

Use of Sequencer Functions in Industrial Control

Max Rabiee, Ph.D., P.E.
University of Cincinnati

Abstract

In this paper we will study one of the most commonly used functions in programmable logic controller (PLC) systems. This function is called the Sequencer. All programmable logic controllers (PLC) have this function. The sequencer function provides powerful capability for the PLC. This function and its applications are studied as part of a logic controller course in Electrical and Computer Engineering Technology (ECET) programs.

Some PLC manufacturers refer to the sequencer function as the “drum controller”. However, in certain cases PLC manufacturers use the Table-to-Register or File-to-Word functions instead of the sequencer function [1]. These functions are not as versatile as the sequencer function.

In this paper we will illustrate how students can utilize the sequencer function in the laboratory for controlling electromechanical actuators and robots. We will also illustrate how to control the time period between each sequencer step.

Introduction

Sequencer functions can be used to control multiple outputs with several step patterns. This means that in every sequenced step, you can change the state of the output devices connected to the output ports housed in output modules. Output can change from the ON to OFF state, or from the OFF to ON state, or remain at the same state.

In this paper, we will study the use of sequencer functions in programmable logic controller systems. We will discuss how Sequencer functions are used to control output with multiple step patterns. Specifically, we will study the use of sequencer functions for the Allen-Bradley SLC500 series programmable logic controllers [2]. These sequencer functions are: (1) SQO (Sequencer Output) function, (2) SQI (Sequencer Input) or SQL (Sequencer Load) function, and (3) SQC (Sequencer Compare) function. An Allen-Bradley SLC 500 Series Programmable Logic

Controller (PLC) will have at least two of the three aforementioned sequencer functions.

The objective of this paper is to illustrate how to: (1) teach students to configure the Sequencer Output function (SQO), (2) illustrate to the students in lab the use of sequencer output function (SQO) in process and industrial control, (3) use the timer function to insert constant time intervals between the sequencer steps, (4) teach students to use two sequencer functions and one timer function to create a sequencer routine with variable time intervals between its steps, (5) configure the Sequencer Load function (SQL), (6) configure the Sequencer Compare function (SQC), and (7) use the sequencer compare function (SQC) to find an input port bit pattern.

Configuring Sequencer Output (SQO) Function

Table 1 illustrates the operations of a process control system. Five output devices are to be turned on and off in a specific pattern. We are assuming that there are three steps in this control procedure. The advancing of the steps can be either event-driven or time-driven. We will configure a sequencer output function for this event-driven control regime.

Table 1. The Output Pattern of a Control System

<u>Step Number</u>	<u>Motor #1</u>	<u>Motor #2</u>	<u>Green Light</u>	<u>Red Light</u>	<u>White Light</u>
1	ON	ON	OFF	OFF	OFF
2	OFF	OFF	ON	ON	OFF
3	OFF	OFF	OFF	ON	ON

Table 1 shows that in this example two motors and three pilot lights are the output devices. Every time a normally open (N.O.) input is closed, the pattern on one of the steps is placed on the PLC output port. In step one, both motors are on. In step two, the green and red pilot lights are on. In step three, the red and white pilot lights are on. Figure 1 displays the ladder logic diagram for accomplishing the tasks described in Table 1. An Allen-Bradley SLC500 Series PLC sequencer function is utilized in this ladder logic diagram [3].

Every time the normally open (N.O.) input device is closed, the sequencer position advances to the next step. The sequencer data can be placed either in bit data file #B3, bit data file #B10 or

integer data file #N7. The sequencer data file contains the On/Off bit patterns or an integer number. Data file #B3 holds the sequencer output file in Figure 1. Figure 2 shows the output pattern for this example stored in the #B3 file. Notice that the motors are connected to both outputs zero and one (O:0/0 and O:0/1). Therefore, bits zero and one in bit data file three (#B3) are used to control the motors. Similarly, the pilot lights are controlled by bits two, three and four in bit data file three (#B3). Sequencer bit patterns in file #B3 are placed on the destination register which for this example is output module zero (O:0).

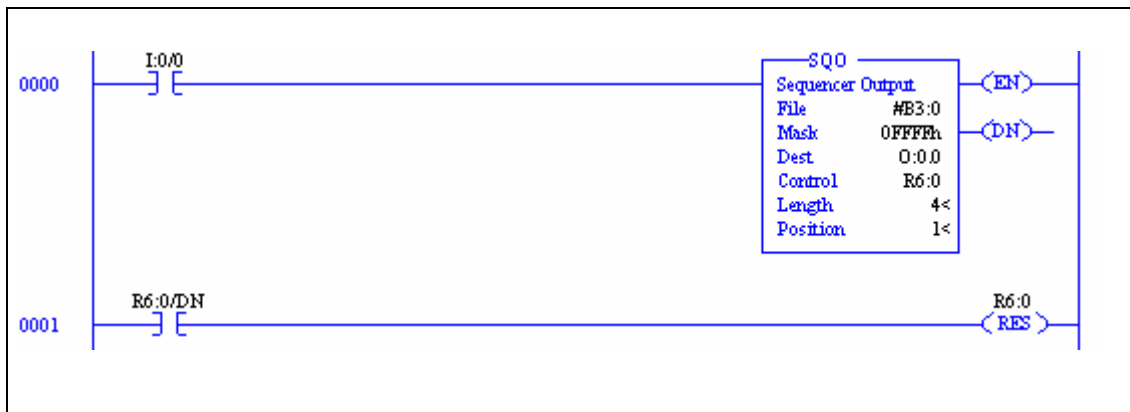


Figure 1. PLC Ladder Logic Diagram for the Output Pattern in Table 1

Offset	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	0	0	0	0	0	0	0	0	0	0	0	0	1	1
B3:2	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0
B3:3	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0
B3:4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2. Data File for the Patterns Described in Table 1

The five digit hexadecimal number in the Mask field can be used to hide the step patterns residing in the data file #B3. In this example we are not masking (i.e., hiding) any bit pattern. These 16-bit words in bit data file three (#B3) are ANDed by a hex number entered in the mask area. The number of steps are set to four in the SQO function displayed in Figure 1.

The control register R6:0 is utilized to control the SQO function. The PLC uses the control register bits for monitoring and controlling the SQO function. Two control register bits R6:0/EN (sequencer enable) and R6:0/DN (sequencer done) are available to the PLC Programmer. The Enable coil (R6:0/EN) is energized when the sequencer is on. The Done coil (R6:0/DN) will energize when the sequencer has completed stepping through the number of steps specified for the SQO function. Contacts with these coil addresses may be used to turn on or off PLC functions and/or PLC output devices. The RES (reset) function in Figure 1 is used to reset the SQO function.

The input I:0/0 in the circuit displayed in Figure 1 is a manual pushbutton. In order to advance the SQO to the next step, one must close the Normally Open (N.O.) input manually. Figure 3 illustrates how to use a timer function to automatically advance the sequencer step once every second [2]. When normally open (N.O.) input zero (I:0/0) is closed, timer ON-Delay function zero (T4:0) will start operating. Notice that the timer zero preset register (T4:0.pre) holds the number 100. The time base for timer zero (T4:0) is set for 0.01 seconds. Therefore the preset time value for timer zero is one seconds ($100 \times 0.01 = 1$). One second after starting the timer, the timer Done status bit (T4:0/DN) will energize. Then the normally open (N.O.) timer done contact (T4:0/DN) in rung one (Rung 0001) will close and the sequencer output (SQO) function's position will advance to next step.

Also notice that the normally closed (N.C.) timer done contact (T4:0/DN) in rung zero (Rung 0000) opens and resets the Timer ON-Delay function. When the Timer ON-Delay function resets, its timer Done status bit coil (T4:0/DN) de-energizes. Then, the normally closed (N.C.) timer Done status contact (T4:0/DN) returns to its normal position (closed) and the timer is energized to start timing again.

Variable Time Intervals Between SQO Steps

In the ladder logic diagram displayed in Figure 3, time intervals between sequencer steps were constant (i.e., one second). In order to have variable time intervals between sequencer steps, one must utilize a second sequencer. Figure 4 contains a ladder logic diagram in which the sequencer will step through a set of events that have variable time intervals between them.

A second sequencer will control the time intervals. The new sequencer file is integer data file seven (#N7). The new sequencer destination register is the accumulated register for timer zero (T4:0.acc). The contents of integer data file seven (#N7) are, 100, 250, 350, 450, 100. Since the time base for the timer zero (T4:0) is 0.01, the time between steps will be 1, 2.5, 3.5, 4.5, and 1 seconds. For example, the delay time between starting step (i.e., step zero) and the first step is one (1) second, and the delay time between step one (1) and two (2) is two and half (2.5) seconds. Note that both sequencers must be synchronized to step through the patterns simultaneously.

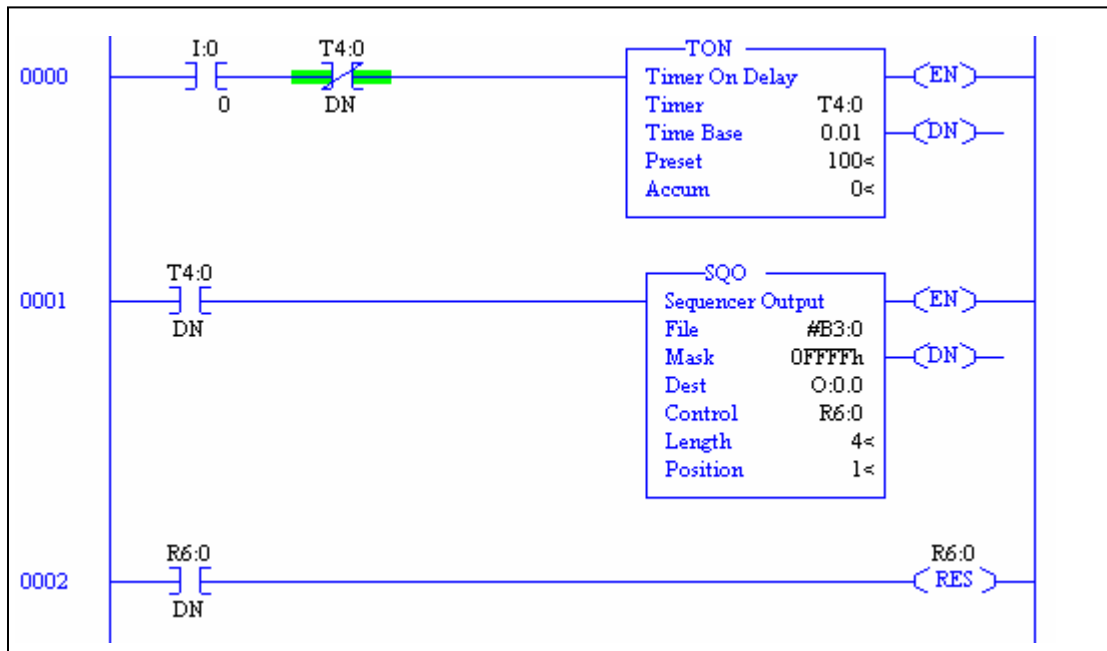


Figure 3. Placing Fixed Time Intervals between Sequencer Steps

Increasing Number of Steps and Outputs

Each sequencers in the Allen-Bradley SLC 500 series PLC can control a 16-bit destination register [3]. This means that in order to control more than sixteen (16) discrete output, one must connect two sequencers in parallel such that they step through the defined pattern simultaneously [1]. Figure 5 displays the connection of two sequencers so that thirty-two (32) output can be stepped through the defined patterns residing in data file #B3. Note that each output modules two and three (O:2 and O:3) in Figure 5 have sixteen (16) output ports. Therefore, thirty-two (32) output ports are controlled by the parallel connection of the two sequencers. Sequencers in Figure 5 are controlled by the R6:0 and R6:1 registers. They both are stepped through the patterns which reside in data file #B3. Input I:1/0 is used to energize both sequencers simultaneously.

In order to increase the number of steps, one must cascade sequencers [1]. Figure 6 shows a connection for which the number of steps may be increased to 512. This means that there are 256 steps stored in data file #B3, and an additional 256 steps stored in data file #B10. Note that when the first sequencer has completed the 256 steps (i.e., step 0 through step 255) residing in data file #B3, then the second sequencer will start stepping through the last 256 steps which is in data file #B10.

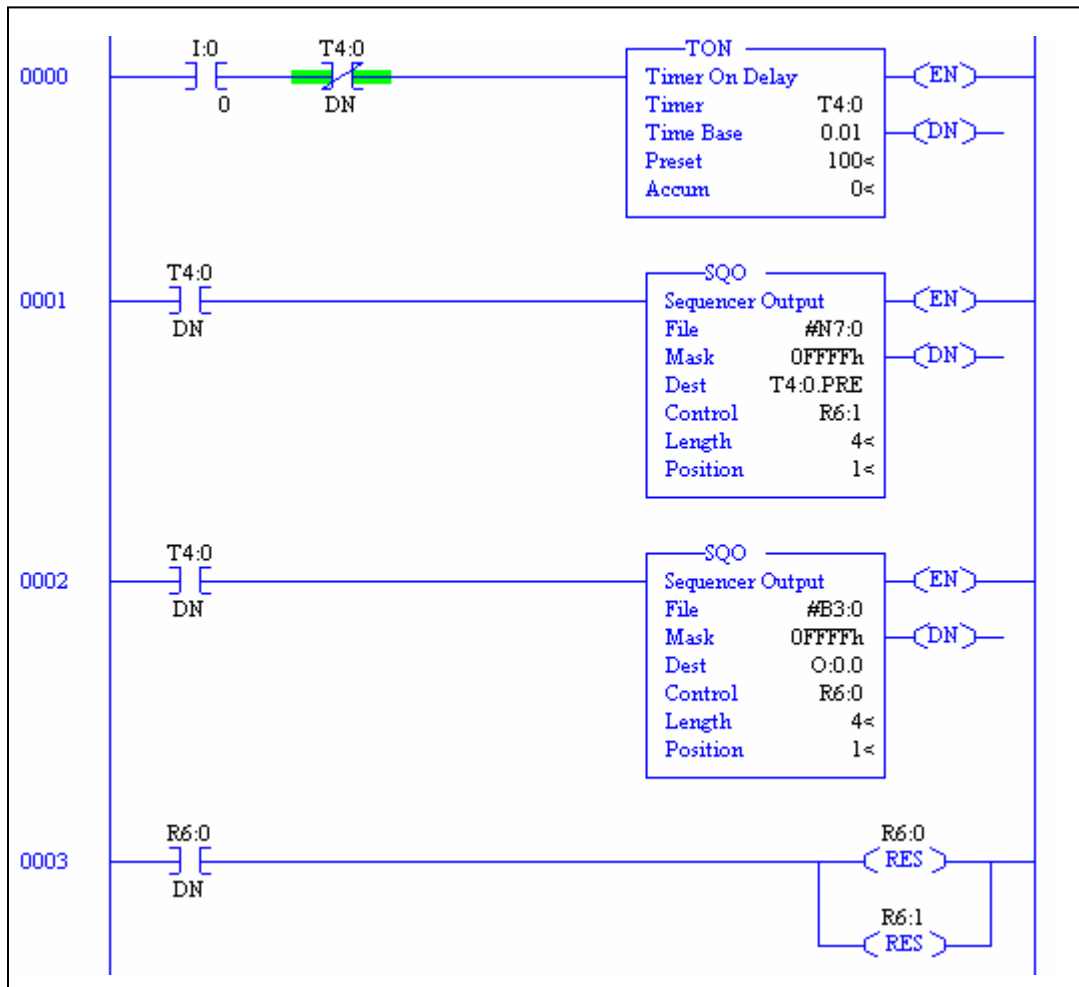


Figure 4. Variable Time Intervals between Sequencer Events

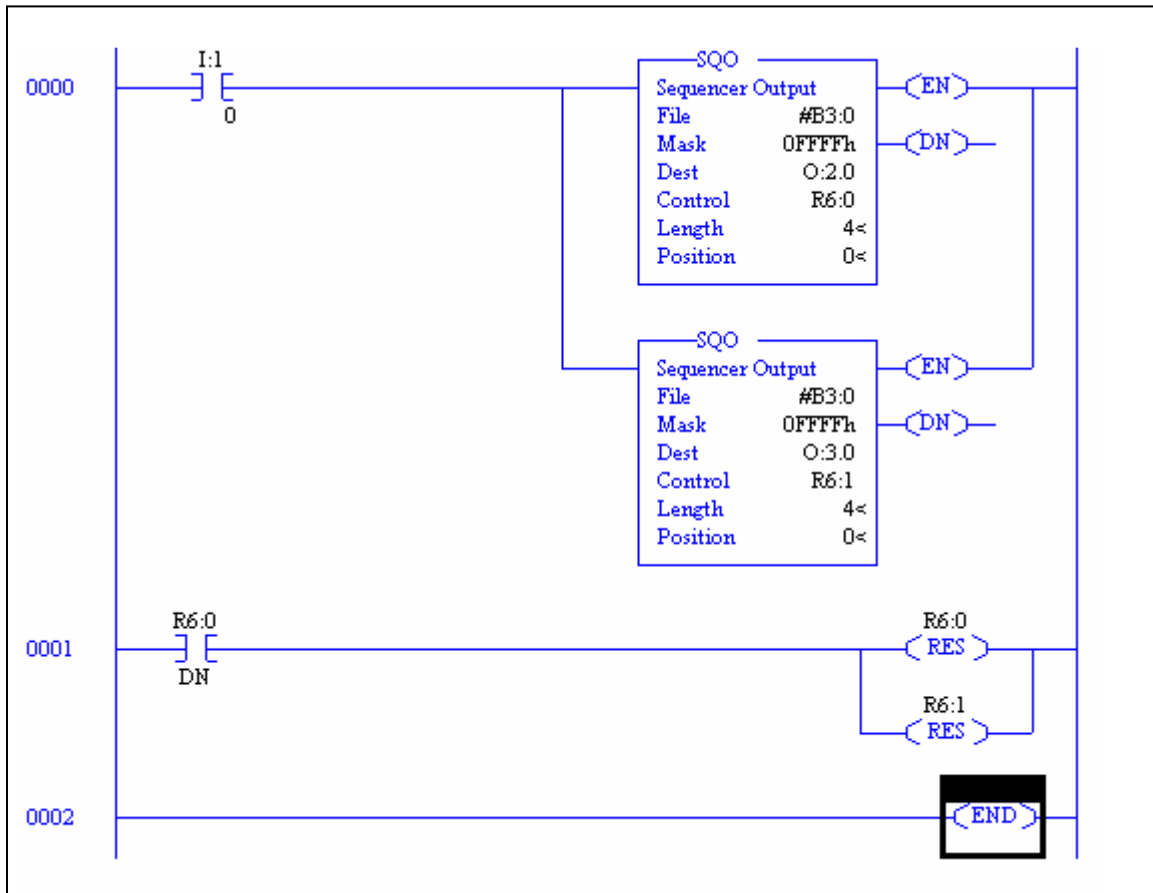


Figure 5. Connecting Sequencers in Parallel to Increase Number Of Outputs

Configuring Sequencer Load (SQL) Function

Sequencer load function (sometimes called Sequencer input function) is used to collect data from input modules. Figure 7 displays a ladder logic diagram that illustrates how to read the input module I:1 every one minute (i.e., 60 seconds) and record the data in data file #B3:0. Notice that 256 data points can be stored in the bit three data file (#B3.) One can increase the capacity by cascading two or more sequencers.

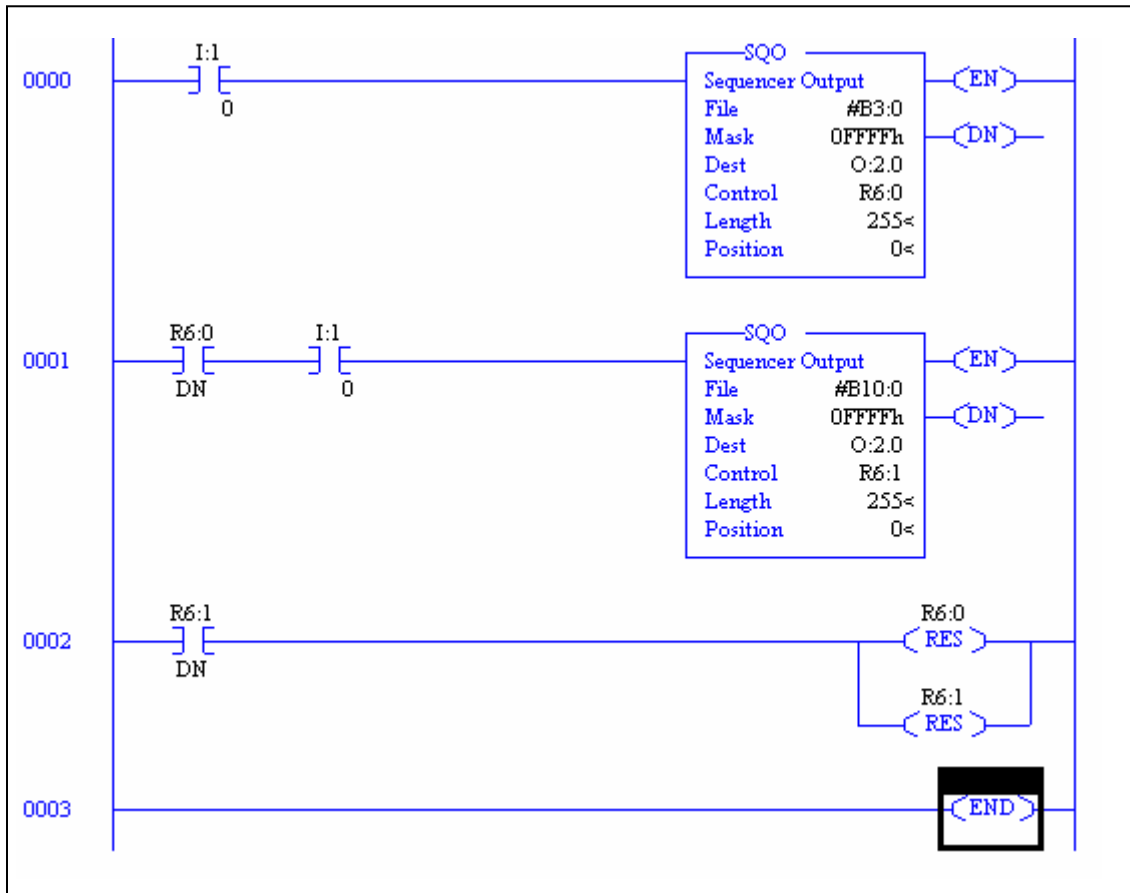


Figure 6. Cascade Connection of Sequencers for Increasing Number Of Steps

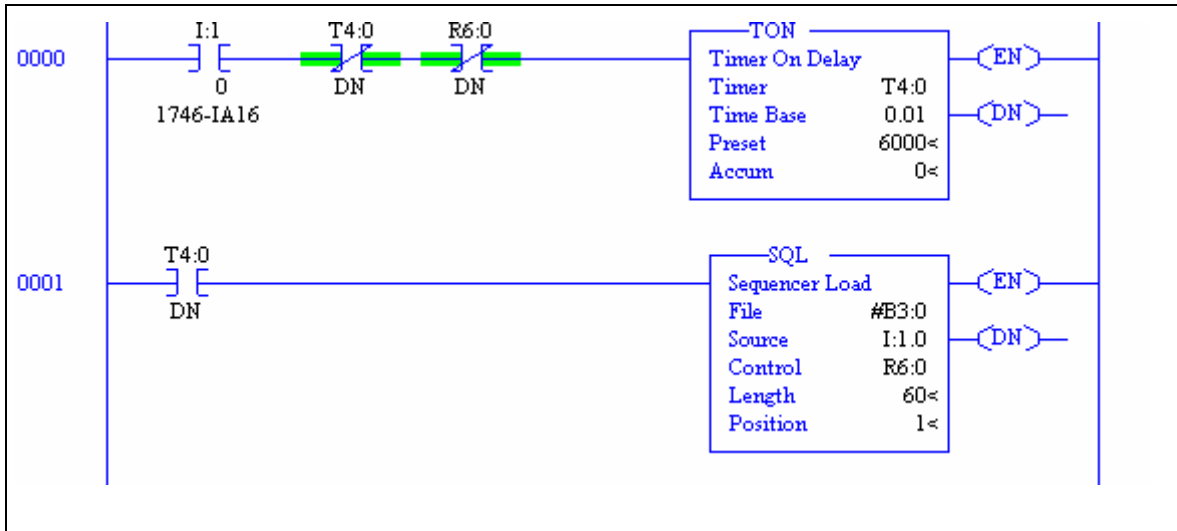


Figure 7. PLC Ladder Logic Diagram for a Sequencer Load (SQL) Application

Configuring Sequencer Compare (SQC) Function

The sequencer compare function is used to read an input module. It will then compare the data to data stored in the sequencer data file. If the collected data from the input module match the data in the data file, the “Bit Found” coil will energize. Figure 8 illustrates how to use the SQC function to turn on an output if the input word is the same as the word in the data file #B3.

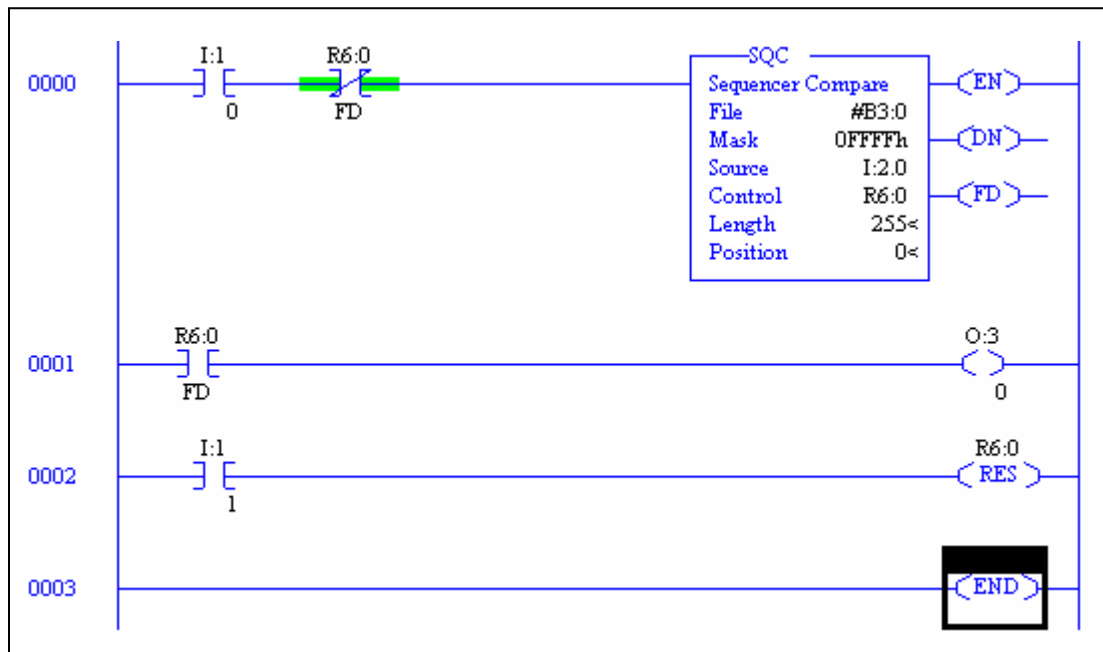


Figure 8. PLC Ladder Logic Diagram for a Sequencer Compare Application

Conclusion

This paper described how to use three (3) very important sequencer functions. These functions are Sequencer Output (SQC), Sequencer Load (SQL), and Sequencer Compare (SQC). Sequencer functions are utilized in many PLC ladder logic diagrams for controlling industrial systems. We illustrated how to configure these functions in PLC ladder logic diagrams. In a programmable logic controller course, students should learn how to use these functions to control industrial systems. Students should also know how to increase the number of output and the number of sequencer steps. These techniques were also illustrated in this paper. Several laboratory experiments pertaining to sequencer functions are assigned to the students during the course of an academic quarter. Student feedbacks have been very positive for integrating their knowledge gained during the classroom lectures in the laboratory.

Bibliography

1. *Programmable Logic Controllers; Principles and Applications*, By John Webb and Ronald Reis, Fifth Edition, 2003, Published by Prentice-Hall, Upper Saddle River, NJ.
2. *Programmable Logic Controllers; Hardware and Programming*, By Max Rabiee, 2002, Published by Goodheart-Willcox (G-W) Publishing Company, Tinley Park, IL.
3. *Allen-Bradley Reference Manual (2001)*. Allen-Bradley SLC500 and MicroLogic-1000 Instruction Set, Rockwell Automation, Milwaukee, WI.

Biography

Max Rabiee earned his Ph.D. in Electrical Engineering from the University of Kentucky in 1987. He is an Associate Professor of Electrical and Computer Engineering Technology at the University of Cincinnati. Dr. Rabiee is a registered professional engineer, and a senior member of the Institute of Electrical and Electronic Engineering (IEEE). He is also a member of the American Society of Engineering Education (ASEE), the National Association of Industrial Technology (NAIT), the Eta Kappa Nu Electrical Engineering Honor Society and the Tau Beta Pi Engineering Honor Society.