UNIT 18
INSTRUCTOR'S MANUAL

OUTPUT SEQUENCING

1. Output sequencing requires a great deal more paperwork than other types of programming. Be sure to allow the students ample time to complete the program design and entry in the Experiment. You may want to plan on two instructional sessions for this unit.

2. To understand how the sequencer is more efficient than cascading timers, this is a sequencer program to replace the cascading timers program used in Unit 15. While the sequencer may take a little more time to plan and program, it uses only one relay within the PLC. This means that the remaining relays can be used to control other parts of a large program.

UNIT 18

OUTPUT SEQUENCING

Objectives

Upon completion of this unit the student will be able to:

1. Explain the operation of an output sequencer.
2. Explain the difference between a time driven sequencer and an event driven sequencer.
3. Write a program using an event driven sequencer.
4. Write a program using a time driven sequencer.
5. Enter sequencer programs into the PLC.

Background

In previous units cascading timers were used to control sequences of outputs. This programming strategy required the use of many timers. It limited how many timers could be used elsewhere in the program. To overcome these limitations of cascading timers, the sequencer function was developed.

The *sequencer* is a special relay function which has many cascaded timers or counters within a single relay element. It allows the PLC user to activate output elements in a specified series at predetermined time intervals or after a preset number of events. A typical sequencer may have up to 100 steps and can control at least six outputs. A specialized addressing system is used and will be discussed in the keystroke sections.

There are two basic types of sequencers. Time driven sequencers use preset time intervals to control when outputs are activated and de-activated. Like cascading timers which resets, a time driven sequencer program runs continuously until the controlling input is turned off. A time-driven sequencer operates similarly to a mechanical drum switch that increments automatically at a preset time period

Event driven sequencers can be thought of as cascading counters. Similar to a counter, turning the controlling input on and off a preset number of times increments the sequencer to the next step in its internal series. An event-driven sequencer operates similarly to a mechanical stepper switch that increments one step for each pulse applied to it.

The use of sequencers greatly simplifies the ladder logic diagram. Instead of 104 rungs representing the 104 steps in the sequencer, one rung is used (see Figure 18-1).

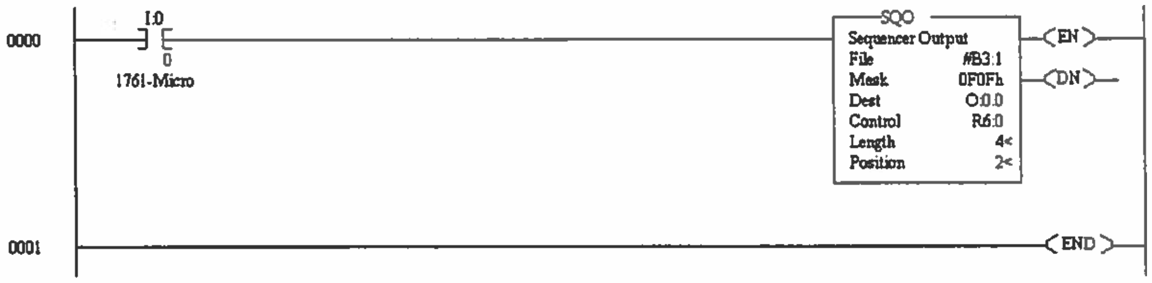


Figure 18-1
Output Sequencer Rung

All of the commands are steps within the sequencer itself. They do not need to be represented on the rung. However, the exact operation of the sequencer must be programmed. A sequencer programming chart contains the information concerning outputs and time delays or event counts. A sequencer programming chart is shown in Figure 18-2. Procedures for coding the sequencer program and entering it to the PLC will be covered in the keystroke section.

To program a sequencer, instructions take the form of a single high level instruction. a memory location is designated within the PLC that forms the desired pattern of the outputs during the programmed sequence.

When a sequencer operates on a entire output word, there may be outputs associated with the word that do not need to be controlled by the sequencer. They could be used elsewhere in the program.

To prevent the sequencer from controlling these bits of the output word, a *mask* word is used. The use of a mask word is illustrated in Figure 18-2. The mask word selectively screens out the data from the sequencer word file to the output word. In this example, for each bit of output word 050 that the sequencer is to control, the corresponding bit of mask word 040 must be set to 1. All other bits of output word 050 are set to 0 and can be used independently of the sequencer.

The advantage of sequencer programming over conventional programming is the significant savings of memory words. Typically the sequencer program does in 20 words or less what a standard program does in 100 words or more.

	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	
Word 050	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Output
Word 040	0	0	0	0	1	0	0	1	1	0	0	1	1	0	0	1	Mask
Word 060	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	Step 1
Word 061	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	1	Step 2
Word 062	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	Step 3
Word 063	0	0	0	0	0	0	0	1	0	0	0	1	1	0	0	0	Step 4

Figure 18-2
Use of a Mask Word

Sequencers may also be used to control the order in which the PLC responds to input elements. Input sequencers are beyond the scope of the unit. All discussions of input sequencing is reserved for advanced PLC programming techniques.

Programming Keystrokes

There are 3 types of sequencers in the Allen-Bradley MicroLogix Controller.

1. Sequencer Output --(SQO)--
2. Sequencer Compare --(SQC)--
3. Sequencer Load --(SQL)--

The output sequencers operate as described. They are addressed with the numbers # B3:0/0 to # B3:0/31 or # N7/0 to # N7/103. All output sequencer programming depends upon conversion of binary numbers to hexadecimal numbers, as you did in Unit 4. If you need to, review the conversion method. The rest of this unit will be easier to understand if you are sure of your conversions.

Sequential Output (SQO) and Sequential Compare (SQC)

These instructions transfer 16-bit data to word addresses for the control of sequential machine operations.

Entering Parameters

Enter the following parameters when programming these instructions:

- File is the address of the sequencer file. You must use the file indicator for this address. (Computer) Sequencer file data is used as follows:

Instruction	Sequencer File Stores
SQO	Data controlling outputs
SQC	Reference data for monitoring inputs

- Mask (SQO, SQC) is a hexadecimal code or the address of the mask word or file through which the instruction moves data. Set mask bits to pass data and clear mask bits to prevent the instruction from operating or corresponding destination bits. Use a mask word or file if you want to change the mask according to application requirements. If the mask is a file, its length will equal to the length of the sequencer file. The two files track automatically.

- Source is the address of the input word or file for a SQC from which the instruction obtains data for comparison to its sequencer file.

- Destination is the address of the output word or file for a SQO to which the instruction moves data from its sequencer file
  Important: You can address the mask, source or destination of a sequencer instruction as a word or file. If you address it a file (using file indicator #), the instruction automatically steps through the source, mask or destination file.

- Control (SQO, SQC) is the control structure that stores the status byte of the instruction, the length of the sequencer file and the current position in the file. You should not use the control address for any other instruction.

	15	13	11	08	00
Word 0	EN	DN	ER	FD	
Word 1	Length of sequencer file				
Word 2	Position				

Status bits of the control structure include:

- ➢ Found bit FD (bit 08)- SQC only. When the status of all non-masked bits in the source address match those of the corresponding reference word, the FD bit is set. This bit is assessed each time the SQC instruction is evaluated while the rung is true.

- ➢ Error bit ER (bit 11) is set when the controller detects a negative position value or a negative while the rung is true.

- ➢ Done Bit DN (bit 13) is set by the SQO or SQC instruction after it has operated on the last word in the sequencer file. It is reset on next false-to-true rung transition after the rung goes false.

- ➢ Enable EN (bit 15) is set by a false-to-true-rung transition and indicates the SQO or SQC instruction is enabled

- Length is the number of steps of the sequencer file starting at position 1. The maximum number you enter is 104 words. Position 0 is the startup position. The instruction resets (wraps) to position 1 at each cycle completion.

- Position is the word location or step in the sequencer file from or to which the instruction moves data. A position value that points past the end of the programmed file causes a runtime error.

Using SQO

This output instruction steps through the sequencer file whose bits have been set to control various output devices.

When the rung goes from false-to-true, the instruction increments to the next step (word) in the sequencer file. Data stored there is transferred through a mask to the destination address specified in the instruction. Data is written to the destination word every time the instruction is executed.

The done bit is set when the last word of the sequencer file is transferred. On the next false to true rung transition, the instruction resets the position to step one.

At start up, if the position is equal to zero when switching the processor from the program to the Verify, Download, Go Online and Run mode, instruction operation depends on whether the rung is TRUE or FALSE on the first scan.

- If true, the instruction transfers the value in step zero.
- If false, the instruction waits for the first rung transition from false-to-true and transfers the value in step one.

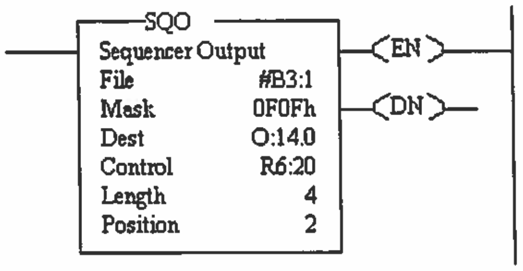
The bits mask data when reset and pass data when set. The instruction will not change the value in the destination word unless you set mask bits.

The mask can be fixed or variable. When a hexadecimal code is entered it is fixed. If an element address or file address for changing the mask with each step is entered it is variable.

Sequencer output instructions can be either block or coil formatted and contain a counter or timer and a file. The instructions require entering more than one address. The instructions require entering more than one address. Sequencer instructions are normally retentive, and there is an upper limit as to the number of external outputs and steps that can be operated upon by a single instruction. Many sequencer instructions reset the sequencer automatically to step 1 upon completion of the last sequence step. Other instructions provide an individual reset control line or combination of both

The figure below shows how the sequencer output instruction works. In the example, the bit data file (B 10:1 through B10:5) contains the data for each step of the sequencer controlled by the sequencer instruction. The bit patterns stored in each of these locations form the output pattern that will be seen for each of the sequencer steps.

<u>Ladder Representation</u>



Entering the instruction

Sequence Output (SQO)

Destination O:14

15	8	7	0
0000	0101	0000	1010

Mask Value 0F0F

15	8	7	0
0000	1111	0000	1111

Sequencer Output file #B10:1

Word					Step
B10:1	0000	0000	0000	0000	0
2	1010	0010	1111	0101	1
3	1111	0101	0100	1010	2
4	0101	0101	0101	0101	3
5	0000	1111	0000	1111	4

Note: To view Hex valves, change the Radix Window to the pull down HEX Window when a Hexadecimal Radix is set, it will remain that way until the operator changes it back to Binary by pulling down the Binary Window.

External Outputs associated with O:14

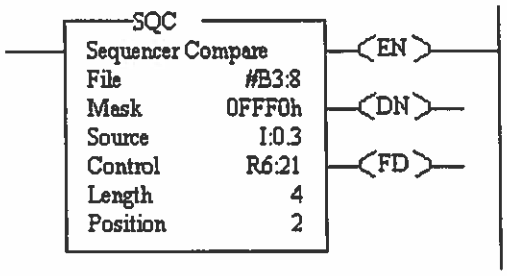
00	
01	← ON
02	
03	← ON
04	
05	
06	
07	
08	← ON
09	
10	← ON
11	
12	
13	
14	
15	

Using SQC

When the status of all non-masked bits in the source word match those of the corresponding reference word, the instruction sets the found bit (FD) in the control word. Otherwise, the found bit (FD) is cleared.

The bits mask data when reset and pass data when set.
The mask can be fixed or variable. If you enter a hexadecimal code, it is fixed. If you enter an element address or file address for changing the mask with each step, it is variable.

When the rung goes from false to true, the instruction increments to the next step (word) in the sequencer file. Data stored there is transferred through a mask and compared against the source of equality. While the rung remains true, the source is compared against the reference data for every scan. If equal, the FD bit is set in the SQCs Control counter.



Entering the instruction

Sequencer Compare (SQC)

Input Word I:3

0010	0100	1001	1101

Mask Value FFF0

1111	1111	1111	0000

Sequencer Ref File #B10:11

Word					Step
B10:11					0
12					1
13	0010	0100	1001	1010	2
14					3
15					4

The SQC FD bit is set when the instruction detects that an input word matches (through mask) its corresponding reference word.

The FD bit R3/FD is set in this example, since the input word matches the sequencer reference value using the mask value.

Sequencer Load (SQL)

The SQL instruction stores 16-bit data into a sequence load file at each step of sequencer operation. The source of this data can be an I/O or internal word address, a file address, or a constant.

Entering parameters

Enter the following parameters when programming this instruction:

- File is the address of the sequencer file. You must use the file indicator (#) for this address.

- Source can be a word address, file address, or a constant (-32,768 to 32767)
  If the source is a file address, the file length equals the length of the sequencer load file. The two files step automatically, according to the position value.

- Length is the number of steps of the sequencer load file (and also of the source if the source address is a file address), starting at position 1. The maximum number you can enter is 104 words. Position 0 is the startup position. The instruction resets to position 1 at each cycle completion.

- Position is the word location or step in the sequencer file to which data is moved.

- Control is a control file address. The status bits, length value and position value are stored in this element. Do not use the control file address for any other instruction.

- The control element is shown below:

	15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00
Word 0	EN DN ER
Word 1	Length
Word 2	Position

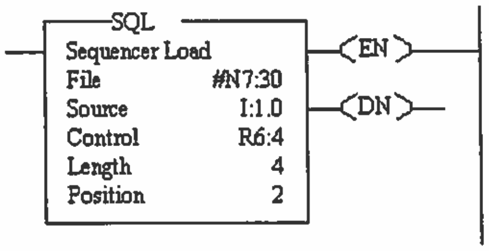
Status bit of the control structure include:

-- Error Bit ER (bit 11) is set when the controller detects a negative position value, or negative value or zero value.

-- Done Bit DN (bit 13) is set after the instruction has operated on the last word in the sequencer load file.

-- Enable Bit EN (bit 15) is set on a false- to -true transition of the SQL rung and reset on a true-to false transition.

Ladder Representation



Entering the instruction

Sequence Load (SQL)

Source I:1.0

15	8	7	0
0000	0101	0000	1010

Sequencer Load File # N7:30

N30	0000	0000	0000	0000	0
31	1010	0010	1111	0101	1
32	0000	0101	0000	1010	2
33	0000	0000	0000	0000	3
34	0000	0000	0000	0000	4

When rung conditions change from false to true, the SQL enable bit (EN) is set. The control element R4 increments to the next position in the sequencer file, and loads the contents of source I0 into the corresponding location in the file. The SQL instruction continues to load the current data into this location each scan that the rung remains true. When the rung goes false, the enable bit (EN) is reset.

The instruction loads data into a new file element at each false-to-true address transition of the rung. When step 4 is completed, the done bit (DN) is set. Operation cycles to position 1 at the next false -to -true transition of the rung after position 4.

If the source were a file address such as #N40 , FILES #N40 and #N30 would have a length of 5 (0-4) and would track through the steps together per position value.

Entering Output Sequencer Programs

Figure 18-4 is the ladder logic rung for the sequencer in this example. To enter the rung into the PLC, use the keystrokes shown, much as you would for any other relay coil.

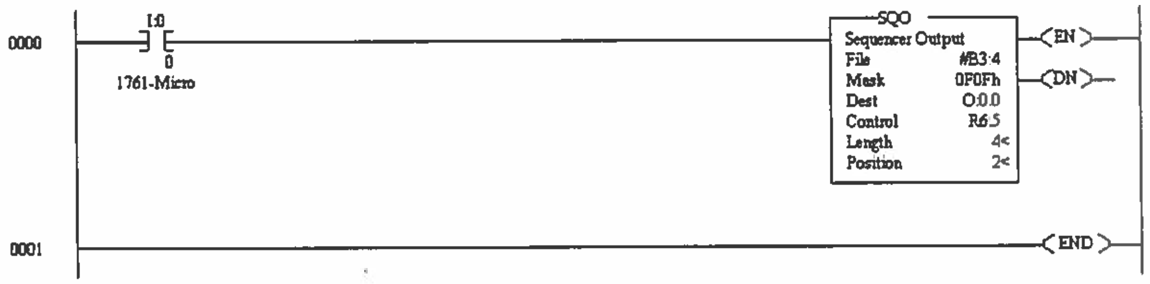


Figure 18-4

In general, to program an output sequencer the following keystroke sequence is used:

New Rung
Examine if Closed I:0/0
Sequencer Output (SQO)
B3:4
0F0FH
O:0
R6:5
4
2

The sequencers in the MicroLogix are retentive. When a sequencer is stopped and restarted, it will begin at the point where it left off, just like a retentive timer. To return a sequencer to the beginning of its cycle, a reset rung is needed, as in Figure 18-5. The reset is programmed exactly as it would be for any timer or counter, using the same address as the sequencer itself.

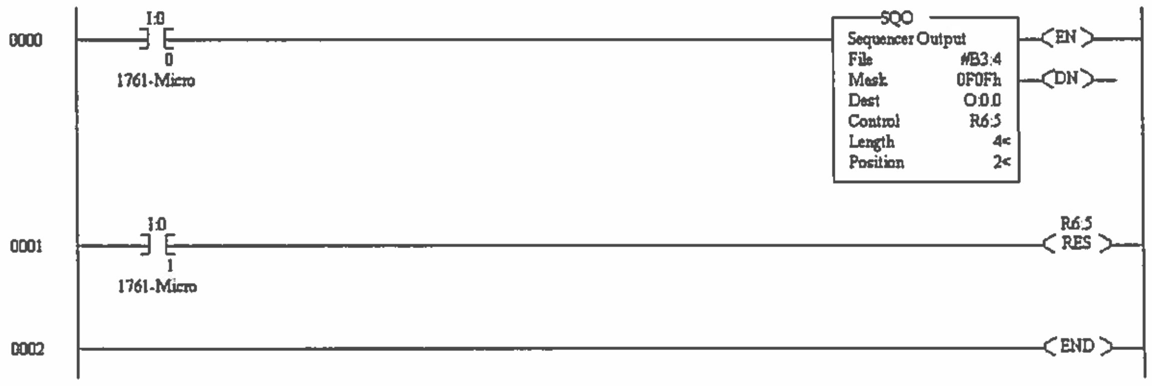


Figure 18-5
Output Sequencer with Reset Rung

The output sequencer in the MicroLogix reset to step 0, whether or not the input controlling the sequencer is energized when the reset is activated. Also upon initiating a sequencer program, the sequencer "waits" at step 0 until the sequencer input element is energized. This means that all Step 0 will be executed as soon as the controller is put in RUN mode.

Many applications of output sequencers require that the sequencer reset itself or initiate another event at the end of its cycle. The cycle completion bit or DN bit, is used for these purposes. It is a special address associated with each sequencer. It turns on whenever the sequencer reaches the end of its last step and remains off at all other times. The address of the cycle completion bit is the control register done bit in the example. The cycle completion bit can also be used to count the number of cycles completed by the output sequencer, as shown in Fig 18-6.

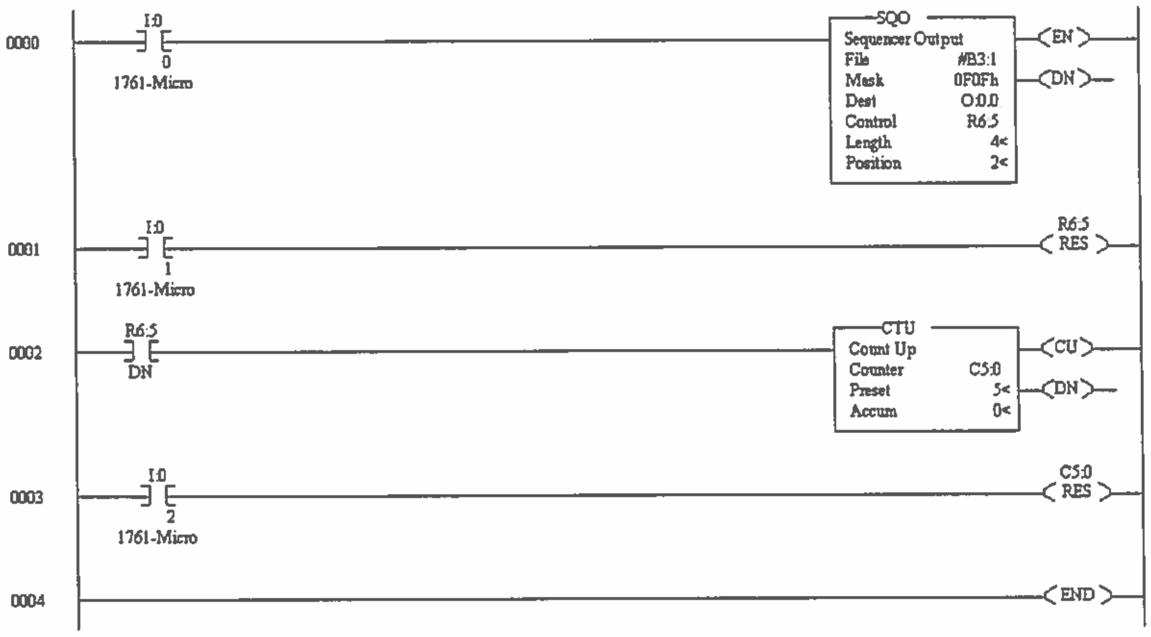


Figure 18-6
Output Sequencer with reset and Cycle Count

The MicroLogix 1000 is an event driven sequencer. Time sensitive events can be executed by using the done bit of a timer as the trigger event for the sequencer. Unfortunately this creates a condition where each time cycle is constrained to the time of the shortest event. As an example, if it takes one second for a robot gripper to close, but three seconds for the arm to rotate between positions, the programmer would need to allow one cycle for the gripper event, but three cycles for the rotational event. An example of this is provided below. Notice that the mask for an event will stay the same for several steps before changing. In this case the gripper event is assigned a mask of 0100, and the rotational event is assigned a mask of 0001.

Step 1	0000	0000	0000	0000	1000
Step 2	0000	0000	0000	0000	1001
Step 3	0000	0000	0000	0000	1001
Step 4	0000	0000	0000	0000	1001

Table 18-3
Time Driven Sequencer Table

Monitoring Output Sequencers

Output sequencers are monitored in much the same way as timers and counters. Once the element is located, use the arrow keys to access sequencer data and steps. Each step is identified and must be monitored separately.

EXPERIMENT

Purpose

To design, enter and run an output sequencing program.

Procedure

1. In Unit 15 you used cascading timers to control a robot arm. You will now develop a program to control the same robot arm using output sequencing.

   First, carefully review the time listing for the arm (Table 15-4, Page 15-12). Also, a program to control the same robot arm using output sequencing.

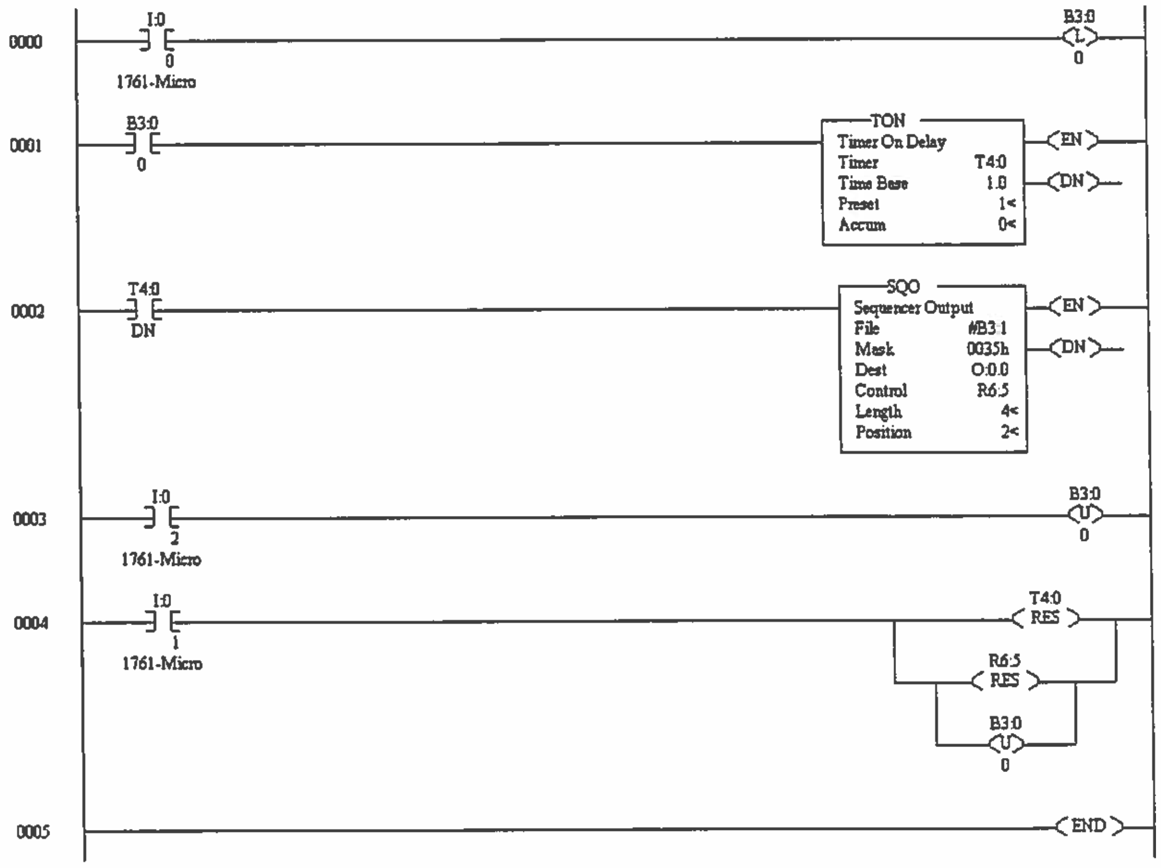
The motions of the robot arm will now be controlled as in Table 18-1. The robot's home position is up, arm in and left, with the gripper open.

Output ON	Motion
O:0/0	Arm out
O:0/2	Arm down
O:0/4	Gripper close
O:0/5	Arm right

Table 18-1
Outputs Controlling Robot Arm Motions

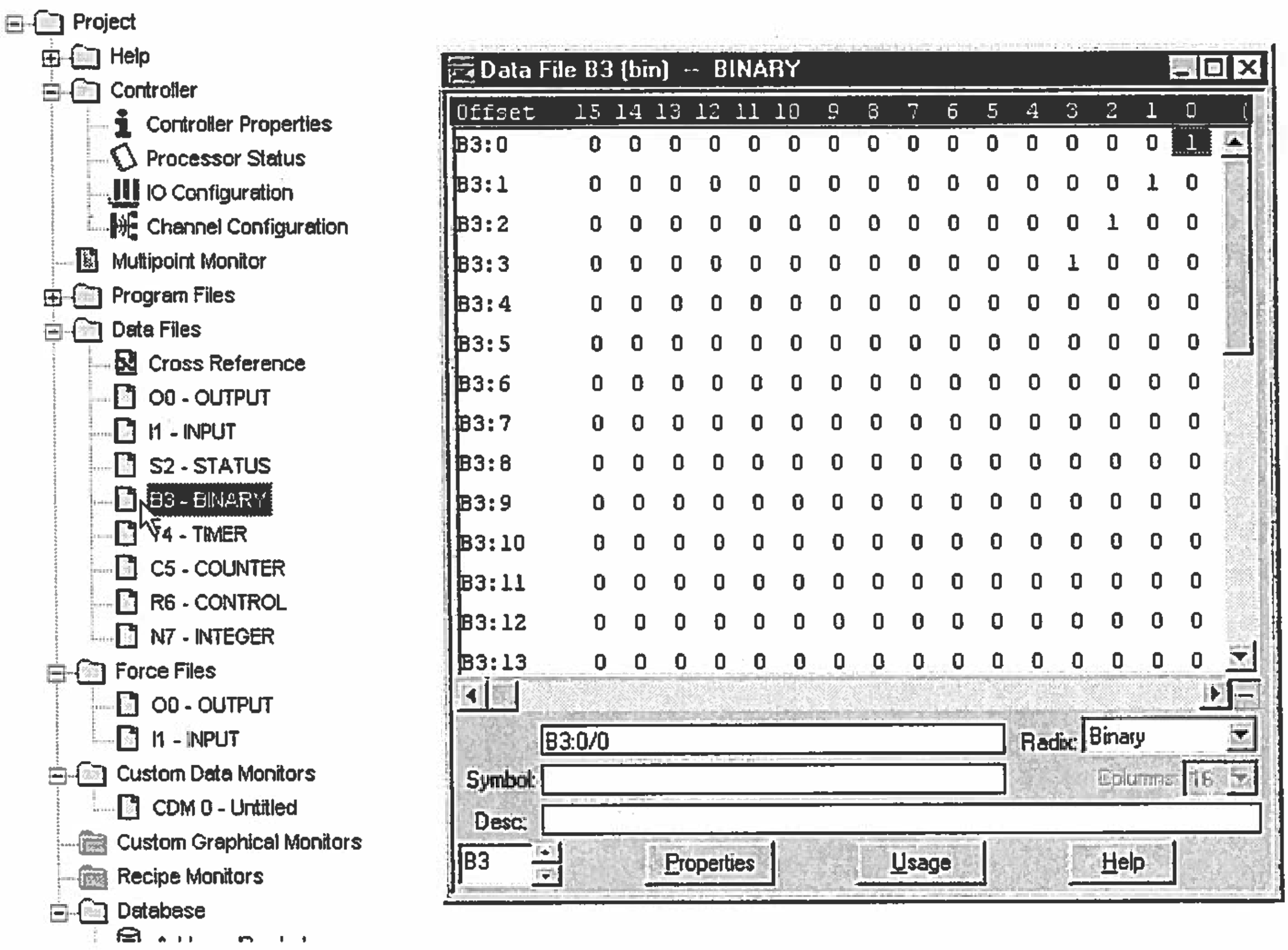
2. Refer to Tables 15-4 and 18-1 as you work. Design the sequencer portion of the program to control the arm. Keep the exact same motion sequence as in Unit 15.

3. Draw the ladder logic diagram for the entire sequencer program. Input I:0/0 must activate the sequencer, latching it on. Input I:0/1 must reset the sequencer. Input I:0/2 should be used to stop the sequencer.



(Relay addresses may vary within their acceptable ranges.)

4. The motions of the robot arm will be controlled in the Data File. The Data file is located on the left side of the screen in the Project tree. Under the Data Files select B3 – BINARY. The Data File B# (bin) -- BINARY window will show the steps of the sequencer. Refer to Figure 18-7. Turning on the outputs can be accomplished by changing a 0 to 1 or 1 to 0. In step 2 (B3:2) output #2 turns on. In step 3 (B3:3) output #2 is off and output #3 turns on.



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

B3:0/0 Radix: Binary
Symbol: Columns: 16
Desc:
B3

5. As you did in Unit 15, use lights to represent the three arm motions and the gripper (see Table 18-2)

Output	Light
O:0/0	Red
O:0/2	Yellow
O:0/4	Green
O:0/5	White

Table 18-2
Simulation Outputs

Use the two normally open momentary switches and one normally open maintain switches to control the sequencer's operation. Make all of the necessary connections.

Identify the switch to be used with each input element:

a. I:0/0 *Either normally open momentary switch.*

b. I:0/1 *The other normally open momentary switch.*

c. I:0/2 *Either normally open maintain switch.*

6. Enter and test your sequencer program.

   What will happen when you activate input I:0/1 ?

   Initially the input I:0/1 will unlatch the sequencer. The sequencer will reset to Step 0. As soon as input I:0/1 is de-activated the sequencer will restart at step 0.

7. Test the operation of input I:0/1 and make sure your explanation in step 5 is correct.

8. Monitor the sequencer at its step 5. When the sequencer is executing Step 5, energize input I:0/2. What happens to the sequencer?

   The sequencer stops at its step 5. Outputs are held on as set in step 5.

9. Restart the sequencer.

a. Record the step on which the sequencer begins operation. *8*

b. What must you do to stop the sequencer mid-cycle and then restart it at the beginning of its cycle?

First, activate input I:0/2 to stop the sequencer. Next activate and deactivate input I:0/1 to reset the sequencer. Last , de-activate input I:0/2 to restart the sequencer.

10. Test your answer to Step 8.b

11. Enter the program and put the controller in RUN mode. Access File #B3:0 so that you can modify it. In binary, the modifications will be:

B 3:0 0000 0000 0000 0000
B 3:1 0000 0000 0000 0000
B 3:2 0000 0000 0000 0001
B 3:3 0000 0000 0000 0010
B 3:4 0000 0000 0000 0100
B 3:5 0000 0000 0000 1000

This bit configuration sets up the sequencer to pass one true bit at a time through the mask.

12. Access Element B3 and prepare to modify it.

a. What would happen if B3:6 remained at its current value? Note its hexadecimal mask value.

N3:6 is all zeros, so no data can be passed through the mask. The outputs would not be activated.

b. What should the hexadecimal mask be if you want to use outputs O: 2/0 through O:2/5 ? Show the mask in 16 bit binary as well.

003F - 0000 0000 0011 1111

c. Modify the mask value to 000Fh. Describe what happens to the outputs?

Outputs 4 and 4 are masked out and become inactive.

13. When you are comfortable with the sequencer's operation, put aside the hardware. Then answer the Questions below.

Questions

1. What advantages does a sequencer have over cascading timers?

Sequencers avoid tying up many timers which might be needed elsewhere in the program.

2. What are some important considerations when programming an SQC instruction?

Event driven or time driven;

Exact mask requirements;

Length must be one longer than number of bits to be sequenced;

Order in which outputs are to operate.

3. Can step data be changed while a sequencer is running?

Yes

4. Why is it necessary to mask unused outputs in a sequencer group ?

If unused outputs are not masked in the sequencer, they cannot be used anywhere else in the program while the sequencer is in operation.

5. How are File and Mask used in a sequencer?

File identifies which bits are held true at each step in the sequence. This is determined by the conversion of outputs to be activated at each step. The mask is used to identify precisely which file bit will pass data to the outputs.

UNIT 19
INSTRUCTOR'S MANUAL

SHIFT REGISTERS

1. Shift registers are a complex function of the PLC. Sequencers are combined with the use of zone control relays to make a shift register operational. Different programming strategies are used for sequencers which operate in conjunction with shift registers. These should be covered with great care before the students are asked to complete the Experiment.

2. The Experiment gives the students the opportunity to test shift register operation. It also provides experience in both left type and right type registers. Make certain the students understand the differences in the set and shift inputs.

3. Note that the solution to Question 2 in the Questions excludes part of the problem program. You may have to review the use of relay addresses with the students if they have incorrect answers to the problem.