Drives and Controls

Timers and Counters Instructions

By,
Vikas M Sampath

Introduction

- Plc's use modern microprocessor technology and have timers and counters included in the instruction set.
- For example, each SLC 500 can have up to 256 timers or counters in each of multiple timers or counter files.
- Software timers or Counters are easily included in your ladder programs by simply programming the desired instruction on the ladder rung.

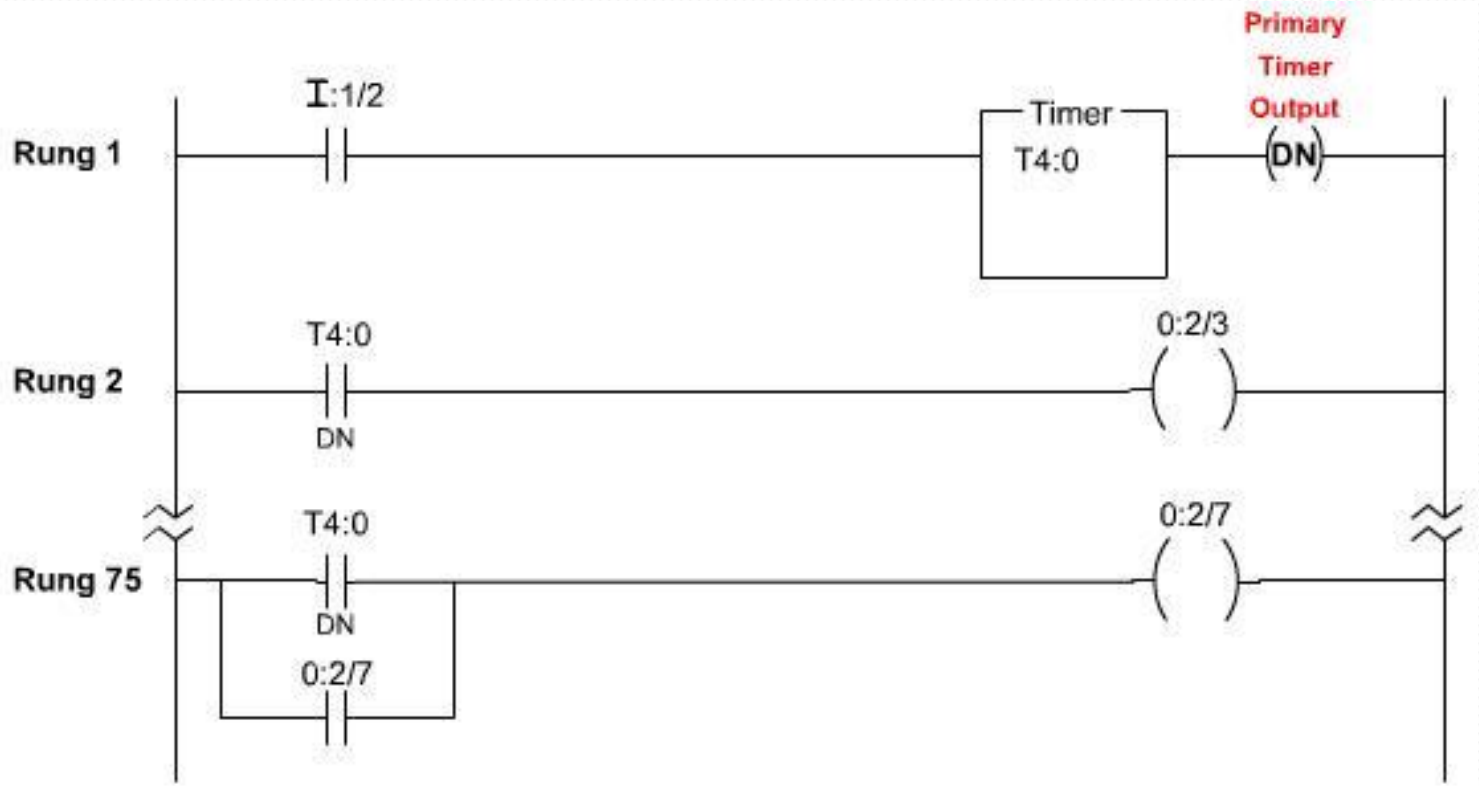
TIMERS

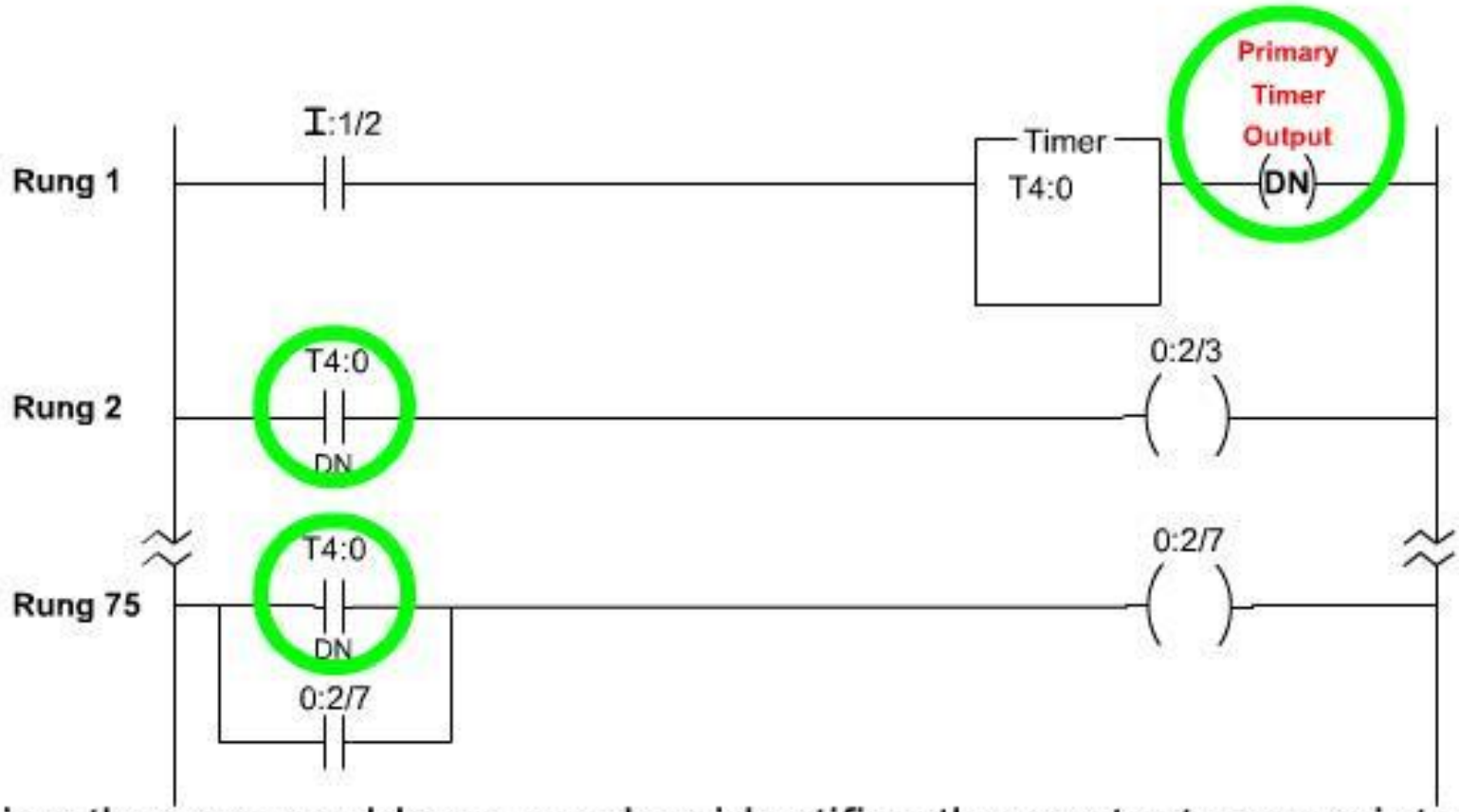
- All PLCs have timer instructions.
- Even though each manufacturer may represent timers differently on the ladder diagram, most timers operate in the same manner.

A Timer consists of the following parts:

- Timer address.
- Preset Value.
- Time Base.
- Accumulated Value.

Timers Usage:



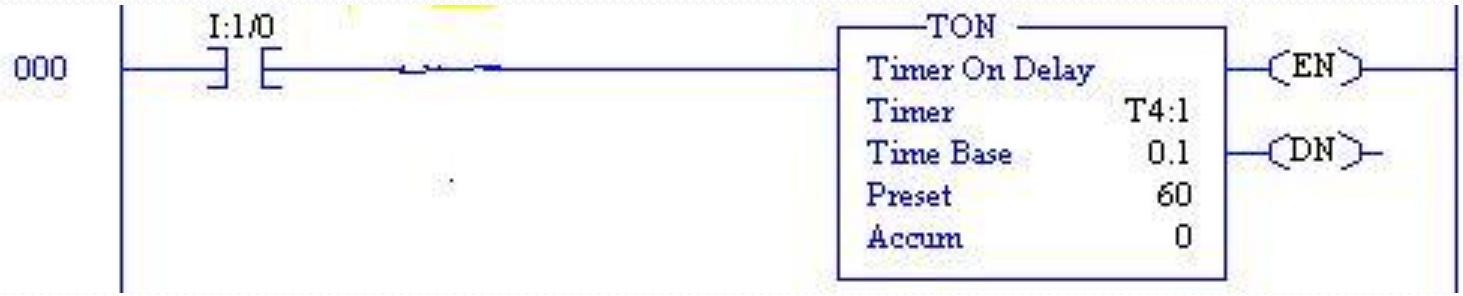


Using the same address number identifies the contacts associated with a specific timer.

Timer Elements:

A timer is one element and it is made up of three 16 bit words

- **Word zero**: Status bits.
- **Word One**: PRE
- **Word two**: ACC



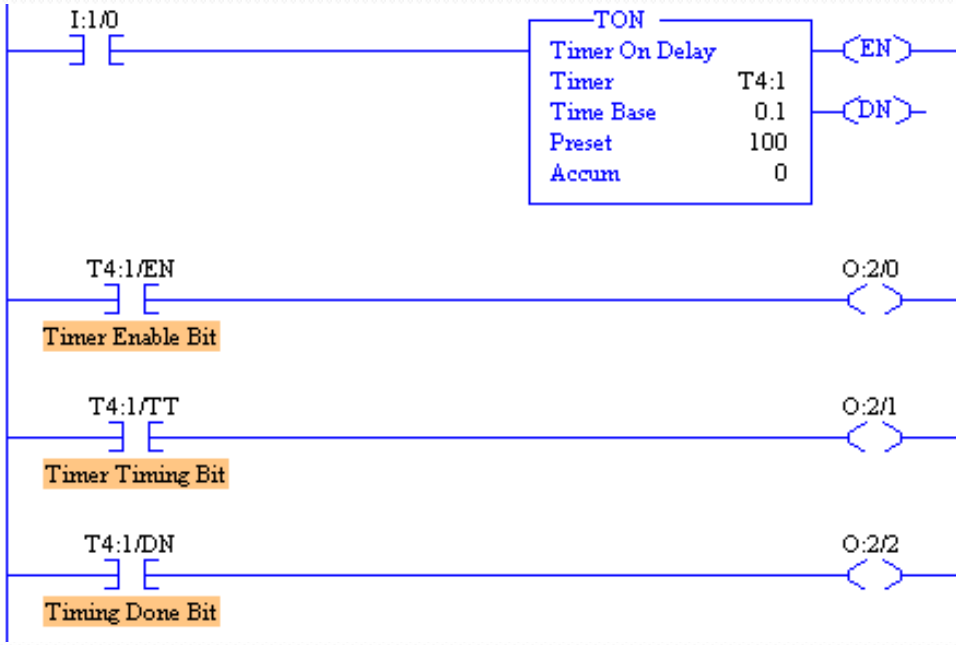
Types of Timers:

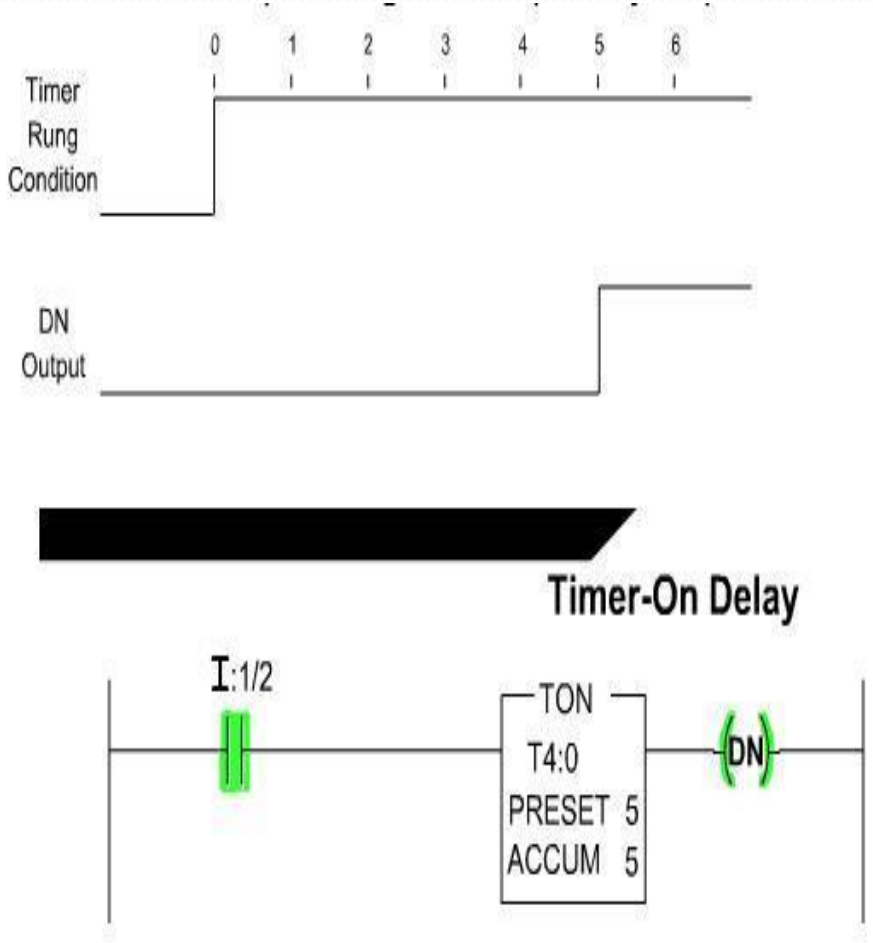
There are three types of timers :

- **On Delay Timer.**
- **Off Delay Timer.**
- **Retentive Timers.**

On Delay Timer

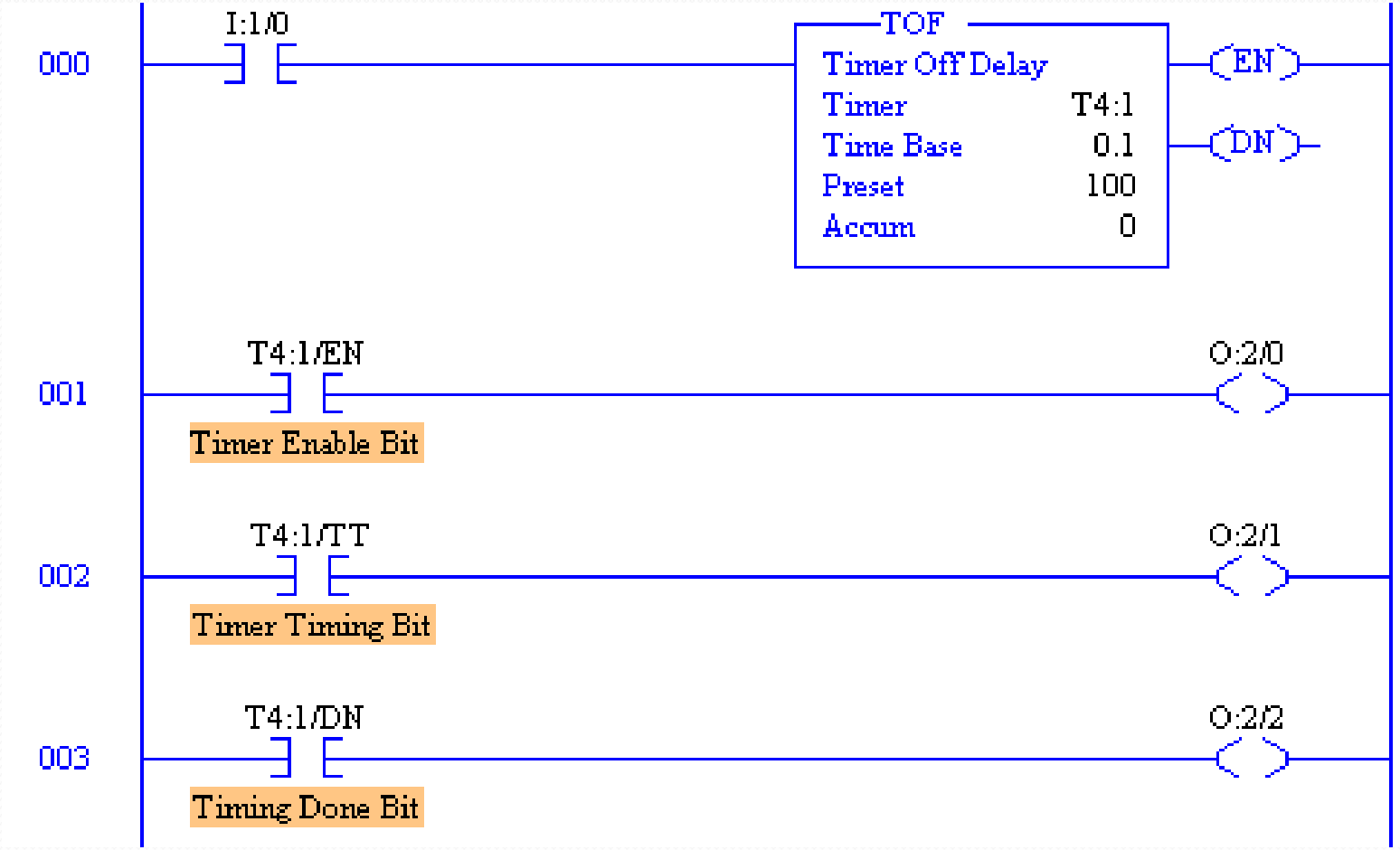
..

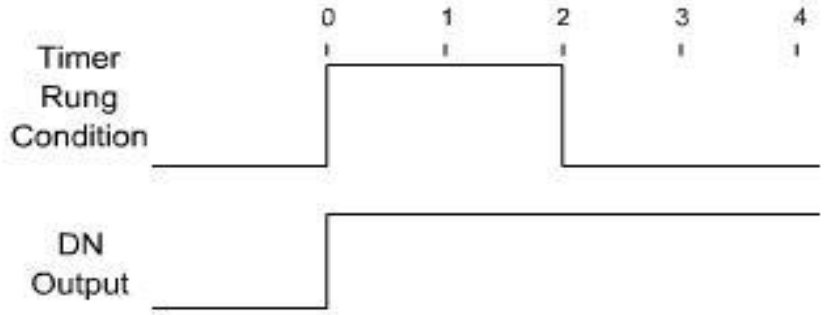


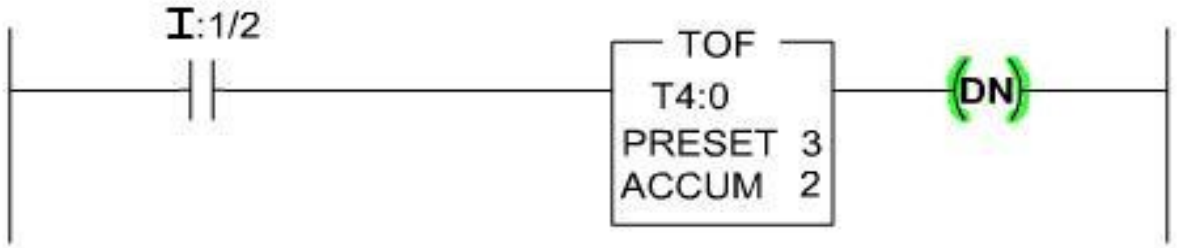


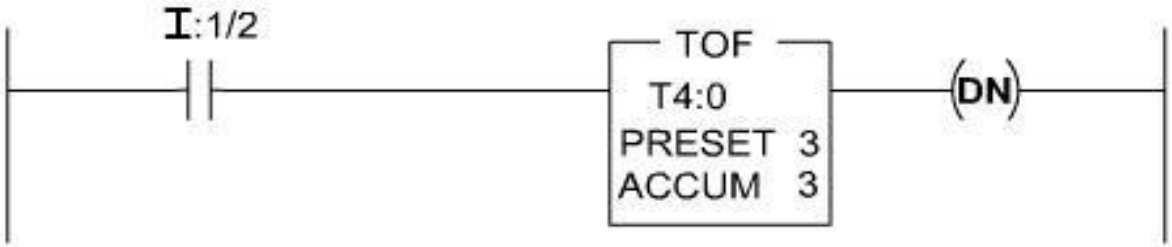
Timer addressing is as follows: T (Timer file number): (Timer element number).The timer address T4:1 is addressing timer file 4 , timer element 1.

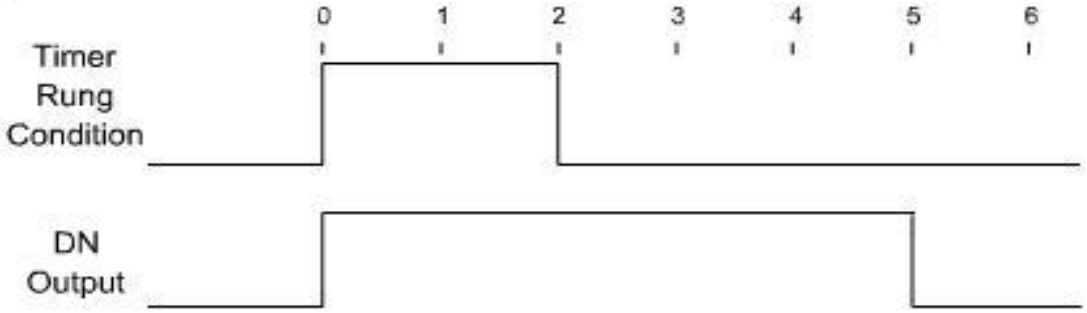
Off -Delay Timer:



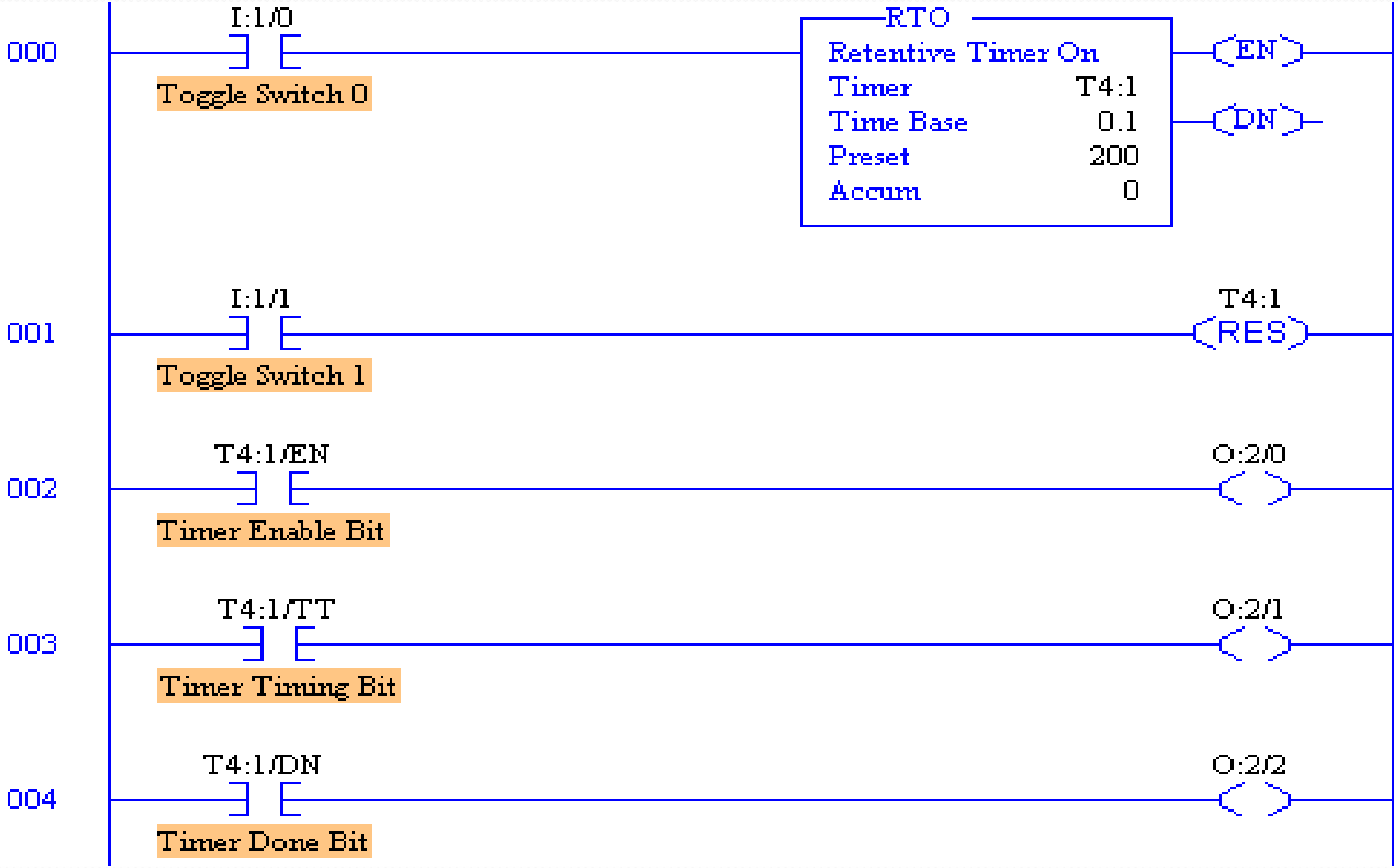

0
1
2
3
4
Timer
Rung
Condition
DN
Output


I:1/2
TOF
T4:0
PRESET 3
ACCUM 2
DN


I:1/2
TOF
T4:0
PRESET 3
ACCUM 3
DN


0
1
2
3
4
5
6
Timer
Rung
Condition
DN
Output

The Retentive Timer :



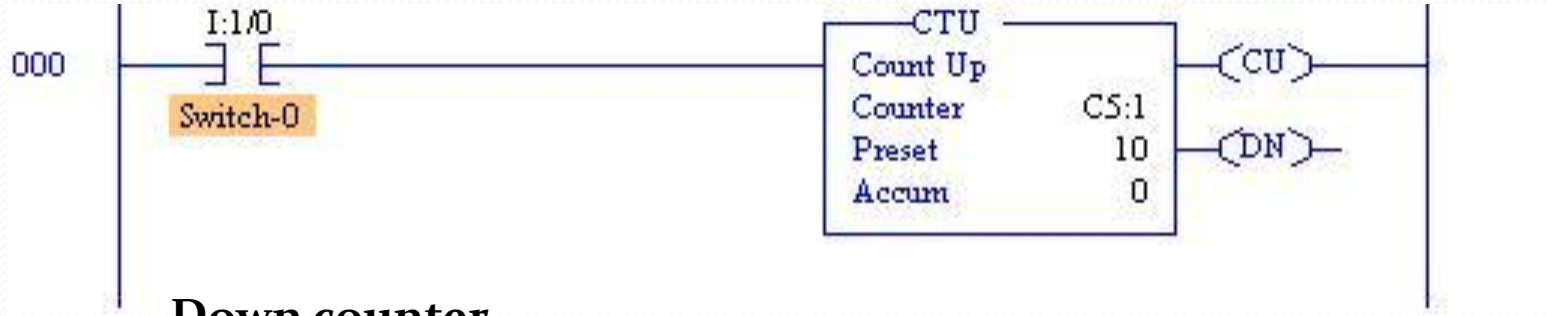
COUNTERS

- Every PLC has counter instructions.
- Although most PLC counters work the same, the instruction symbols used and method of programming will change for different manufacturers.
- The Typical counter counts from **0** up to desired value, called the **PRESET** value
- The accumulated value is the current or accumulated count.

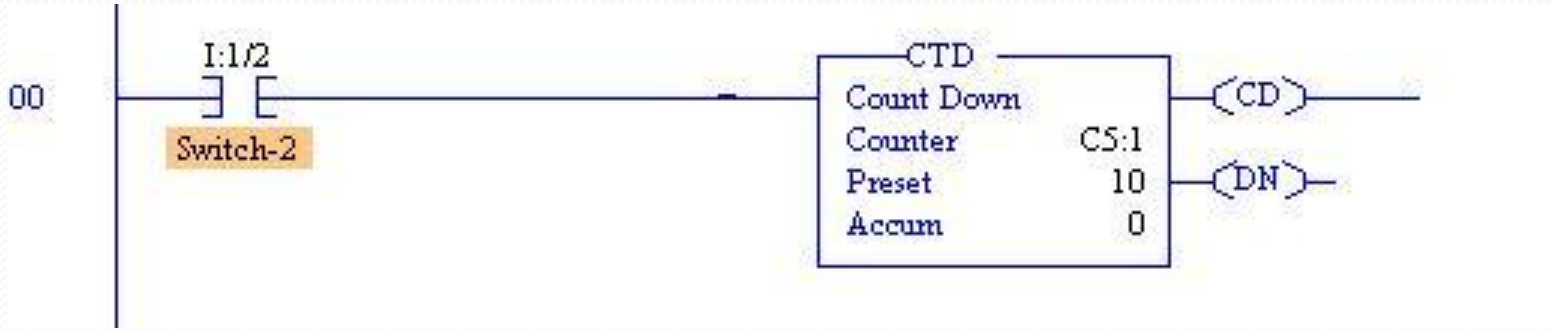
Types of Counters:

Up counter and Down counter

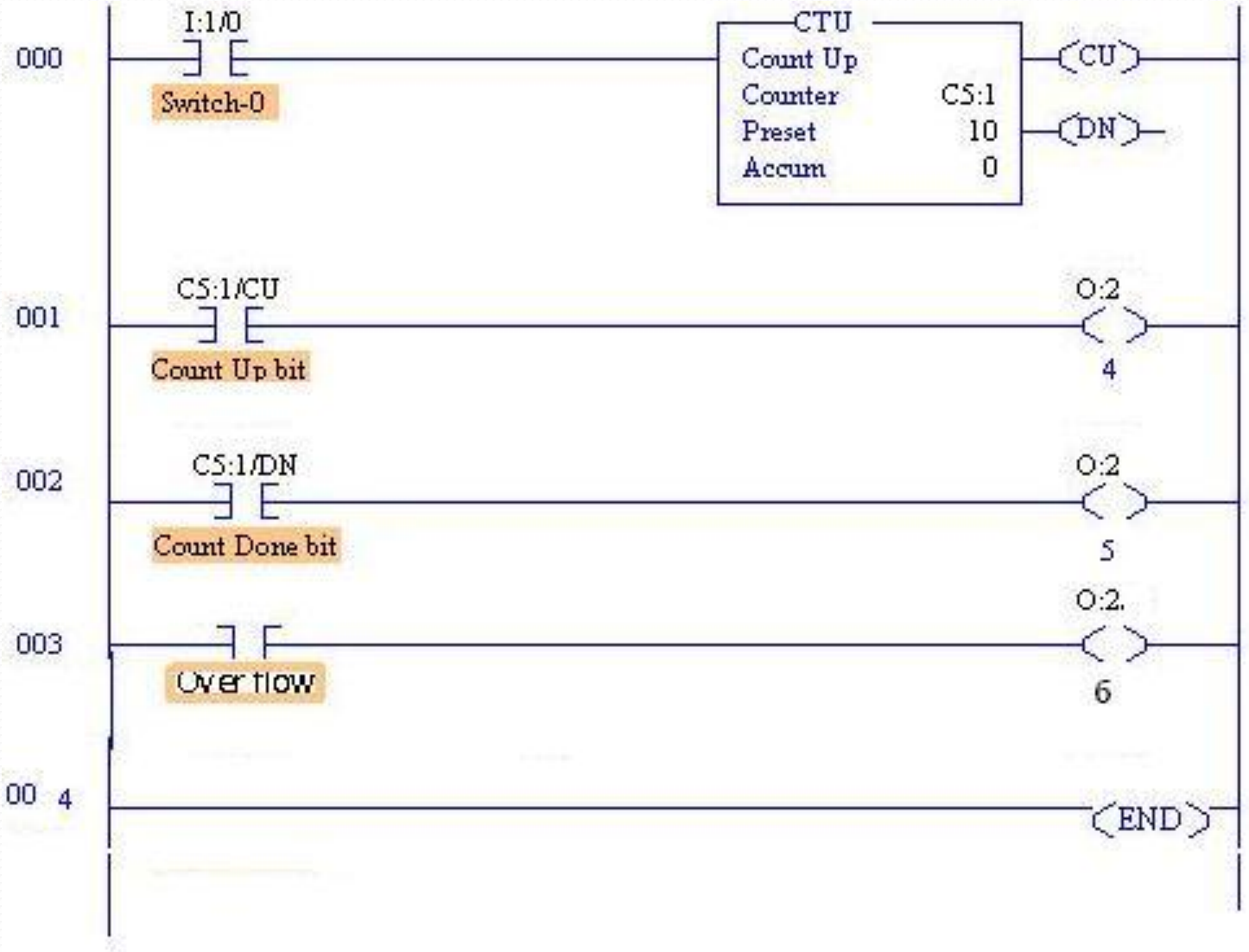
Up counter

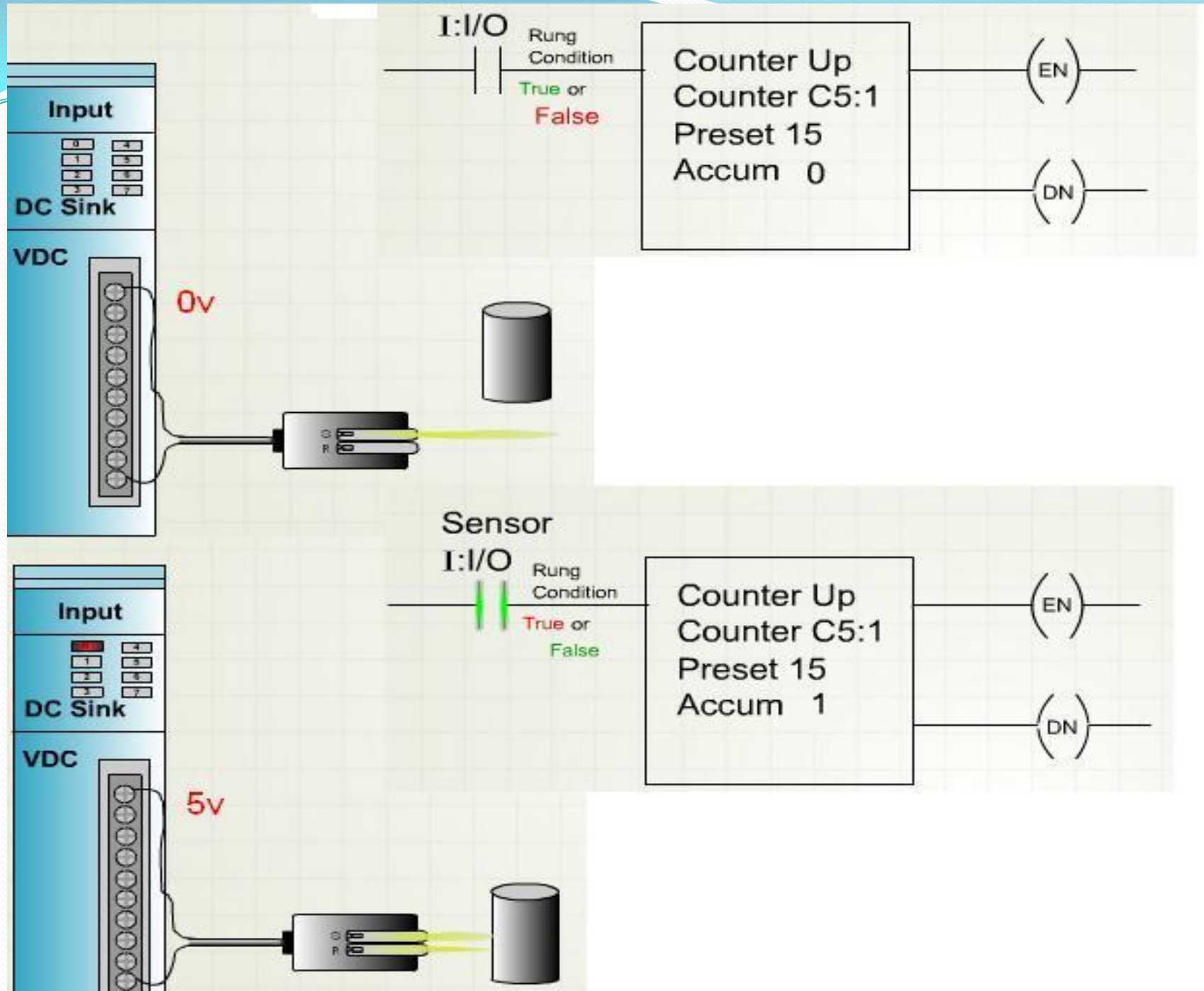


Down counter

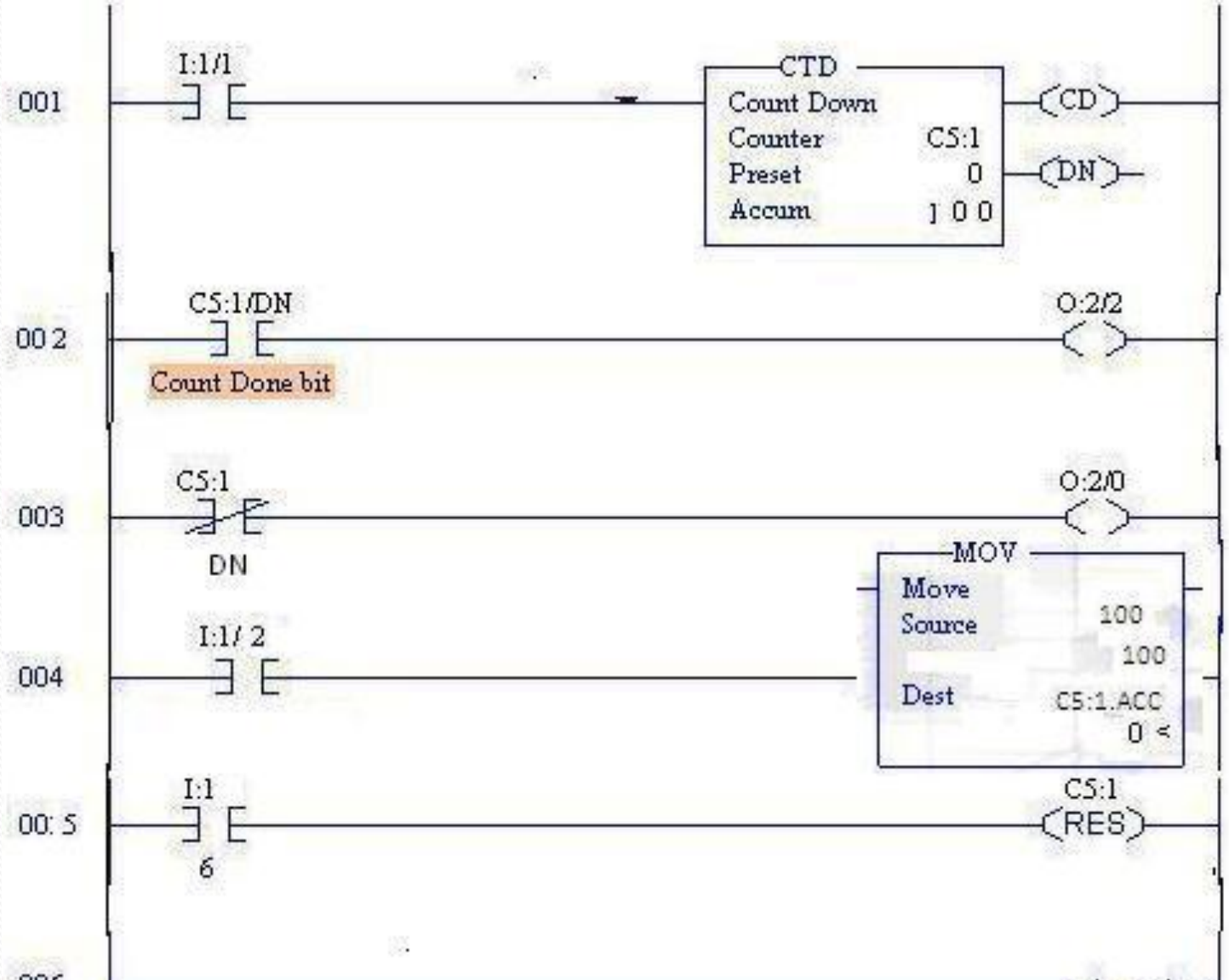


Counter –Up Instruction:




I:I/O
Rung
Condition
True or
False
Counter Up
Counter C5:1
Preset 15
Accum 0
EN
DN
Input
DC Sink
VDC
0v
Sensor
I:I/O
Rung
Condition
True or
False
Counter Up
Counter C5:1
Preset 15
Accum 1
EN
DN
Input
DC Sink
VDC
5v

Count-Down Instruction:

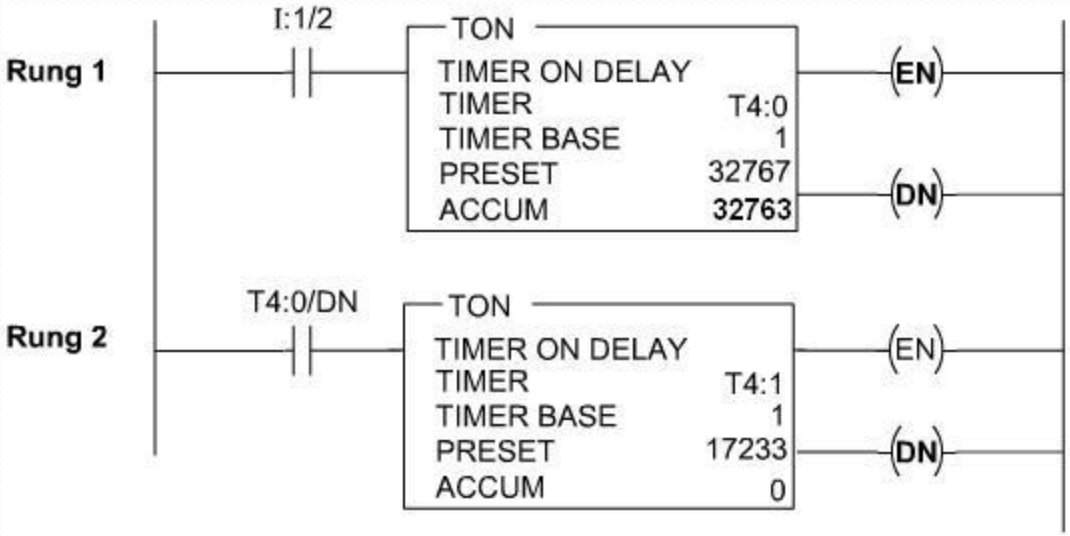


Combining Counters and Timers:

- Timers and Counters can be programmed to work together.
- Timers or Counters can be connected or cascaded together to increase the time or count.
- One counter can be used to count the number of cycles another counter as completed.
- There are many applications and ways in which timers and counters can be programmed to work together.

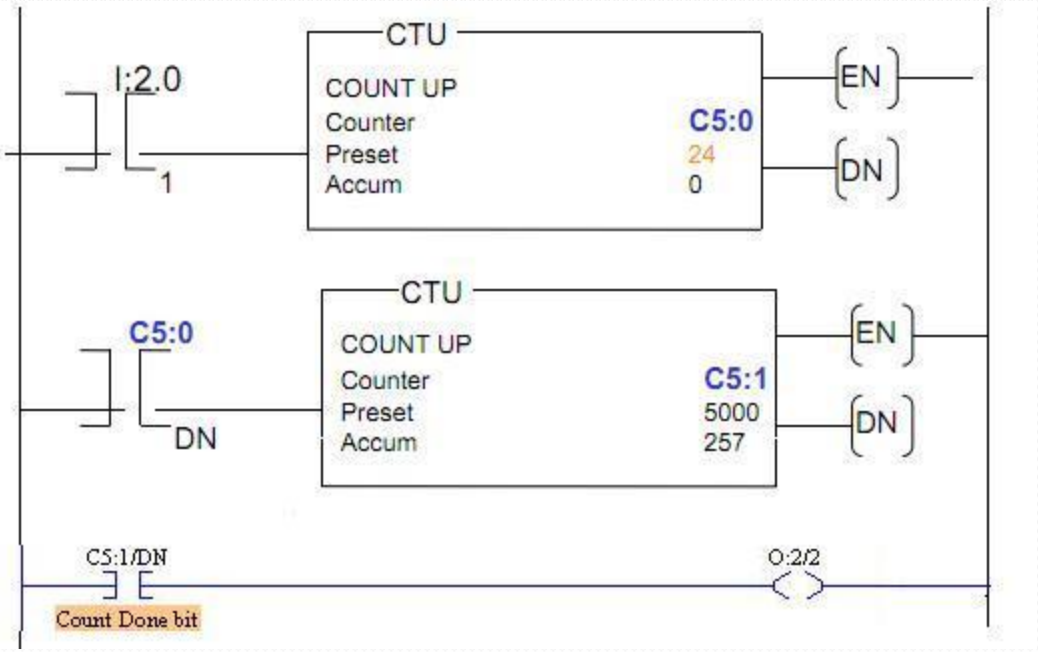
Timers cascaded together

- Two timers cascaded together so as to lengthen the time that can be counted. The maximum preset of an SLC 500 timer is 32,767 seconds (9.10 hrs).



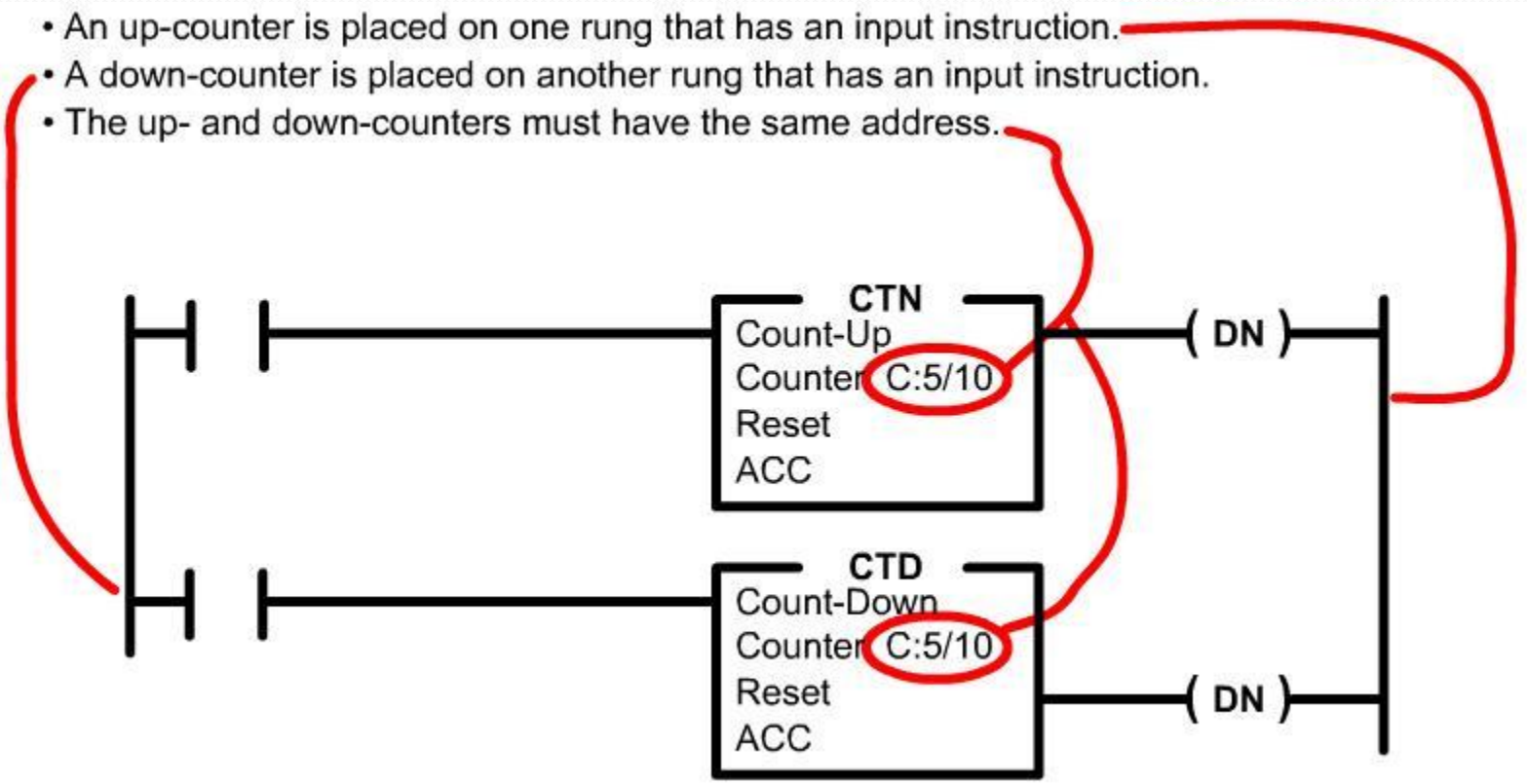
Counters cascaded together

- In the same way of the Timers and the counters can also be cascaded



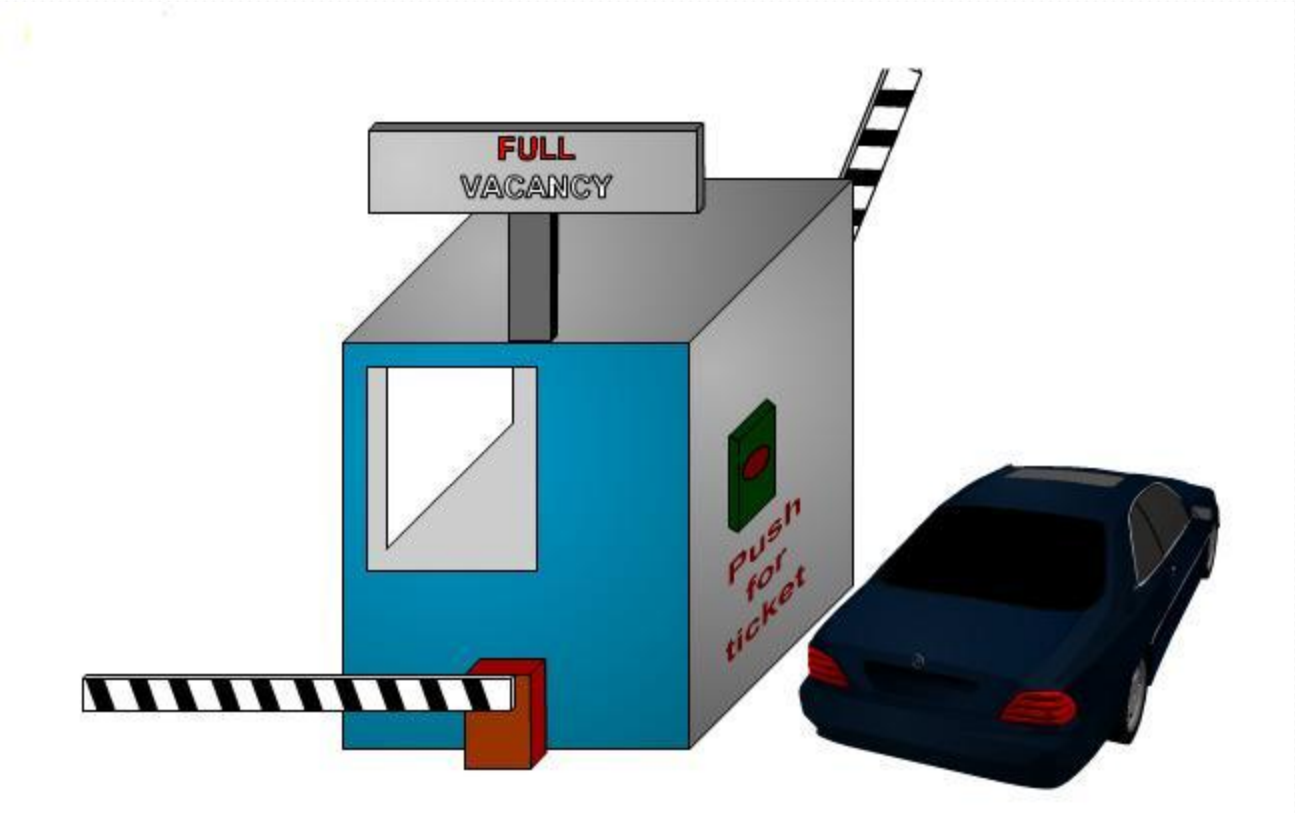
Combining UP and Down Counters

- An up-counter is placed on one rung that has an input instruction.
- A down-counter is placed on another rung that has an input instruction.
- The up- and down-counters must have the same address.

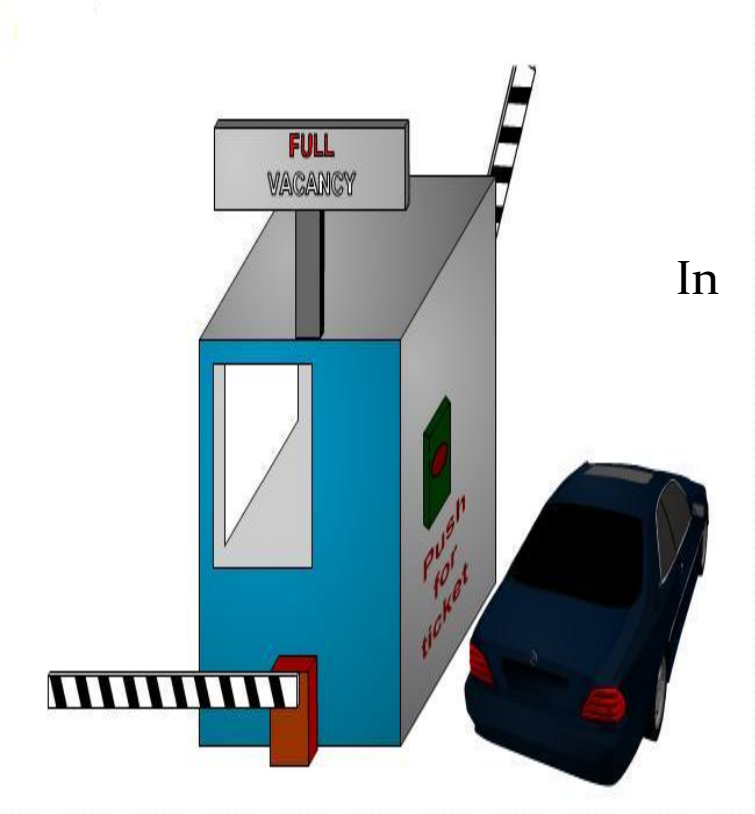


Up - Down Counter Application:

- One example of an up down counter application is a semi automated parking gate.

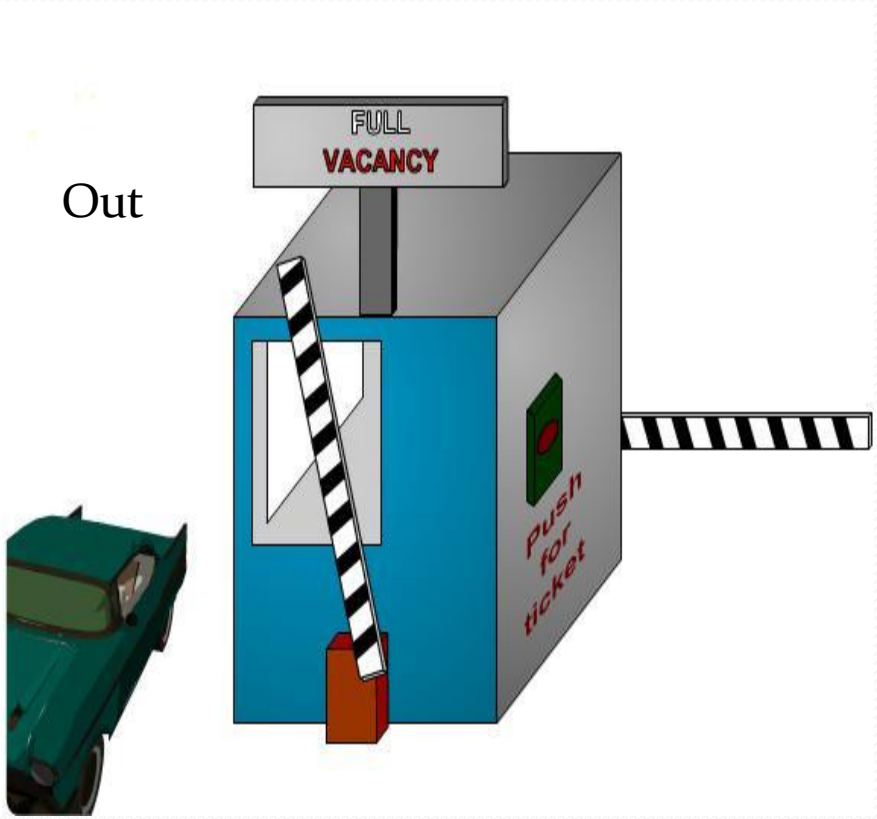


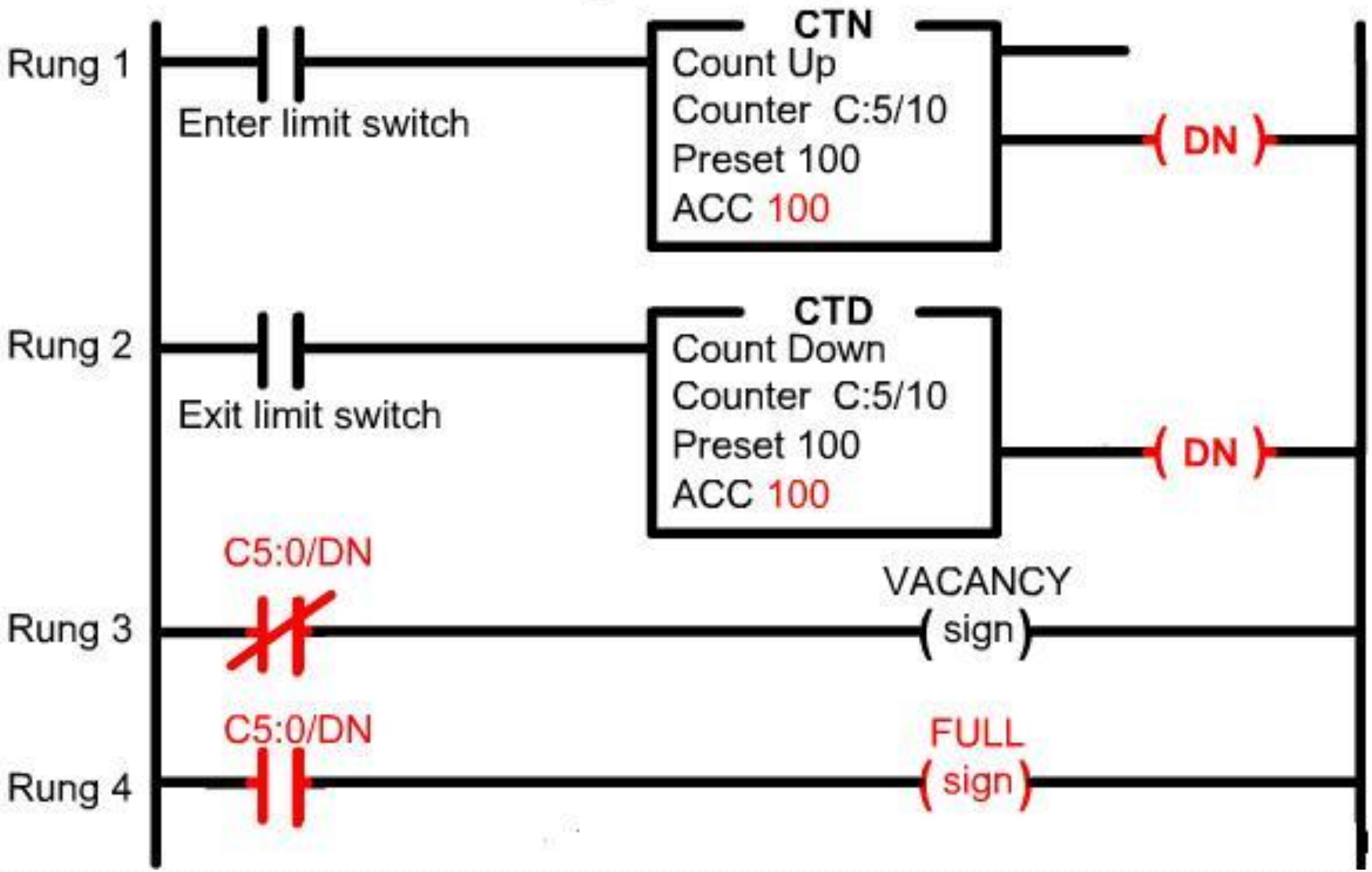
When 100th car goes in and 100th car come out



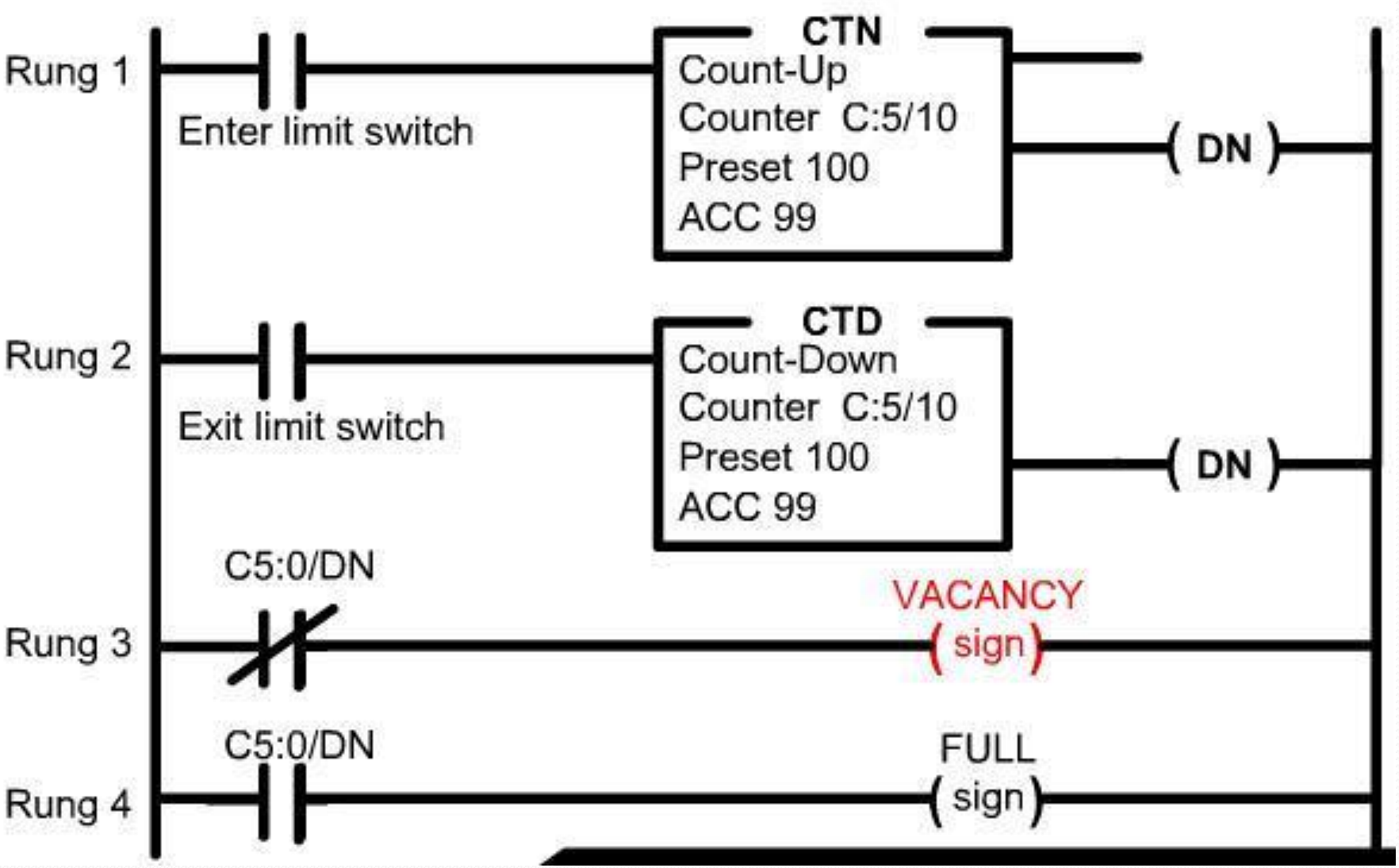
In

Out





When the last car enters inn



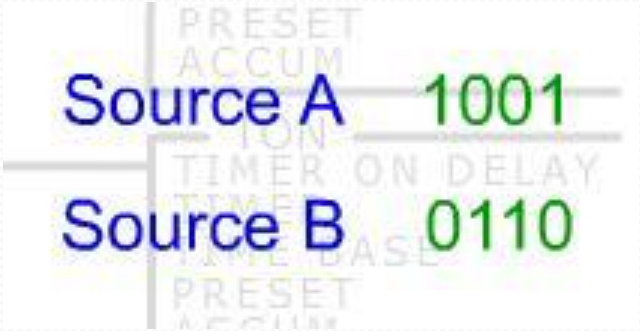
When a car goes out, after the House full

Comparison and Data Handling Instructions:

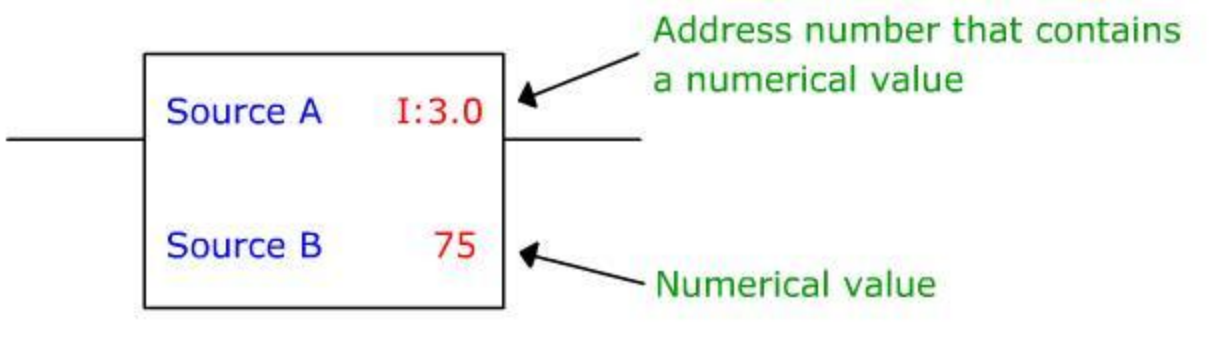
Comparison Instructions:

- Data manipulation is a category in which multi bits are used.

- Data manipulation is the data compare instruction that compares two numerical values. The number being compared are identified as **Source A** and **Source B**
- **Source A:** This is the address of the data to be tested.
- **Source B:** Source B can be either a constant or a word address.



- If you see a constant as **Source B**. The constant will be tested with the data residing in the address specified in **source A.**
- And if use **Source B as address** then it will test with data residing in **Source A address.**



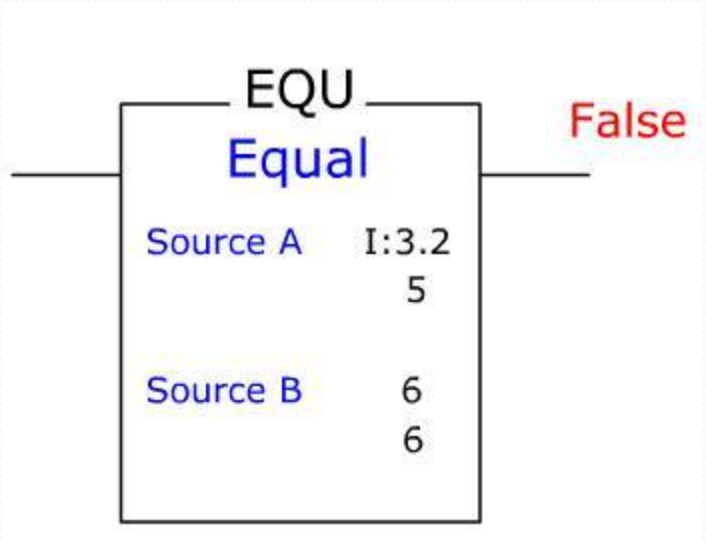
Types of Comparison Instructions:

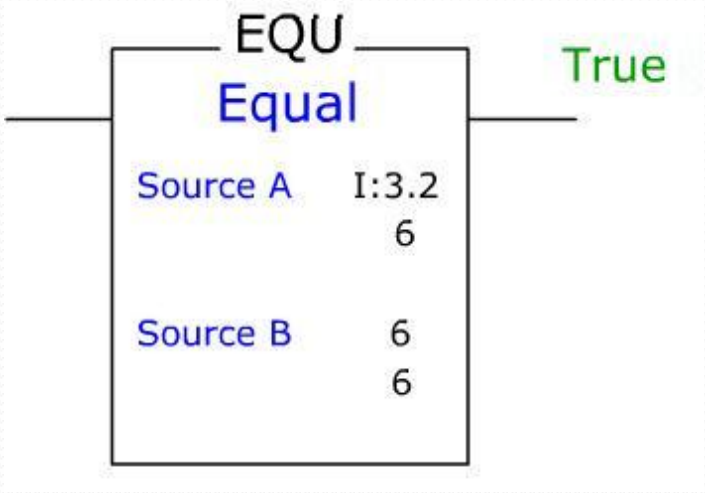
We can divide them to mainly 3 types:

- **EQU instruction**
- **LES instruction**
- **GRT instruction**

Equal (EQU):

- The output of the EQU produces a logic true if the numerical value in source A address is equal to source B's address or constant.



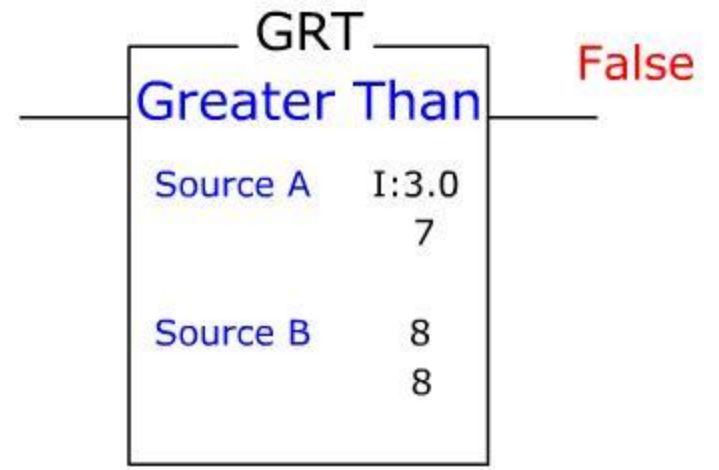


Less than Instruction (LES):

- The out put of less than instruction produces a **logic true** if the constants of **source A** is less than the constant of **source B**

Greater Than Instruction (GRT):

- The out put of **GRT** instruction produces a **logic true** if the constants of **source A** is greater than the constant of **source B.**



Data Handling Instructions:

- Data handling instructions are used when data needs to be **moved**, or **copied** from one data file source to a desired destination.
- There are also, instruction when data must be **converted from one format to another** before it is sent on a desired destination.
- The next section will address the SLC 500 and Micro Logix data handling instructions.

Types of Data Handling Instruction:

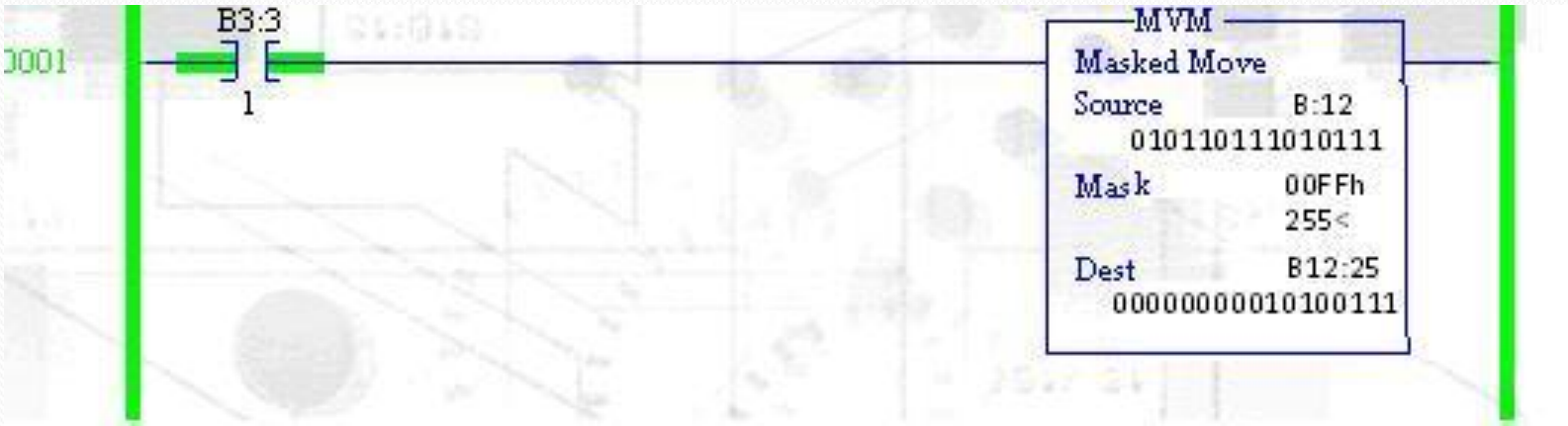
- **MOV** (Move Instruction).
- **MVM** (Masked Move Instruction).
- **FRD** (Converts BCD to an Integer)
- **TOD** (Converts Integer to an BCD).
- **COP** (Copy Instruction).
- Logical Instructions**(AND,OR,XOR,NOT).**
- **LIM** (The Limit test Instruction).
- **SCP**(Scale with Parameters Instruction).

The ”Move” (MOV) Instruction:

- The move instruction is an out-put instruction that moves a copy of one data file word to a specified destination.
- This instruction can be used to move a copy information stored in one data file location to another.

The Masked Move Instruction (MVM):

- The masked move instruction is an output instruction that moves a copy of one data file word through a mask to a specified destination.
- A mask out source bit that are not to be transferred to the destination.



Masking Rules:

- **Mask Rules:**
- The mask is a hexadecimal value.
- Data is passed through the mask bit by bit. The mask bit in the same position as the source bit determines if the data is to pass or not.
- To pass data through the mask, set the appropriate bit (setting a bit is making it equal to **1**).
- To mask data from passing from the source to the destination, Reset the appropriate bit(resetting a bit means making it equal to **0**).
- Destination bits that correspond to zeros in the mask are not changed.
- Mask bits can be either a constant or the address where that mask will be found.

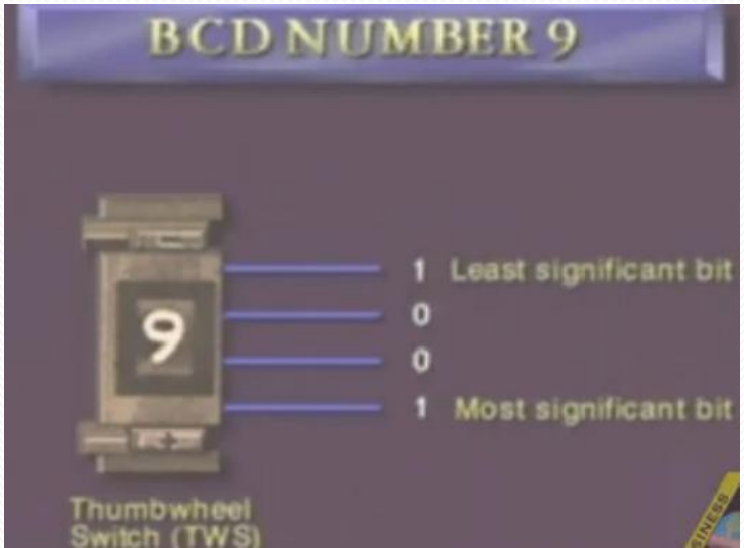
Masking Example:

- For Example, if the source data is **1010 1010 1010 1010** and the hexadecimal mask is **00FF** (**0000 0000 1111 1111**)What will the destination contain??

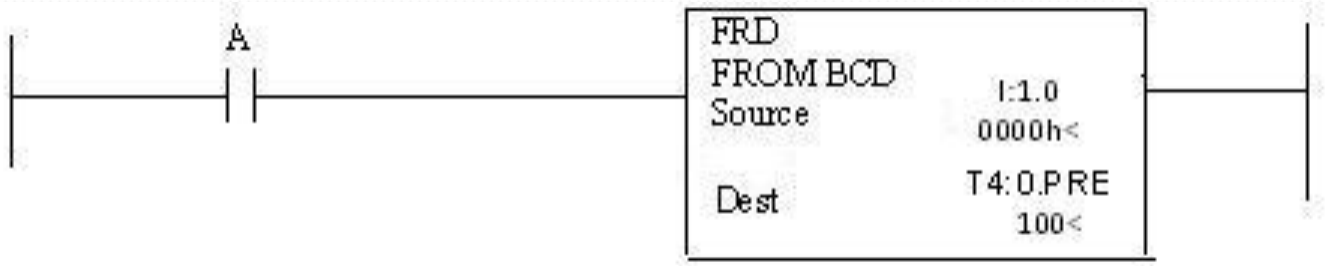
Source:	1010	1010	1010	1010
Mask 00FF:	0000	0000	1111	1111
Destination:	0000	0000	1010	1010

Converting BCD to an Integer (FRD):

- The FRD instruction is an output instruction that converts BCD value into integer data.
- Use of this instruction is to convert BCD input data, such as thumbwheel data from the input status file data into an integer with an FRD instruction

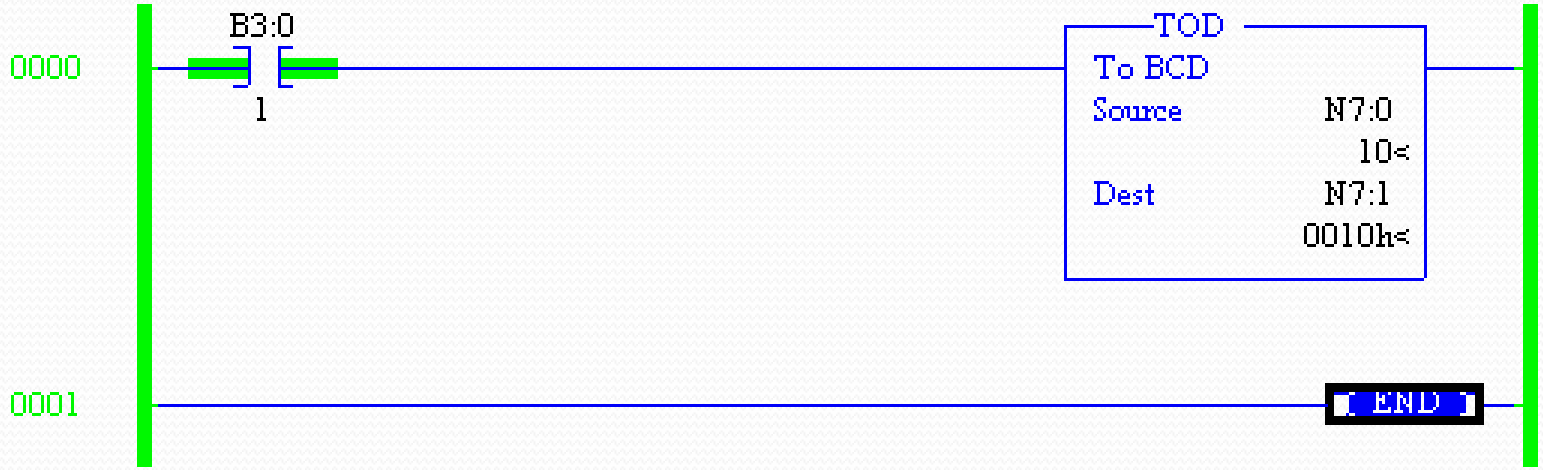


- FRD instruction converts the BCD input data from a address source and moves it into the Preset value say T:0.PRE of the timer instruction



Converting an Integer to BCD (TOD):

- The output data bound for a formatted BCD data display needs to be converted to BCD data before it is sent to the out-put module.



Copy Instruction:

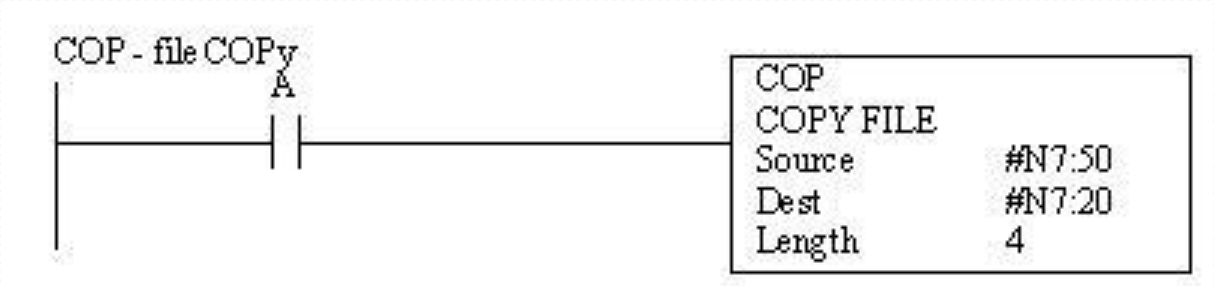
- The copy is an output instruction which copies a user defined group of 16 bit data file words called a user source file, to a destination file.
- The **difference between** the **Move** and **Copy** instruction is that ,the move instruction moves a copy of one 16-bit data word to a new location.
- the Copy instruction will copy up to 128 one word elements or 42 three-word elements to another location.

Elements transferred for mixed file copy examples

If the source is	And the destination is	And the Length is	What will be transferred?
#N7:2	#N7:200	10	10 integers file elements or 10 words.
#N7:0	#T4:12	6	The equivalent of 6 timer elements or 18 integer words or elements.
#T4:0	#C5:12	6	Copies 6 timer elements or 18 words

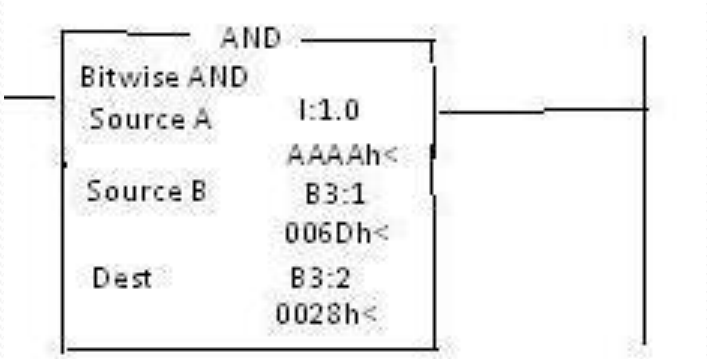
Example with a Ladder rung:

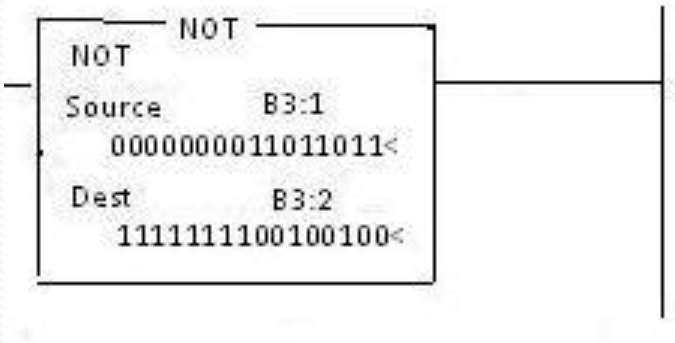
- As long as the input is true, 4 words will be copied from user defined file N7:50 to user defined file N7:20. A one shot instruction could be used if copy instruction is to be executed only once.



Logical Instructions:

- **AND instruction:**
- **OR instruction:**
- **NOT instruction:**
- **Ex-OR instruction:**



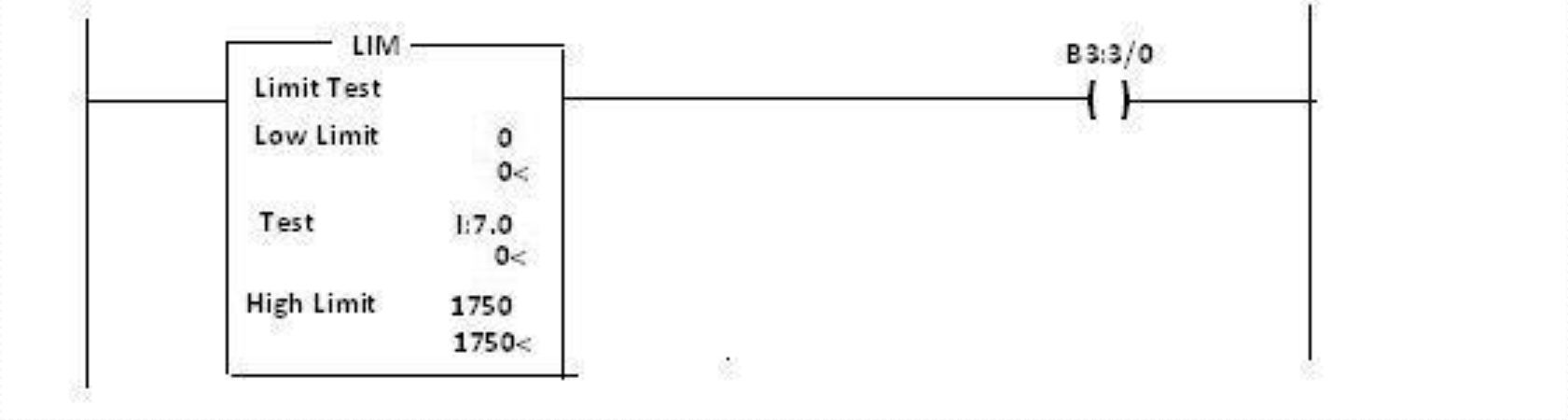


Limit test Instruction(LIM):

- The limit test instruction is used to test for values whether they are within or outside of the specified range.
- Programming the LIM instruction consists of entering three parameters:
- **Low limit**,
- **Test**
- **High limit**

Limit instruction in ladder:

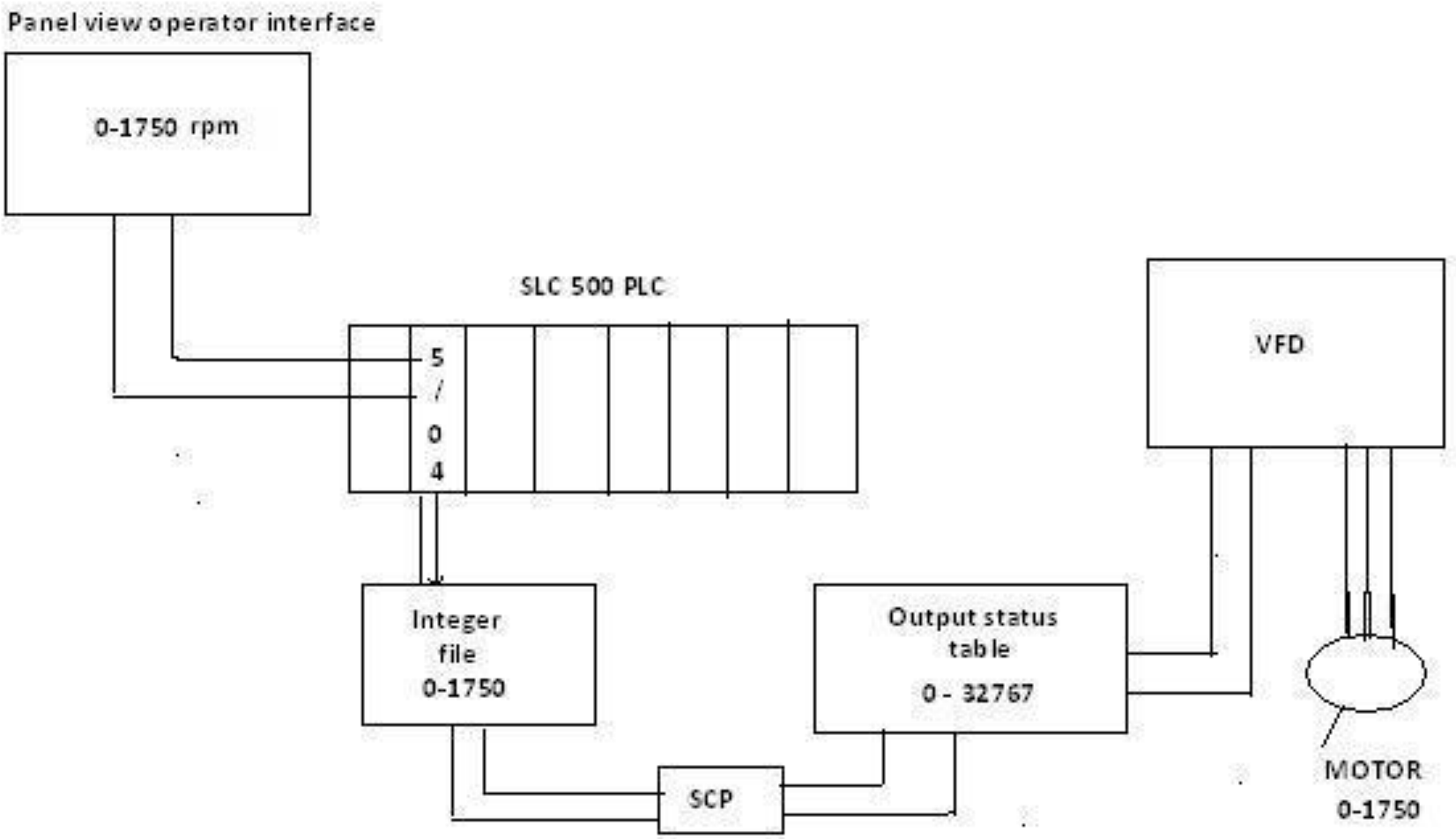
- The test parameter provides the information the high and low limits will be evaluated against.



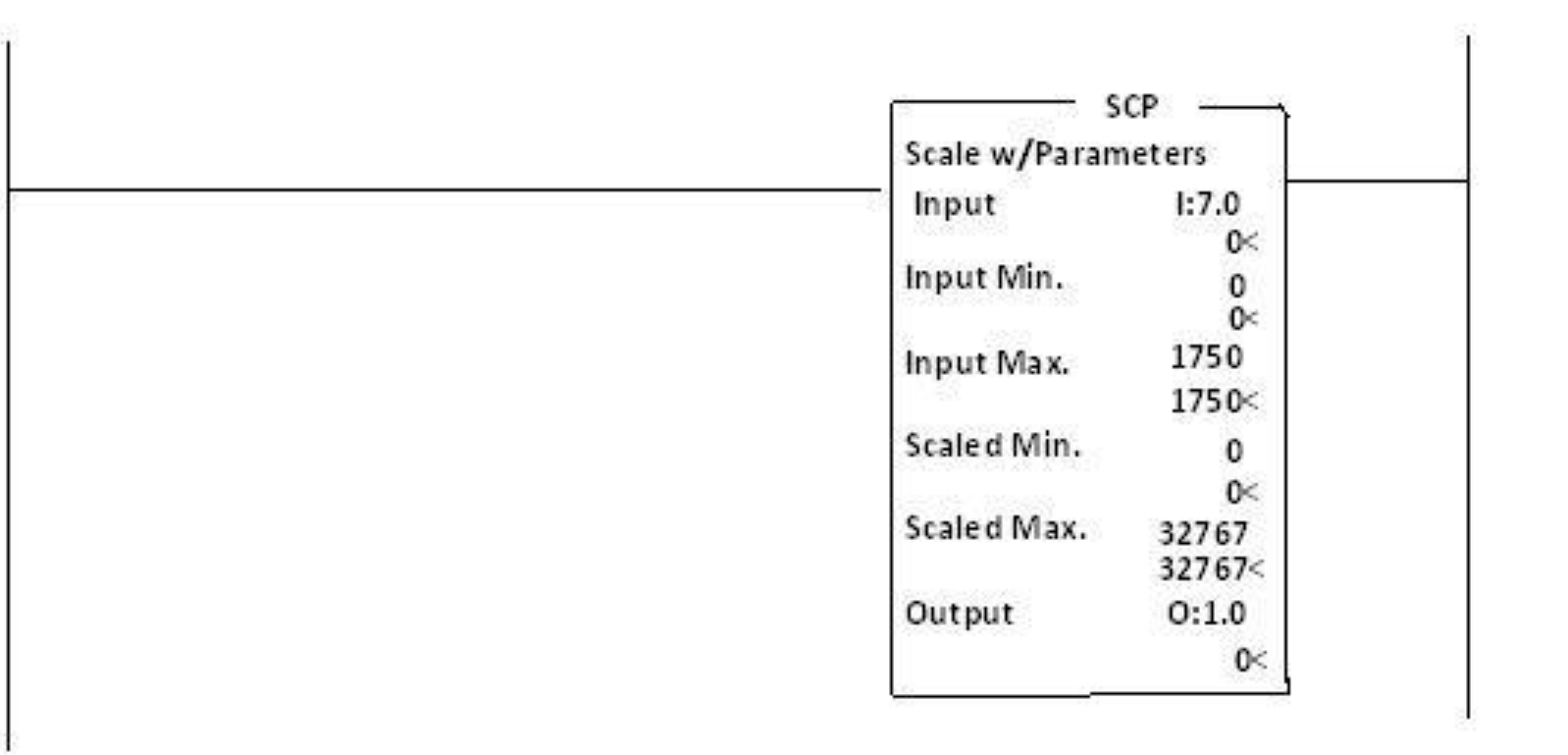
Scale with parameter (SCP) Instruction:

- Analog input data needs to be scaled (Converted) before it can be used in the PLC program.
- Input data from a field device needs to be scaled (Converted) before it can be used in the PLC program.
- Analog output data needs to be scaled (Converted) before it can be sent from an analog module channel to field devices.
- Output data needs to be scaled (Converted) before it is sent to an operator interface device or variable frequency drive.

Data flow from a panel view operator interface terminal in and out to the PLC and VFD

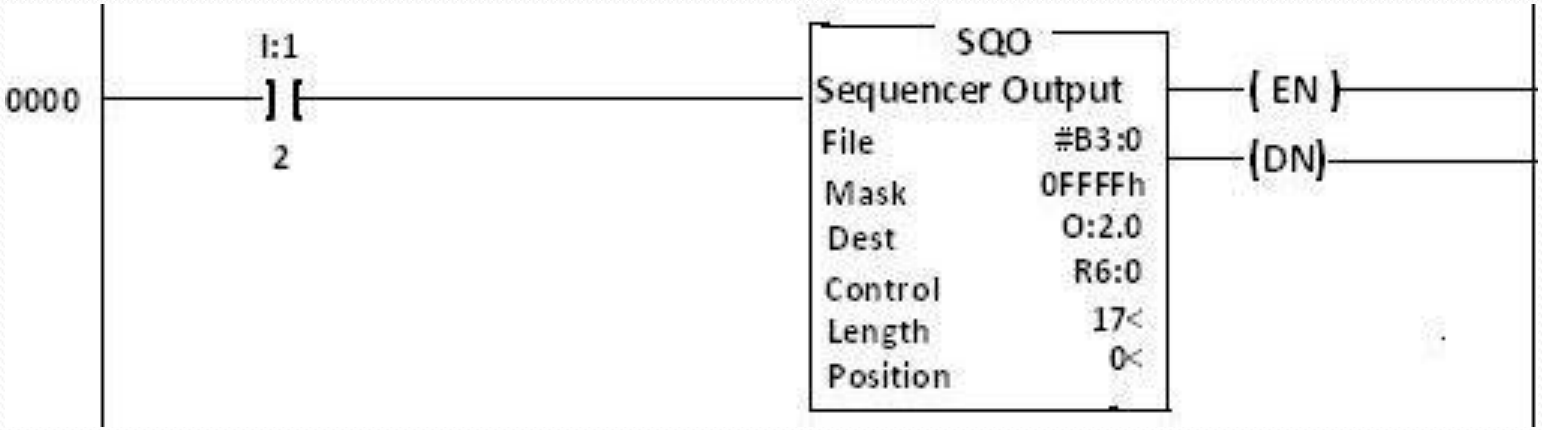


Ladder of SCP instruction:



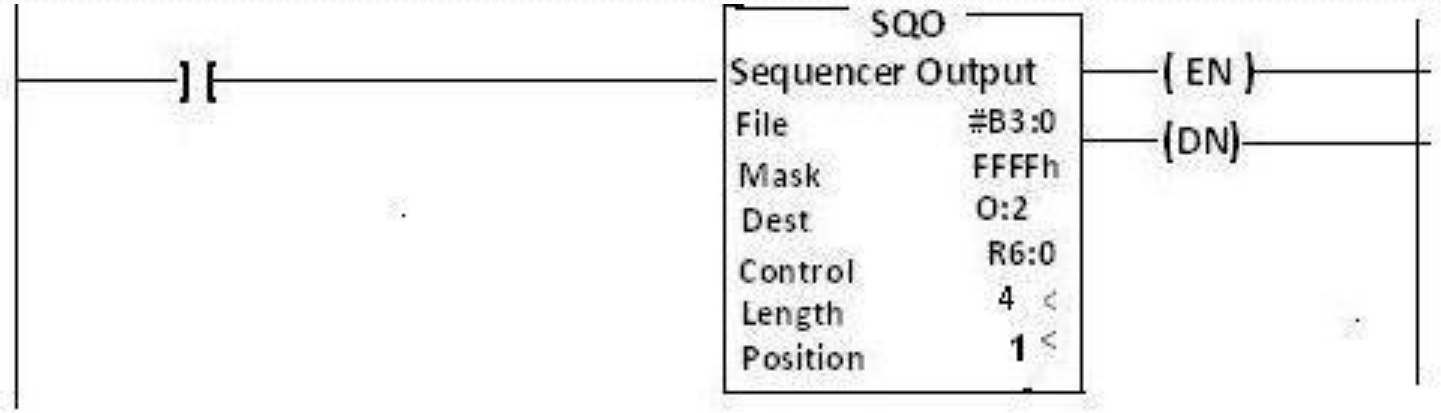
Sequence Out put Instructions(SQO):

- The sequencer output (SQO) instruction is programmed as an output on a ladder rung.
- Each false to true rung transition causes the sequencer output instruction to increment to the next sequence step.



Example of a 4 step sequence:

- The length is 4.
- And when the sequencer starts it will at the start position and when the SQO instruction goes from false to true for the first time. The Position points to step 1 and as 16-bit 0,1,2,3 are set they get energized.



Bit file Containing sequencer step Data:

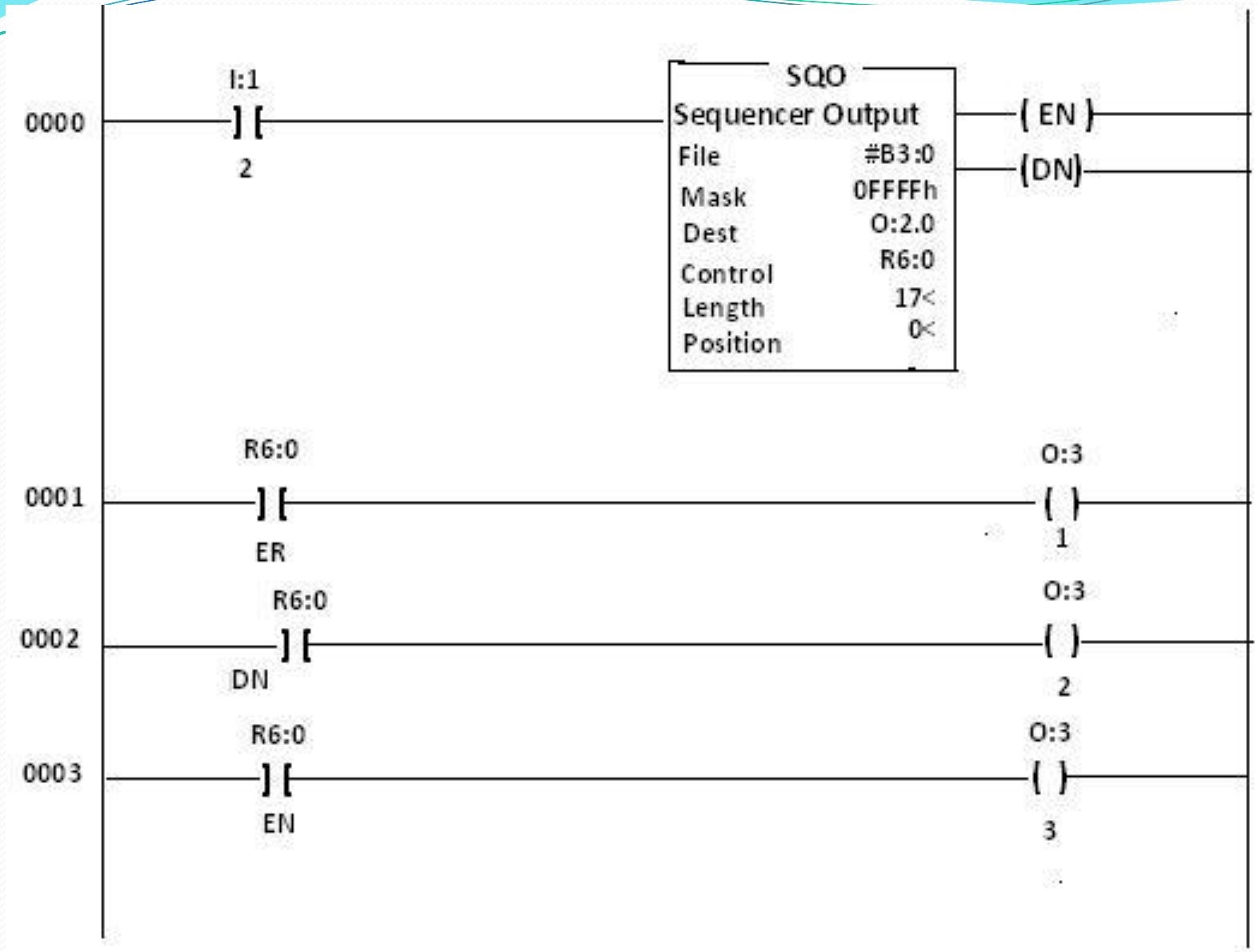
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Start
0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	Step 1
0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	Step 2
0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	Step 3
1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	Step 4

- For Position 2, the SQO instruction points to step 2,for each false to true transition, and energies bit 4,5,6,7
- Then for position 3 to Step 3, and energies 8,9,10,11
- Like this after completion of 4 steps the cycle Repeats, as the length is for 4.

Sequence Masking

- The mask parameter is a hexadecimal mask or the address where the hexadecimal mask will be found.
- In the example the masking parameter is 0FFFFh

Source bit file word	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
Mask passes all bits	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Bits to output	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1


0000
I:1
2
SQO
Sequencer Output
File #B3:0
Mask 0FFFFh
Dest O:2.0
Control R6:0
Length 17
Position 0
EN
DN
0001
R6:0
ER
O:3
1
0002
R6:0
DN
O:3
2
0003
R6:0
EN
O:3
3

The **Control Parameter (R6:0)** is control structure to do the following:

- To Store the status byte of the instruction.
- Length of the sequencer file.
- And the instantaneous Position in the file.
- As sequencer does not have a designated file like a timer or Counter. The Data R:6 was created for instructions that don’t have a specific data file assigned.

Thank You