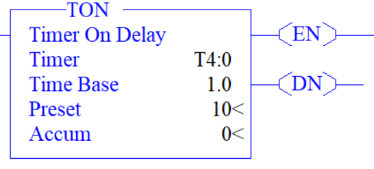
Title: **Timers** Handout: 2

Course: Introduction to Automation Unit: Introduction of PLC CLO: 4

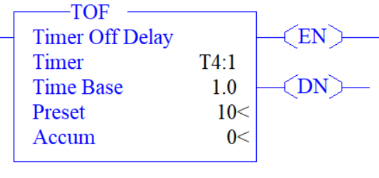
**Objectives**

1. Student shall develop a knowledge of the Timer On-Delay (TON) output instruction.
2. Student shall develop a knowledge of the Timer Off-Delay (TOF) output instruction.

**Theory**

Timer On-Delay (TON)

This output instruction begins timing when the rung obtains rung continuity and continues until it reaches a predetermine amount of time. The target time value is stored in the preset variable. The accumulated variable stores the amount of time that has occurred since achieving rung continuity. Three bits keep track of when the timer is enable, the timer is timing and when the timer is done. When rung continuity is achieved, the enable and timer timing bits are set, and the timer increments the accumulated value by the time base until the accumulated value reaches the preset value. Once this occurs, timing is complete, the timer timing bit is reset, and the done bit is set, and the accumulative value shall equal the preset value. Even though the rung still has logical continuity the timer does not continue to time. When rung continuity is lost, the accumulative value is set to 0, the enable and done bits are rest.

Timer Off-Delay (TOF)

This output instruction is similar to the Timer On-Delay but begins timing when the rung loses rung continuity. When rung continuity is obtain, the enable bit is set, the timer timing bit is reset, and the done bit is set. When rung continuity is lost, the enable bit is reset, the timer timing bit is set, and the done bit remains set. The timer increments the accumulated value by the time base until the accumulated value reaches the preset value. Once this occurs, timing is complete, the timer timing and done bits are reset, and the accumulative value shall equal the preset value. Even though the rung still has no logical continuity the timer does not continue to time. When rung continuity is gained, the accumulative value is set to 0 and the enable and done bits are set.

Both timers are non-retentive timers meaning that they don’t remember a partial time sequence. If during timing the instruction loses its qualify timing status the accumulated value is set to 0. Timer variables (T4 table) use three words per element.

*Accum* = Accumulated time (one word)

*Preset* = Target time value (one word)

*Time Base, done bit, timer timing bit, enable bit* (components in one word)

13 = DN (done bit)

14 = TT (timer timing bit)

15 = EN (enable bit)

**Graphics**

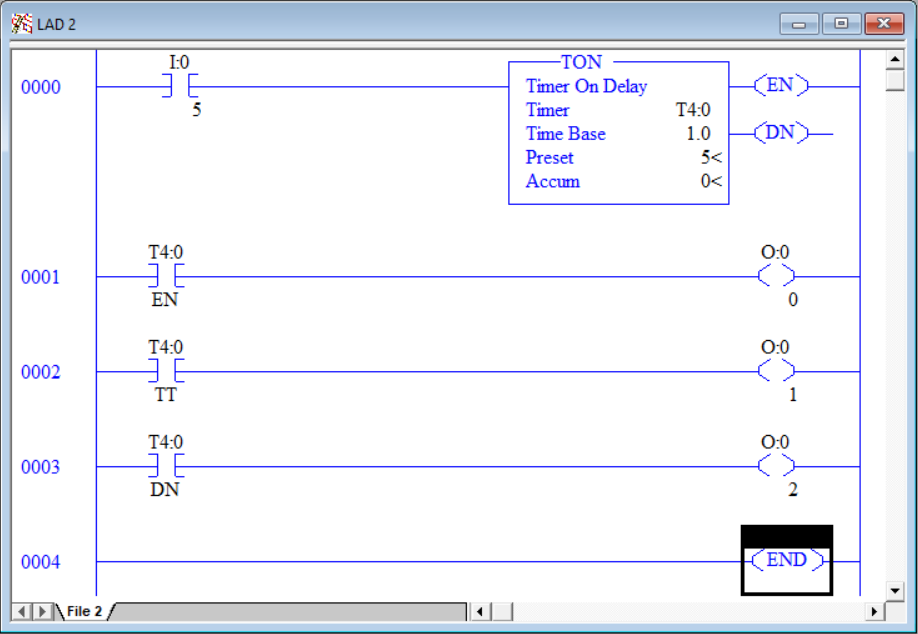
|  |  |
| --- | --- |
| TON | TOF |
|  |  |

**Devices**

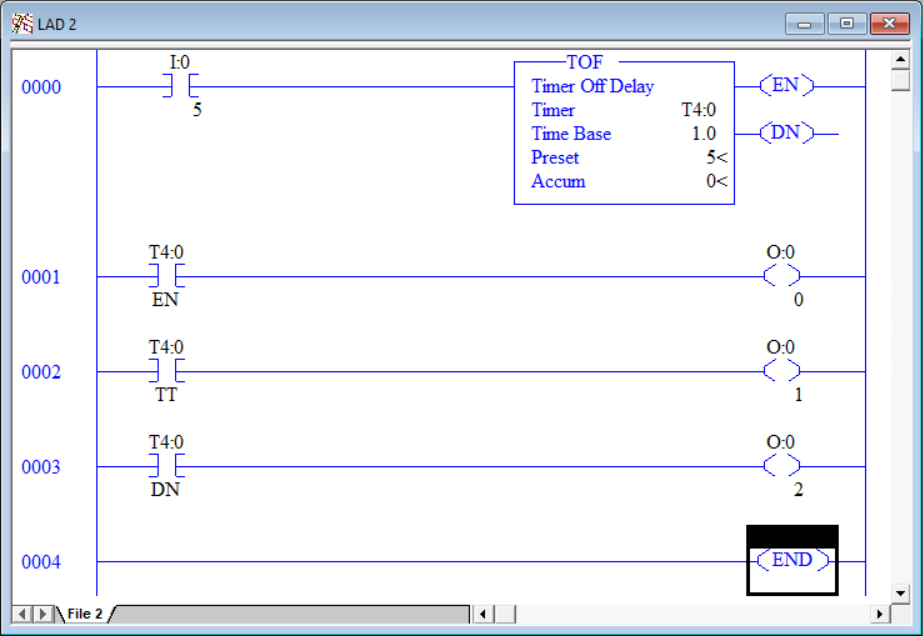
|  |  |  |
| --- | --- | --- |
| Inputs | | |
| *Device* | *Description* | *Symbol* |
| Two-Position Selector Switch | Timer Selector | TMR\_SEL |
| Outputs | | |
| *Device* | *Description* | *Symbol* |
| Green Pilot Light | Timer Enabled | TMR\_EN |
| Yellow Pilot Light | Timer Timing | TMR\_TT |
| Red Pilot Light | Timer Done | TMR\_DN |
| Blue Pilot Light | Most Significant Bit | MS\_BIT |

**Instructions**

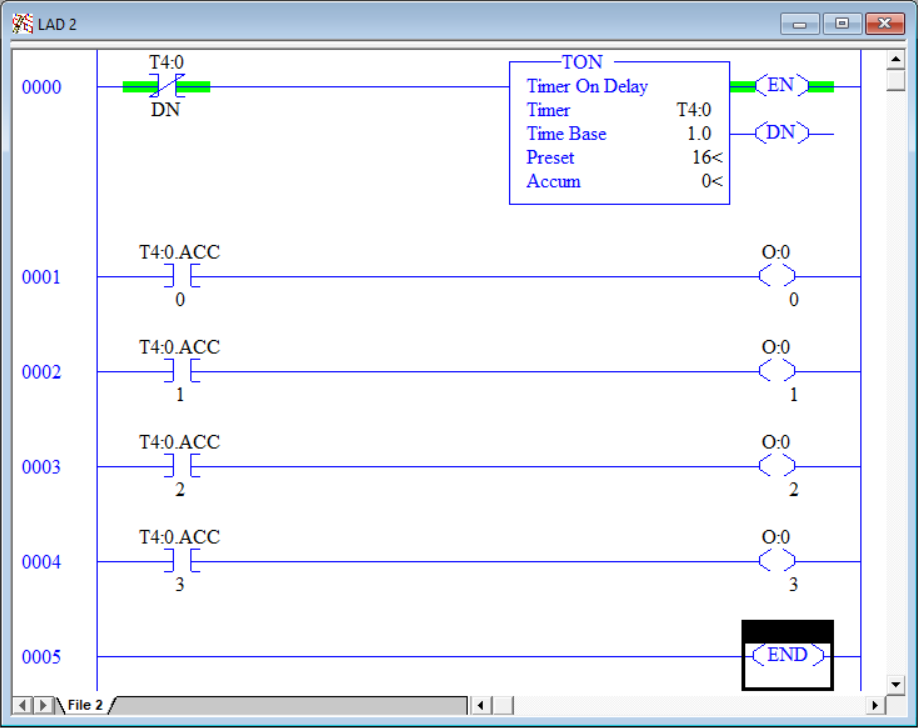
Program the logic should below.



1. Place the two-position selector switch in the “B” (down) position.
2. Download the program and observe its status online.
3. At this point, the green, yellow and red pilot lights shall be off.
4. Place the two-position selector switch in the “B” (down) position. Notice the enable (EN) bit (green light) illuminates immediately to indicate that the timer has logical continuity. Also, the timer timing (TT) bit (yellow light) illuminates while the timer is incrementing the *Accum* variable. Once the timer’s *Accum* = *Preset,* several things happen.
   1. The enable (EN) bit remains 1 since we still have rung continuity. Green light on.
   2. The timer timing (TT) bit is set to 0 and the yellow light turns off.
   3. The done (DN) bit is set to 1 and the red light illuminates.
   4. The timer stops incrementing *Accum* even though the rung still has continuity.
5. Place the two-position selector switch in the “B” (down) position.
6. All the bits are reset to 0 and all three lights go off. The *Accum* is set to 0. The TON instruction is a “non-retentive” timer meaning that when the instruction loses rung continuity, the timer does not retain its accumulated time.
7. Program the following logic.



1. Move the two-position selector switch to the “A” (up) position.
2. Download the program and observe its status online.
3. With the TOF instruction having rung continuity, the timer will not run.
4. Observe the status of the three lights.
   1. The enable (EN) bit is set to 1 since we have rung continuity. Green light is on.
   2. The timer timing (TT) bit is reset to 0 since the timer is not running. Yellow light is off.
   3. The done (DN) bit is set to 1 and the red light is on.
   4. The timer clears *Accum* since the rung still has continuity.
5. Move the two-position selector switch to the “B” (down) position.
6. With the TOF instruction losing rung continuity, the timer begins to run.
7. Observe the status of the three lights.
8. The enable (EN) bit is reset to 0 since we lost rung continuity. Green light is off.
9. The timer timing (TT) bit is set to 1 since the timer is running. Yellow light is on.
10. The done (DN) bit remains a 1 and the red light is on.
11. The timer increments *Accum* since the rung has lost continuity.
12. Once the timer’s *Accum* = *Preset,* several things happen.
13. The enable (EN) bit remains 0 since we still don’t have rung continuity. Green light off.
14. The timer timing (TT) bit is set to 0 and the yellow light turns off.
15. The done (DN) bit is set to 0 and the red light turns off.
16. The timer stops incrementing *Accum* even though the rung still has continuity.
17. Program the following logic. Rungs 0001-0004 reference bits inside the timers Accumulator word. To address these instructions, type T4:0.ACC/n where n = the bit number.



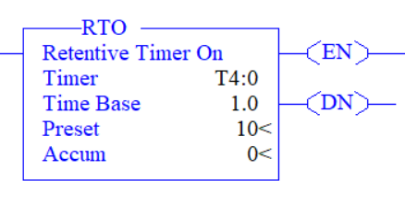
1. This configuration is a self-resetting timer. When the timer completes, the done (DN) bit is set which in turn disrupts rung continuity to its own timer instruction. This timer will time repeatedly for 16 seconds then reset and restart counting.
2. Every decimal integer can be broken down into its binary equivalent. Knowing how to convert decimal to binary is key to decompacting information within a PLC.

*Accum* Integer Value

Bit Number

|  |  |  |  |
| --- | --- | --- | --- |
| Blue | Red | Yellow | Green |
| 3 | 2 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 2 |
| 0 | 0 | 1 | 1 | 3 |
| 0 | 1 | 0 | 0 | 4 |
| 0 | 1 | 0 | 1 | 5 |
| 0 | 1 | 1 | 0 | 6 |
| 0 | 1 | 1 | 1 | 7 |
| 1 | 0 | 0 | 0 | 8 |
| 1 | 0 | 0 | 1 | 9 |
| 1 | 0 | 1 | 0 | 10 |
| 1 | 0 | 1 | 1 | 11 |
| 1 | 1 | 0 | 0 | 12 |
| 1 | 1 | 0 | 1 | 13 |
| 1 | 1 | 1 | 0 | 14 |
| 1 | 1 | 1 | 1 | 15 |

**Theory**

Retentive Timer On-Delay (RTO)

This output instruction begins timing when the rung obtains rung continuity and continues until it reaches a predetermine amount of time. The target time value is stored in the preset variable. The accumulated variable stores the amount of time that has occurred since achieving rung continuity. When rung continuity is lost the accumulative time is retained. Three bits keep track of when the timer is enable, the timer is timing and when the timer is done. When rung continuity is achieved, the enable and timer timing bits are set, and the timer increments the accumulated value by the time base. If rung continuity is lost before the accumulated value reaches the preset value, the accumulated value is retained and the enable and timer timing bits are reset. If logical continuity is regained, the accumulated value will be incremented from its retained value. The timer will continue to time until the accumulated value reaches the preset value. Once this occurs, timing is complete, the timer timing bit is reset, and the done bit is set, and the accumulative value will equal the preset value. Even though the rung still has logical continuity the timer does not continue to time. Resetting the timer is most often down through the -( RES )- output function, but it is possible to move a zero into the accumulative variable to achieve the same result.

**Graphics**

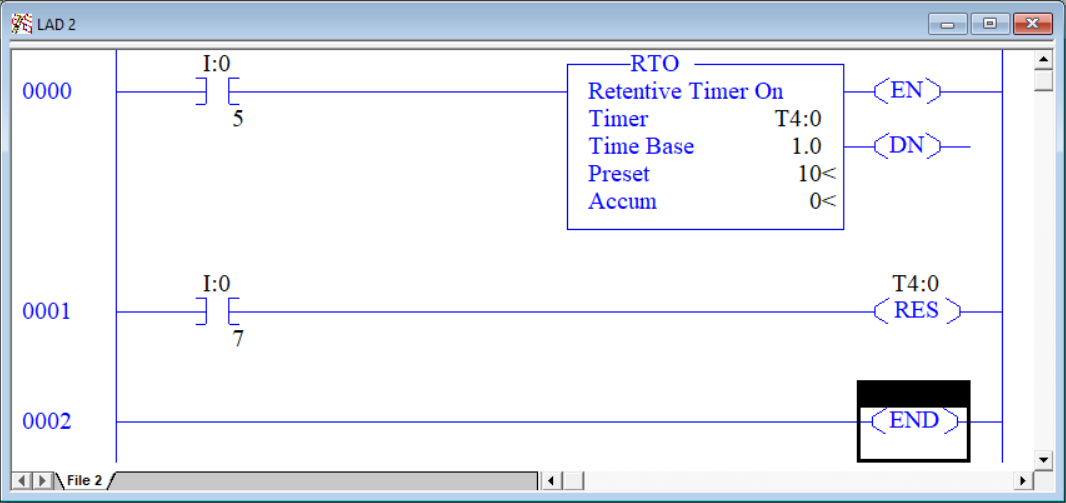


**Devices**

|  |  |  |
| --- | --- | --- |
| Inputs | | |
| *Device* | *Description* | *Symbol* |
| Two-Position Selector Switch | Timer Selector | TMR\_SEL |
| Outputs | | |
| *Device* | *Description* | *Symbol* |
| Green Pilot Light | Timer Enabled | TMR\_EN |
| Yellow Pilot Light | Timer Timing | TMR\_TT |
| Red Pilot Light | Timer Done | TMR\_DN |

**Instructions**

Program the logic should below.



1. Place the two-position selector switch in the “B” (down) position.
2. Download the program and observe that the accumulated timer *Accum* is 0.
3. Place the two-position selector switch in the “A” (up) position for 5 seconds then place it back in the “B” position. Notice that the accumulated time was retained even though rung 0000 no longer has rung continuity.
4. Press PB2 to reset the timer.
5. Place the two-position selector switch in the “A” position for at least 10 seconds then place it back in the “B” position. Notice that the accumulated stopped at 10 and the done bit was set but losing rung continuity still did not set the accumulated value to 0.
6. Press PB2 to reset the retentive timer. Observe that the done bit is reset as well.

This page left intentionally almost blank