Teaching Novices how to Program PLC's

Durward K. Sobek, II
Montana State University

Abstract

Programmable Logic Controllers (PLC's) are specialized microcomputers specifically designed for automated control of industrial processes. The most commonly used programming language for PLC's is ladder logic. This paper describes a classroom experiment designed to test the effectiveness of a graphical tool called "I/O mapping" in improving ladder programming. The experimental results are consistent with instructor observation that the technique helps students improve ladder program quality and/or problem-solving efficiency.

1. Introduction

Programmable Logic Controllers (PLC's) are specialized microcomputers specifically designed for automated discrete control of industrial processes. Any engineering student interested in industrial automation would benefit from a strong grounding in this technology. At Montana State University, the Industrial and Management Engineering program offers PLC programming as part of the first course in computer integrated manufacturing. This paper describes the results of a classroom research project conducted to test the efficacy of an alternative approach to solving PLC programming problems.

PLC's are unique from other microcomputer-based systems in that the most commonly used programming language, ladder logic, is graphical. Most novice students, when presented with a process description and asked to create a program in ladder logic to control the process, attempt to move directly from problem statement to solution. This approach to programming is difficult and the resulting solutions are often poor. I experimented with different solution approaches (i.e., introducing intermediate steps) and found that different forms of representing the problem space seemed to produce different levels of problem-solving efficacy. I then conducted a classroom experiment to test these observations with quantifiable data.

This paper describes several alternative approaches to ladder logic programming, the most effective of which seems to be an "I/O Mapping" technique. The paper then describes a before-and-after semi-controlled experiment to test this technique's effectiveness. The data show significant improvements in quality of solution, time spent working the problem, or both for most students. Although the data on their own are not conclusive (the experiment was not fully controlled), they triangulate well with instructor observations and student feedback, strongly suggesting that the technique helps students learn PLC programming quickly and effectively.

2. Ladder Logic and PLC Programming

Ladder logic evolved from relay logic developed in the days of manufacturing automation before computers were widely available. As microcomputers became more cost effective and reliable, special purpose computers were developed in the late 1960's to replace the cumbersome relay switchboard.[1] Ladder logic was developed based on relay logic to facilitate technician training. Implementation using a microcomputer allowed PLC developers to expand PLC capability to include timers, counters, analog-digital conversion, better user interfaces, and many other advanced functions.[2]

Figure 1 describes a very simple control problem and shows the solution in ladder logic. The way to interpret a ladder diagram is to think of the left rail as a positive electric terminal and right rail as ground. Each rung has only one output. The inputs are logically "open" or "closed", and can be arranged in series (logical AND) or in parallel (logical OR). If, when the microcomputer scans the inputs, the rung's logic is true, then the output is activated; otherwise, it is not activated. The scans occur very rapidly, so response time to an input change is virtually instantaneous (within a few microseconds). A ladder diagram can contain multiple rungs (although the diagram in Figure 1 has only one rung), inputs can be used on more than one rung, and outputs can be inputs making interlock situations possible.

Problem:
A fan is to be started and stopped from either of two locations. Each location has a start and a stop button. Construct a PLC ladder diagram to control the fan.

Solution:

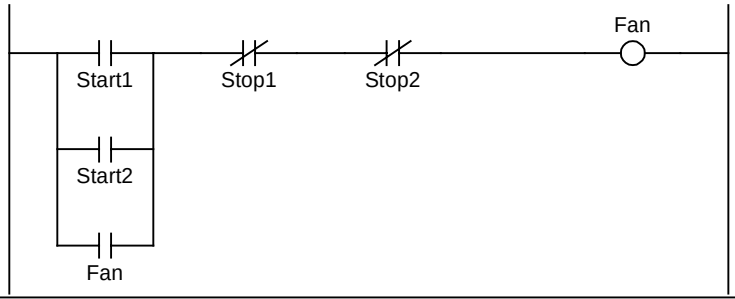


FIGURE 1: PLC PROBLEM AND SOLUTION

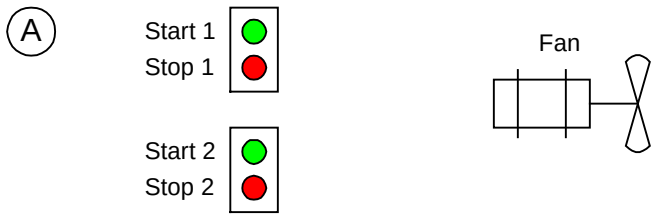
The solution approach depicted in Figure 1 implies that the problem-solver was able to develop a solution directly from the problem statement. This is certainly possible if the programmer is fluent in ladder logic and the problem is sufficiently simple. This is the solution approach presented in a number of textbooks.[1, 3, 4] However, students learning PLC programming are barely familiar with ladder logic. They typically find drawing a logic diagram directly from the problem statement a cognitively taxing exercise. Approaching the problem this way often results logically incorrect solutions; and if correct, the process often takes a surprisingly long time.

Webb and Reis recommend a 9-step approach to ladder programming.[2] This approach includes making a sketch of process to be controlled, then writing a step sequence list for the process. Figure 2 shows what this approach might look like for the fan control problem. The programmer

first drew a diagram of the system (A), then made a step sequence list (B), then finally developed the ladder program. A variation of this approach is to display the written sequence list graphically in the form of a flowchart.

Problem:
A fan is to be started and stopped from either of two locations. Each location has a start and a stop button. Construct a PLC ladder diagram to control the fan.

Solution:



B

1. Press Start 1, Fan turns on. Stays on when Start 1 released.
2. Press Stop 1, Fan turns off.
3. Press Start 2, Fan turns on. Stays on when Start 2 released.
4. Press Stop 2, Fan turns off.

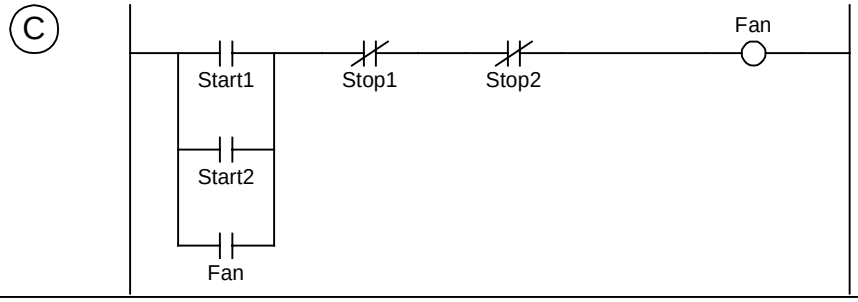


FIGURE 2: PLC PROBLEM AND SOLUTION WITH SEQUENCE LIST

I introduced my students to a different technique I call *I/O mapping*. The technique is to draw a diagram of the system that includes all inputs and outputs. Then, the student reads the problem statement phrase by phrase and draws arrows from every device that affects another to the affected device. Finally, the student constructs a rung for each output in the system. The rung should include an input for each incoming arrow. To do this last step, creating a Boolean algebraic expression is often helpful. Boolean algebra represents the logical relationships of the different rung elements, from which the ladder diagram can be readily derived. (My students gain some familiarity with Boolean algebra earlier in the course, including: basic operators, truth tables, basic identities, DeMorgan's Theorem, Karnaugh Maps, latches, and flip flops.)

Figure 3 illustrates this alternative approach. From the problem statement, a simple diagram of a fan and two control switches is drawn (A). Then, since the problem statement says that the fan is to be controlled from both locations, arrows are drawn from the control switches to the fan. Next, a Boolean expression (B) describes the logic for the only output in the system, the fan. In English, the Boolean expression reads, "The fan is true (or on) if Start1 or Start2 is on, or if the fan is on,

and Stop1 is off and Stop2 is off." From the algebra, the ladder logic (C) is straightforward.

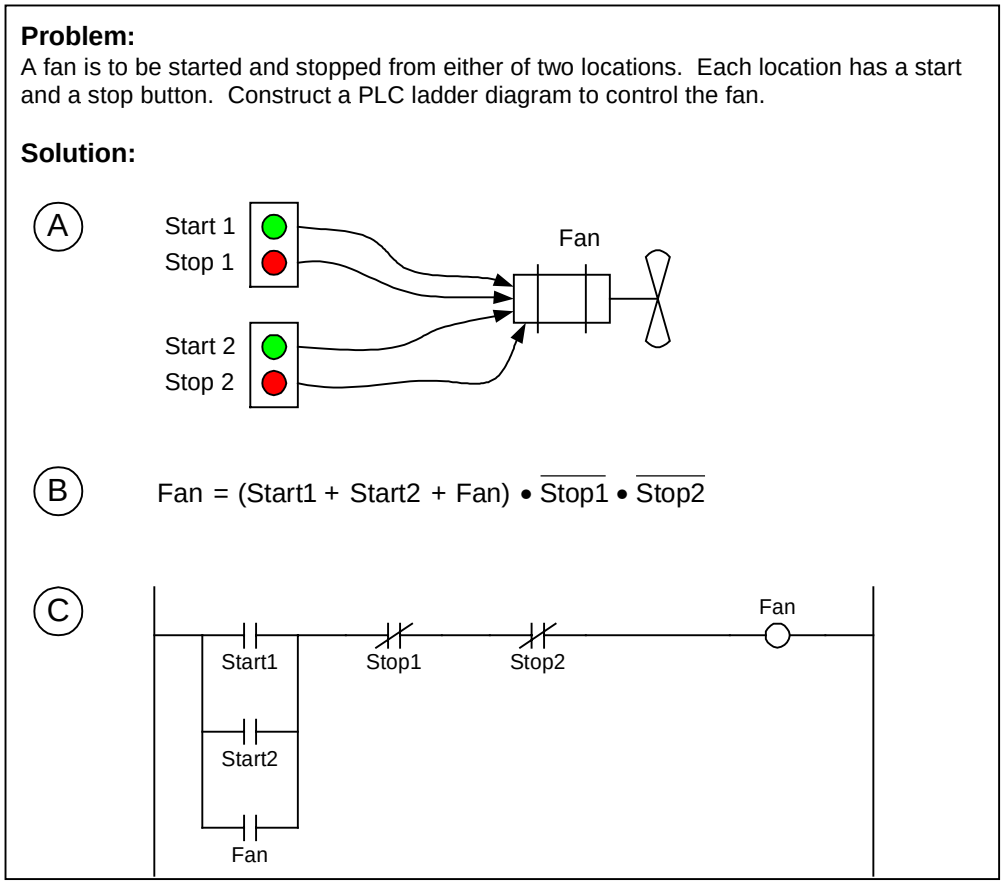


FIGURE 3: PLC PROBLEM AND SOLUTION USING I/O MAPPING TECHNIQUE

The I/O mapping technique seems to work well for me (the course instructor) and for at least some of the students vis-à-vis the other approaches. To gain a broader perspective on its effectiveness, I conducted a classroom with help from an undergraduate student to gather empirical, quantitative evidence of its effectiveness.

3. Research Method

The experiment was conducted in the Fall 1999 semester of I&ME 271 *Microcomputers in Industry*. The objective of the study was to supply some data in support of the hypothesis that the I/O map helps improve problem solving effectiveness for ladder logic programming problems (part of larger study[5]). A homework sequence was designed to determine the "before and after" effect of the tool (see Figure 4). Students solved a homework problem using whatever method they considered appropriate. After turning the first problem in, students learned the I/O mapping technique through an example problem in class, then solved a second homework problem of comparable difficulty to the first. The second homework instructed students to use the tool taught in class. Students were asked to self-report completion times for both homework assignments. Thus, we were able to measure both the quality of solution and time spent working on the problem.

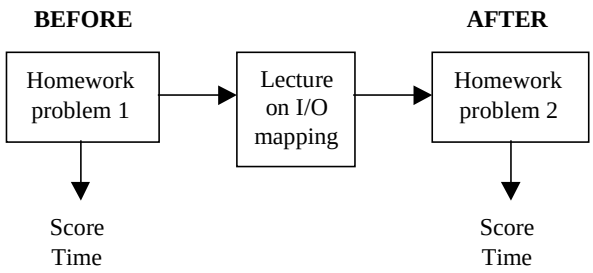


FIGURE 4: CHRONOLOGICAL HOMEWORK SEQUENCE

The homework problems were carefully designed to be of comparable difficulty (see Appendix for actual assignments given). They were open-ended "story" problems, meaning that data were presented in narrative form and that multiple solutions were possible. As such, they were simple design problems. The two assignments were given in a span of 7 days (HW 1 given on a Monday due on a Friday, HW 2 given that same Friday, due on Monday) to minimize learning effects that might occur between the two assignments.

The instructor (Sobek) evaluated problem solutions, and gave each one of three scores: very satisfactory, satisfactory, and unsatisfactory (S+, S, and S-). For a solution to be very satisfactory, the ladder program should perform the required task with minimal or no flaws. A satisfactory program would not perform as required due to two or more minor errors in logic. An unsatisfactory solution would be a significant attempt to solve the problem, but with one or more major logical flaws in the program. Solutions were not graded if considered an insignificant attempt to solve the problem. Homework assignments specifically asked subjects to report as accurately as possible the amount of time spent on the problem, and indicated that times would not impact grades.

For analysis, we first calculated a two-tailed paired t-test statistic to determine if changes in overall class performance (scores or times) were significant from the first to second homework. Numerical scores were assigned to each grade category: 10 for S+, 8 for S, and 5 for S-. Due to limited sample size and small number of measured variables, more sophisticated statistical analysis methods were not likely to produce useful results.

Then, we conducted an individual analysis to determine whether individual subjects improved in score or time from the first homework to the second. Scores were coded as "improved," "same," or "worse" depending on whether an individual's scores changed from the first homework to the second. Likewise, times were coded similarly depending on whether completion time increased or decreased by more than 10%. Frequencies in each category and in combined time and score categories were tallied. For the combined results, we deemed an "improvement" in problem solving as an increase in score (quality of solution) without a corresponding increase in time spent working on the problem, or spending less time to reach the same quality of solution, or both (less time spent to achieve a higher quality solution). Consequently, a decrease in problem solving performance would be indicated by a worse score for the same amount of time spent on the problem, spending more time to achieve the same quality of solution, or both (more time spent to achieve a lower quality solution). Cases where students spent more time to achieve a higher score, or received a lower score but spent less time on the homework, are indeterminate.

4. Analysis Results

Table 1 presents the results for the group analysis. It contains scores and times with means and standard deviations in the bottom row. The sample size was 24 for each homework set, but ranged between 18 and 21 for analysis depending on student response (note that a number of students did not turn in one of the assignments). T-test of the mean scores indicates a significant improvement of 22% (p-value = 0.0288). T-test of mean completion times indicates that the average 10-minute improvement was only slightly significant (p-value = 0.1412) due to the large variation in completion times for the first homework. The results for score support the hypothesis that using the I/O mapping technique improves quality of the solution. The results for time provide some support for the hypothesis because times on average did not increase even with an added (and unfamiliar) task—using the graphical mapping tool.

Subject Code	HW 1 Score	HW 2 Score	HW 1 Time	HW 2 Time
1	5	5	120	60
2	5	8	25	40
3	8	10	40	30
4	5	10	56	43
5	5	8	40	25
6	10	10	32	38
7		8		30
8	5	5	75	20
9	5	5	40	40
10	5	5	30	70
11	8	8	45	35
12	5	8	60	45
13	5	8	30	30
14	5	5	60	40
15	5	5	30	20
16		8		45
17	5	8	60	30
19	5	5	26	50
20	5	5	45	45
21	8	5	30	20
22	5	8	60	60
23	8	5	150	30
25	8	8	45	35
26	5		45	
28	5		15	
29	5	10	45	90
Mean	5.8	7.1	50.2	40.5
St. Dev.	1.52	1.93	30.03	16.63

p (scores, $\gamma = 21$) = **0.0288**
p (times, $\gamma = 18$) = **0.1412**
(times are in minutes)

TABLE 1: GROUP ANALYSIS RESULTS

Figure 5 below summarizes the results of the individual analysis. All but two subjects remained the same or improved scores, with a slightly higher number of subjects staying the same than improving (see Figure 5a). Thus the quality of solutions, on average, did not worsen due to the new tool. Figure 5b shows that a majority of subjects improved completion time by more than 10%, despite the statistically insignificant difference in mean completion times for the class. The combined results show that the vast majority of the subjects improved in either one or both score and time, without worsening the other (see Figure 5c). Two subjects performed worse on the second HW than on the first. Four subjects were indeterminate, with offsetting scores and times. Overall, these results seem to support the hypothesis that I/O mapping helped improve quality and efficiency in the problem solving process.

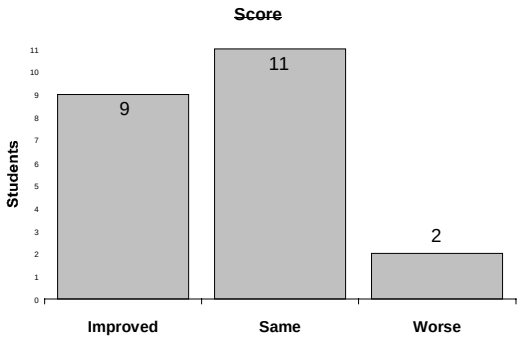


Figure 5a Differences in Score between using IR or not

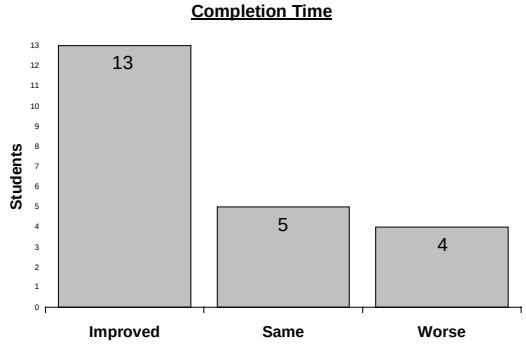


Figure 5b Differences in Time between using IR or not

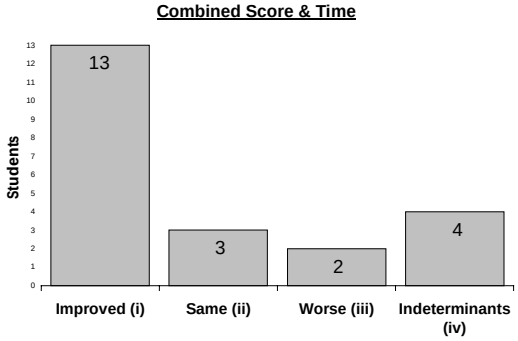


Figure 5c Combined differences in Score and Time

Figure 5c key:
(i) Improved score and/or time, without decreasing other
(ii) Same score and completion time
(iii) Worse score and/or time, without changing other
(iv) Offsetting score and completion time

FIGURE 5: SUMMARY OF INDIVIDUAL ANALYSIS RESULTS

5. Discussion

The experimental results strongly suggest that the I/O mapping technique helped students improve the quality and/or efficiency in solving ladder logic problems. The majority of subjects improved in both score and time, or in one of the two without worsening the other, as presented in Figure 5c.

The results, however, must be interpreted in light of several factors. First, self-reported times are often only an approximation, with varying accuracy among subjects. Also, students knew they were participating in a study, and that the I/O map was intended to help them with the particular problem, which may have introduced bias into the reporting of completion times.

Second, the score categories used (S+, S, and S-) may lack resolution for statistical analysis. Students were deemed to have "improved" or "worsened" only if they changed categories. A number of students improved the quality of their solutions, but not enough to change categories. In addition, the subjectivity in the score evaluations leaves room for bias or inconsistency on the part of the evaluator. We attempted to minimize these noise factors by basing the scores on very specific aspects of the problem.

Third is a comparability issue. Are the different homework problems comparable? We took pains to select comparable problems, considering the students' proficiency level in the course. The number of variables involved, and number of rungs required, and logical complexity are comparable. But these were subjective evaluations, and what the instructor predicts will be challenging for students may in fact be quite easy for them, and vice versa. The reason that students on average scored higher and spent less time on the problems could simply reflect that the later problem was easier for the students than the earlier problem. Asking students for their assessment of difficulty is problematic as the tool is designed to make problem solving easier, so in fact students may perceive the problem to be easier than they might have otherwise--we wouldn't really know. Or, some students may have found the problem more difficult because they were simultaneously learning the new method. Thus, we did not attempt to measure problem difficulty through student assessment.

An obvious solution to the comparability issue is to set up a more controlled experiment, where a control group solves a problem using any method they wish, and an experimental group solves the same problem using the I/O map method. But in a classroom setting, this is nearly impossible to achieve (how do you keep the experimental group from 'contaminating' the control group?), and is ethically dubious. Having the two groups in different sections helps mildly, but does not completely alleviate the problem as students in different sections will talk and work together, and the ethics question remains. Separating the two groups by time, such as having the control group in one semester and experimental group in the next, is problematic because one would want to use the same homework problems, and students may share solutions. One way to completely alleviate the issue is to conduct an experiment outside the classroom. But this has its own set of issues, such as funding and time availability. The current experimental design takes into account these significant constraints imposed by the classroom environment.

Lastly, the results may simply reflect that learning occurred between homework assignments. Students may have learned something in solving the first problem that helped them solve the second. We tried to minimize the opportunity for learning between the homework assignments by keeping a short time period between homework problems—the two homework problems were completed in 7 days. Students did not receive the first homework problem back graded until after turning in the second. And, we used different problems similar in difficulty in the homework sequences to minimize the effects of knowledge accumulation.

The limited scope of the experiment, and the possibility for noise and bias in the data, likely mean that the results are not conclusive. However, the results triangulate well with instructor observation and student feedback, and thus provide reasonably sound evidence for the effectiveness of the solution approach.

6. Explaining Why the Technique Helps

What explains the apparent effectiveness of the I/O mapping technique? First, reformulating the problem statement into a different form forces the student to think about the problem statement more carefully than s/he might otherwise.[6] Simply taking in the information as presented and trying to process it is difficult. A tool like the I/O map helps the student identify the different system elements through the physical act of drawing each element. Having to choose a physical

location on the diagram for each element initiates the understanding interactions between elements. The direction of the arrows helps identify inputs versus outputs.

Secondly, the I/O map lends itself to breaking the problem into sub-problems that can be solved somewhat independently. In larger problems than used in this illustration, ones with multiple outputs (and thus solutions with multiple rungs), each output has its own rung and the logic for each rung can often be determined independent of the rest of the system.

Finally, the I/O map presents the problem in way that is logically consistent with ladder diagrams. My experience is that flowcharting or sequence lists, while effective for many code-based programming languages, is NOT effective for ladder diagrams. In fact, I've found that flowcharting actually makes the problem more difficult to solve. The reason for this is that the PLC microcomputer executes ladder programs in a simultaneous fashion. It scans all the inputs simultaneously, then updates all outputs simultaneously. A flowchart by its very nature is a sequential representation. The I/O map, on the other hand, is a synchronous representation and quite easily converts to logic statements of the simultaneous nature required for ladder programs. (Further theoretical argument may be found in a companion paper.[7])

7. Conclusion

This paper has described solution approach to ladder programming that aids the programmer in bridging the gap between problem and ladder program solution. It represents one way for the programmer to analyze the process control problem and structure it for a solution in ladder logic. Its effectiveness as a tool was verified empirically through an in-class experiment. Although the results are not 100% conclusive, they do provide strong evidence in support of its adoption.

Bibliography

1. Groover, M. P., Automation, *Production Systems, and Computer-Integrated Manufacturing*, 2nd edition, Prentice Hall, Upper Saddle River, NJ, 2001; pp. 264-277.

2. Webb, J. W. and R. A. Reis, *Programmable Logic Controllers: Principles and Applications*, 4th edition, Prentice Hall, Upper Saddle River, NJ, 1999.

3. Chang, T-C., R. A. Wysk, and H-P. Wang, *Computer-Aided Manufacturing*, 2nd edition, Prentice-Hall, Upper Saddle River, NJ, 1998; pp. 235-272.

4. Kissell, T. A., *Industrial Electronics*, 2nd edition, Prentice-Hall, Upper Saddle River, NJ, 2000; pp. 71-126.

5. Sobek, II, D. K., "The Role of Intermediate Representations in Engineering Problem Solving," Proceedings of the ASEE 62nd Annual Pacific Northwest Section Meeting, April 27-29, 2000.

6. Sobek, II, D. K., V. A. Cundy, and V. L. Briggeman, "Assessing the Given-Find-Solution Method in an Undergraduate Thermodynamics Course," *International Journal of Mechanical Engineering Education* (under review).

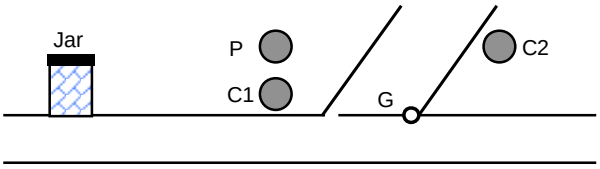
7. Sobek, II, D. K., "Understanding the Importance of Intermediate Representations in Engineering Problem-Solving," submitted to the 2001 ASEE Conference Proceedings.

DURWARD K. SOBEK, II
Durward Sobek is currently Assistant Professor of Industrial and Management Engineering at Montana State University. He holds a B.A. degree in Engineering Science from Dartmouth College, and the M.S. and Ph.D. degrees in Industrial and Operations Engineering from The University of Michigan. His current research interests are in the areas of new product development and engineering design education. Please visit his web site for additional information: http://www.coe.montana.edu/ie/faculty/sobek.

Appendix

Homework Problem 1

Glass jars with metal lids come down a conveyer to a test station consisting of a photoelectric cell (C1) and a proximity switch (P). The photocell outputs a high signal until something breaks the light beam. The proximity switch outputs a high signal only when a metal object enters the sensor's field. Thus these two sensors can distinguish whether no jar is present, a jar with a lid is present, or a jar without a lid is present.



If a jar without a lid is detected, the controller activates a gate (G) that diverts the lidless jar off the conveyor to a defect area. Photocell C2 signals that the defective jar has cleared the exit conveyer. No jar will activate C1 and C2 at the same time.

Incoming jars are sufficiently spaced so that a new jar will not reach the C1/P sensor field before a lidless jar has reached C2. However, your program should account for this contingency by stopping the conveyor if a "good" jar is detected before the lidless jar has reached C2 and the gate is open (the conveyor should run continuously otherwise). The conveyor resumes operation once the lidless jar passes C2.

A momentary push button (P1) starts the conveyer, and a second momentary push button (P2) stops the conveyer. Activating G opens the gate (part diverted), deactivating G closes it (part passes through).

Write a ladder program to implement this control sequence.

Homework Problem 2

A food processing company prepares a starch solution in a mixer as shown below. A N.O. push button (X1) starts the process. When pressed, the controller opens valve V1 to fill the tank with water until the water level reaches LS2. Then it closes V1 and opens V2 to add a starch slurry to the water until the mixture reaches LS3. When LS3 is enabled, V2 closes. Then a mixing motor (M) is turned on.

After two minutes, the controller begins to sample the viscosity of the mixture. The viscosity probe has two outputs: $n1$ is true if the mixture is too thick, $n2$ is true if the mixture is too thin. If $n1$ is true, then more water should be added. If $n2$ is true, then more starch should be added. If both $n1$ and $n2$ are false, the viscosity is within the acceptable range.

When the mixture is in the acceptable range for viscosity, the mixing motor is turned off and exit valve V3 is opened to drain the contents of the tank. The tank is considered empty when the water level falls below LS1.

a) Create a map of which elements control the system outputs (a graphic or tabulation, like we did in class).

b) Create a ladder program to control this process.