EET165 Lecture #11

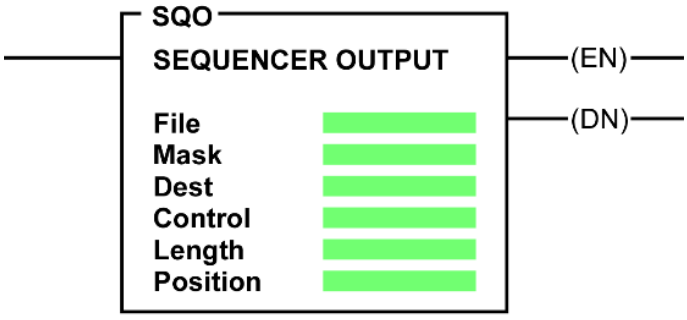
1) **Review:** Answer any questions from last week.

2) **Sequencers:** A sequencer will grab a value from a file and send it to another location. The value can be sent to an output card or to a memory location.

    a. When the value is sent to an output card, the binary version of the value is copied to the output card. Each bit is coped to one of the pins on the card.

    b. The value can also be sent to a control register in a timer or counter (such as the accumulator or preset).

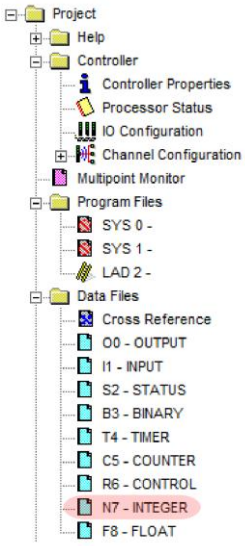
3) **SQO:** The symbol used for a sequencer is the SQO, as shown below and is found on the "File shift / Sequencer" tab. A sequencer uses the SQO symbol. Each time the input of the sequencer goes from Low to High, the next value from the file is moved to the destination. Sequencers are used in different recipes or a "state machine" such as a stop light. Sequencers are a bit more complicated because they need a lot more information to perform their task.



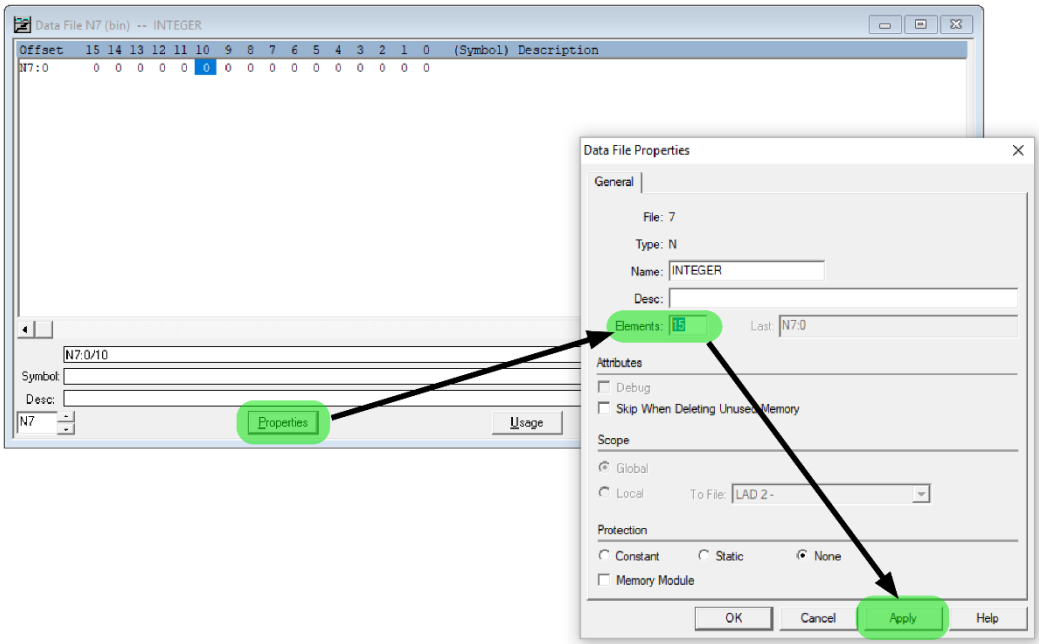
©2025, Dr. Dan Wedding

4) **Data file:** A sequencer needs a data file to work. A data file is just a series of values stored in the N7 (or F8) file. In this course we will only use N7. In order to create a data file you need to perform the following steps.

   a. **Open N7 or F8:** First, double click on the N7 folder on the left side of the screen.

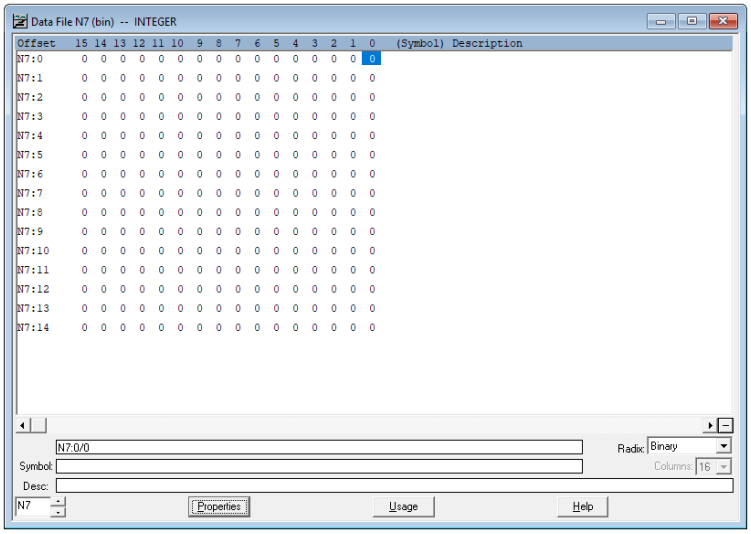


   b. **Add Lines:** When you open the data file, you will only see one line. Press the Properties button to bring up the properties window. Add lines using the elements box and then press Apply.

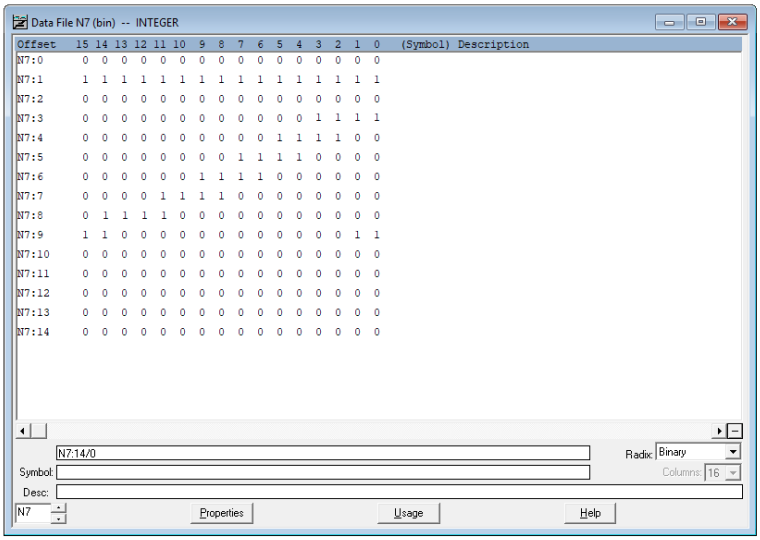


©2025, Dr. Dan Wedding

c. **<u>Updated File:</u>** Once the N7 data file is updated it will look like the image below. The values are currently shown in Binary, and that is useful if you are trying to make a pattern on an output. If you are going to load values into a preset or an accumulator, you might want to change the Radix (how the numbers are shown) to decimal.

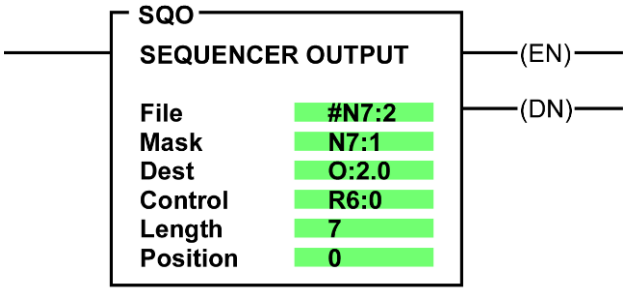


d. **<u>Add Data:</u>** Update the data file to the desired data.

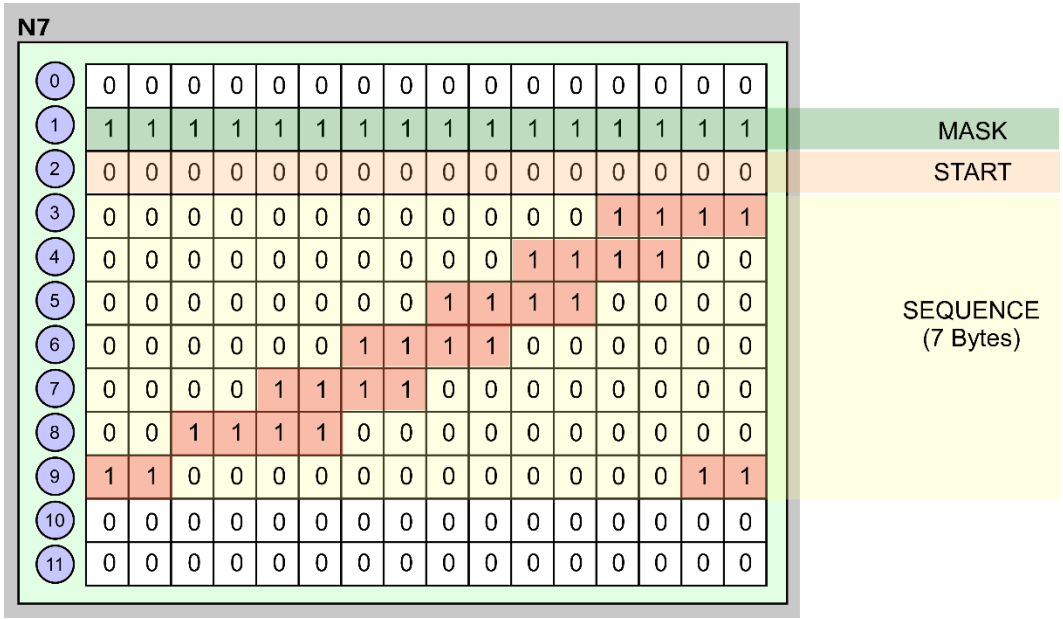


©2025, Dr. Dan Wedding

5) **<u>SQO Output pattern example:</u>** In this example, the SQO will copy a pattern on the output pins.



A sequencer needs a data file to work. A data file is just a series of values stored in the N7 (or F8) file. In this example the following data file will be used. You will need to refer back to this file to understand the following example.



©2025, Dr. Dan Wedding

a. **File:** A file starts with a # sign, and if you forget to add it, the PLC will add it for you. In this example the file starts in integer file N7 at location 2 (**#N7:2**). The file location is the entry point and is NOT part of the program. Think of it as instruction 0. The sequence starts at the next location (N7:3). The Length tells the PLC how many of these words are part of the sequence. In this example the Length is 7, so the sequence is N7:3, N7:4, N7:5, N7:6, N7:7, N7:8, and N7:9, it then repeats forever.

b. **Mask:** The mask will pass bits or block bits when it is copied to the destination. When there is a 1, the bit will be passed and when there is a 0 the bit will be blocked.

For example:

File value:	0011	1010	0110	1001
Mask:	0000	1111	0000	1111
Final output:	0000	1010	0000	1001

The mask can be either a number or a location in the data file. In this example the mask is inside the integer file N7 at Location 1 (N7:1). The mask can NOT be inside the sequence. Because the file starts at N7:2, and the program is 7 words long (N7:3 – N7:9) the mask must NOT be in any of these locations. So, I put the mask in front of the sequence. In this example, the mask is all ones, so all of the value are passed and none are blocked.

c. **Dest:** The Destination is where the final output is stored. This can be inside of a file (N7, F8) or on the outputs of a card (O:2.0), remember that when an output ends in .0 the information is put on all of the values on the output.

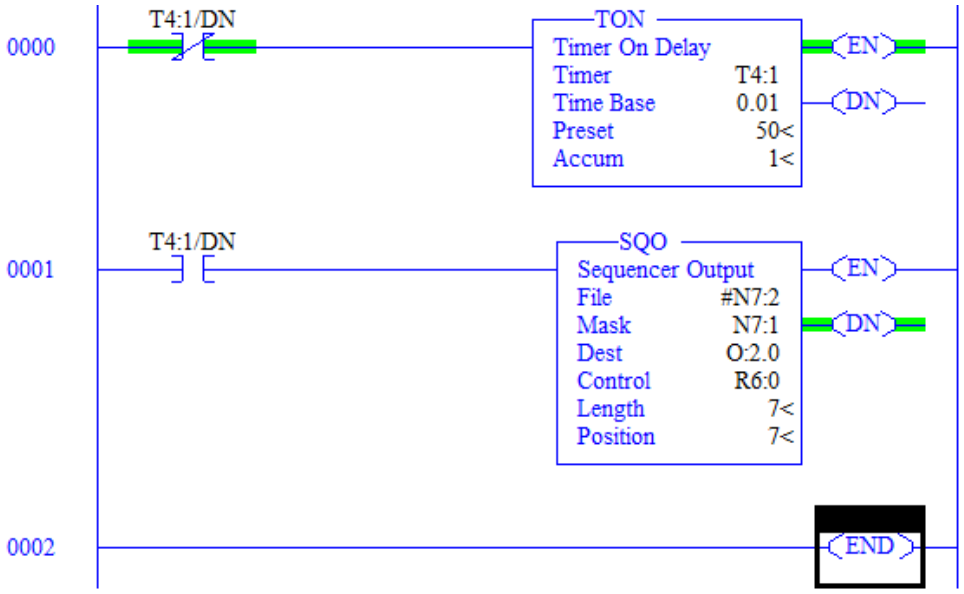
d. **Control:** Just like you need to include a timer in the timer command (T4:3) or a counter in the counter command (C5:0), you need to include a sequencer in the sequence command. The sequencer is loaded into the Control location, and it is R6 and a sequence number. In this example R6:0 is the output sequencer.

e. **Length:** The Length is how long the sequence is. In this example the length is 7 so the sequence is N7:3, N7:4, N7:5, N7:6, N7:7, N7:8, and N7:9. The mask value cannot be in any of these locations.

f. **Position:** This is simply the memory location that has been copied to the output. When it reaches the end, the sequence starts over at Location 1. In this example N7:3 is the starting location (**not N7:2**).

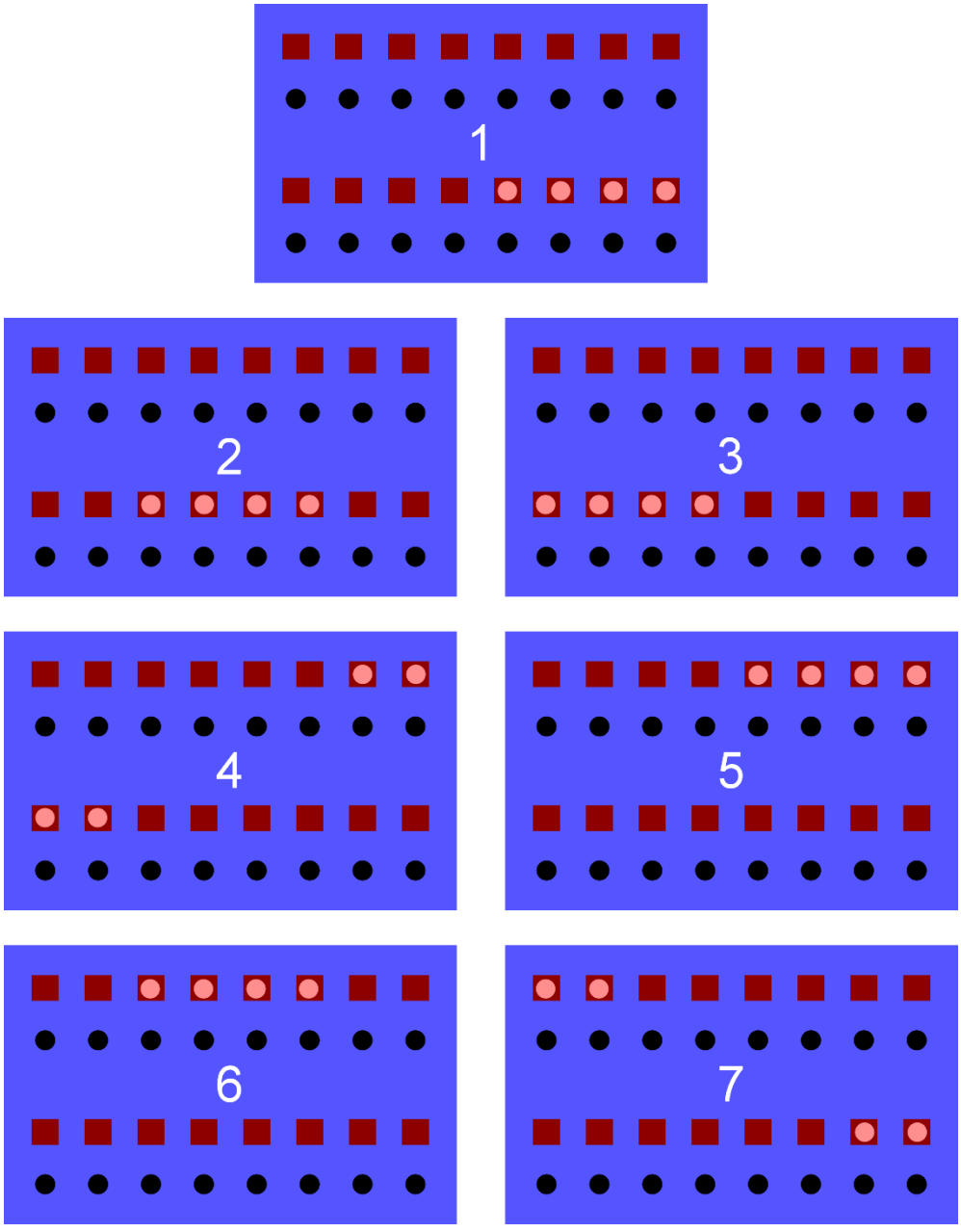
©2025, Dr. Dan Wedding

6) **Program:** Below is a program where timer is used to step through the sequence. Every half second DN goes high. When this happens, the sequence copies the next pattern on the lamps. When DN goes high, the input of the TON goes low and the timer resets.



©2025, Dr. Dan Wedding

7) **Pattern:** When the program runs, the lamps will cycle through the patterns below. The sequence is the same as the pattern stored in the file. In this example, each of the patterns will be displayed for half of a second.

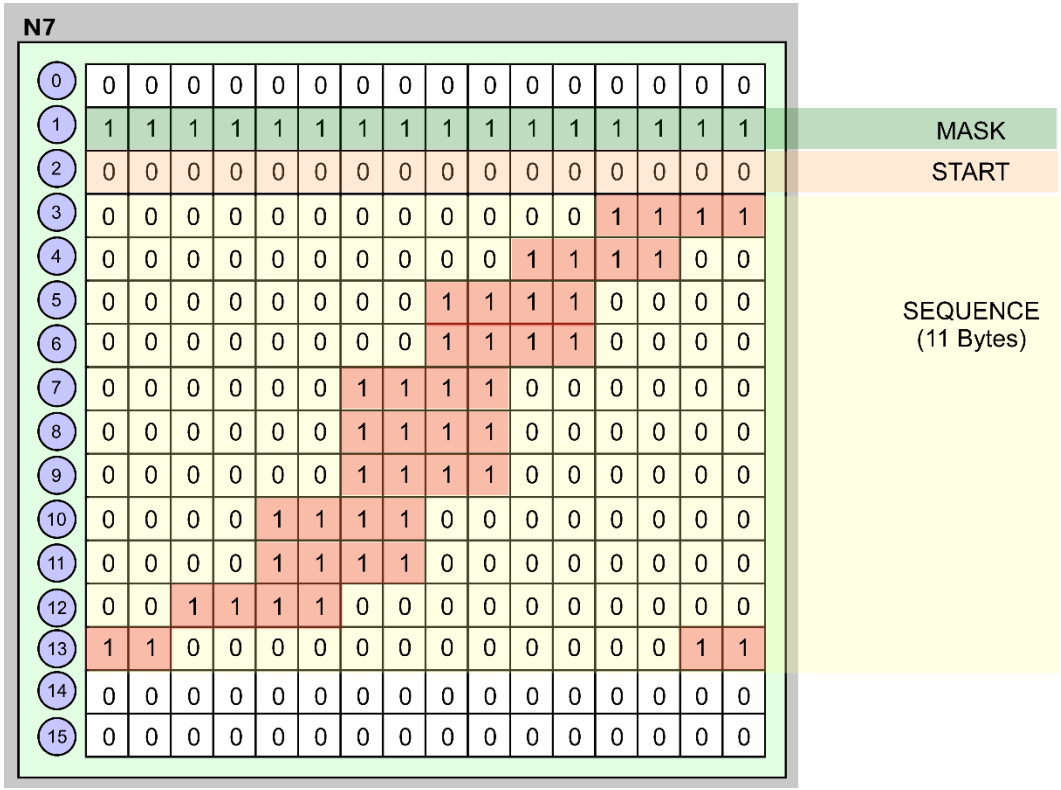


©2025, Dr. Dan Wedding

8) **Timing:** In the previous example, each pattern was visible on the lamps for the same amount of time. Sometimes that is not what you want. You may wish to have the patterns on for different lengths of time. There are two ways to do this.

   a. **Repeat a Pattern:** One way to keep a pattern on the pins for different lengths of time is to use a timer with a fixed time (as in the previous example). But if you want a pattern to stay on the output longer, just repeat the pattern.

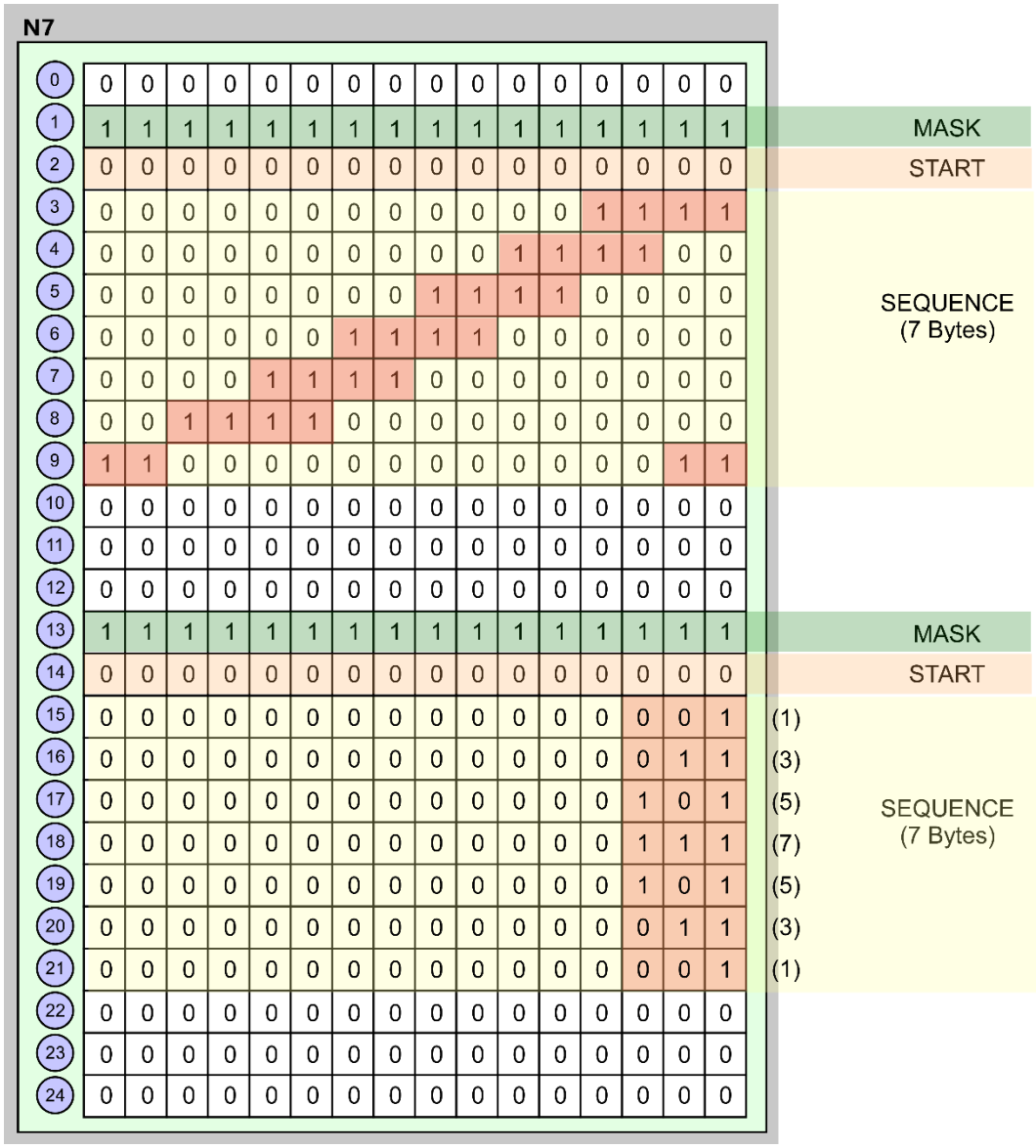
      Assume each state is displayed on the lamps for 1 second. That means the middle state will be repeated 5 times and it will be on the output for 5 seconds. This method is simplistic and is only useful if you have just one (or maybe 2) states that are different. Also, they need to be even multiples of each other (meaning all of the states can be evenly divided by the timer time).



0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
4	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0
5	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0
6	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0
7	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0
8	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0
9	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0
10	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0
11	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0
12	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0
13	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

©2025, Dr. Dan Wedding

b. **<u>Two SQO commands running at the same time:</u>** If there are two sequencers, one for the pattern and one for the time it takes for each step.



©2025, Dr. Dan Wedding

In the program below, the timer is set for a time base of 1, and a preset of 1.

After one second, DN goes high.

TON is disabled and both SQO are enabled.

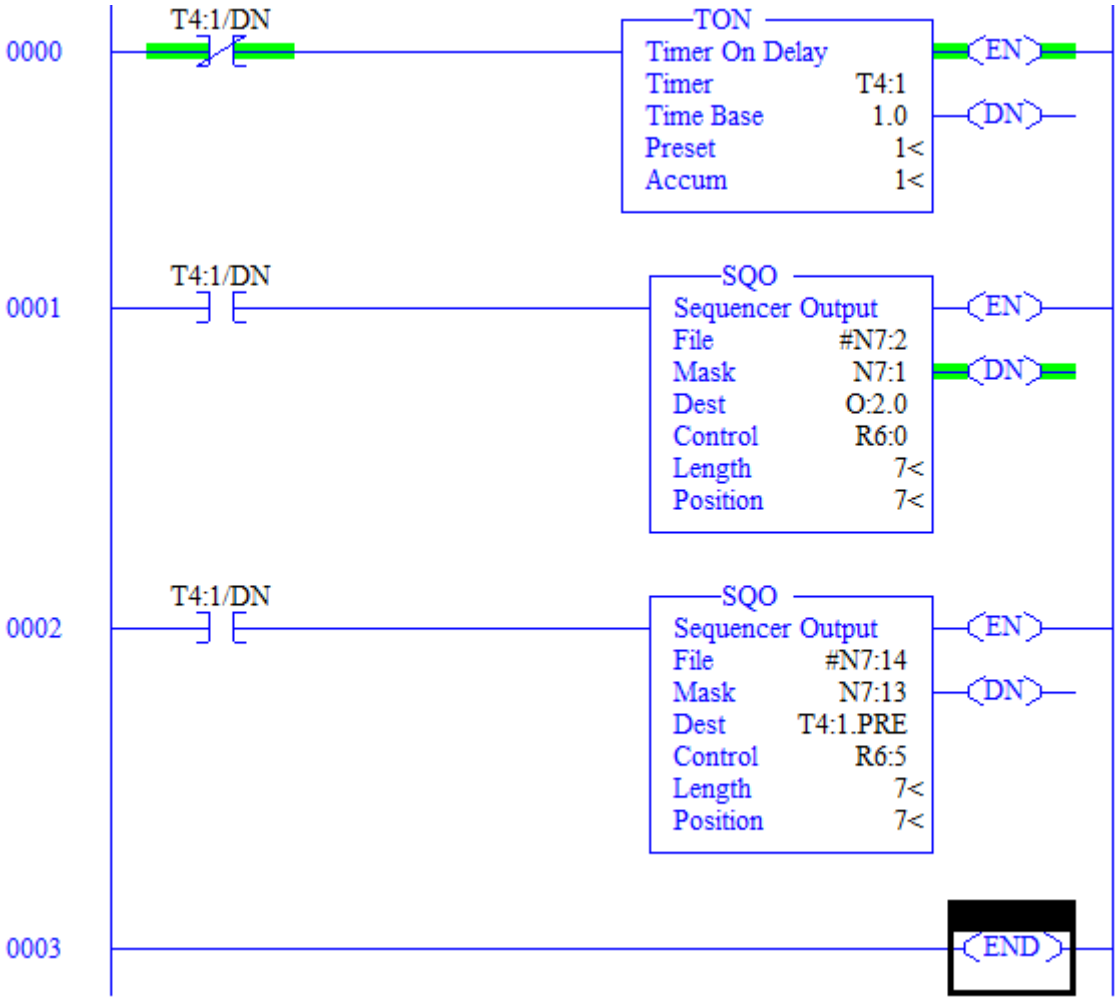
The first SQO pulls the first value from the first file and sends it to the lamps.

The second SQO pulls the first value from the second file and loads the TON preset.

On the next pass, DN is low and TON is re-enabled and both SQOs are disabled.

The process repeats all values in the file.

At the end of the file, the sequence starts over.



©2025, Dr. Dan Wedding