AC 2007-838: A NEW APPROACH TO TEACHING PROGRAMABLE LOGIC CONTROLLER PROGRAMMING

Donald Harby, UCM

Donald Harby is an associate professor of Engineering Technology at University of Central Missouri. His research interests include automation, machine tool design and control, and technical education. He has over 15 years of industrial manufacturing and automation experience. He received his BS from Parks College of St. Louis University in 1991, and his MS from the University of Missouri-Columbia in 2002, and expects his PhD from the University of Missouri-Colubmia in 2007.

Patricia Polastri, UCM

PATRICIA POLASTRI is a Ph.D. fellow at Indiana State University, and an instructor at the University of Central Missouri. She has several years experience in the automobile manufacturing industry in Europe. She received her BS from University of Orebro in 1998, and her MS degree from the University of Central Missouri in 2005.

Chakapong Chuenprasertsuk, UCM

Chakapong Chuenprasertsuk is an international graduate student at University of Central Missouri. He is pursuing M.S. Industrial Technology while working as a Graduate Assistant in the School of Technology. He graduated from Kasetsart University, Thailand with B.Eng. Electromechanic Manufacturing Engineering in 2005. His undergraduate thesis was the Design and Implementation of Biomechanical Force Plate. He has two internship experiences at University of Wollongong, Australia and PepsiCo (FritoLay), Thailand.

A NEW APPROACH TO TEACHING PROGRAMMABLE LOGIC CONTROLLER PROGRAMMING

ABSTRACT

The Programmable Logic Controller (PLC) is used widely in industries to control processes and manufacturing systems. There are several analytical approaches for developing PLC programs which are state diagrams, Petri Network and diagrams, truth tables, and Boolean algebra. All of these approaches are useful in developing a sequence of controlling PLC programs in complex systems, but none of these lead to a full automated way of developing a program. In the traditional way of teaching PLC programming such as the state diagram, a diagram is first constructed showing all the possible paths the process can take. Then the Boolean conditions necessary for each path are added. The diagram is then converted to a PLC program and tested. Changes are made to the Boolean conditions and then the new program is tested again. This process of trial and error is continued until a bug free program is developed. This trial and error step does not lead to an automated program development technique. This paper proposes a new approach to controller design and sequence control programming to be used by teachers and industrial trainers. This new method would be a useful tool for teaching beginners a faster way of developing a PLC program. This new approach combines some of the key concepts from various techniques to deliver a novel but effective automated method. This new technique translates in two easy steps truth tables into Ladder Logic Diagram, which is the most common graphical language for PLC programming. An example of a typical hydraulic station was used to demonstrate the analytical method. This example shows that this new analytical and systematic approach can be used to develop the sequence control program of complex systems. This new teaching technique has been applied to engineering technology students. A research study was conducted to verify the usefulness of this method. The study separated engineering technology students in two groups. One group was taught PLC programming in the traditional way, while the other was taught using the new method. Using the same test, the results showed that the group exposed to the new technique spent less time programming, when compared the other group.

INTRODUCTION

In recent years many approaches to developing a systematic or analytical method for PLC programming have been investigated. These approaches include various types of programming, such as state diagrams, Truth Tables, Petri networks and diagrams and Boolean algebra[2]. All these methods are unquestionably useful in the development of PLC programs, but none has led to a fully automated way of developing a program. The project here presented, combines some of the key concepts from the above mentioned techniques to produce a unique and effective automated method.

The state diagram approach is one of the most used methods[2], showing the flow diagram for sequential processes. First a diagram is constructed, showing all possible paths the process can take; and then Boolean conditions are added for each present path. The diagram is then easily converted to a PLC program and tested. Changes are made to the Boolean conditions and then the new program is tested. This process of trial and error continues until a bug free program is

developed. This trial and error method is time consuming and does not lead to an automated program development technique. New PLC trends aim to increase communication capability, reduce size and improve the current software[8]. This must be compatible with the working environment of PLCs, which must be durable and designed for operations in industrial environments[2].

Many PLC programs will not be sequential in nature but will require several processes to execute simultaneously. State diagrams are not efficient when the system requires parallel processing. Petri networks were first developed in computer programming as a structured way of dealing with parallel processes; in PLC programming a Petri diagram is sometimes used to help with the development of parallel branching and simultaneous processing problems. One important concept that Petri diagrams introduce is that of tokens. Tokens are passed to and from the main program and the parallel branches. The program will not continue to execute at a point of convergence until all the tokens are collected. Just like the state diagram approach Petri networks and diagrams require some trial and error process to produce a usable PLC program.

The approach here presented introduces a technique that translates in two easy steps truth tables into Ladder Logic Diagram. This is the most common graphical language for PLC programming. The approach was tested with engineering technology students and showed positive results. It could be proven that the method discussed aided in the learning of PLC programming, making it easier and faster to understand.

THE RESEARCH METHOD

In order to create fully automate the design process, a method must be developed which completely eliminates the trial and error process. To eliminate the trial and error or programming as an art, a clear and analytical relationship must be developed between the design variables and performance variables[9]. Truth tables and Boolean algebra have been used for many years to automate the design of digital electronic circuits. However, the techniques used in circuit design do not lead directly to an automated approach for PLC programs. In general, a truth table will be developed for a given circuit then every combination of that truth table will be considered. In a PLC we may not care about every combination or some combinations may not be physically possible. Also, in the electronic circuits only the low (off) or high (on) states of the outputs are of concern, depending on whether or not it is active high or active low. But in the PLC both the high and low states of the outputs must be considered at all times.

In conducting this research, the direction control of hydraulic cylinders (as shown in Fig.1) have been used to demonstrate the PLC program created by 17 engineering technology students currently enrolled at the University of Central Missouri.

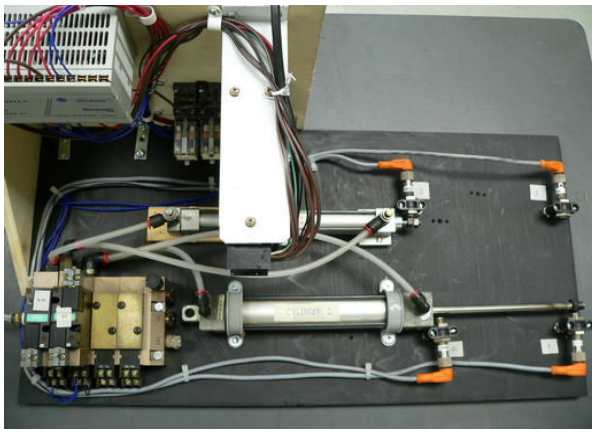

FIGURE 1. TYPICAL DIRECTION CONTROL OF HYDRAULIC CYLINDER

The following is a list of steps required to first generate a truth table, then Boolean equations from the high and low outputs, and finally a method to convert the equation to a PLC program.

1. Write a list of steps that the program must complete.
2. List, in truth table form, all of the systems physical inputs and outputs.
3. In the first input row, enter the input conditions required for the first step.
4. In the first output row enter the output condition required to make the system complete the first step.
5. In the second input row, enter the states that the inputs will be in after the previous step is finished.
6. In the second output row enter the output condition required to make the system complete the next step.
7. Repeat Steps 5 and 6 until all the steps in the program are complete.
8. Verify that all of the input and output rows in the truth table are unique. If so, go to Step 9. If not for each duplicate row add one internal relay (register).
9. For each output create two Boolean equations. One for the low states and one for the high states.
10. Create the PLC program using the high equations to latch on the outputs and the low equations to unlatch the outputs.

Note: Timers and Counters are treated just like physical I/Os.

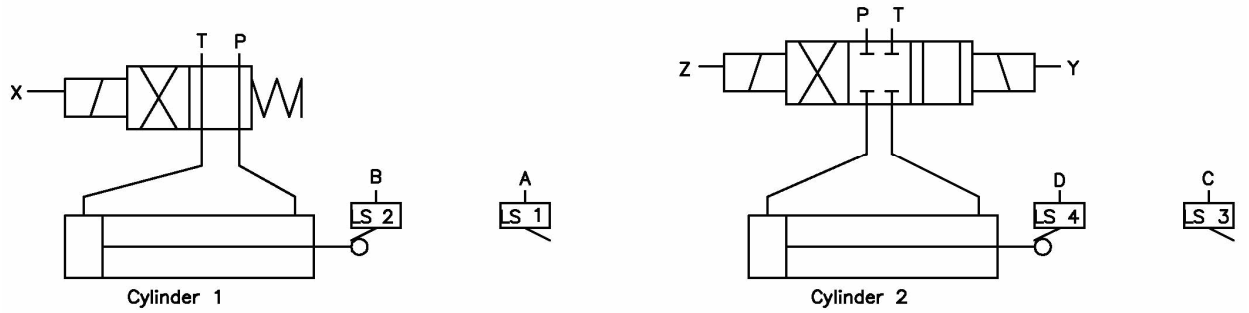

FIGURE 2. DIAGRAM OF THE RESEARCH TOOL USED IN THIS PROJECT

To demonstrate this method a simple typical hydraulic system is shown in Fig.1. Assume this is one station of a machine and for the process Cylinder 1 needs to fully extend then retract. Also, assume there is an input E for cycle start (receive token) and an input W for an emergency stop.

Input/Output Symbols	I/O on Allen-Bradley MicroLogix™ 1000	Definition
A	I:0/1	Limit switch 1 (Cylinder 1 extend)
B	I:0/2	Limit switch 2 (Cylinder 1 retract)
C	I:0/3	Limit switch 3 (Cylinder 2 extend)
D	I:0/4	Limit switch 4 (Cylinder 2 retract)
E	I:0/5	Start cycle switch (Receive token)
F	B3:0/1	Internal relay
W	I:0/6	Emergency Stop
X	O:0/0	Activate cylinder 1 (Extend)
Y	O:0/1	Activate cylinder 2 (Extend)
Z	O:0/2	Activate cylinder 2 (Retract)

TABLE 1. DEFINITIONS OF I/O USED IN THE PROJECT

STEP 1: LIST THE PROGRAM STEPS.

1. Cycle Start input (E) is activated
2. Extend cylinder 1
3. Retract cylinder 1
4. Extend cylinder 2
5. Retract cylinder 2
6. Repeat step 2-5 until E-stop is activated (input W)

STEP 2: LIST I/O IN TRUTH TABLE FORM.

Input					Output		
A	B	C	D	E	X	Y	Z

STEP 3: LIST THE INPUT STATES REQUIRED FOR THE PROGRAM TO START.

Input					Output		
A	B	C	D	E	X	Y	Z
0	1	0	1	1			

STEP 4: List the output states for the first step

Input					Output		
A	B	C	D	E	X	Y	Z
0	1	0	1	1	1	0	0

STEP 5: ADD ANOTHER ROW TO THE TRUTH TABLE AND ADD INPUT STATES THAT WILL RESULT FROM THE PREVIOUS OUTPUT STATES.

Input					Output		
A	B	C	D	E	X	Y	Z
0	1	0	1	1	1	0	0
1	0	0	1	1			

STEP 6: ADD THE OUTPUT STATES REQUIRED FOR THE NEXT STEP.

Input					Output		
A	B	C	D	E	X	Y	Z
0	1	0	1	1	1	0	0
1	0	0	1	1	0	0	0

STEP 7: REPEAT UNTIL THE TRUTH TABLE IS COMPLETE.

Input					Output		
A	B	C	D	E	X	Y	Z
0	1	0	1	1	1	0	0
1	0	0	1	1	0	0	0
0	1	0	1	1	0	1	0
0	1	1	0	1	0	0	1

STEP 8: INPUT ROWS 1 AND 3 ARE NOT UNIQUE. AN INTERNAL RELAY (**F**) IS ADDED AS THE FOLLOWING STEPS:

8.1 Add an internal relay (**F**) in both input and output states.

Input						Output			
A	B	C	D	E	**F**	X	Y	Z	**F**
0	1	0	1	1		1	0	0	
1	0	0	1	1		0	0	0	
0	1	0	1	1		0	1	0	
0	1	1	0	1		0	0	1	

8.2 Add the input states to make row 1 and 3 unique.

Input						Output			
A	B	C	D	E	**F**	X	Y	Z	**F**
0	1	0	1	1	**0**	1	0	0	
1	0	0	1	1		0	0	0	
0	1	0	1	1	**1**	0	1	0	
0	1	1	0	1		0	0	1	

8.3 Consider the internal relay states in row 3 (**F=1**), the output must be turned on in row 2 which will result the input state of internal relay (**F**) in the next row (row 3)

Input						Output			
A	B	C	D	E	**F**	X	Y	Z	**F**
0	1	0	1	1	**0**	1	0	0	
1	0	0	1	1		0	0	0	**1**
0	1	0	1	1	**1**	0	1	0	
0	1	1	0	1		0	0	1	

8.4 To repeat step 2-5, internal relay (**F**) in the last row of the truth table (row 4) must be turned off (**F=0**) which will result the input state of internal relay (**F**) in the first row (row 1)

Input						Output			
A	B	C	D	E	**F**	X	Y	Z	**F**
0	1	0	1	1	**0**	1	0	0	
1	0	0	1	1		0	0	0	**1**
0	1	0	1	1	**1**	0	1	0	
0	1	1	0	1		0	0	1	**0**

8.5 Once the similar input rows (row 1 and 3) are solved, complete the truth table by using the same method of 8.3 and 8.4

Input						Output			
A	B	C	D	E	**F**	X	Y	Z	**F**
0	1	0	1	1	**0**	1	0	0	**0**
1	0	0	1	1	**0**	0	0	0	**1**
0	1	0	1	1	**1**	0	1	0	**1**
0	1	1	0	1	**1**	0	0	1	**0**

STEP 9: CREATE THE BOOLEAN EQUATIONS FOR EACH OUTPUT.

$X = \bar{A}B\bar{C}DE\bar{F}$
$\bar{X} = AB\bar{C}DE\bar{F} + \bar{A}B\bar{C}DEF + \bar{A}BC\bar{D}EF$
$Y = \bar{A}B\bar{C}DEF$
$\bar{Y} = \bar{A}B\bar{C}DE\bar{F} + AB\bar{C}DE\bar{F} + \bar{A}BC\bar{D}EF$
$Z = \bar{A}BC\bar{D}EF$
$\bar{Z} = \bar{A}B\bar{C}DE\bar{F} + AB\bar{C}DE\bar{F} + \bar{A}B\bar{C}DEF$
$F = AB\bar{C}DE\bar{F} + \bar{A}B\bar{C}DEF$
$\bar{F} = \bar{A}B\bar{C}DE\bar{F} + \bar{A}BC\bar{D}EF$

STEP 10: THE PLC PROGRAM IS CREATED FROM THE BOOLEAN EQUATIONS.

This step is trivial, because most PLC's have an option to be programmed using Boolean algebra. Also, the International Electronic Technical Commission (IEC) has standard architectures and programming methods for PLC's (IEC 1131-3) [1, 2]. This standard has defined 5 standard programming methods for PLC's. Most PLC manufactures have adopted them. Three of these standards are for graphical programming methods that do not lead to an automated technique, but

the other two are text-based methods. The first is structured text (ST), which is a high level language that is similar to C++ or Pascal. The other is instruction list (IL), which is a very low level language that is like an assembly language. Most programmers find IL very difficult to use but it is very easy to convert the Boolean equations from Step 9 in to a IL program that would work with most major PLC brands.

PROGRAM EXAMPLE

This research has used Allen-Bradley MicroLogix™ 1000 PLC with the RSLogix™ family of IEC 1131-compliant ladder logic programming packages. This program supports the Allen-Bradley SLC™ 500 and MicroLogix™ families of PLC processors. To create the ladder logic diagram from the Boolean equation, latch on the outputs for high equations and unlatch the outputs for low equations as shown in Appendix 1 and 2.

RESULTS

The experimental test was conducted with the collaboration of 17 engineering technology students currently enrolled at the University of Central Missouri. The experiment would give data regarding the easiness to learn PLC programming using Truth Tables as a way of creating the Ladder Logic Program. The students were divided into two groups, and each group was assigned a given method: the Trial/Error (T/E) or the Truth/Table (TT) Method. Prior to the experiment, each participant was asked if he/she had prior PLC experience. Of all participants only 12 percent considered he/she had prior considerable knowledge of PLCs. The experimental test showed that there is a considerable time advantage between students using the TT method compared to the other group. TT students accomplished in less time the task of creating a fully operating Ladder Logic Program. The results of the test showed that TT students spent 44 min/avg to create a Ladder Logic Program, while T/E students needed 55 min/avg and their PLC programming demonstrated faulty. TT students required only 6.67 min/avg to create a well functioning Truth Table, which led to a fully operational PLC program. On the other side, only 25 percent of T/E students without prior knowledge could accomplish the task of creating a working PLC program. This supports our thesis that Truth Tables are a faster and more logical way to absorb the complexity of PLCs and Ladder Logic Programming.

CONCLUSION

The main purpose of this research project was to develop a clear analytical method for developing a program for the control of any equipment controlled by PLC. This new approach would allow the controller for a wide variety of automated industry machine to be quickly configured or reconfigured, and then quickly and automatically programmed. This new method of program generation would be completely unrelated to the size, type of operation (machining, grinding, laser, etc.), or number of stations. The test conducted with the participation of engineering technology students showed clearly that the approach presented lead to a faster and easier understanding of PLC programming. Our approach combines key concepts from various techniques, delivering an efficient and automated method of PLC programming.

APPENDIX 1

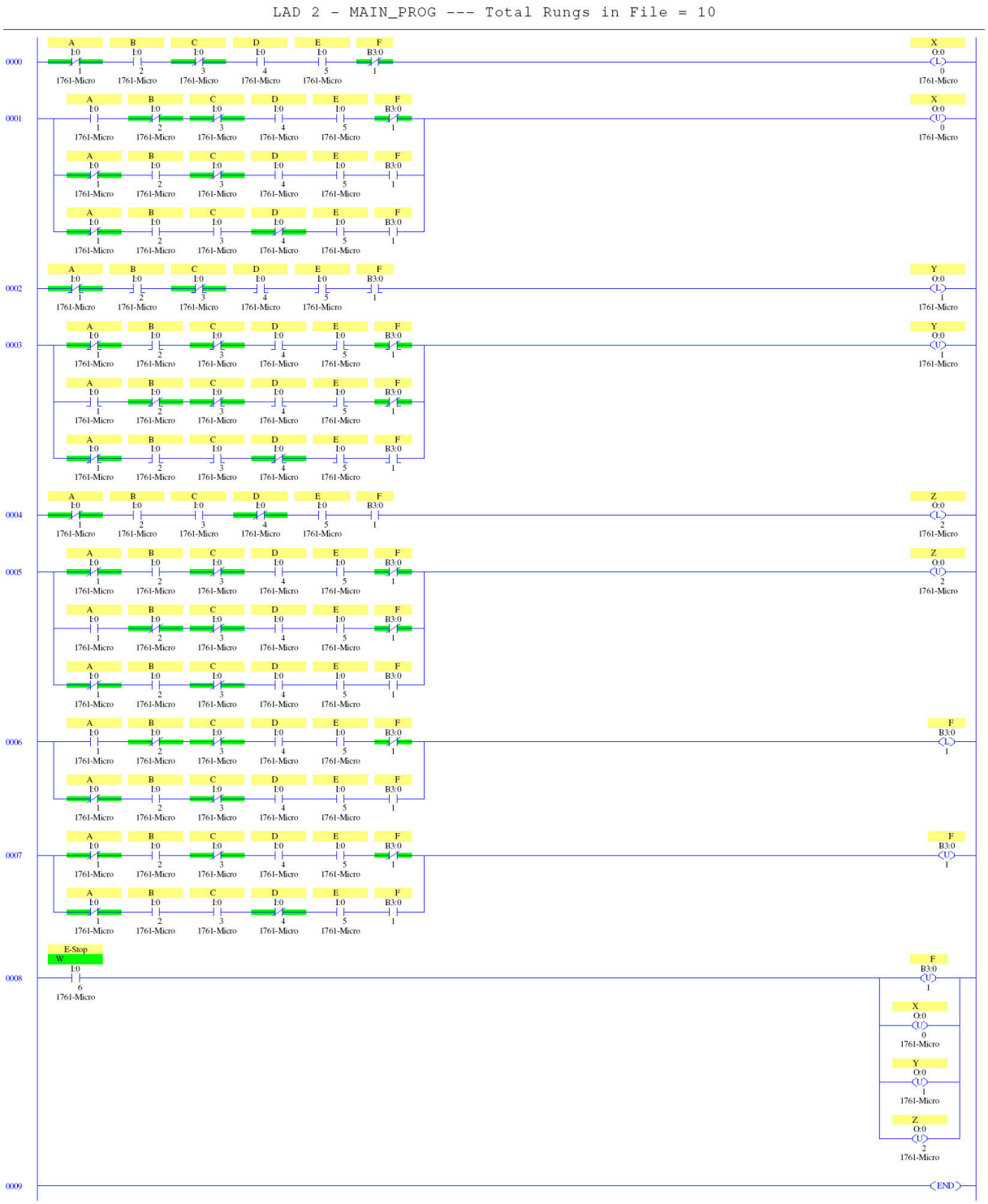

FIGURE 3. LADDER LOGIC DIAGRAM (BIT LEVEL) FROM THE GIVEN EXAMPLE

Note: Input states (logic High or Low) are substituted with XIC (Examined if closed) or Normally opened and XIO (Examined if opened) or Normally closed respectively. For example, logic high (on) will be replaced with normally opened symbol and logic low (off) will be replaced with normally closed symbol.

APPENDIX 2

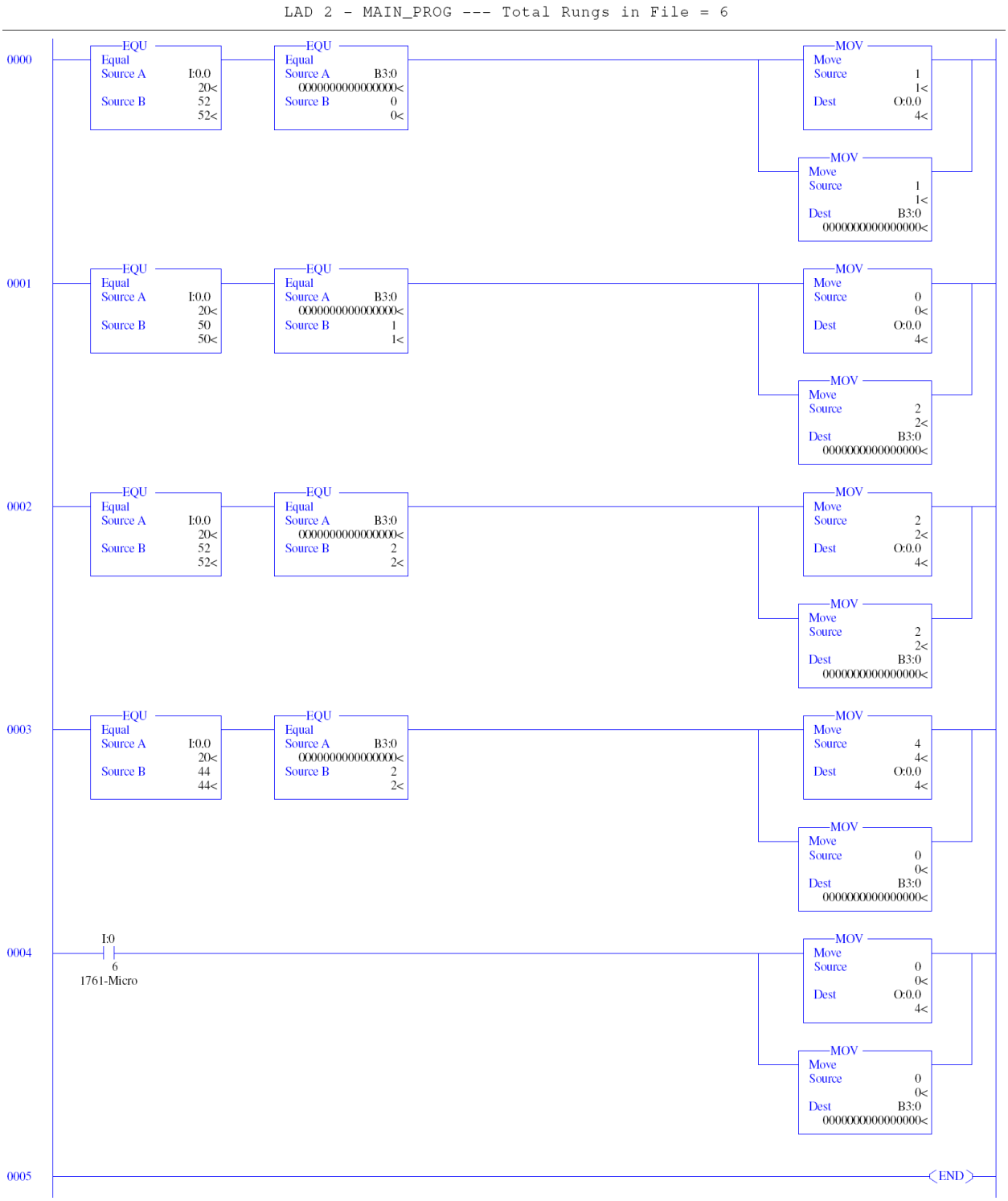

FIGURE 4. EXAMPLE OF LADDER LOGIC DIAGRAM (WORD LEVEL)

Bibliography

1. David A. Geller. *Programmable Controllers Using the Allen-Bradley SLC-500 Family*. 2nd Edition, 2005. Pearson Prentice Hall.

2. Filer, Rober and Leinone, George. *Programmable controllers & Designing Sequential Logic*, 1992. Saunder College Publishing.

3. Frank D. Petruzella, 2005. *Programmable Logic Controllers. 3rd. Edition.* Mc Graw Hill.

4. Hughes, Thomas A, 2001. *Programmable Controllers*. 3rd Edition. The Instrumentation, Systems and Automation Society.

5. Lin, Yuyi, Guoping Wen, Huachao Li and Kang Xue, 1996, *The Design of a 300-ton Hydraulic Press for High Speed Compaction of Coal Logs*, ASM paper 96-WE/DE-5.

6. Max Rabiee, 2002. *Programmable Logic Controller. Hardware and Programming*. The Goodheart –Willcox Company, Inc.

7. Parr, E.A., *Programmable Controllers an Engineer's Guide*, 2nd. Edition, 1993, Newnes.

8. *PLC use to increase; collaborative systems effective for discrete Manufacturers*. Reed Business Information, Control Engineering. Dec. 1, 2005.

9. Stenerson, Jon., *Fundamentals of Programmable Logic Controllers, Sensors, and Communications*, 2nd. Edition, 1993, Prentice-Hall.

10. Terry Bartlet, 2002. *Industrial Control Electronics. Devices, Systems, and Applications*. 2nd Edition. DELMAR, Thomson Learning.

11. *The International Journal of Advanced Manufacturing Technology*. Springer London. August, 2004,Volume 24, Number 3-4.