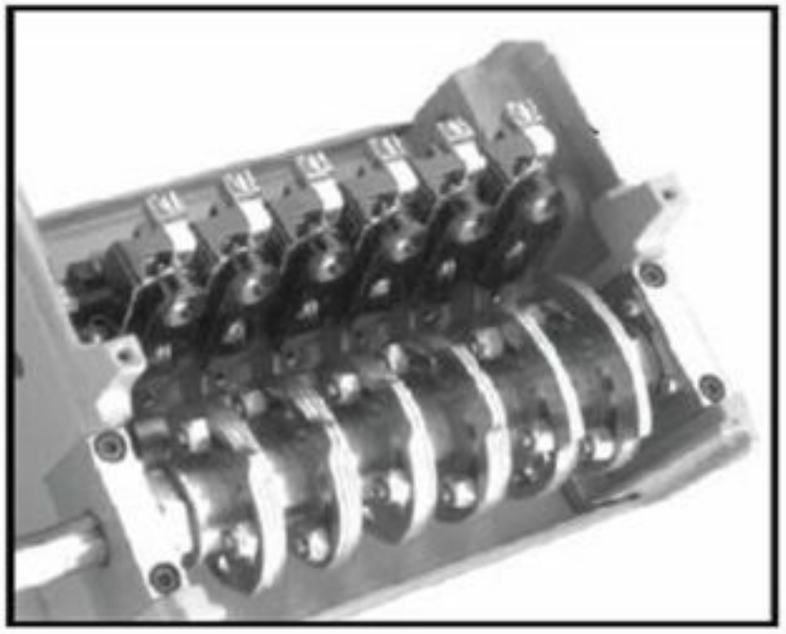
Chapter 12

Sequencer and Shift Register Instructions

12.1

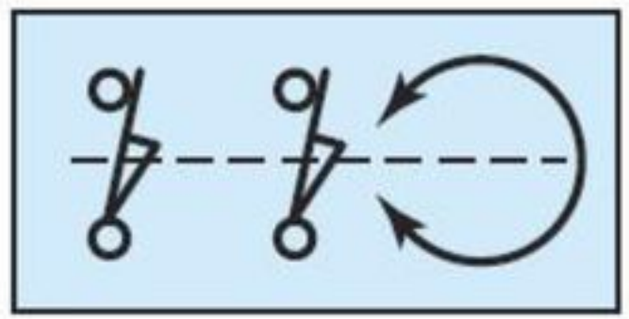
Mechanical Sequencers

Mechanical **sequencers are often referred to as drum switches, rotary switches, stepper switches, or cam switches.**



Switch Assembly

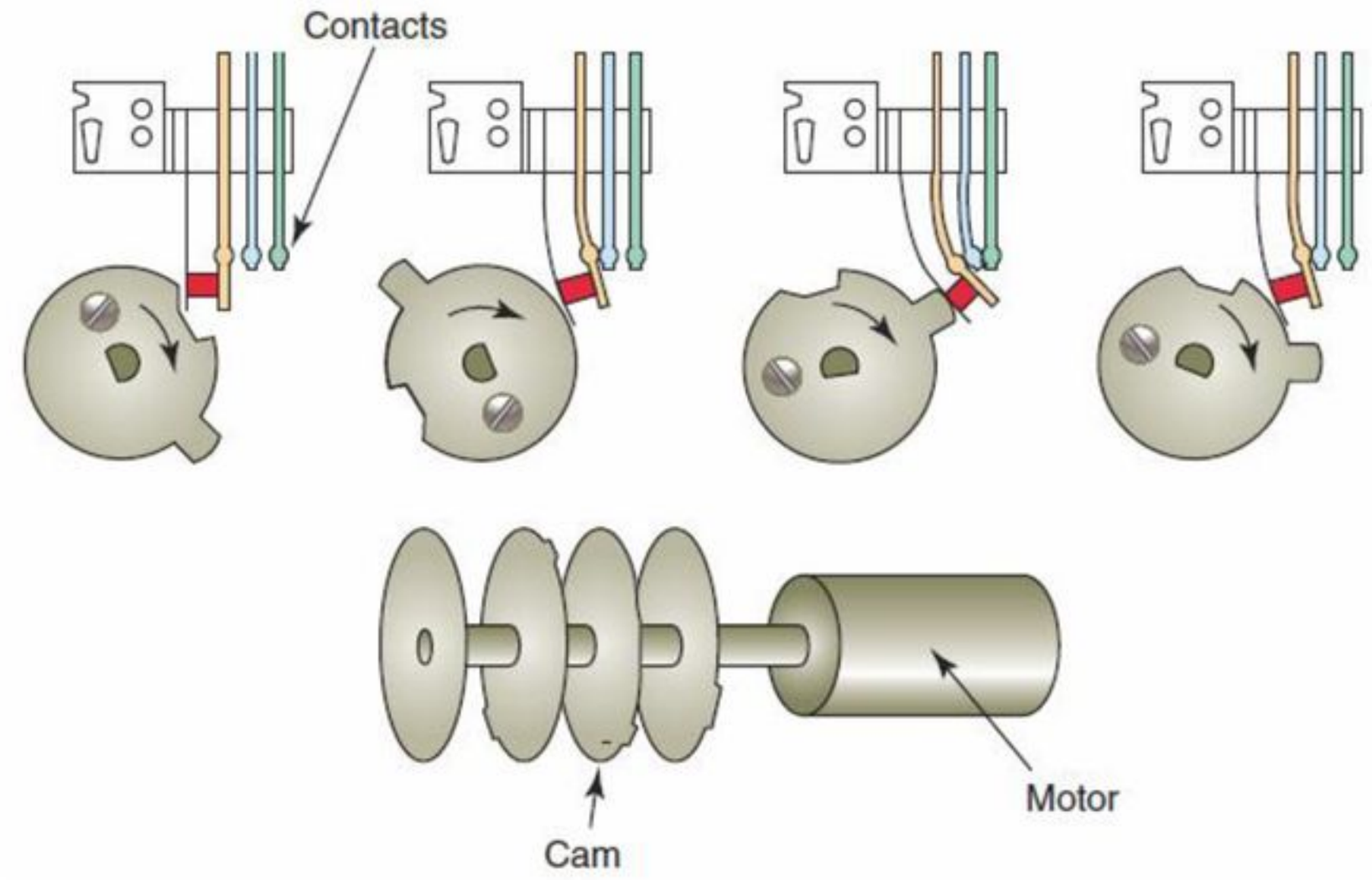
Enclosure



Symbol

Sequencers are used to control machinery that has a repetitive cycle of operation.

A *cam-operated* sequencer switch uses an electric motor to drive the cams.

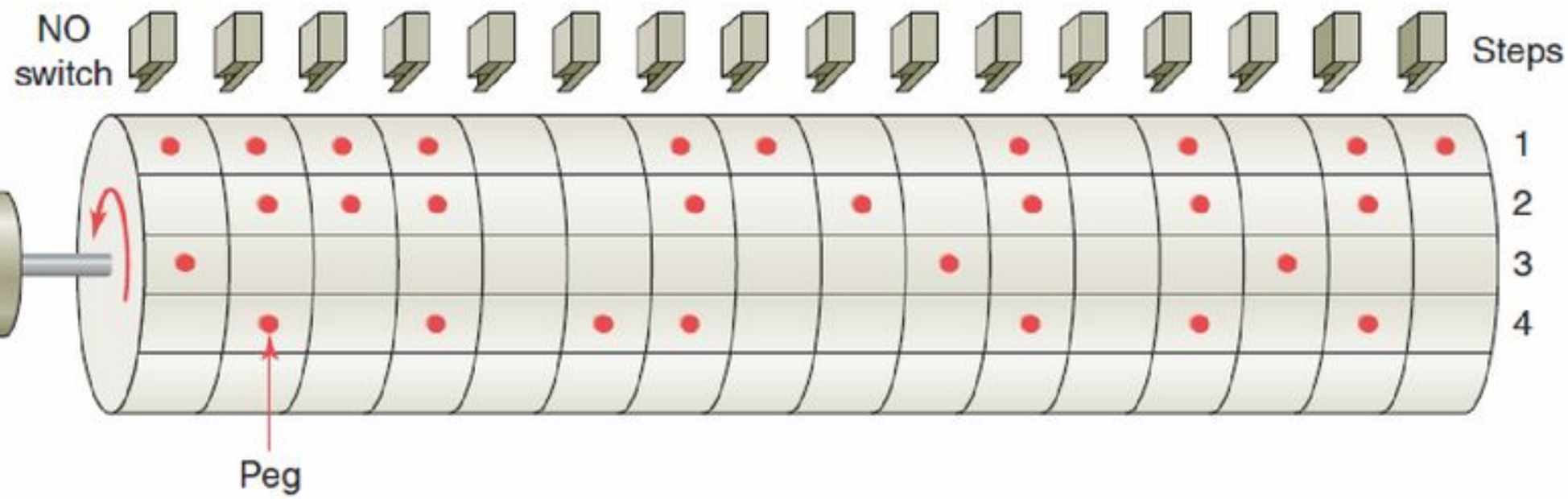


As the cams rotate, load devices connected to the contacts can change from an on to an off state, from an off to an on state, or remain at the same state.

Mechanical *drum-operated* sequencer switch.

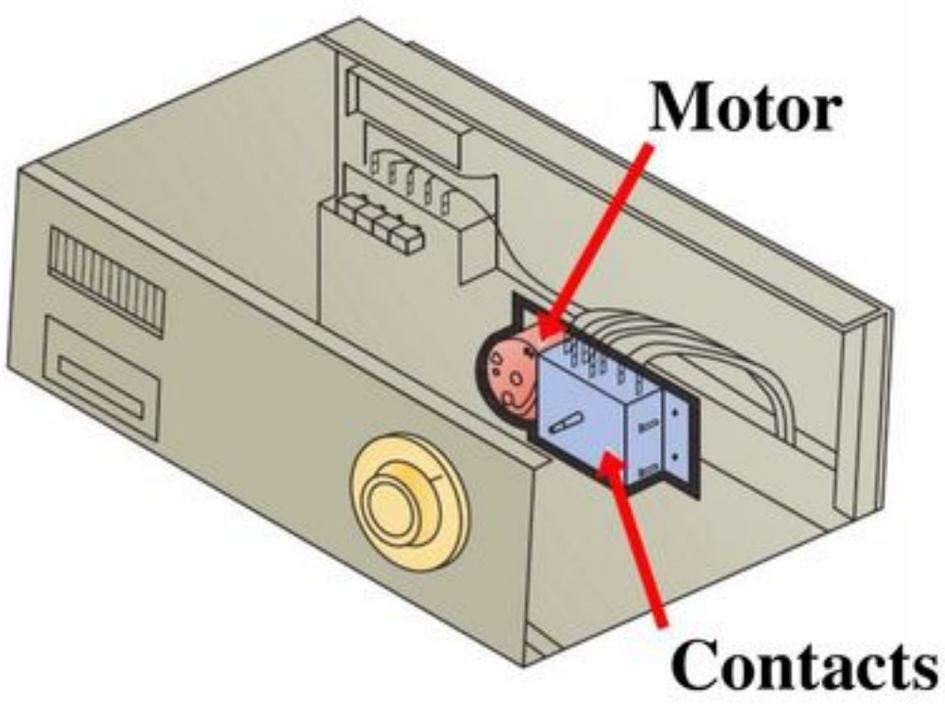
Equivalent sequencer data table

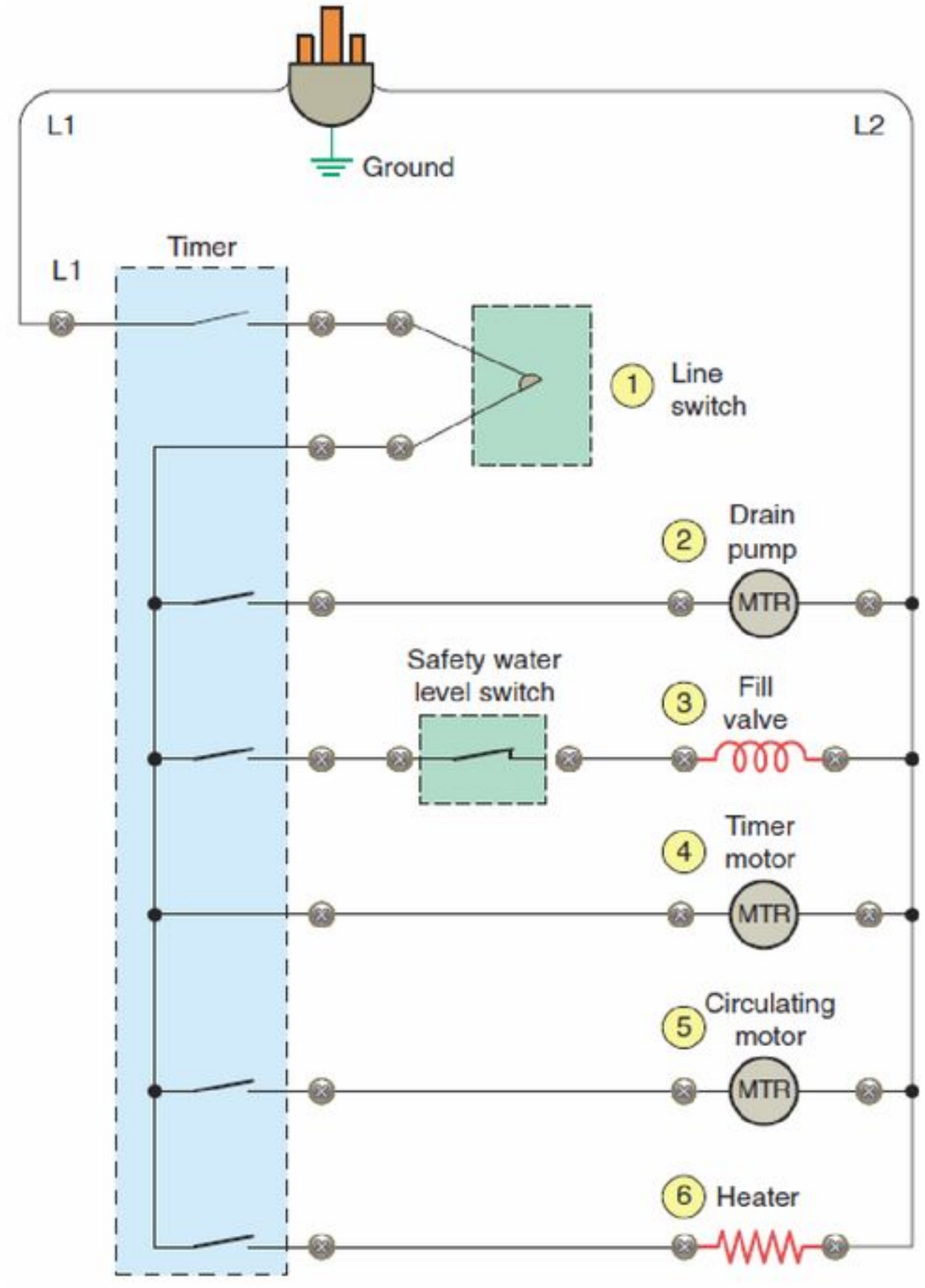
0	1	0	1	0	1	1	0	0	0	1	0	1	0	1	0	4
1	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	3
0	1	1	1	0	0	1	0	1	0	1	0	1	0	1	0	2
1	1	1	1	0	0	1	1	0	0	1	0	1	0	1	1	1



Each location where there was a peg is represented by a 1 (on), and the positions where there were no pegs are each represented by a 0 (off).

A washing machine is an example of the use of a timed sequencer, as are dryers and similar time-clock controlled devices.

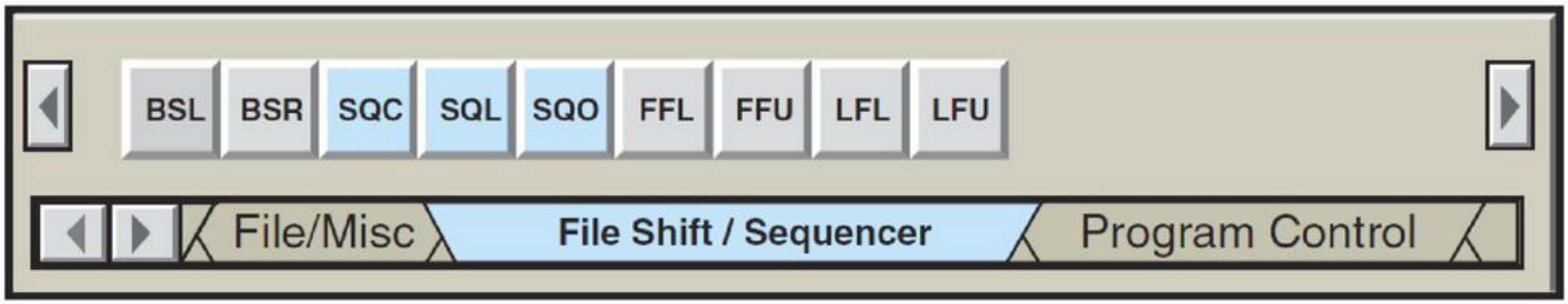




12.2

Sequencer Instructions

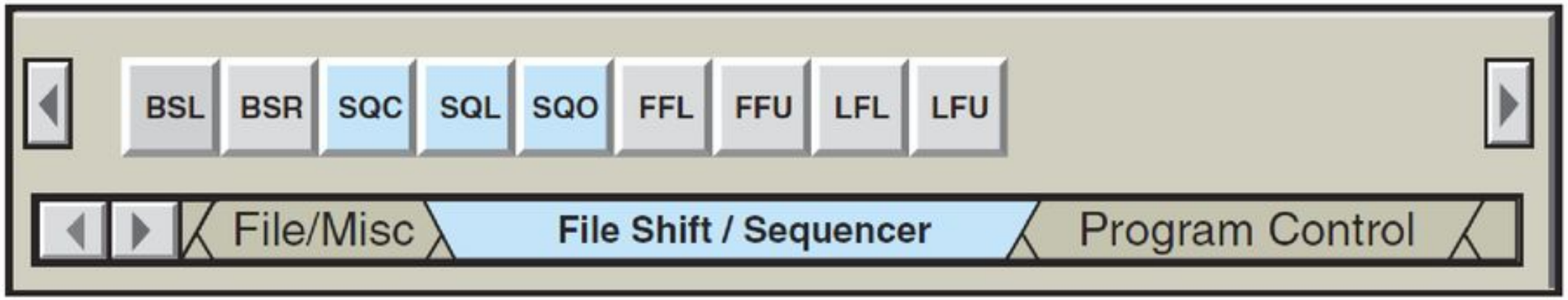
Programmed sequencers perform on or off patterns similar to that of a drum switch.



Positions

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	1	0	0	0	0	0	0	1	1	0	0	1	1
2	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
3	1	1	0	0	0	0	0	0	1	1	0	0	1	1	0	0
4	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
5	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0
6	0	0	1	1	0	0	0	0	1	1	1	1	1	1	1	1
7	0	0	0	1	0	0	0	0	0	0	0	1	1	1	1	1
8	0	1	0	1	0	0	0	0	0	1	0	1	0	1	0	1

Sequencer File

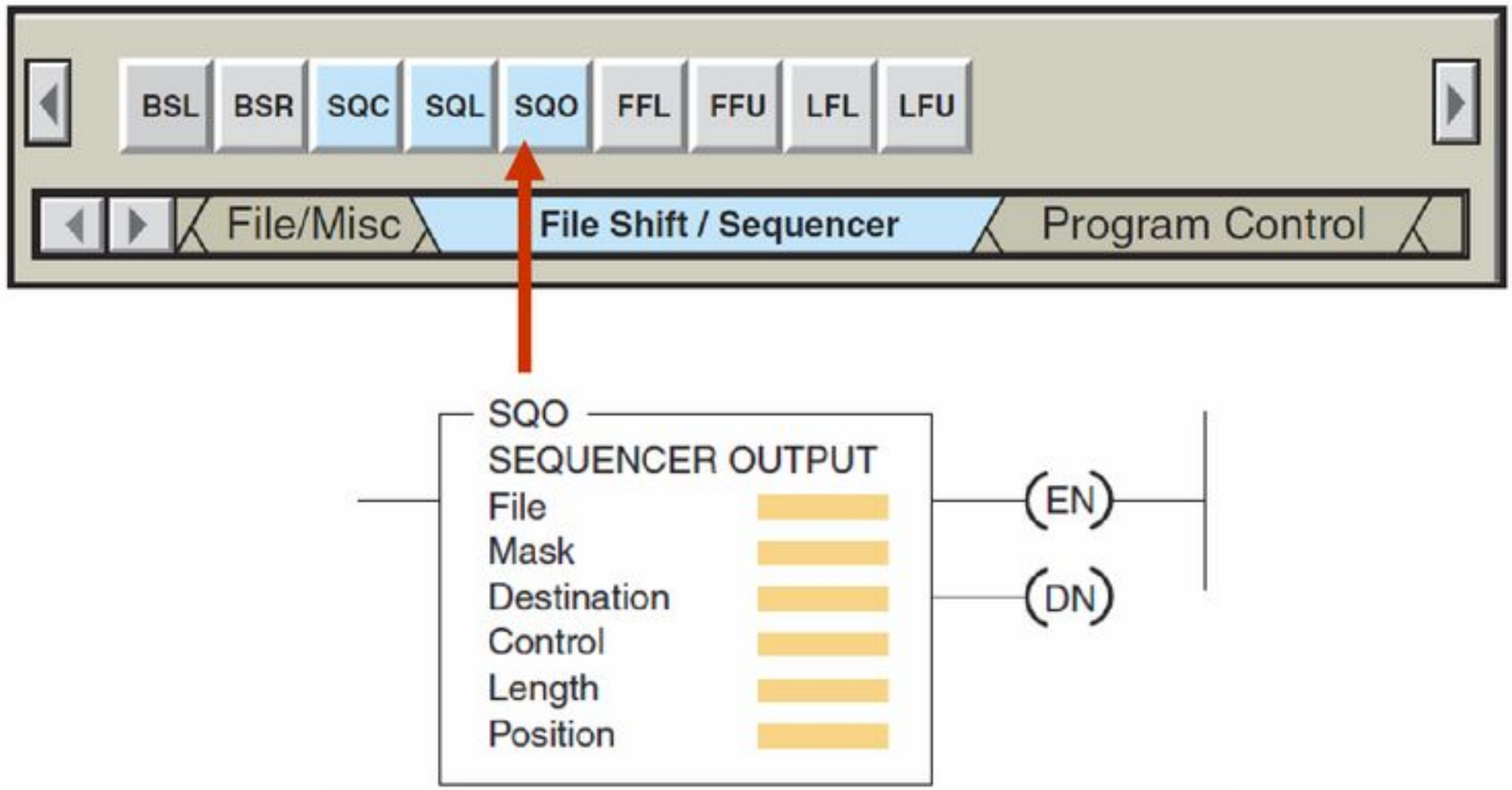


SQO (Sequencer Output) - uses a file to control various output devices.

SQL (Sequencer Load) - captures reference conditions by manually stepping the machine through its operating sequences.

SQC (Sequencer Compare) - compares bits from an input source file to corresponding bits from data words in a sequence file.

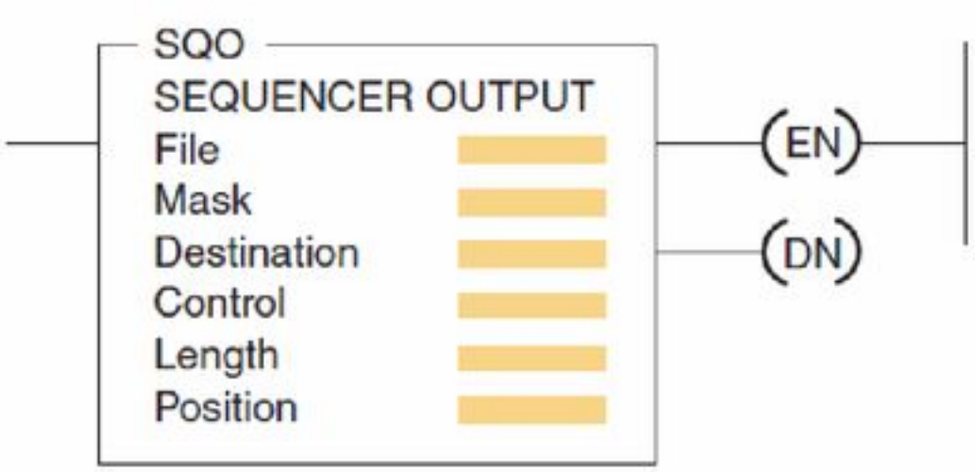
SQO (Sequencer Output) instruction.



File - starting address for the sequencer file.

Mask - bit pattern through which data are moved.

Destination - the address of the output word or file.

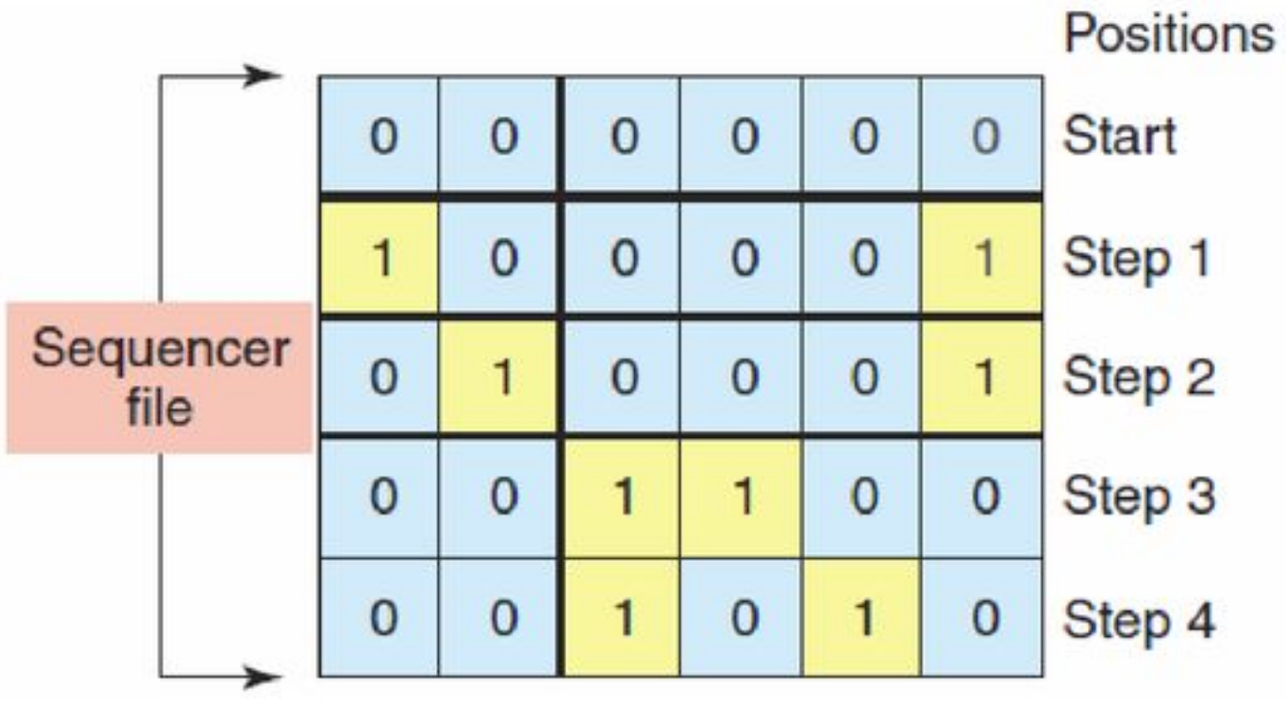


Control - contains the enable bit (EN), done bit (DN) and error bit (ER) parameters.

Length - the number of steps of the sequencer file starting at position 1. Position 0 is the start-up position. The actual file length will be 1 plus the file length entered in the instruction.

Position - the word location or step in the sequencer file from which the instruction moves data.

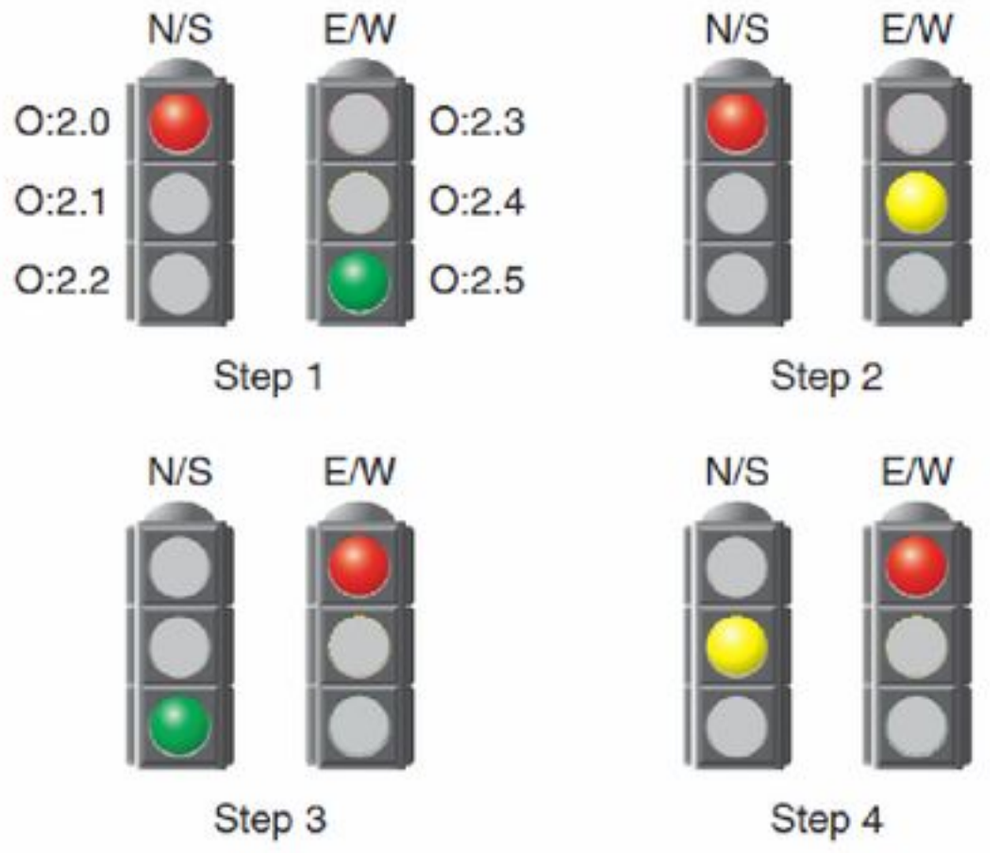
To program a sequencer, binary information is first entered into the *sequencer file* made up of a series of *consecutive memory words*.

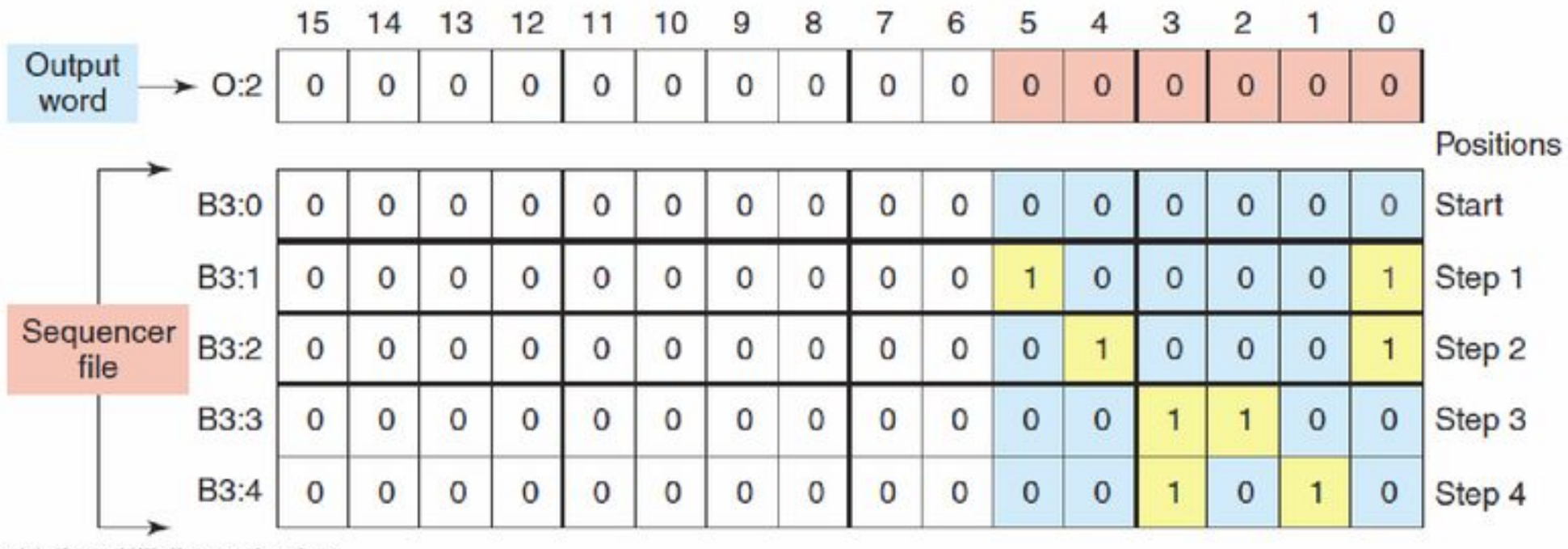


The sequencer bit file contains bits representing the output action required for each step of the sequence.

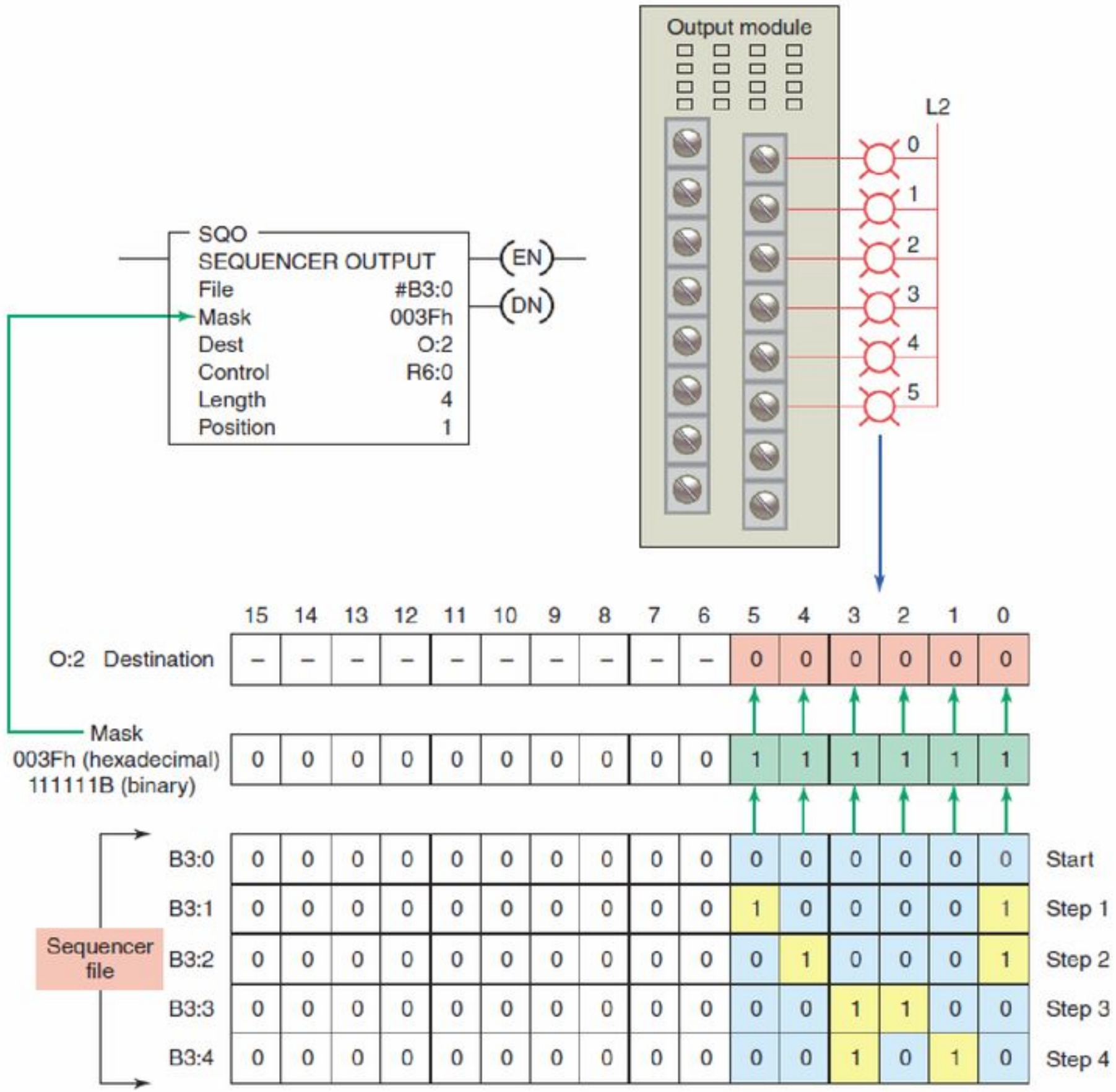
As the sequencer advances through the steps, binary information is transferred from the sequencer file to the output word.

Sequencer used to control traffic in two directions.

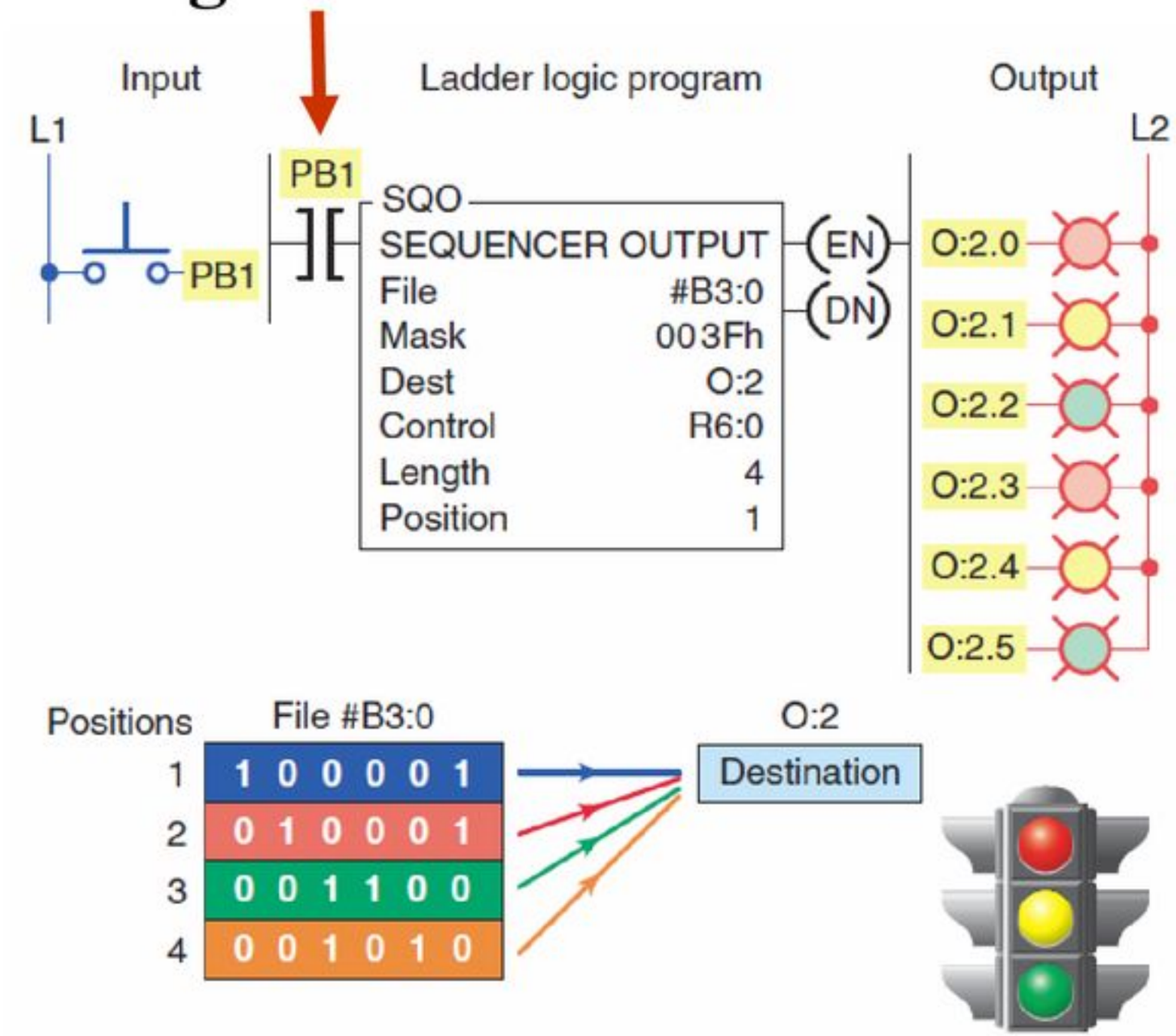




Sequencer program used to control traffic.

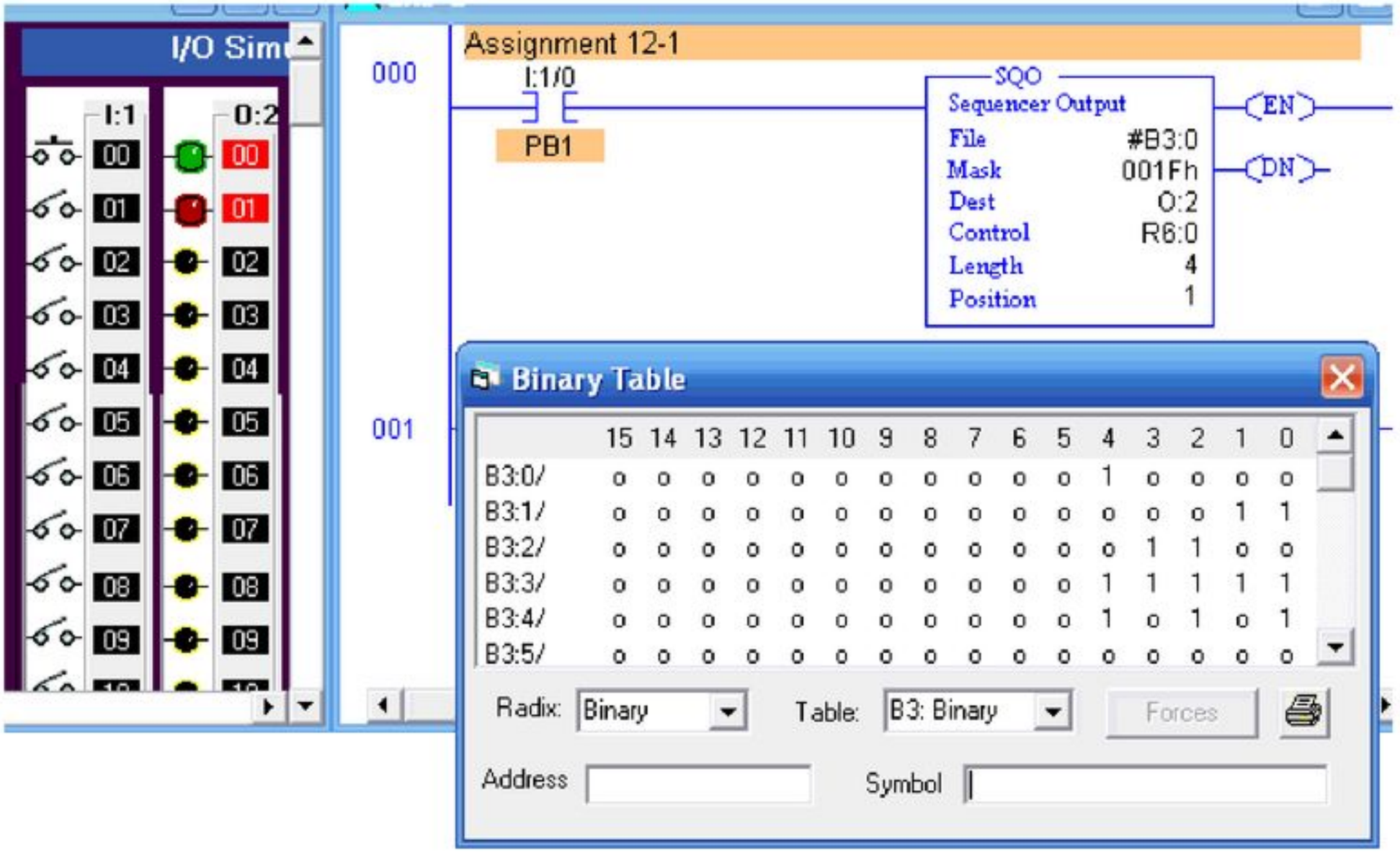


The sequencer output instruction requires preceding logic on the rung where it is located.



When this logic goes from false to true, it triggers the sequencer to perform its functions.

Simulation of the stop light program.



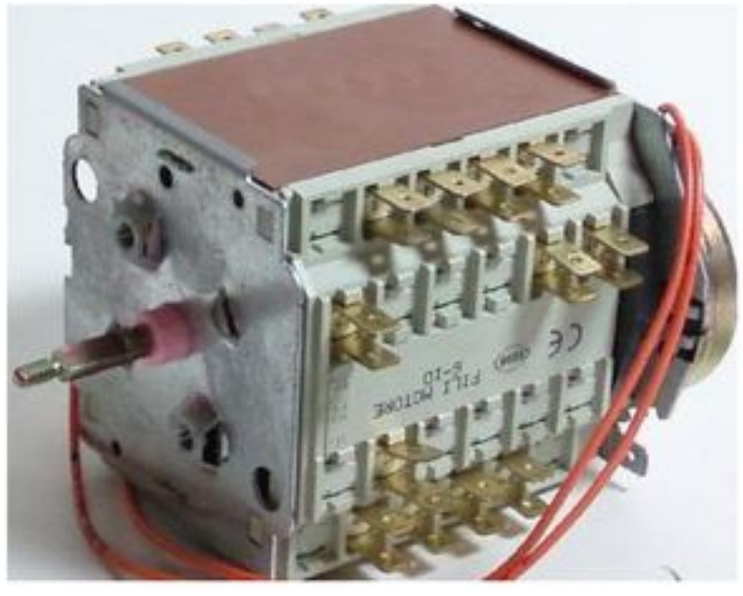
12.3

Sequencer Programs

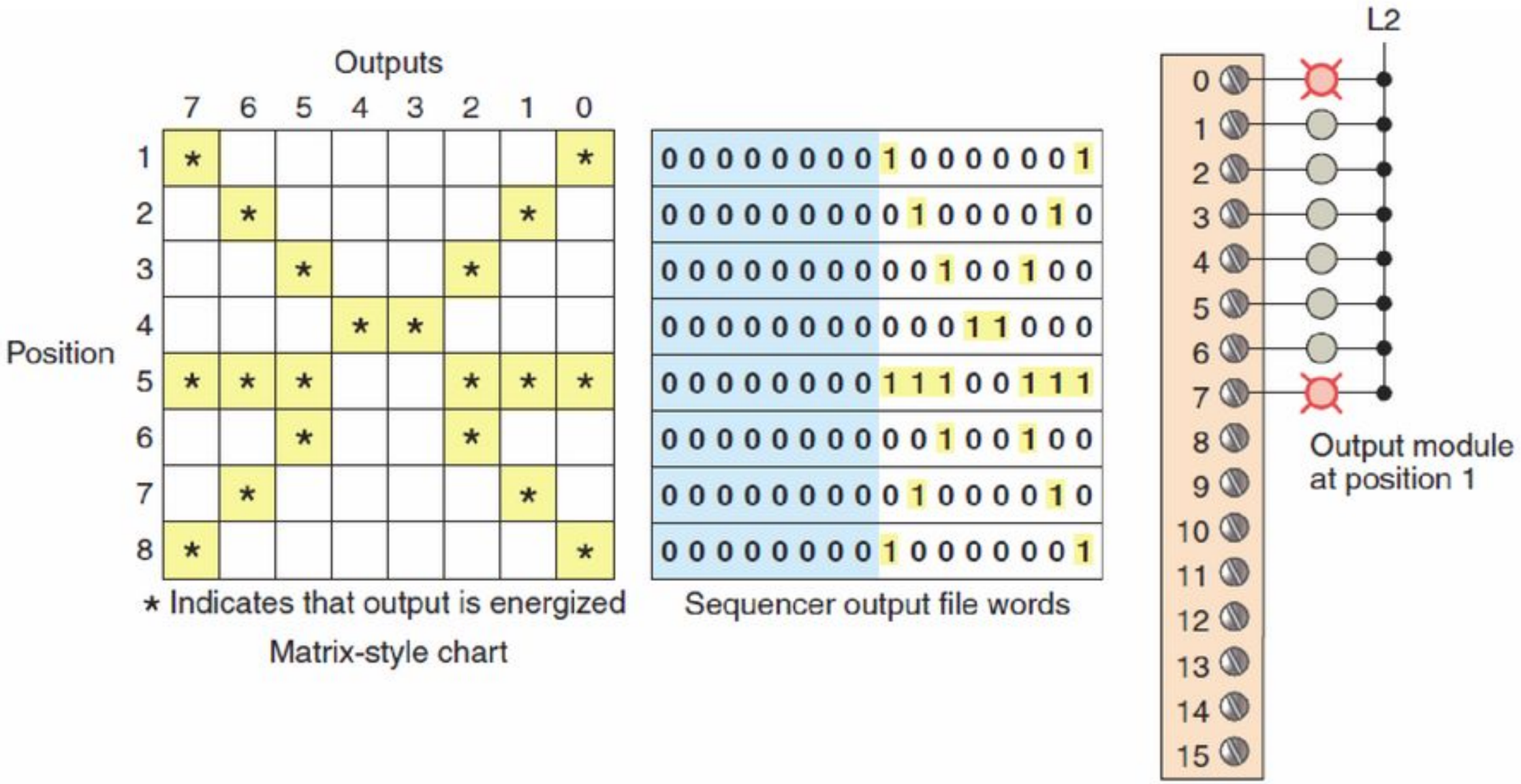
A sequencer program can be *event-driven* or *time-driven.*

An event-driven sequencer operates similarly to a mechanical stepper switch that increments by one step for each pulse applied to it.

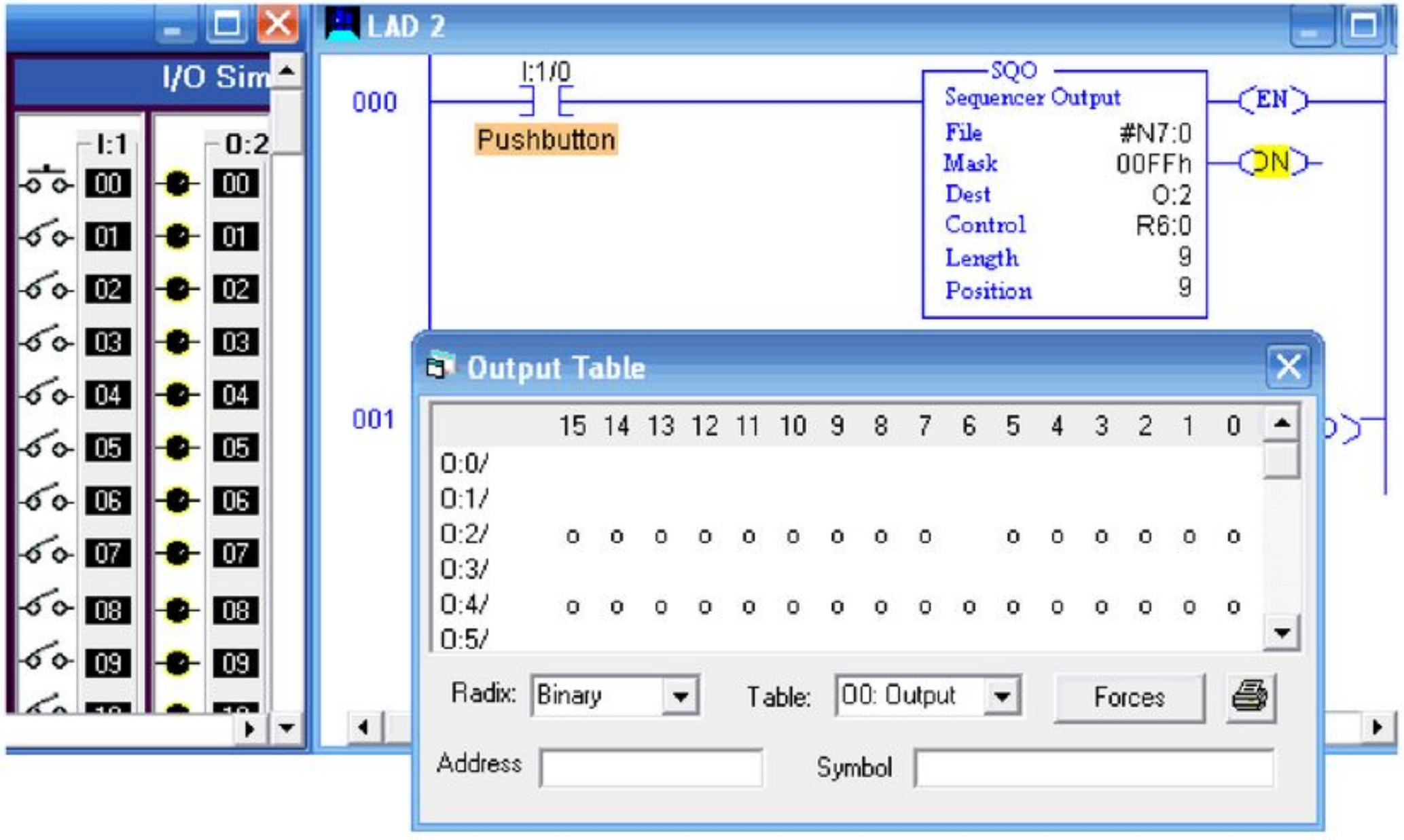
A time-driven sequencer operates similarly to a mechanical drum switch that increments automatically after a preset time period.



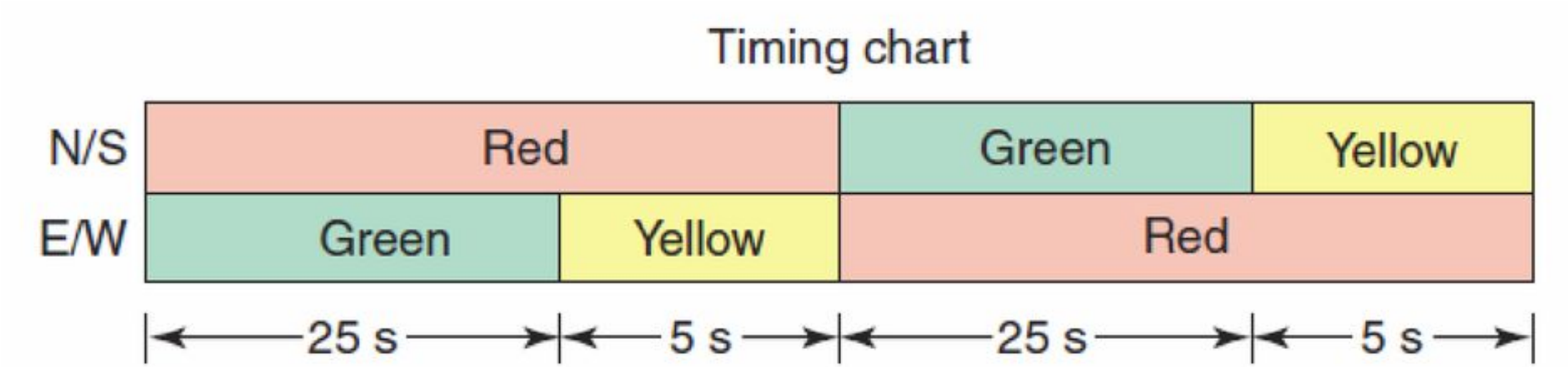
A *sequencer chart* is a table that lists the sequence of operation of the outputs controlled by the sequencer instruction.



Simulated sequencer chart event driven program.

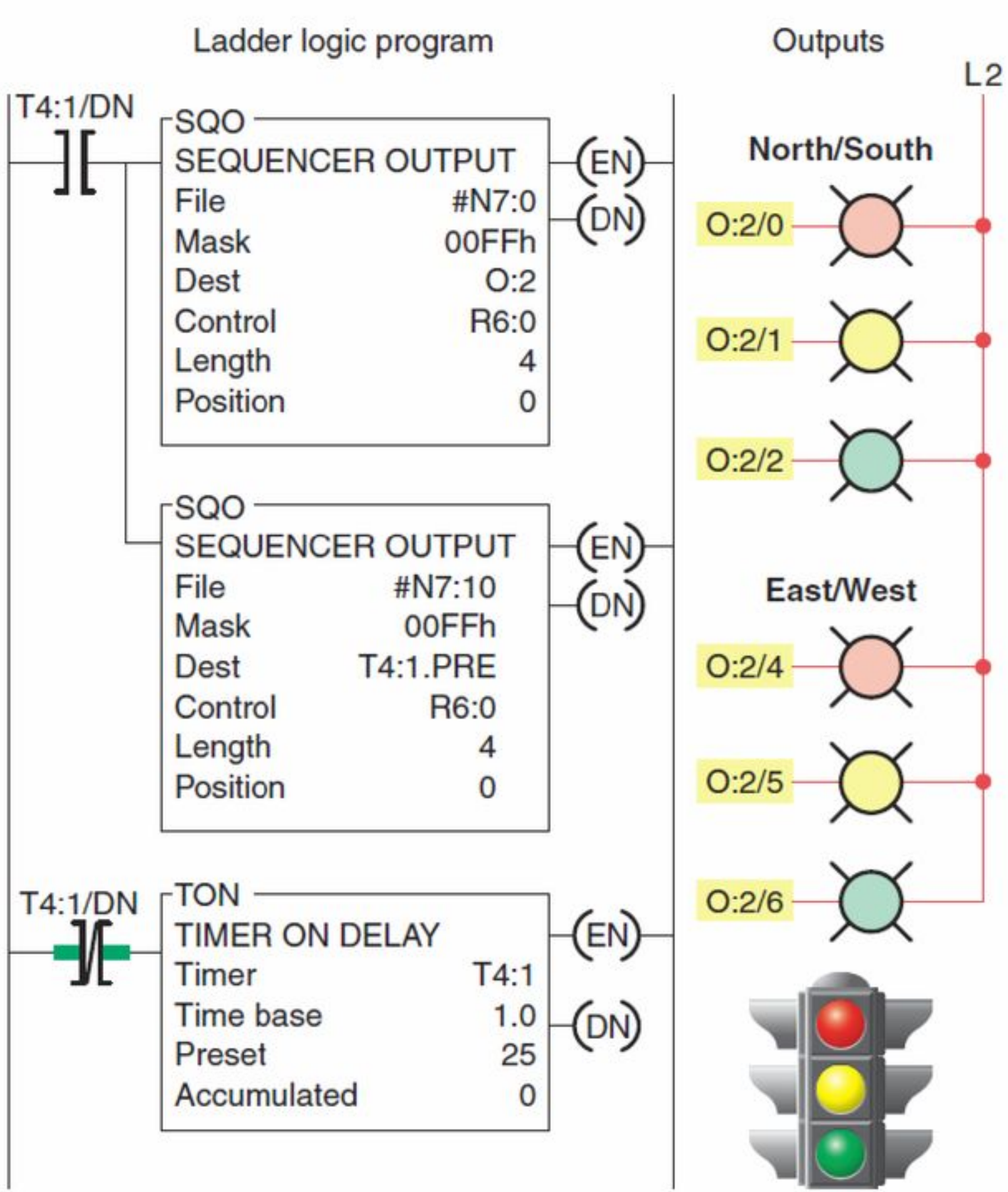


Time-driven sequencer with timed *steps that are not all the same*.

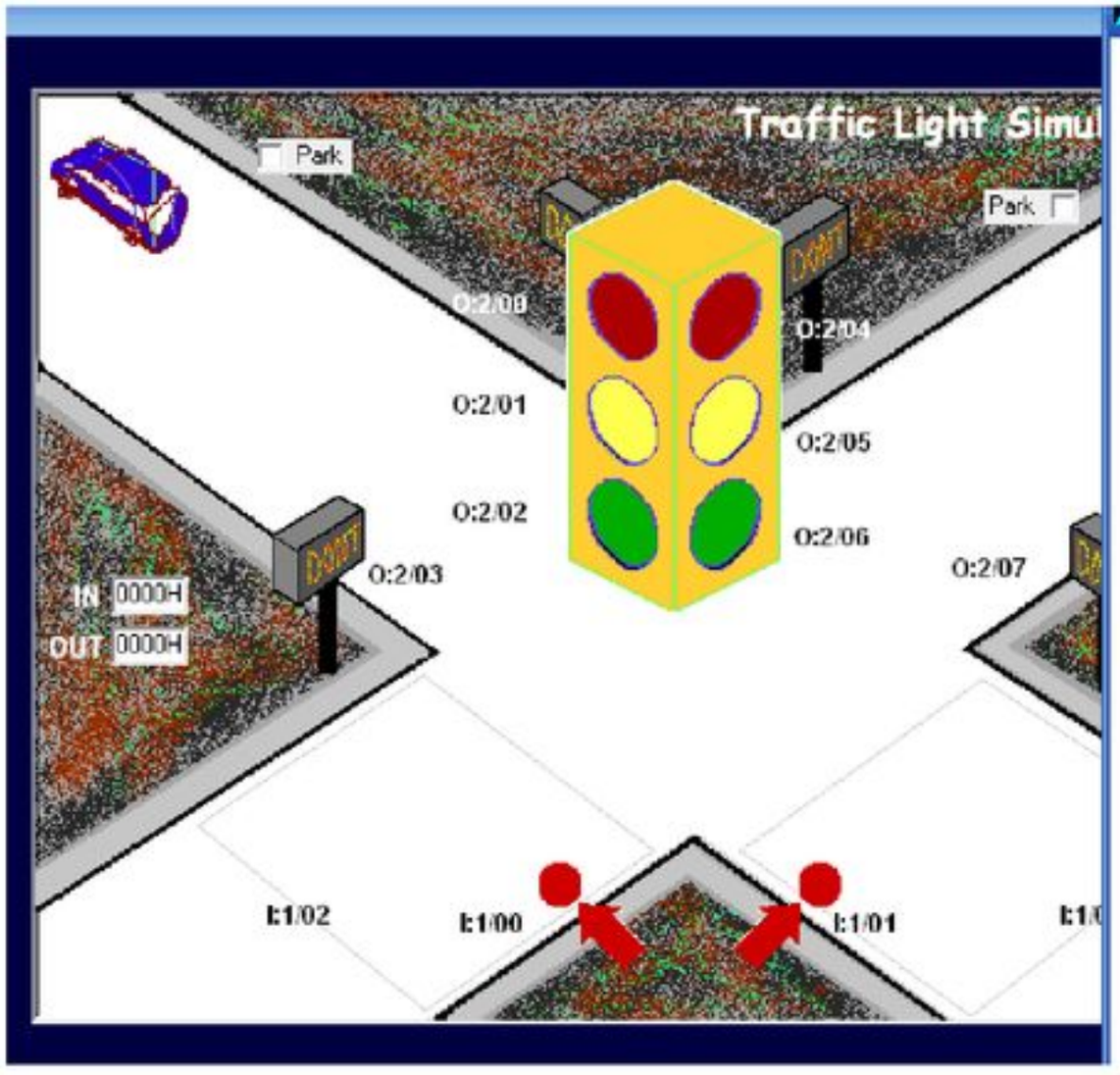


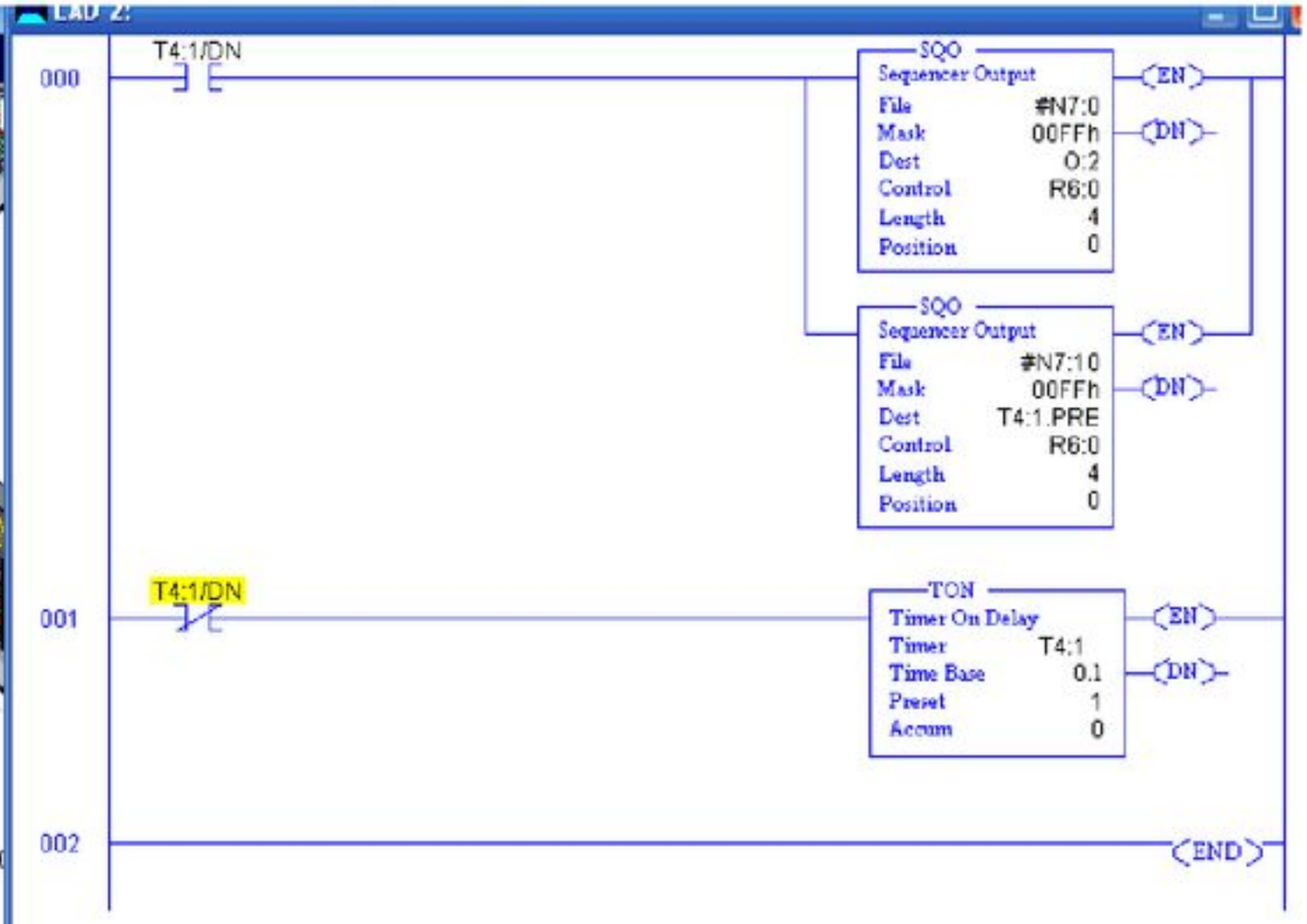
This sequencer program is used for automatic traffic light control at a four-way intersection.

Traffic Light Program



Simulated traffic light program.





Integer Table

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
N7:0/	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o
N7:1/	o	o	o	o	o	o	o	o	o	1	o	o	o	o	o	1
N7:2/	o	o	o	o	o	o	o	o	o	o	1	o	o	o	o	1
N7:3/	o	o	o	o	o	o	o	o	o	o	o	1	o	1	o	o
N7:4/	o	o	o	o	o	o	o	o	o	o	o	1	o	o	1	o

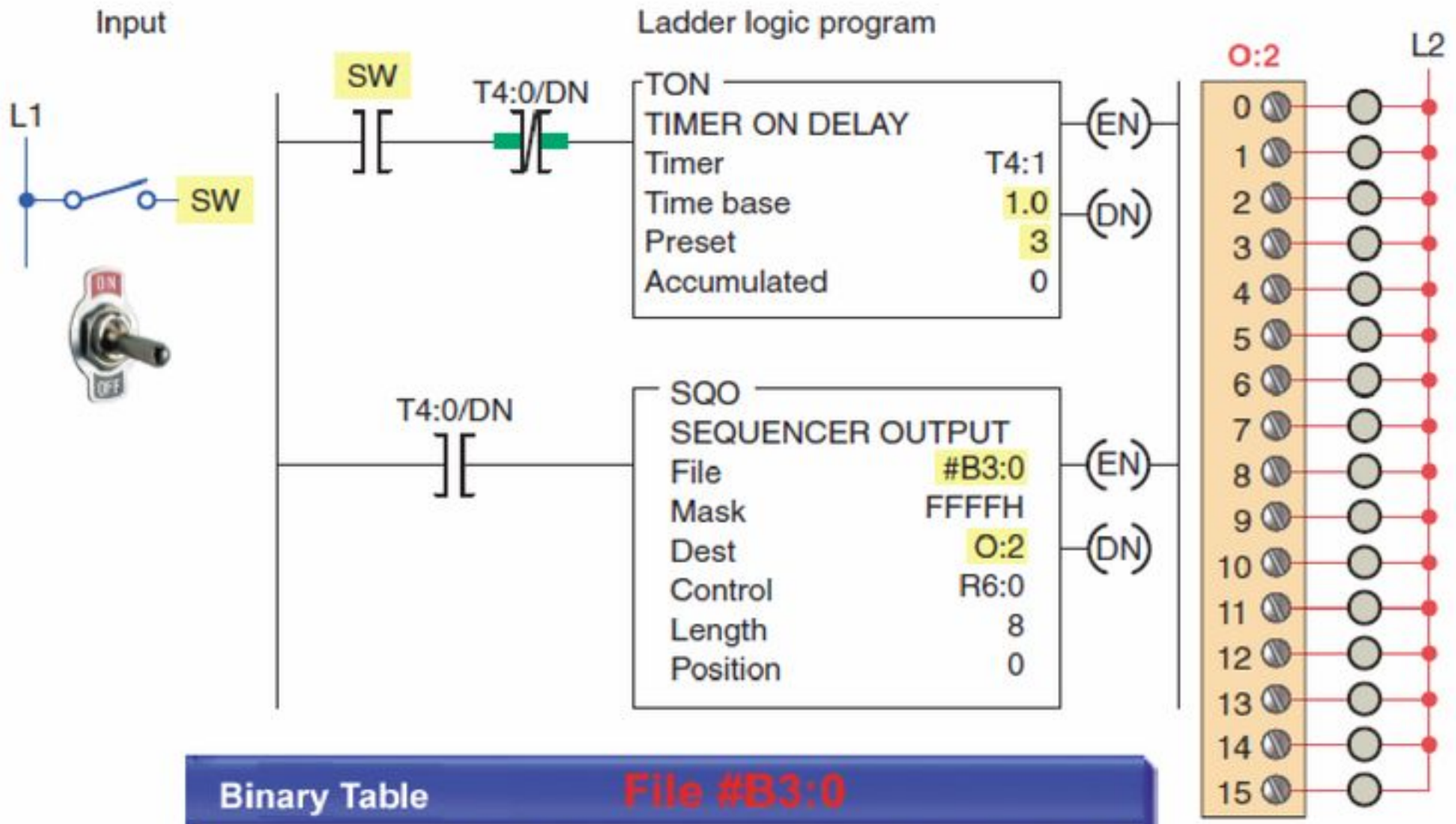
Radix: Binary Table: N7: Integer

Integer Table

	Value
N7:10	0
N7:11	250
N7:12	50
N7:13	250
N7:14	50

Radix: Decimal

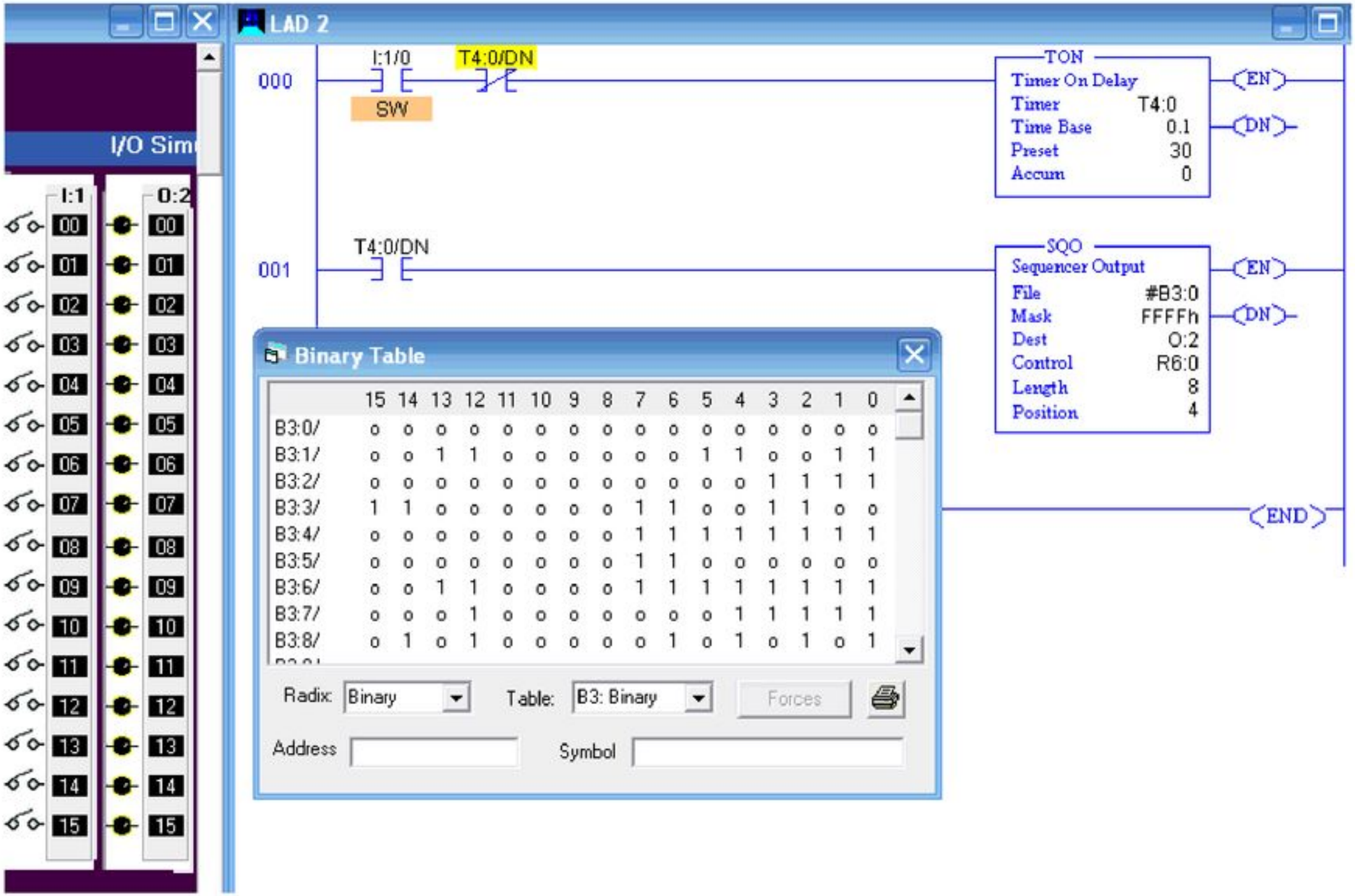
Sequencer with constant time intervals.



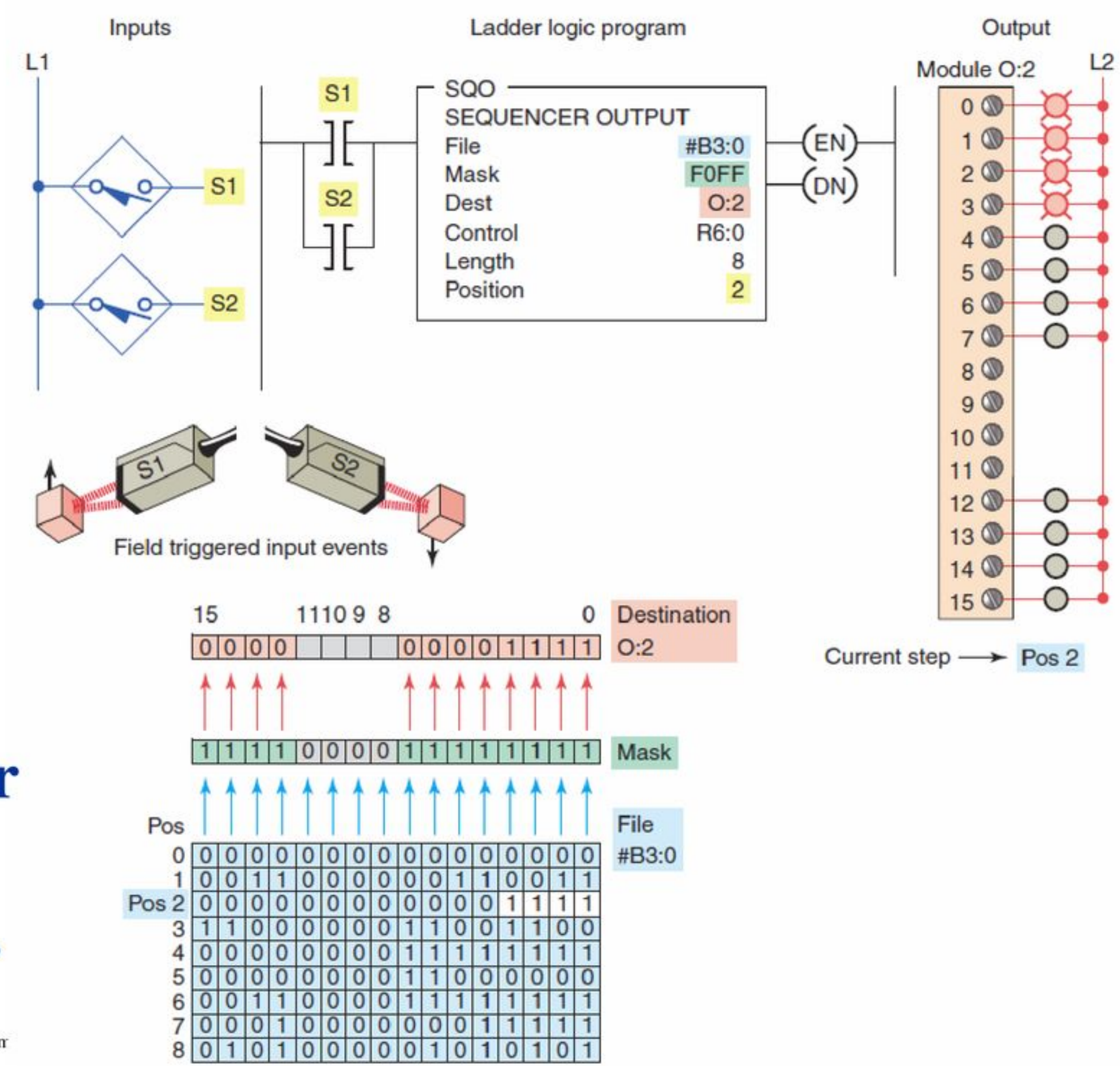
Binary Table **File #B3:0**

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
B3:0/	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B3:/1	0	0	1	1	0	0	0	0	0	0	1	1	0	0	1	1
B3:/2	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
B3:/3	1	1	0	0	0	0	0	0	1	1	0	0	1	1	0	0
B3:/4	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
B3:/5	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0
B3:/6	0	0	1	1	0	0	0	0	1	1	1	1	1	1	1	1
B3:/7	0	0	0	1	0	0	0	0	0	0	0	1	1	1	1	1
B3:/8	0	1	0	1	0	0	0	0	0	1	0	1	0	1	0	1

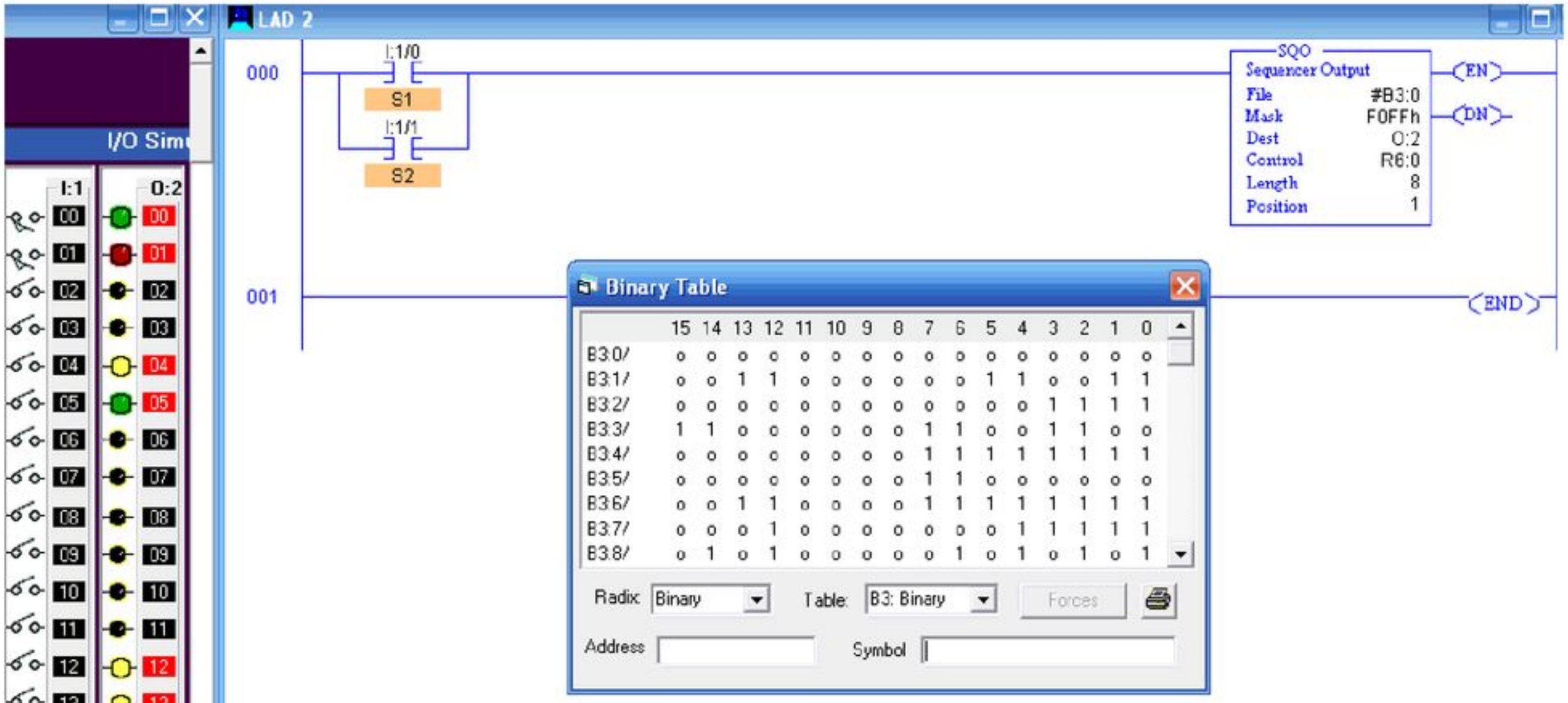
Simulated sequencer with constant time intervals.



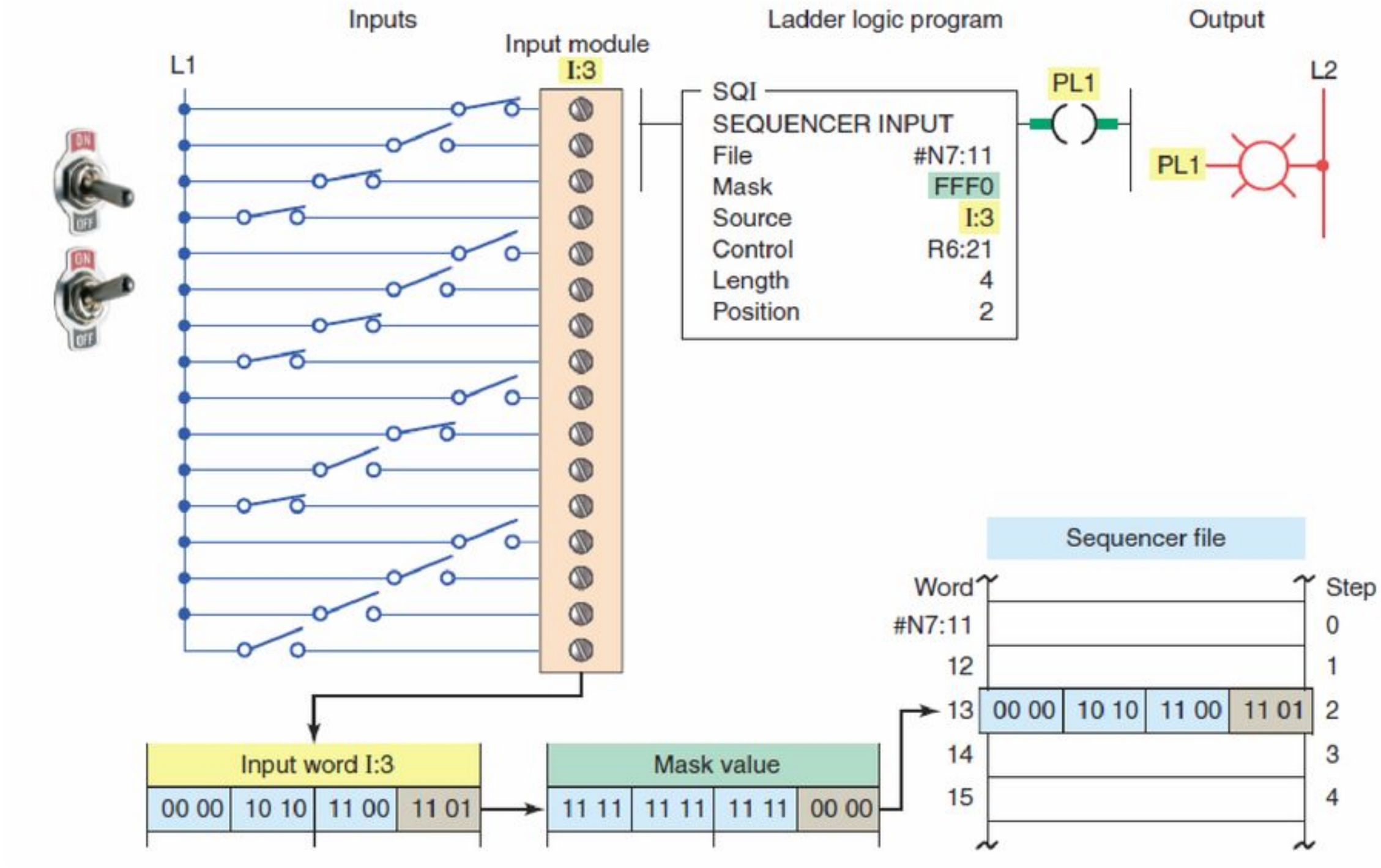
Event-driven sequencer output program.



Simulated event-driven sequencer output program.

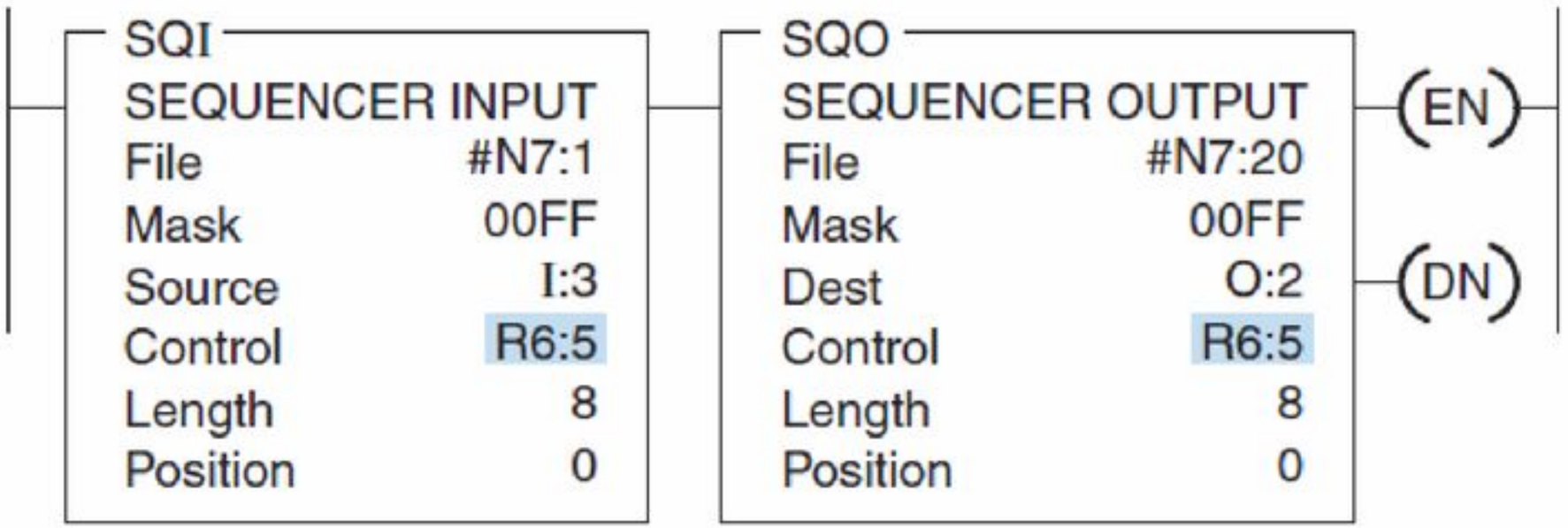


The sequencer *input* (SQI) instruction allows input data to be compared for equality against data stored in the sequencer file.



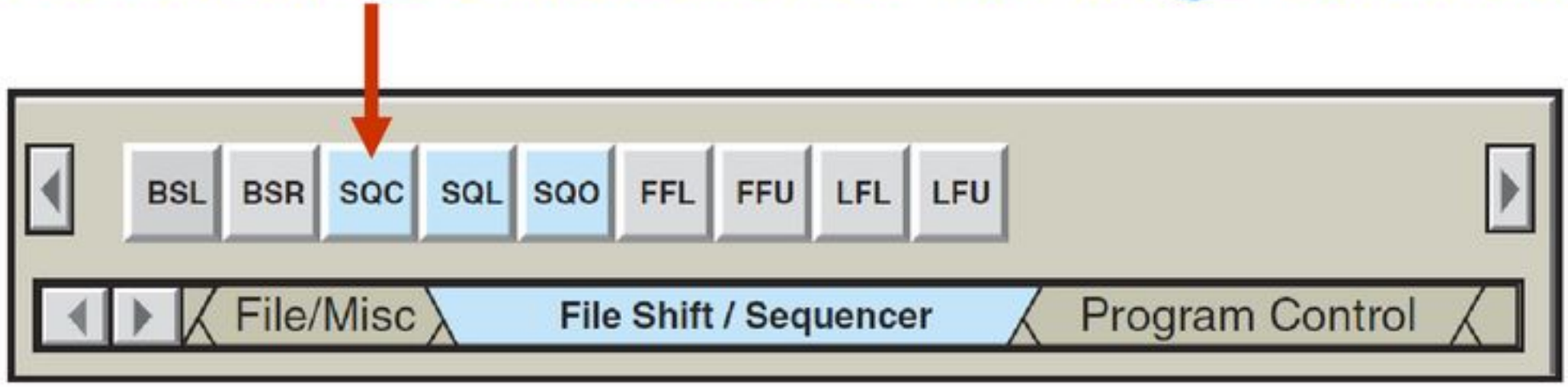
When the SQI is paired with an SQO instruction with identical control addresses the position is incremented by the SQO instruction for both.

Ladder logic program



This type of programming technique is used to monitor and control, respectively, a sequential operation.

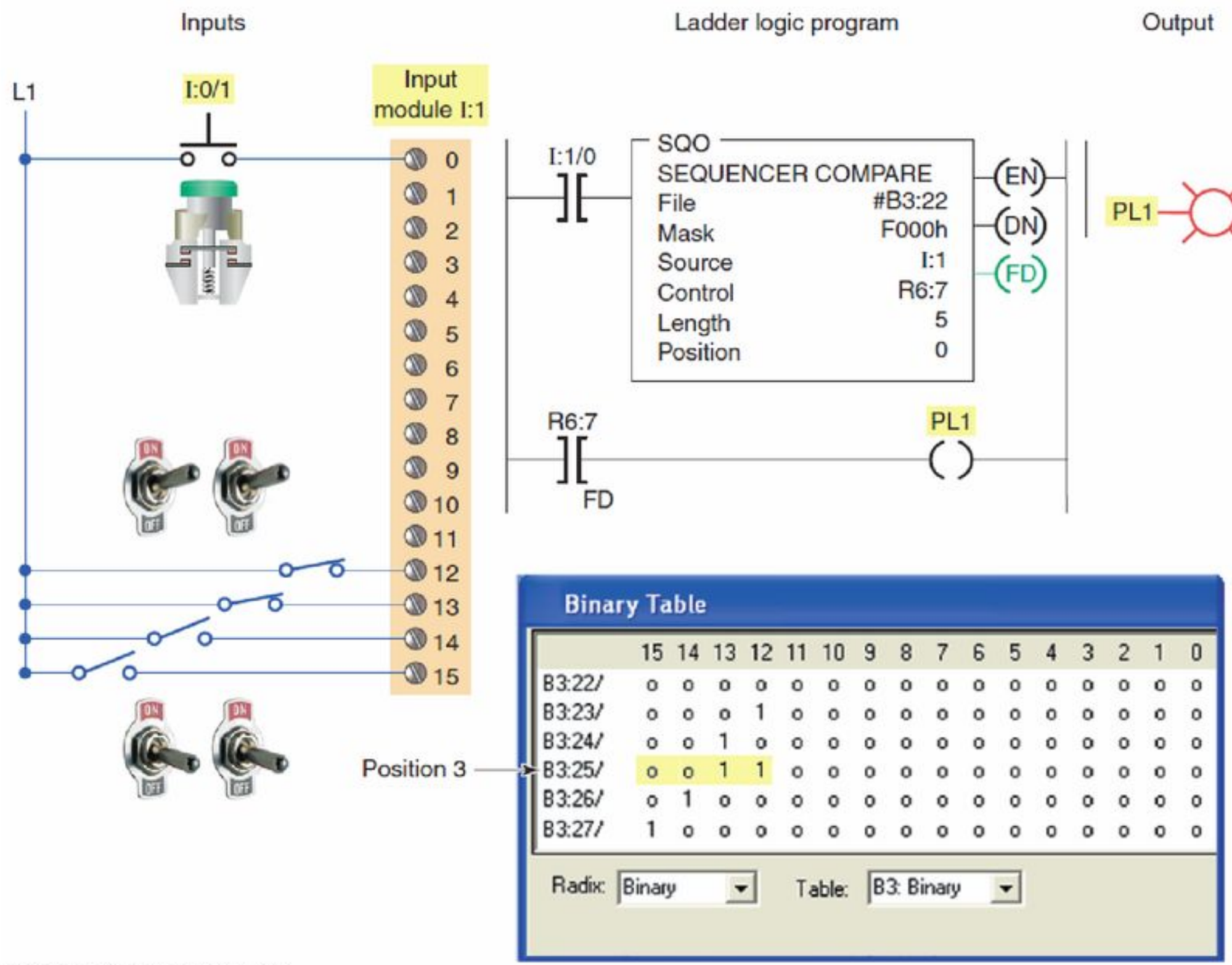
The SLC 500's *sequencer compare (SQC)* instructions are similar to the SQI instruction.



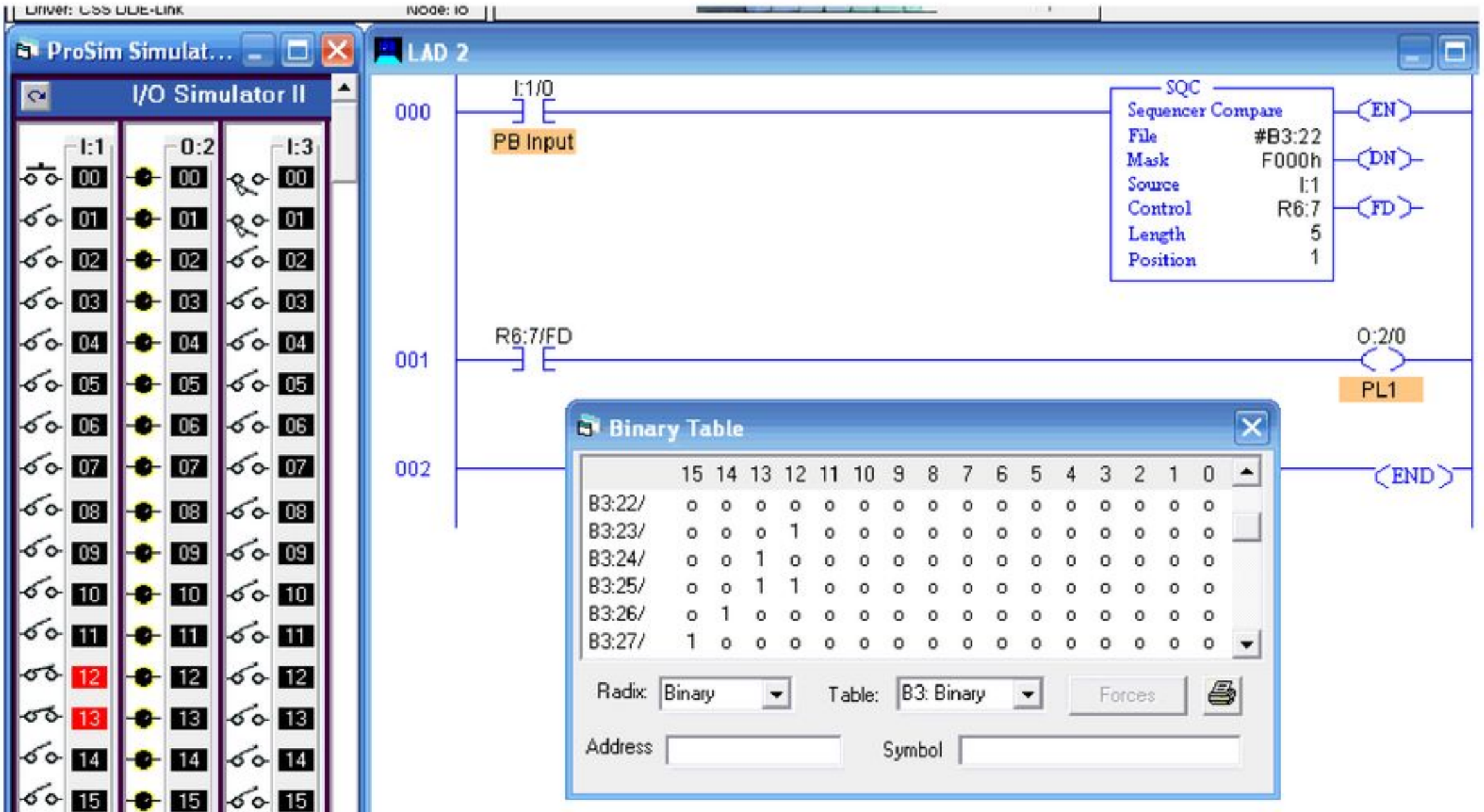
Differences between the two include:

- **The SQC instruction is an output rather than an input instruction.**
- **The SQC instruction increments the position parameter**
- **The SQC instruction has an additional status bit – the *found bit (FD).* When the source pattern matches the sequencer file word the FD is set to 1.**

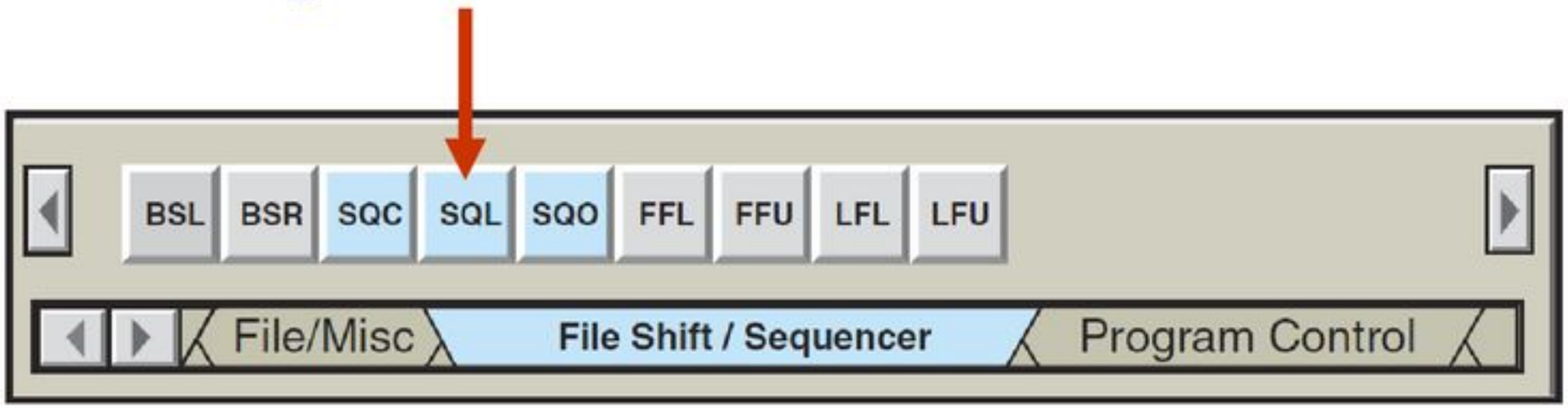
Sequencer compare (SQC) instruction program.



Simulated sequencer compare (SQC) program.

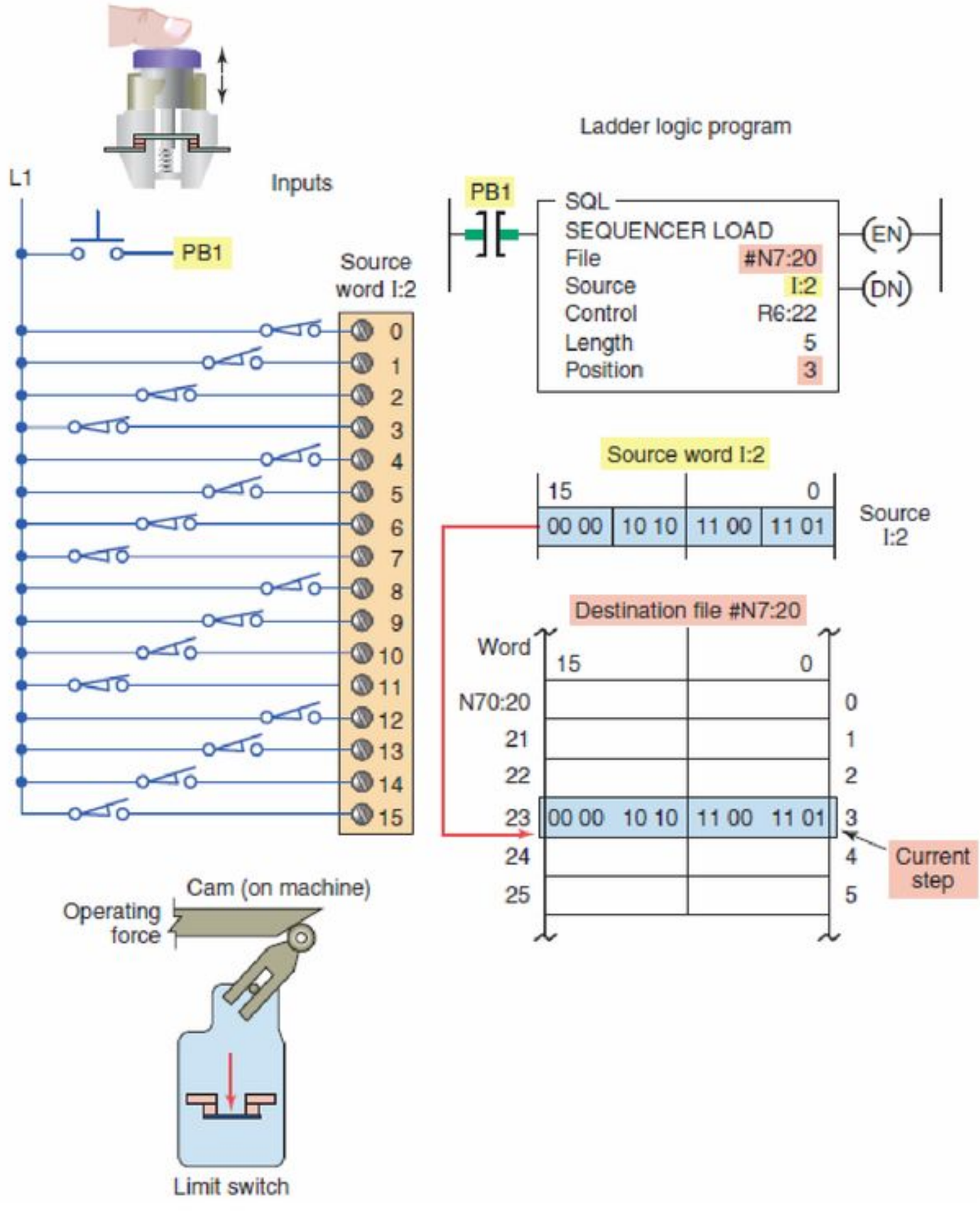


The *sequencer load (SQL)* instruction is used to read the PLC input module and store the input data in the sequencer file.

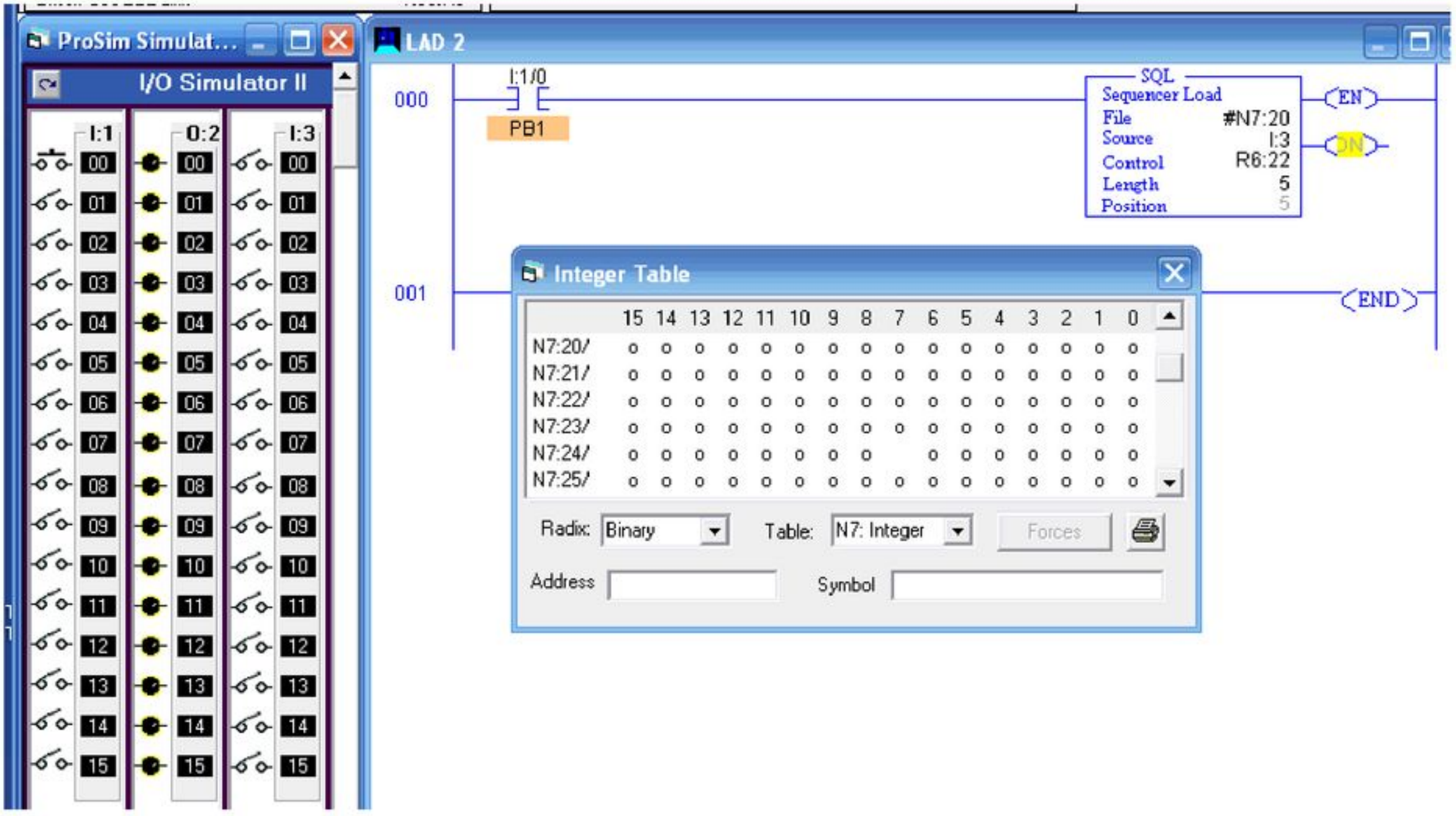


For example, a robot may be jogged manually through its sequence of operation, with its input devices read at each step.

Sequencer load (SQL) instruction program.



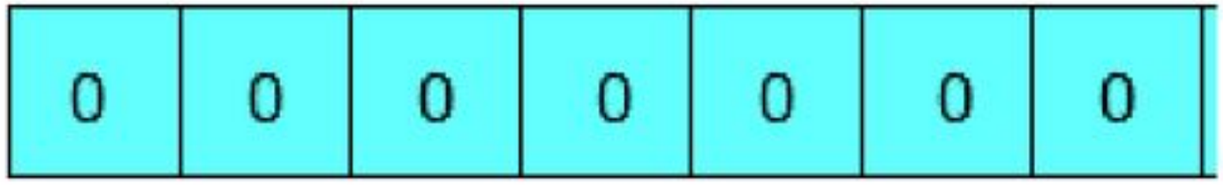
Simulated sequencer load (SQL) program.



12.4

Bit Shift Registers

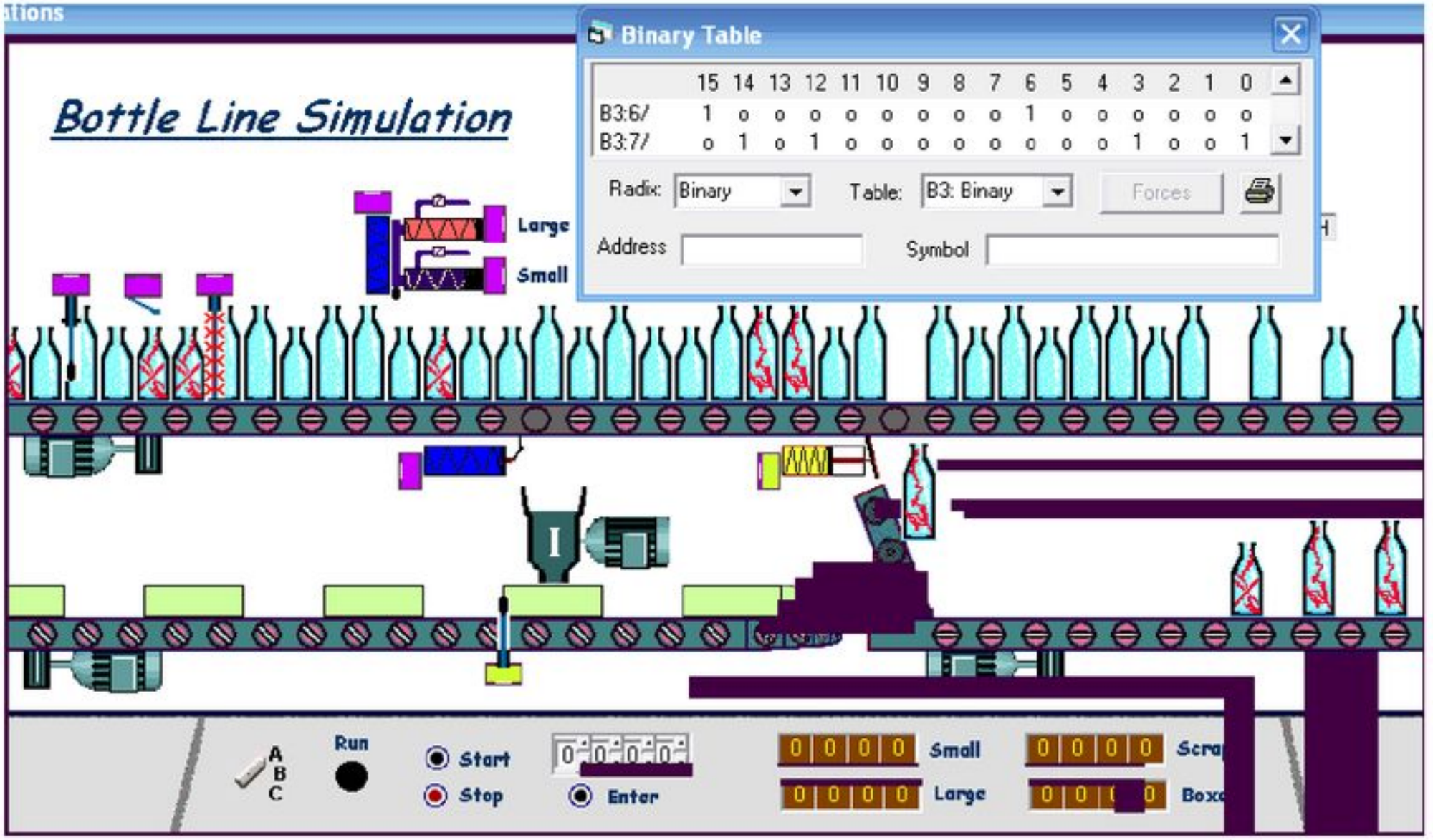
A bit *shift register* is a register that allows the shifting of bits through a single register or group of registers.



- **The bit shift register shifts bits serially (from bit to bit) through an array in an orderly fashion.**

- **A shift register can be used to simulate the movement, or *track* the flow, of parts and information.**

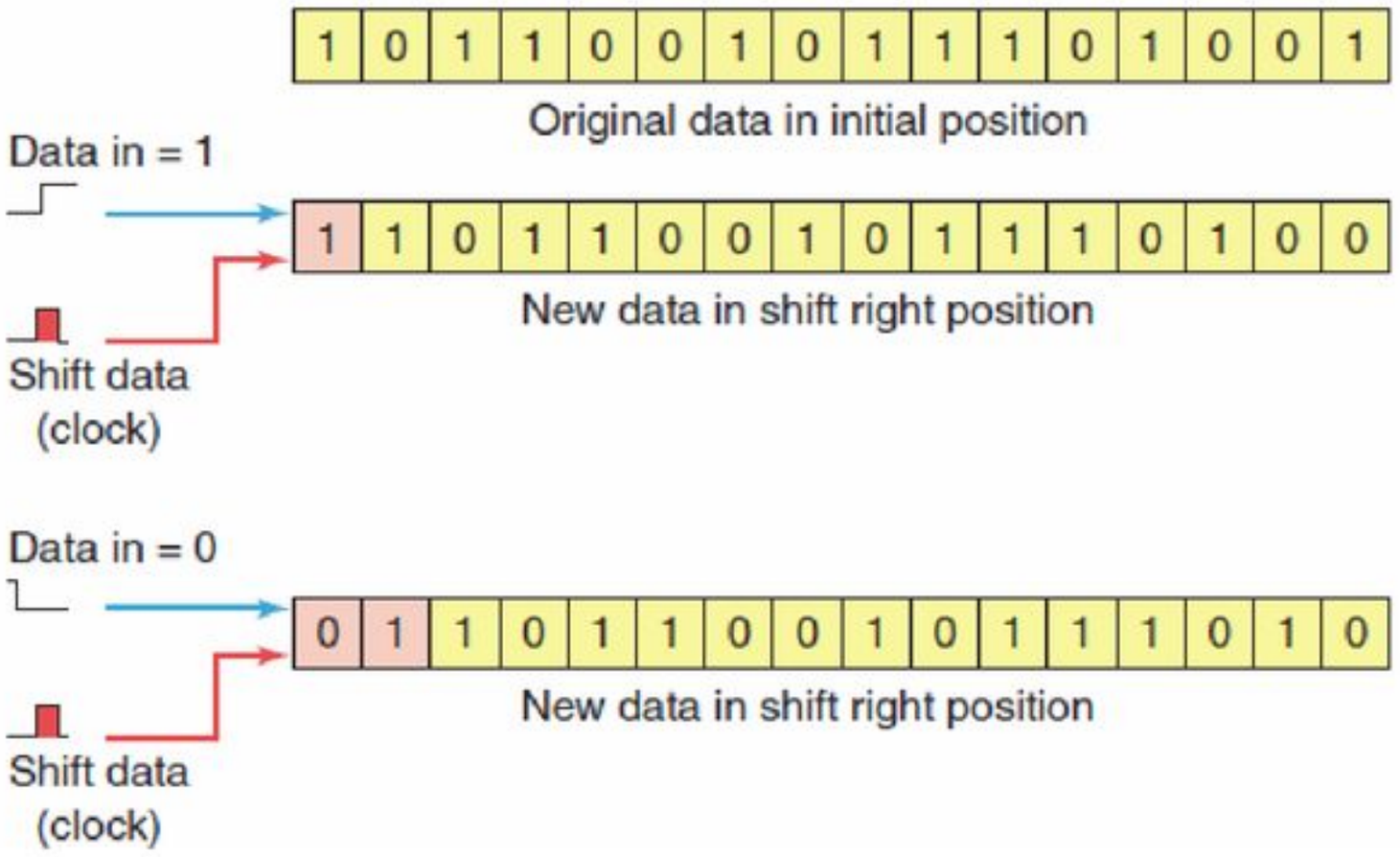
Bit *shift register simulation*



➢Sensor S1 detects all bottles while sensor S2 detects broken bottles only.

➢ The bit shift instruction is used to automatically energize the broken bottle divert gate

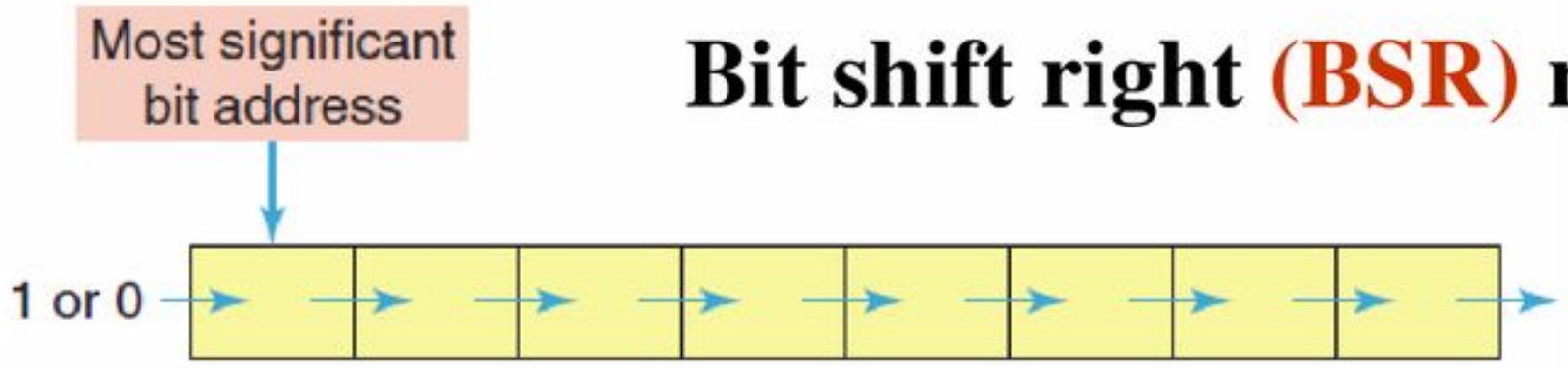
Basic concept of a shift register.



- A **shift pulse** or **clock** causes each bit in the shift register to move one position to the right.

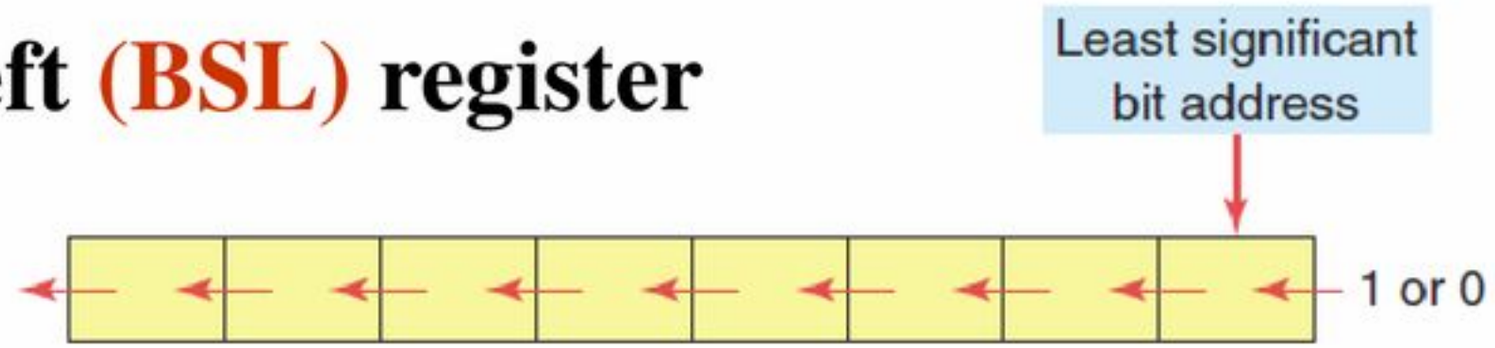
- The **data** (1 or 0), can represent the presence or absence of parts.

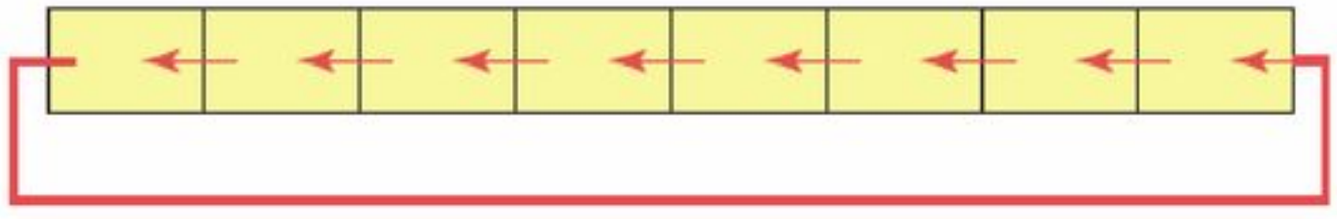
Types of shift registers.



Bit shift right (BSR) register

Bit shift left (BSL) register



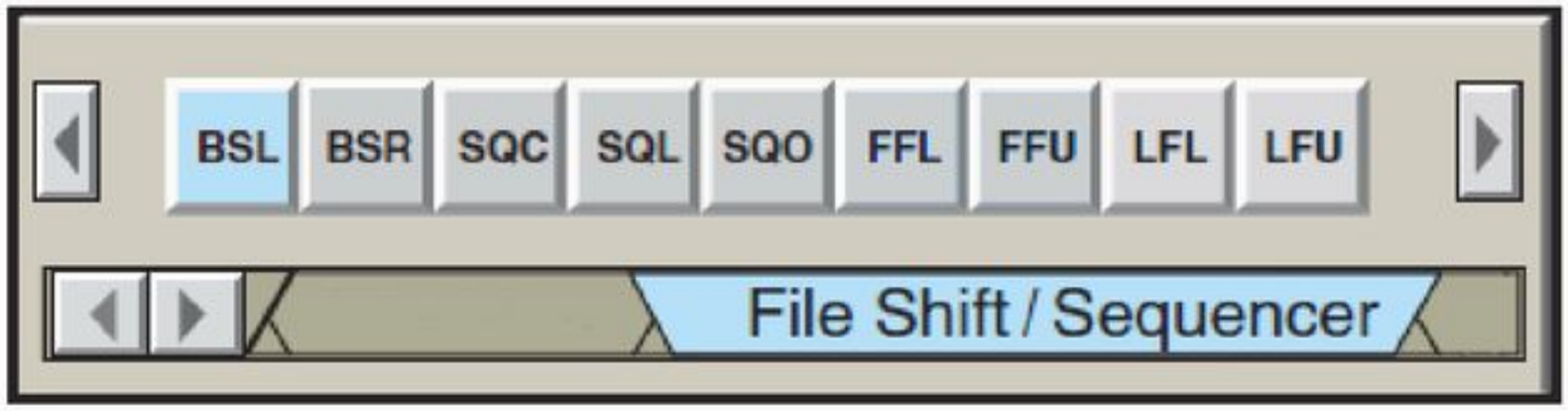


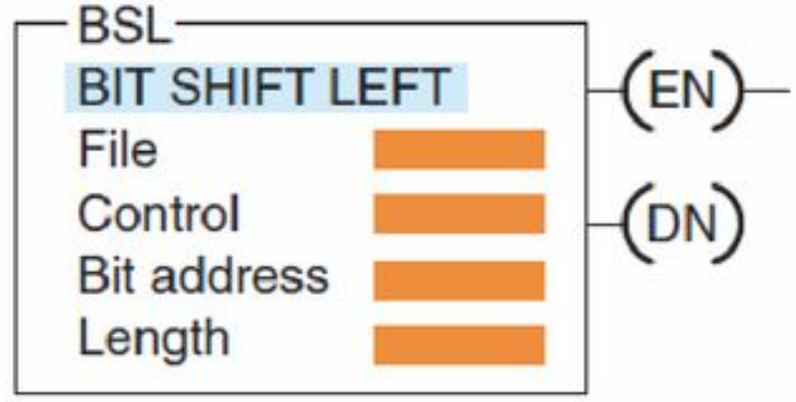
Wraparound or circulating shift register

When working with a bit shift register you identify each bit by its position in the register.

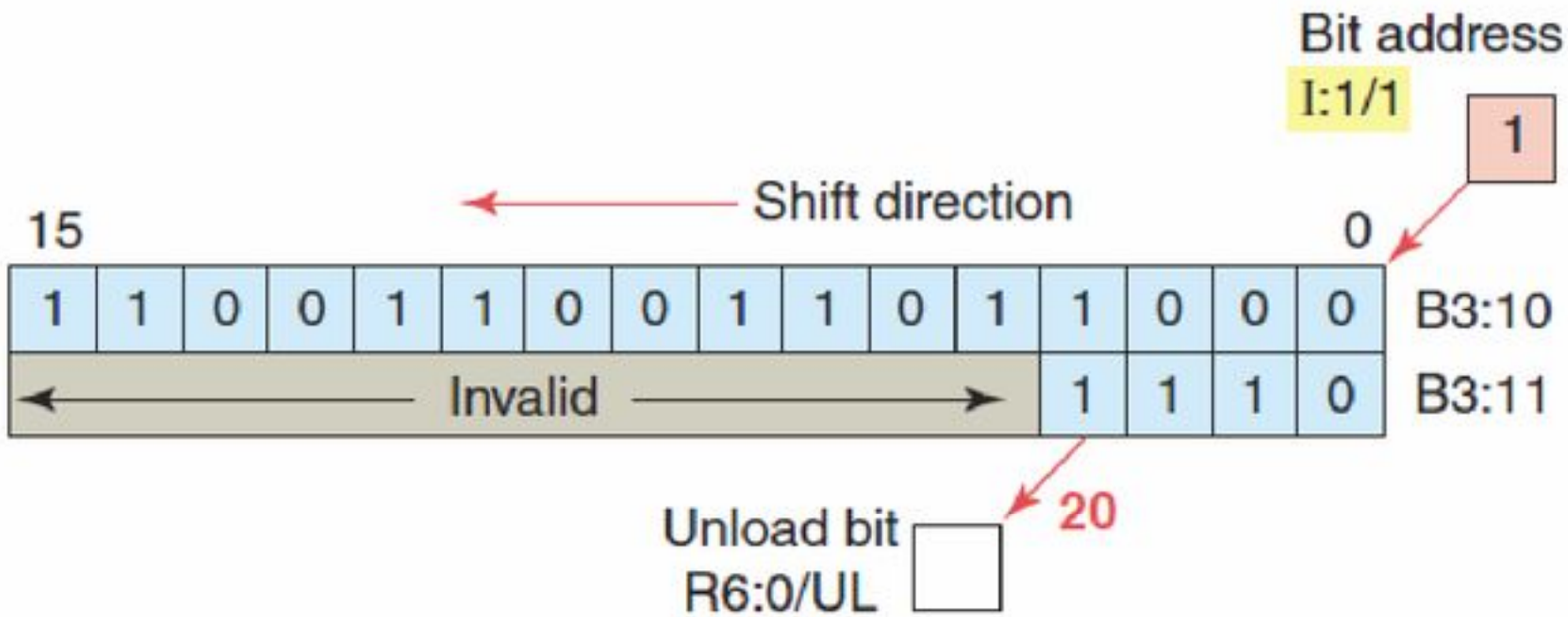
For example, there may be a specific type of package that you want to track and you only move a 1 into the register when that type comes on to your system. On the discharge side you may have an arm that pushes your package off the belt when the shift register shows it is in the proper location - but ignores other packages on the belt.

Bit shift left instruction.

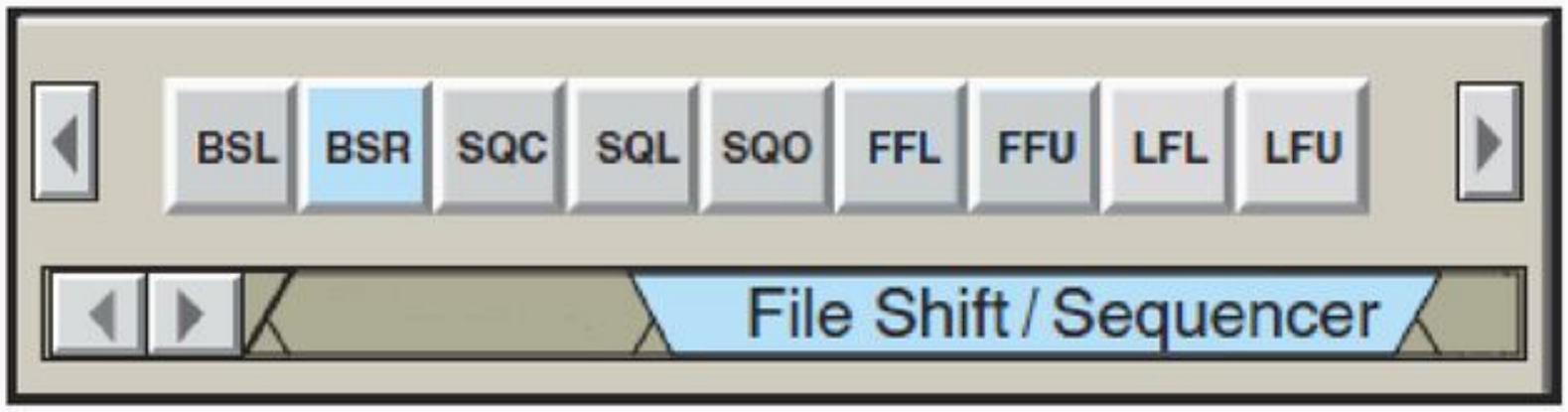


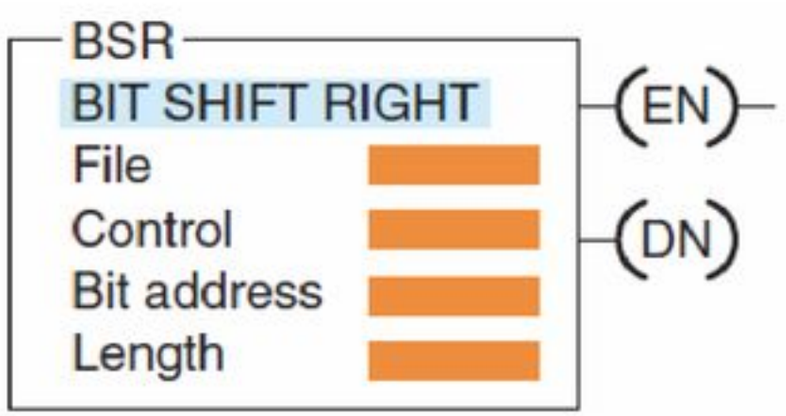


BSL (Bit Shift Left) - Loads a bit of data into a bit array, shifts the pattern of data through the array to the left, and unloads the last bit of data in the array.

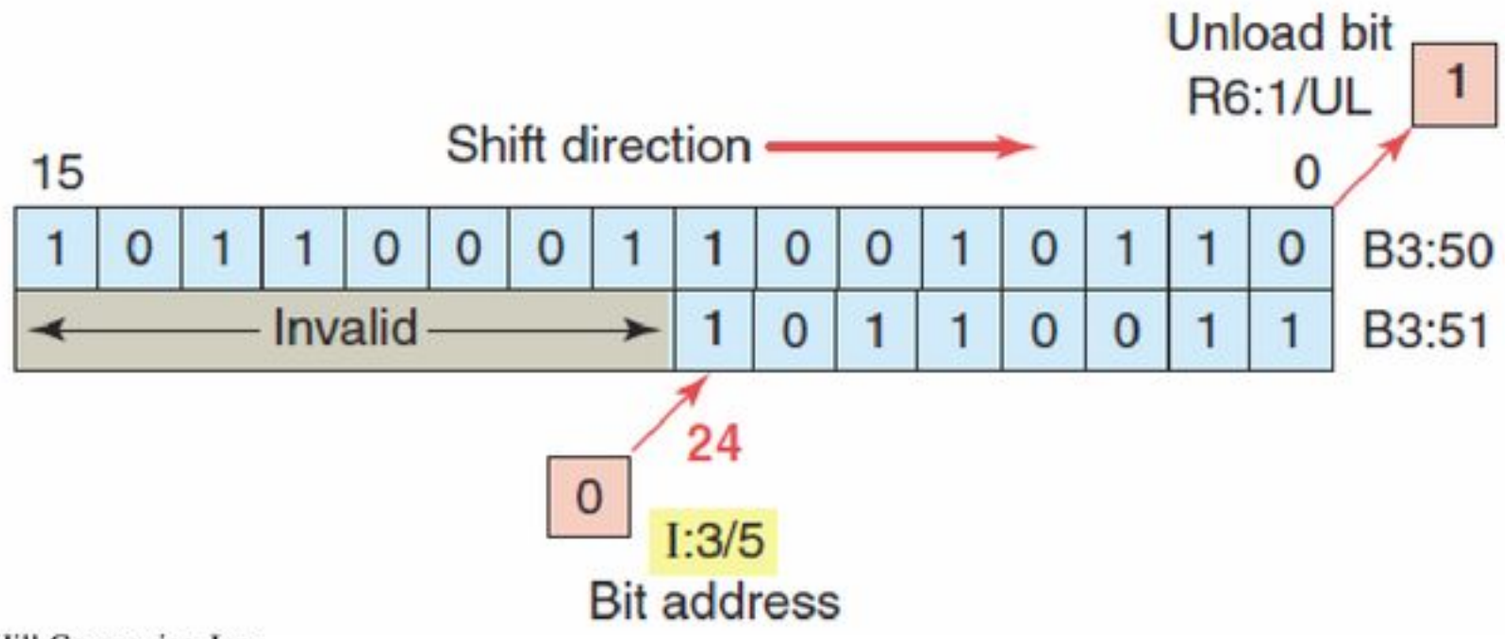


Bit shift right instruction.

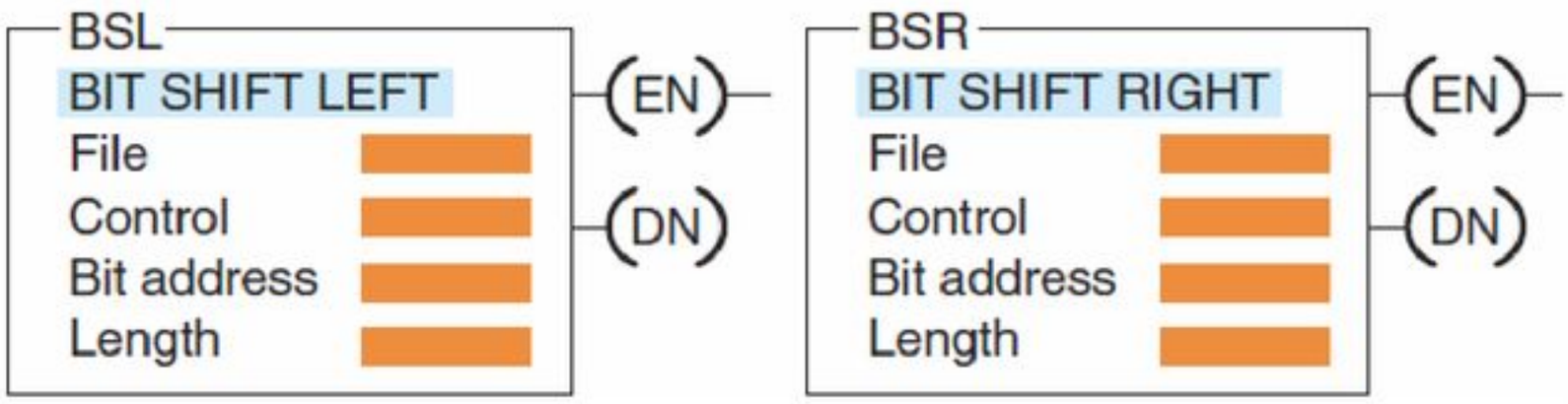




BSR (Bit Shift Right) - Loads a bit of data into a bit array, shifts the pattern of data through the array to the right, and unloads the last bit of data in the array.



The BSL and BSR instructions have the same operands.



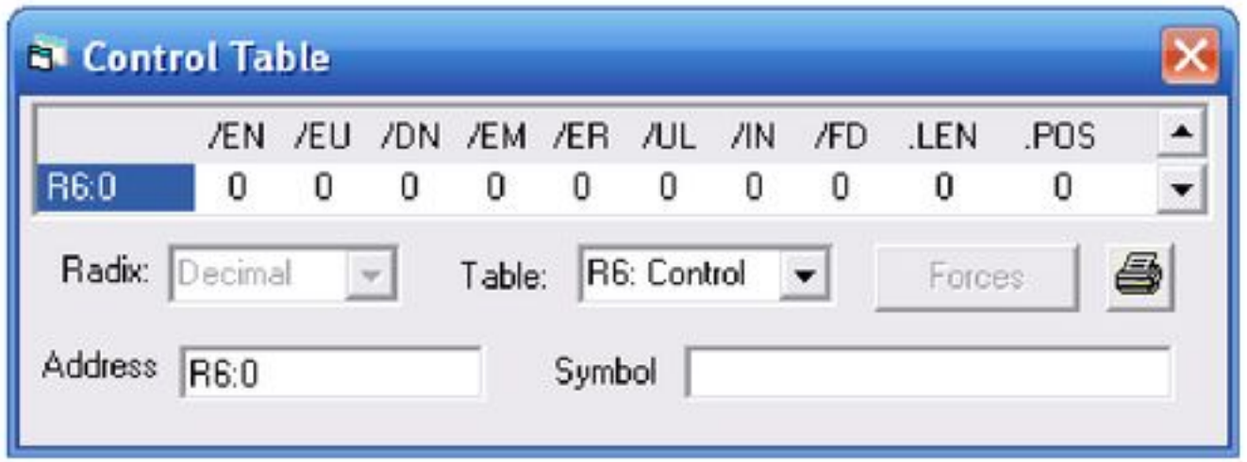
File - Address of the bit array you want to manipulate.

Control - R data table type used to control the instruction.

Bit address - Address of the source bit.

Length - Indicates the number of bits to be shifted, or the file length, in bits.

The status bits of the control word are:



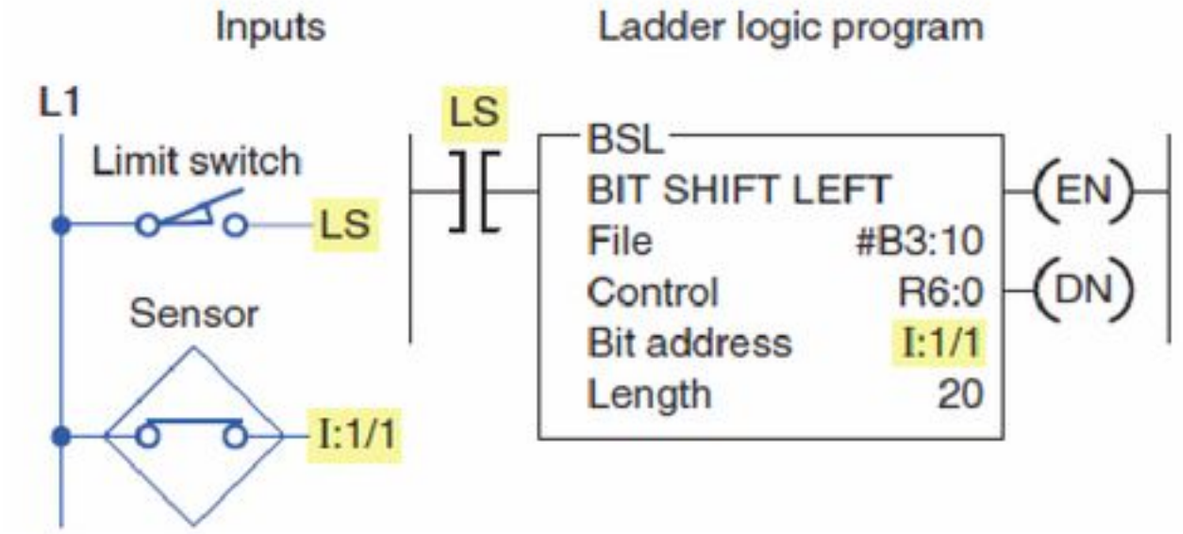
Enable Bit (EN) - is set to 1 when the instruction is true.

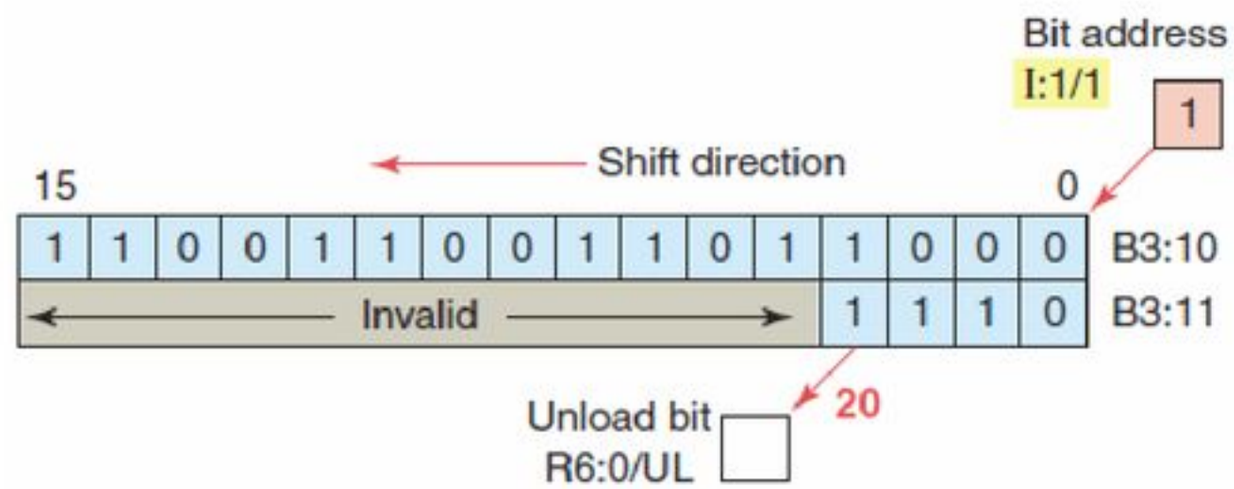
Done Bit (DN) -The done bit is set to 1 when the instruction has shifted all bits in the file one position.

Error Bit (ER) -The error bit is set to 1 when the instruction has detected an error, which can happen when a negative number is entered in the length.

Unload Bit (UL) - It is the bit location into which the status from the last bit in the file shifts

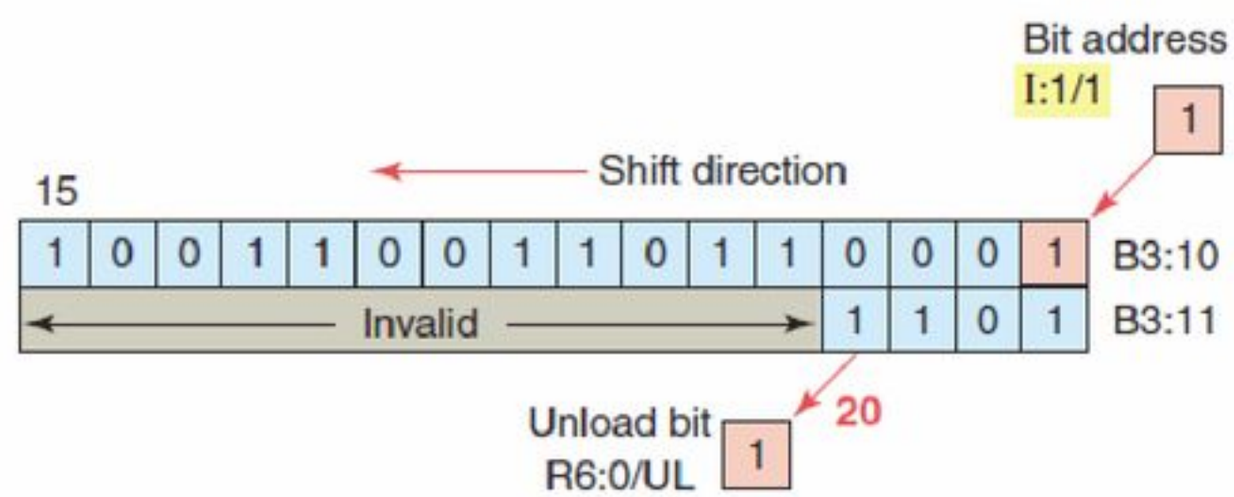
Bit shift left (BSL) instruction program.





B3:Table - Before limit switch clock pulse

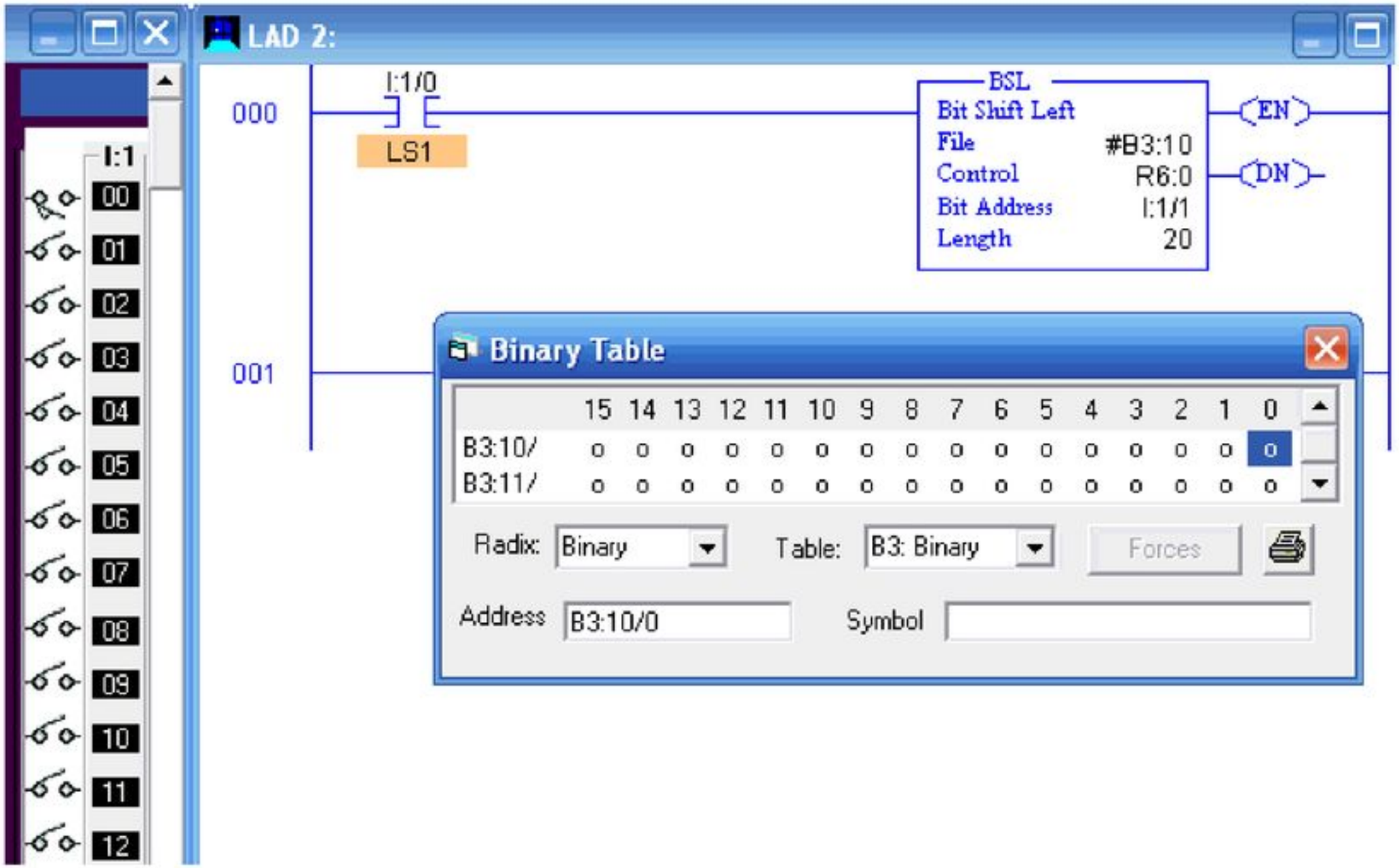
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
B3:10/	1	1	0	0	1	1	0	0	1	1	0	1	1	0	0	0
B3:11/	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0



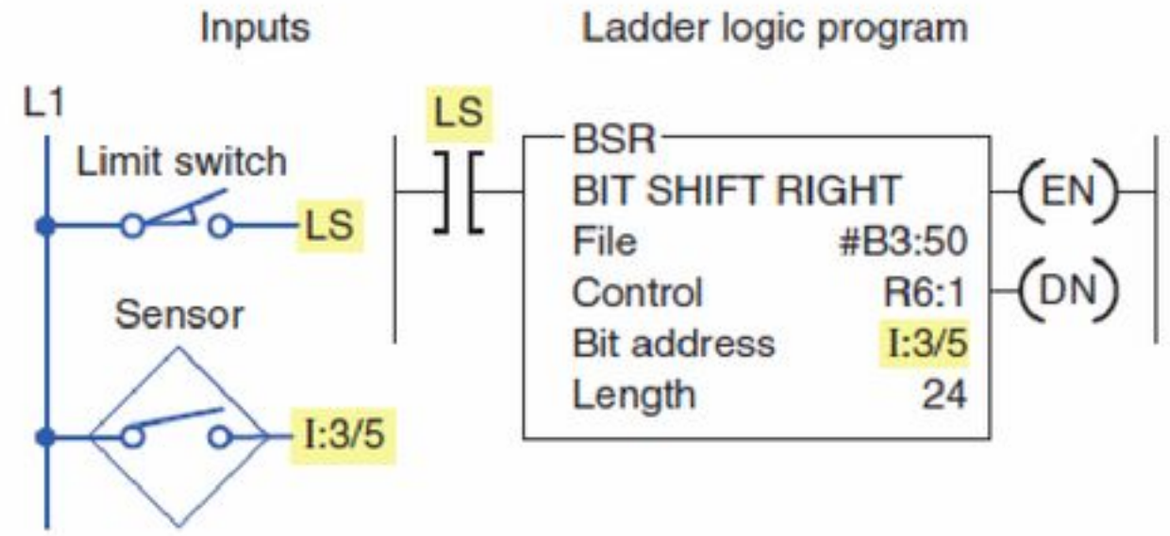
B3:Table - After limit switch clock pulse

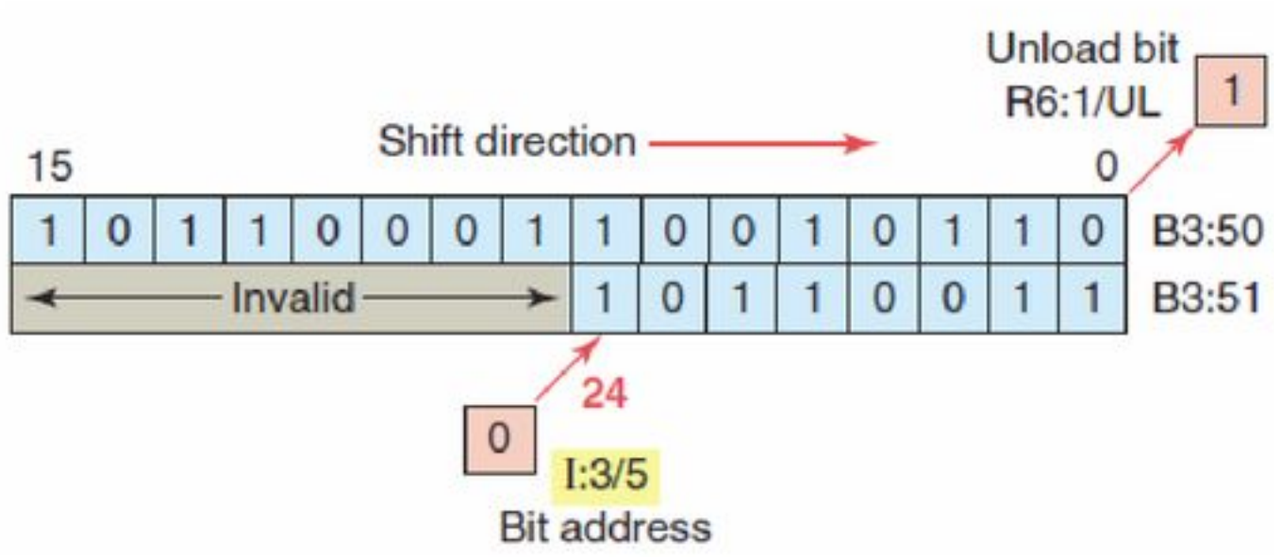
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
B3:10/	1	0	0	1	1	0	0	1	1	0	1	1	0	0	0	(1)
B3:11/	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1

Simulated bit shift left (BSL) program.



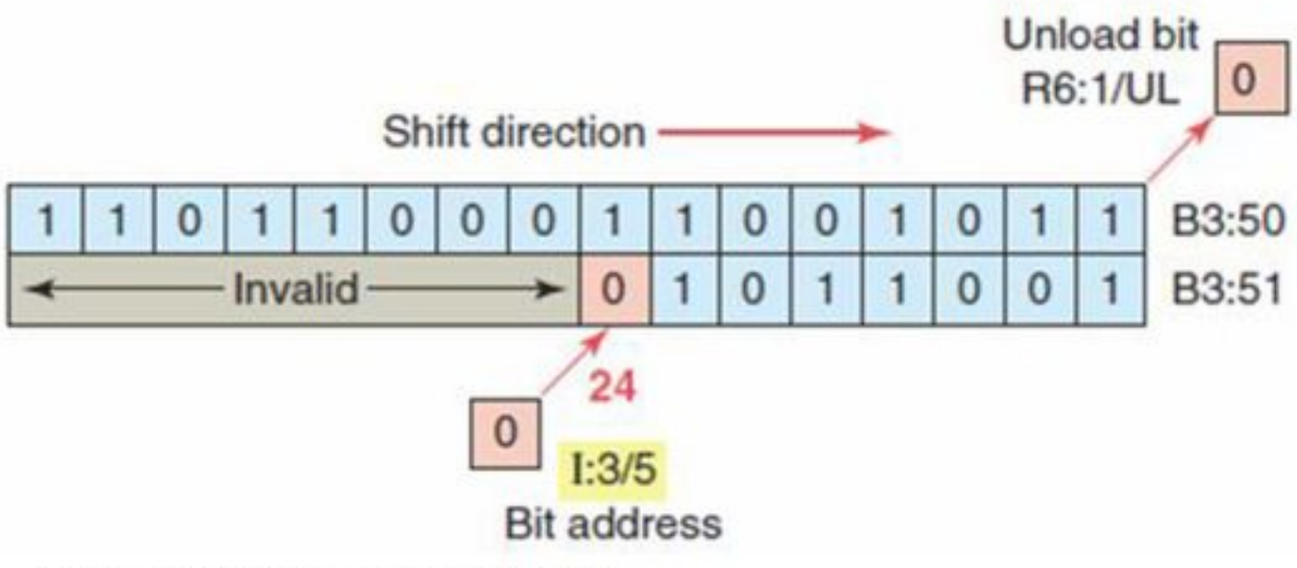
Bit shift right (BSR) instruction program.





B3: Table - Before limit switch clock pulse

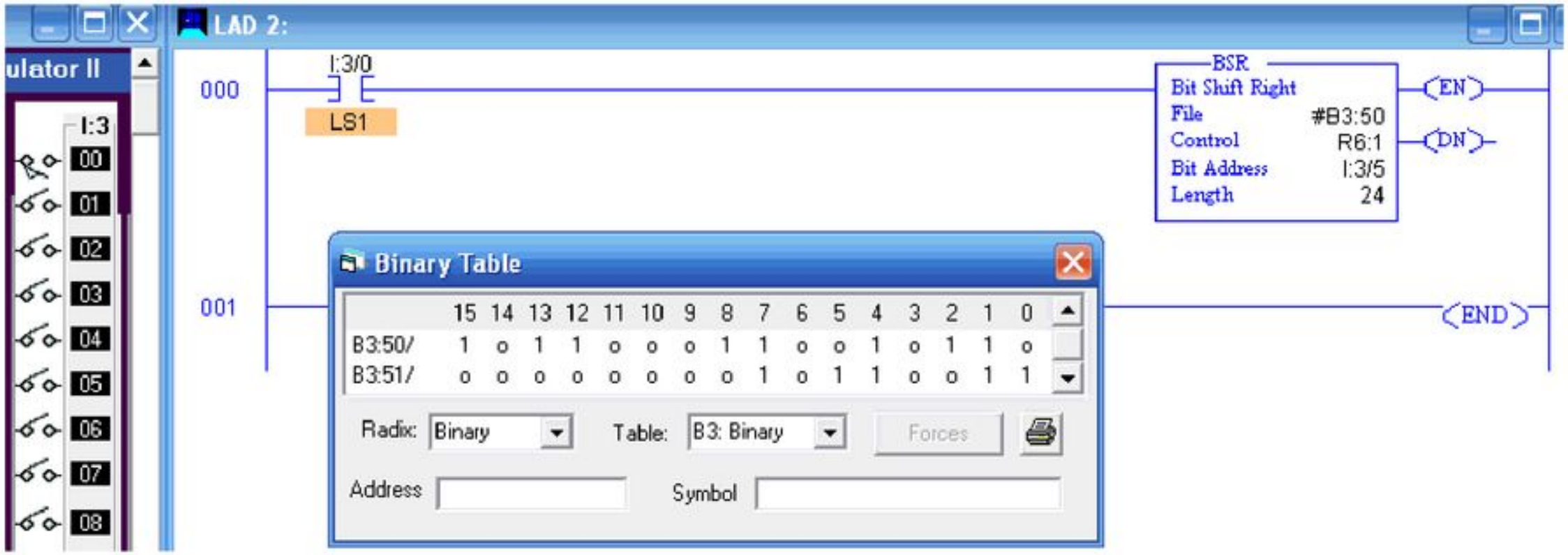
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
B3:50/	1	0	1	1	0	0	0	1	1	0	0	1	0	1	1	0
B3:51/	0	0	0	0	0	0	0	0	1	0	1	1	0	0	1	1



B3: Table - After limit switch clock pulse

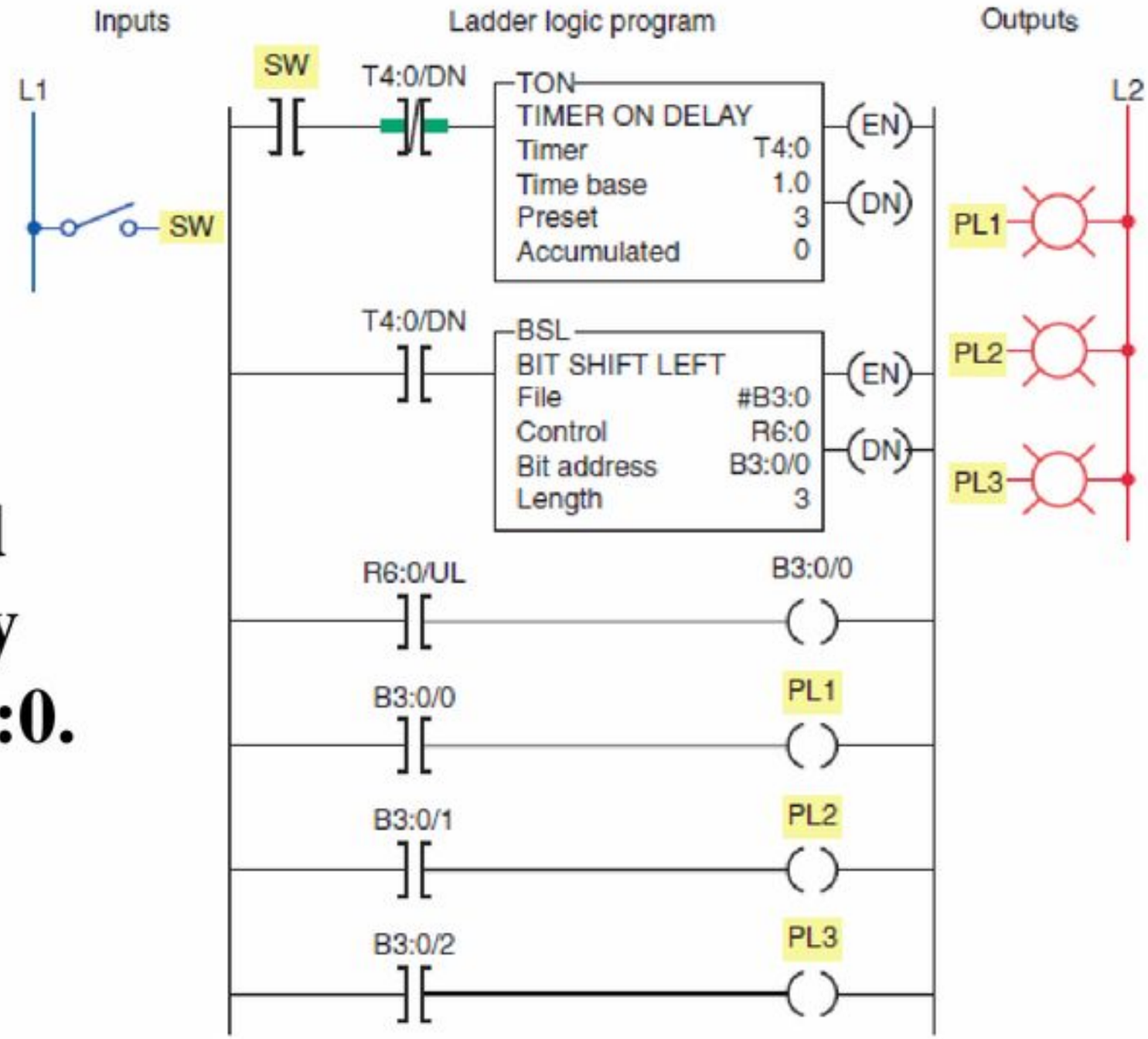
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
B3:50/	1	1	0	1	1	0	0	0	1	1	0	0	1	0	1	1
B3:51/	0	0	0	0	0	0	0	0	(0)	1	0	1	1	0	0	1

Simulated bit shift right (BSR) program.

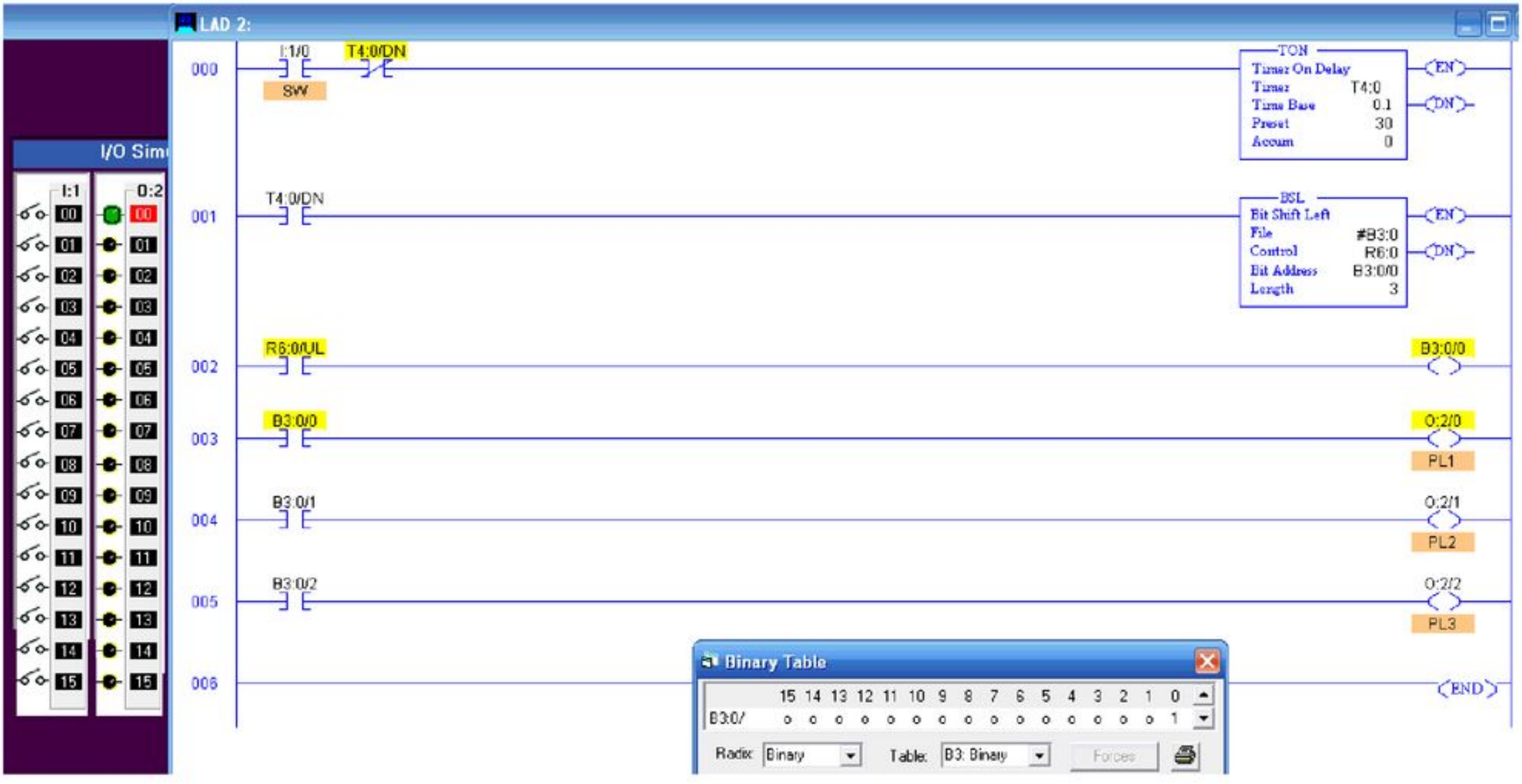


Bit BSL instruction program with wraparound operation.

The clock pulse input is a 3 second pulse generated by on-delay timer T4:0.



Simulated BSL instruction program with wraparound operation.

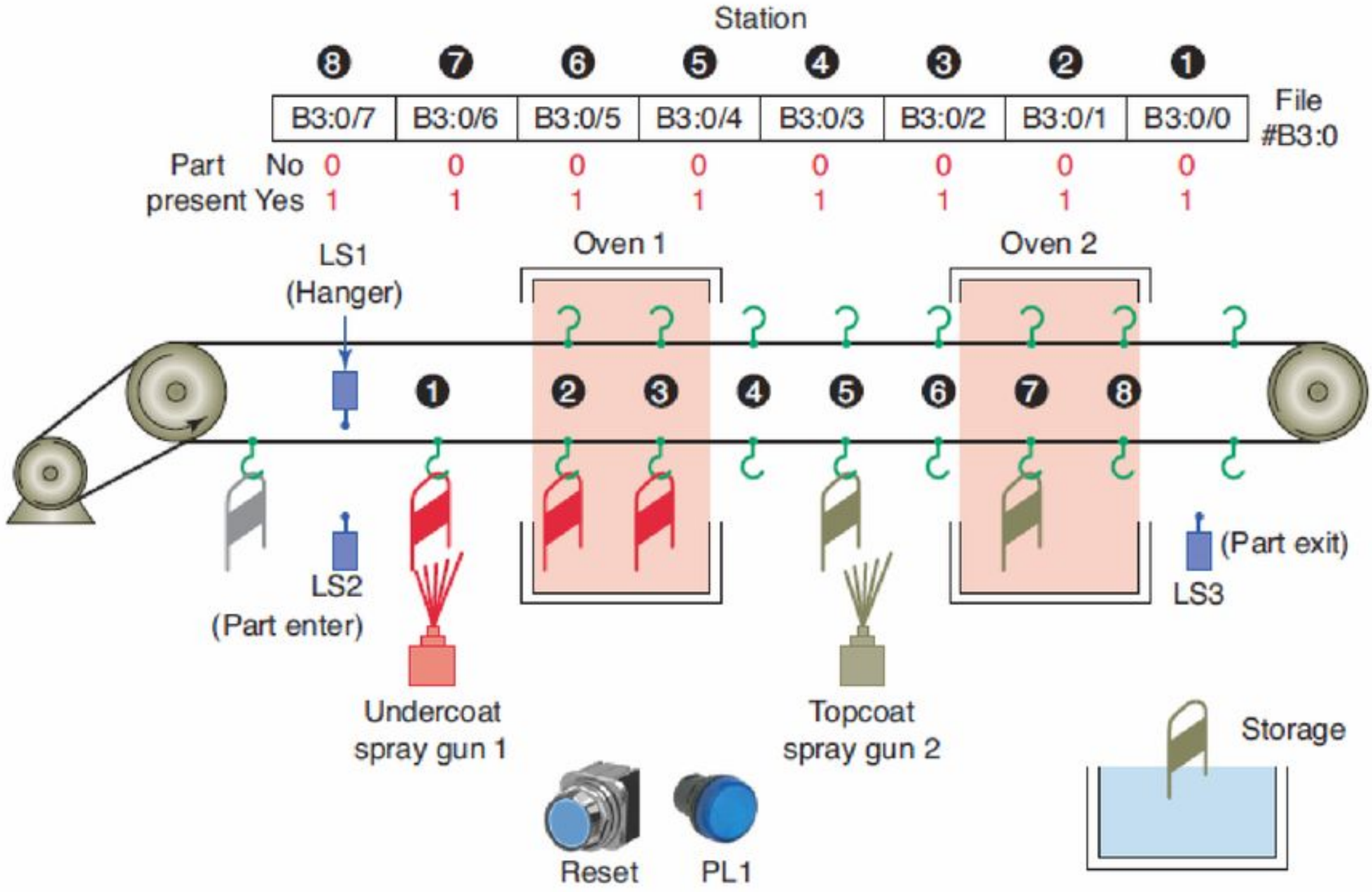


Shift registers are used in material handling processes where some form of binary information must be synchronized with a moving part on a conveyor.

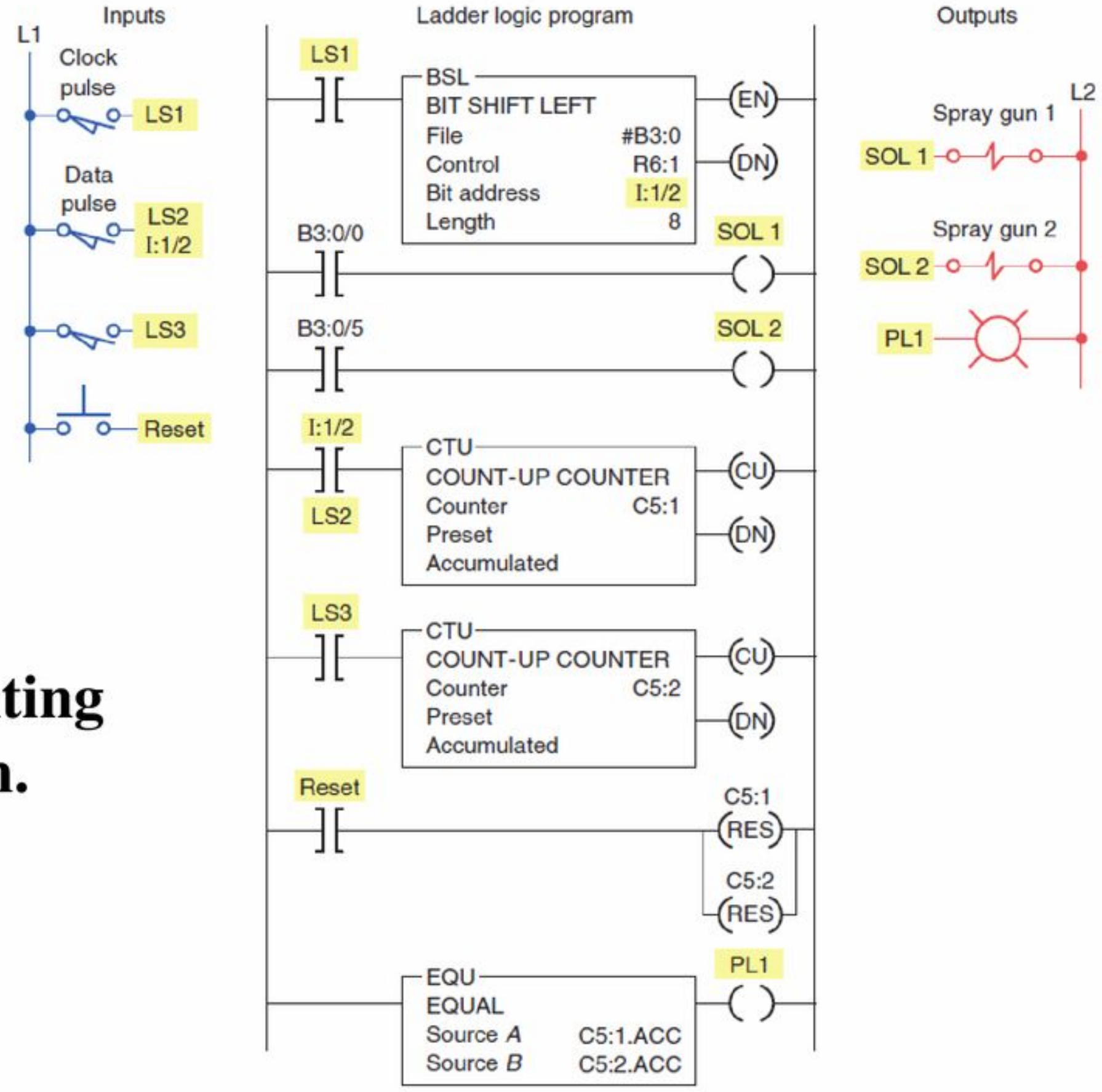
- **The sensor that drives the data line on the shift register detects the presence or absence of a carton.**

- **A logic 1 sensor condition state can indicate the presence of a carton, and a 0 the absence.**

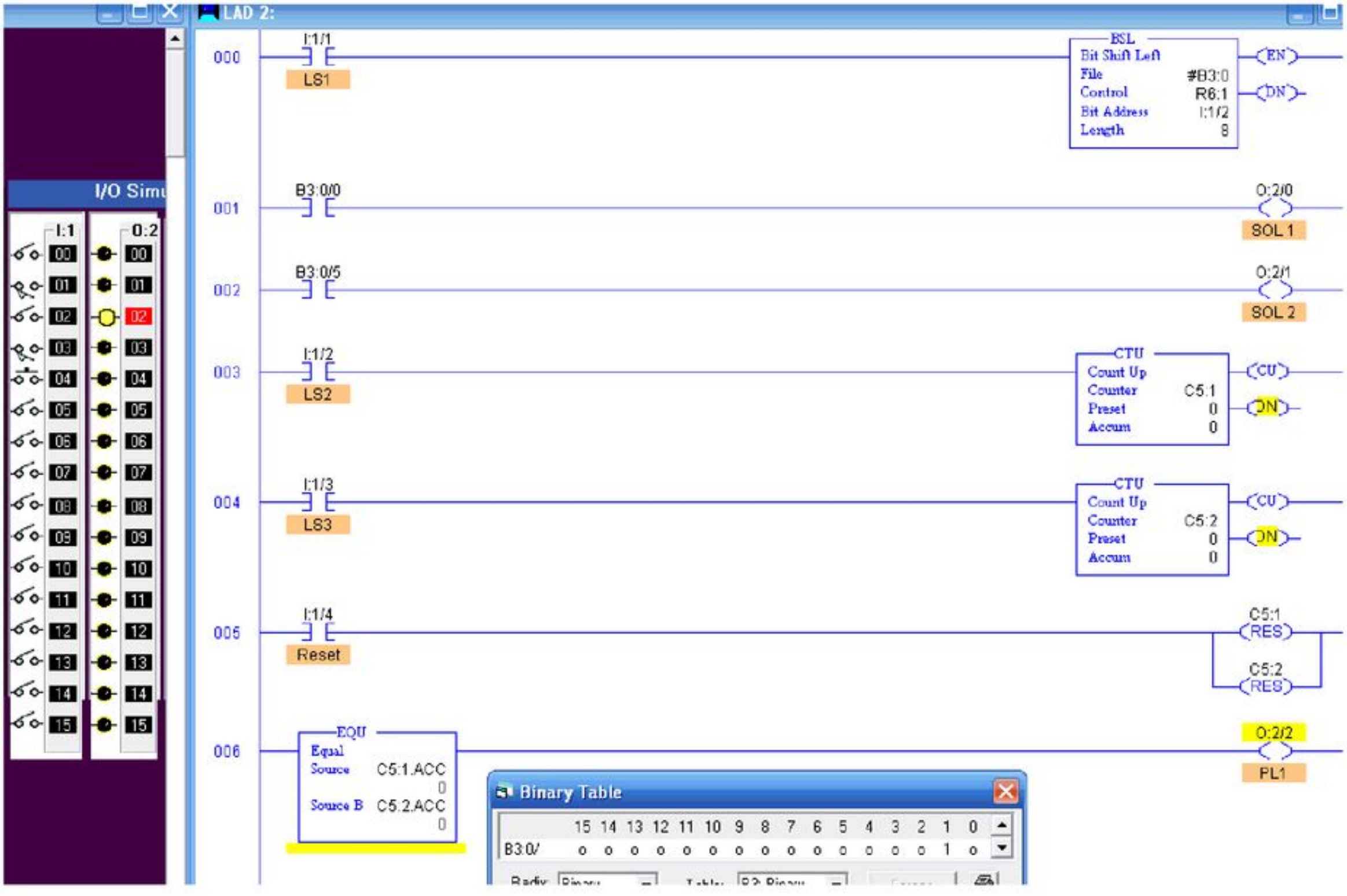
Spray-painting operation controlled by a shift left register.



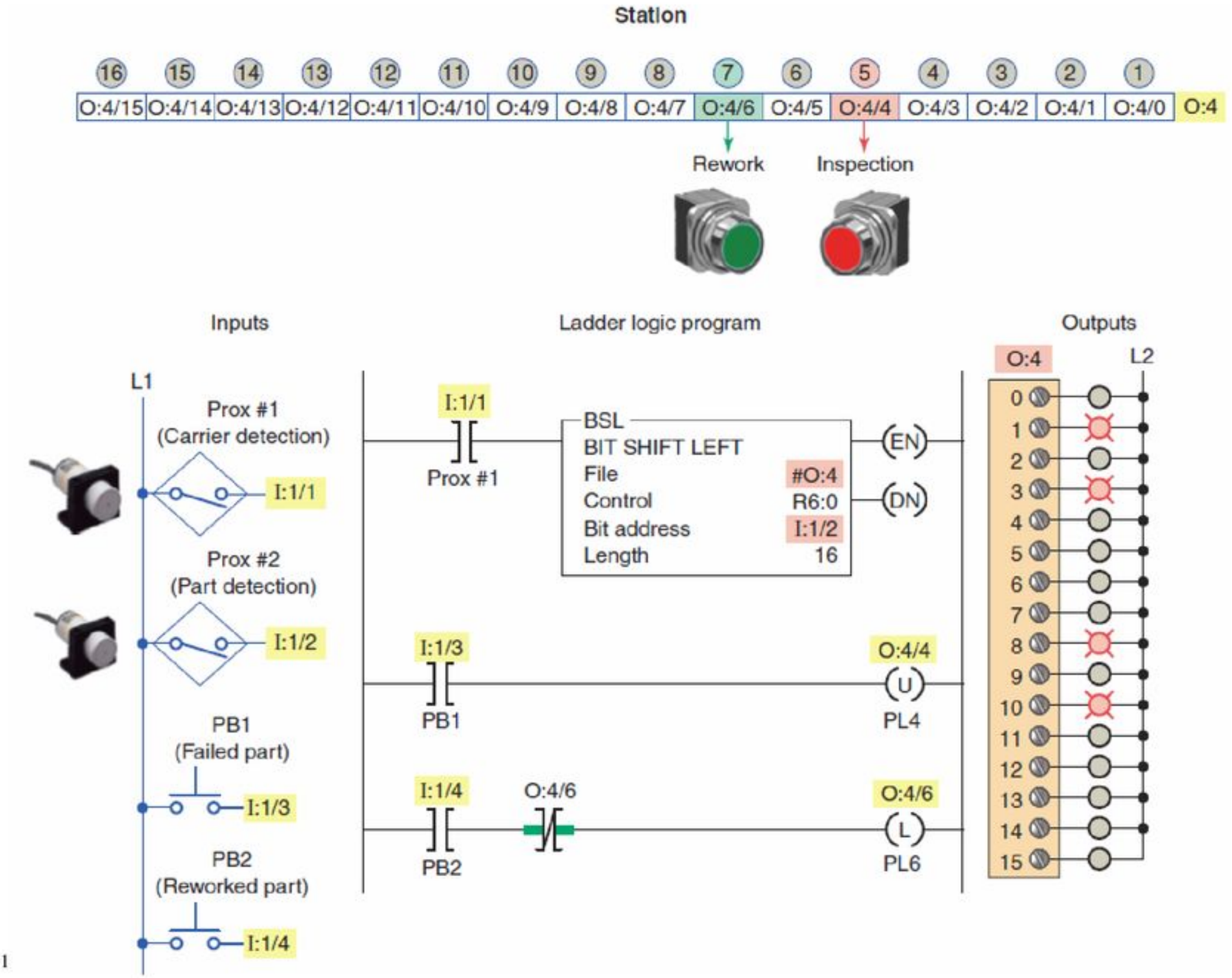
Spray-painting program.



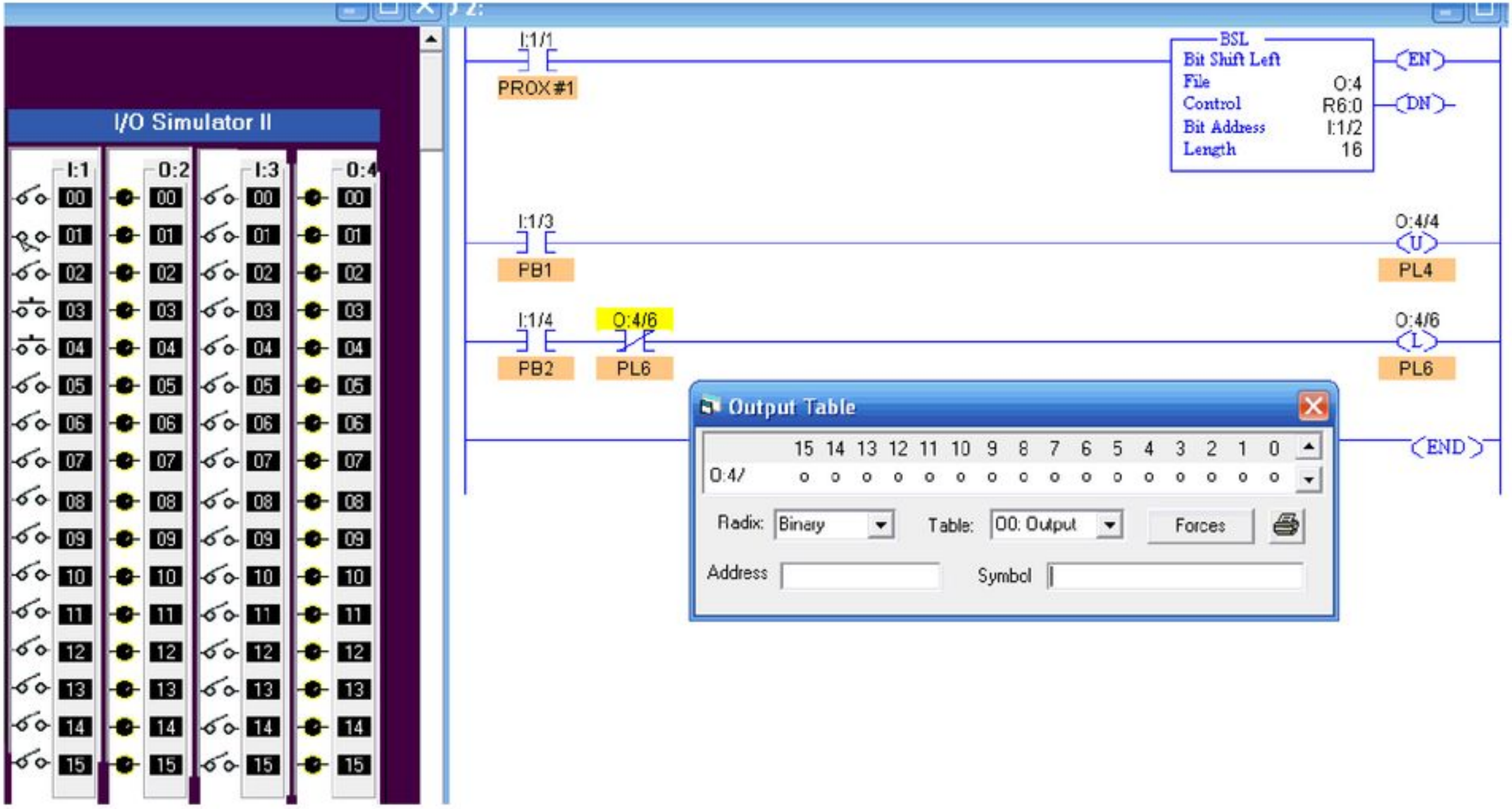
Simulated spray-painting program.



Tracking of carriers through a 16-station machine.



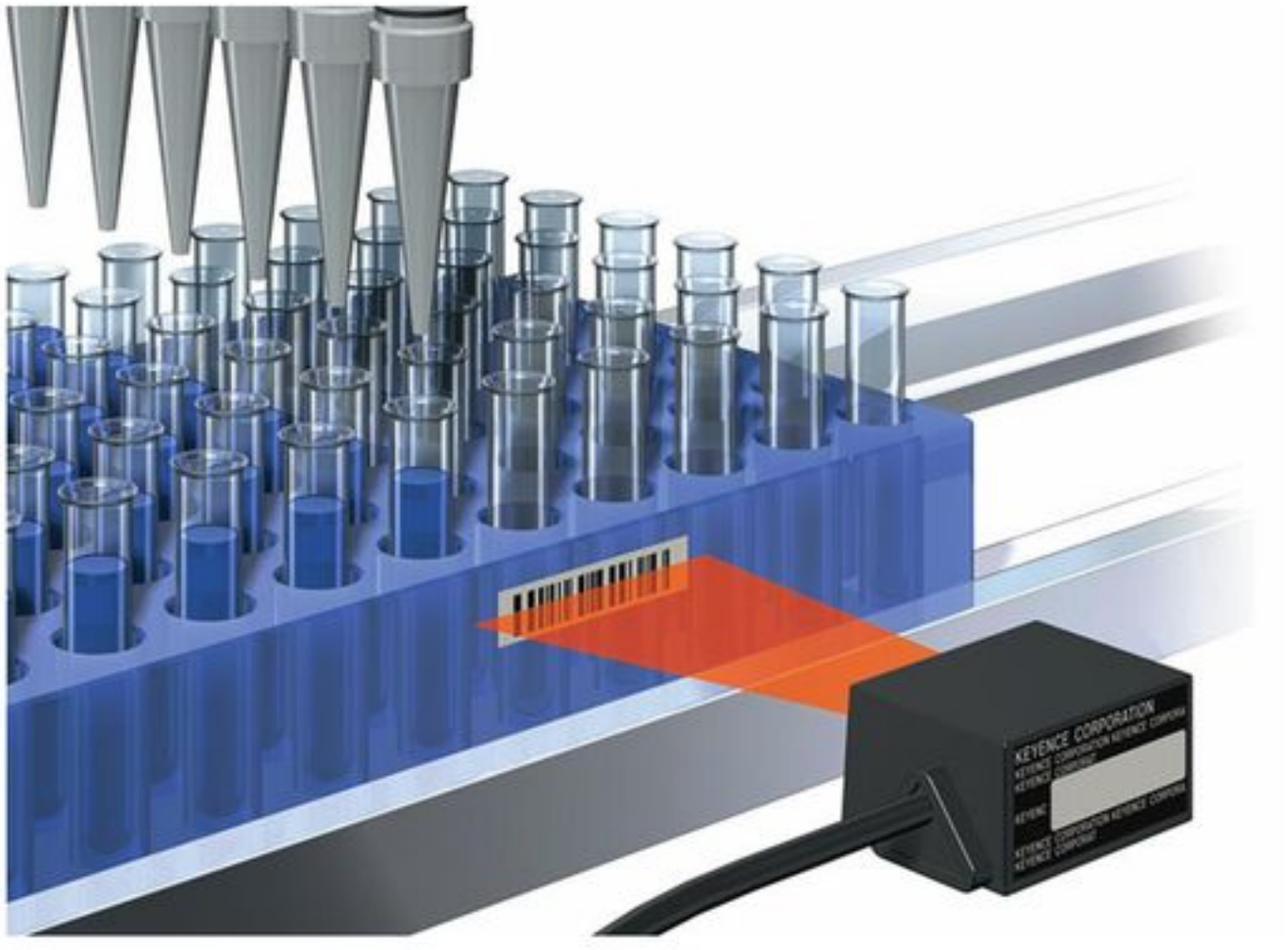
Simulation tracking of carriers through a 16-station machine.



12.5

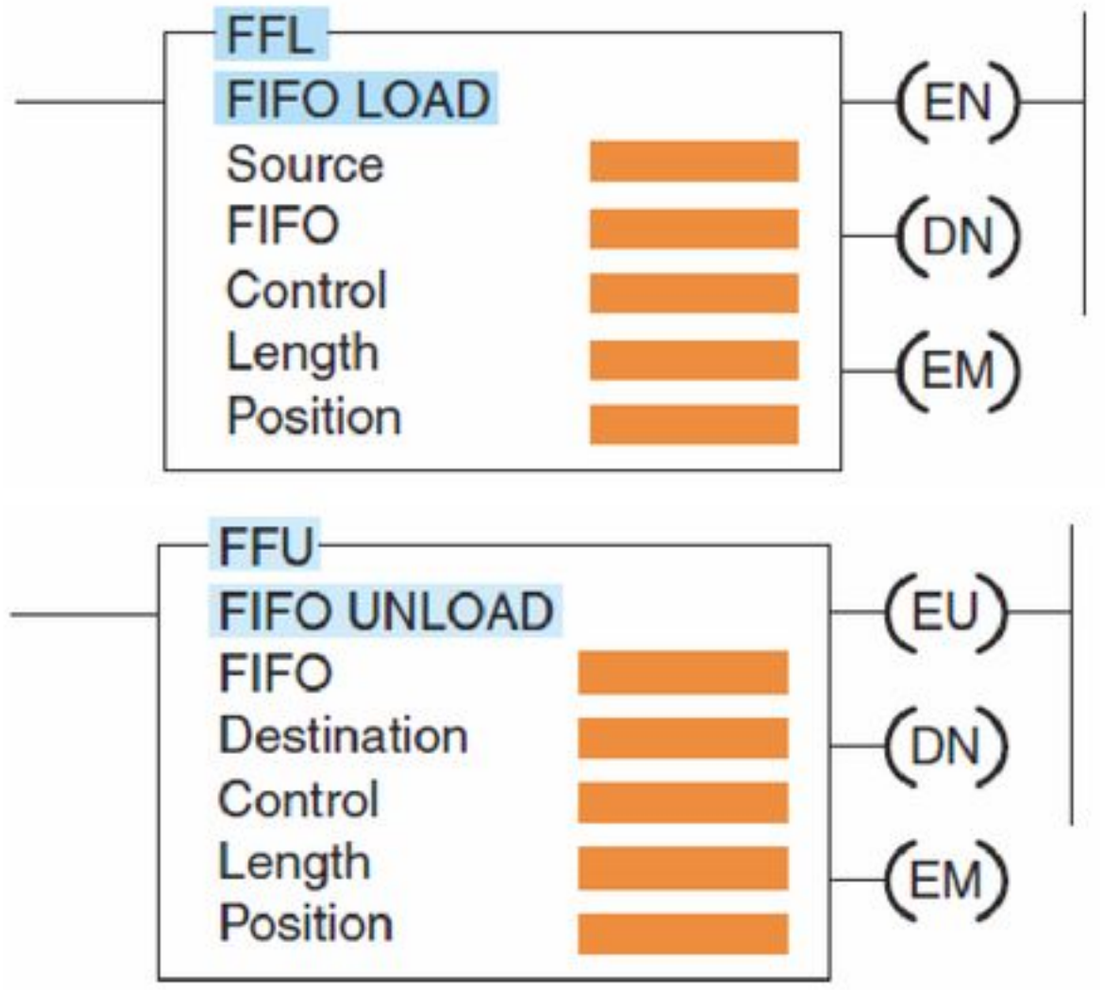
Word Shift Operations

Word shift operations provide a simpler method of loading and unloading data into a *file*, usually called the *stack*.

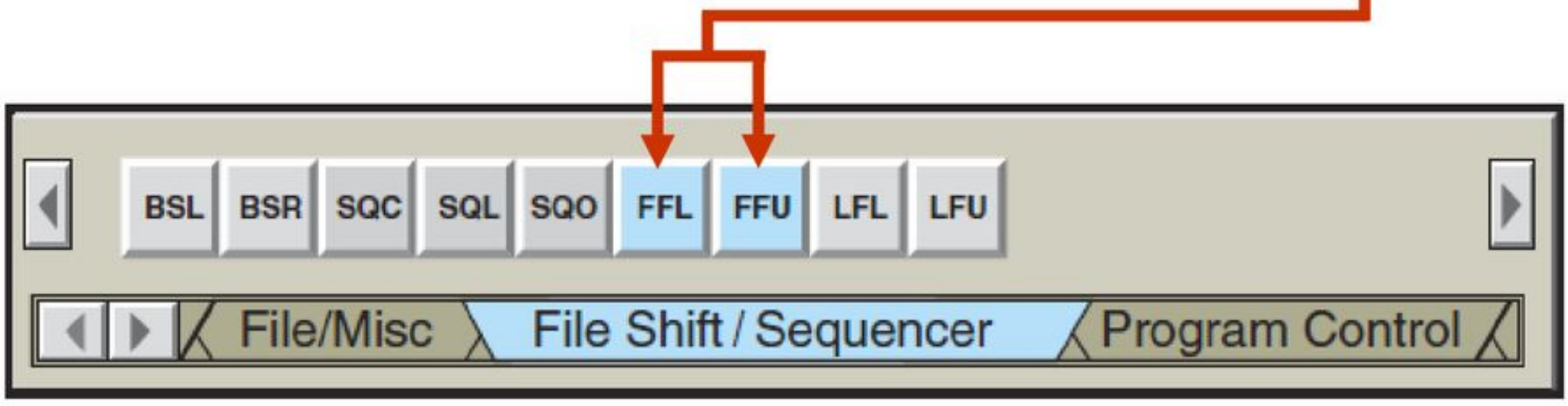


They can be used for tracking parts through an assembly line, where parts are represented by values that have a part number or an assembly code.

The *first in, first out (FIFO)* instructions are word shift operations.



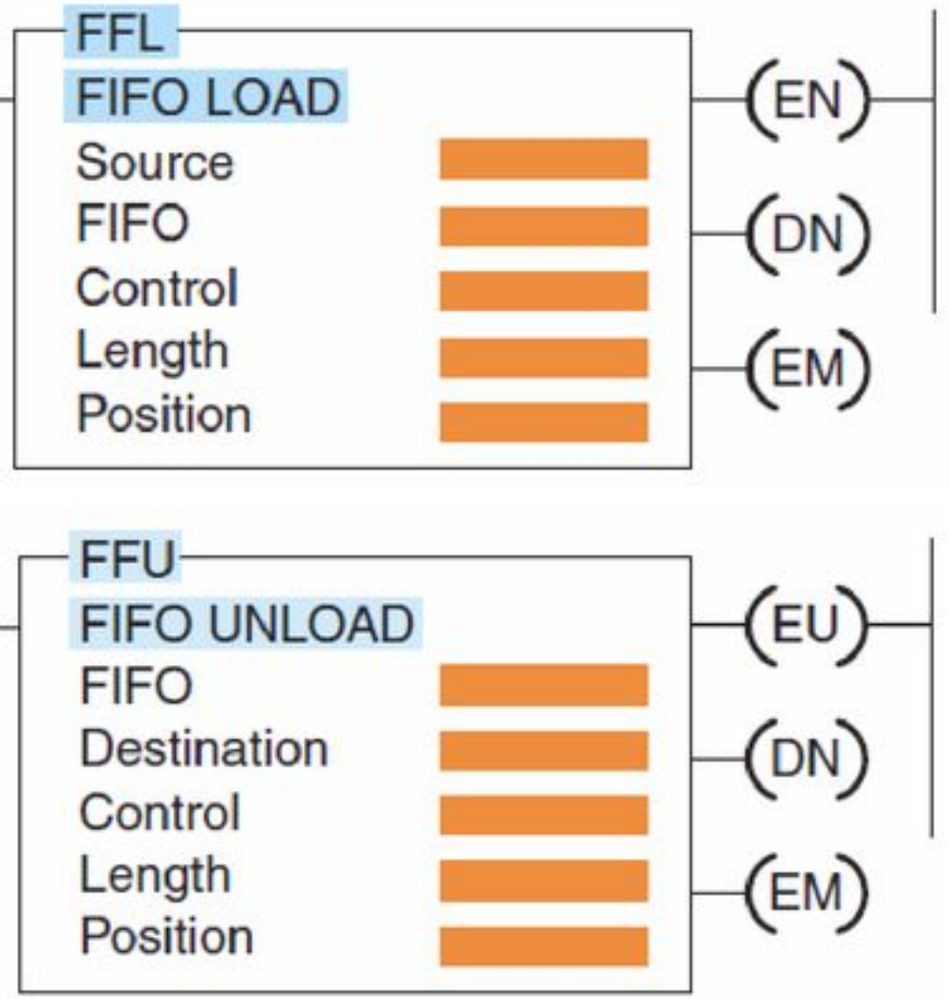
The FFL and FFU instruction are used in pairs.



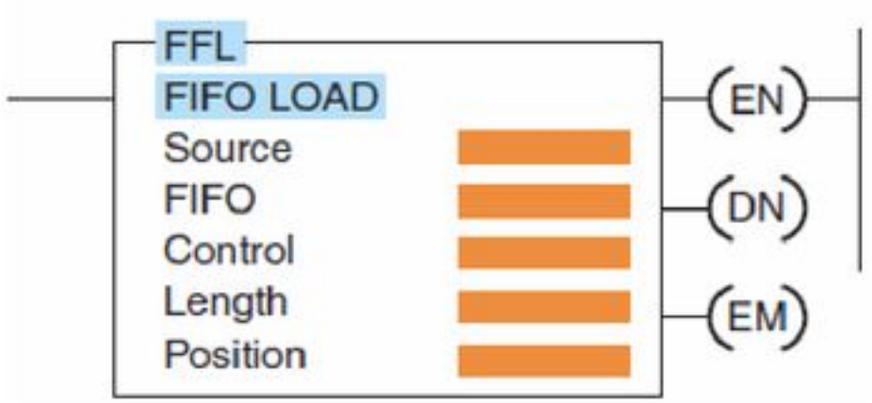
The FIFO function shifts the data from a *complete word* rather than shifting bits of information within a word.

The FFL *loads* logic words into a user-created file called a FIFO stack.

Two separate shift pulses are required: one to shift data into the file (load) and one to shift data out of the file (unload).



The parameters required for the FIFO load (FFL) instruction are:



Source - Word address from which the data are entered into the FIFO file.

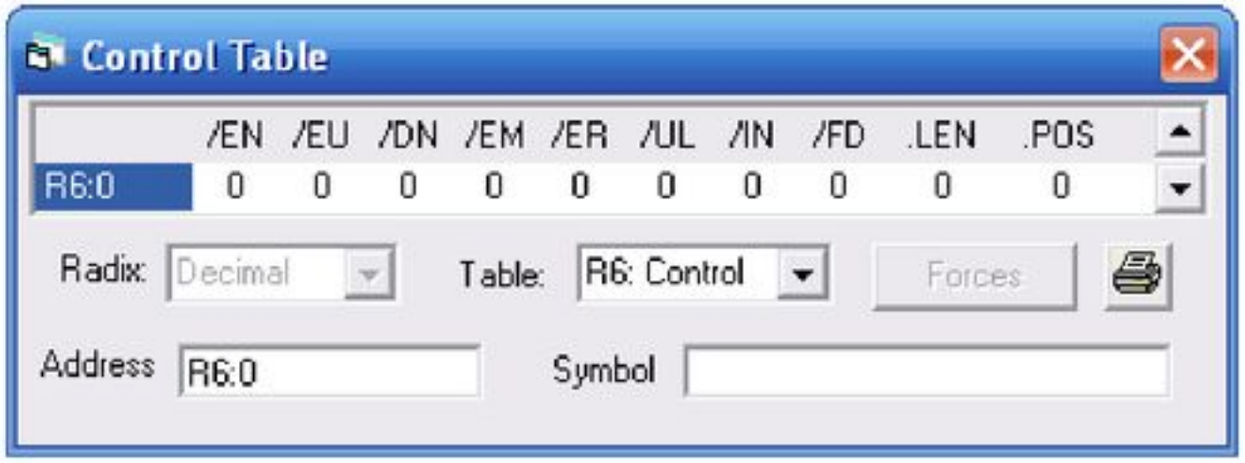
FIFO - Address of the file in which the data are entered.

Control - R data-table type and is the file address of the control structure.

Length - File length in words.

Position - Is the next available location where the instruction loads data into the stack.

The status bits of the control word are:

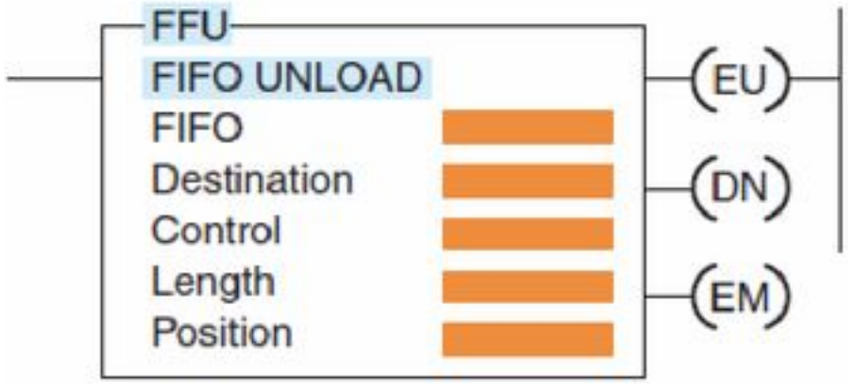


Enable Bit (EN) - is set to 1 when the instruction is true.

Done Bit (DN) -The done bit is set to 1 when the instruction's position equals the length.

Empty Bit (EM) –The empty bit is set to 1 when all the data have been unloaded from the FIFO file.

The parameters required for the FIFO unload (FFU) instruction are:



FIFO - Address of the file in which the data are entered.

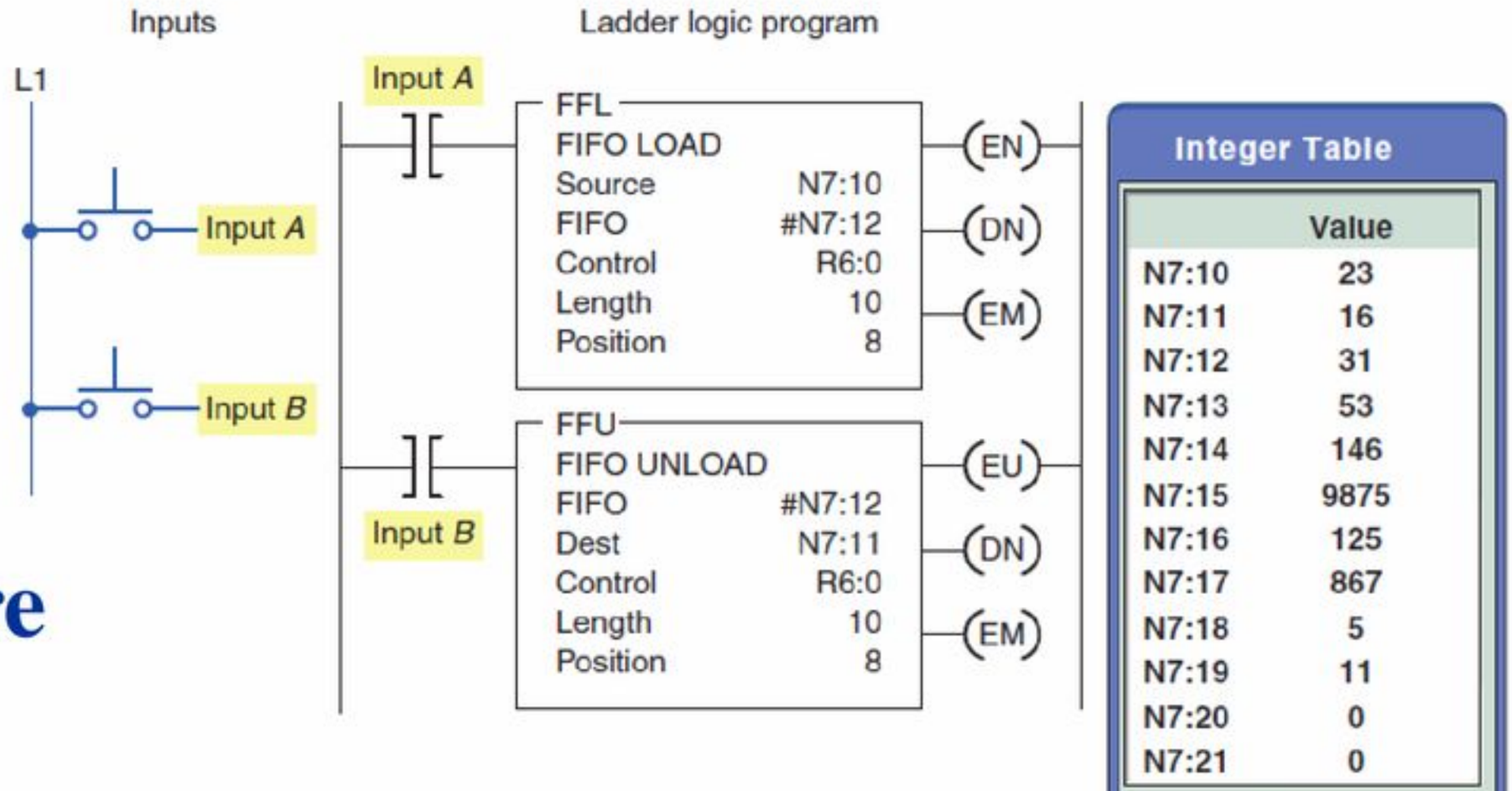
Destination - Address to which the FFU unloads data.

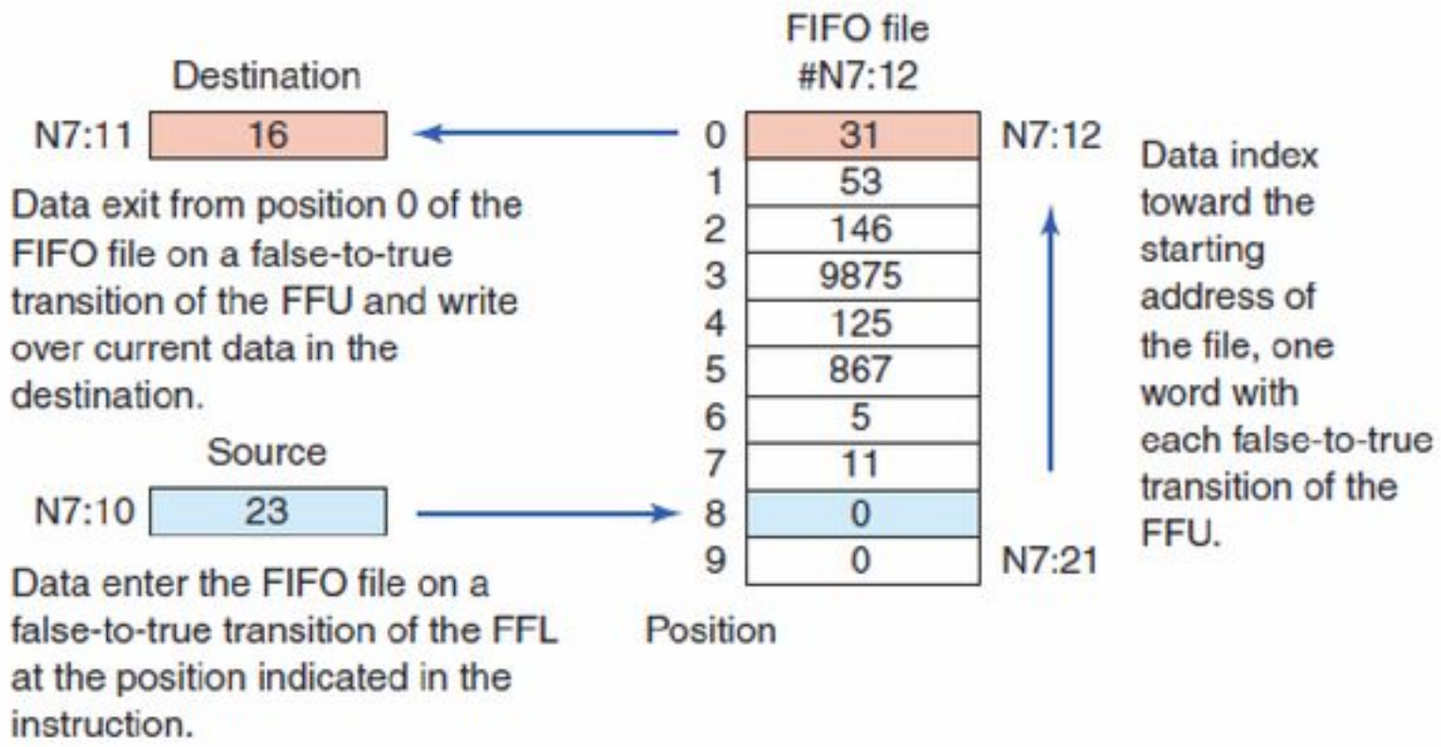
Control - A three-word element that consists of the status word, the length, and the position. When it is paired with the FFL, the control addresses are the same.

Length - File length in words.

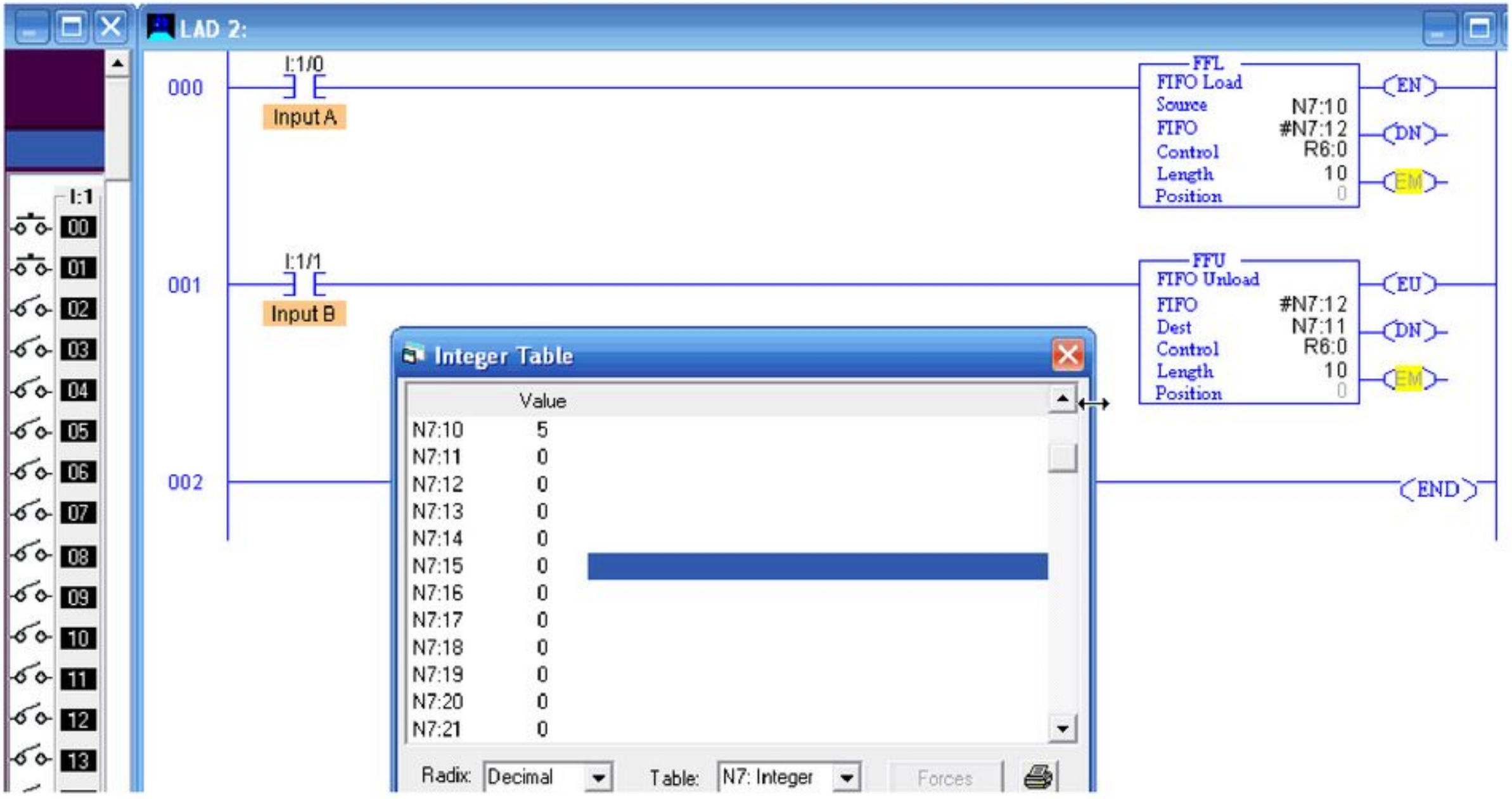
Position - Next location from which data are unloaded.

How data are indexed in and out of a FIFO file.





Simulation of data in and out of a FIFO file.

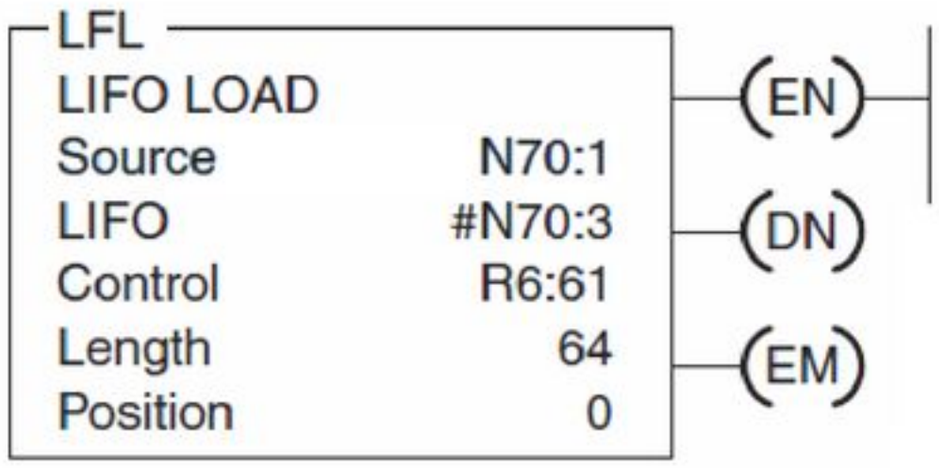


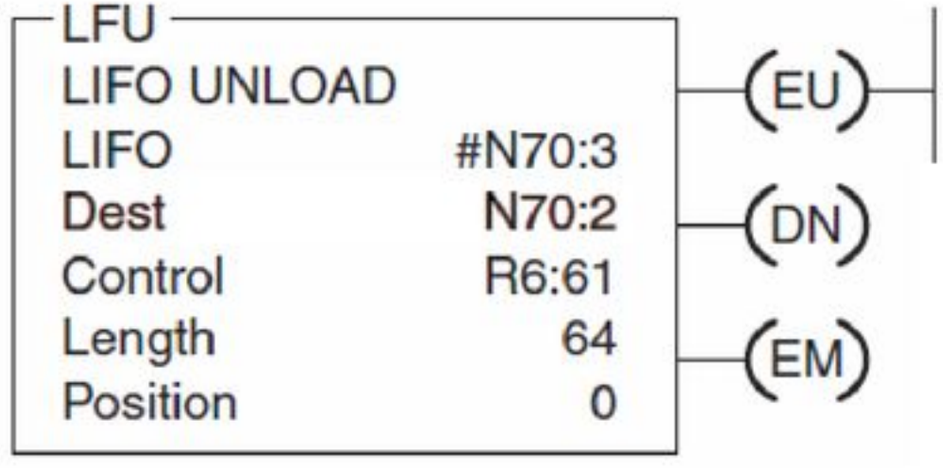
The *First in, First out (FIFO)* instruction is often used for inventory control.

Each part is assigned a unique code, which is loaded into a FIFO stack, and parts are removed in the order prescribed by the stack.

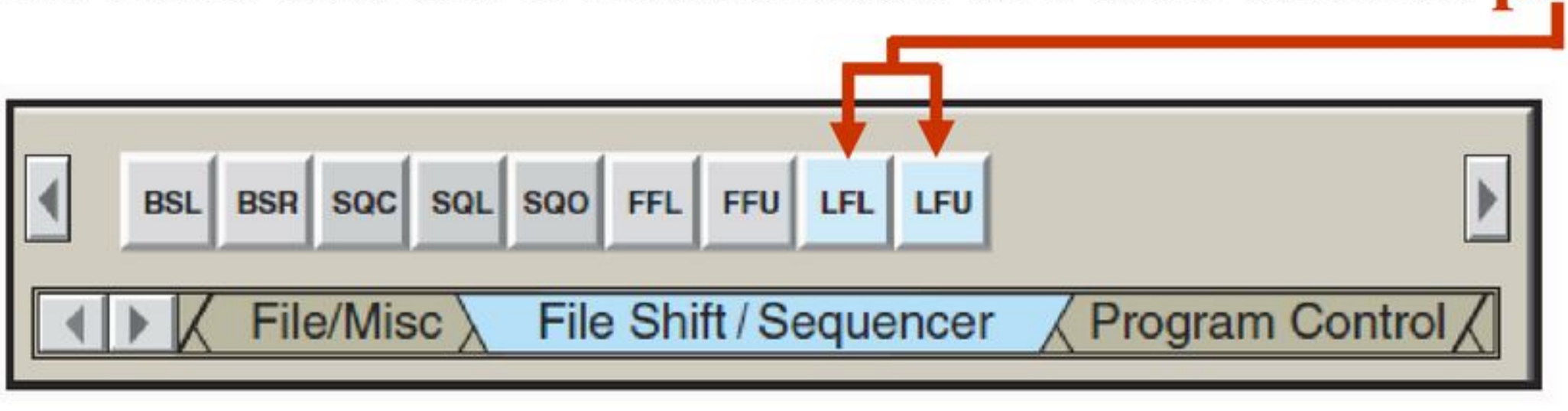
This type of control ensures that the oldest part in the inventory is used first as the first part entered is the first part removed.

The *Last in, First out (LIFO)* instruction inverts the order of the data it receives by outputting the last data received first and the first data received last.

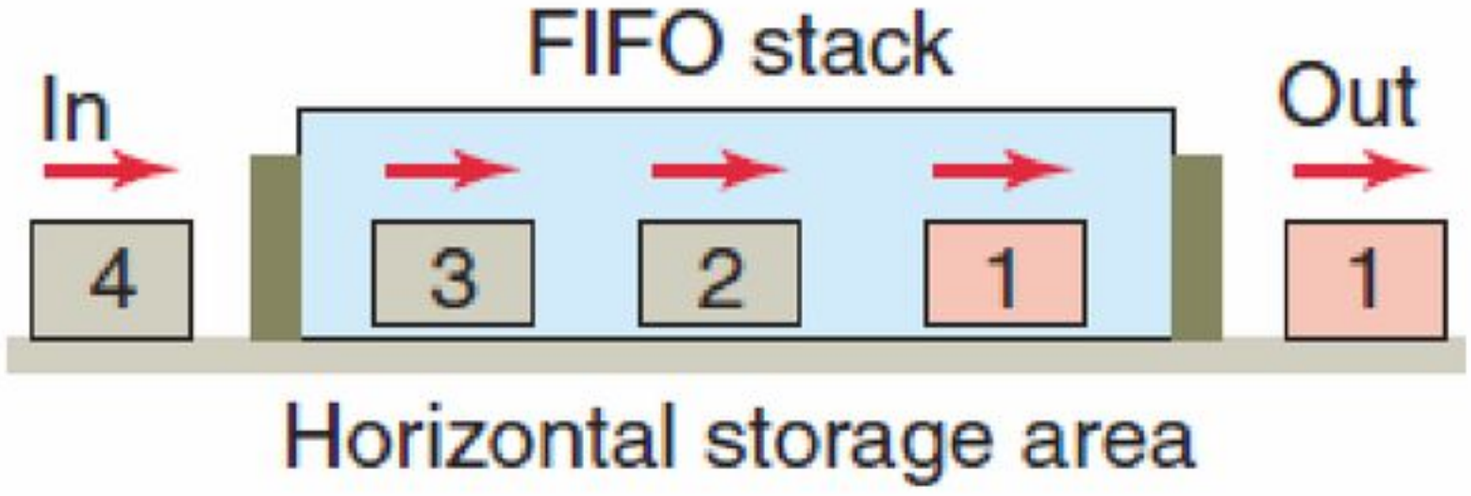


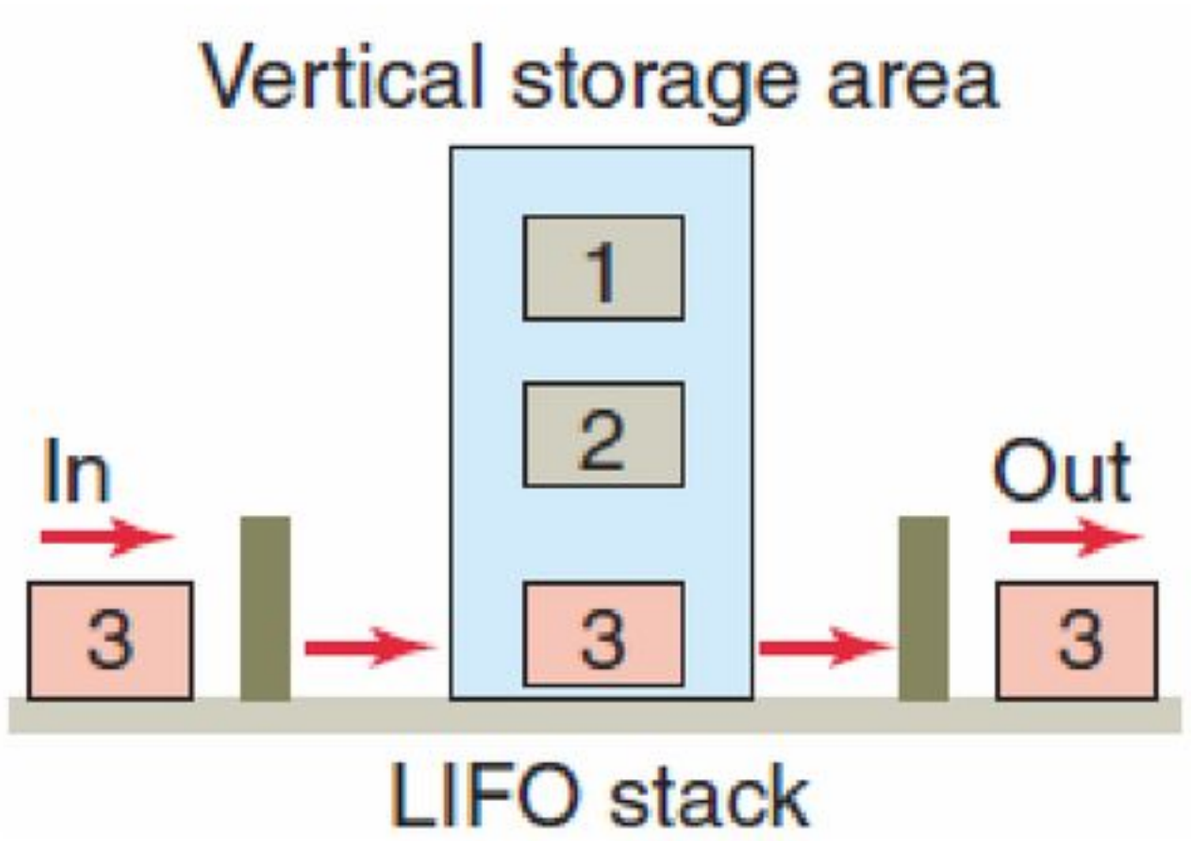


The LFL and LFU instruction are also used in pairs

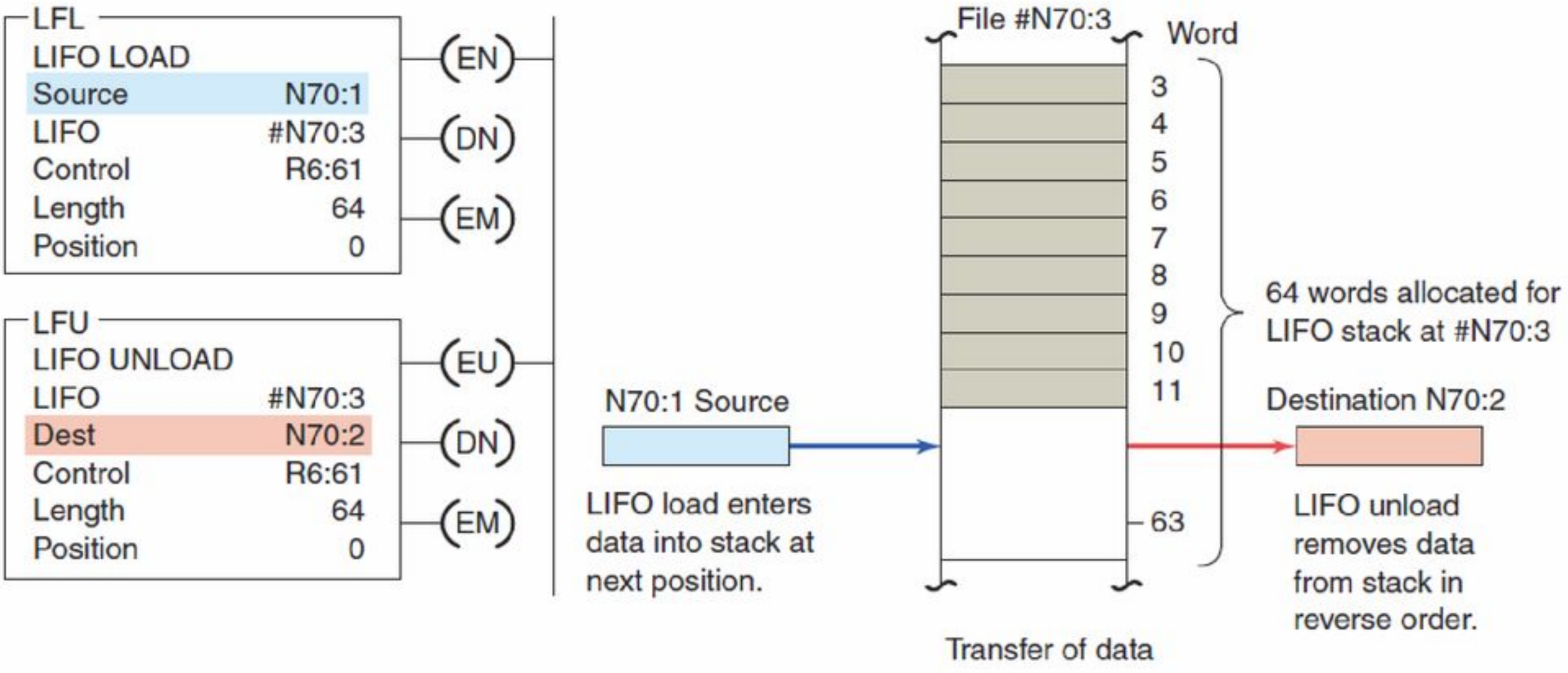


FIFO and LIFO container stacking operations.





LIFO instruction pair program.



The LIFO stack operates similarly to that of the FIFO stack, except that the last word in the LIFO stack is the first word that is unloaded from the stack.