ET 438B
Sequential Control and Data Acquisition
Laboratory 8
IEC 1131-3 PLC Programming Languages: Introduction to Function Block and Structured Text Programming of a PLC

Laboratory Learning Objectives

1.) Explain the differences between the PLC programming languages specified in the IEC 1131 standard.
2.) Identify simple function blocks and utilize them in a PLC control program.
3.) Use Connected Components Workbench software function block programming to implement a control program.
4.) Convert an electromechanical ladder logic diagram into an equivalent function block PLC program.
5.) Convert a ladder logic program into structured text using the Connected Components Workbench software

Technical Background

Programmable Logic Controllers use ladder logic diagram programming because it is easier to understand compared to other forms of programming. Most industrial maintenance technicians can examine a ladder diagram program and determine how it works due to its similarity to electromechanical contact and coil circuits. Ladder diagram programming becomes unwieldy as controls systems become more complex. Other PLC programming methods provide more concise and easier to implement code but require more sophisticated knowledge of computer programming languages.

Industrial automation companies have complete lines of PLCs, expansion modules, communications interfaces and programming software to meet control applications. These applications range from simple relay/timer/counter replacements to plant-wide supervisory control and data acquisition systems (SCADA). In the past each manufacturer created proprietary hardware, communication, and programming systems requiring automation engineers to purchase all equipment from a single source. The current trend is to also utilize existing standards in communication and programming to leverage existing technologies. One of these standards is IEC 1131-3.

IEC 1131-3 attempts to standardize PLC programming, which allows control engineers to reuse code from different PLC manufacturers with minimum recoding. The specification includes four traditional languages and one high-level language. The standard divides these programming languages into graphical and text-based groups. The graphical group includes Ladder Diagrams (LD), Function Block Diagrams (FBD), and Sequential Function Chart (SFC). The text-based

languages are Instruction List (IL) and Structured Text (ST). The following sections give short descriptions of each language type.

Ladder Diagrams (LDs)

Ladder diagrams are very similar to electromechanical relay logic. Graphic symbols similar to coil and contact schematic symbols represent instructions placed on rungs representing lines of program code. The instructions operate on logical continuity not electrical continuity. The specification includes functions for tasks common to PLCs such as timing, counting and sequential logic implementations. Labs 6 and 7 introduced Ladder Diagram programming using Rockwell Automation's Connected Component Workbench (CCW).

Function Block Diagrams (FBDs)

Function Block Diagram programming uses a technique similar to LabVIEW to code control applications. Function Block Diagrams connect variables with functions using a point to point wiring method similar to connecting components on a schematic diagram. Each function block requires inputs and produces outputs. Global and user-defined local variables provide sources and sinks for the function block input/output ports. Figure 1 shows an example of function block programming using CCW.

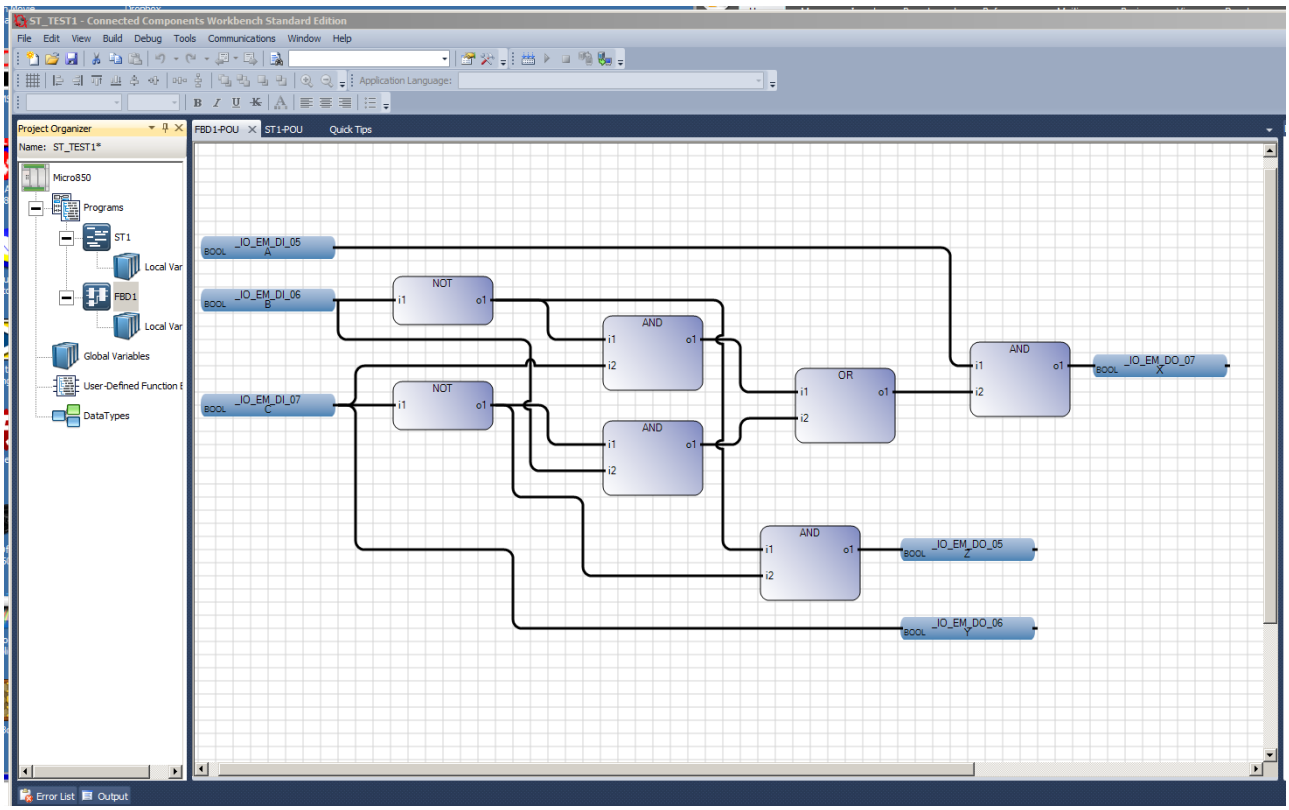


Figure 1. Function Block Programming Example from CCW.

Instruction Lists (ILs)

Instruction List programming is a low-level language similar to Assembly Language in microcontroller programming. Instructions are in a list format with only one operation allowed per line. The low-level nature of this language restricts is use to smaller applications. Figures 2 and 3 show a ladder program and its corresponding Instruction List (called statement list in this manufacturer's literature). Notice the similarity between these instructions and Assembly.

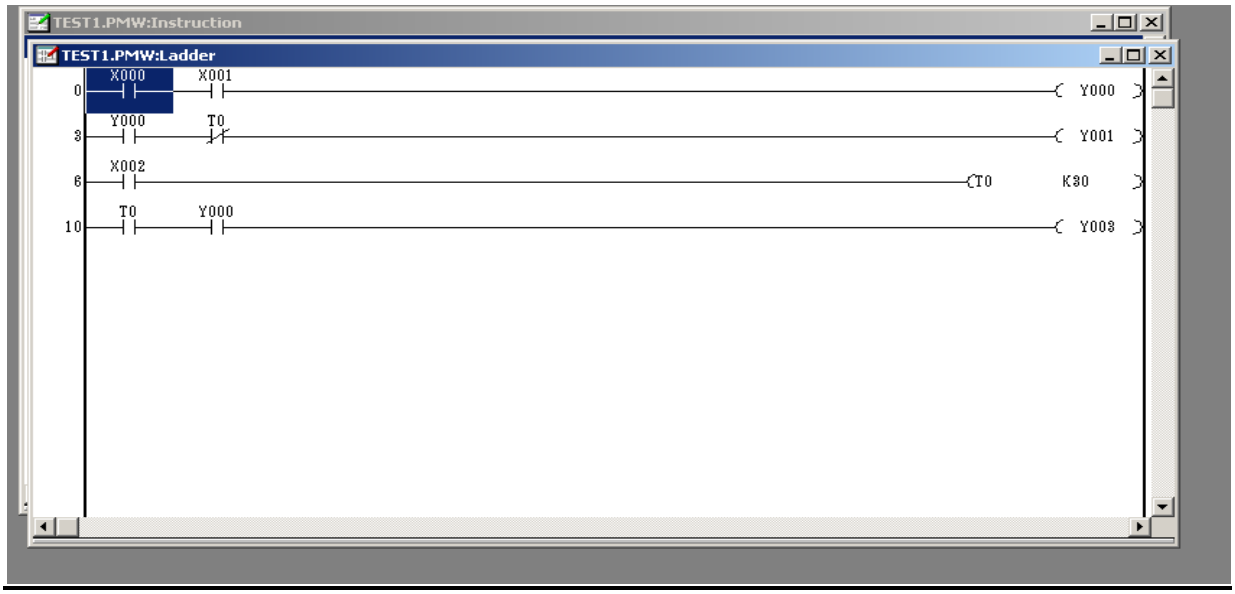


Figure 2. Ladder Logic for Mitsubishi PLCs

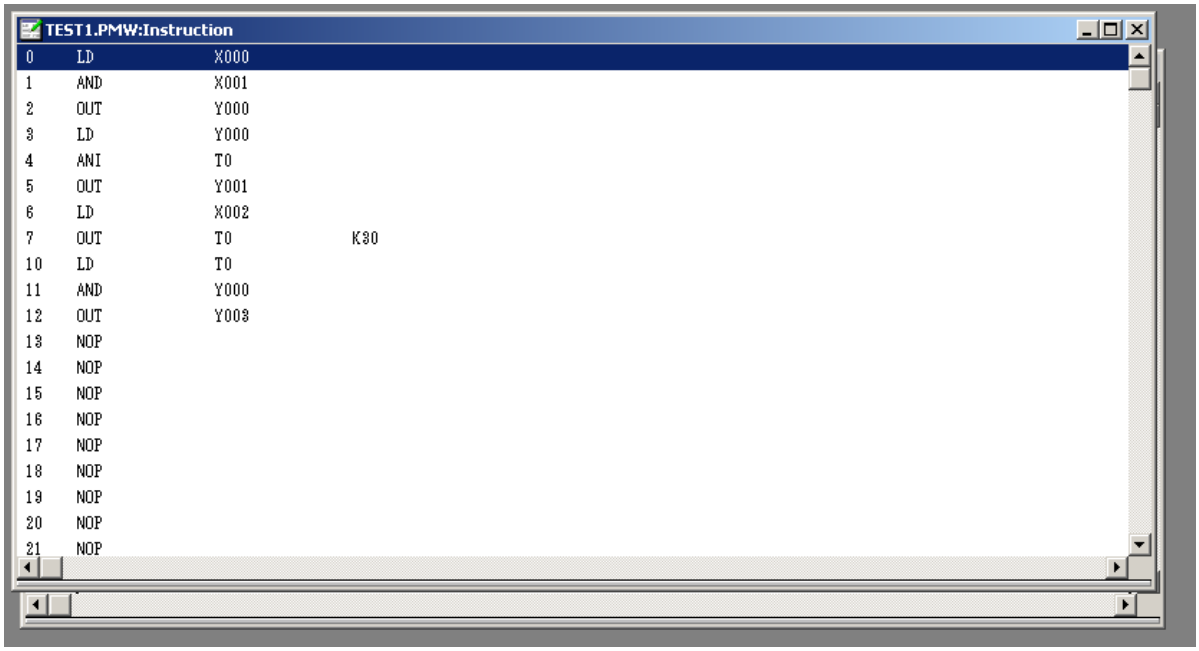


Figure 3. Statement List Program of Above Ladder Diagram.

Structured Text (ST)

Structured Text Programming uses keywords, defined variables and high-level programming structures that resemble other programming languages like BASIC or C. Structured Text programming can handle all functions commonly performed in ladder logic. A well-written and documented ST program is easily followed by any programmer familiar with high-level programming. This type of program would be difficult for most industrial technicians to follow without additional training and instruction. Figure 4 shows an example of structured text programming from CCW. Note that each line ends in a semicolon. This is the end of line character in this programming system and omitting it produces an error at build-time.

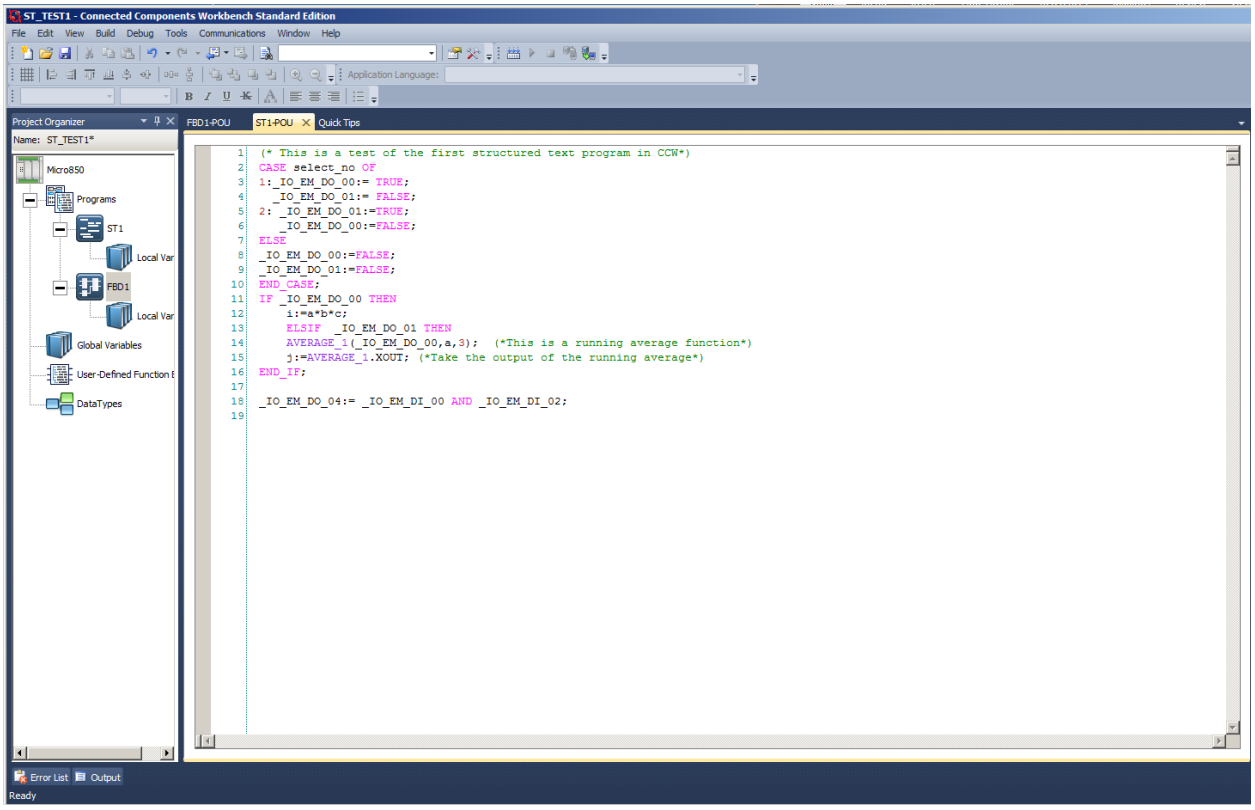


Figure 4. Structured Text Programming Example from CCW.

This program fragment shows the use of CASE and IF-THEN statements. All variables must be defined and given a data type or an error will occur at build time. The CASE statement sets or resets PLC output bits based on a variable called **select_no**. This CASE has two enumerated values: 1 and 2. The ELSE clause handles all other values of **select_no**. The IF-THEN statement also shows how to use built-in functions. The function AVERAGE_1 is part of the Micro850 PLC instruction set. This function computes the moving average of a series of input values given by the "a" parameter in the function argument list. The initial parameter is a Boolean variable that actives the function when its value is TRUE. The final parameter sets the number of data used in the moving average. It is three in this case.

The code in line 19 uses the Boolean AND function to combine the state of two input points. If both inputs are TRUE the output point updates to TRUE.

Sequential Function Chart (SFC)

Sequential Function Chart programs consist of steps and transitions. A step includes one or more actions and represents a box on the SFC diagram. A step is complete when the program executes all instructions in a box. The program then must wait for a transition condition to become true before the next step executes The program executes the steps in a sequence after each transition condition becomes true. Previously executed steps are not executed again. Sequential function chart programming produces a structured code made up of program segments that perform specific tasks. One of the advantages of this programming language is that only one step of code executes, while all others are inactive after a transition. This CCW does not have this type of programming available, but it can be simulated using user-defined function blocks and the state machine model introduced in Lab 2.

Programming Combinational Logic with FBD

Function block diagram programming works well in a number of applications and is similar to other graphical programming languages such as LabVIEW. Program execution moves from left to right and down in a function block program. The variable menu selection from the FBD tool box inserts inputs, outputs and user defined local variables into the FBD. These are the starting points for a FBD program. Figure 5 shows inputs to a FBD program that address digital input points 06 and 07 on the trainer. Inputs connect to outputs and user defined local variables by connecting the function blocks together with a single line. The variable types and the function block I/O must be of the correct type or an error results. Function blocks, such as timers wire in series with the desired outputs and must have all input parameters or an error results.

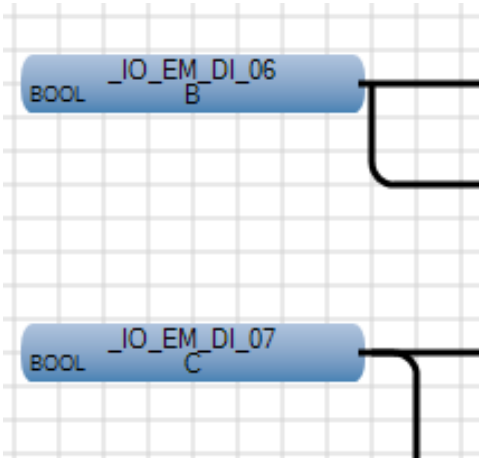


Figure 5. Boolean Inputs on a FBD Program.

Figure 6 shows an on-delay timer used in a FBD program. The input connection comes from a Boolean output on a previous block. A local variable sets the 2 second time delay, and the output, Q, becomes TRUE after this delay, setting the output point _IO_EM_DO_05 TRUE. This energizes the output connected to this point on the PLC.

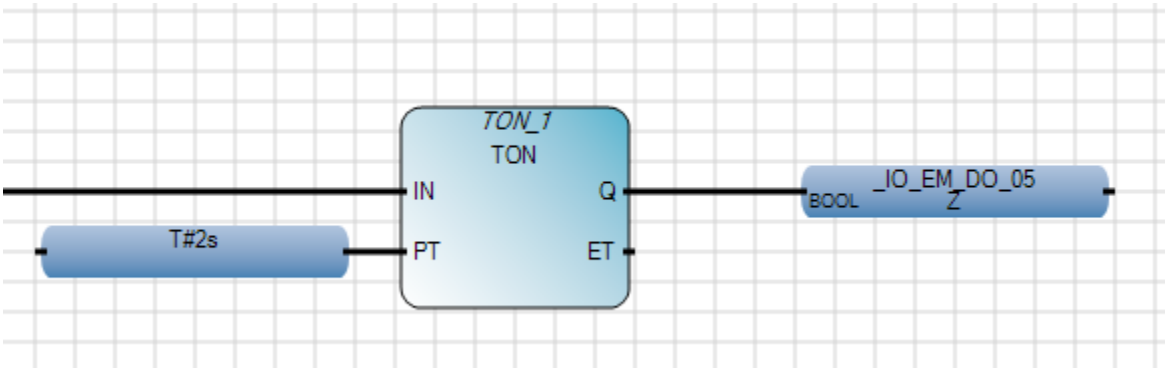


Figure 6. On-Delay Timers in FBD programs

The FBD programming environment allows feedback of signals from outputs to latch variables. The three-wire motor control scheme is a simple control example requiring a feedback signal. Figure 7 shows the ladder logic rung for this control. The motor contactor coil, M, links

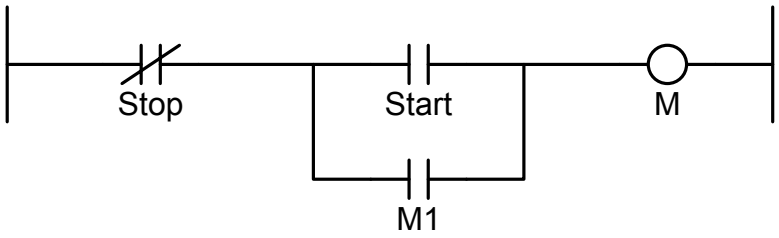


Figure 7. Three-wire Motor Control Ladder Logic

mechanically to the auxiliary contact M1. Pressing the normally open start switch completes the circuit energizing M and closing M1. The coil remains energized though the circuit including STOP and the M1 contact. Figure 8 shows the FBD implementation using the local variable CR1 to represent the M coil and contact M1.

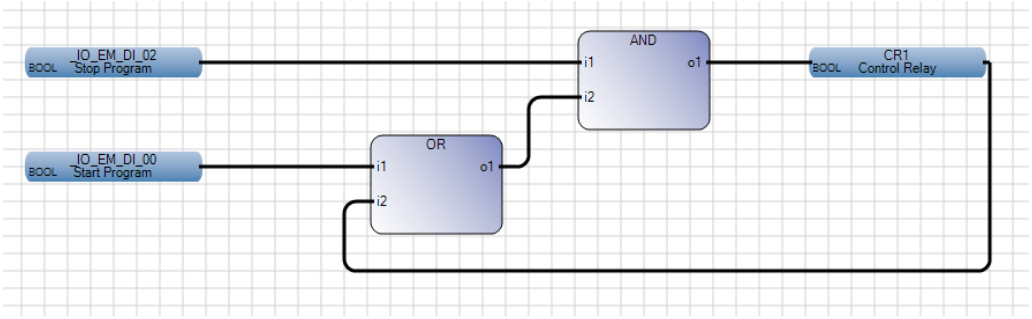
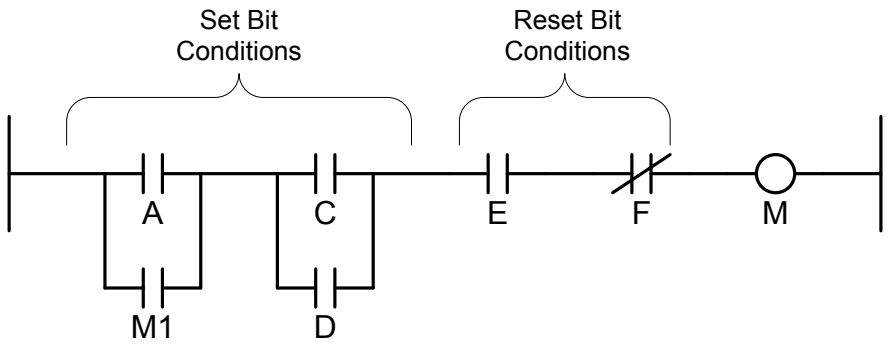


Figure 8. FBD implementation of Three-wire Motor Control Demonstrating Variable Feedback.\

FBD Programming Example

Write the Boolean algebra equation for the following ladder logic rung and the implement it using FBD programming



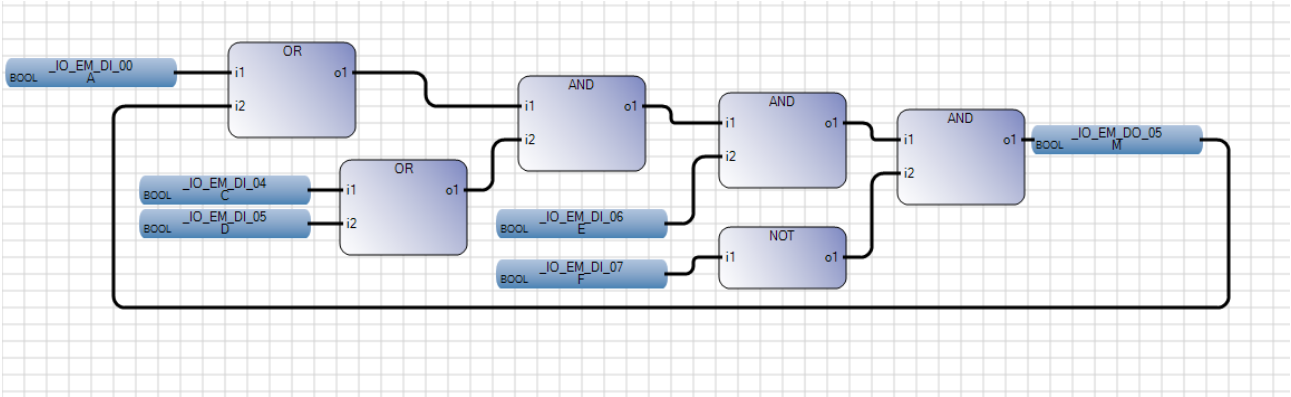
Use the following Iuput/Output mapping.

Inputs			Outputs
A =DI0	M1=DO5	C=DI4	M=DO5
D=DI5	E=DI6	F=DI7	

Notice that the contacts on the left of the rung represent conditions the cause the output bit to be set and are all OR'ed together. The contacts on the right of the rung represent conditions that wil reset the output bit. The contact M1 and the coil M are logically linked. The following Boolean expression represents the rung conditions.

$$M = (A + M1) \cdot (C + D) \cdot E \cdot \overline{F}$$

The FBD implementation follows directly from the Boolean equation. The feedback from the output M to the input provides latching action for the output. View the video presentation in the



learning management systems for the step-by-step construction details of this program.

Structured Text Programming Example

Recode the FBD example using Structured Text language statements.

Figure 9 shows the statements required to code the ladder logic rung. This example uses user-defined local variables, Out1, Out2, and Out 3 to make the code more readable. The first line in the program is a comment defined by the (*, *) strings. The second code line produces the Out1 result by OR'ing input DI_00 with the output point DO_05. Each line of code must end with a semicolon or an error appears when building the program. This provides the latching action since

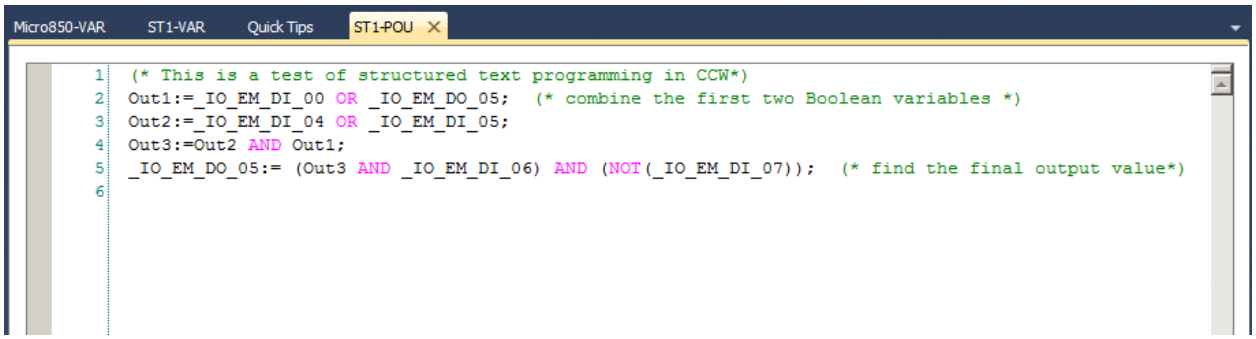


Figure 9, Structured Text Code for the Example.

the switch connected to this input is a momentary contact device. Line 3 combines the inputs labeled C and D on the ladder rung with another OR function. The user-defined local variable Out2 holds the result. The next line uses the results of the two previous lines to produce the result stored in Out3. The last line combines the final two inputs with this result.

The structured text program requires the least amount of effort to enter, but is the most difficult to understand by viewing the code. This type of program requires knowledge of programming in a high-level language and can be difficult for technicians with basic electronic and electrical knowledge to understand. The structured text program supports all commonly used program control structures such a IF-THEN-ELSE, CASE, FOR and WHILE loops. For skilled programmers, this method of programming a PLC will seem the most natural.

Block functions enter into ST programs as function calls with specified input and output parameters. The code below shows how to include an off delay timer in a ST program.

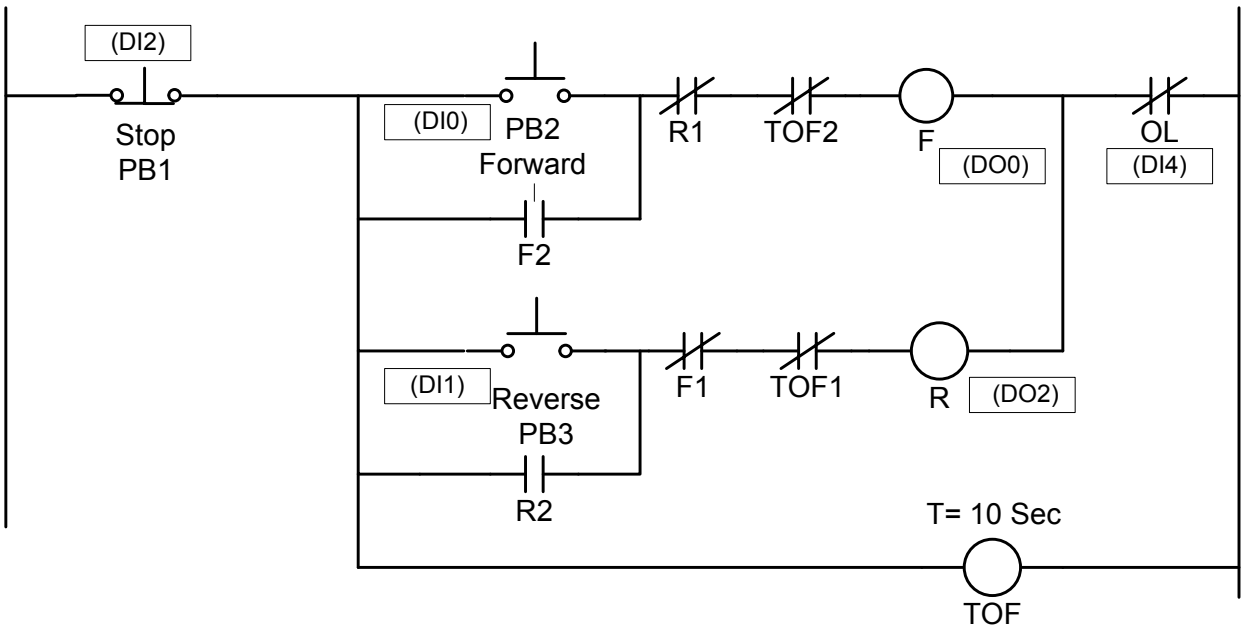
```
Max_time:= T#3s;  (*define a preset time *)
TOF_1(in,Max_time);  (*create an instance of the TOF function *)
output:=TOF_1.Q;   (* place the timer output into a new variable *)
```

The general function call is given as:

void **TOF_1**(BOOL IN, TIME PT)

where IN is the Boolean input parameter and TIME is the programmed time value in time format. Commas separate parameters in ST functions. Typing the name of the function invokes a list of possible choices for selection during the programming process. Consult the Micro800 Programmable Controllers General Instructions Reference Manual (Publication 2080-RM001B-ENE-March 2014) for more details on the ST equivalent of the PLC standard functions. This document is available from the course documents in the learning management system.

Function Block Diagram and Structured Text Programming Project



1.) A manufacturer wishes to replace the electromechanical motor starting scheme shown above with a PLC control program. The schematic is for a reversing motor starter. The starting control has a normally-open forward run push button (PB2) and a normally-open reverse run push button (PB3). The stop push button (PB1) is a normally closed switch. The forward and reverse motor contactor coils are given by F and R respectively. A motor overload relay contact shuts down motor operation in either direction when a mechanical overload occurs. A restarting off-delay timer prevents starting of the motor in the opposite direction until it has coasted to a stop. Experience shows that this occurs after 10 seconds. Convert the electromechanical scheme above into a PLC control using Function Block Diagram programming. Download the developed program to the Micro 800 trainer and test it for proper operation.

Use the following input/output mapping.

PB1 =DI2 PB2=DIO PB3=DI1 OL=DI5 F=DO0 R=DO2

Remember that electromechanical and PLC ladder diagram program do not use identical symbols in some cases.

2.) Code the same control using structured text programming. Download the resulting program to the trainer to test its operation.

Lab 8 Assessment

Complete and submit the following items for grading and perform the listed actions to complete this laboratory assignment.

1.) Complete the online quiz over Lab 8 technical background.
2.) Code programs 1 and 2 into the PLC trainer
3.) Demonstrate working programs to the lab TA
4.) Write short descriptions (one page double-spaced or less) describing the operation of the program
5.) Submit pdf files of the working programs and the appended descriptions to the Lab 8 dropbox.