10. STRUCTURED LOGIC DESIGN

10.1 INTRODUCTION

Traditionally ladder logic programs have been written by thinking about the process and then beginning to write the program. This always leads to programs that require debugging. And, the final program is always the subject of some doubt. Structured design techniques, such as Boolean algebra, lead to programs that are predictable and reliable. The structured design techniques in this and the following chapters are provided to make ladder logic design routine and predictable for simple sequential systems.

Topics:

Objects:
• Know examples of applications to industrial problems.
• Know how to design time base control programs.

Timing diagrams
Design examples
Designing ladder logic with process sequence bits and timing diagrams

Note: Structured design is very important in engineering, but many engineers will write software without taking the time or effort to design it. This often comes from previous experience with programming where a program was written, and then debugged. This approach is not acceptable for mission critical systems such as industrial controls. The time required for a poorly designed program is 10% on design, 30% on writing, 40% debugging and testing, 10% documentation. The time required for a high quality program design is 30% design, 10% writing software, 10% debugging and testing, 10% documentation. Yes, a well designed program requires less time! Most beginners perceive the writing and debugging as more challenging and productive, and so they will rush through the design stage. If you are spending time debugging ladder logic programs you are doing something wrong. Structured design also allows others to verify and modify your programs.

Axiom: Spend as much time on the design of the program as possible. Resist the temptation to implement an incomplete design.
Most control systems are sequential in nature. Sequential systems are often described with words such as mode and behavior. During normal operation these systems will have multiple steps or states of operation. In each operational state the system will behave differently. Typical states include start-up, shut-down, and normal operation. Consider a set of traffic lights - each light pattern constitutes a state. Lights may be green or yellow in one direction and red in the other. The lights change in a predictable sequence. Sometimes traffic lights are equipped with special features such as cross walk buttons that alter the behavior of the lights to give pedestrians time to cross busy roads.

Sequential systems are complex and difficult to design. In the previous chapter timing charts and process sequence bits were discussed as basic design techniques. But, more complex systems require more mature techniques, such as those shown in Figure 10.1. For simpler controllers we can use limited design techniques such as process sequence bits and flow charts. More complex processes, such as traffic lights, will have many states of operation and controllers can be designed using state diagrams. If the control problem involves multiple states of operation, such as one controller for two independent traffic lights, then Petri net or SFC based designs are preferred.

Figure 10.1
Sequential Design Techniques

10.2 PROCESS SEQUENCE BITS
A typical machine will use a sequence of repetitive steps that can be clearly identified as a single process.

- **STATE DIAGRAM**: Suitable for complex processes with many states.
- **FLOW CHART**: Clear and straightforward, good for simple processes.
- **SEQUENCE BITS**: Suitable for processes with deviations.
- **FLOW CHART**: Good for processes with no waiting, single states.
- **PETRI NET**: Suitable for processes that need to be buffered (waiting).
- **SFC/GRAFSET**: Suitable for processes with state triggers.
plc design - 10.3

Fied. Ladder logic can be written that follows this sequence. The steps for this design method are:

1. Understand the process.
2. Write the steps of operation in sequence and give each step a number.
3. For each step assign a bit.
4. Write the ladder logic to turn the bits on/off as the process moves through its states.
5. Write the ladder logic to perform machine functions for each step.
6. If the process is repetitive, have the last step go back to the first.

Consider the example of a flag raising controller in Figure 10.2 and Figure 10.3. The problem begins with a written description of the process. This is then turned into a set of numbered steps. Each of the numbered steps is then converted to ladder logic.
Figure 10.2

A Process Sequence Bit Design Example

Description:
A flag raiser that will go up when an up button is pushed, and down when a down button is pushed, both push buttons are momentary. There are limit switches at the top and bottom to stop the flag pole. When turned on at first the flag should be lowered until it is at the bottom of the pole.

Steps:
1. The flag is moving down the pole waiting for the bottom limit switch.
2. The flag is idle at the bottom of the pole waiting for the up button.
3. The flag moves up, waiting for the top limit switch.
4. The flag is idle at the top of the pole waiting for the down button.

This section of ladder logic forces the flag raiser to start with only one state on, in this case it should be the first one, step 1.

Ladder Logic:

- **down**: 
  - The ladder logic for step 1 turns on the motor to lower the flag and when the bottom limit switch is hit it goes to step 2.

- **motor**: 
  - The ladder logic for step 2 only waits for the push button to raise the flag.
The previous method uses latched bits, but the use of latches is sometimes discouraged. A more common method of implementation, without latches, is shown in Figure 10.4.

Step 3

The ladder logic for step 3 turns on the motor to raise the flag and when the top limit switch is hit it goes to step 4.

Motor

Step 4

The ladder logic for step 4 only waits for the push button to lower the flag.

Flag down button
Timing diagrams can be valuable when designing ladder logic for processes that are only dependant on time. The timing diagram is drawn with clear start and stop times. Ladder logic is constructed with timers that are used to turn outputs on and off at appropriate times. The basic method is:

1. Understand the process.

---

![Diagram](image-url)
2. Identify the outputs that are time dependent.
3. Draw a timing diagram for the outputs.
4. Assign a timer for each time when an output turns on or off.
5. Write the ladder logic to examine the timer values and turn outputs on or off.

Consider the handicap door opener design in Figure 10.5 that begins with a verbal description. The verbal description is converted to a timing diagram, with $t=0$ being when the door open button is pushed. On the timing diagram the critical times are 2s, 10s, 14s. The ladder logic is constructed in a careful order. The first item is the latch to seal-in the open button, but shut off after the last door closes. `auto` is used to turn on the three timers for the critical times. The logic for opening the doors is then written to use the timers.
Description: A handicap door opener has a button that will open two doors. When the button is pushed (momentarily) the first door will start to open immediately, the second door will start to open 2 seconds later. The first door power will stay open for a total of 10 seconds, and the second door power will stay on for 14 seconds. Use a timing diagram to design the ladder logic.

Ladder Logic:

```
open button
auto

T4:2/DN
TON
Timer T4:0
Delay 2s
TON
Timer T4:1
Delay 10s
TON
Timer T4:2
Delay 14s

auto
T4:1/TT

T4:2/TT

T4:0/DN
```

Figure 10.5
Design With a Timing Diagram
10.4 DESIGN CASES

Timing diagrams can show how a system changes over time. Process sequence bits can be used to design a process that changes over time. Timing diagrams can be used for systems with a time driven performance.

10.6 PRACTICE PROBLEMS

1. Write ladder logic that will give the following timing diagram for B after input A is pushed. After A is pushed any changes in the state of A will be ignored.

2. Design ladder logic for the timing diagram below. When an input A becomes active the sequence should start.

3. A wrapping process is to be controlled with a PLC. The general sequence of operations is described below. Develop the ladder logic using process sequence bits.
   1. The folder is idle until a part arrives.
   2. When a part arrives it triggers the part sensor and the part is held in place by actuating the hold actuator.
3. The first wrap is done by turning on output paper for 1 second.
4. The paper is then folded by turning on the crease output for 0.5 seconds.
5. An adhesive is applied by turning on output tape for 0.75 seconds.
6. The part is released by turning off output hold.
7. The process pauses until the part sensors go off, and then the machine returns to idle.

10.7 PRACTICE PROBLEM SOLUTIONS
1. TON Timer T4:0 Base 1s
   ---
   T4:0/DN TON Timer T4:2 Base 1s
   ---
   T4:2/DN
   ---
   T4:2/TT
   ---
   T4:4/TT
PLC design - 10.11

TON T4:0 0.100 s
TON T4:1 0.300 s
TON T4:2 0.500 s
TON T4:3 0.700 s
TON T4:4 0.900 s
TON T4:5 1.100 s
TON T4:6 1.900 s

A T4:0/TT
T4:2/DN

X T4:0/EN
T4:2/DN T4:3/DN
T4:4/DN T4:5/DN
T4:5/TT

Y Z
ASSIGNMENT PROBLEMS

1. Convert the following timing diagram to ladder logic. It should begin when input 'A' becomes active.

   - **Step 1:**
     - Latch
     - TON T4:0 delay 1 s
     - L
     - Step 2:
     - TON T4:1 delay 0.5 s
     - L
     - Step 3:
     - TON T4:2 delay 0.75 s
     - L

   - **Step 2:**
     - T4:0/DN
     - Step 3:
     - T4:1/DN

   - **Step 4:**
     - T4:2/DN

   - **Step 5:**
     - T4:0/DN

2. Use the timing diagram below to design ladder logic. The sequence should start when input X turns on. X may only be on momentarily, but the sequence should continue to execute until it ends at 26 seconds.

3. Use the timing diagram below to design ladder logic. The sequence should start when input X turns on. X may only be on momentarily, but the sequence should execute anyway.

4. A PLC is to control an amusement park water ride. The ride will fill a tank of water and splash a tour group. 10 seconds later a water jet will be ejected at another point. Develop ladder logic for the process that follows the steps listed below.

1. The process starts in 'idle'.
2. The 'cart_detect' opens the 'filling' valve.
3. After a delay of 30 seconds from the start of the filling of the tank the tank 'outlet' valve opens. When the tank is 'full' the 'filling' valve closes.
4. When the tank is empty the 'outlet' valve is closed.
5. After a 10 second delay, from the tank outlet valve opening, a water 'jet' is ejected.
6. After '2' seconds the water 'jet' is closed and the process returns to the 'idle state'.

5. Write a program that will execute the following steps. When in steps b) or d), output C will be true. Output X will be true when in step c).

a) Start in an idle state. If input G becomes true go to b)

b) Wait until P becomes true before going to step c). If input S becomes true then go to step a).

c) Wait for 3 seconds then go to step d).

d) Wait for P to become false, and then go to step b).

6. Write a ladder logic program to extend and retract a cylinder after a start button is pushed. There are limit switches at the ends of travel. If the cylinder is extending if more than 5 seconds the machine should shut down and turn on a fault light. If it is retracting for more than 3 seconds it should also shut down and turn on the fault light. It can be reset with a reset button.

7. 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.

8. Write a program that will execute the following steps. When in steps b) or d), output C will be true. Output X will be true when in step c).

a) Start in an idle state. If input G becomes true go to b)

b) Wait until P becomes true before going to step c). If input S becomes true then go to step a).

c) Wait for 3 seconds then go to step d).

d) Wait for P to become false, and then go to step b).

9. A machine has been built for filling barrels. Use process sequence bits to design ladder logic for the sequential process as described below.

1. The process begins in an idle state.

2. If the 'fluid_pressure' and 'barrel_present' inputs are on, the system will open a flow valve for 2 seconds with output 'flow'.

3. The 'flow' valve will then be turned off for 10 seconds.

4. The 'flow' valve will then be turned on until the 'full' sensor indicates the barrel is full.

5. The system will wait until the 'barrel_present' sensor goes off before going to the idle state.

10. Design ladder logic for an oven using process sequence bits. (Note: the solution will only be graded if the process sequence bit method is used.) The operations are as listed below.

1. The oven begins in an IDLE state.

2. An operator presses a start button and an ALARM output is turned on for 1 minute.

3. The ALARM output is turned off and the HEAT is turned on for 3 minutes to allow the temperature to rise to the acceptable range.

4. The CONVEYOR output is turned on.

5. If the STOP input is activated (turned off) the HEAT will be turned off, but the CONVEYOR output will be kept on for two minutes. After this the oven returns to IDLE.
11. We are developing a safety system (using a PLC-5) for a large industrial press. The press is activated by turning on the compressor power relay (R, connected to O:013/05). After R has been on for 30 seconds the press can be activated to move (P connected to O:013/06). The delay is needed for pressure to build up. After the press has been activated (with P for 1.0 seconds) the system must be shut down (R and P off), and then the cycle may begin again. For safety, there is a sensor that detects when a worker is inside the press (S, connected to I:011/02), which must be off before the press can be activated. There is also a button that must be pushed 5 times (B, connected to I:011/01) before the press cycle can begin. If at any time the worker enters the press (and S becomes active) the press will be shut down (P and R turned off). Develop the process sequence and sequence bits, and then ladder logic for the states. State all assumptions, and show all work.

12. A machine is being designed to wrap boxes of chocolate. The boxes arrive at the machine on a conveyor belt. The list below shows the process steps in sequence.

1. The box arrives and is detected by an optical sensor (P), after this the conveyor is stopped (C) and the box is clamped in place (H).
2. A wrapping mechanism (W) is turned on for 2 seconds.
3. A sticker cylinder (S) is turned on for 1 second to put consumer labelling on the box.
4. The clamp (H) is turned off and the conveyor (C) is turned on.
5. After the box leaves the system returns to an idle state.

Develop ladder logic programs for the system using the following methods. Don't forget to include regular start and stop inputs.

i) a timing diagram
ii) process sequence bits