Teaching Finite State Machines (FSMs) as Part of a Programmable Logic Control (PLC) Course

Dr. Curtis Cohenour Ph.D., P.E. P.E., Ohio University

Dr. Cohenour is an Assistant Professor in the Ohio University Engineering Technology and Management Department, in Athens, Ohio. He received a Bachelor of Science degree from West Virginia Institute of Technology in 1980, a Master of Science degree from Ohio University in 1988, and a Ph. D. in Electrical Engineering from Ohio University in 2009. He is a registered professional engineer in West Virginia, and Ohio.

Dr. Cohenour has worked in Industry as an electrical engineer and project manager. He joined Ohio University in 2002 as a research engineer working for the Ohio University Avionics Engineering Center. He has worked on projects covering a wide variety of avionics and navigation systems such as, the Instrument Landing System (ILS), Microwave Landing System (MLS), Distance Measuring Equipment (DME), LAAS, WAAS, and GPS.

His recent work has included research with the Air Force Research Laboratory in Dayton, Ohio, aimed at understanding and correcting image geo-registration errors from a number of airborne platforms.

Teaching Finite State Machines (FSMs) as Part of a Programmable Logic Control (PLC) Course

Abstract:
Industrial control courses in Electrical and Computer Engineering (ECE) include programming of Programmable Logic Controllers (PLCs). Text books are available to support these courses but few provide any content on Finite State Machines (FSMs). This is unfortunate because a great deal of PLC applications in industry involve sequence logic which lends itself to the FSM.

A FSM consists of a defined (finite) number of states that a system can be in, and well defined rules for how the system moves from one state to the next. The FSM can perform entry chores, exit chores, and state tasks. Inputs and calculations made within the state define the movement from state to state. The action that results from an input may depend on the current state. A PLC is a computer that solves relay logic also referred to as ladder logic. PLCs are typically connected to limit switches and solenoid valves and provide low level control functions. Supervisor computer systems may also direct the action of a PLC.

The course content described in this paper offers the author's industry tested implementation of the FSM for PLC implementation. The FSM connects the PLC input devices to the PLC output devices in a straight forward manner that makes the system easier to understand for engineers and maintenance personnel.

Students learn not just how to program a PLC, but how to create ladder logic that provides a reliable and maintainable PLC solution. PLCs are ubiquitous in industry and students with experience and ability are in high demand.

Motivation:
Electrical Engineering (EE) and Engineering Management and Technology (ETM) students can increase their employability with Programmable Logic Controller (PLC) skills. PLC positions exist in supervision, maintenance, operations, and sales. Supervisory positons can be in maintenance or operations. These may include supervisory responsibility over processes or personnel involved production or maintenance. Maintenance positions include troubleshooting PLC systems and maintaining PLCs and Input Output (IO) devices. Operations may include using PLC controlled equipment in production. Sales may include the sale of PLC controlled equipment, or the sale of PLC processors, IO cards, and accessories.

To meet these needs an undergraduate senior elective in PLCs is offered. The course has long been offered without any specific training in sequences. In the fall of 2015 the course was modified to include Finite State Machines (FSMs). With this change students are now trained to recognize the difference between combinational logic and state logic. Combinational logic is logic that is derived exclusively from current inputs and does not depend on previous inputs. State logic includes memory of previous events specifically embodied in the "states".

Additional learning outcomes for the modified course are:

1. Recognize the difference between combinational and state logic
2. Recognize a sequence
3. Create a state diagram
4. Implement an FSM in Ladder Logic

5. Create outputs from FSM states
6. Test and debug an FSM on a laboratory system

These learning outcomes are assessed via assignments, exams, and laboratory exercises.

Background:
Ladder Logic (LL) or Ladder diagram (LD) is one of five programming languages defined in IEC 61131-3 [1]. The defining feature of IEC 61131-3 is its cyclic execution model. In addition the IEC 61499 [2] provides an event driven execution model. These standards and conversion to IEC 61499 are described in [3] [4].

Combinational logic, also known as time-independent logic, is PLC logic that depends only on the current inputs. Because the ladder logic is solved cyclically, per IEC 61131-3, combinational logic can be defined as logic that depends only on logic that is solved earlier in the cycle. If the logic depends on information from a previous cycle (memory), it is state logic. Memory from previous states could be numeric, but for the purposes of this paper, a state is defined as any coil where the value from a previous scan is used in the current scan.

Our focus here is on the implementation of the FSM in LL. Alternatives exist such as the Sequential Function Chart (SFC), also contained in IEC 61131-3 [1]. The SFC is not LL and may be unfamiliar to maintenance personnel in some organization. In addition, there are sequencers and drum controllers available in some PLCs, but again these may not be suitable for some organizations.

A review of undergraduate level PLC text books was conducted to determine if there were any texts that dealt with FSMs in PLCs.

The closest match is found in [3], were sequential applications are addressed using latching relays in chapter 6, this is repeated using Allen-Bradley PLCs in chapter 6 of [4]. SFC programming and sequence logic using control relays is discussed in [6]. Sequential Function Chart programming is discussed in [8]. The Allen Bradley “Sequencer” instruction is discussed in [8]. There is no mention of sequences in [5]. The text used at our institution is [9]. In this text SFCs are mentioned in passing. Short sequences using control relays are covered with no overt explanation that the control relays define states, or that there is a sequence of events.

FSM Implementation:
PLC LL programs are often filled with state machines. Every seal in circuit is a two state FSM. What is required of the student is to recognize the difference between combinational logic, and state logic, and to perform the implementation accordingly.

Combinational logic is the direct assignment of an output or internal coil based on inputs with no memory. The inputs may be connected to real world devices such as limit switches and push buttons or internal coils. The outputs may be real world outputs or internal coils. As an example assume that there are two pushbuttons and a limit switch that control a solenoid valve. There are three inputs that can be on or off for a total of eight combinations. Of the eight combinations some of these combinations turn

the solenoid valve on and some turn it off. In combinational logic three binary inputs yield 8 possibilities which are assigned to the output. The previous values of the inputs or the outputs are not considered.

State logic is any logic that is dependent on previous events. The most common example is a seal in circuit. Figure 1 shows a combinational circuit in Rung 1 and a two state FSM in Rung two. In rung one there are three inputs and one output. The state of the output is dependent only on the state of the three inputs. In Rung two there are only two inputs and one output. The output Y008 is used in conjunction with the two inputs X008 and X009 to determine the output. The output Y008 depends not only on the inputs but also on the previous value of the output. Y008 is a state.

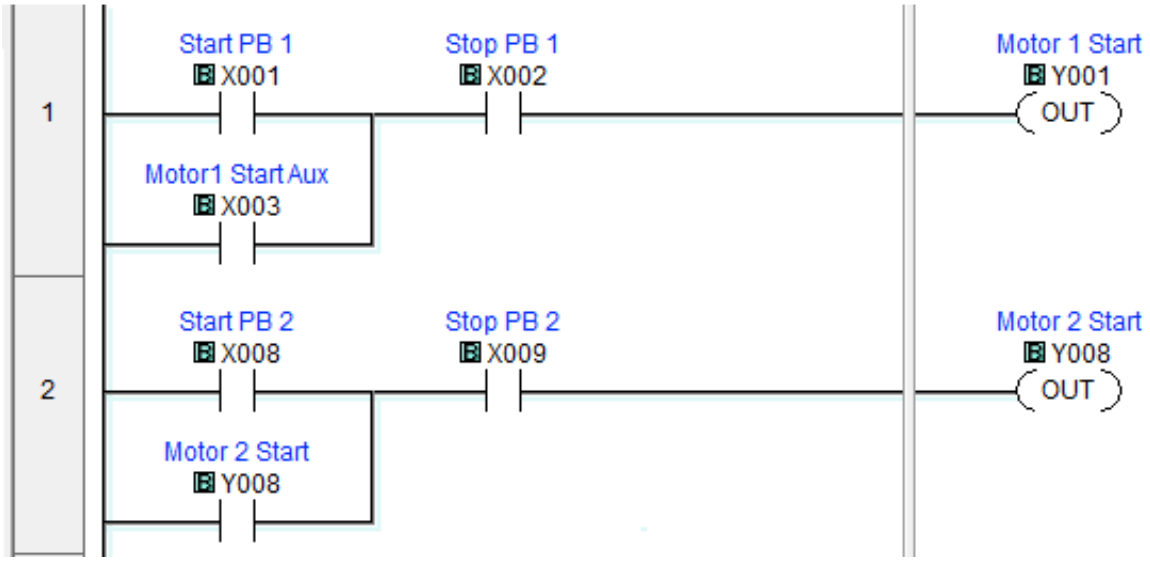

Figure 1Two similar motor start circuits implemented in PLC logic. Rung one (top) is combinational logic. Rung two (bottom) is a two state FSM. Note that in rung 2 the output Y008 is dependent on the previous value, or state of Y008.

The motor start circuit in Rung one is preferred because if the motor quits for any reason, the seal X003 will be lost, and the motor will not restart. This is a basic safety feature of the circuit. The circuit in Rung one is also preferred because it is combinational.

Rung two is a state machine with two states. The two states are on and off based on the condition of Y008. The PLC logic of Rung two might be used in a situation where an unplanned start would not create a safety or process risk, or where an auxiliary contact for the motor is unavailable.

An FSM Example

Consider the box fill operation, shown in Figure 2. Boxes move along the conveyor to a fill station where they stop and are filled. After the box is filled, the conveyor is started to remove the full box and bring an empty box to the fill positon. There is a level switch to detect the full condition, and a proximity to indicate that the box is in position. The system can be in auto or standby. If the system is in auto the conveyor runs until a box is detected by the proximity switch. When the box is detected the conveyor stops, the fill valve is energized and material flows from the hopper into the box. The box full condition is indicated by the level switch. When the box is full, the fill valve is closed and the conveyor starts and

runs until the box is clears the proximity switch. The conveyor continues to run until an empty box is detected by the proximity. The cycle repeats as long as the system is in auto.

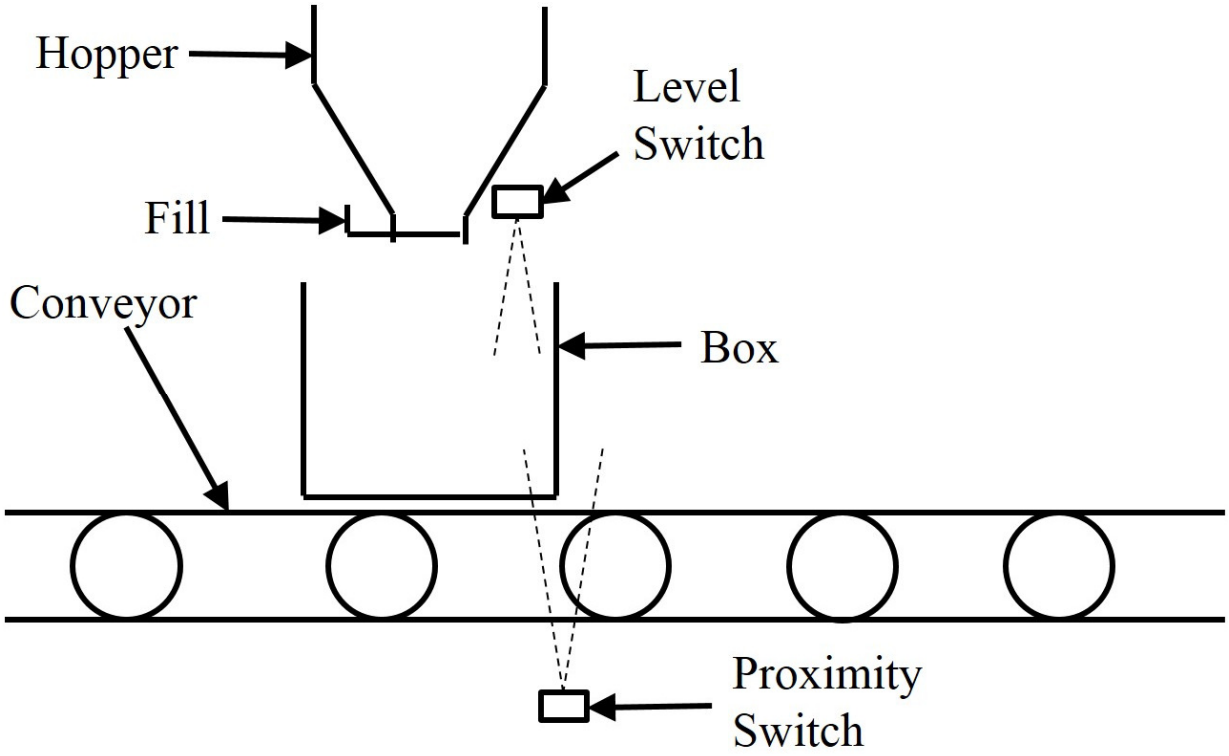

Figure 2 A box fill operation.

A state machine diagram is given in Figure 3. There are three states. The author's practice is to number the states by tens to allow for modifications. In state 10 the conveyor is running and we are looking for an empty box as indicated by the proximity switch. In state 20 the box is filled until the level switch is made. In state 30 the conveyor runs until the proximity switch clears indicating the full box is gone. The transition from standby is made to state 30 so that if there is a box on the conveyor it is removed before the fill operation begins.

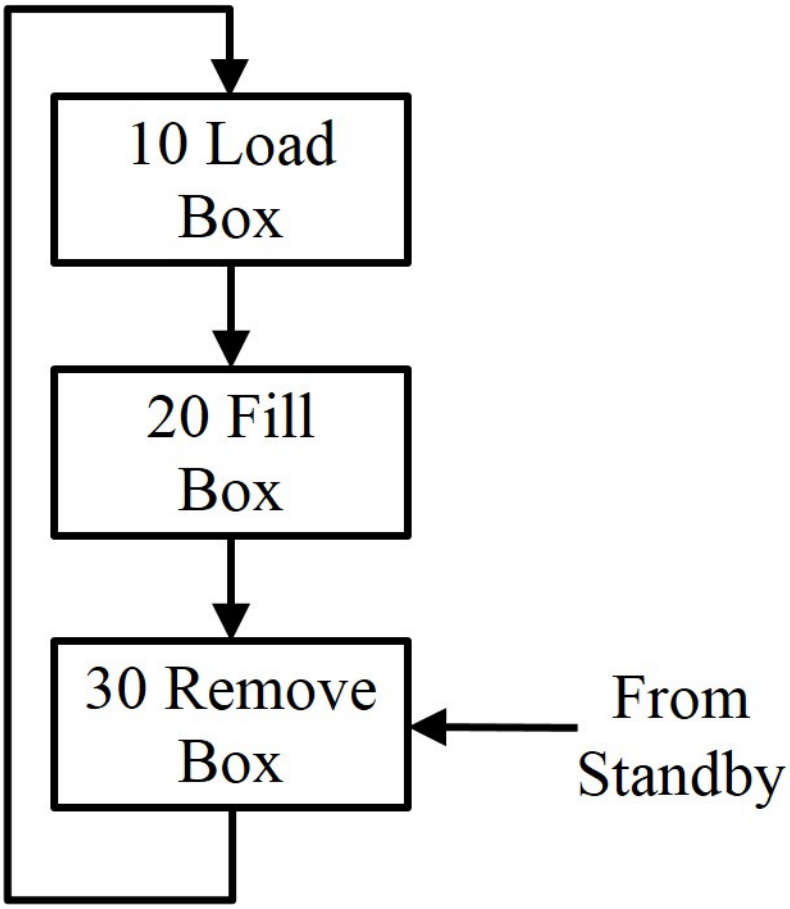

Figure 3 A state diagram for a three state FSM for a box fill operation

The system can be in auto or standby, and for safety includes a Normally Open (NO) contact from an external estop relay. The auto standby circuit is operated by an NO Pushbutton (PB) for auto start, and a Normally Closed (NC) PB for stop. This two state auto/standby FSM is shown in rung one of Figure 4. Note that rung one is a two state FSM and rungs two through four are a separate three state FSM. The outputs on rungs five and six are combinational logic.

Coil C101 is a state because it is dependent on the previous value of C101, C102, and C103. Coil C102 is a state because it depends on the previous scan value of C102, and C103. In Rung four the new values from the current scan are available for C101, and C102, but the previous scan value for C103 is used making C103 a state.

One might argue that because coils C101, C102, and C103 are states that the outputs Y001, and Y002 are also states. This is not the case. The outputs are dependent on the values of the state coils computed on the current cycle, not on the previous cycle. Thus the outputs are combinational, not states.

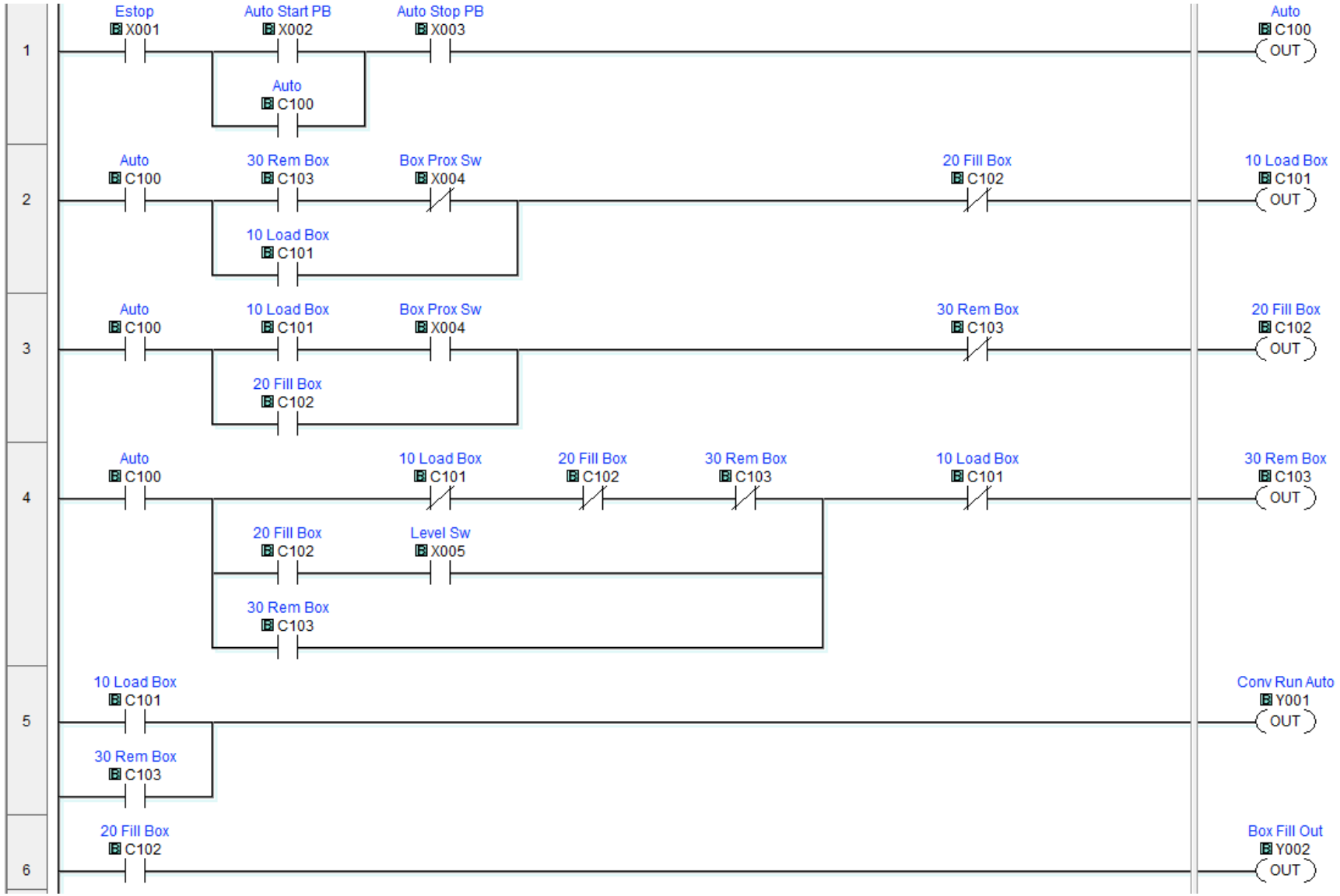

Figure 4 A three state FSM controls a box fill operation.

Rung two is the load box state, state 10. From Figure 3 we see that we enter state 10 from state 30, and we exit to state 20. From the description we know that in state 30 the conveyor runs until the box clears. Thus in rung two we have a permissive, auto C100, followed by the from-state C103 and the entry condition. State 10 is sealed in by C101. The load state exits when the fill state starts due to the NC contact from C102.

Each rung of the FSM consists of a permissive, an entry condition, a seal in, and an exit condition. The entry condition normally consists of a from-state and some additional entry logic. The logic for a generic state can be given in English as: If the permissive, and the entry condition or the seal, and it is not the exit condition turn the state on.

There may be more than one path to enter a state. If this is the case the entry conditions are ORed by placing them in parallel paths. This can be seen in the generic state shown in Figure 5.

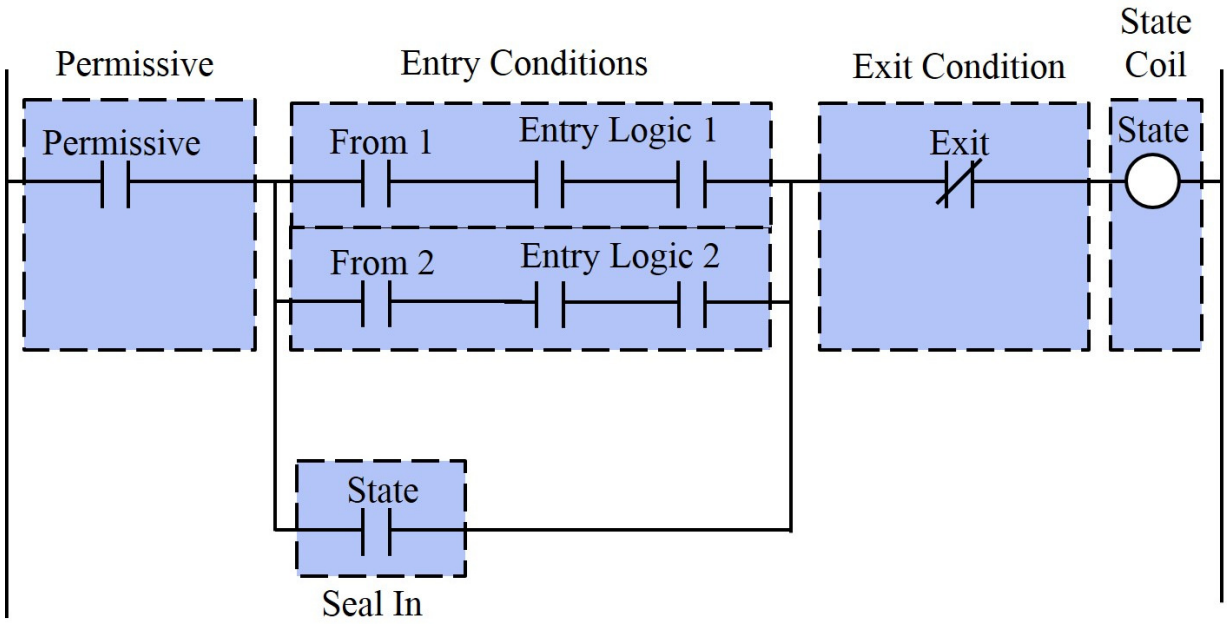

Figure 5 PLC Ladder Logic for a generic state

The generic state has five parts. The permissive includes anything that is required for the FSM to be operational. In the example above it is just the auto coil which contains the estop. In practice one might include both auto and the estop in the permissive as a reminder that the estop will stop the sequence. The estop might also be included in the outputs even though it is redundant as a reminder that the safety circuit is active.

The entry conditions are placed in parallel paths as shown in Figure 5. Each entry condition has a from-state, and some entry logic. Each entry condition comes from an arrow pointing into the state in the state diagram, see Figure 3. There is one parallel row for each entry into the state. The entry conditions are sealed by the state coil.

The state is exited when a new state is activated. A NC contact for each exit state is placed in series with the permissive and the seal in. There is one NC exit contact for each outgoing arrow in the state diagram.

An FSM can have entry chores, exit chores, state chores, etc. In Figure 4 the state chores are the outputs Y001, and Y002. There are no entry or exit chores but if there were they could be activated using transitional contacts. The generic form of Figure 5 may not cover every aspects of a traditional FSM but it does cover all of the items necessary for a PLC sequence.

A complete description of states 20, 30, and the outputs in Figure 4 follows. State 20 is the box fill state. If the box is loading and the proximity switch is activated the state 20 coil, C102 energizes. This drops out the C101 coil, and C102 seals in the fill state. The fill state is complete when state 30 is energized.

In state 30 there are two entry conditions. The first condition has no from-state, but it does have entry logic. If the system is in auto and there is no active state coil C103 is energized, and seals in. Thus on the transition to auto the conveyor starts and runs until the proximity clears. The second path into state 30 is from-state 20. When in state 20 and the level switch activates the box is full and the FSM moves to state 30. The exit condition for state 30 is state 20.

Finally the outputs Y001 conveyor run in auto, and Y002 box fill are functions of the states. The conveyor runs in state 10 load box, and state 30 remove box. The box fill output Y002 is energized when in state 20. The logic for the outputs is combinational.

The logic of Figure 4 represents a sub-process. A typical PLC application might consists of dozens of small sub-processes like this. When combined they can control a complete process or multiple process. Each sub-process consists of three components, entry logic which is combinational, the FSM, and the output logic which is also combinational. This format promotes robust design, programming, and maintenance of the PLC system.

For a sub-process the entry logic might consist of combinations of permissive, input devices, and contacts from other sub-processes. The FSM contains the state logic. The output logic is combinational and generates internal coils which go to other sub-processes, and external coils which go to solenoid valves, motor starters, and other external devices.

Results

The PLC class consists of lecture and laboratory. The lecture portion of the class covers the fundamentals of PLCs including hardware, logic, number systems, math functions, timers, and counters. The laboratory portion allows the student use these items on a PLC trainer. There are only two lab exercises that deal specifically with FSM. The first is the box fill operation described above and the second is the final project.

The PLC class includes a final project using a Bytronic Fluid Process Rig (FPR) shown in Figure 6. Students are given a process description and an IO list and directed to program the system using a Click PLC. Classes prior to spring semester 2015 were given no specific instruction in combinational vs. state logic.

Students in the spring 2015 class were given the additional material contained above. The results were compared to see if the FSM improved the readability, program length, and reliability of the code.

Students with the FSM training were able to distinguish between combinational and state logic. Their ladder logic consisted of combinational logic in the rungs preceding the FSM, and combinational after the FSM to combine the automatic and manual operation of the process. The students generated a state diagram and programmed the FSM in the manner described in this paper.

Figure 6 Bytronic "Fluid Process Rig" used as a final project for the PLC class.

The process in Figure 6 is a simple 3 state FSM. State 10 fills the Process tank. State 20 heats the fluid. State 30 empties the tank. There is also a cooling fan, pump, mixer, and few solenoid valves.

Students with the FSM training were able to quickly develop the necessary state logic to control the system. Most students had five states including the 3 above for the FSM and one additional state each for the mixer, and the cooling fan. The average program length was 20 rungs. Lab time for students to compete the project was two to four hours.

Students prior to spring 2015 used a variety of methods to solve this problem including drum controllers and control relays. The programs were longer 25 to 30 rungs, less organized, harder to understand (subjectively). In addition, the students took 4 to 8 hours of lab time to complete the lab and experienced higher levels of frustration.

Not every student works with PLCs after graduation but for those who do understanding the difference between combinational logic and state logic increases their productivity. Generating well thought out and implemented PLC code improves operation, maintenance and safety.

Bibliography

[1] International Electrotechnical Commission, "Programmable controllers - Part 3: Programming languages," International Electrotechnical Commission, 2013.

[2] International Electrotechnical Commission, "Function Blocks, International Standard IEC 61499,," International Electrotechnical Commission, 2012.

[3] W. Dai and V. Vyatkin, "Redesign Distributed PLC Control Systems Using IEC 61499 Function Blocks," *IEEE Transactions on Automation Science and Engineering,* vol. 9, no. 2, pp. 390-401, 2012.

[4] W. Dai and V. Vyatkin, "Redesign Distributed IEC 61131-3 PLC System in IEC 61499 Function Blocks," in *010 IEEE 15th Conference on Emerging Technologies & Factory Automation (ETFA 2010)* , Bilbao, Spain, 2010.

[5] K. T. Erickson, Programmable Logic Controllers: An Emphasis on Design and Application Second Edition, Rolla, MO: Dogwood Valley Press, LLC, 2011.

[6] K. T. Erickson, Allen-Bradley PLCs: An Emphasis on Design and Application, Rolla, MO: Dogwood Valley Press, LLC, 2013.

[7] G. A. Mazure and W. J. Weindorf, Introduction to Programmable Logic Controllers, Second Edition, Orland Park: American Technical Publishers, 2011.

[8] L. A. Bryan and E. A. Bryan, Programmable Controllers Theory and Implementaion, Second Edition, Marietta: American Technical Publishers, 1988.

[9] J. A. REhg and G. J. Sartori, Programmable Logic Controllers, Second Edition, Upper Saddle River: Pearson Education, Inc, 2009.

[10] M. Rabiee, Programmable Logic Controllers, Tinley Park: Goodheat-Willcox Company, Inc., 2013.

[11] F. D. Petruzella, Programmable Logic Controllers, Fourth Edition, New York: McGraw-Hill, 2005.