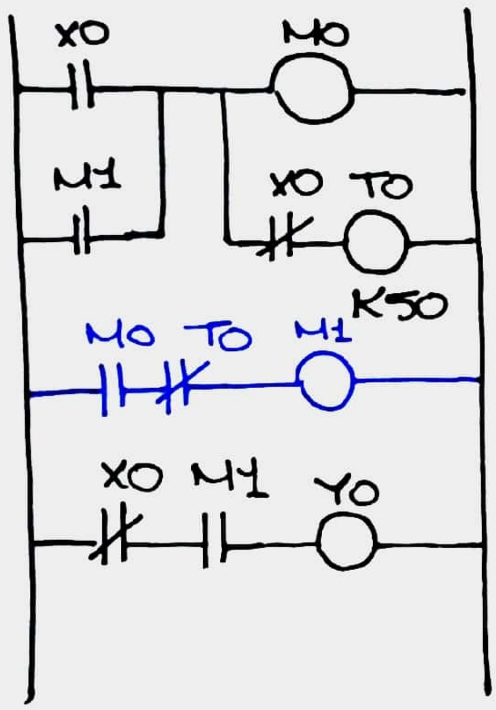
↔

LD X0
OUT T0
K 80
ANI T0
OUT Y0

} Corresponding logic instruction Program



2)



LD X0
OR M1
OUT M0
ANI X0
OUT T0
K50
LD M0
ANI T0
OUT M1
LDI X0
AND M1
OUT Y0

• $M0 \equiv (X0 + M1)$

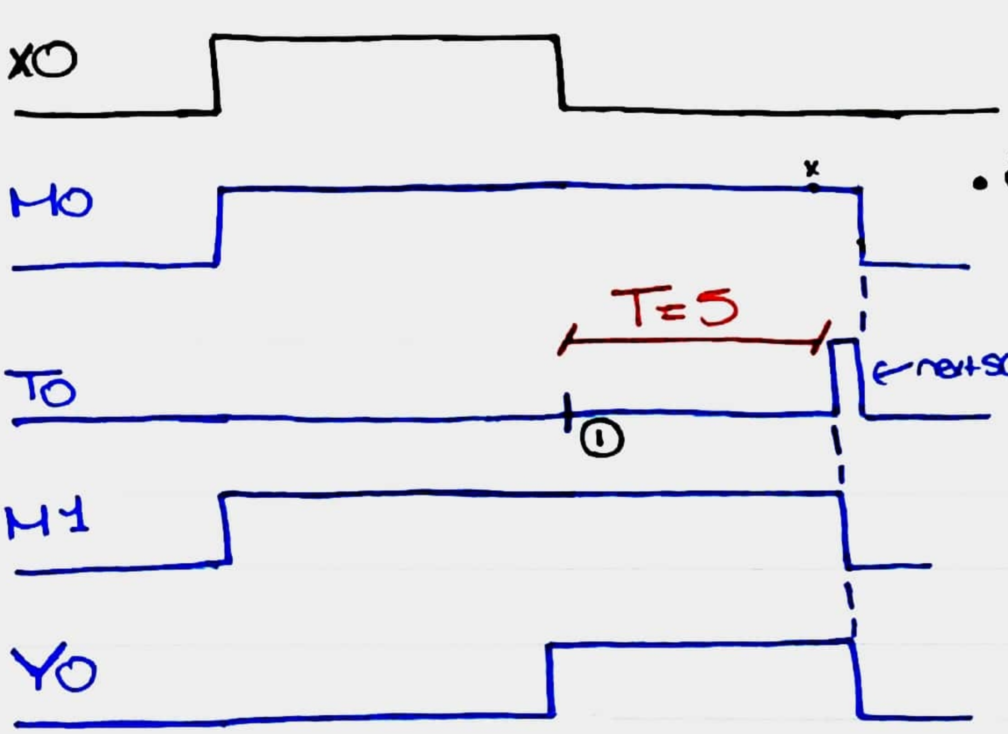
$T1(K50) \equiv (X0 + M1)\overline{X0}$

• $M1 = M0.\overline{T0}$

Trace sequentially

• $Y0 = \overline{X0}.M1$

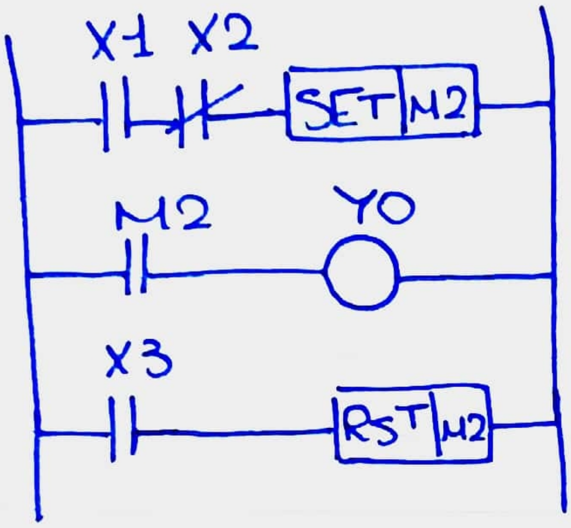
X0	0	1	1	0	0	0
M0	0	1	1	1	1	0
T0	0	0	0	0 $\xrightarrow{3}$ 1	1	0
M1	0	1	1	1	0	0
Y1	0	0	0	1	0	0
				1		



} given

• Should actually drop here but it's okay to miss that (TA) and drop at x.

3)

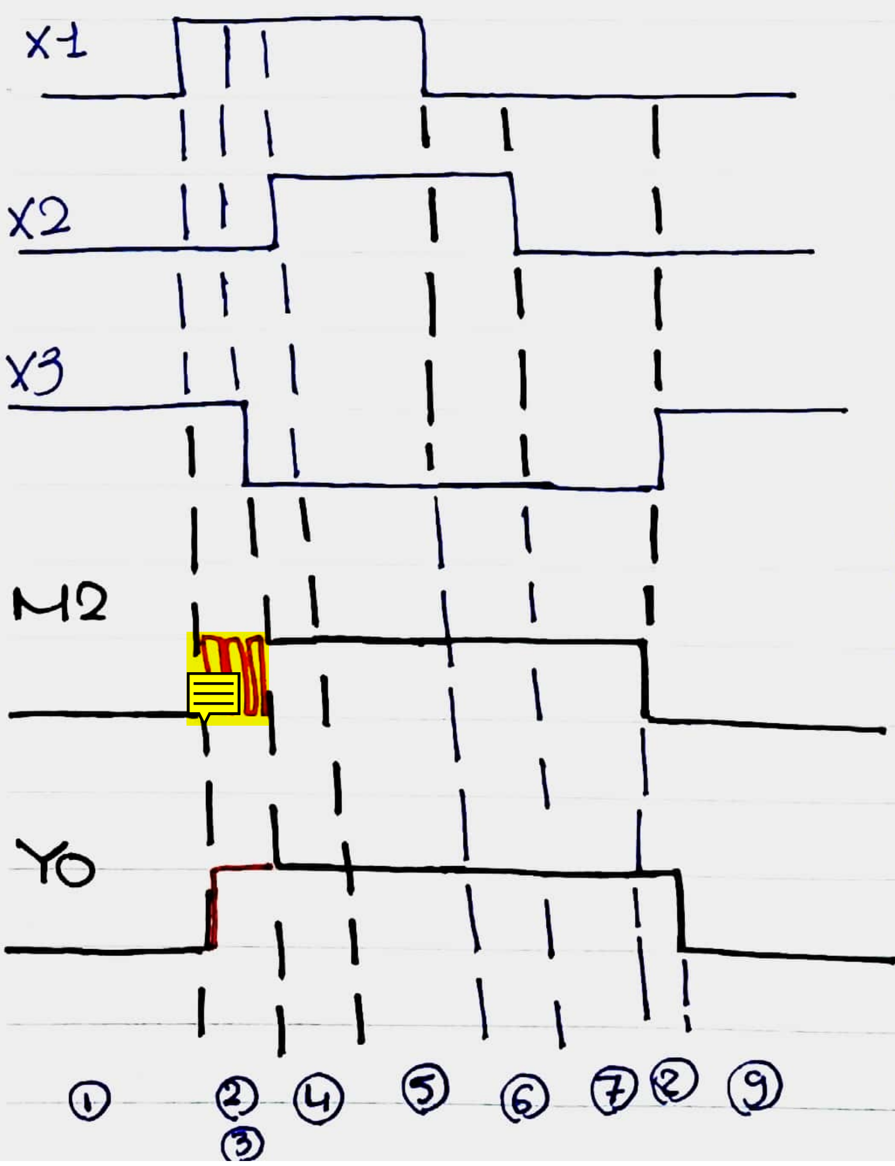


LD X1
ANI X2
SET M2
LD M2
OUT YO
LD X3
RST M2

- $SET(M2) \equiv X1 \cdot \overline{X2}$
- $YO \equiv M2$
- $RST(M2) \equiv X3$

	①	②	③	④	⑤
X1	0	1	1	1	1
X2	0	0	0	0	1
X3	1	1	1	0	0
M2	0	1→0	1→0	1	1 (remain)
YO	0	1	1	1	1

- Now even if X1, X2 change it will only drop when X3 is high and whenever it drops, Y has took value already → reflected next cycle



	⑥	⑦	⑧	⑨
X1	0	0	0	0
X2	1	0	0	0
X3	0	0	1	1
M2	1	1	1→0	0
YO	1	1	1	0

4) Draw Ladder Diagram Such that

- X0 becomes on → Y0 is activated
- From the Moment X0 is off → Y0 is deactivated in 20 sec

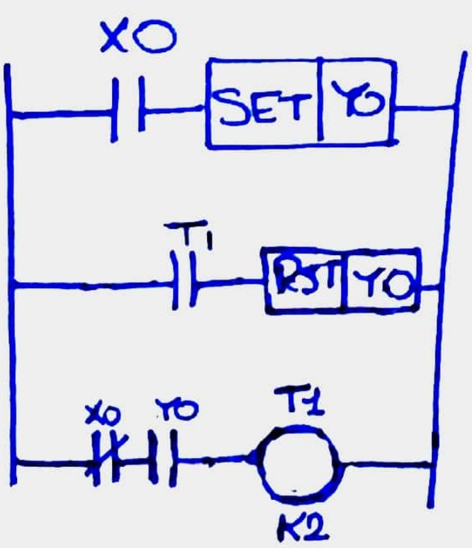
Set(Y0) ≡ X0

RST(Y0) = T1

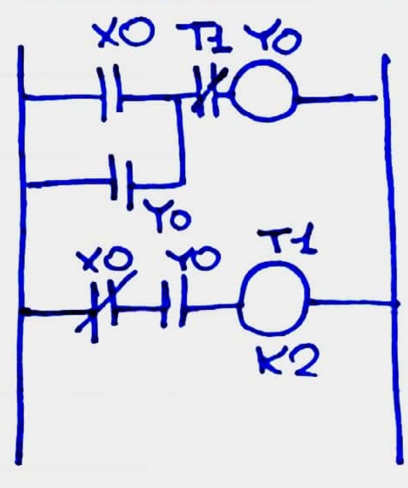
- when X0 becomes low we don't want Y0 ≡ 0 immediately (so Self.latch or Set/reset)

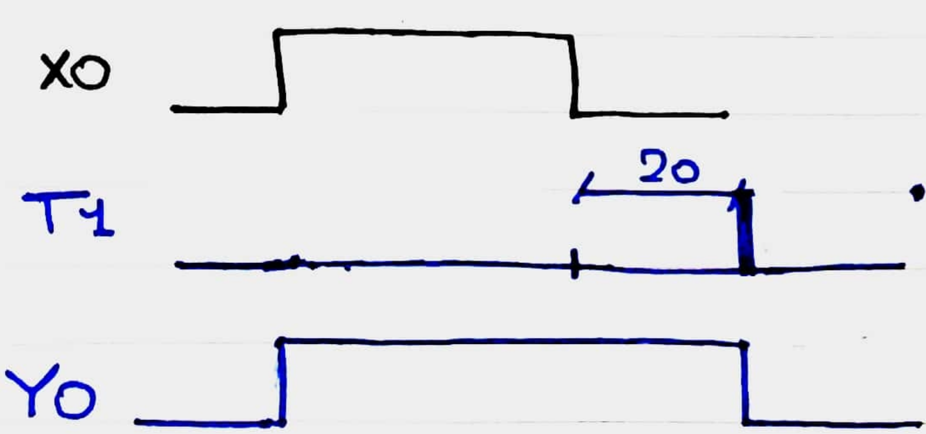
$T1(K2) \equiv \overline{X0} . Y0$

if it's low already then should not time (no automatic turn off)



or equivalently





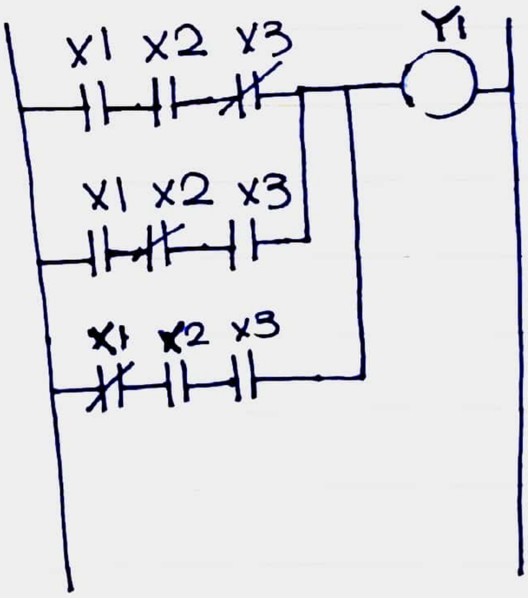
. T1=1 → Y0=0 → T1=0

5) Draw ladder diagram such that

- There are 3 inputs X1, X2, X3
- One output Y1
- Activated when exactly 2 out of the 3 inputs become on and deactivated o.w.

$$Y1 \equiv (X1 . X2 . \overline{X3}) + (X1 . \overline{X2} . X3) + (\overline{X1} . X2 . X3)$$

LD X1
AND X2
ANI X3
LD X1
ANI X2
AND X3
ORB
LDI X1
AND X2
AND X3
ORB
OUT Y1

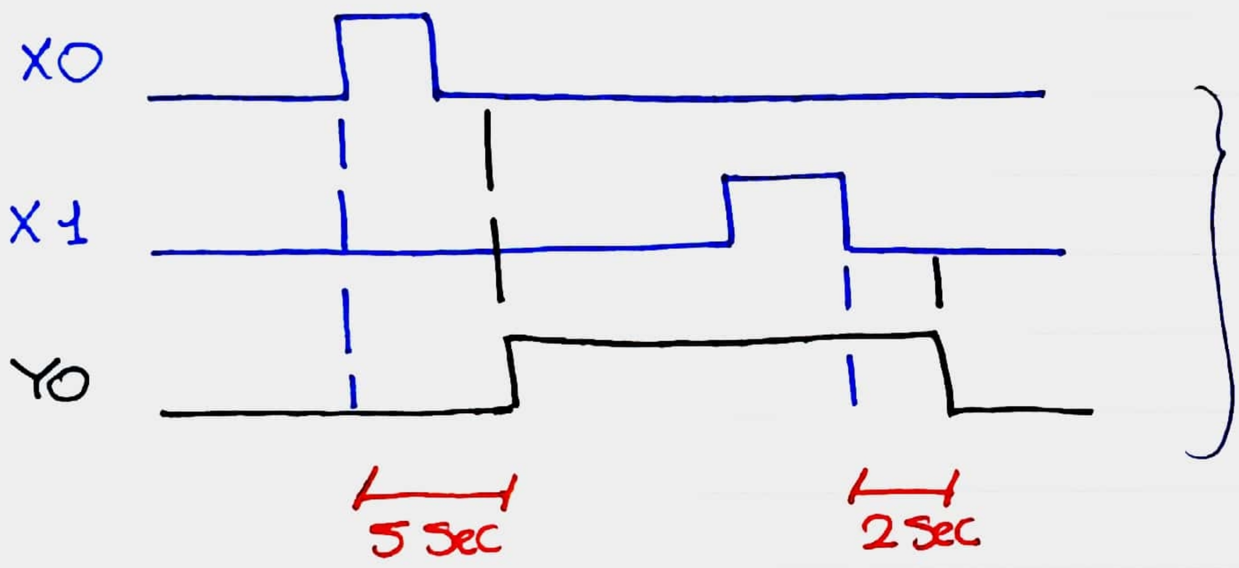


6)

#IMPortant!

Given:

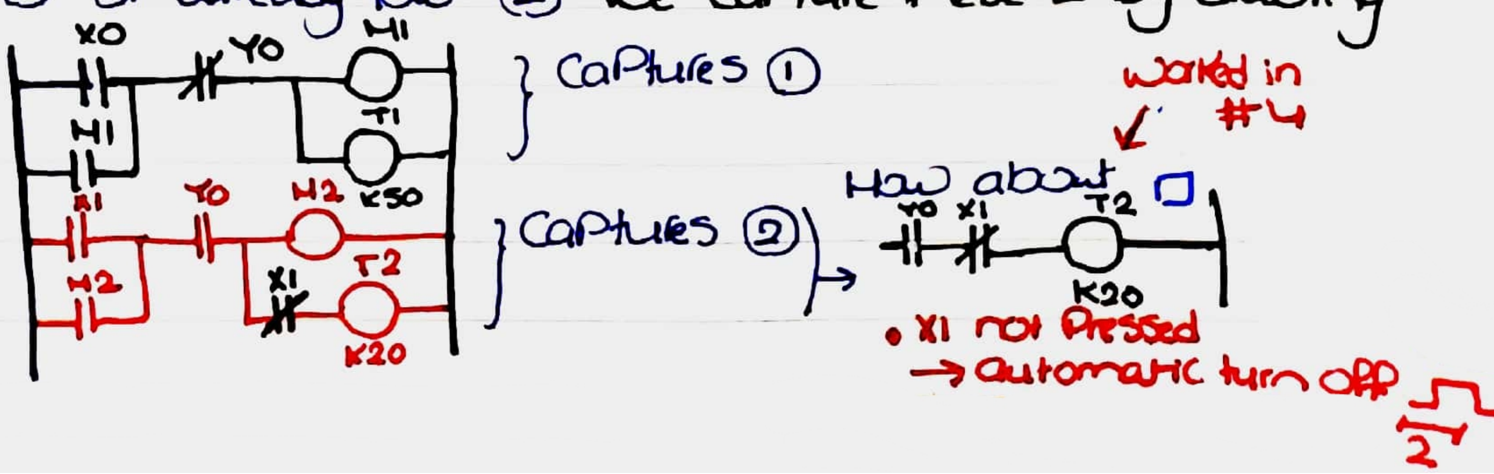
.Also solved in lecture



1. Rising edge on X0 sets a 5 second timer and it keeps timing even after X0=0 → The timer sets Y0 high at timeout

2. Falling edge on X1 sets a 2 second timer
→ Such falling edge must come after a rising edge (i.e check for rising then add a condition to trigger at falling)

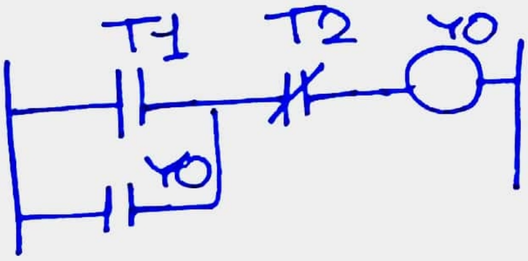
→ Since no timer should be set if Y0 is already high ① or already low ② we capture these 2 by drawing



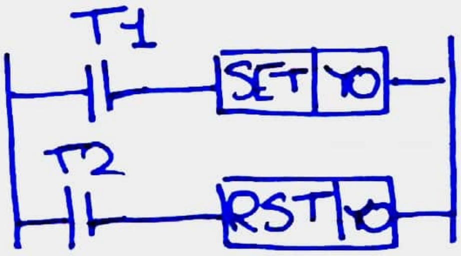
- Note that M1, M2 are needed so that the rising edge on X0, X1 resp. is captured whenever it occurs

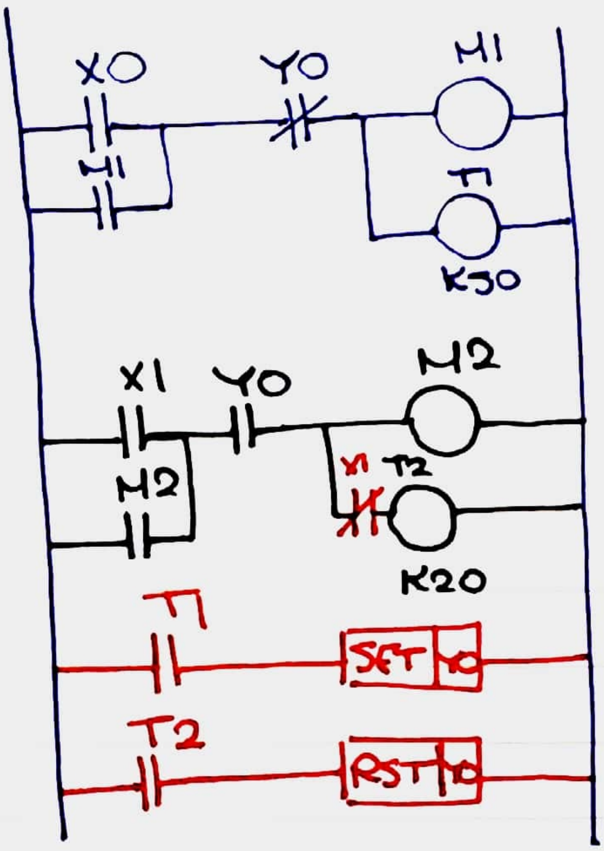
Now need to add as well

T1 time out → Y0 ≡ 1 } should maintain its
T2 time out → Y0 ≡ 0 } State (timers ⎍)



or





LD X0
OR M1
ANI Y0
OUT M1
OUT T1
K50
LD X1
OR M2
AND Y0
Out M2
ANI X1
OUT T2
K20
LD T1
SET Y0
LD T2
RST Y0

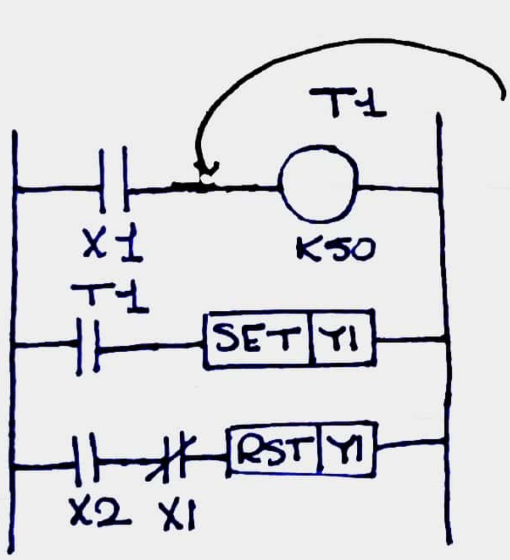
7)

- X1 Continuously Pressed for 5sec or more → activate Y1

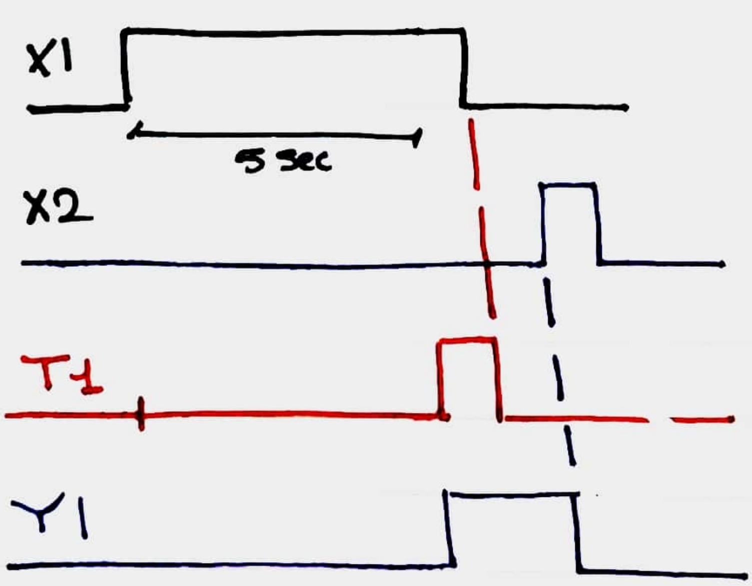
  else → not activated

- X2 Pressed while X1 isn't → Deactivate Y1

Y1 (only when it's off)

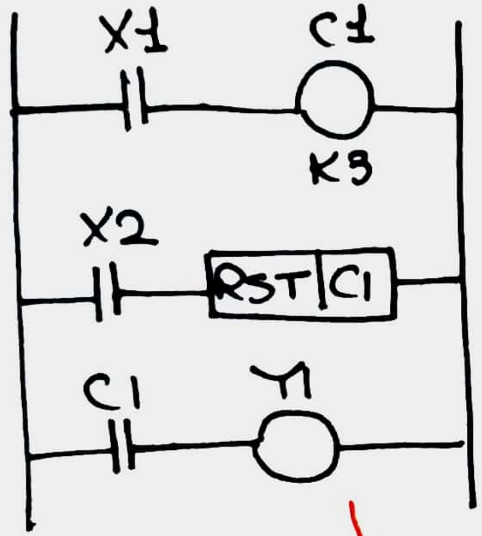


LD X1
ANI Y1
OUT T1
K50
LD T1
SET Y1
LD X2
LDI X1
RST Y1



8)

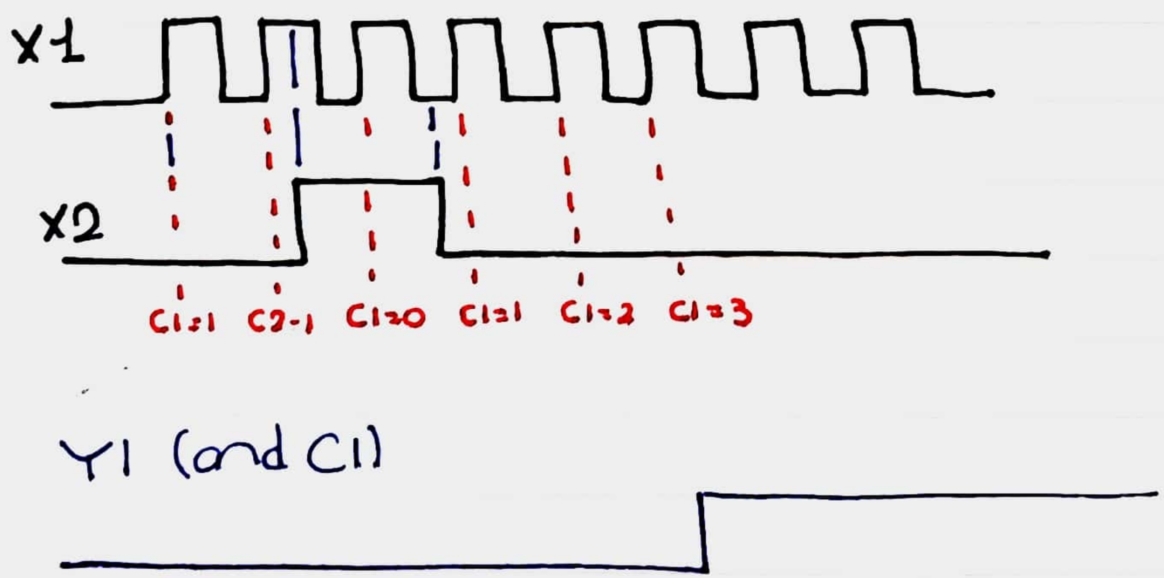
Will be 1 after 3 rising edges on X1




```
LD X1
OUT C1
  K3
LD X2
RST C1
LD C1
OUT Y1
```

Reset when X2 is Pressed

Make value available at Y1

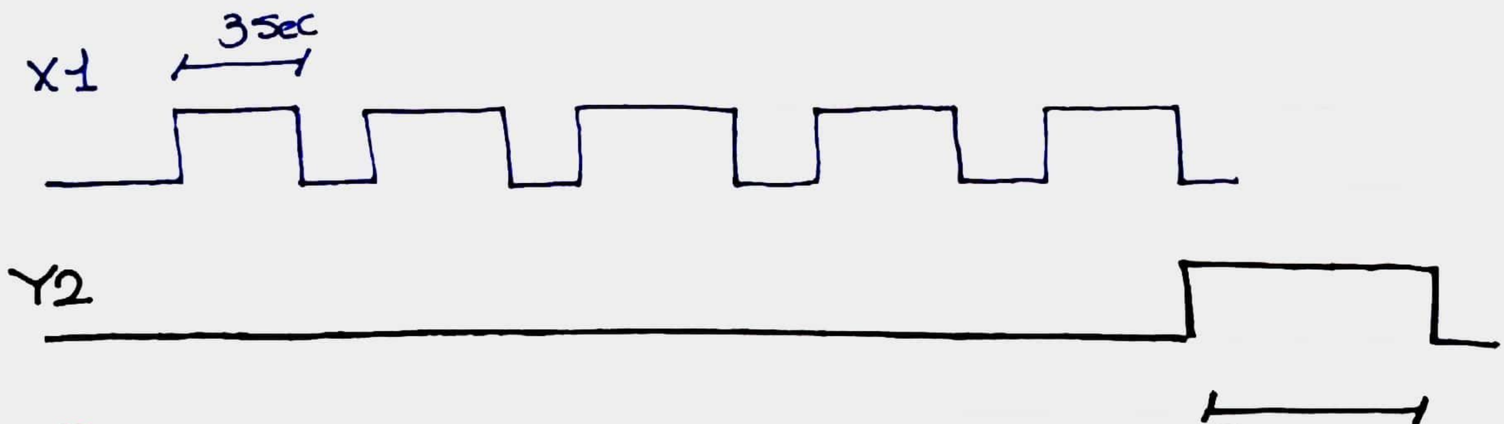


9) Draw ladder diagram such that

- X1 is pressed and released 5 times with each press lasting $\geq$ 3 seconds

$\longrightarrow$

Y2 is activated for 20 minutes



20 min. = 12 000 sec

- Let's start by ignoring the 20 minute constraint.

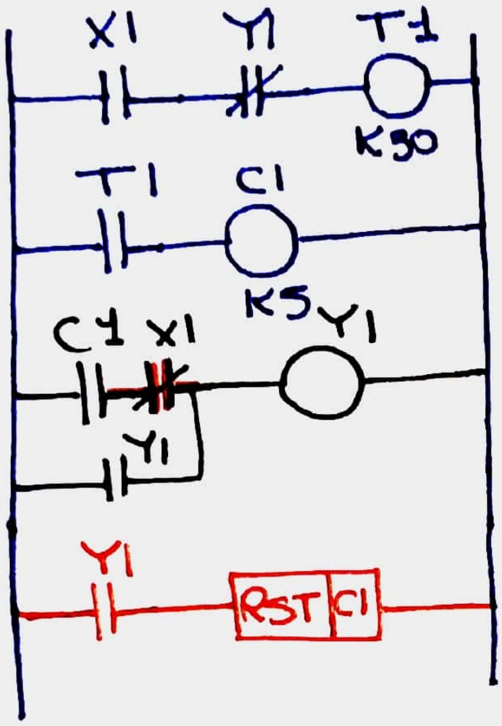
→ Use a timer that is set when X1 is high for $\geq$ 3 seconds

→ Let a counter count the rising edges on that timer (each is associated with an X1 pulse lasting $\geq$ 3 seconds)

→ Such counter would be reset when Y1 goes high (Objective fulfilled)

→ Hence, when it sets Y2's value we'll need to self-latch

Thus,



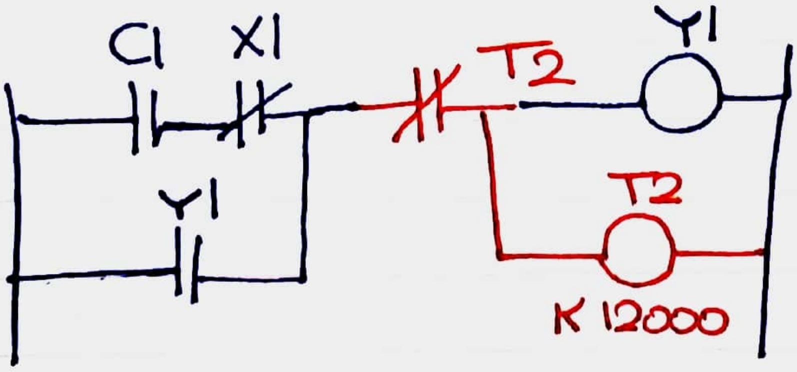
Count Pulses of T > 3 sec (and set at 5 of them)

whenever 5 pulses are counted (and it's assured that the last one is also over) then set Y1

reset counter when Y1 is high

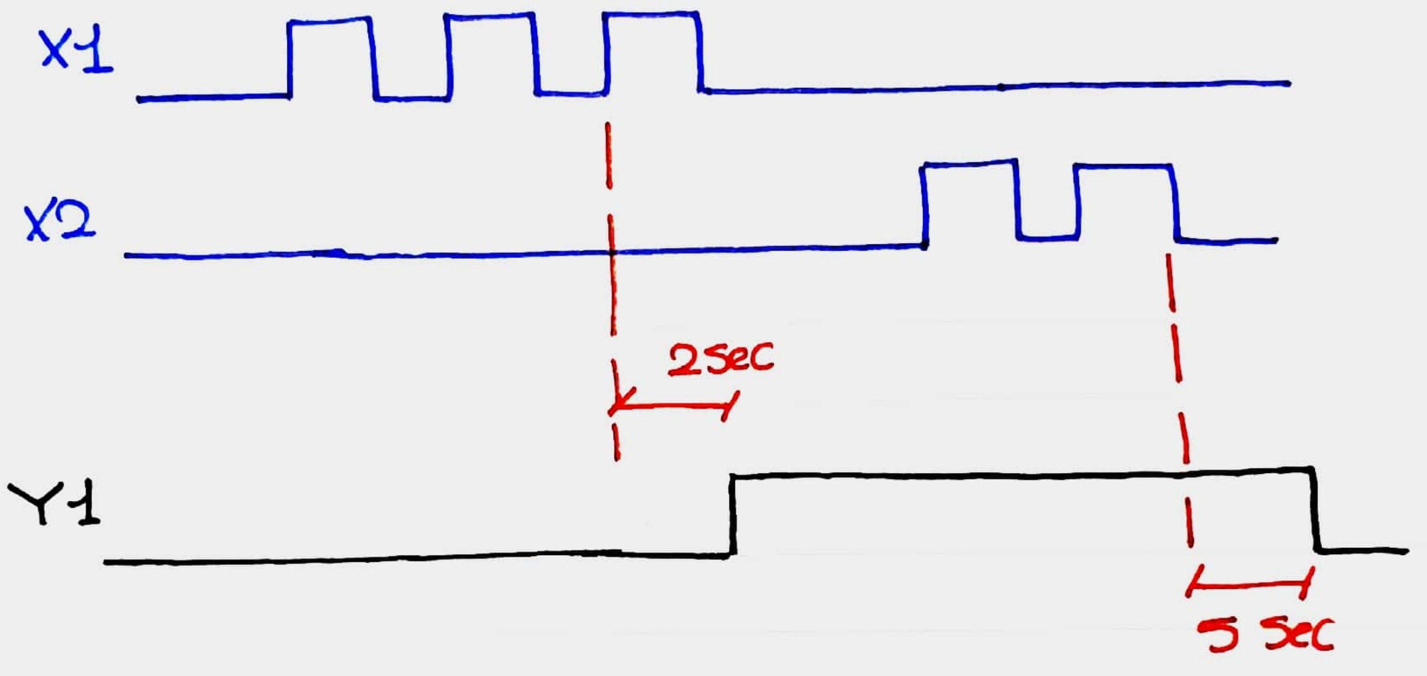
- Now if we want the output to only last for 12000 deciseconds we can introduce a disable latch condition to the 3rd ring (it has none so far) that comes from a timer that starts when Y1 is set.

i.e. update the 3rd ring to



10) 6-a)

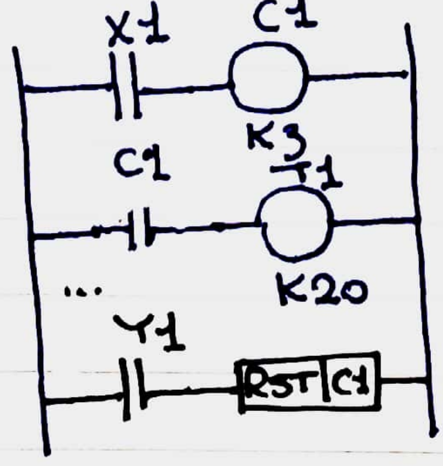
• Solved in lecture

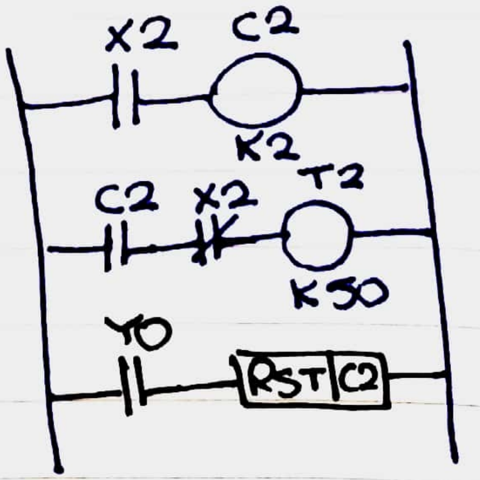


1. Y1 is set once X1 rises 3 times and then 2 seconds have lasted → reset counter at Y1=1

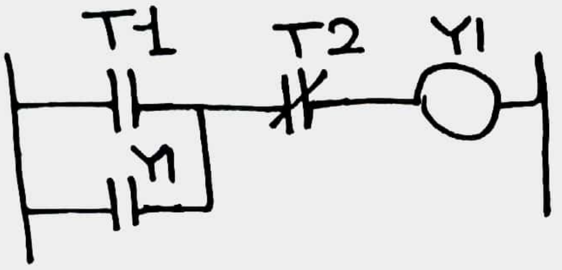
2. It's reset once X2 falls 2 times and then 5 seconds have lasted → reset counter at Y1=0

• Once we're done with these, timer output logic will be next





Timer-Output logic

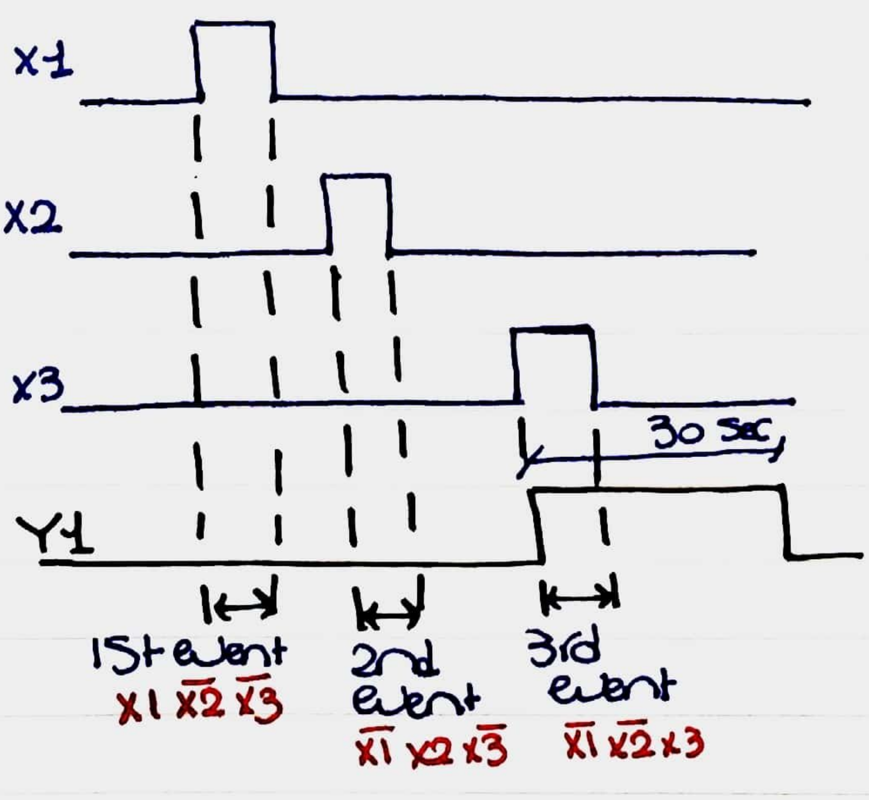


- Set whenever T1 timesout
- Reset whenever T2 timesout

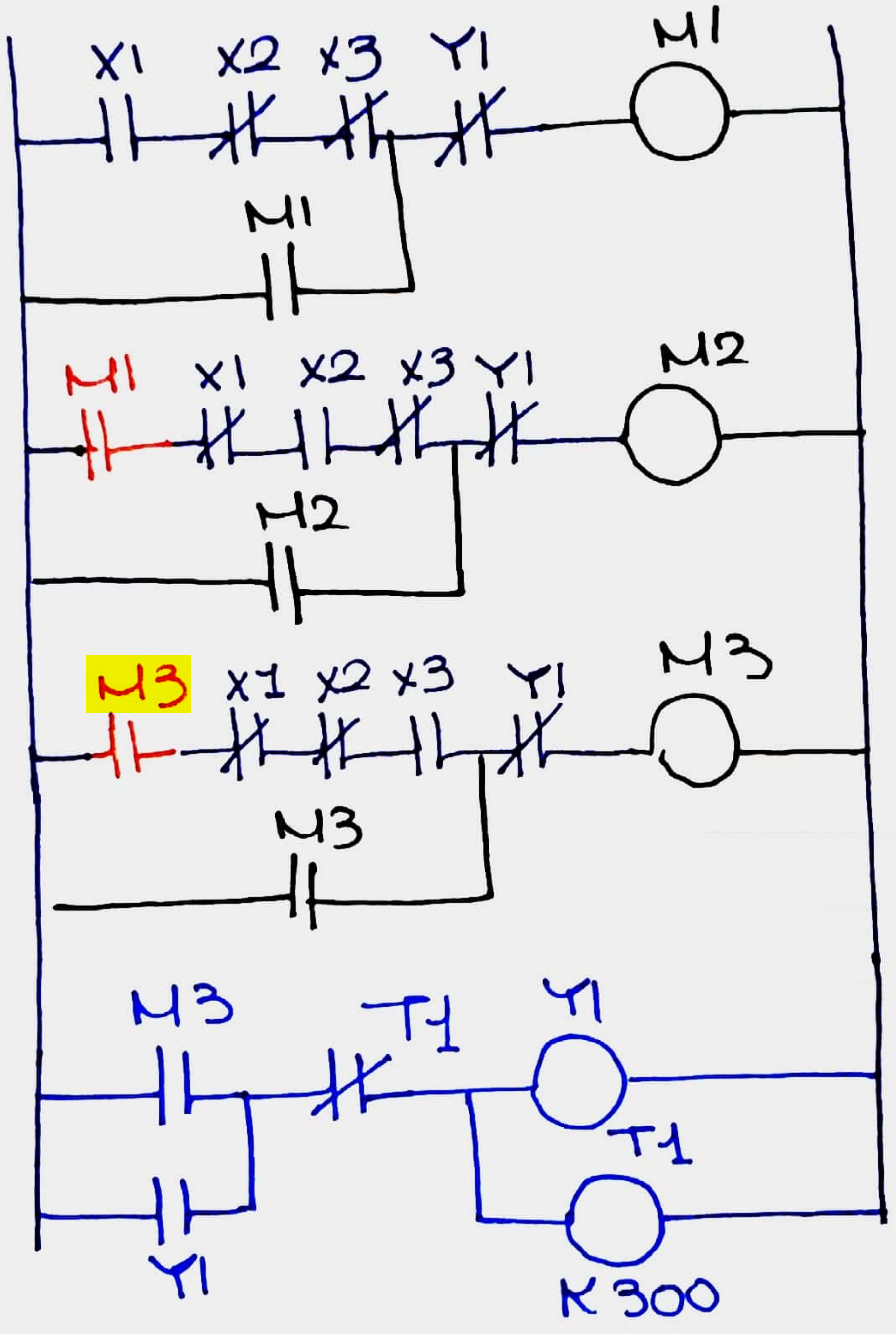
- Now need to revisit ①, ② so that
  - → Counting is disabled if Y is already in the right state
  - → Rising edges don't count if the other input is high (perhaps* good practice but optional)

*Check [CE lecture 9, 13] to round it out.

10) 16-b)



- Have a sequence of events that should take place at a particular order.
  - → when event #1 occurs, store in a marker
  - → use it while checking for next event & store in another marker...



• M1 stores occurence of 1st event

• stores occurence of 1st & 2nd event

• Stores occurence of all 3 events (rising edge at 3rd)

• take action based on M3 (all events occured)

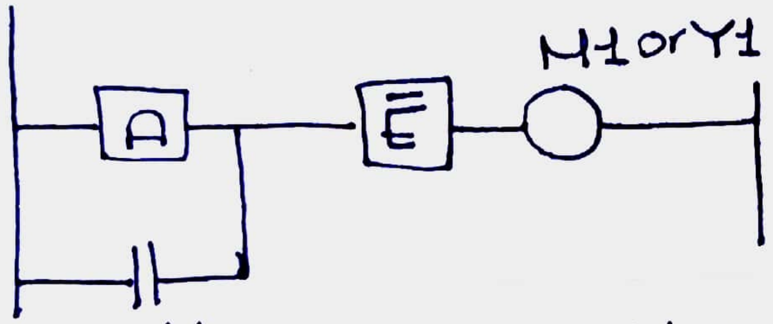
* Notice that once Y1 goes high we reset the markers (Objective of Y1 = 1 is done and so they can be reused)

* Confirm that you have the ability to write the instruction programs for last 2 ladders.

Thank you !! :)

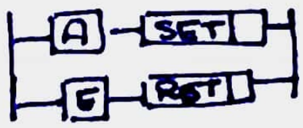
Some PLC Common Patterns

1. Self-latching

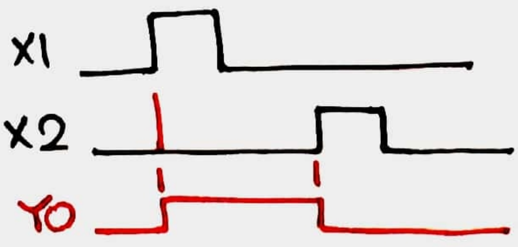


- Want M1 or Y1 to be high once A is satisfied.
- and to reset once / be disabled as long as E is satisfied.

• Also equivalent to

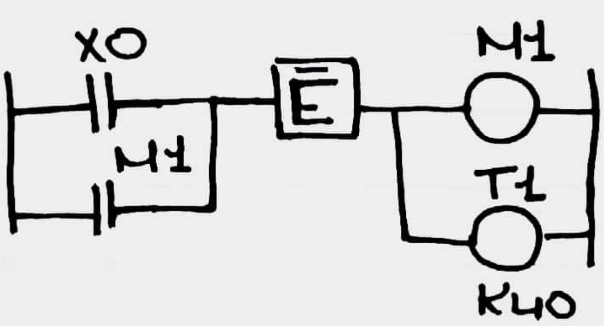


Basic example) A = –| |– X0 Ē = –|/|– X2



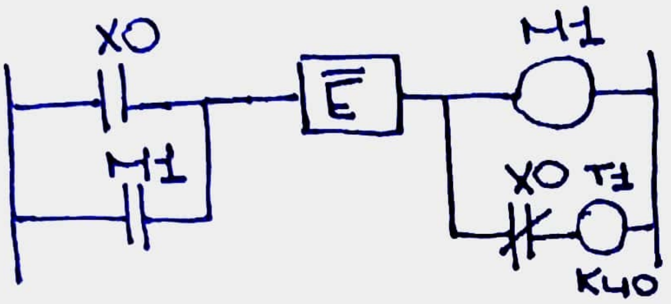
• Models a Push button

2. Timer starts at rising edge of X0 (e.g. button press)



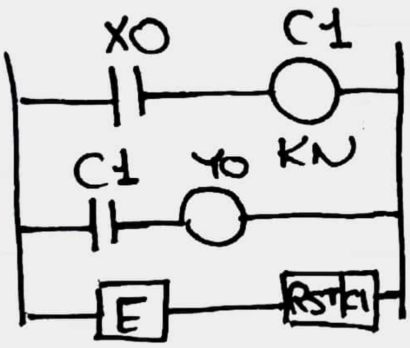
- E is what resets the timer
- M1 makes it possible to drop X0 again (+ve edge triggered)

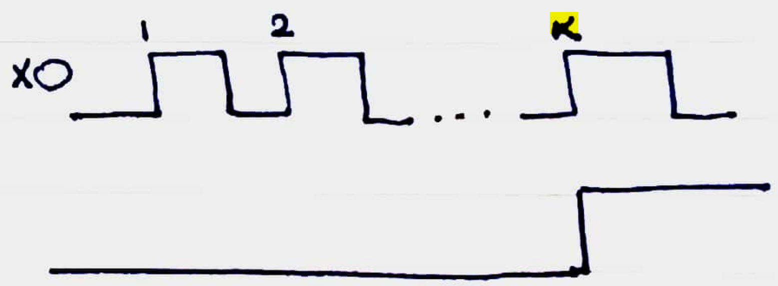
3. Timer Starts at falling edge of X0 (button release)



- Just like the rising edge case but we wait first for X0 ⫽ (falling edge) to trigger the timer.

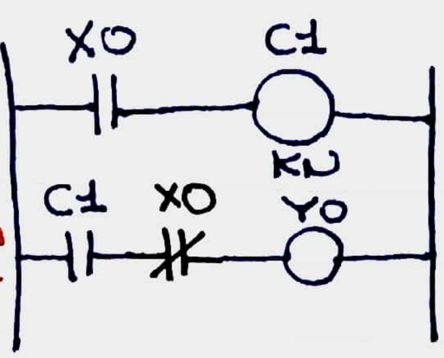
4. Output enabled after N rising edges





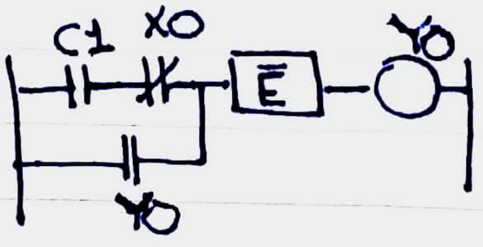
5. Output enabled after N falling edges

Y0 will = 0 when Counter reset (or X0 ↓)



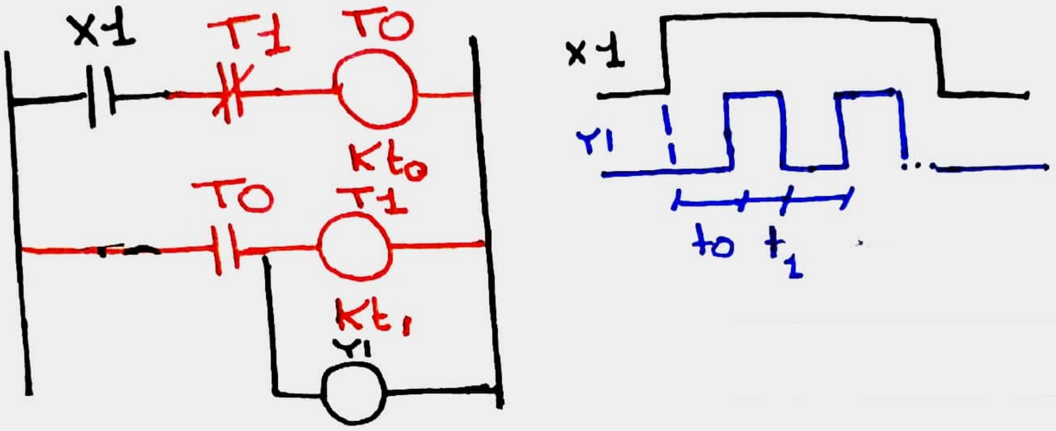
- This counts N rising edges
- Then waits for one more falling edge before energizing output.

Solution

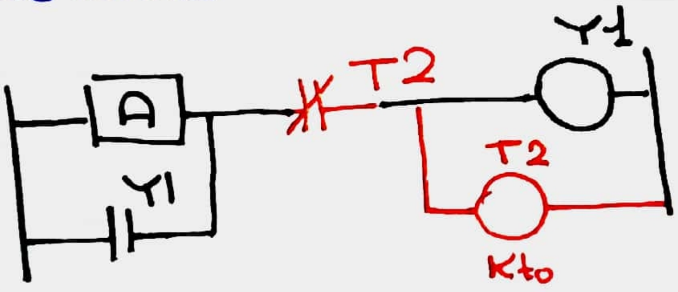


← Y0 is set if N falling edges occur & reset upon E.

6. Clock that is off for t_0 deciseconds and on for t_1 deciseconds as long as X1 is high

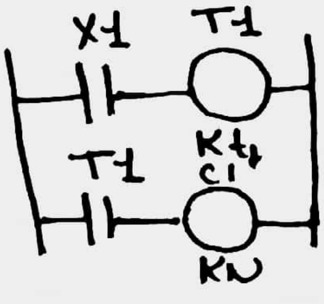


7. Output that last to deciseconds only since activated

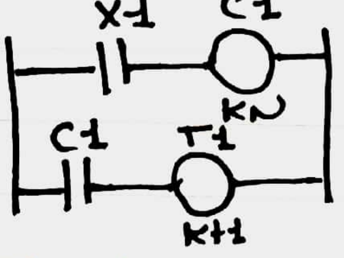


• This assume no other exit cond. (else, AND it)

8. Count the no. of input pulses (N) each of duration $\geq t_1$ | rising edge



9. Wait for t_1 after N rising edges



← the timer will be reset by the counter