

## 9. LATCHES, TIMERS, COUNTERS AND MORE

Topics:

- Latches, timers, counters and MCRs
- Design examples
- Internal memory locations are available, and act like outputs

Objectives:

- Understand latches, timers, counters and MCRs.
- To be able to select simple internal memory bits.

### 9.1 INTRODUCTION

More complex systems cannot be controlled with combinatorial logic alone. The main reason for this is that we cannot, or choose not to add sensors to detect all conditions. In these cases we can use events to estimate the condition of the system. Typical events used by a PLC include;

first scan of the PLC - indicating the PLC has just been turned on  
time since an input turned on/off - a delay  
count of events - to wait until set number of events have occurred  
latch on or unlatch - to lock something on or turn it off

The common theme for all of these events is that they are based upon one of two questions "How many?" or "How long?". An example of an event based device is shown in Figure 95. The input to the device is a push button. When the push button is pushed the input to the device turns on. If the push button is then released and the device turns off, it is a logical device. If when the push button is released the device stays on, it will be one type of event based device. To reiterate, the device is event based if it can respond to one or more things that have happened before. If the device responds only one way to the immediate set of inputs, it is logical.

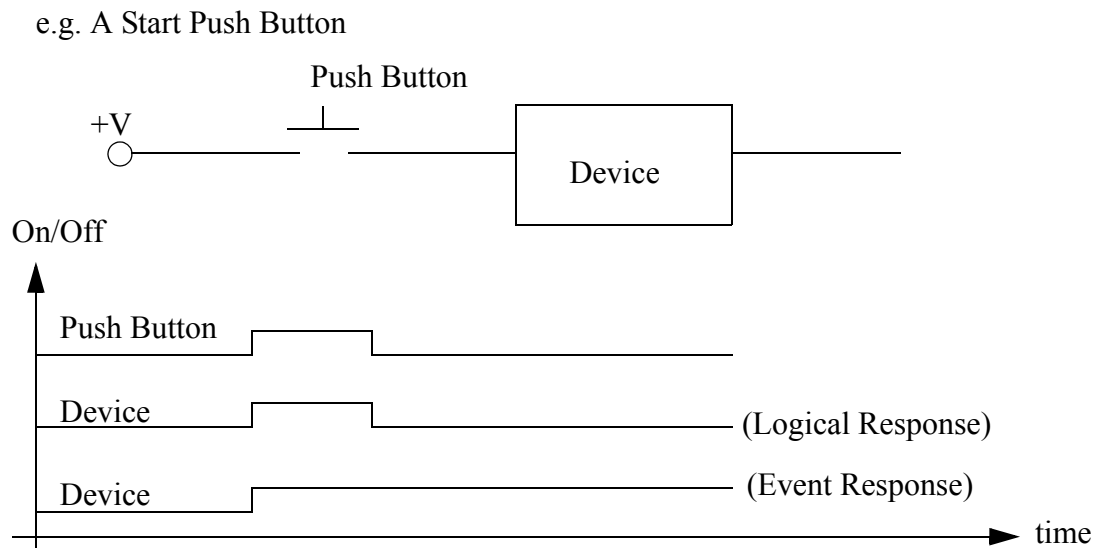


Figure 95 An Event Driven Device

## 9.2 LATCHES

A latch is like a sticky switch - when pushed it will turn on, but stick in place, it must be pulled to release it and turn it off. A latch in ladder logic uses one instruction to latch, and a second instruction to unlatch, as shown in Figure 96. The output with an *L* inside will turn the output *D* on when the input *A* becomes true. *D* will stay on even if *A* turns off. Output *D* will turn off if input *B* becomes true and the output with a *U* inside becomes true (Note: this will seem a little backwards at first). If an output has been latched on, it will keep its value, even if the power has been turned off.

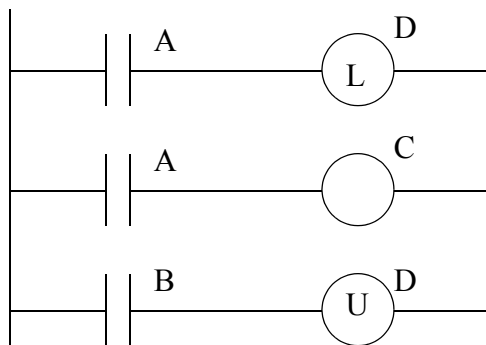
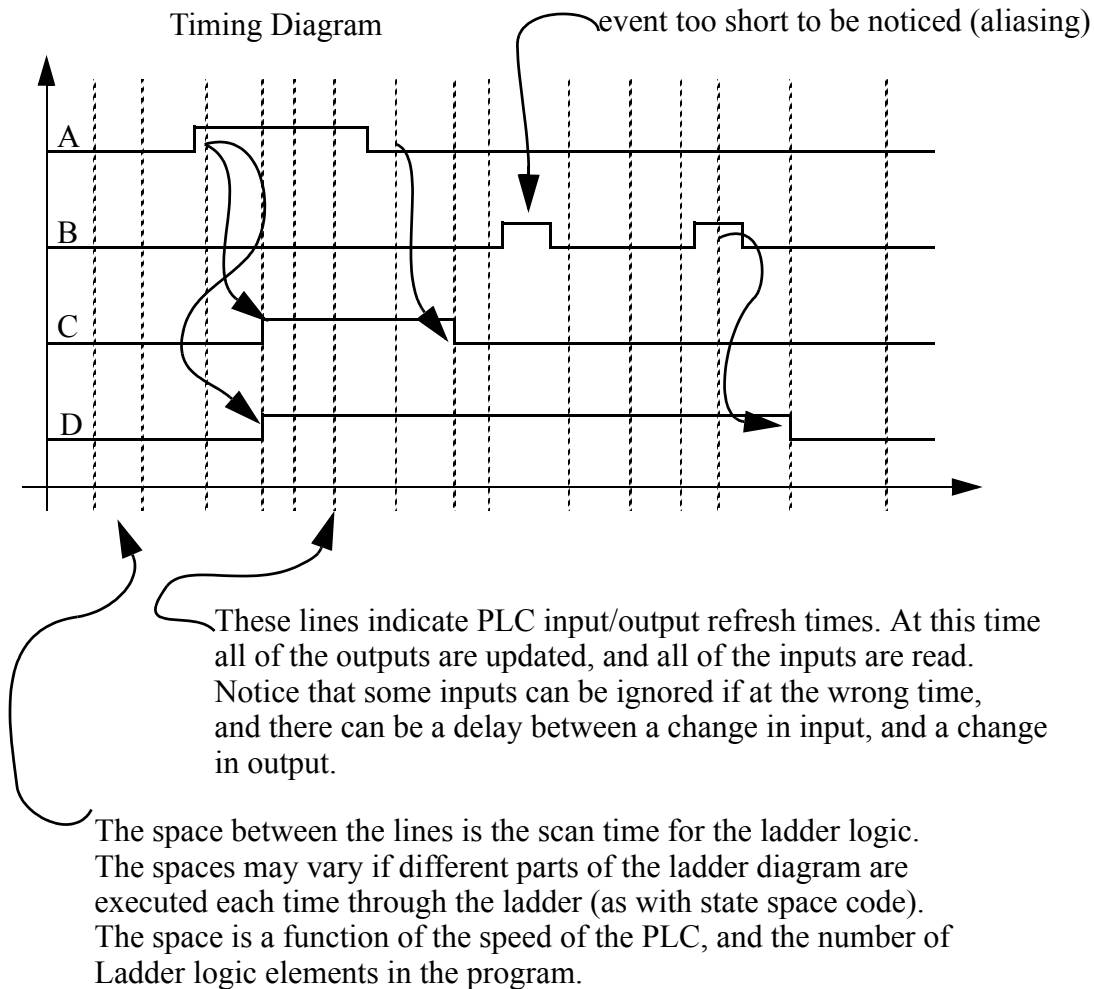


Figure 96 A Ladder Logic Latch

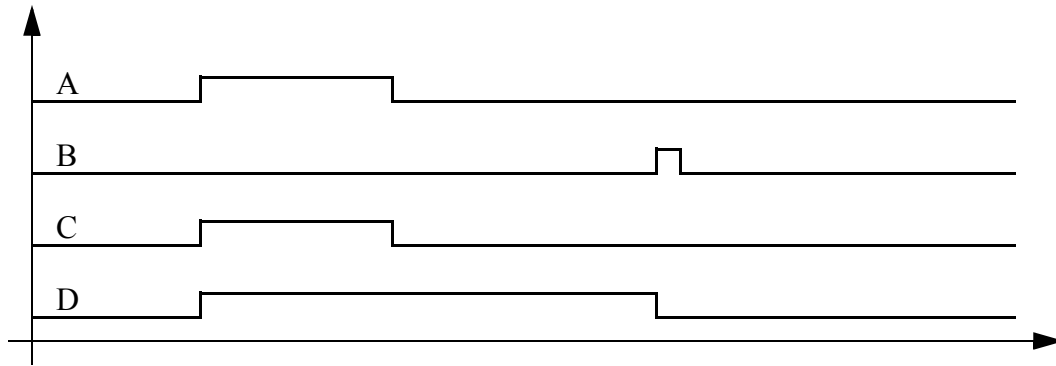
The operation of the ladder logic in Figure 96 is illustrated with a timing diagram in Figure 97. A timing diagram shows values of inputs and outputs over time. For example the value of input *A* starts low (false) and becomes high (true) for a short while, and then goes low again. Here when input *A* turns on both the outputs turn on. There is a slight delay between the change in inputs and the resulting changes in outputs, due to the program scan time. Here the dashed lines represent the output scan, sam-

ity check and input scan (assuming they are very short.) The space between the dashed lines is the ladder logic scan. Consider that when *A* turns on initially it is not detected until the first dashed line. There is then a delay to the next dashed line while the ladder is scanned, and then the output at the next dashed line. When *A* eventually turns off, the normal output *C* turns off, but the latched output *D* stays on. Input *B* will unlatch the output *D*. Input *B* turns on twice, but the first time it is on is not long enough to be detected by an input scan, so it is ignored. The second time it is on it unlatches output *D* and output *D* turns off.



*Figure 97* A Timing Diagram for the Ladder Logic in Figure 96

The timing diagram shown in Figure 97 has more details than are normal in a timing diagram as shown in Figure 98. The brief pulse would not normally be wanted, and would be designed out of a system either by extending the length of the pulse, or decreasing the scan time. An ideal system would run so fast that aliasing would not be possible.



*Figure 98* A Typical Timing Diagram

A more elaborate example of latches is shown in Figure 99. In this example the addresses are for an older Allen-Bradley Micrologix controller. The inputs begin with *I/*, followed by an input number. The outputs begin with *O/*, followed by an output number.

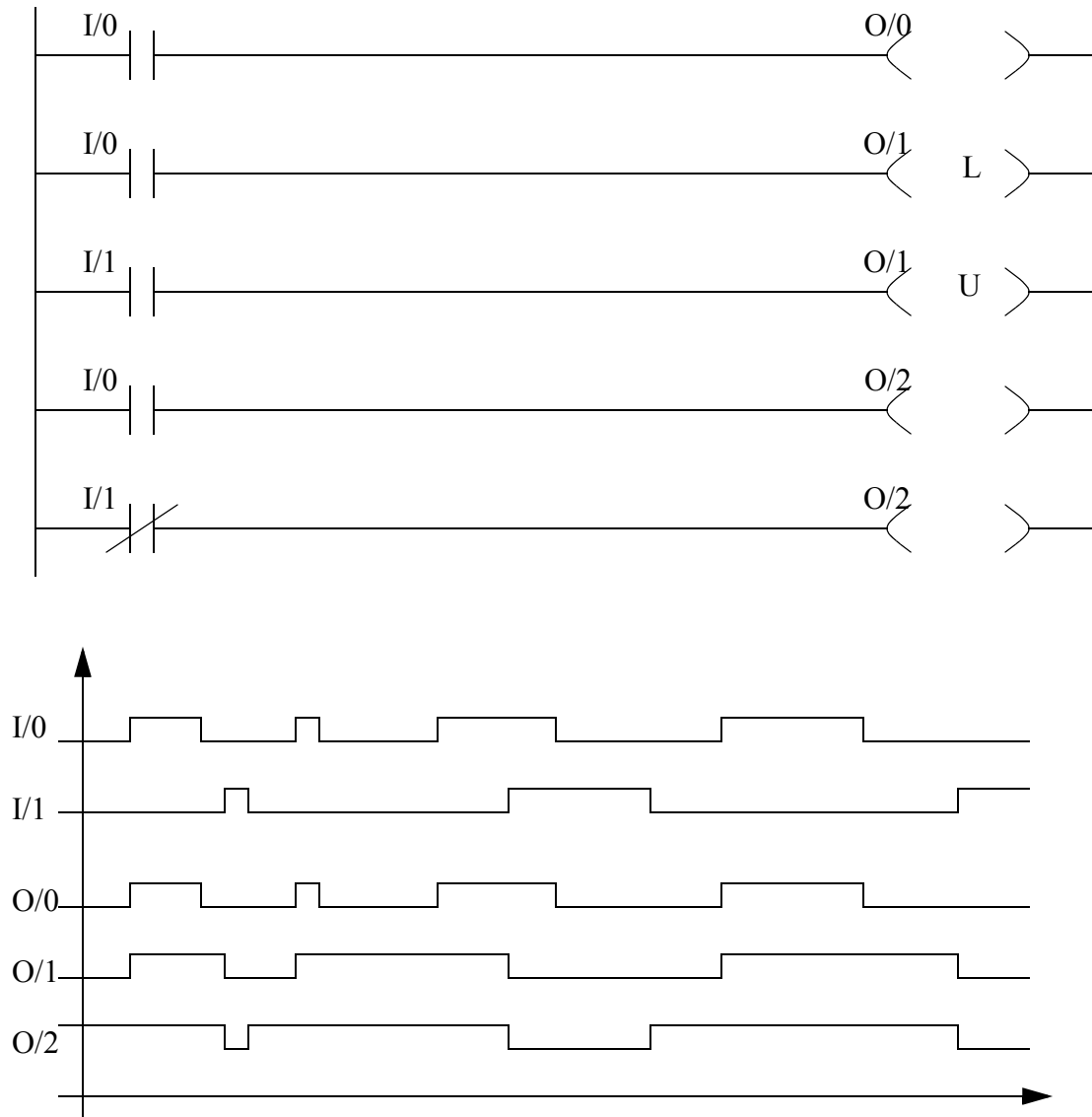


Figure 99 A Latch Example

A normal output should only appear once in ladder logic, but latch and unlatch instructions may appear multiple times. In Figure 99 a normal output  $O/2$  is repeated twice. When the program runs it will examine the fourth line and change the value of  $O/2$  in memory (remember the output scan does not occur until the ladder scan is done.) The last line is then interpreted and it overwrites the value of  $O/2$ . Basically, only the last line will change  $O/2$ .

Latches are not used universally by all PLC vendors, others such as Siemens use flip-flops. These have a similar behavior to latches, but a different notation as illustrated in Figure 100. Here the flip-flop is an output block that is connected to two different logic rungs. The first rung shown has an input  $A$  connected to the  $S$  setting terminal. When  $A$  goes true the output value  $Q$  will go true. The second rung has an input  $B$  connected to the  $R$  resetting terminal. When  $B$  goes true the output value  $Q$  will be turned off. The output  $Q$  will always be the inverse of  $\bar{Q}$ . Notice that the  $S$  and  $R$  values are equivalent to the  $L$  and  $U$  values from earlier examples.

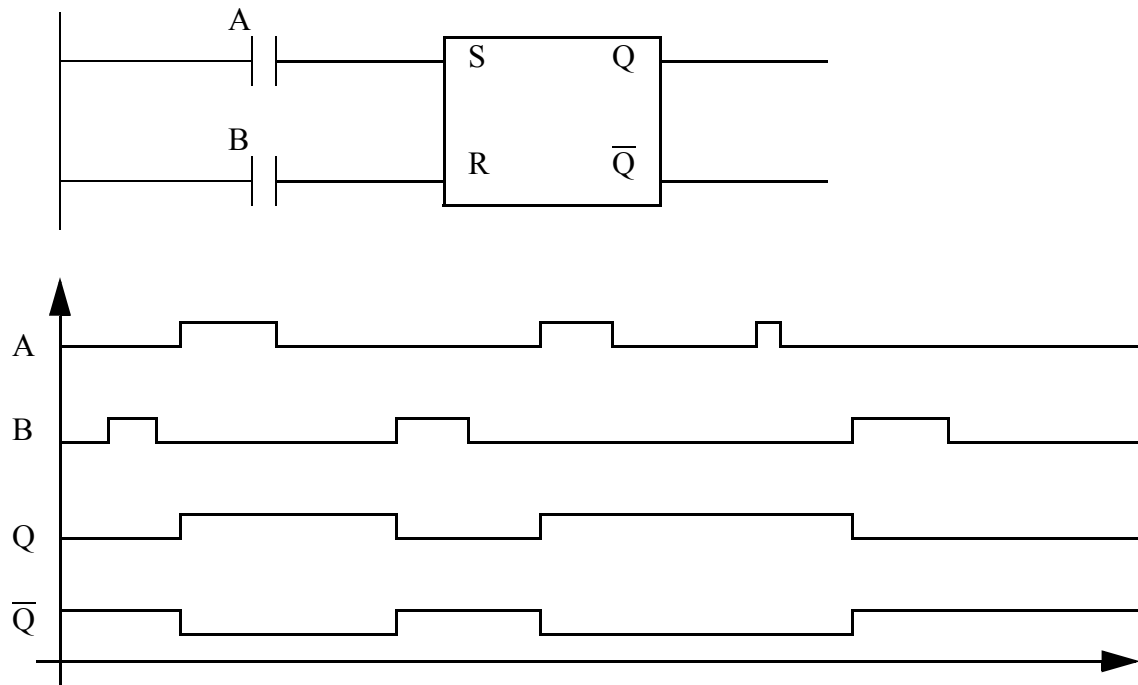


Figure 100 Flip-Flops for Latching Values

### 9.3 TIMERS

There are four fundamental types of timers shown in Figure 101. An on-delay timer will wait for a set time after a line of ladder logic has been true before turning on, but it will turn off immediately. An off-delay timer will turn on immediately when a line of ladder logic is true, but it will delay before turning off. Consider the example of an old car. If you turn the key in the ignition and the car does not start immediately, that is an on-delay. If you turn the key to stop the engine but the engine doesn't stop for a few seconds, that is an off delay. An on-delay timer can be used to allow an oven to reach temperature before starting production. An off delay timer can keep cooling fans on for a set time after the oven has been turned off.

	on-delay	off-delay
retentive	RTO	RTF
nonretentive	TON	TOF

TON - Timer ON

TOF - Timer OFF

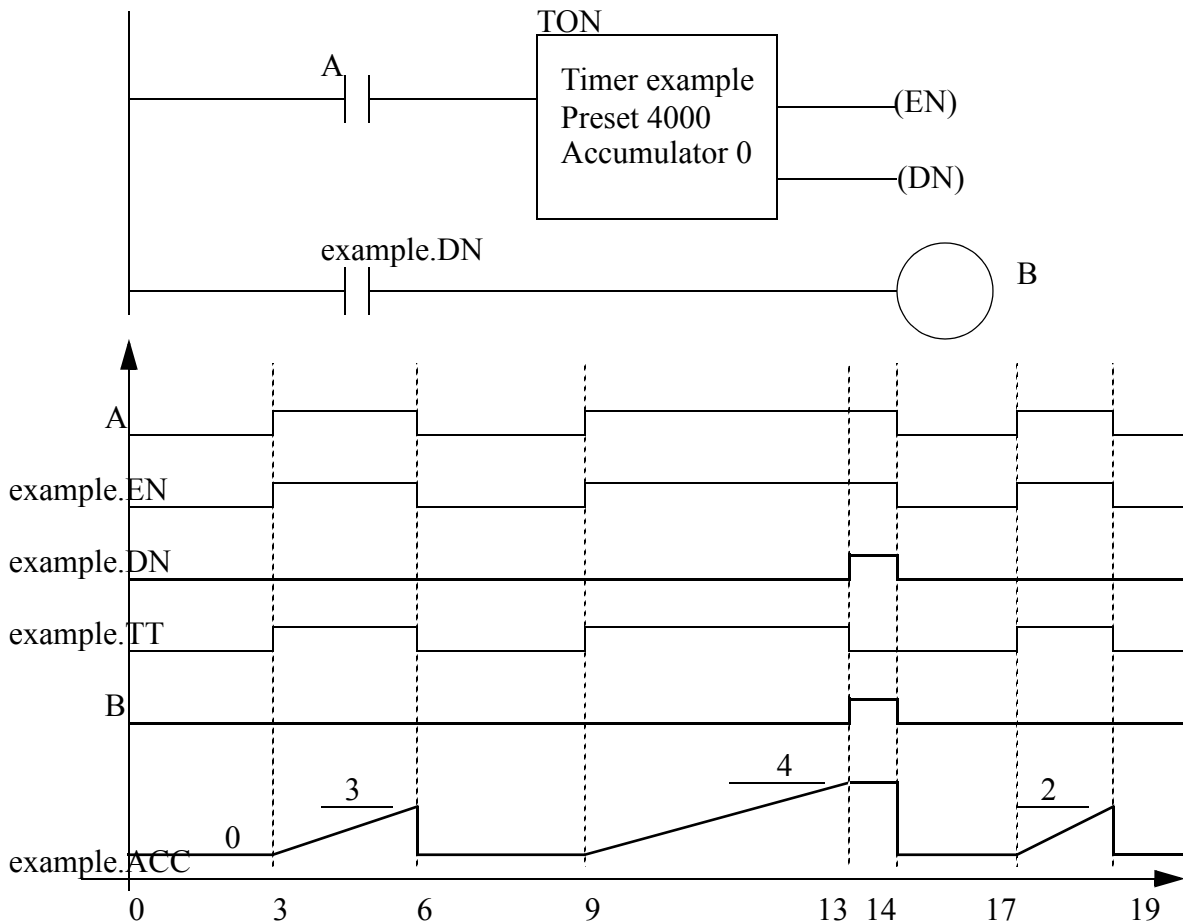
RTO - Retentive Timer On

RTF - Retentive Timer oFF

*Figure 101* The Four Basic Timer Types

A retentive timer will sum all of the on or off time for a timer, even if the timer never finished. A nonretentive timer will start timing the delay from zero each time. Typical applications for retentive timers include tracking the time before maintenance is needed. A non retentive timer can be used for a start button to give a short delay before a conveyor begins moving.

An example of an Allen-Bradley TON timer is shown in Figure 102. The rung has a single input *A* and a function block for the *TON*. (Note: This timer block will look different for different PLCs, but it will contain the same information.) The information inside the timer block describes the timing parameters. The first item is the timer 'example'. This is a location in the PLC memory that will store the timer information. The preset is the millisecond delay for the timer, in this case it is 4s (4000ms). The accumulator value gives the current value of the timer as 0. While the timer is running the accumulated value will increase until it reaches the preset value. Whenever the input *A* is true the *EN* output will be true. The *DN* output will be false until the accumulator has reached the preset value. The *EN* and *DN* outputs cannot be changed when programming, but these are important when debugging a ladder logic program. The second line of ladder logic uses the timer *DN* output to control another output *B*.



Note: For the older Allen-Bradley equipment the notations are similar, although the tag names are replaced with a more strict naming convention. The timers are kept in 'files' with names starting with 'T4:', followed by a timer number. The examples below show the older (PLC-5 and micrologix notations compared to the new RS-Logix (5000) notations. In the older PLCs the timer is given a unique number, in the RSLogix 5000 processors it is given a tag name (in this case 't') and type 'TIMER'.

Older	Newer
T4:0/DN	t.DN
T4:0/EN	t.EN
T4:0.PRE	t.PRE
T4:0.ACC	t.ACC
T4:0/TT	t.TT

Figure 102 An Allen-Bradley TON Timer

The timing diagram in Figure 102 illustrates the operation of the TON timer with a 4 second on-delay. *A* is the input to the timer, and whenever the timer input is true the *EN* enabled bit for the timer will also be true. If the accumulator value is equal to the preset value the *DN* bit will be set. Otherwise,



the *TT* bit will be set and the accumulator value will begin increasing. The first time *A* is true, it is only true for 3 seconds before turning off, after this the value resets to zero. (Note: in a retentive time the value would remain at 3 seconds.) The second time *A* is true, it is on more than 4 seconds. After 4 seconds the *TT* bit turns off, and the *DN* bit turns on. But, when *A* is released the accumulator resets to zero, and the *DN* bit is turned off.

A value can be entered for the accumulator while programming. When the program is downloaded this value will be in the timer for the first scan. If the TON timer is not enabled the value will be set back to zero. Normally zero will be entered for the preset value.

The timer in Figure 103 is identical to that in Figure 102, except that it is retentive. The most significant difference is that when the input *A* is turned off the accumulator value does not reset to zero. As a result the timer turns on much sooner, and the timer does not turn off after it turns on. A reset instruction will be shown later that will allow the accumulator to be reset to zero.

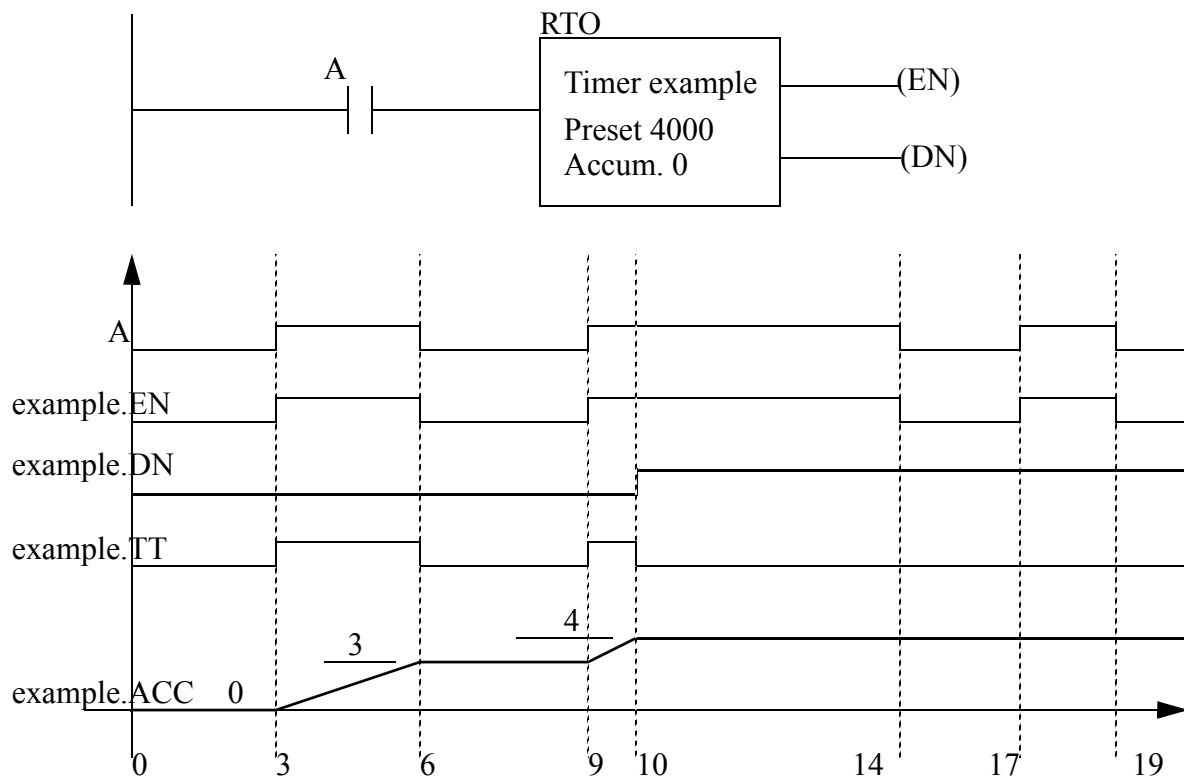


Figure 103 An Allen Bradley Retentive On-Delay Timer

An off delay timer is shown in Figure 104. This timer has a time base of 0.01s, with a preset value of 3500, giving a total delay of 3.5s. As before the *EN* enable for the timer matches the input. When the input *A* is true the *DN* bit is on. It is also on when the input *A* has turned off and the accumulator is counting. The *DN* bit only turns off when the input *A* has been off long enough so that the accumulator value reaches the preset. This type of timer is not retentive, so when the input *A* becomes true, the accumulator resets.

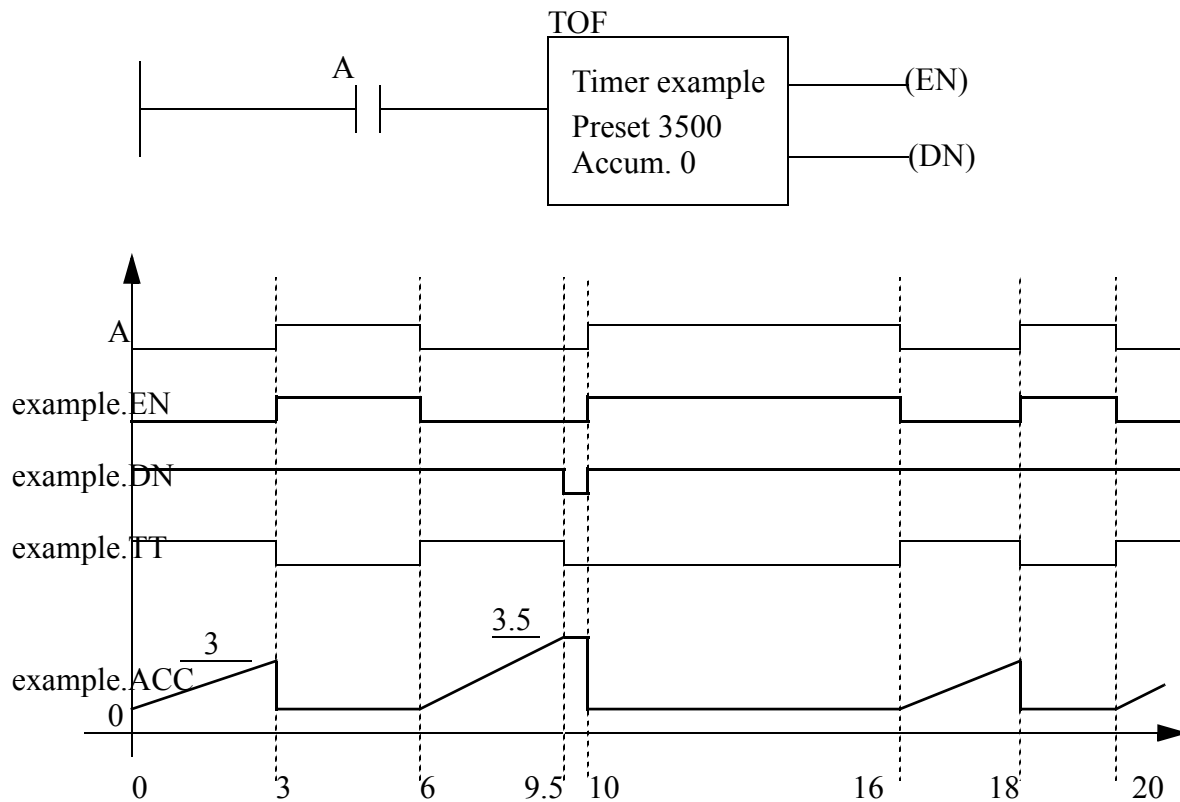


Figure 104 An Allen Bradley Off-Delay Timer

Retentive off-delay (RTF) timers have few applications and are rarely used, therefore many PLC vendors do not include them.

An example program is shown in Figure 105. In total there are four timers used in this example,  $t_1$ ,  $t_2$ ,  $t_3$ , and  $t_4$ . The timer instructions are shown with the accumulator values omitted, assuming that they start with a value of zero. All four different types of counters have the input 'go'. Output 'done' will turn on when the TON counter  $t_1$  is done. All four of the timers can be reset with input 'reset'.

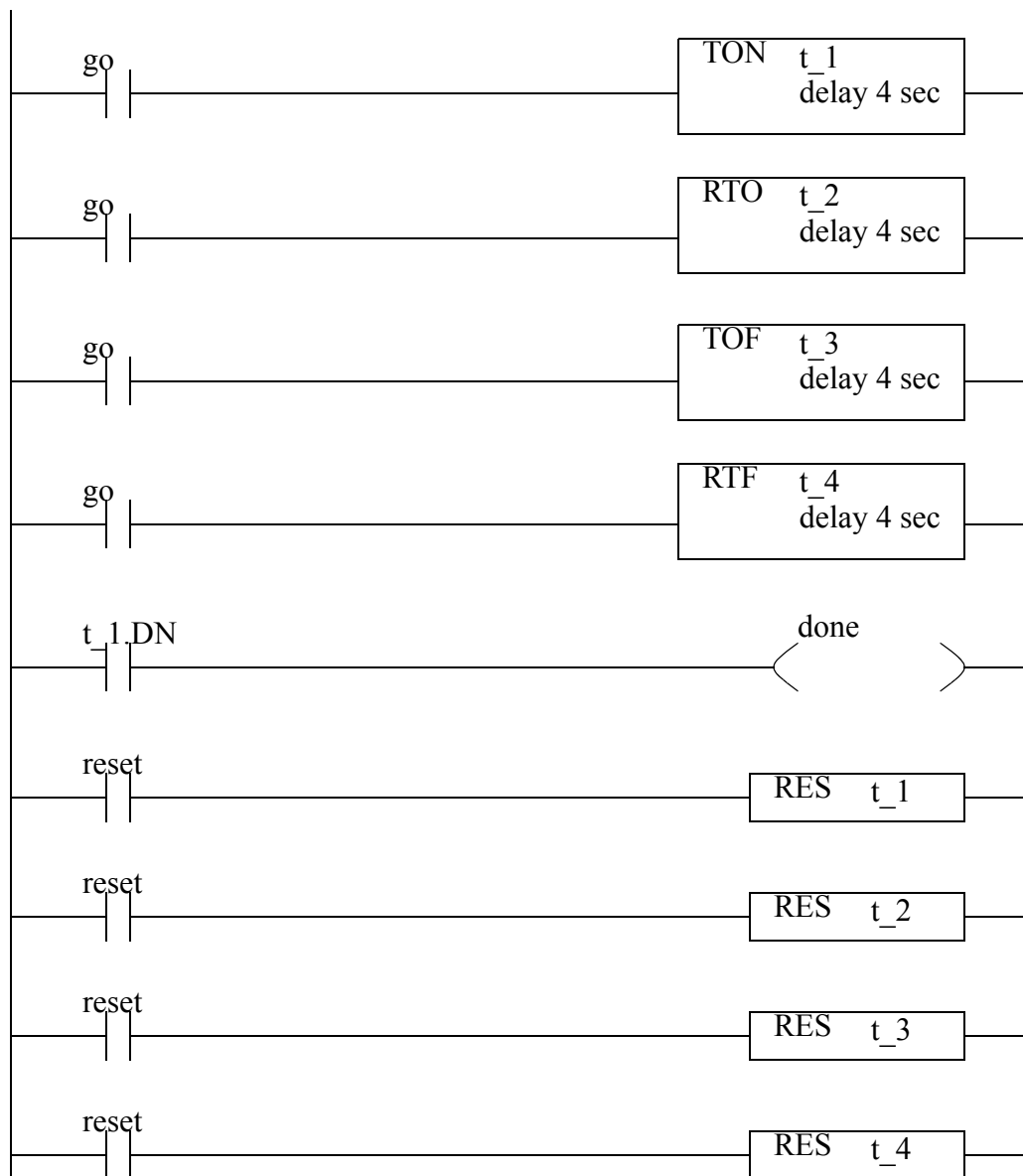


Figure 105 A Timer Example

A timing diagram for this example is shown in Figure 106. As input *go* is turned on the TON and RTO timers begin to count and reach 4s and turn on. When *reset* becomes true it resets both timers and they start to count for another second before *go* is turned off. After the input is turned off the TOF and RTF both start to count, but neither reaches the 4s preset. The input *go* is turned on again and the TON and RTO both start counting. The RTO turns on one second sooner because it had 1s stored from the 7-8s time period. After *go* turns off again both the off delay timers count down, and reach the 4 second delay, and turn on. These patterns continue across the diagram.

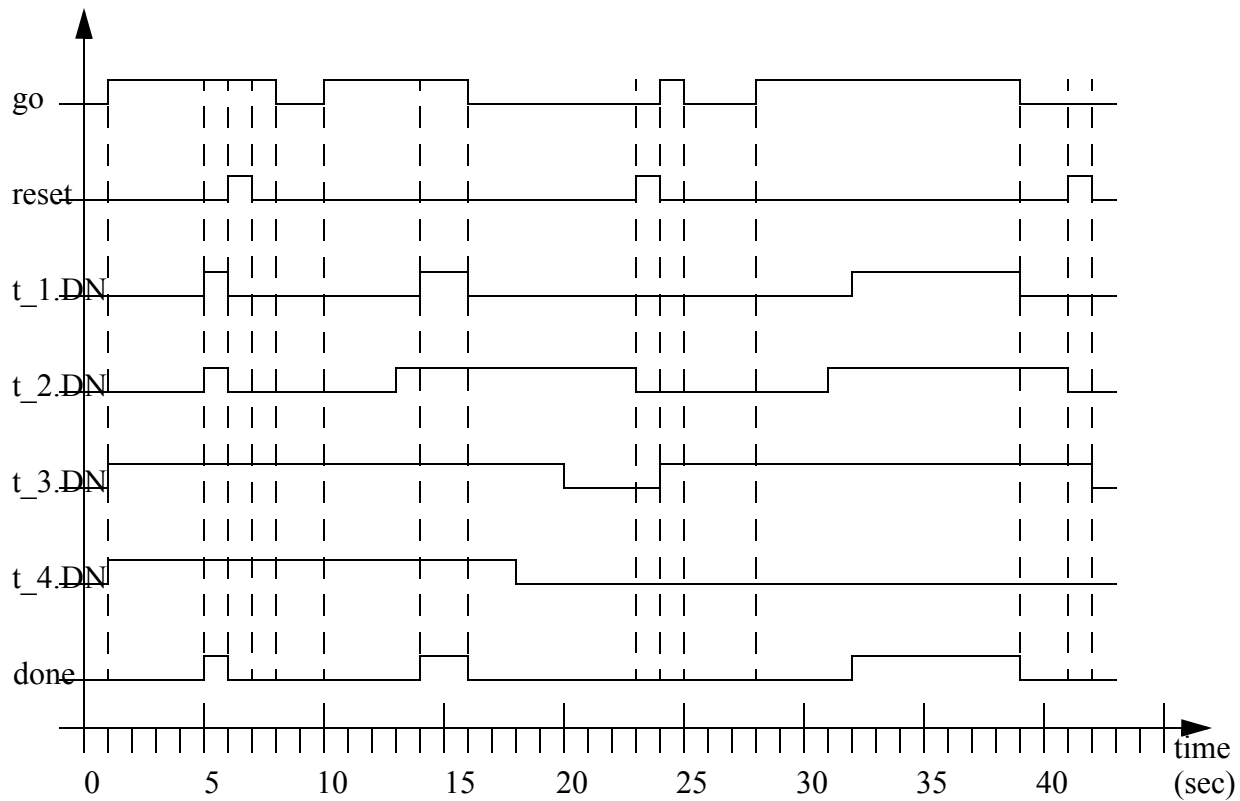


Figure 106 A Timing Diagram for Figure 105

Consider the short ladder logic program in Figure 107 for control of a heating oven. The system is started with a *Start* button that seals in the *Auto* mode. This can be stopped if the *Stop* button is pushed. (Remember: Stop buttons are normally closed.) When the *Auto* goes on initially the TON timer is used to sound the horn for the first 10 seconds to warn that the oven will start, and after that the horn stops and the heating coils start. When the oven is turned off the fan continues to blow for 300s or 5 minutes after.

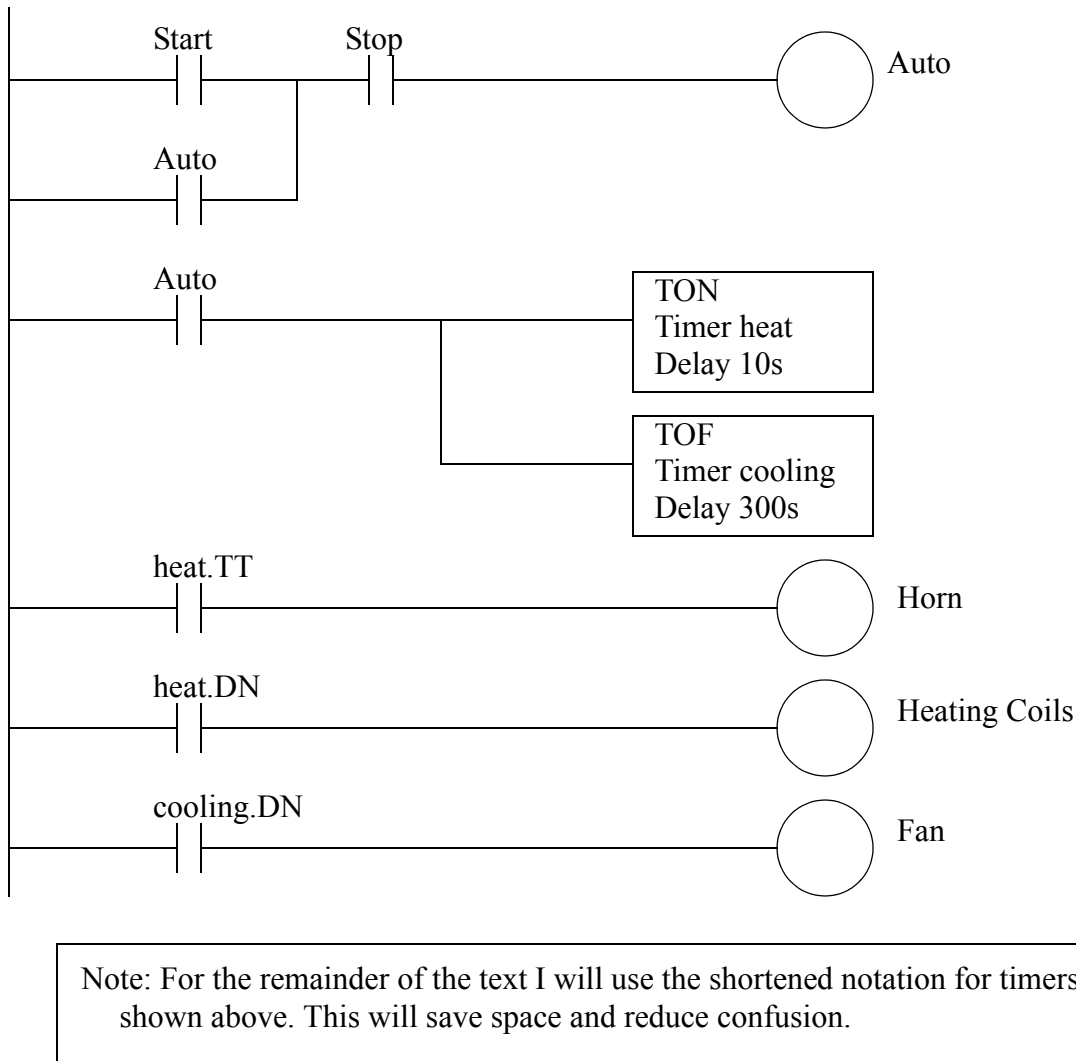


Figure 107 A Timer Example

A program is shown in Figure 108 that will flash a light once every second. When the PLC starts, the second timer will be off and the  $t_{on}.DN$  bit will be off, therefore the normally closed input to the first timer will be on.  $t_{off}$  will start timing until it reaches 0.5s, when it is done the second timer will start timing, until it reaches 0.5s. At that point  $t_{on}.DN$  will become true, and the input to the first timer will become false.  $t_{off}$  is then set back to zero, and then  $t_{on}$  is set back to zero. And, the process starts again from the beginning. In this example the first timer is used to drive the second timer. This type of arrangement is normally called cascading, and can use more than two timers.

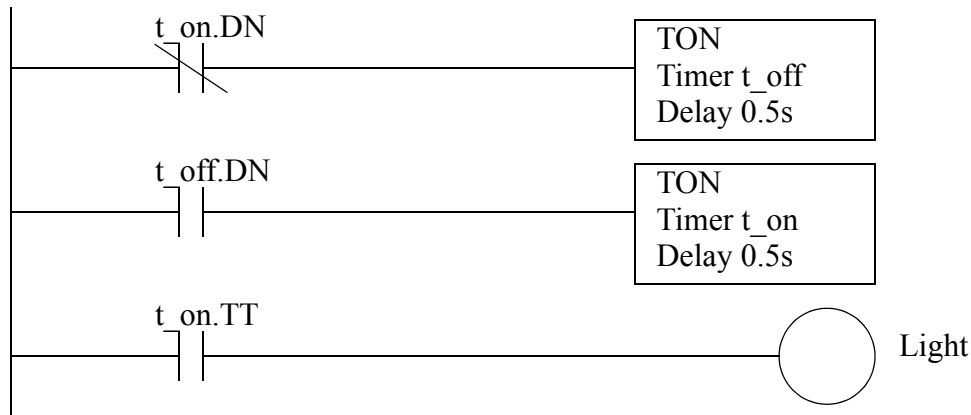
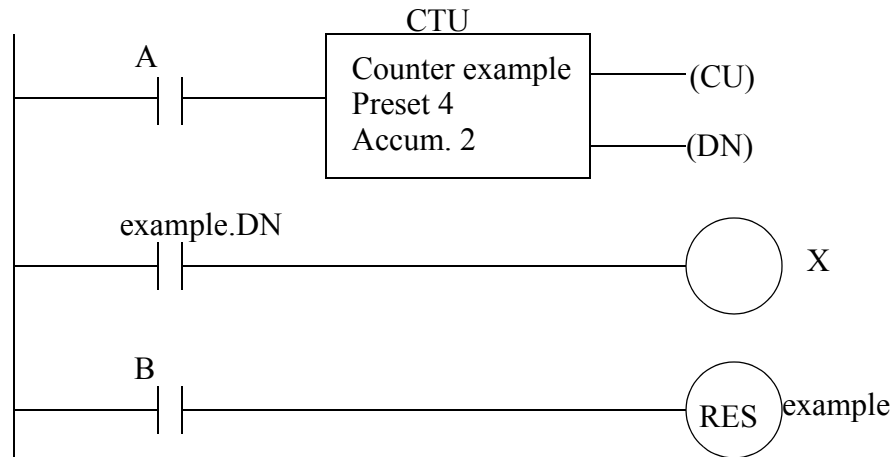


Figure 108 Another Timer Example

## 9.4 COUNTERS

There are two basic counter types: count-up and count-down. When the input to a count-up counter goes true the accumulator value will increase by 1 (no matter how long the input is true.) If the accumulator value reaches the preset value the counter *DN* bit will be set. A count-down counter will decrease the accumulator value until the preset value is reached.

An Allen Bradley count-up (CTU) instruction is shown in Figure 109. The instruction requires memory in the PLC to store values and status, in this case is *example*. The preset value is 4 and the value in the accumulator is 2. If the input *A* were to go from false to true the value in the accumulator would increase to 3. If *A* were to go off, then on again the accumulator value would increase to 4, and the *DN* bit would go on. The count can continue above the preset value. If input *B* becomes true the value in the counter accumulator will become zero.



Note: The notations for older Allen-Bradley equipment are very similar to the newer notations. The examples below show the older (PLC-5 and micrologix notations compared to the new RS-Logix (5000) notations. In the older PLCs the counter is given a unique name, in the RSLogix 5000 processors it is given a name (in this case 'c') and the type 'COUNTER'.

Older	Newer
C5:0/DN	c.DN
C5:0/CU	c.CU
C5:0.PRE	c.PRE
C5:0.ACC	c.ACC
C5:0/CD	c.CD

Figure 109 An Allen Bradley Counter

Count-down counters are very similar to count-up counters. And, they can actually both be used on the same counter memory location. Consider the example in Figure 110, the example input *cnt\_up* drives the count-up instruction for counter *example*. Input *cnt\_down* drives the count-down instruction for the same counter location. The preset value for a counter is stored in memory location *example* so both the count-up and count-down instruction must have the same preset. Input *reset* will reset the counter.

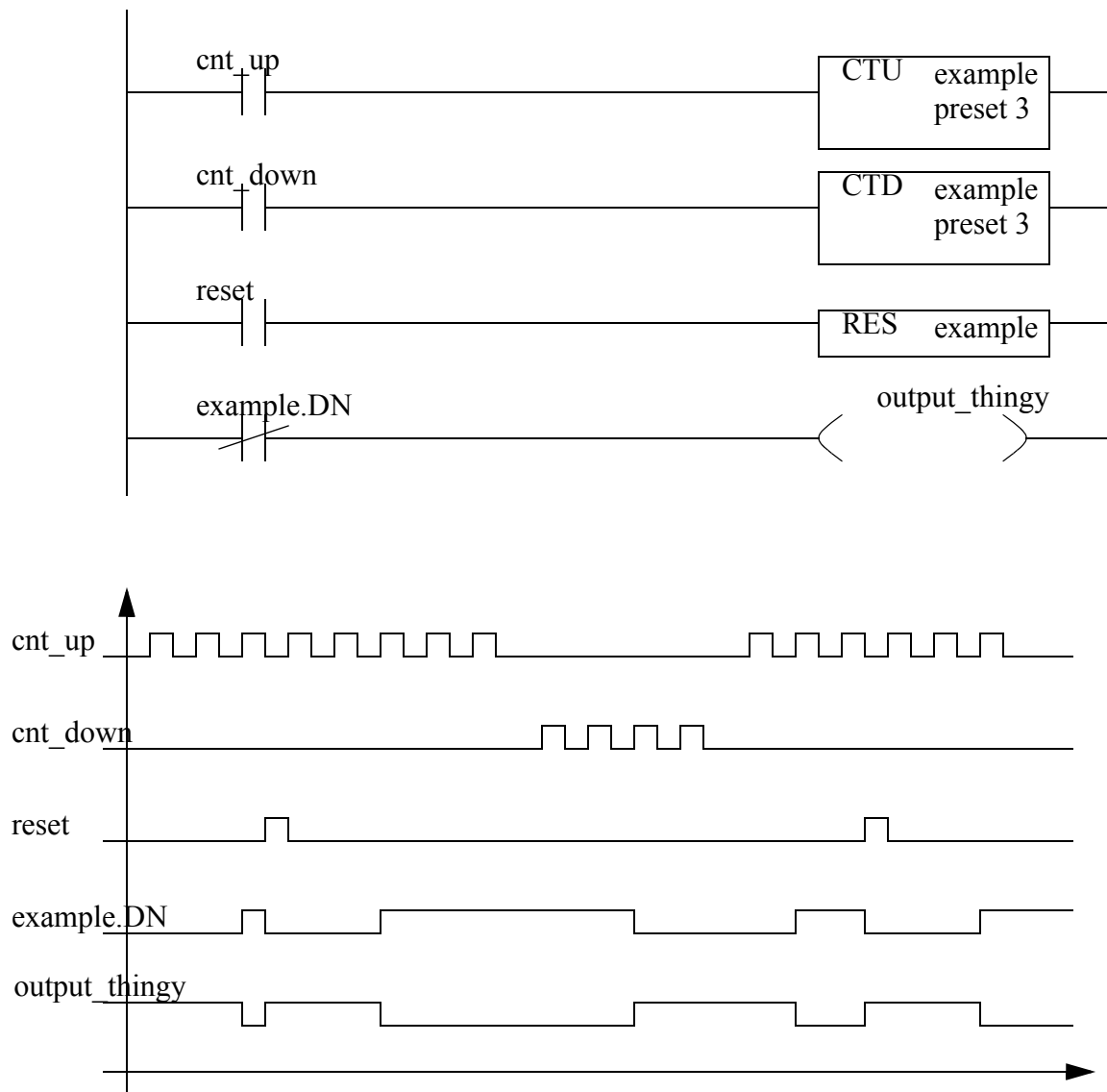


Figure 110 A Counter Example

The timing diagram in Figure 110 illustrates the operation of the counter. If we assume that the value in the accumulator starts at 0, then the positive edges on the *cnt\_up* input will cause it to count up to 3 where it turns the counter *example* done bit on. It is then reset by input *reset* and the accumulator value goes to zero. Input *cnt\_up* then pulses again and causes the accumulator value to increase again, until it reaches a maximum of 5. Input *cnt\_down* then causes the accumulator value to decrease down below 3, and the counter turns off again. Input *cnt\_up* then causes it to increase, but input *reset* resets the accumulator back to zero again, and the pulses continue until 3 is reached near the end.

The program in Figure 111 is used to remove 5 out of every 10 parts from a conveyor with a pneumatic cylinder. When the part is detected both counters will increase their values by 1. When the sixth part arrives the first counter will then be done, thereby allowing the pneumatic cylinder to actuate for any part after the fifth. The second counter will continue until the eleventh part is detected and then both of the counters will be reset.



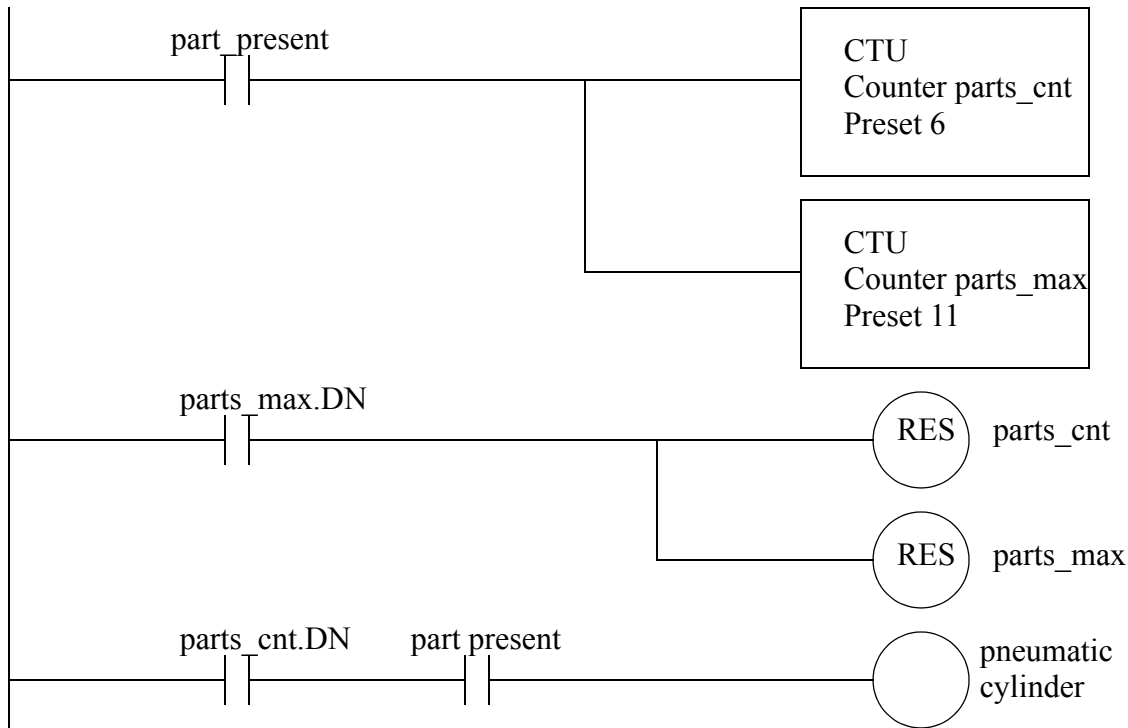
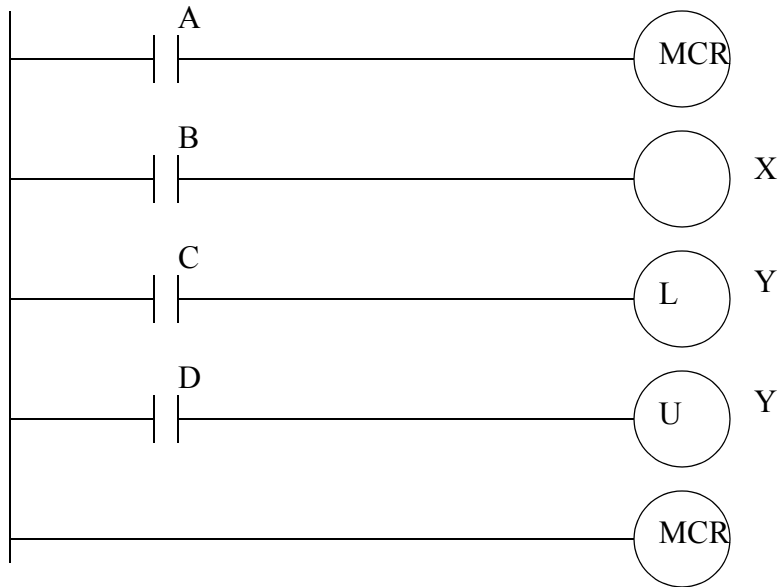


Figure 111 A Counter Example

## 9.5 MASTER CONTROL RELAYS (MCRs)

In an electrical control system a Master Control Relay (MCR) is used to shut down a section of an electrical system, as shown earlier in the electrical wiring chapter. This concept has been implemented in ladder logic also. A section of ladder logic can be put between two lines containing MCR's. When the first MCR coil is active, all of the intermediate ladder logic is executed up to the second line with an MCR coil. When the first MCR coil is inactive, the ladder logic is still examined, but all of the outputs are forced off.

Consider the example in Figure 112. If  $A$  is true, then the ladder logic after will be executed as normal. If  $A$  is false the following ladder logic will be examined, but all of the outputs will be forced off. The second MCR function appears on a line by itself and marks the end of the MCR block. After the second MCR the program execution returns to normal. While  $A$  is true,  $X$  will equal  $B$ , and  $Y$  can be turned on by  $C$ , and off by  $D$ . But, if  $A$  becomes false  $X$  will be forced off, and  $Y$  will be left in its last state. Using MCR blocks to remove sections of programs will not increase the speed of program execution significantly because the logic is still examined.



Note: If a normal input is used inside an MCR block it will be forced off. If the output is also used in other MCR blocks the last one will be forced off. The MCR is designed to fully stop an entire section of ladder logic, and is best used this way in ladder logic designs.

*Figure 112* MCR Instructions

If the MCR block contained another function, such as a TON timer, turning off the MCR block would force the timer off. As a general rule normal outputs should be outside MCR blocks, unless they must be forced off when the MCR block is off.

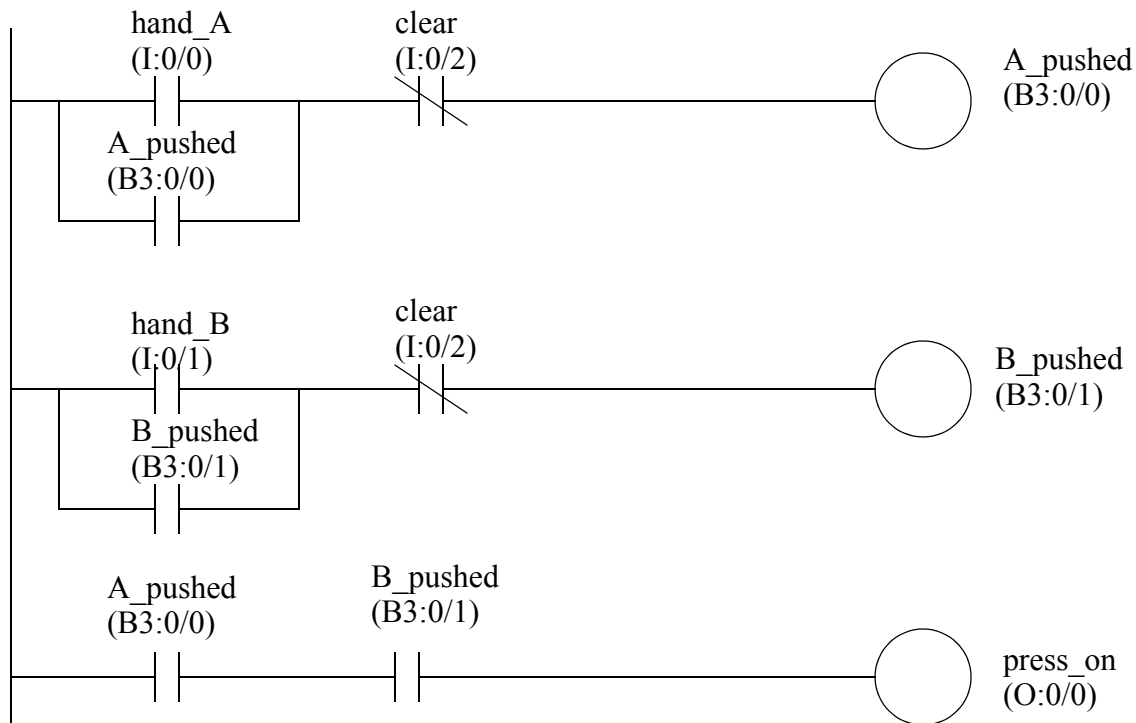
## 9.6 INTERNAL BITS

Simple programs can use inputs to set outputs. More complex programs also use internal memory locations that are not inputs or outputs. These Boolean memory locations are sometimes referred to as 'internal relays' or 'control relays'. Knowledgeable programmers will often refer to these as 'bit memory'. In the newer Allen Bradley PLCs these can be defined as variables with the type 'BOOL'.

The programmer is free to use these memory locations however they see fit.

NOTE: In the older Allen Bradley PLCs these addresses begin with 'B3' by default. The first bit in memory is 'B3:0/0', where the first zero represents the first 16 bit word, and the second zero represents the first bit in the word. The sequence of bits is shown to the right.	bit number	memory location	bit number	memory location
	0	B3:0/0	18	B3:1/2
	1	B3:0/1	19	B3:1/3
	2	B3:0/2	20	B3:1/4
	3	B3:0/3	21	B3:1/5
	4	B3:0/4	22	B3:1/6
	5	B3:0/5	23	B3:1/7
	6	B3:0/6	24	B3:1/8
	7	B3:0/7	25	B3:1/9
	8	B3:0/8	26	B3:1/10
	9	B3:0/9	27	B3:1/11
	10	B3:0/10	28	B3:1/12
	11	B3:0/11	29	B3:1/13
	12	B3:0/12	30	B3:1/14
	13	B3:0/13	31	B3:1/15
	14	B3:0/14	32	B3:2/0
	15	B3:0/15	33	B3:2/1
	16	B3:1/0	34	B3:2/2
	17	B3:1/1	etc...	etc...

An example of bit memory usage is shown in Figure 113. The first ladder logic rung will turn on the internal memory bit 'A\_pushed' (e.g., B3:0/0) when input 'hand\_A' is activated, and input 'clear' is off. (Notice that the Boolean memory is being used as both an input and output.) The second line of ladder logic similar. In this case when both inputs have been activated, the output 'press on' is active.



*Figure 113* An example using bit memory (older notations are in parentheses)

Bit memory was presented briefly here because it is important for design techniques in the following chapters, but it will be presented in greater depth after that.

## 9.7 DESIGN CASES

The following design cases are presented to help emphasize the principles presented in this chapter. I suggest that you try to develop the ladder logic before looking at the provided solutions.

### 9.7.1 Basic Counters And Timers

**Problem:** Develop the ladder logic that will turn on an output light, 15 seconds after switch *A* has been turned on.

Solution:

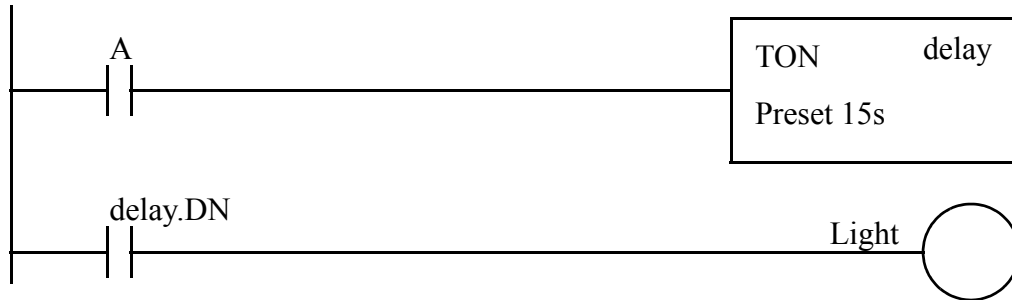


Figure 114 A Simple Timer Example

Problem: Develop the ladder logic that will turn on a light, after switch *A* has been closed 10 times. Push button *B* will reset the counters.

Solution:

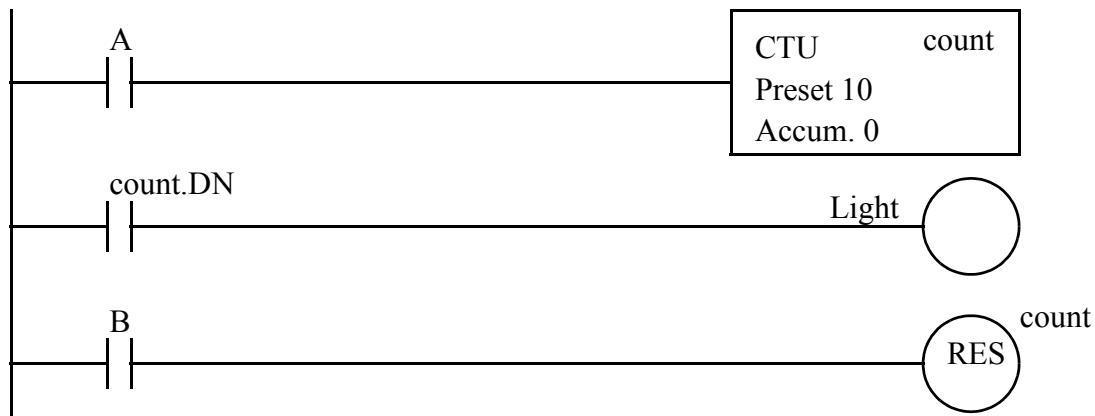


Figure 115 A Simple Counter Example

### 9.7.2 More Timers And Counters

Problem: Develop a program that will latch on an output *B* 20 seconds after input *A* has been turned on. After *A* is pushed, there will be a 10 second delay until *A* can have any effect again. After *A* has been pushed 3 times, *B* will be turned off.

Solution:

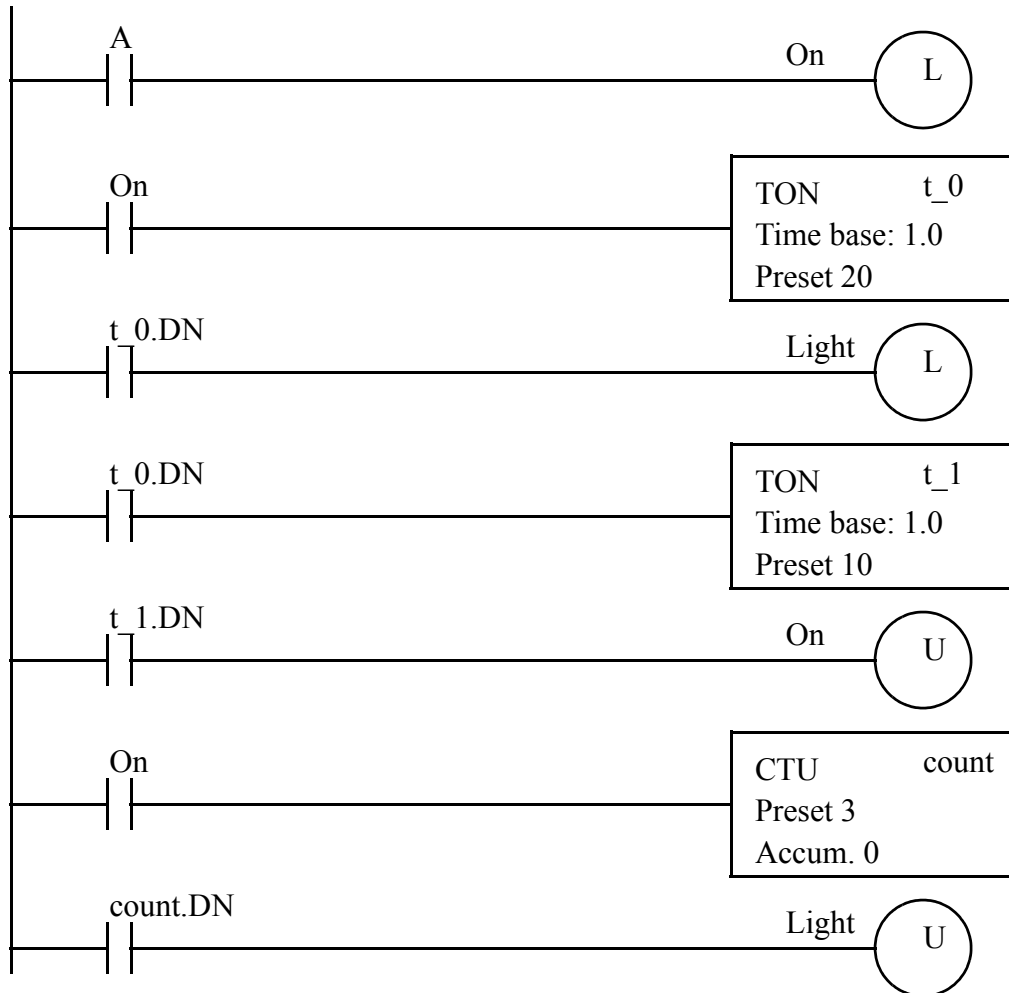
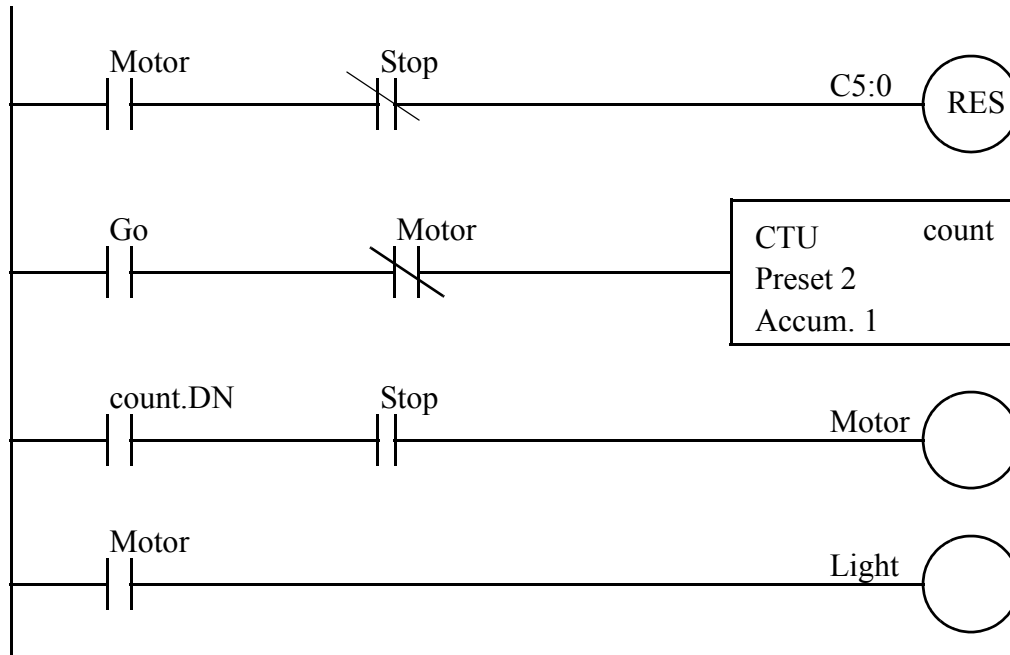


Figure 116 A More Complex Timer Counter Example

### 9.7.3 Deadman Switch

Problem: A motor will be controlled by two switches. The *Go* switch will start the motor and the *Stop* switch will stop it. If the *Stop* switch was used to stop the motor, the *Go* switch must be thrown twice to start the *motor*. When the *motor* is active a *light* should be turned on. The *Stop* switch will be wired as normally closed.

Solution:



Consider:

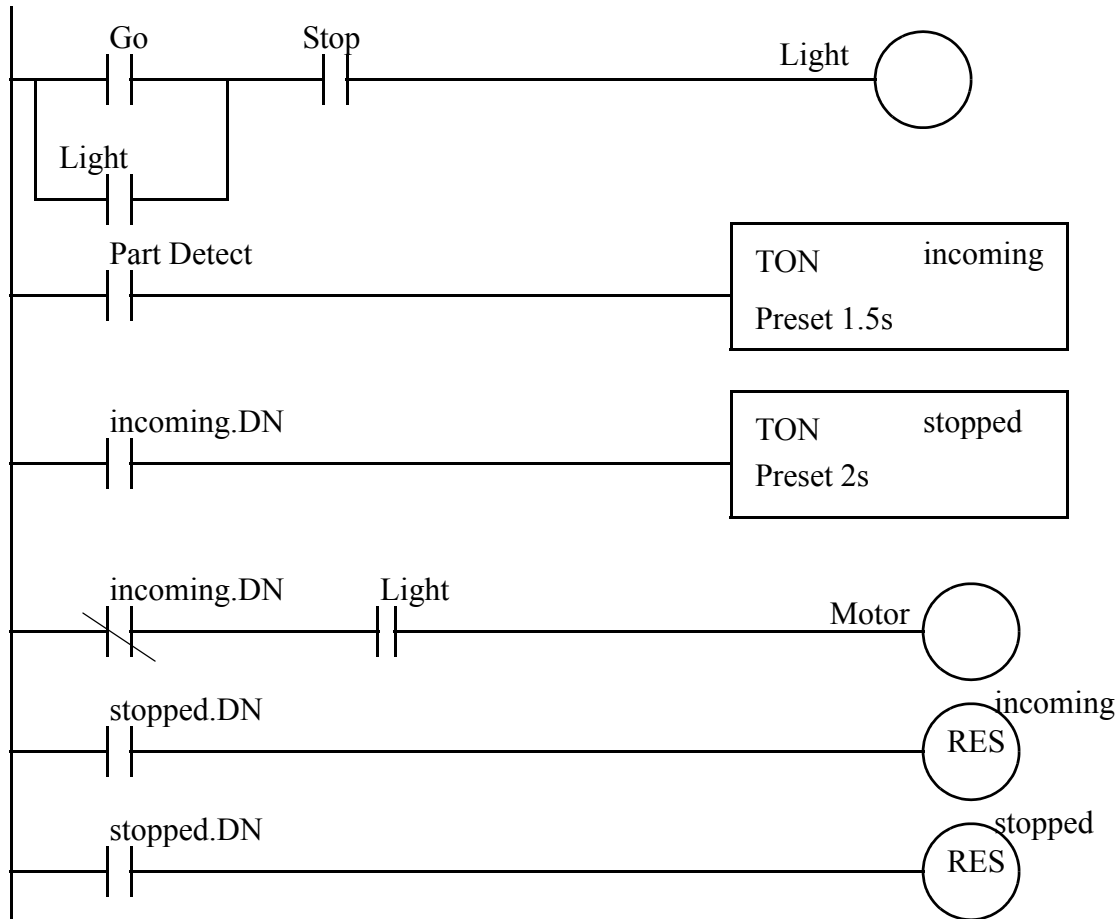
What will happen if stop is pushed and the motor is not running?

Figure 117 A Motor Starter Example

### 9.7.4 Conveyor

Problem: A conveyor is run by switching on or off a motor. We are positioning parts on the conveyor with an optical detector. When the optical sensor goes on, we want to wait 1.5 seconds, and then stop the conveyor. After a delay of 2 seconds the conveyor will start again. We need to use a start and stop button - a light should be on when the system is active.

Solution:



Consider: What is assumed about part arrival and departure?

Figure 118 A Conveyor Controller Example

### 9.7.5 Accept/Reject Sorting

Problem: For the conveyor in the last case we will add a sorting system. Gages have been attached that indicate good or bad. If the part is good, it continues on. If the part is bad, we do not want to delay for 2 seconds, but instead actuate a pneumatic cylinder.



Solution:

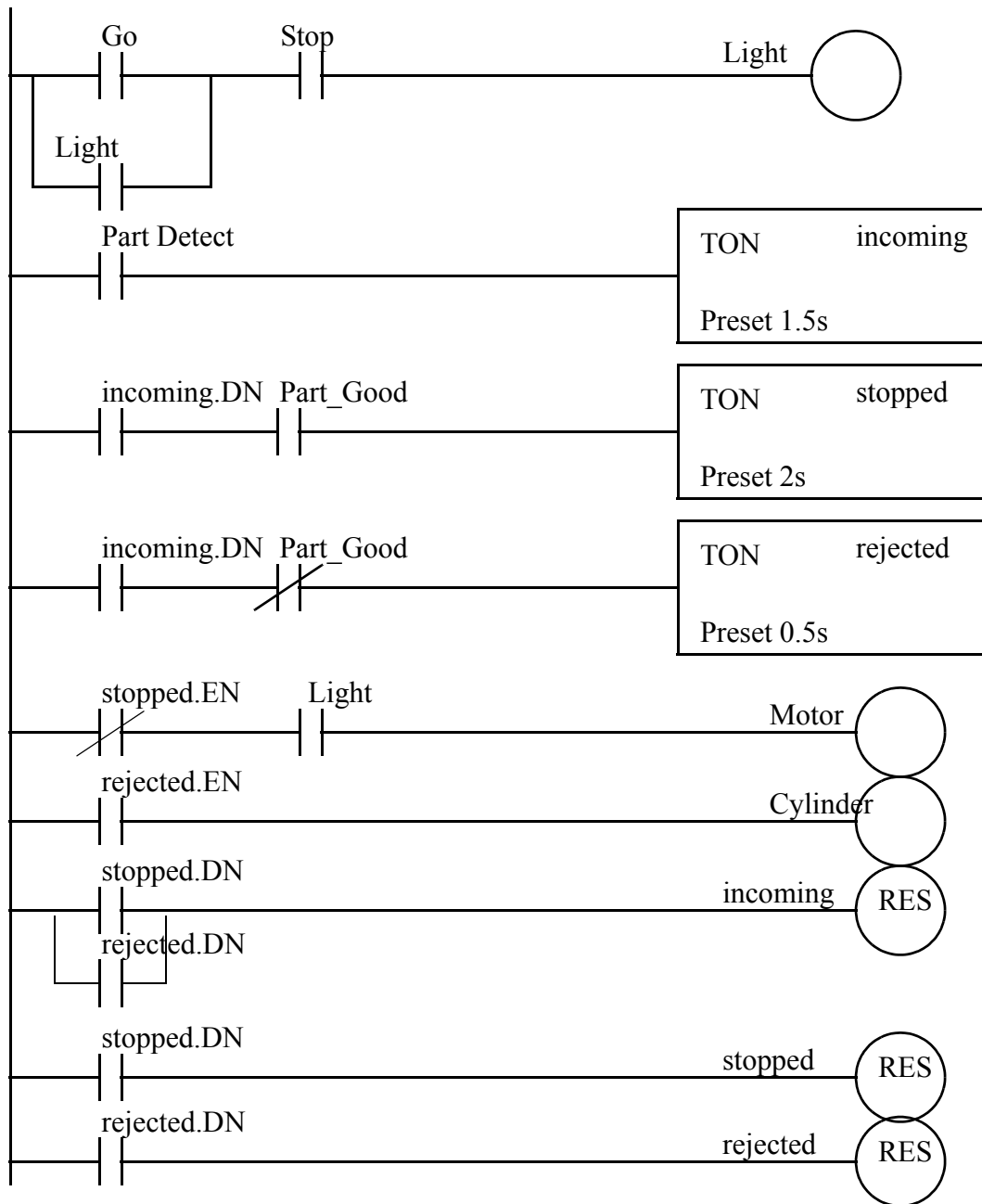


Figure 119 A Conveyor Sorting Example

### 9.7.6 Shear Press

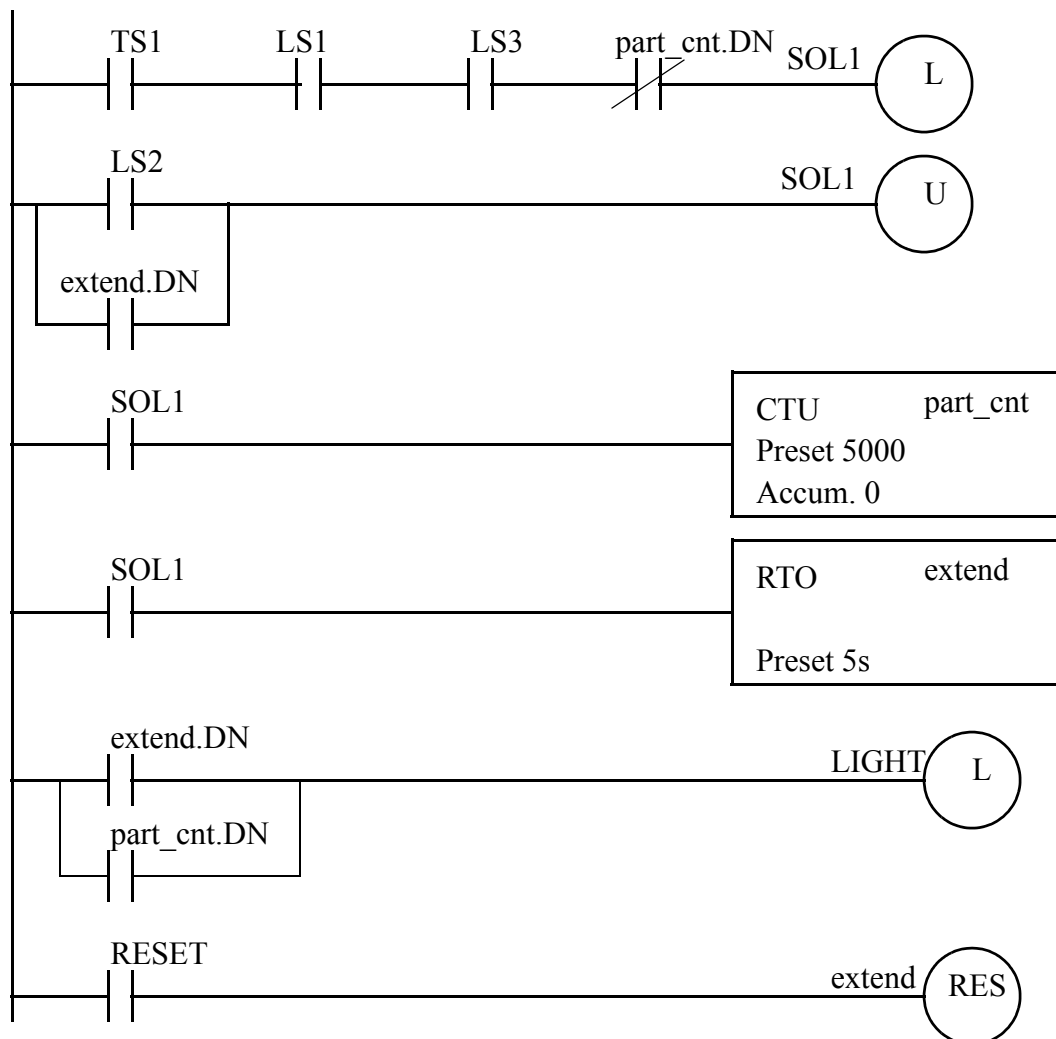
Problem: The basic requirements are,

1. A toggle start switch (TS1) and a limit switch on a safety gate (LS1) must both be on before a solenoid (SOL1) can be energized to extend a stamping cylinder to the top of a part.
2. While the stamping solenoid is energized, it must remain energized until a limit switch (LS2)

is activated. This second limit switch indicates the end of a stroke. At this point the solenoid should be de-energized, thus retracting the cylinder.

3. When the cylinder is fully retracted a limit switch (LS3) is activated. The cycle may not begin again until this limit switch is active.
4. A cycle counter should also be included to allow counts of parts produced. When this value exceeds 5000 the machine should shut down and a light lit up.
5. A safety check should be included. If the cylinder solenoid has been on for more than 5 seconds, it suggests that the cylinder is jammed or the machine has a fault. If this is the case, the machine should be shut down and a maintenance light turned on.

Solution:



- what do we need to do when the machine is reset?

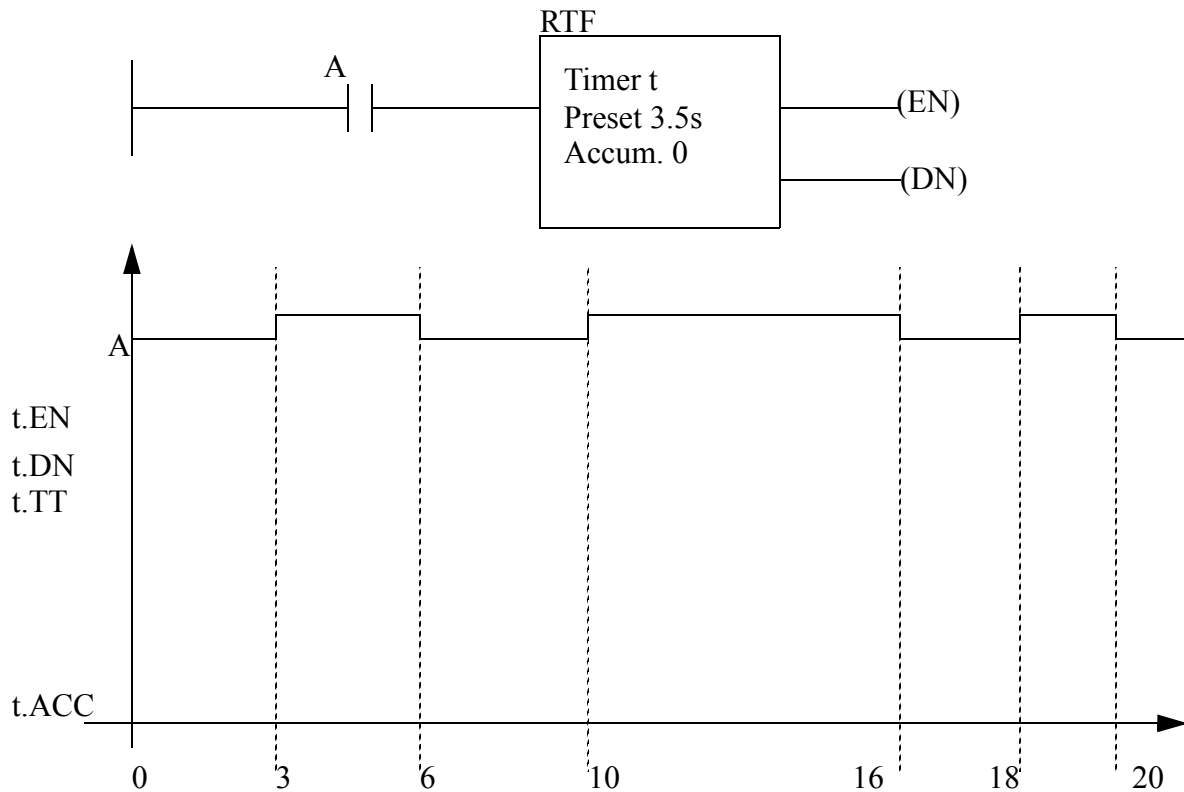
Figure 120 A Shear Press Controller Example

## 9.8 SUMMARY

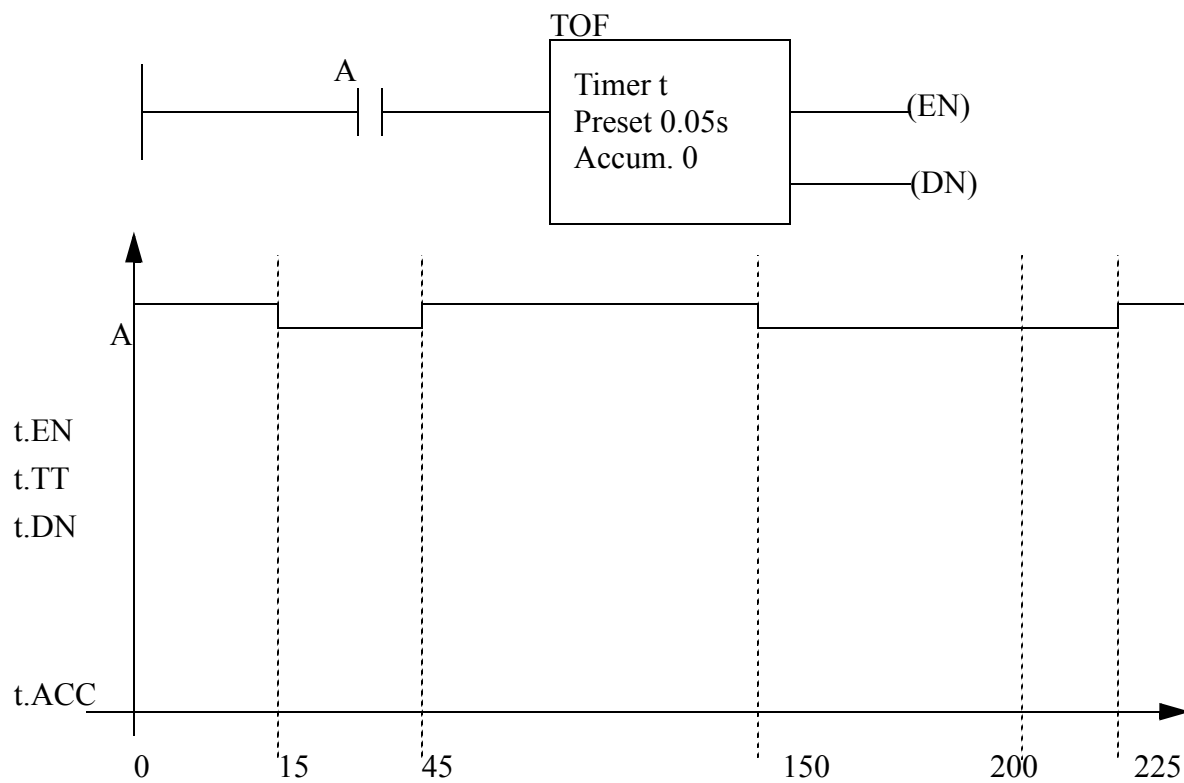
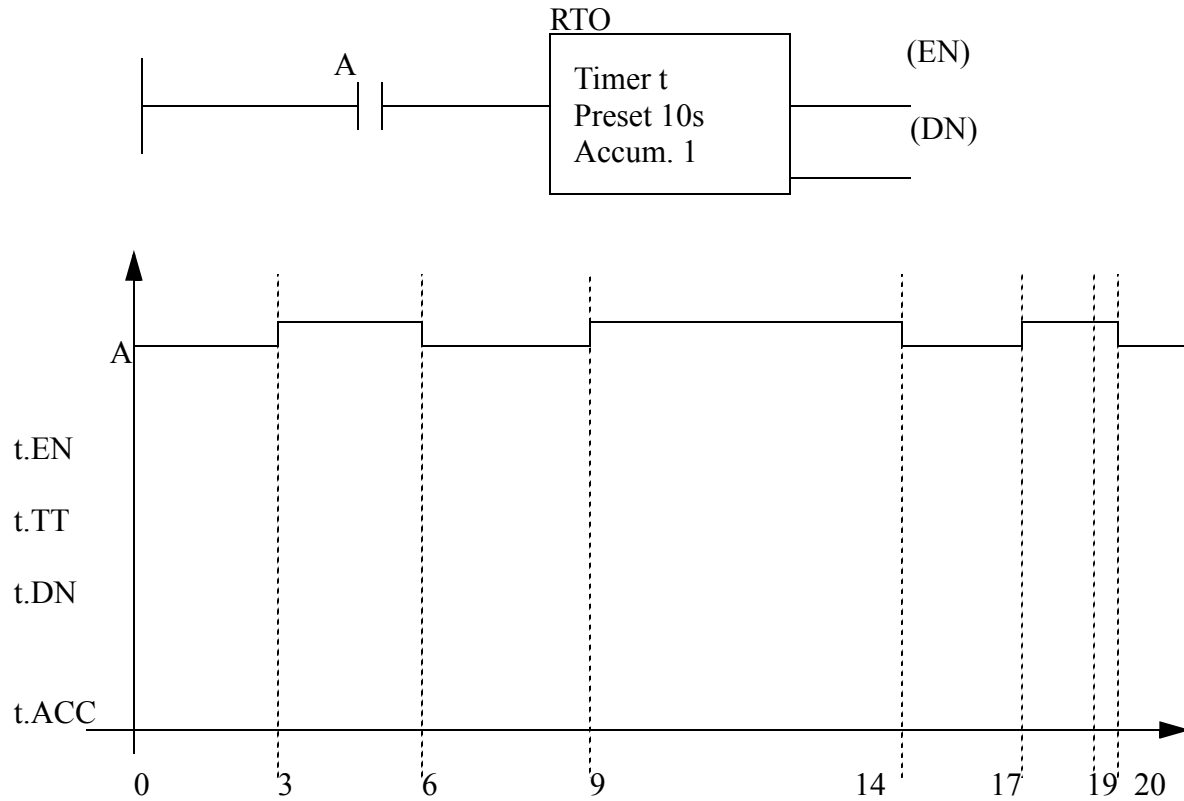
- Latch and unlatch instructions will hold outputs on, even when the power is turned off.
- Timers can delay turning on or off. Retentive timers will keep values, even when inactive.  
Resets are needed for retentive timers.
- Counters can count up or down.
- When timers and counters reach a preset limit the *DN* bit is set.
- MCRs can force off a section of ladder logic.

## 9.9 PRACTICE PROBLEMS

1. What does edge triggered mean? What is the difference between positive and negative edge triggered?
2. Are reset instructions necessary for all timers and counters?
3. What are the numerical limits for typical timers and counters?
4. If a counter goes below the bottom limit which counter bit will turn on?
5. a) Write ladder logic for a motor starter that has a start and stop button that uses latches. b) Write the same ladder logic without latches.
6. Use a timing diagram to explain how an on delay and off delay timer are different.
7. For the retentive off timer below, draw out the status bits.

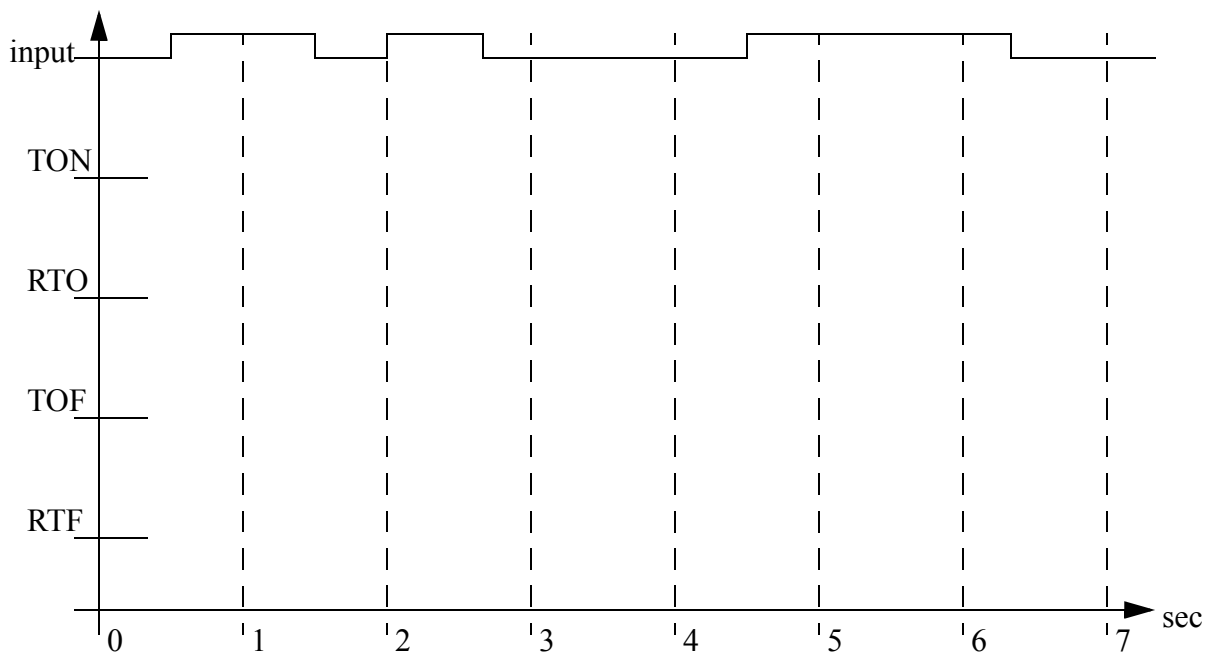


8. Complete the timing diagrams for the two timers below.



9. Given the following timing diagram, draw the done bits for all four fundamental timer types. Assume all

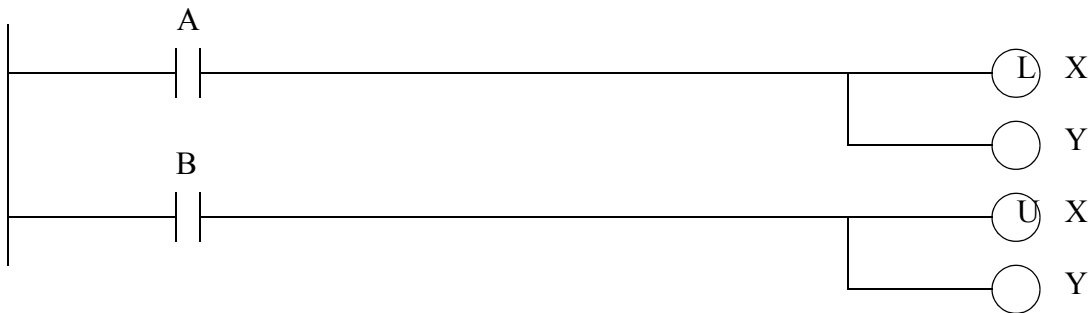
start with an accumulated value of zero, and have a preset of 1.5 seconds.



10. Design ladder logic that allows an RTO to behave like a TON.
11. Design ladder logic that uses a timer and counter to measure a time of 50.0 days.
12. Develop the ladder logic that will turn on an output (light), 15 seconds after switch (A) has been turned on.
13. Develop the ladder logic that will turn on a output (light), after a switch (A) has been closed 10 times. Push button (B) will reset the counters.
14. Develop a program that will latch on an output (B), 20 seconds after input (A) has been turned on. The timer will continue to cycle up to 20 seconds, and reset itself, until A has been turned off. After the third time the timer has timed to 20 seconds, B will be unlatched.
15. A motor will be connected to a PLC and controlled by two switches. The GO switch will start the motor, and the STOP switch will stop it. If the motor is going, and the GO switch is thrown, this will also stop the motor. If the STOP switch was used to stop the motor, the GO switch must be thrown twice to start the motor. When the motor is running, a light should be turned on (a small lamp will be provided).
16. In dangerous processes it is common to use two palm buttons that require a operator to use both hands to start a process (this keeps hands out of presses, etc.). To develop this there are two inputs that must be turned on within 0.25s of each other before a machine cycle may begin.
17. Design a conveyor control system that follows the design guidelines below.
  - The conveyor has an optical sensor *S1* that detects boxes entering a workcell
  - There is also an optical sensor *S2* that detects boxes leaving the workcell
  - The boxes enter the workcell on a conveyor controlled by output *C1*
  - The boxes exit the workcell on a conveyor controlled by output *C2*
  - The controller must keep a running count of boxes using the entry and exit sensors
  - If there are more than five boxes in the workcell the entry conveyor will stop
  - If there are no boxes in the workcell the exit conveyor will be turned off
  - If the entry conveyor has been stopped for more than 30 seconds the count will be reset to

zero, assuming that the boxes in the workcell were scrapped.

18. Write a ladder logic program that does what is described below.
- When button *A* is pushed, a light will flash for 5 seconds.
  - The flashing light will be on for 0.25 sec and off for 0.75 sec.
  - If button *A* has been pushed 5 times the light will not flash until the system is reset.
  - The system can be reset by pressing button *B*
19. Write a program that will turn on a flashing light for the first 15 seconds after a PLC is turned on. The light should flash for half a second on and half a second off.
20. A buffer can hold up to 10 parts. Parts enter the buffer on a conveyor controller by output *conveyor*. As parts arrive they trigger an input sensor *enter*. When a part is removed from the buffer they trigger the *exit* sensor. Write a program to stop the conveyor when the buffer is full, and restart it when there are fewer than 10 parts in the buffer. As normal the system should also include a start and stop button.
21. What is wrong with the following ladder logic? What will happen if it is used?

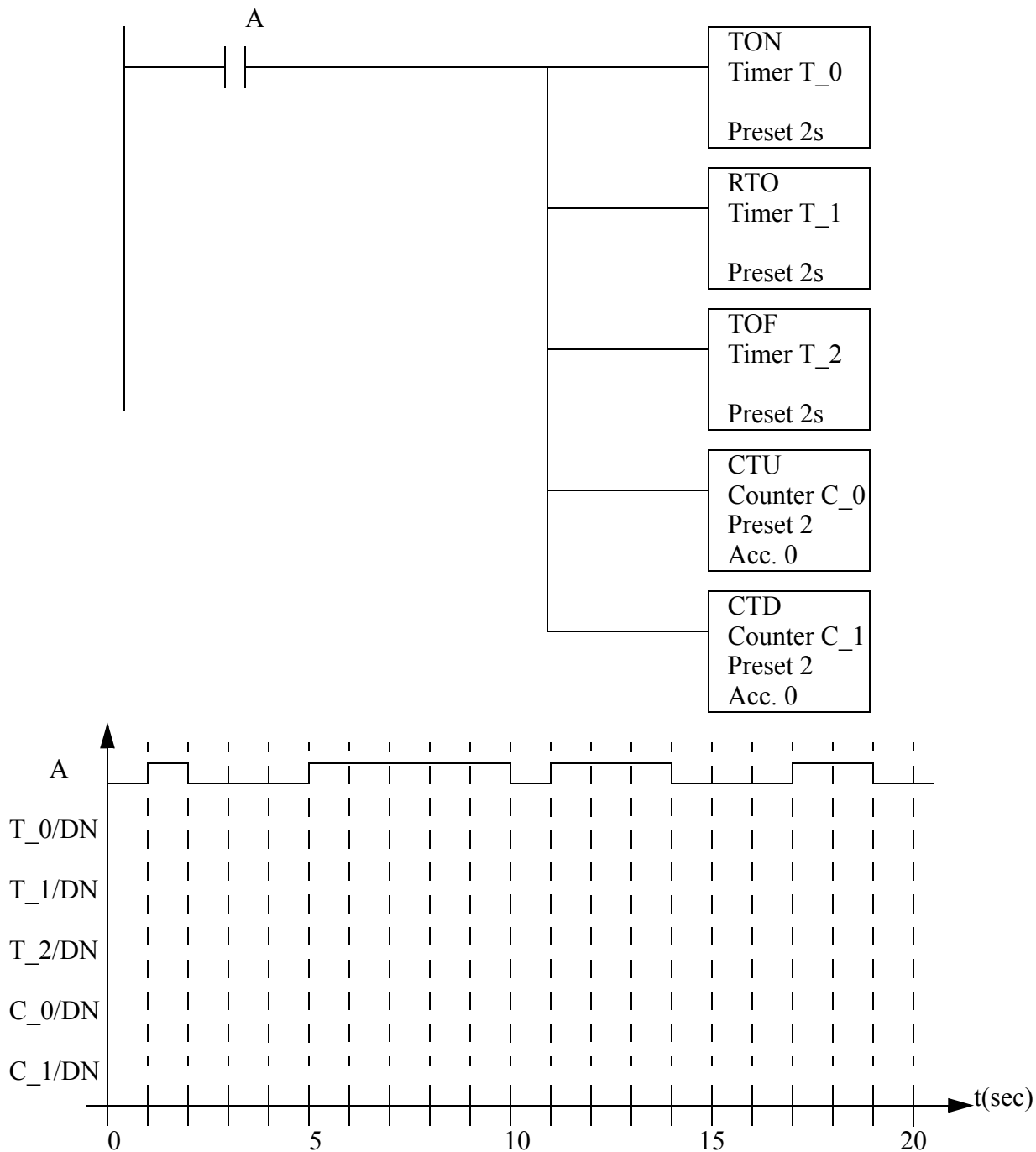


22. We are using a pneumatic cylinder in a process. The cylinder can become stuck, and we need to detect this. Proximity sensors are added to both endpoints of the cylinder's travel to indicate when it has reached the end of motion. If the cylinder takes more than 2 seconds to complete a motion this will indicate a problem. When this occurs the machine should be shut down and a light turned on. Develop ladder logic that will cycle the cylinder in and out repeatedly, and watch for failure.

## 9.10 ASSIGNMENT PROBLEMS

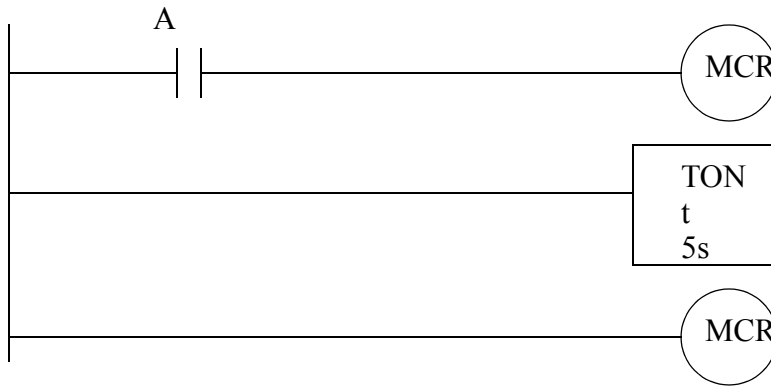
1. Draw the timer and counter done bits for the ladder logic below. Assume that the accumulators of all the tim-

ers and counters are reset to begin with.



2. Write a ladder logic program that will count the number of parts in a buffer. As parts arrive they activate input *A*. As parts leave they will activate input *B*. If the number of parts is less than 8 then a conveyor motor, output *C*, will be turned on.

3. Explain what would happen in the following program when A is on or off.



4. Write a simple program that will use one timer to flash a light. The light should be on for 1.0 seconds and off for 0.5 seconds. Do not include start or stop buttons.
5. We are developing a safety system (using a PLC-5) for a large industrial press. The press is activated by turning on the compressor power relay (R, connected to O:013/05). After R has been on for 30 seconds the press can be activated to move (P connected to O:013/06). The delay is needed for pressure to build up. After the press has been activated (with P) the system must be shut down (R and P off), and then the cycle may begin again. For safety, there is a sensor that detects when a worker is inside the press (S, connected to I:011/02), which must be off before the press can be activated. There is also a button that must be pushed 5 times (B, connected to I:011/01) before the press cycle can begin. If at any time the worker enters the press (and S becomes active) the press will be shut down (P and R turned off). Develop the ladder logic. State all assumptions, and show all work.
6. Write a program that only uses one timer. When an input A is turned on a light will be on for 10 seconds. After that it will be off for two seconds, and then again on for 5 seconds. After that the light will not turn on again until the input A is turned off.
7. A new printing station will add a logo to parts as they travel along an assembly line. When a part arrives a 'part' sensor will detect it. After this the 'clamp' output is turned on for 10 seconds to hold the part during the operation. For the first 2 seconds the part is being held a 'spray' output will be turned on to apply the thermoset ink. For the last 8 seconds a 'heat' output will be turned on to cure the ink. After this the part is released and allowed to continue along the line. Write the ladder logic for this process.
8. Write a ladder logic program. that will turn on an output Q five seconds after an input A is turned on. If input B is on the delay will be eight seconds. YOU MAY ONLY USE ONE TIMER.