Session 2648

Structured PLC Programming with Sequential Function Charts

James A. Rehg
Pennsylvania State University

Abstract

Programmable logic controllers (PLCs) have been programmed using ladder logic since their introduction in the early 1970s. The programming technique most often used follows an empirical process that begins with entering the first rung to satisfy the first output requirement. Rungs continue to be entered leading toward a solution to the problem. At some point in the programming process the target system fails to respond as desired when the program is tested, and corrections are made to rungs in order to fix the problems. This process continues until a program solution is reached. The result is a program without sequential structure that is difficult to analyze and understand. With large PLC programs, troubleshooting the program when system problems develop is difficult.

This paper describes how a technique called sequential function charts (SFC) is used in a two-course sequence covering PLC programming. The SFC process adds structure to the PLC programming process and produces programs that are easy to analyze and troubleshoot. The paper also includes a description of the SFC process; types of problems assigned for student work; and the benefit derived from using SFC in a PLC programming laboratory or class.

Introduction

PLCs have been used for industrial control since their development in the early 1970s. The PLC programming language, called *ladder logic*, was designed to look like two-wire relay control logic schematics. This similarity was necessary at the start of PLC applications because the electricians who were responsible for the maintenance of the systems were familiar with relay diagrams. As a result, the integration of PLCs into industrial control applications was so successful that PLCs are used in industrial plants of all sizes in most automated manufacturing plants around the globe. However, the overwhelming success of PLCs and the ladder logic language has caused the following problems:

- No industry-wide standards for ladder logic program syntax exists
- Programs are not interchangeable among PLCs from different vendors
- Programs written by different programmers for the same problem rarely are the same
- Ladder logic programs are difficult to read and troubleshoot
- Ladder logic is not well suited to some of the current high-end automation applications

As a result of the large market for PLCs, many vendors provide PLC devices and a ladder logic programming language. However, the program command syntax from various vendors is not common because it was not developed around any industry-wide standard. Therefore, programs are not interchangeable among PLCs from different vendors. The typical ladder logic language program is developed without the program flow structure commonly used in programs developed with the current higher-level languages like C or Java. As a result, programs can be very different for similar projects; are often hard to read and understand; and are difficult to troubleshoot when automation failures occur. Currently PLCs are used to control a wide variety of industrial control applications from simple discrete control of motors to position and speed control of multiple-motor servo systems. The ladder logic structure does not support programming of these high-end applications as well as a more traditional programming language. As the size and complexity of PLC control applications increases, the ability of ladder logic to provide a program that is easy to maintain decreases.

Several techniques have been adopted to add some structure to PLC program development, including Grafcet, Sequential Function Charts (SFCs), and Stage Programming. Each technique is similar and provides structure while continuing to support the ladder logic syntax. Grafcet was developed in France in 1977 using theory from State Machines and Petri Nets. In 1988 Grafcet evolved into an IEC (International Electrotechnical Commission) standard 848 under the name Sequential Function Charts. Texas Instruments developed Stage Programming for its line of PLCs in the mid 1980s. While the techniques are not *silver bullets* for PLC programming issues, they provide structure that improves program readability. Also, they improve troubleshooting of programs controlling automation systems. The use of the SFCs approach in automation control labs at Penn State Altoona is described in this paper

Empirical PLC Programming

Programmable logic controller programming is accomplished using two techniques: empirical and pseudo structured. The empirical approach follows the following steps:

1. Clearly state and defined the control problem.
2. Identify all of the control outputs and the control requirements for each.
3. For each control output identify all input conditions combinations that are required to produce an active output state.
4. Using the input conditions identified in Step 3, create ladder rungs for each control output.
5. Scan the program from the first rung to the last to verify that the outputs produce the desired system operation when the input states change.
6. Add additional rungs to correct problems identified in Step 5.
7. Document the ladder logic by identifying all inputs and outputs and by labeling the program and all the program rungs.

The empirical approach described by the seven step process works well for small ladder logic applications. However, as the control problem gets more complex, the number of problems located in Step 5 increases and the number of rungs added to patch problems increases. As a result, the program gets more difficult to read and understand.

Another problem created by this approach is the non-sequential nature of the program structure. While the PLC scans the program from the first rung down to the last, turning on outputs based on the input logic, the sequential change in the output states does not necessarily follow in that order. For example, the output on the last rung may turn on first and be followed by another output somewhere else in the ladder. It is not uncommon for a large system control problem to have a ladder with several thousand rungs present. Determining the output sequence and machine operation in such a large system would be difficult if the program was developed using this empirical technique.

Pseudo Structured PLC Programming

The techniques identified earlier, like Grafcet and Sequential Function Chart (SFC), provide a mechanism to add structure to ladder logic programming. The structure comes with some trade-offs; the program development time and the number of program rungs increases, but the implementation time decreases. In addition, the program is easier to follow, and the system is less difficult to troubleshoot when problems occur.

The development of a SFC program uses the following steps:

1. Study the system operation in order to become familiar with every detail of the process.
2. Make a numbered list of every step in the production sequence.
3. Identify and list all outputs for the system and the action produced by the output.
4. Identify and list all of the sensors in the system and the input condition necessary to produce an output.
5. Use the data from Steps 2 through 4 to create a transition condition (TC) list that indicates the system condition that would cause the process to move from one production sequence to the next. The transition conditions are indicated by the term *TCi.j* where *i* represents the number of the current system state or step, and *j* represents the next state or step number. The following transition condition in equation form defines the condition for transition from Step 2 to Step 3:

   TC2.3 = Part Clamped = LS2 (on)

6. Use the data from Steps 1 through 5 to create a *sequential function chart* with the step numbers (Step 2) included in the left box, and the output action (Step 3) included in the box next to the associated step number. The transition condition (Step 5) that moves the process from the active step to the next step is placed between the two events in the chart. An example *sequential function chart* is illustrated in Figure 1.
7. Draw a process-timing diagram for the output actuators. The diagram has the step numbers from the *sequential function chart* (Step 6) listed across the top and the outputs listed in the left-hand column. Each vertical line in the chart represents a different step in the machine process. The distance between the vertical lines has no time reference for event driven steps; however, if a timer or counter drives an event, then the distance between the event lines has the units of time or counts. A process-timing diagram is illustrated in Figure 2.

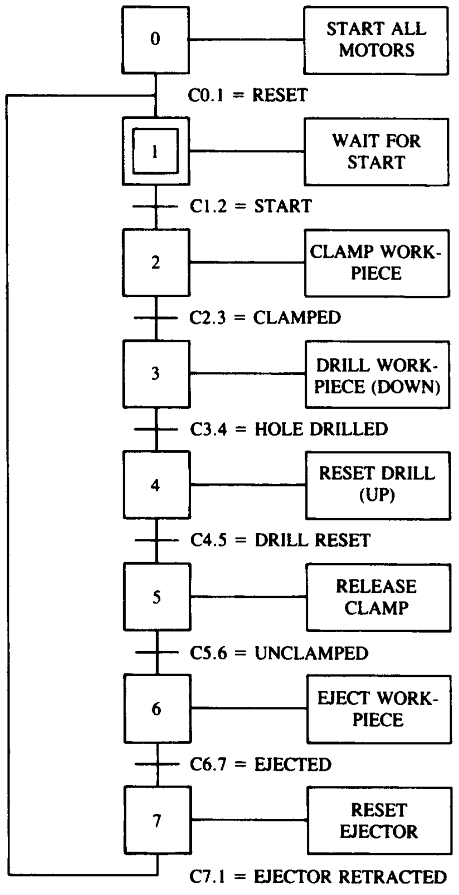

Figure 1

8. Draw the timing waveforms (Figure 2) for the outputs on the process-timing diagram indicating during which steps the output is active or on and those steps where the output is off.
9. Define the output Boolean functions using the Output Timing Diagram (Figure 2) and create the ladder logic necessary to control the output actuators. The following output function example indicates that solenoid A is energized only during Step 3, 4 and 5.

Sol C = Step 3 + Step 4 + Step 5

Sol C = CR 3 + CR 4 + CR 5

Sol C = B3/3 + B3/4 + B3/5

The equation is written in three forms. In the first, the steps in the SFC are referenced; in the second, the output is expressed as a function of control relays used with relay ladder logic; and in the last, the output is expressed using Binary Bits, assuming that a PLC is used to control the process.

10. Draw the sequential function logic circuits for every step. The circuit is illustrated in Figure 3. For a PLC implementation the control relays (i.e. CR1) would be binary bits (i.e. B3/0), and in a

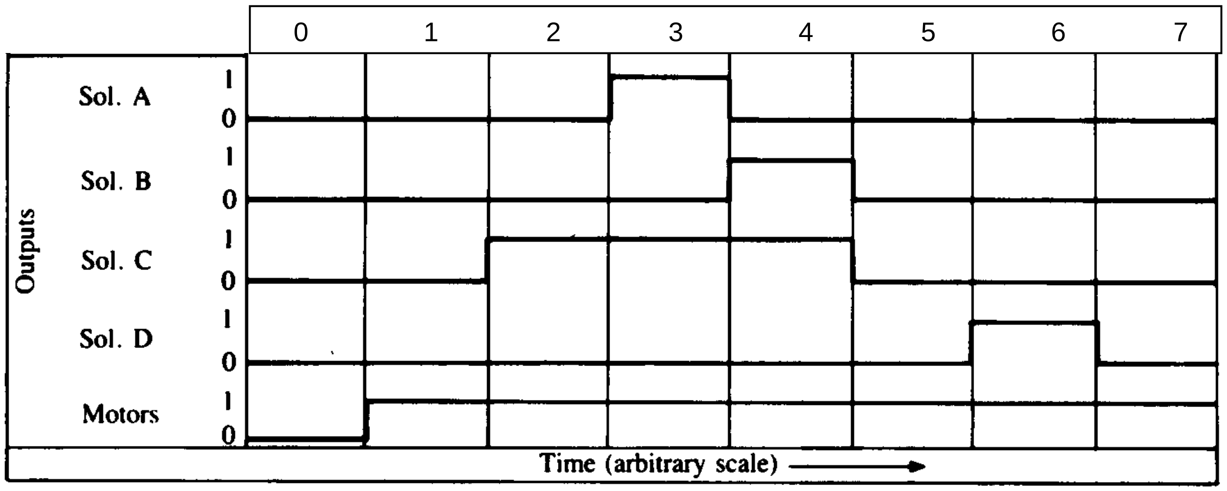

Figure 2

similar fashion the contacts would be referenced to the corresponding binary bit. The limit switch contacts would be referenced to PLC input channels.

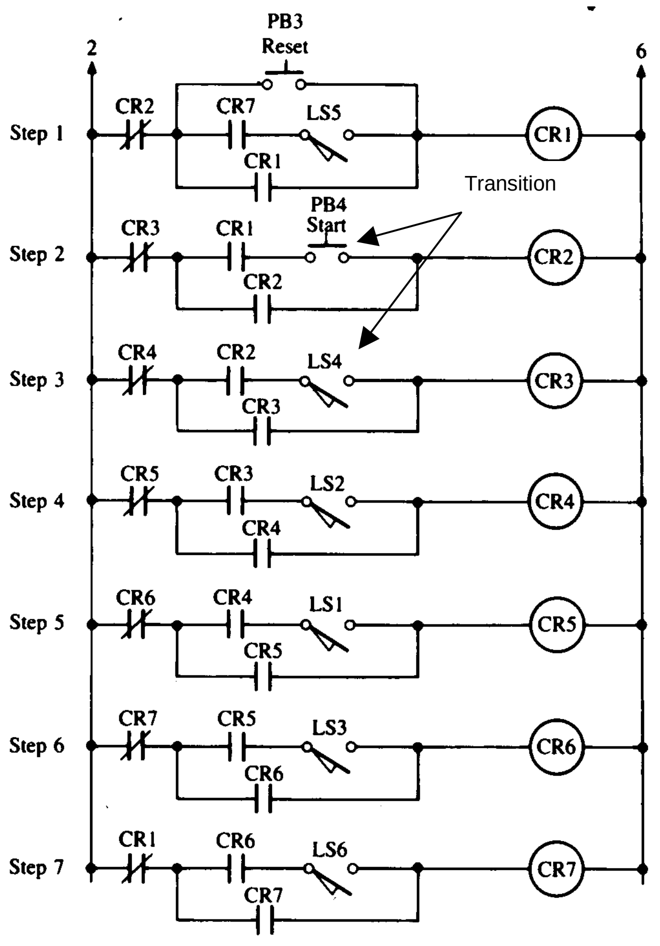

Figure 3

11. Draw the output ladder diagram with a rung for each output element in the process. The ladder logic is illustrated in Figure 4. The output bits or control relays from the sequential function ladder logic (Figure 4) are used to turn on the system outputs at the desired step or stage. If an output must be on for more than one state, then the corresponding SFC ladder bits are ORed together to produce this desired function.
12. Document the design by annotating the PLC ladder logic including the information from the previous steps in the design process.

The SFC design has some significant differences compared to empirical designs. For example, only one of the SFC logic outputs (i.e; CR1 or B3/1) can be on at any one time. Also, the transition through the outputs is sequential, and the transition condition triggers the next step or stage to the on state, thus turning off the previous stage.

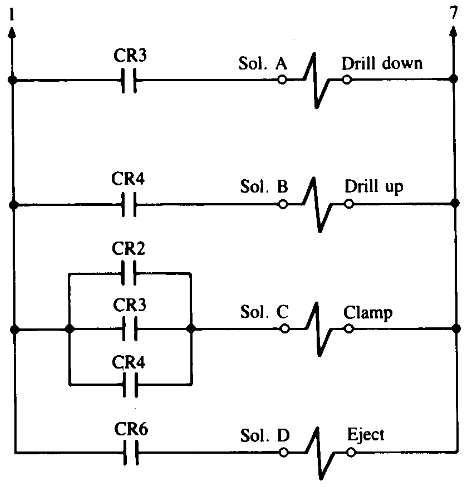

Figure 4

SFC at Penn State Altoona

The BS in Electromechanical Engineering Technology at Penn State Altoona has a two-course sequence covering discrete control of automation machines. In the second course students are required to program the operation of an automation system that sorts two different parts and then combines then into an assembly. The process also includes a quality control check for completed assembly and proper part diameter. Bad parts are rejected, and good parts are passed to a good part queue. In the final stage of the assembly, the system must operate with two parts present in the quality checking area where a series of five quality sensors and gages and a reject actuator must be able to distinguish between the two assemblies as they move through.

In the past student teams were permitted to choose any PLC programming strategy for the solution to the assembly system problem, and they used the empirical method exclusively. The complex nature of the final quality testing area and the unstructured nature of the empirical process combined to leave students frustrated after spending many hours trying to fix programming problems. Three years ago students were required to use the SFC technique on the quality testing part of the system. The following results were observed:

- Program planning and development time increased significantly.
- Program implementation and troubleshooting of programming problems reduced significantly.
- The PLC ladder logic had slightly more rungs.
- Students had a better understanding of system operation.
- Both team members could use the program documentation with equal ease.
- The total time required to develop an automation program for the system dropped from 24 laboratory meetings to 16.

Students find the planning and development of SFC programs to be more difficult than the traditional empirical method. However, after they complete the SFC process they understand the machine control problem to a much greater depth.

References

1. *Reference Manual*, Rockwell Automation, Inc., Milwaukee, WI, 1996.
2. *Discrete I/O Manual,* Rockwell Automation, Inc., Milwaukee, WI, 1996.
3. *Step-by-step Guide to Project Development*, Rockwell Automation, Inc., Milwaukee, WI, 1996.
4. *Bateson, R.N., Introduction to Control System Technology, Prentice Hall,1999*

BIOGRAPHY

JAMES A. REHG – James Rehg received a B. S. and an M. S. in Electrical Engineering from St. Louis University and has completed additional graduate work at the University of South Carolina and Clemson University. Since August 1995, Jim has been working as an assistant professor of engineering and as Program Coordinator of the B. S. degree program in Electro-mechanical Engineering Technology at Penn State Altoona. He is the author of five texts, including the following books published by Prentice Hall: *Introduction to Robotics in CIM Systems 4th ed.* and *Computer Integrated Manufacturing 2nd ed*.