12. STATE BASED DESIGN

12.1 INTRODUCTION

A system state is a mode of operation. Consider a bank machine that will go through very carefully selected states. The general sequence of states might be idle, scan card, get secret number, select transaction type, ask for amount of cash, count cash, deliver cash/return card, then idle.

A State based system can be described with system states, and the transitions between those states. A state diagram is shown in Figure 12.1. The diagram has two states, State 1 and State 2. If the system is in state 1 and A happens the system will then go into state 2, otherwise it will remain in State 1. Likewise if the system is in state 2, and B happens the system will return to state 1. As shown in the figure this state diagram could be used for an automatic light controller. When the power is turned on the system will go into the lights off state. If motion is detected or an on push button is pushed the system will go to the lights on state. If the system is in the lights on state and 1 hour has passed, or an off pushbutton is pushed then the system will go to the lights off state. The else statements are omitted on the second diagram, but they are implied.
The most essential part of creating state diagrams is identifying states. Some key questions to ask are:
1. Consider the system,
   - What does the system do normally?
   - Does the system behavior change?
   - Can something change how the system behaves?
   - Is there a sequence to actions?
2. List modes of operation where the system is doing one identifiable activity that will start and stop. Keep in mind that some activities may just be to wait.

Consider the design of a coffee vending machine. The first step requires the identification of vending machine states as shown in Figure 12.2. The main state is the idle state. There is an inserting coins state where the total can be displayed. When enough coins have been inserted the user may select their drink of choice. After this the make coffee state will be entered.

This diagram could describe the operation of energy efficient lights in a room operated by two push buttons. State 1 might be lights off and state 2 might be lights on. The arrows between the states are called transitions and will be followed when the conditions are true. In this case if we were in state 1 and A occurred we would move to state 2. The else loop indicates that a state will stay active if a transition are is not followed. These are so obvious they are often omitted from state diagrams.
PLC states - 12.3

Be active while coffee is being brewed. If an error is detected the service needed state will be activated.

Figure 12.2

Definition of Vending Machine States

The states are then drawn in a state diagram as shown in Figure 12.3. Transitions are added as needed between the states. Here we can see that when powered up the machine will start in an idle state. The transitions here are based on the inputs and sensors in the vending machine. The state diagram is quite subjective, and complex diagrams will differ from design to design. These diagrams also expose the controller behavior. Consider that if the machine needs maintenance, and it is unplugged and plugged back in, the service needed statement would not be reentered until the next customer paid for but did not receive their coffee. In a commercial design we would want to fix this oversight.

STATES

idle - the machine has no coins and is doing nothing
inserting coins - coins have been entered and the total is displayed
user choose - enough money has been entered and the user is making coffee selection
make coffee - the selected type is being made
service needed - the machine is out of coffee, cups, or another error has occurred

NOTES:
1. These states can be subjective, and different designers might pick others.
2. The states are highly specific to the machine.
3. The previous/next states are not part of the states.
4. There is a clean difference between states.
12.1.1 State Diagram Example

Consider the traffic lights in Figure 12.4. The normal sequences for traffic lights are a green light in one direction for a long period of time, typically 10 or more seconds. This is followed by a brief yellow light, typically 4 seconds. This is then followed by a similar light pattern in the other direction. It is understood that a green or yellow light in one direction implies a red light in the other direction. Pedestrian buttons are provided so that when pedestrians are present a cross walk light can be turned on and the duration of the green light increased.
The first step for developing a controller is to define the inputs and outputs of the system as shown in Figure 12.5. First we will describe the system variables. These will vary as the system moves from state to state. Please note that some of these together can define a state (alone they are not the states). The inputs are used when defining the transitions. The outputs can be used to define the system state.

A simple diagram can be drawn to show sequences for the lights. Note that each state will lead to a different set of outputs. The inputs are often part, or all of the transitions.
Previously state diagrams were used to define the system, it is possible to use a state table as shown in Figure 12.6. Here the light sequences are listed in order. Each state is given a name to ease interpretation, but the corresponding output pattern is also given. The system state is defined as the bit pattern of the 6 lights. Note that there are only 4 patterns, but 6 binary bits could give as many as 64.

Figure 12.6
System State Table for Traffic Lights

Transitions can be added to the state table to clarify the operation, as shown in Figure 12.7. Here the transition from Green E/W to Yellow E/W is S1. What this means is that a cross walk button must be pushed to end the green light. This is not normal, normally the lights would use a delay. The transition from Yellow E/W to Green N/S is caused by a 4 second delay (this is normal.) The next transition is also abnormal, requiring that the cross walk button be pushed to end the Green N/S state. The last state has a 4 second delay before returning to the first state in the table. In this state table the sequence will always be the same, but the times will vary for the green lights.

Step 1: Define the System States and put them (roughly) in sequence

<table>
<thead>
<tr>
<th>L1</th>
<th>L2</th>
<th>L3</th>
<th>L4</th>
<th>L5</th>
<th>L6</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

System State

<table>
<thead>
<tr>
<th>#</th>
<th>System Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Green North/South</td>
</tr>
<tr>
<td>2</td>
<td>Yellow North/South</td>
</tr>
<tr>
<td>3</td>
<td>Green East/West</td>
</tr>
<tr>
<td>4</td>
<td>Yellow East/West</td>
</tr>
</tbody>
</table>

Here the four states determine how the 6 outputs are switched on/off.
A state diagram for the system is shown in Figure 12.8. This diagram is equivalent to the state table in Figure 12.7, but it can be valuable for doing visual inspection.

12.1.2 Conversion to Ladder Logic

12.1.2.1 - Block Logic Conversion

Step 2: Define State Transition Triggers, and add them to the list of states

<table>
<thead>
<tr>
<th>Transition</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
</tr>
<tr>
<td>Delay 4 sec</td>
</tr>
<tr>
<td>S2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Green North/South</td>
</tr>
<tr>
<td>Yellow North/South</td>
</tr>
<tr>
<td>Green East/West</td>
</tr>
<tr>
<td>Yellow East/West</td>
</tr>
</tbody>
</table>

Step 3: Draw the State Transition Diagram

1. First scan
State diagrams can be converted directly to ladder logic using block logic. This technique will produce larger programs, but it is a simple method to understand, and easy to debug. The previous traffic light example is to be implemented in ladder logic. The inputs and outputs are defined in Figure 12.9, assuming it will be implemented on an Allen Bradley Micrologix.

The first scan indicator will execute the MCR block when the PLC is first turned on, and the latches will turn on the value for state 1 and turn off the others.

**STATE S**
- B3/1 - state 1 - green E/W
- B3/2 - state 2 - yellow E/W
- B3/3 - state 3 - green N/S
- B3/4 - state 4 - yellow N/S

**OUTPUTS**
- O/1 - L1
- O/2 - L2
- O/3 - L3
- O/4 - L4
- O/5 - L5
- O/6 - L6

**INPUTS**
- I/1 - S1
- I/2 - S2
- S2:1/14 - first scan
The next section of ladder logic only deals with outputs. For example the output O/1 is the N/S red light, which will be on for states 1 and 2, or B3/1 and B3/2 respectively.

Putting normal outputs outside the MCR blocks is important. If they were inside the
S2: 1/14
MCR
B3/1
B3/2
B3/3
B3/4
MCR
L
U
U
RESET THE STATES

A
MCR
MCR
If A is true then the MCR will cause the ladder in between to be executed. If A is false the outputs are forced off.

Note: We will use MCR instructions to implement some of the state based programs. This allows us to switch off part of the ladder logic. The one significant note to remember is that any normal outputs (not latches and timers) will be FORCED OFF. Unless this is what you want, put the normal outputs outside MCR blocks.
Block states - 12.10

Often it is possible for the PLC to assume block states they could only be on when the MCR block was active, otherwise they would be forced off. Note: Many beginners will make the careless mistake of repeating outputs in this section of the program.

Figure 12.11
General Output Control Logic

The first state is implemented in Figure 12.10. If state 1 is active this will be active. The transition is $S1$ or $I/1$ which will end state 1 and start state 2.

TURN ON LIGHTS AS REQUIRED

$B3/1$
$B3/2$
$B3/3$
$B3/4$

$O/1$
$O/2$
$O/3$
$O/4$

$O/5$
$O/6$
The second state is more complex because it involves a time delay, as shown in Figure 12.13. When the state is active the RTO timer will be timing. When the timer is done state 2 will be unlatched, and state 3 will be latched on. The timer is retentive, so it must also be reset when the state is done, so that it will start at zero the next time the state starts.
The third and fourth states are shown in Figure 12.14 and Figure 12.15. Their layout is very similar to that of the first two states.
Figure 12.14 Ladder Logic for State Three

<table>
<thead>
<tr>
<th>Logic Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>B3/3</td>
<td>I/2</td>
</tr>
<tr>
<td>MCR</td>
<td></td>
</tr>
</tbody>
</table>

Third State: Wait for Transitions

Figure 12.15 Ladder Logic for State Four

<table>
<thead>
<tr>
<th>Logic Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>B3/4</td>
<td>T4:2/DN</td>
</tr>
<tr>
<td>MCR</td>
<td></td>
</tr>
</tbody>
</table>

Fourth State: Wait for Transitions

<table>
<thead>
<tr>
<th>Logic Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>B3/4</td>
<td>T4:2/DN</td>
</tr>
<tr>
<td>T4:2</td>
<td>RTO, delay 4s</td>
</tr>
<tr>
<td>B3/1</td>
<td></td>
</tr>
</tbody>
</table>

T4:2 Reset (RST)
The previous example only had one path through the state tables, so there was never a choice between states. The state diagram in Figure 12.16 could potentially have problems if two transitions occur simultaneously. For example, if state STB is active and A and C occur simultaneously, the system could go to either STA or STC (or both in a poorly written program.) To resolve this problem we should choose one of the two transitions as having a higher priority, meaning that it should be chosen over the other transition. This decision will normally be clear, but if not an arbitrary decision is still needed.

Figure 12.16
A State Diagram with Priority Problems

The state diagram in Figure 12.16 is implemented with ladder logic in Figure 12.17 and Figure 12.18. The implementation is the same as described before, but for state STB additional ladder logic is added to disable transition A if transition C is active, therefore giving priority to C.
Note: if A and C are true at the same time then C will have priority. Prioritization is important when simultaneous branches are possible.
The Block Logic technique described does not require any special knowledge and the programs can be written directly from the state diagram. The final programs can be easily modified, and finding problems is easier. But, these programs are much larger and less efficient.

12.1.2.2 - State Equations

State diagrams can be converted to Boolean equations and then to Ladder Logic. The first technique that will be described is state equations. These equations contain three main parts, as shown below in Figure 12.19. To describe them simply - a state will be on if it is already on, or if it has been turned on by a transition from another state, but it will be turned off if there was a transition to another state. An equation is required for each state in the state diagram.
The state equation method can be applied to the traffic light example in Figure 12.8. The first step in the process is to define variable names (or PLC memory locations) to keep track of which states are on or off. Next, the state diagram is examined, one state at a time. The first equation is for ST1, or state 1 - green NS.

The start of the equation can be read as ST1 will be on if it is on, or if ST4 is on, and it has been on for 4s, or if it is the first scan of the PLC. The end of the equation can be read as ST1 will be turned off if it is on, but S1 has been pushed and S2 is off. As discussed before, the first half of the equation will turn the state on, but the second half will turn it off. The first scan is also used to turn on ST1 when the PLC starts. It is put outside the terms to force ST1 on, even if the exit conditions are true.

Informally, State X = (State X + just arrived from another state) and has not left for another state. Form all y, where:

- State i: A variable that will reflect if state i is on
- n: the number of transitions to state i
- m: the number of transitions out of state i
- T ji: The logical condition of a transition from state j to i
- T ik: The logical condition of a transition out of state i to k
The equations in Figure 12.20 cannot be implemented in ladder logic because of the NOT over the last terms. The equations are simplified in Figure 12.21 so that all NOT operators are only over a single variable.

Defined state variables:

The state entrance and exit condition equations:

Note: Timers are represented in these equations in the form $\text{TON}_i(A, \text{delay})$. $\text{TON}_i$ indicates that it is an on-delay timer, $A$ is the input to the timer, and $\text{delay}$ is the timer delay value. The subscript $i$ is used to differentiate timers.
These equations are then converted to the ladder logic shown in Figure 12.22 and Figure 12.23. At the top of the program the two timers are defined. (Note: it is tempting to combine the timers, but it is better to keep them separate.) Next, the Boolean state equations are implemented in ladder logic. After this we use the states to turn specific lights on.

Now, simplify these for implementation in ladder logic.

\[
ST_1 \quad ST_1 \quad ST_{44} \quad (\quad ) \cdot + (\quad ) \cdot S_1 \quad S_2 ++ \cdot \quad + (\quad ) \cdot TON_1 \quad ST_{24}, () \cdot + (\quad ) \cdot = ST_3 \quad ST_3 \quad ST_2 \quad TON_1 \quad ST_{24}, () \cdot + (\quad ) \cdot S_1 \quad S_2 ++ \cdot \quad + (\quad ) \cdot TON_2 \quad ST_{44}, () \cdot + (\quad ) \cdot =
\]
Figure 12.22

Ladder Logic for the State Equations

- **First Scan**: Timer on T4:2 delay 4 sec
  - ST 4
  - ST 1
  - ST 4 X
  - ST 1 X

- **Second Scan**: Timer on T4:1 delay 4 sec
  - ST 2
  - ST 3
  - ST 2 X
  - ST 3 X

- **State Equations**
  - ST 1
  - S1
  - S2
  - ST 2
  - T4:1/DN
  - ST 3
  - S1
  - S2
  - ST 4
  - T4:2/DN

Define the Timers
This method will provide the most compact code of all techniques, but there are potential problems. Consider the example in Figure 12.23. If push button S1 has been pushed the line for ST1 should turn off, and the line for ST2 should turn on. But, the line for ST2 depends upon the value for ST1 that has just been turned off. This will cause a problem if the value of ST1 goes off immediately after the line of ladder logic has been scanned. In effect the PLC will get lost and none of the states will be on. This problem arises because the equations are normally calculated in parallel, and then all values are updated simultaneously. To overcome this problem the ladder logic could be modified to the form shown in Figure 12.24. Here some temporary variables are used to hold the new state values. After all the equations are solved the states are updated to their new values.
Figure 12.24
Delayed State Updating

When multiple transitions out of a state exist we must take care to add priorities.

THE STATE EQUATIONS

ST 1

ST 1X

ST 2

ST 2X

ST 3

ST 3X

ST 4

ST 4X
Each of the alternate transitions out of a state should be given a priority, from highest to lowest. The state equations can then be written to suppress transitions of lower priority when one or more occur simultaneously. The state diagram in Figure 12.25 has two transitions $A$ and $C$ that could occur simultaneously. The equations have been written to give $A$ a higher priority. When $A$ occurs, it will block $C$ in the equation for $STC$. These equations have been converted to ladder logic in Figure 12.26.
A state diagram may be converted to equations by writing an equation for each state and each transition. A sample set of equations is seen in Figure 12.27 for the traffic light example of Figure 12.8. Each state and transition needs to be assigned a unique variable name. (Note: It is a good idea to note these on the diagram) These are then used to write the equations for the diagram. The transition equations are written by looking at the each state, and then determining which transitions will end that state. For example, if ST1 is true, and crosswalk button \( S_1 \) is pushed, and \( S_2 \) is not, then transition \( T_1 \) will be true.

The state equations are similar to the state equations in the previous State Equation method, except they now only refer to the transitions. Recall, the basic form of these equations is that the state will be on if it is already on, or it has been turned on by a transition. The state will be turned off if an exiting transition occurs. In this example the first scan was given it's own transition, but it could have also been put into the equation for \( T_4 \).

**Figure 12.27**

**State-Transition Equations**

These equations can be converted directly to the ladder logic in Figure 12.28, Figure 12.29 and Figure 12.30. It is very important that the transition equations all occur before the state equations. By updating the transition equations first and then updating the state equations the problem of state variable values changing is negated - recall this problem was discussed in the State Equations section.

<table>
<thead>
<tr>
<th>State Variable</th>
<th>Transition Equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>ST1</td>
<td>( T_{ST1} )</td>
</tr>
<tr>
<td>ST2</td>
<td>( T_{ST2} )</td>
</tr>
<tr>
<td>ST3</td>
<td>( T_{ST3} )</td>
</tr>
<tr>
<td>ST4</td>
<td>( T_{ST4} )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Transition</th>
<th>Equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>( ST_{T1} = S_1 \land \neg S_2 )</td>
</tr>
<tr>
<td>T2</td>
<td>( ST_{T2} = S_1 \land \neg S_2 )</td>
</tr>
<tr>
<td>T3</td>
<td>( ST_{T3} = S_1 \land \neg S_2 )</td>
</tr>
<tr>
<td>T4</td>
<td>( ST_{T4} = S_1 \land \neg S_2 )</td>
</tr>
</tbody>
</table>

**Defined state and transition variables:**

- \( ST_1 \): state 1 - green NS=
- \( ST_2 \): state 2 - yellow NS=
- \( ST_3 \): state 3 - green EW=
- \( ST_4 \): state 4 - yellow EW=
- \( T_{ST1} \): transition from \( ST_1 \) to \( ST_2 \)
- \( T_{ST2} \): transition from \( ST_2 \) to \( ST_3 \)
- \( T_{ST3} \): transition from \( ST_3 \) to \( ST_4 \)
- \( T_{ST4} \): transition from \( ST_4 \) to \( ST_1 \)
- \( T_{ST5} \): transition to \( ST_1 \) for first scan

\( T_{ST5} = \)
Figure 12.28

Ladder Logic for the State-Transition Equations

```
timer on T4:2 delay 4 sec
ST 4
ST 4
T4
ST 1 S1 S2 T1
timer on T4:1 delay 4 sec
ST 2
ST 2 T4:1/DN T2
CALCULATE TRANSITION EQUATIONS
T4:2/DN
ST 3 S1 S2 T3
FS T5
UPDATE TIMERS
```
Figure 12.29

Ladder Logic for the State-Transition Equations

ST 1
T4 → ST1

ST 2
T1 → ST2

ST 3
T2 → ST3

ST 4
T3 → ST4

T5
CAL CUL ATE  STATE  EQUATIONS
The problem of prioritization also occurs with the State-Transition equations. Equations were written for the State Diagram in Figure 12.31. The problem will occur if transitions $A$ and $C$ occur simultaneously. In the example transition $T_2$ is given a higher priority, and if it is true, then the transition $T_3$ will be suppressed when calculating $STC$. In this example the transitions have been considered in the state update equations, but they can also be used in the transition equations.

<table>
<thead>
<tr>
<th>ST 1</th>
<th>ST 2</th>
<th>ST 3</th>
<th>ST 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L_1$</td>
<td>$L_2$</td>
<td>$L_3$</td>
<td>$L_4$</td>
</tr>
<tr>
<td>$L_5$</td>
<td>$L_6$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

UPDATE OUTPUTS
12.2 SUMMARY
State diagrams are suited to processes with a single flow of execution. They are suited to problems that have clearly defined modes of execution. Controller diagrams can be converted to ladder logic using MCR blocks. State diagrams can also be converted to ladder logic using equations. The sequence of operations is important when converting state diagrams to ladder logic.

12.3 PRACTICE PROBLEMS
1. Draw a state diagram for a microwave oven.
1. Convert the following state diagram to equations.

2. Implement the following state diagram with equations.

Inputs:
- A
- B
- C
- D
- E
- F

Outputs:
- P
- Q
- R
- S

States:
- S0
- S1
- S2

Transition:
- S0 → S1: AC + D
- S1 → S2: BA + EC + D + F
- S2 → S1: FS

Initial State: ST 1
Given the following state diagram, use equations to implement ladder logic.

Convert the following state diagram to logic using equations.

You have been asked to program a PLC-5 that is controlling a handicapped access door opener. The client has provided the electrical wiring diagram below to show how the PLC inputs and outputs have been wired. Button A is located inside and button B is located outside. When either button is pushed the motor will be turned on to open the door. The motor is to be kept on for a total of 15 seconds to allow the person to enter. After the motor is turned off the door will fall closed. In the event that somebody gets caught in the door the thermal relay will go off, and the motor should be turned off. After 20,000 cycles the door should stop working and the light...
plc states - 12.32
should go on to indicate that maintenance is required.
24 V DC
Output Card
rack 00
slot 0
COM
00
01
02
03
04
05
06
07
24 V lamp
Relay
+24 V DC
Power
Motor
Supply
COM.
GND
a) Develop a state diagram for the control of the door.

b) Convert the state diagram to ladder logic. (list the input and the output addresses first)

c) Convert the state diagram to Boolean equations.

7. Design a garage door controller using a) block logic, and b) state-transition equations. 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.
plc states - 12.35

\[ T_1 \cdot T_2 \cdot T_3 \cdot T_4 \cdot T_5 \cdot T_6 = ST_1 \]

\[ ST_2 \cdot T_1 \cdot T_3 \cdot T_5 = ST_3 \]

\[ ST_4 \cdot T_5 = ST_5 \]

\[ ST_6 \cdot T_5 = ST_7 \]

[Diagram of plc states and transitions]
6. Door idle motor on
Door opening
Service mode
Button A + button B
Thermal relay + 15 sec delay
Counter > 20,000
Reset button - assumed

a) Legend
- Button A
- Button B
- Motor
- Thermal relay
- Reset button
- State 1
- State 2
- State 3

I:001/01
I:001/02
O:000/03
I:001/03
I:001/04 - assumed
B3:0/0
B3:0/1
B3:0/2
Lamp O:000/07

b)
st ate 3
st ate 1
st ate 3
reset button ??
MCR counter
S0
S0
S1
delay 15
( + ) thermal +
S0 buttonA buttonB +
S1 S1 S0 buttonA buttonB +
delay 15 ( + ) thermal +
S3 counter
S3 = S3 S2
counter ( + )
S3 reset ( + )
motor S1 =
light S3 =
c)
plc states - 12.42

a) block logic method
   door closing
   door opening
   door opened
   door closed
   remote OR button
   remote OR button OR top limit
   remote OR button OR bottom limit
   (state 2)
   (state 3)
   (state 1)
b) state-transition equations

$T_1 = \text{state 1 to state 2}$

$T_2 = \text{state 2 to state 3}$

$T_3 = \text{state 2 to state 4}$

$T_4 = \text{state 3 to state 4}$

$T_5 = \text{state 4 to state 1}$

$ST_1 = \text{state 1}$

$ST_2 = \text{state 2}$

$ST_3 = \text{state 3}$

$ST_4 = \text{state 4}$

using the previous state diagram.

$ST_1 + (ST_2 \cdot T_1) = ST_2$

$ST_2 + (ST_3 \cdot T_2 \cdot T_3) = ST_3$

$ST_3 + (ST_4 \cdot T_3 \cdot T_4) = ST_4$

$ST_1 \cdot T_5 = ST_1$
plc states - 12.48
ST 1
T5
T1 ST1
ST 2
T1
T2 ST2
T3
ST 3
T2
T4 ST3
T4
ST 4
T3
T5 ST4
T4
ST2 close doo
ST4 open doo
T4:0
preset 300s
T4:0/DN garage light
1. Describe the difference between the block logic, delayed update, and transition equation methods for converting state diagrams to ladder logic.

2. Write the ladder logic for the state diagram below using the block logic method.

3. Convert the following state diagram to ladder logic using the block logic method. Give the stop button higher priority.
4. Convert the following state diagram to ladder logic using the delayed update method.

5. Use equations to develop ladder logic for the state diagram below using the delayed update method. Be sure to deal with the priority problems.

```
idle
active
fault
reset
jam
part
part
FS
STA
ST B
ST C
ST D
FS
A
BC
D + E
EE
```
6. Implement the State-Transition equations in the figure below with ladder logic.

7. Write ladder logic to implement the state diagram below using state transition equations.

8. Convert the following state diagram to ladder logic using a) an equation based method, b) a first scan (FS)
The state diagram below is for a simple elevator controller. a) Develop a ladder logic program that implements it with Boolean equations. b) Develop the ladder logic using the block logic technique. c) Develop the ladder logic using the delayed update method.

10. Write ladder logic for the state diagram below a) using an equation based method. b) without

---

Diagram:

- ST A
- ST B
- ST C
- STD
- START
- 5s delay
- STOP
- LIMIT FAULT
- DONE
- RESET
- move
- up
- pause
- up
- idle
- move
- down
- pause
- down
- up_request
- down_request
- at_floor
- door_closed
- at_floor
- up_request
- down_request
- door_closed
11. For the state diagram for the traffic light example, add a 15 second green light timer and speed up signal for an emergency vehicle. A strobe light mounted on fire trucks will cause the lights to change so that the truck doesn't need to stop. Modify the state diagram to include this option. Implement the new state diagram with ladder logic.

12. Design a program with a state diagram for a hydraulic press that will advance when two palm buttons are pushed. Top and bottom limit switches are used to reverse the advance and stop after a retract. At any time the hands removed from the palm button will stop an advance and retract the press. Include start and stop buttons to put the press in and out of an active mode.

13. In dangerous processes it is common to use two palm buttons that require an operator to use both hands to start a process (this keeps hands out of presses, etc.). To develop this there are two inputs (P1 and P2) that must both be turned on within 0.25s of each other before a machine cycle may begin.

Develop ladder logic with a state diagram to control a process that has a start (START) and stop (STOP) button for the power. After the power is on the palm buttons (P1 and P2) may be used as described above to start a cycle. The cycle will consist of turning on an output (MOVE) for 2 seconds. After the press has been cycled 1000 times the press power should turn off and an output (LIGHT) should go on.
14. Use a state diagram to design a parking gate controller.

15. This morning you received a call from Mr. Ian M. Daasprate at the Old Fashioned Widget Company. In the past when they built a new machine they would use punched paper cards for control, but their supplier of punched paper readers went out of business in 1972 and they have decided to try using PLCs this time. He explains that the machine will dip wooden parts in varnish for 2 seconds, and then apply heat for 5 minutes to dry the coat, after this they are manually removed from the machine, and a new part is put in. They are also considering a premium line of parts that would call for a dip time of 30 seconds, and a drying time of 10 minutes. He then refers you to the project manager, Ann Nooyed.

You call Ann and she explains how the machine should operate. There should be start and stop buttons. The start button will be pressed when the new part has been loaded, and is ready to be coated. A light should be mounted to indicate when the machine is in operation. The part is mounted on a wheel that is rotated by a motor. To dip the part, the motor is turned on until a switch is closed. To remove the part from the dipping bath the motor is turned on until a second switch is closed. If the motor to rotate the wheel is on for more than 10 seconds before hitting a switch, the machine should be turned off, and a fault light turned on. The fault condition will be cleared by manually setting the machine back to its initial state, and hitting the start button twice. If the part has been dipped and dried properly, then a done light should be lit. To select a premium product you will use an input switch that needs to be pushed before the start button is pushed. She closes by saying she will be going on vacation and you need to have it done before she returns.

You hang up the phone and, after a bit of thought, decide to use the following outputs and inputs:

- **Keycard entry**
- **Gate**
- **Car detector**
- **Light**

- The gate will be raised by one output and lowered by another. If the gate gets stuck an over current detector will make a PLC input true. If this is the case the gate should reverse and the light should be turned on indefinitely.
- If a valid keycard is entered a PLC input will be true. The gate is to rise and stay open for 10 seconds.
- When a car is over the car detector a PLC input will go true. The gate is to open while this detector is active. If it is active for more than 30 seconds the light should also turn on until the gate closes.
a) Draw a state diagram for the process.

b) List the variables needed to indicate when each state is on, and list any timers and counters used.

c) Write a Boolean expression for each transition in the state diagram.

16. Design ladder logic with a state diagram for the following process description.

a) A toggle start switch (TS1) and a limit switch on a safety gate (LS1) must both be on before a solenoid (SOL1) can be energized to extend a stamping cylinder to the top of a part. Should a part detect sensor (PS1) also be considered? Explain your answer.

b) While the stamping solenoid is energized, it must remain energized until a limit switch (LS2) is activated. This second limit switch indicates the end of a stroke. At this point the solenoid should be de-energized, thus retracting the cylinder.

c) When the cylinder is fully retracted a limit switch (LS3) is activated. The cycle may not begin again until this limit switch is active. This is one way to ensure that a new part is present, is there another?

d) A cycle counter should also be included to allow counts of parts produced. When this value exceeds some variable amount (from 1 to 5000) the machine should shut down, and a job done light lit up.

e) A safety check should be included. If the cylinder solenoid has been on for more than 5 seconds, it suggests that the cylinder is jammed, or the machine has a fault. If this is the case the machine should be shut down, and a maintenance light turned on.

f) Implement the ladder diagram on a PLC in the laboratory.

g) Fully document the ladder logic and prepare a short report - This should be of use to another engineer that will be maintaining the system.

I/1 - start push button
I/2 - stop button
I/3 - premium part push button
I/4 - switch - part is in bath on wheel
I/5 - switch - part is out of bath on wheel

INPUTS

O/1 - start button
O/2 - in operation
O/3 - fault light
O/4 - part done light
O/5 - motor on
O/6 - heater power supply

OUTPUTS