19. SEQUENTIAL FUNCTION CHARTS

Topics:

- Describing process control SFCs
- Conversion of SFCs to ladder logic

Objectives:

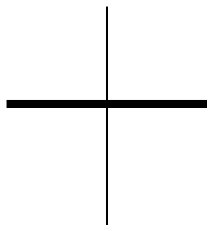
- Learn to recognize parallel control problems.
- Be able to develop SFCs for a process.
- Be able to convert SFCs to ladder logic.

19.1 INTRODUCTION

All of the previous methods are well suited to processes that have a single state active at any one time. This is adequate for simpler machines and processes, but more complex machines are designed perform simultaneous operations. This requires a controller that is capable of concurrent processing - this means more than one state will be active at any one time. This could be achieved with multiple state diagrams, or with more mature techniques such as Sequential Function Charts.

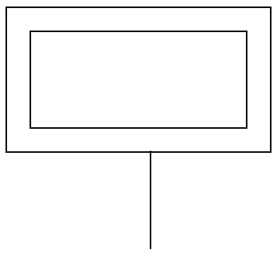
Sequential Function Charts (SFCs) are a graphical technique for writing concurrent control programs. (Note: They are also known as Grafcet or IEC 848.) SFCs are a subset of the more complex Petri net techniques that are discussed in another chapter. The basic elements of an SFC diagram are shown in Figure 292 and Figure 293.

flowlines - connects steps and transitions (these basically indicate sequence)
transition - causes a shift between steps, acts as a point of coordination

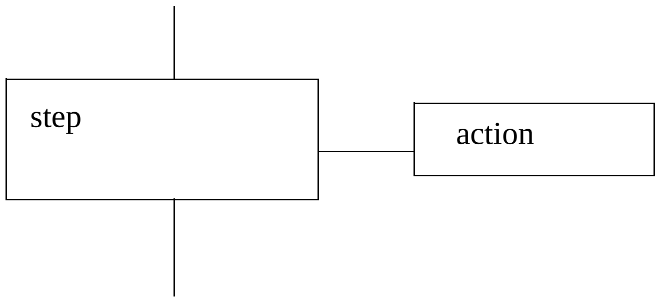


Allows control to move to the next step when conditions met (basically an if or wait instruction)

initial step - the first step



step - basically a state of operation. A state often has an associated action



macrostep - a collection of steps (basically a subroutine

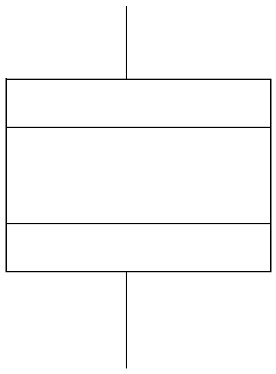
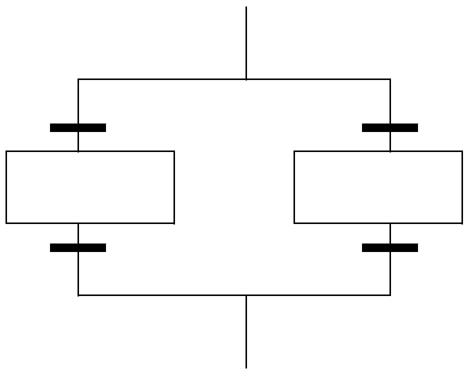


Figure 292 Basic Elements in SFCs

selection branch - an OR - only one path is followed



simultaneous branch - an AND - both (or more) paths are followed

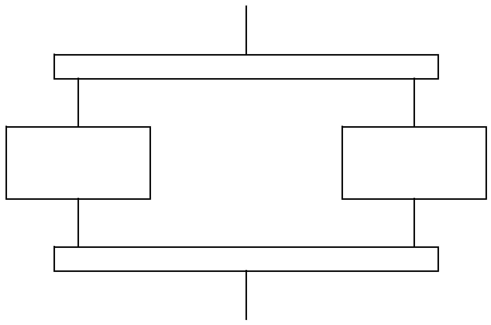


Figure 293 Basic Elements in SFCs

The example in Figure 294 shows a SFC for control of a two door security system. One door requires a two digit entry code, the second door requires a three digit entry code. The execution of the system starts at the top of the diagram at the *Start* block when the power is turned on. There is an action associated with the *Start* block that locks the doors. (Note: in practice the SFC uses ladder logic for inputs and outputs, but this is not shown on the diagram.) After the start block the diagram immediately splits the execution into two processes and both steps 1 and 6 are active. Steps are quite similar to states in state diagrams. The transitions are similar to transitions in state diagrams, but they are drawn with thick lines that cross the normal transition path. When the right logical conditions are satisfied the transition will stop one step and start the next. While step 1 is active there are two possible transitions that could occur. If the first combination digit is correct then step 1 will become inactive and step 2 will become active. If the digit is incorrect then the transition will then go on to wait for the later transition for the 5 second delay, and after that step 5 will be active. Step 1 does not have an action associated, so nothing should be done while waiting for either of the transitions. The logic for both of the doors will repeat once the cycle of combination-unlock-delay-lock has completed.

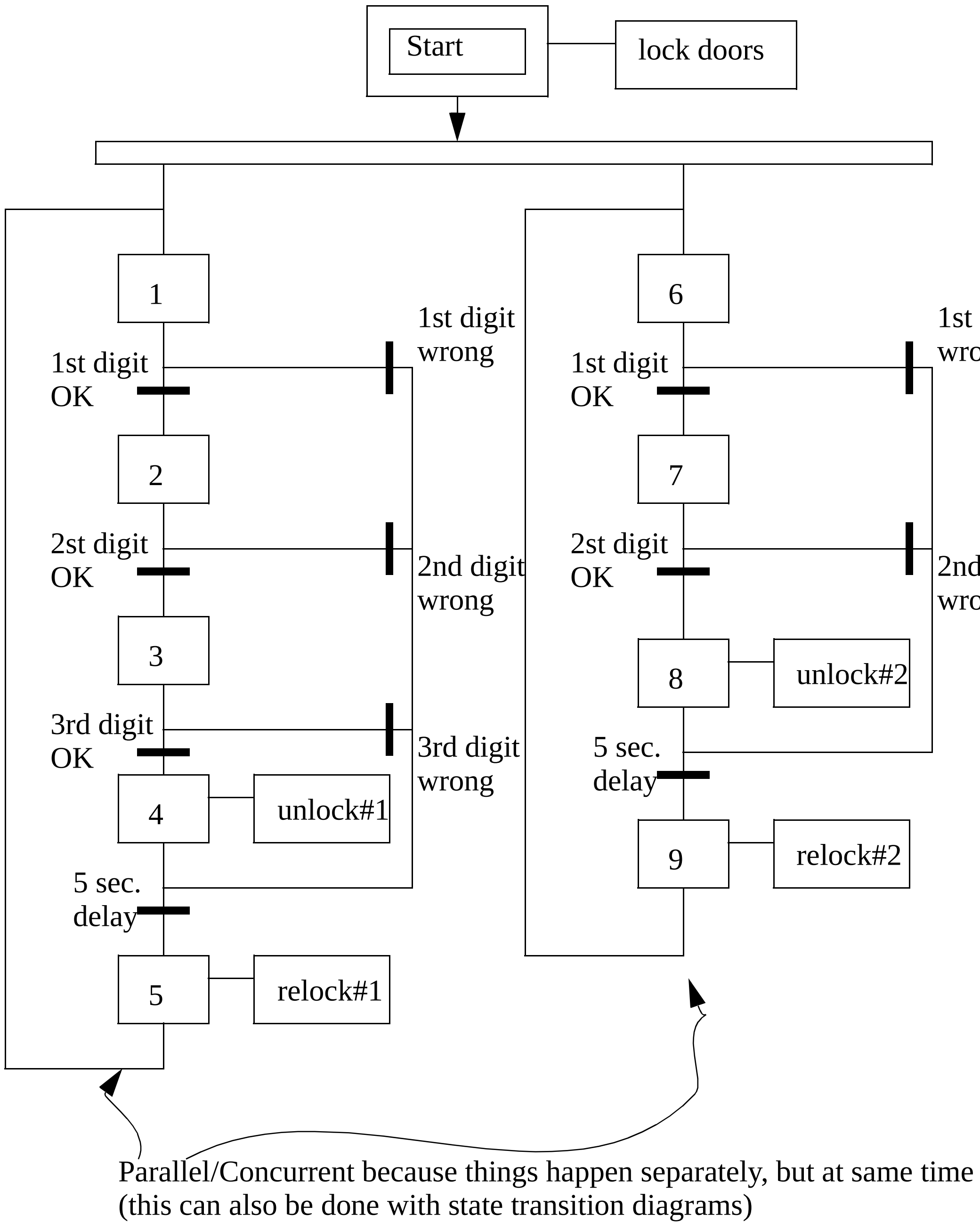


Figure 294 SFC for Control of Two Doors with Security Codes

A simple SFC for controlling a stamping press is shown in Figure 295. (Note: this controller only has a single thread of execution, so it could also be implemented with state diagrams, flowcharts, or other methods.) In the diagram the press starts in an idle state. when an *automatic* button is pushed the press will turn on the press power and lights. When a part is detected the press ram will advance down to the bottom limit switch. The press will then retract the ram until the top limit switch is contacted, and the ram will be stopped. A stop button can stop the press only when it is advancing. (Note: normal designs require that stops work all the time.) When the press is stopped a *reset* button must be pushed before the *automatic* button can be pushed again. After step 6 the press will wait until the part is not present before waiting for the next part. Without this logic the press would cycle continuously.

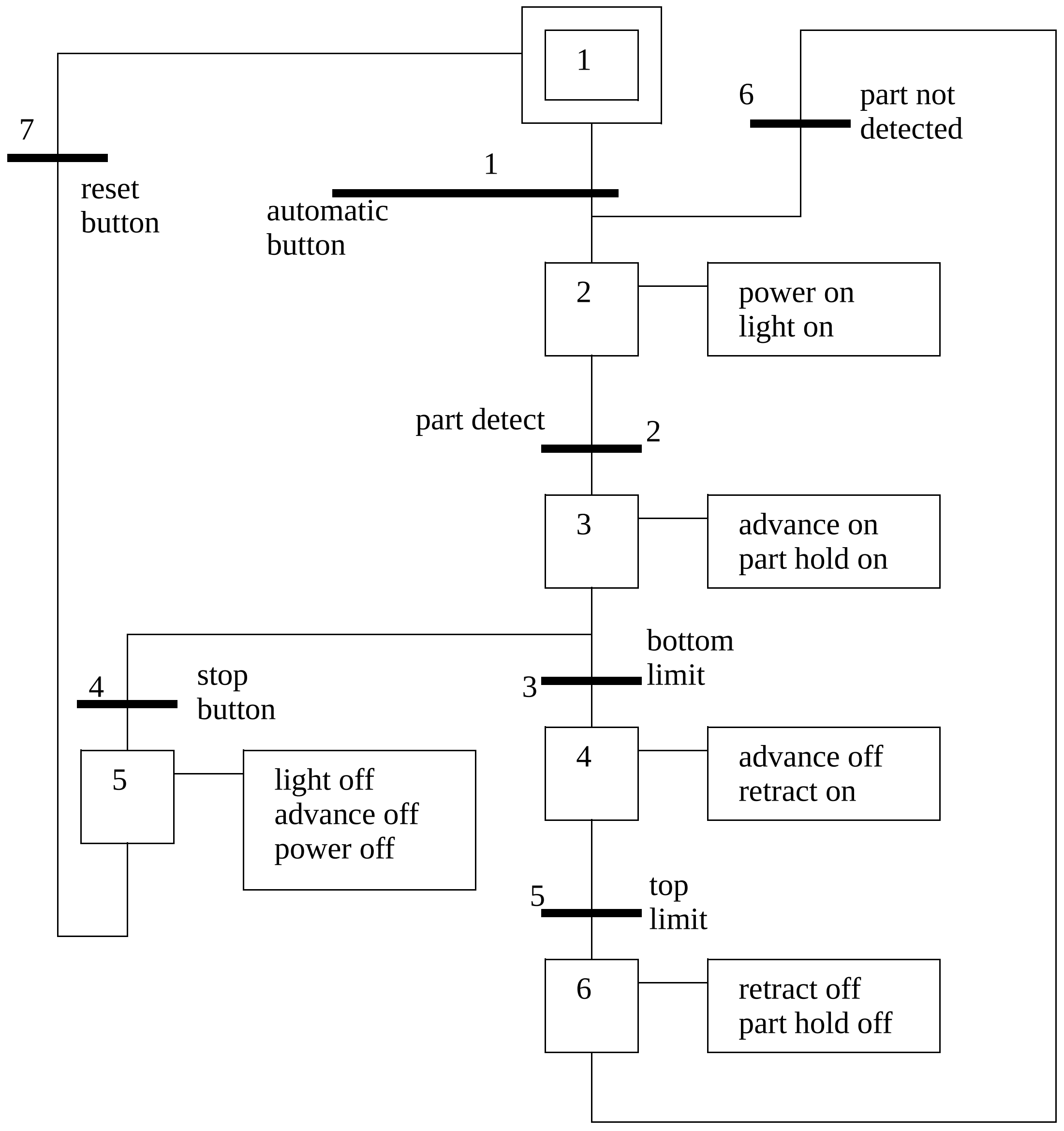


Figure 295 SFC for Controlling a Stamping Press

The SFC can be converted directly to ladder logic with methods very similar to those used for state diagrams as shown in Figure 296 to Figure 300. The method shown is patterned after the block logic method. One significant difference is that the transitions must now be considered separately. The ladder logic begins with a section to initialize the states and transitions to a single value. The next section of the ladder logic considers the transitions and then checks for transition conditions. If satisfied the following step or transition can be turned on, and the transition turned off. This is followed by ladder logic to turn on outputs as requires by the steps. This section of ladder logic corresponds to the actions for each step. After that the steps are considered, and the logic moves to the following transitions or steps. The sequence *examine transitions*, *do actions* then *do steps* is very important. If other sequences are used outputs may not be actuated, or steps missed entirely.

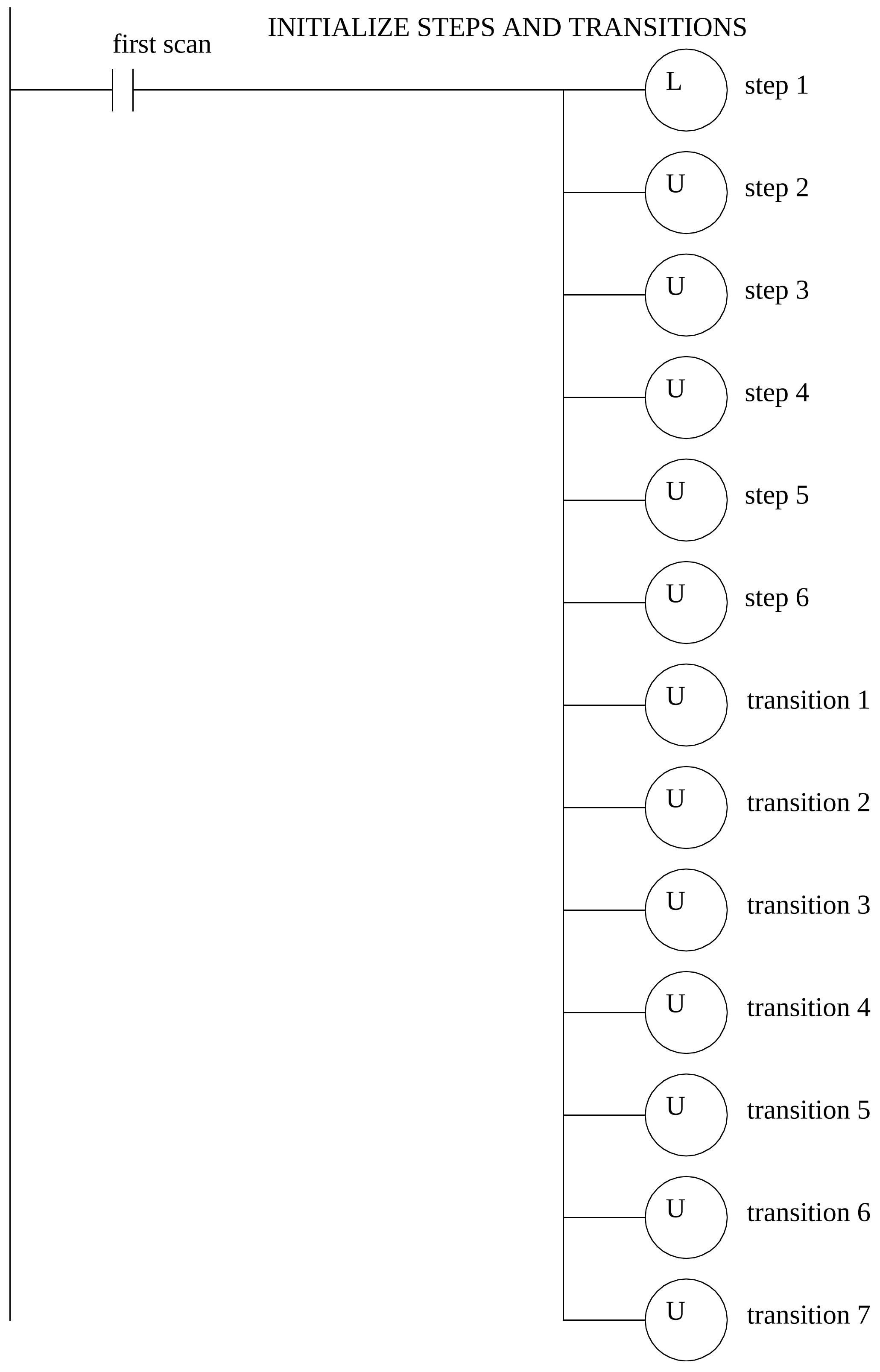


Figure 296 SFC Implemented in Ladder Logic

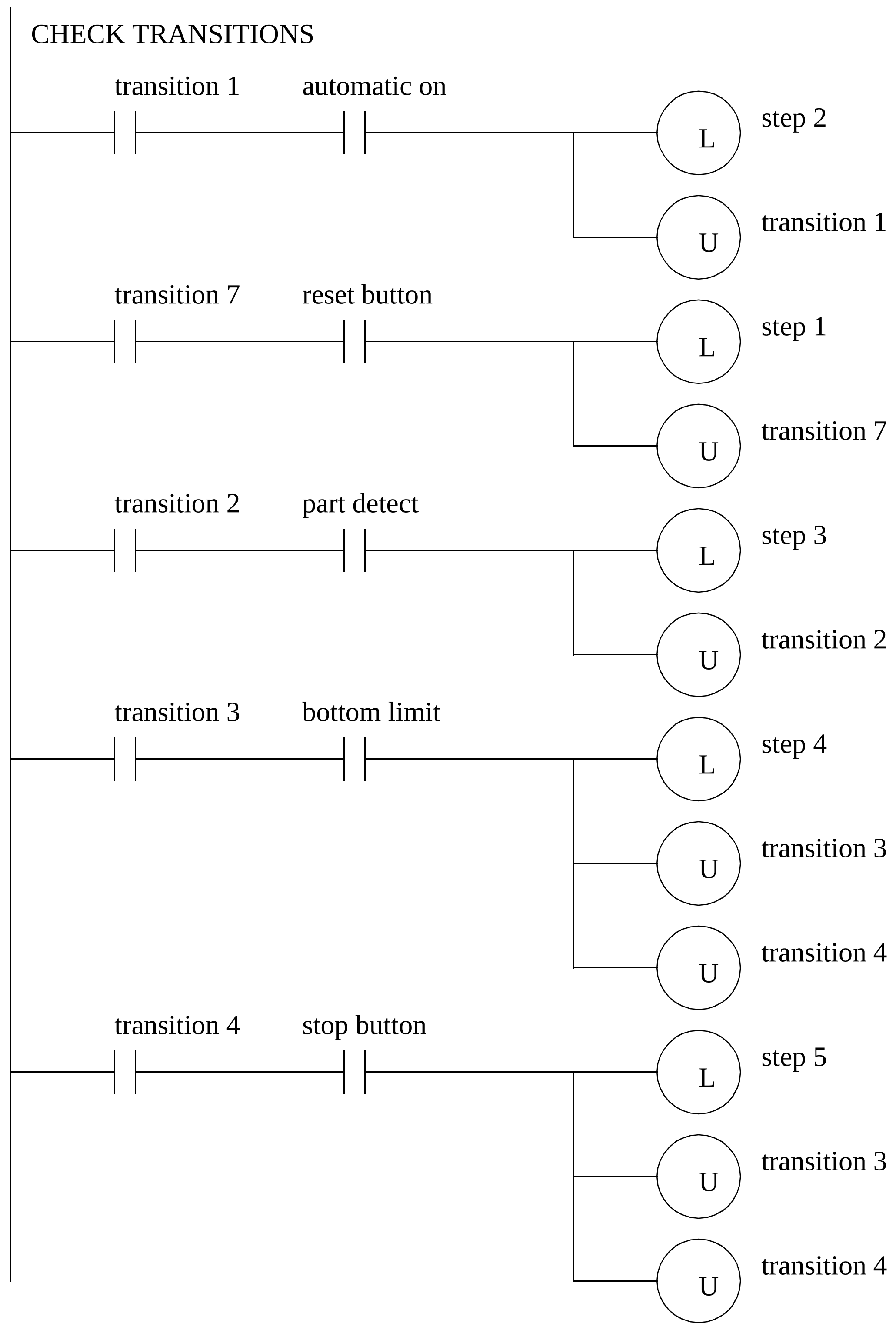


Figure 297 SFC Implemented in Ladder Logic

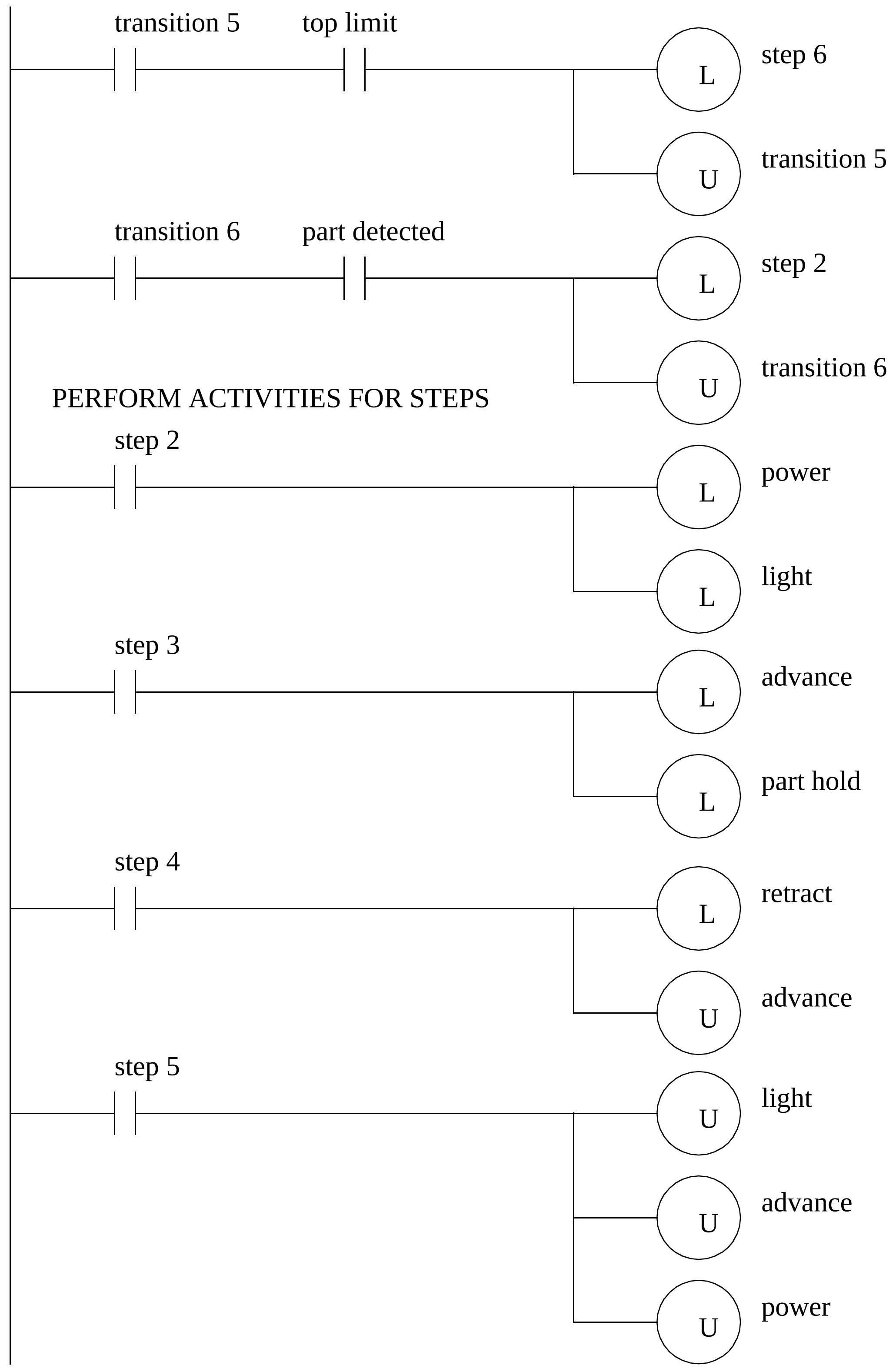


Figure 298 SFC Implemented in Ladder Logic

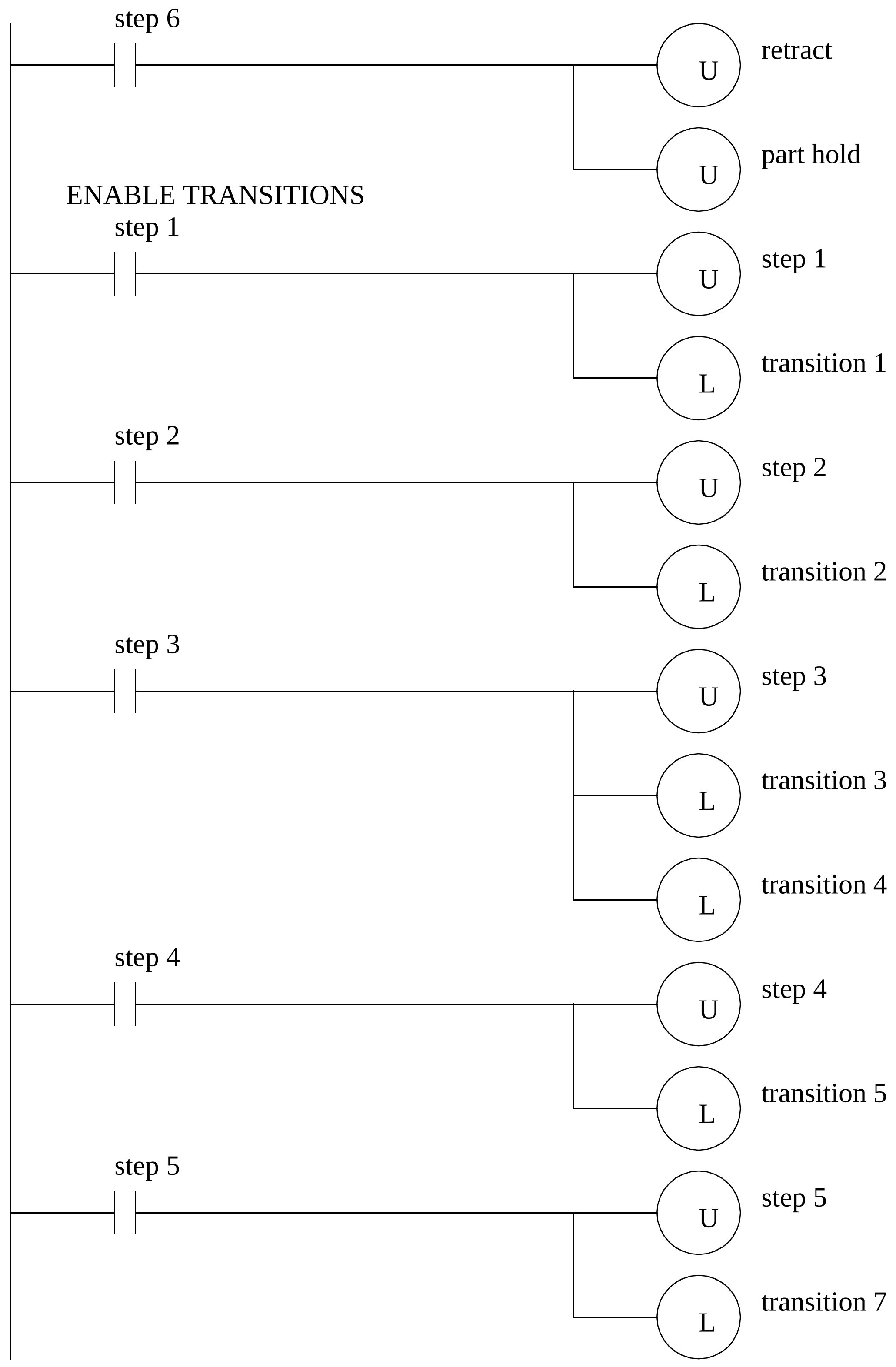


Figure 299 SFC Implemented in Ladder Logic

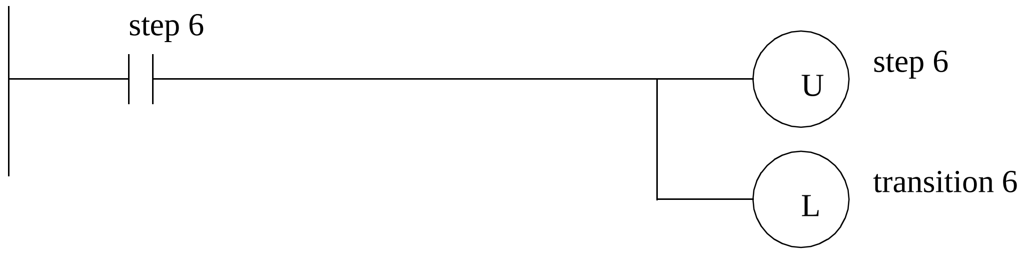


Figure 300 SFC Implemented in Ladder Logic

Many PLCs also allow SFCs to entered be as graphic diagrams. Small segments of ladder logic must then be entered for each transition and action. Each segment of ladder logic is kept in a separate program. If we consider the previous example the SFC diagram would be numbered as shown in Figure 301. The numbers are sequential and are for both transitions and steps.

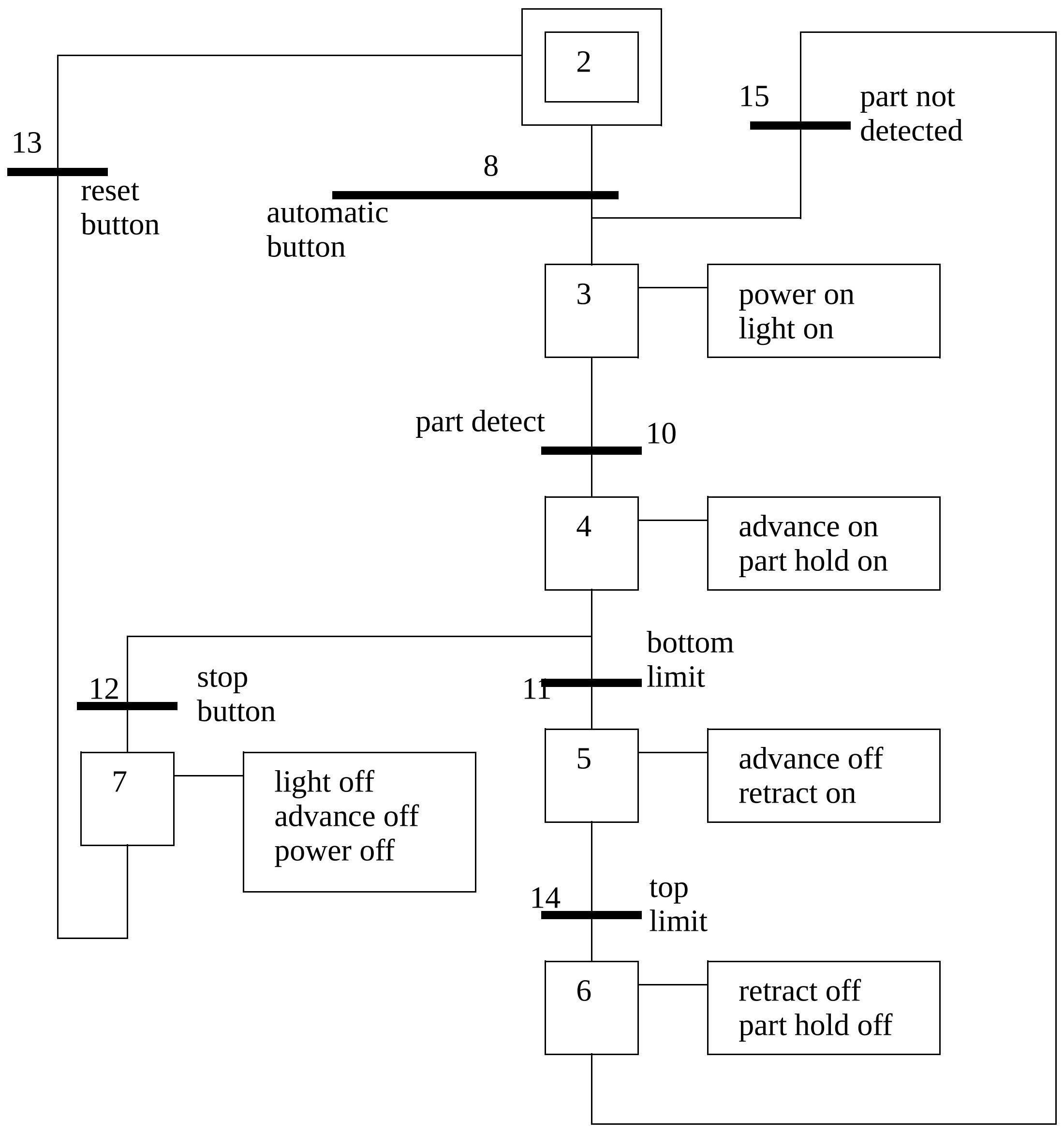


Figure 301 SFC Renumbered

Some of the ladder logic for the SFC is shown in Figure 302. Each program corresponds to the number on the diagram. The ladder logic includes a new instruction, EOT, that will tell the PLC when a transition has completed. When the rung of ladder logic with the EOT output becomes true the SFC will move to the next step or transition. when developing graphical SFCs the ladder logic becomes very simple, and the PLC deals with turning states on and off properly.

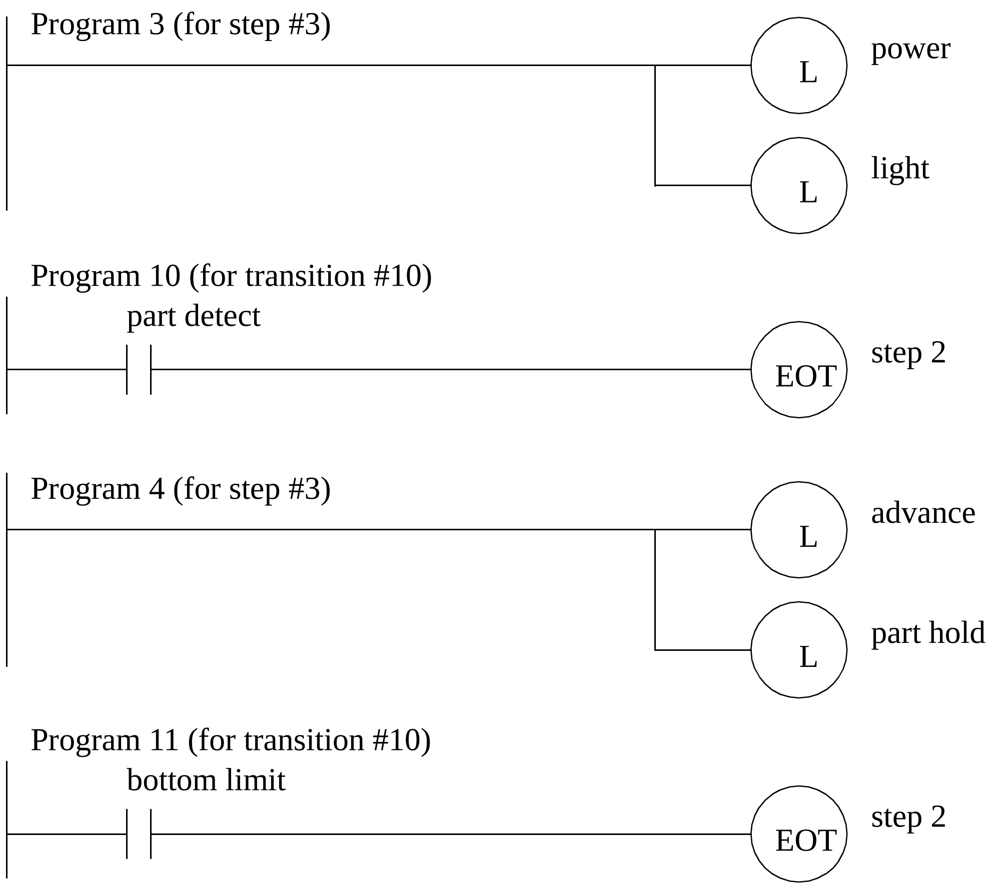


Figure 302 Sample Ladder Logic for a Graphical SFC Program

SFCs can also be implemented using ladder logic that is not based on latches, or built in SFC capabilities. The previous SFC example is implemented below. The first segment of ladder logic in Figure 303 is for the transitions. The logic for the steps is shown in Figure 304.

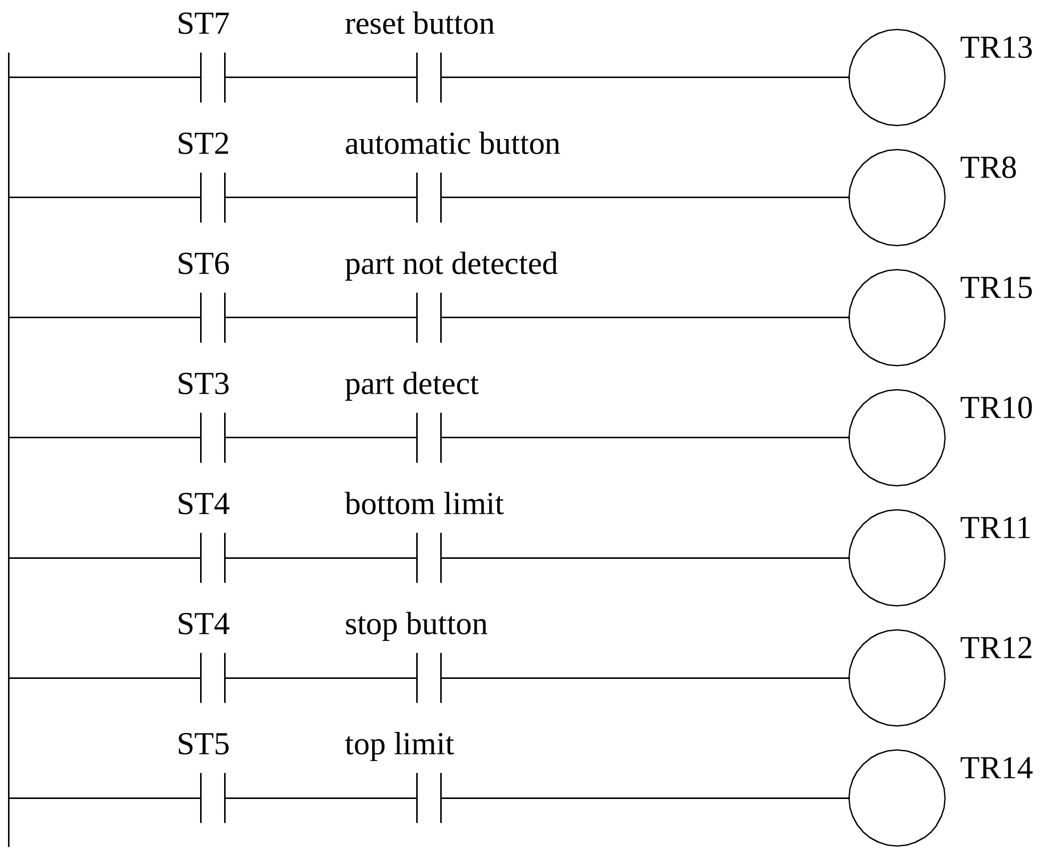


Figure 303 Ladder logic for transitions

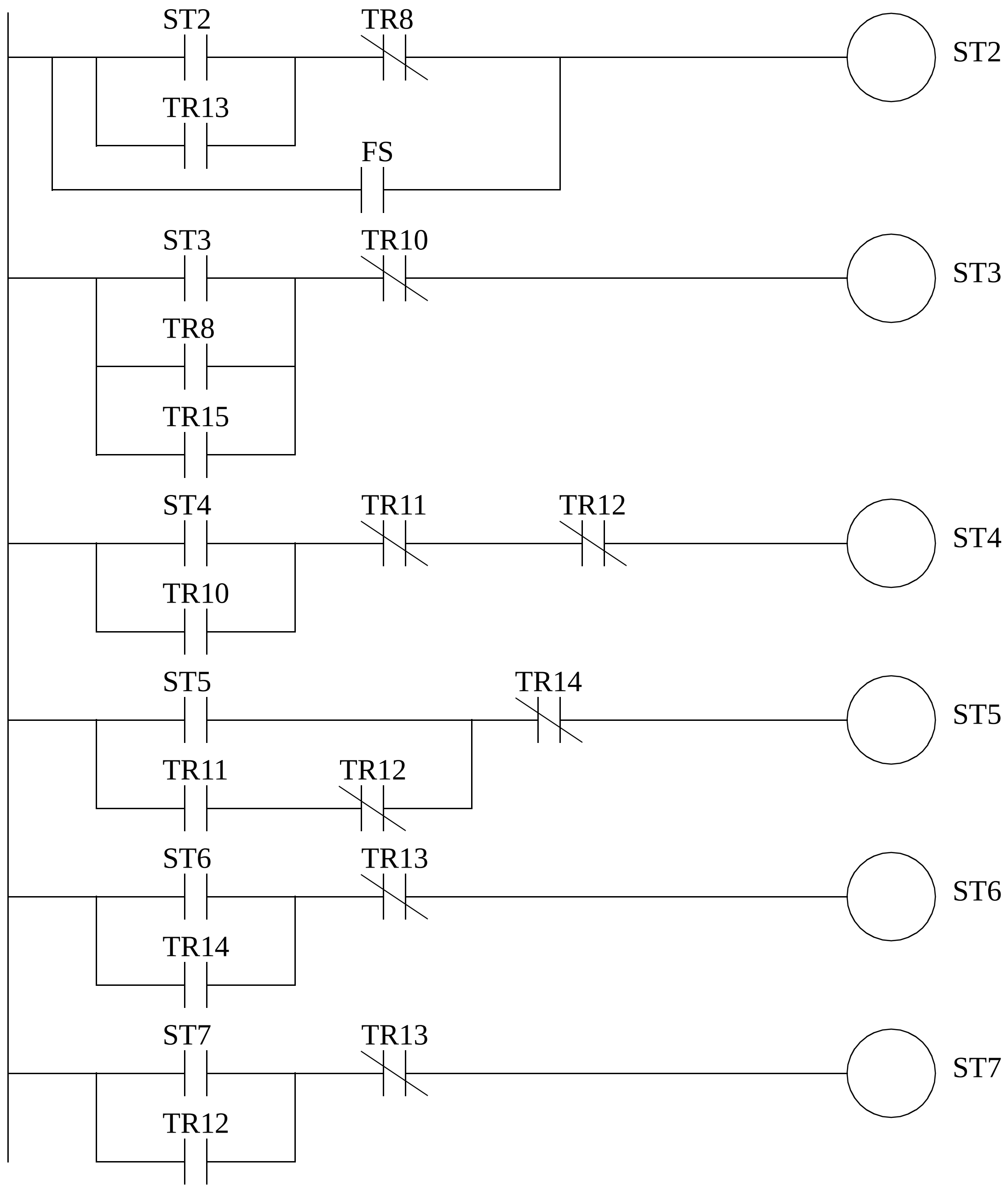


Figure 304 Step logic

Aside: The SFC approach can also be implemented with traditional programming languages. The example below shows the previous example implemented for a Basic Stamp II microcontroller.

```
autoon = 1; detect=2; bottom=3; top=4; stop=5;reset=6 ‘define input pins
input autoon; input detect; input button; input top; input stop; input reset
s1=1; s2=0; s3=0; s4=0; s5=0; s6=0 ‘set to initial step
advan=7;onlite=8; hold=9;retrac=10 ‘define outputs
output advan; output onlite; output hold; output retrac
step1: if s1<>1 then step2; s1=2
step2: if s2<>1 then step3; s2=2
step3: if s3<>1 then step4; s3=2
step4: if s4<>1 then step5; s4=2
step5: if s5<>1 then step6; s5=2
step6: if s6<>1 then trans1; s6=2
trans1: if (in1<>1 or s1<>2) then trans2;s1=0;s2=1
trans2: (if in2<>1 or s2<>2) then trans3;s2=0;s3=1
trans3: ..................
stepa1: if (st2<>1) then goto stepa2: high onlite
.................
goto step1
```

Figure 305 Implementing SFCs with High Level Languages

19.2 A COMPARISON OF METHODS

These methods are suited to different controller designs. The most basic controllers can be developed using process sequence bits and flowcharts. More complex control problems should be solved with state diagrams. If the controller needs to control concurrent processes the SFC methods could be used. It is also possible to mix methods together. For example, it is quite common to mix state based approaches with normal conditional logic. It is also possible to make a concurrent system using two or more state diagrams.

19.3 SUMMARY

- Sequential function charts are suited to processes with parallel operations
- Controller diagrams can be converted to ladder logic using MCR blocks
- The sequence of operations is important when converting SFCs to ladder logic.

19.4 PRACTICE PROBLEMS

1. Develop an SFC for a two person assembly station. The station has two presses that may be used at the same time. Each press has a cycle button that will start the advance of the press. A bottom limit switch will stop the advance, and the cylinder must then be retracted until a top limit switch is hit.

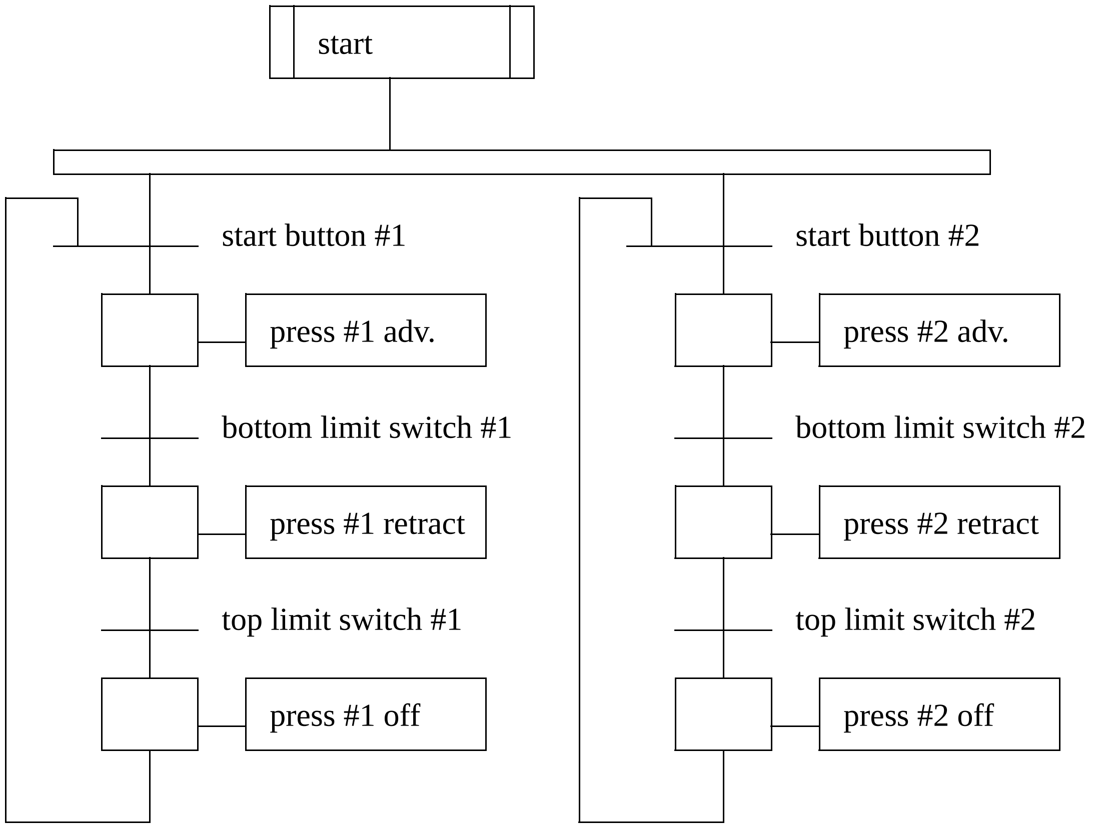
2. Create an SFC for traffic light control. The lights should have cross walk buttons for both directions of traffic lights. A normal light sequence for both directions will be green 16 seconds and yellow 4 seconds. If the cross walk button has been pushed, a walk light will be on for 10 seconds, and the green light will be extended to 24 seconds.

3. Draw an SFC for a stamping press that can advance and retract when a cycle button is pushed, and then stop until the button is pushed again.

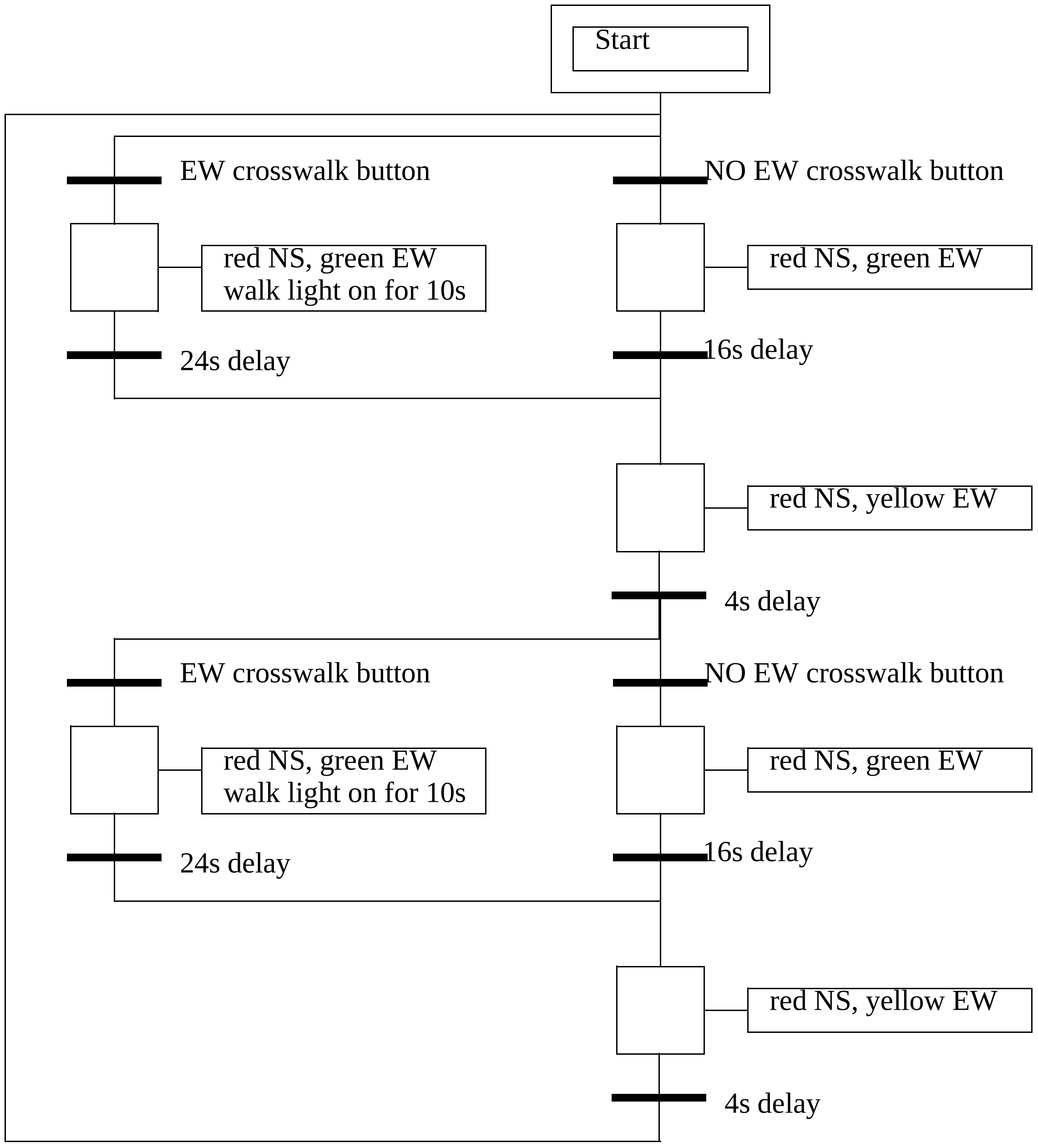
4. Design a garage door controller using an SFC. The behavior of the garage door controller is as follows,
    - there is a single button in the garage, and a single button remote control.
    - when the button is pushed the door will move up or down.
    - if the button is pushed once while moving, the door will stop, a second push will start motion again in the opposite direction.
    - there are top/bottom limit switches to stop the motion of the door.
    - there is a light beam across the bottom of the door. If the beam is cut while the door is closing the door will stop and reverse.
    - there is a garage light that will be on for 5 minutes after the door opens or closes.

19.5 PRACTICE PROBLEM SOLUTIONS

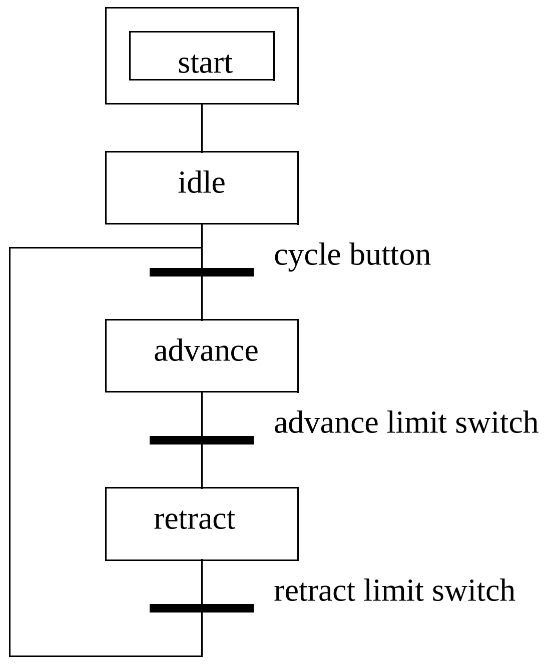
1.



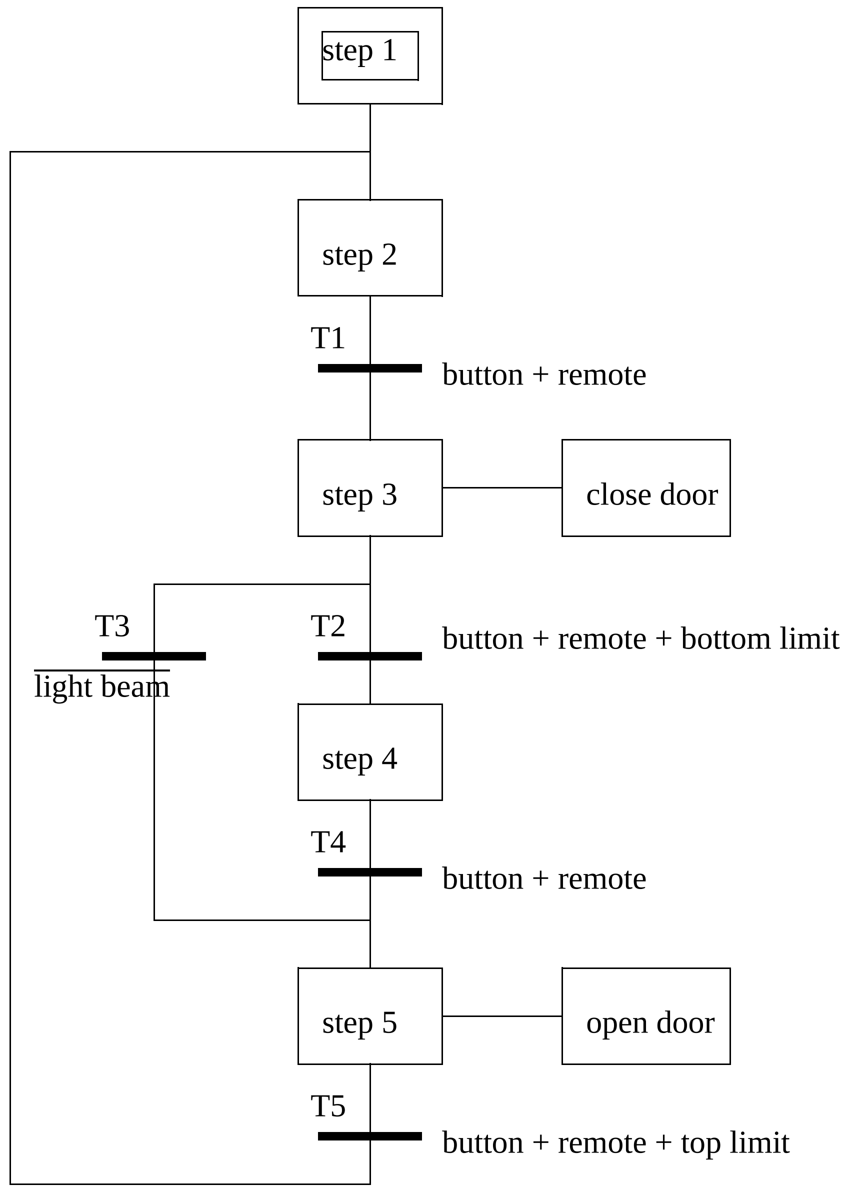
2.

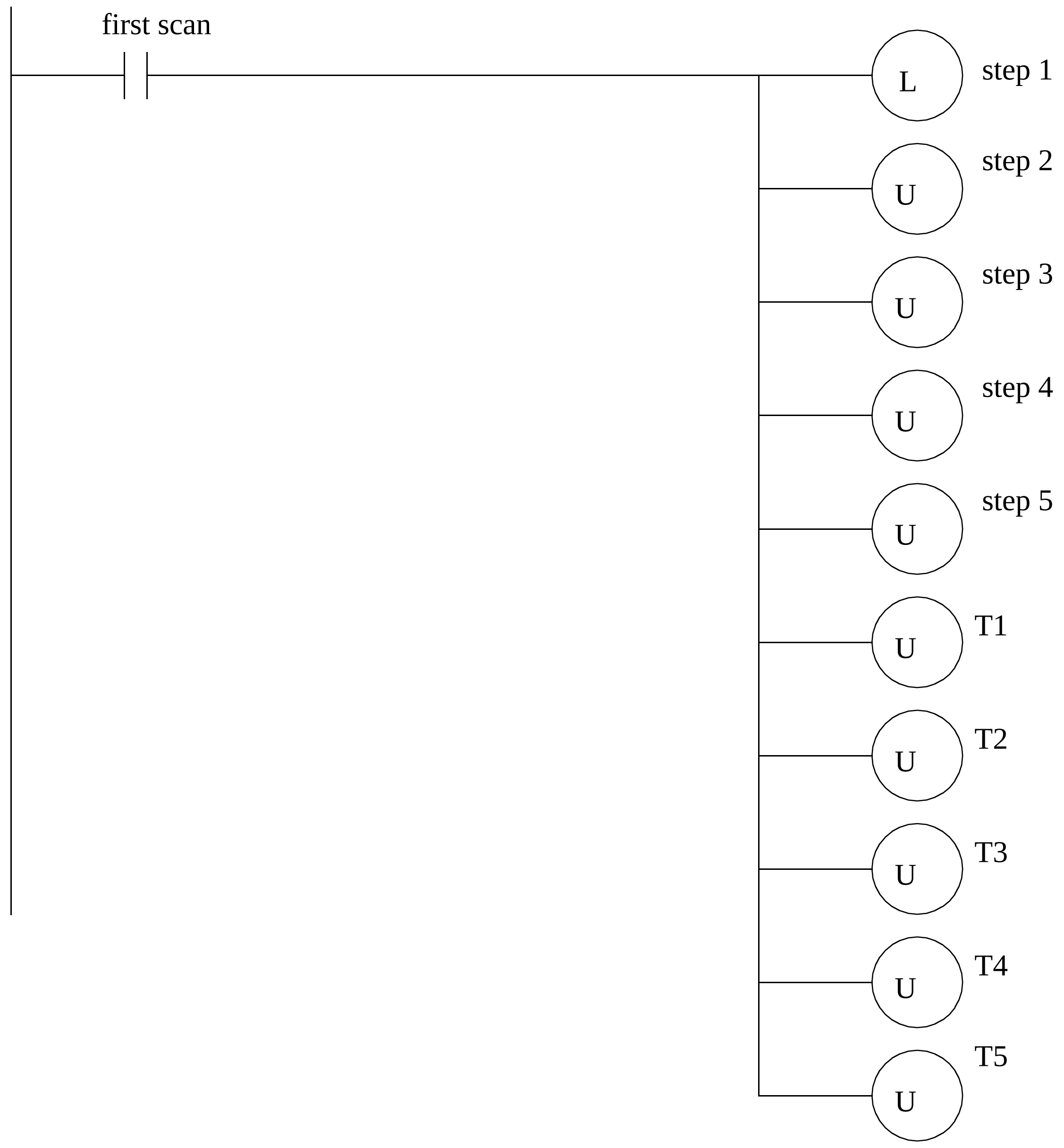

Start
EW crosswalk button
NO EW crosswalk button
red NS, green EW
walk light on for 10s
red NS, green EW
24s delay
16s delay
red NS, yellow EW
4s delay
EW crosswalk button
NO EW crosswalk button
red NS, green EW
walk light on for 10s
red NS, green EW
24s delay
16s delay
red NS, yellow EW
4s delay

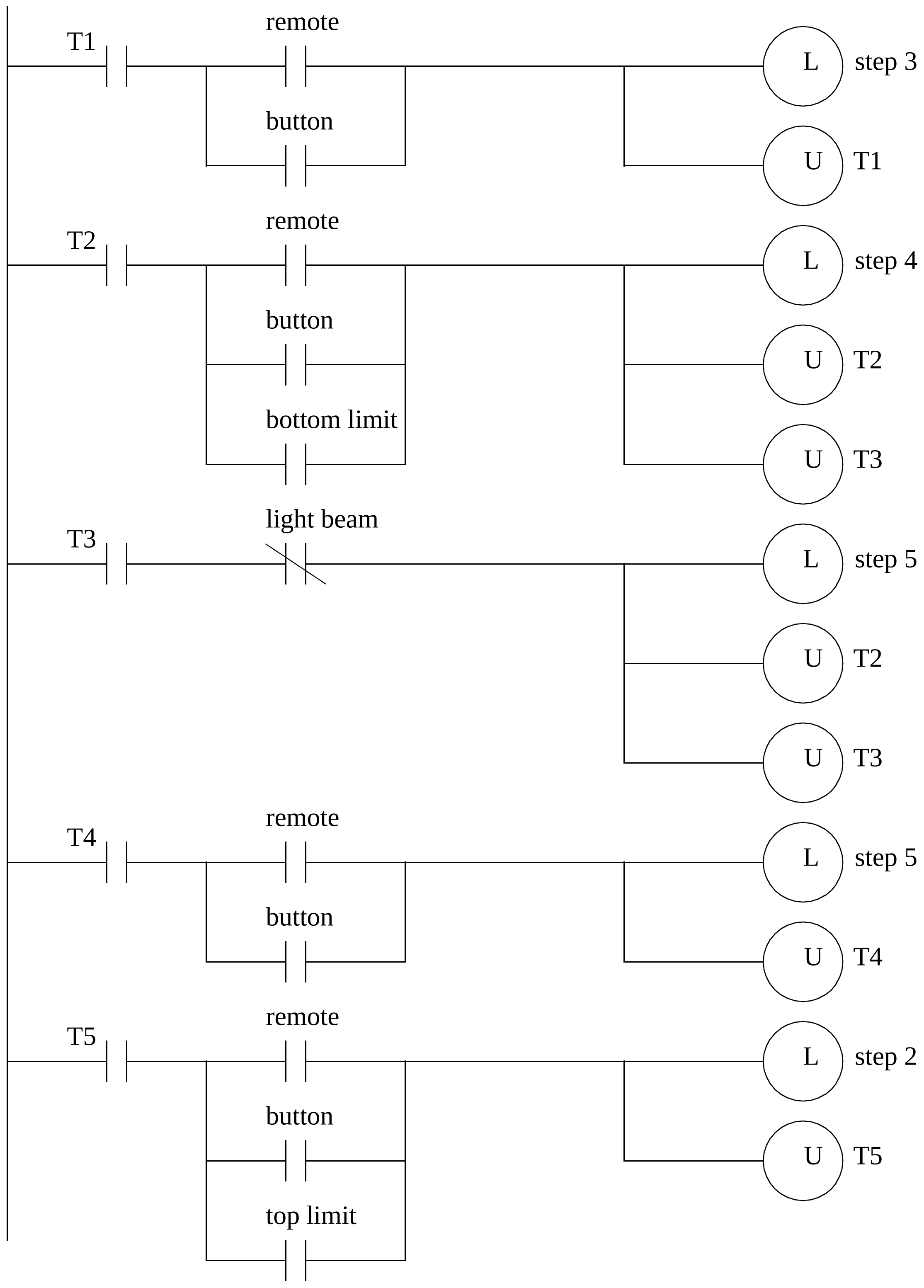
3.

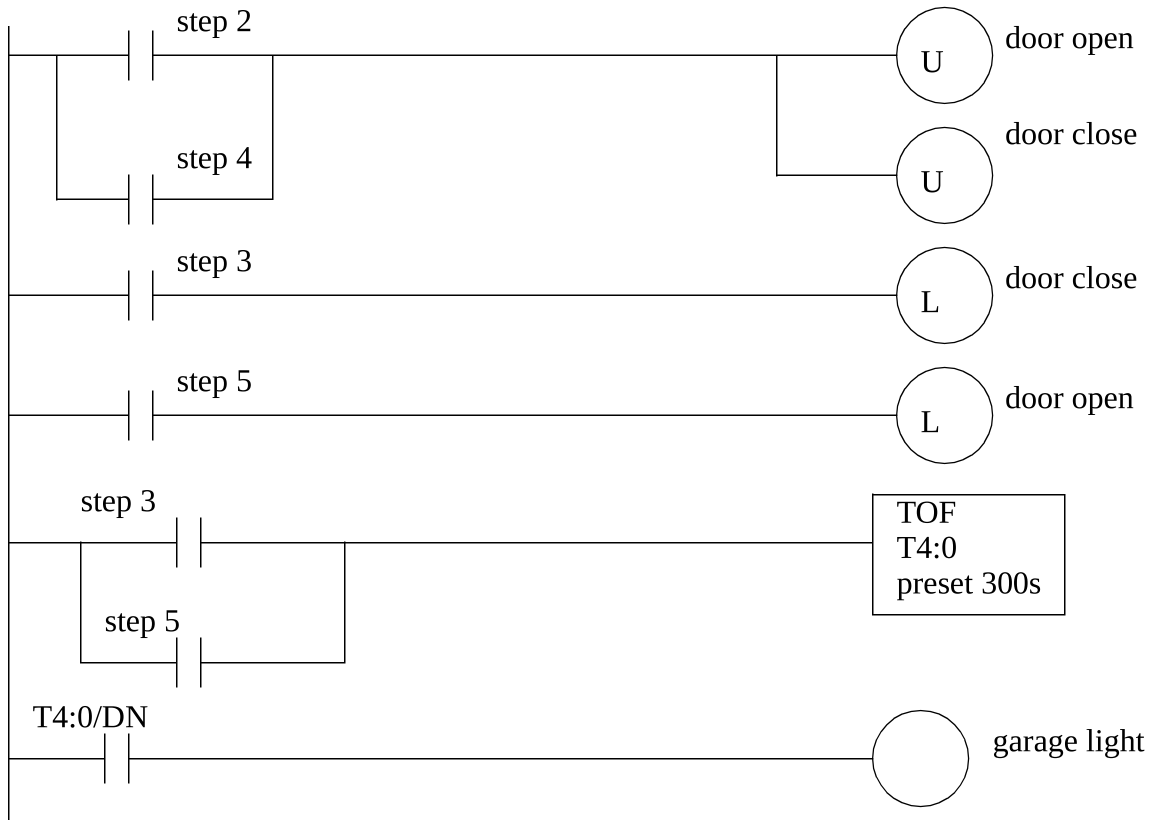

start
idle
cycle button
advance
advance limit switch
retract
retract limit switch

4.

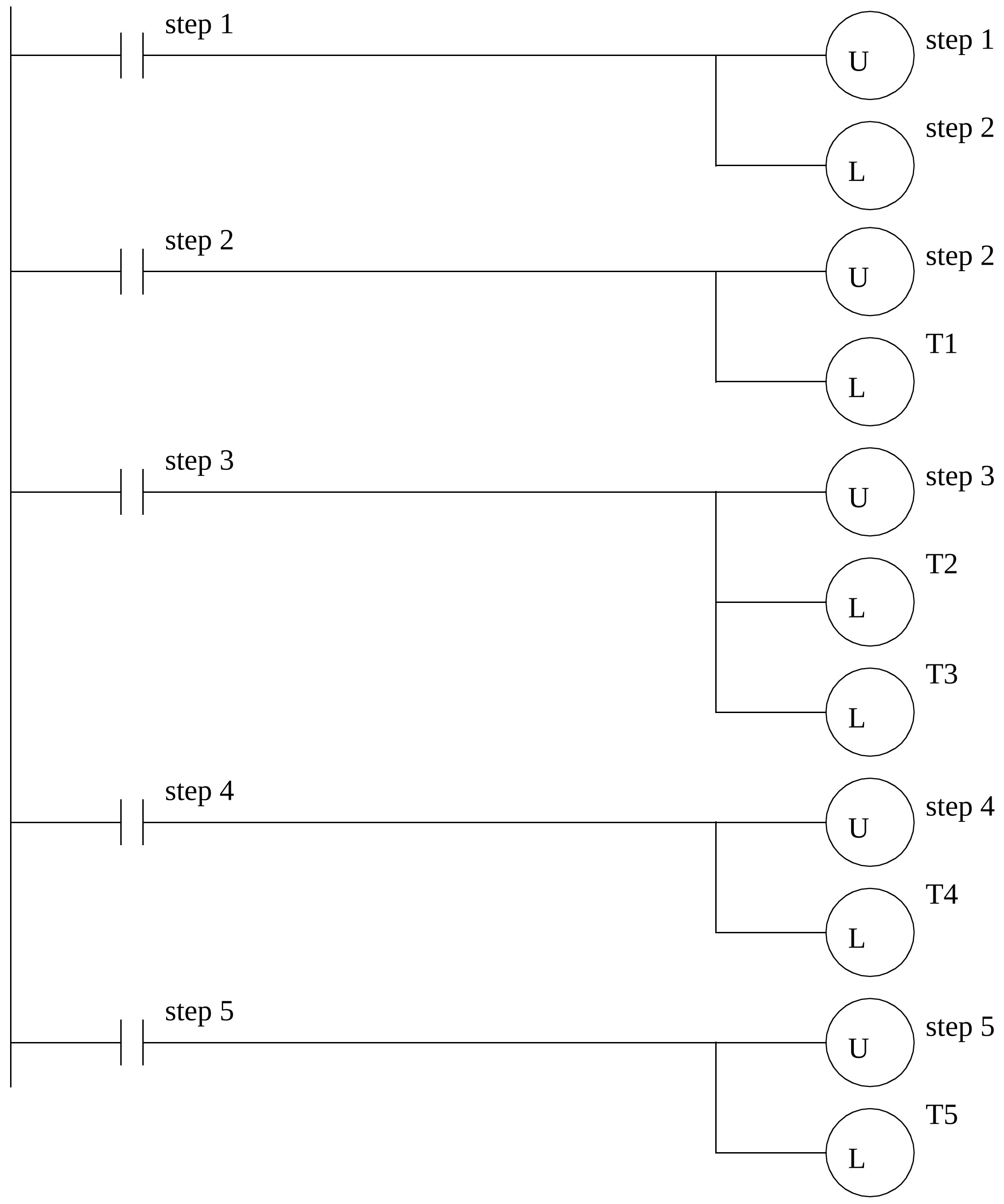

step 1
step 2
T1
button + remote
step 3
close door
T3
T2
button + remote + bottom limit
$\overline{\text{light beam}}$
step 4
T4
button + remote
step 5
open door
T5
button + remote + top limit


first scan
L
step 1
U
step 2
U
step 3
U
step 4
U
step 5
U
T1
U
T2
U
T3
U
T4
U
T5


T1
remote
button
L step 3
U T1
T2
remote
button
bottom limit
L step 4
U T2
U T3
T3
light beam
L step 5
U T2
U T3
T4
remote
button
L step 5
U T4
T5
remote
button
top limit
L step 2
U T5



step 2
step 4
U
door open
U
door close
step 3
L
door close
step 5
L
door open
step 3
step 5
TOF
T4:0
preset 300s
T4:0/DN
garage light



19.6 ASSIGNMENT PROBLEMS

1. Develop an SFC for a vending machine and expand it into ladder logic.