

EET 438B
Sequential Control and Data Acquisition
Laboratory 6
Introduction to Connected Component Workbench and RSLogix Software

Laboratory Learning Objectives

- 1.) Navigate the Connected Component Workbench and RSLogix Software.
- 2.) Create and save a simple ladder logic program using the software packages
- 3.) Download the program to a PLC and monitor its operation using debug/monitor functions in the software
- 4.) Use the contact commands and output commands to create basic ladder logic programs.

Technical Background

Programmable Logic Controls (PLCs) implement sequential control logic in many industrial applications. PLCs are industrially hardened microprocessors that perform logic operations, timing, counting, sequencing, proportional-integral-derivative (PID) control, and other advanced control functions. PLCs use software programs to implement these functions, which makes these functions easy to update and modify. The first PLCs were developed in the late 1960's by the automotive industry as a replacement for hard-wired electromechanical relay panels used to control assembly lines and other tooling. This device eliminated the high costs related to reconfiguring tooling for each new model. The PLC has evolved into a multifunction device that can communicate with other PLC and host computers using a number of protocols, collect and process data, and link with front-office software to give plant management real-time information regarding machine operations.

PLCs have the following advantages: a.) they are flexible and easily reconfigured to do different tasks or include additional tasks. b.) they use solid-state electronics and are highly reliable when installed correctly. c.) they cost less than electromechanical components. d.) their installed programs can be easily documented by simply printing out the completed source code. The PLC finds application at several levels in modern control applications. The trend is to use smaller PLCs in a distributed control system. Locating smaller PLCs on individual machines that communicate among other controllers and human-machine interfaces (HMIs) allows control engineers to divide large complex problems into a series of smaller more easily coded control applications.

A PLC can come in two forms: expandable or fixed. Fixed PLCs have a defined hardware configuration that cannot include other input/outputs or special function modules. Expandable PLCs come with a base number of input/output points but can grow by adding other modules to the base configuration. These modules include PID control, analog inputs, analog outputs and communication modules. These modules install into a backplane or rack along with a power supply.

Figure1 shows an expandable PLC configuration. Expandable PLCs require a backplane that usually includes a power supply. A designer can specify either ac or dc power inputs for the PLC supply. Older model PLCs accepted 120-240 ac input voltages. The industry trend is to supply all control devices from lower voltage dc supplies, typically 24 Vdc.

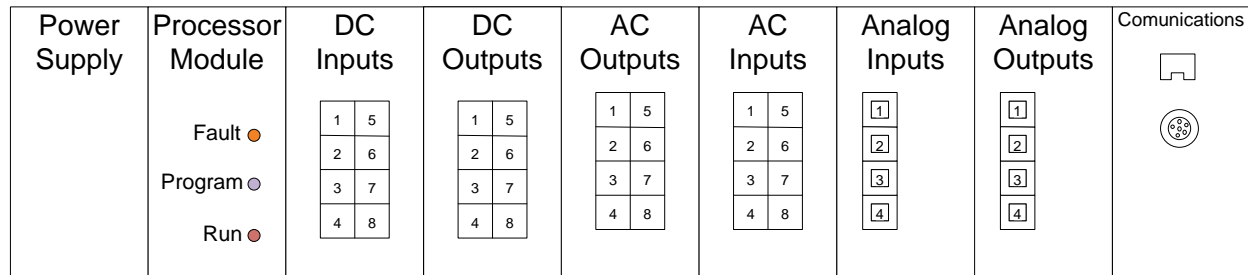


Figure 1. Expandable Processor Configuration

Some configurations integrate the power supply with the processor. Other configurations allow system designers to specify the processor in the backplane.

The input modules come in groups of 8, 16 and 32 points and can handle either ac or dc and plug into the backplane. The system designer can configure the dc modules to be either current sources or sinks. The output modules install into the backplane also. Output modules can include dc source/sinking types, ac output, and relay outputs. Relay outputs can accept either ac or dc voltages on the relay contacts but the circuit current cannot exceed the manufacturer's maximum rating.

Modern PLC modules include analog input and outputs. These devices convert analog voltages or currents into digital values for processing in PLC programs. Communications modules allow system designers to download programs to the PLC and provide networking capability so communication with other PLCs and HMIs is possible. Programming PLCs requires a host computer, usually a PC. The PLC communicates with the host using several standards. Older PLCs communicate using the high-level serial communications protocol RS-485. Other communications links include RS-232 serial communication, USB, TCP/IP and Modbus.

PLC Processor Operation

PLCs are a specialized application of microprocessors. They have many parts in common with other computer systems. They take inputs, evaluate the inputs against a user-supplied program, and produce usable outputs. PLCs use memory-map input/output (I/O) to correlate physical input/output points to individual memory locations in the PLCs.

Previous generations of PLCs used an addressing scheme that required programmers to specify the physical location of I/O points starting with the type of module: input = I and output = O followed by the slot in an expandable PLC. The final part of the address is the physical point on

the module. Figure 2 shows a byte of PLC processor memory used to map input points. The input module has 16 dc inputs and is in slot 1 of the processor backplane. The following syntax locates the logic 1 bits for Allen-Bradley PLCs in the SLC 5/500 and Micrologix families.

15														0	
1	0	0	0	1	0	0	1	0	0	0	0	1	0	1	1

Figure 2. Input memory location for 16 bit PLC

Addresses of the logic 1 bits in the memory word:

I:1/0

I:1/1

I:1/3

I:1/8

I:1/11

I:1/15

The first part of this syntax identifies the memory location as an input. The number after the colon is the slot location of the input points. The final number in the address is the physical input point. The expression below gives the general syntax for the input memory map address.

I:(slot number)/(point number)

The addresses of the output points are similar, but the initial identifier changes from I to O for output. The following expression gives the general syntax for output addresses.

O:(slot number)/(point number)

Specifying memory locations for individual bits not associated with an I/O point requires use of bit memory, also known as the bit file. The general syntax shown below identifies a bit file memory location.

B:(memory word)/(bit address)

The B identifies the desired memory location to be part of general memory. The number after the colon identifies the addressed word in memory and the final number identifies the bit within the word.

Example: B:0/3

This address locates the forth bit in word 0 of the PLCs' bit file memory.

Programming instructions require I/O addresses to be complete. Instructions without addresses produce error when programs compile.

The addressing scheme for the Micro800 series is much simpler. The process comes with defined variable names that identify the I/O points of the PLC. Specifying the PLC configuration creates the I/O variable listing. These variable are of type BOOLEAN in the programming environment. A programmer creates additional bit value variables by defining them in the programming environment. These become the user-defined variables in the programming project.

External switches cause memory locations in input memory to change. Figure 3 shows how external input devices connect to PLC inputs and appear in the memory map. The input

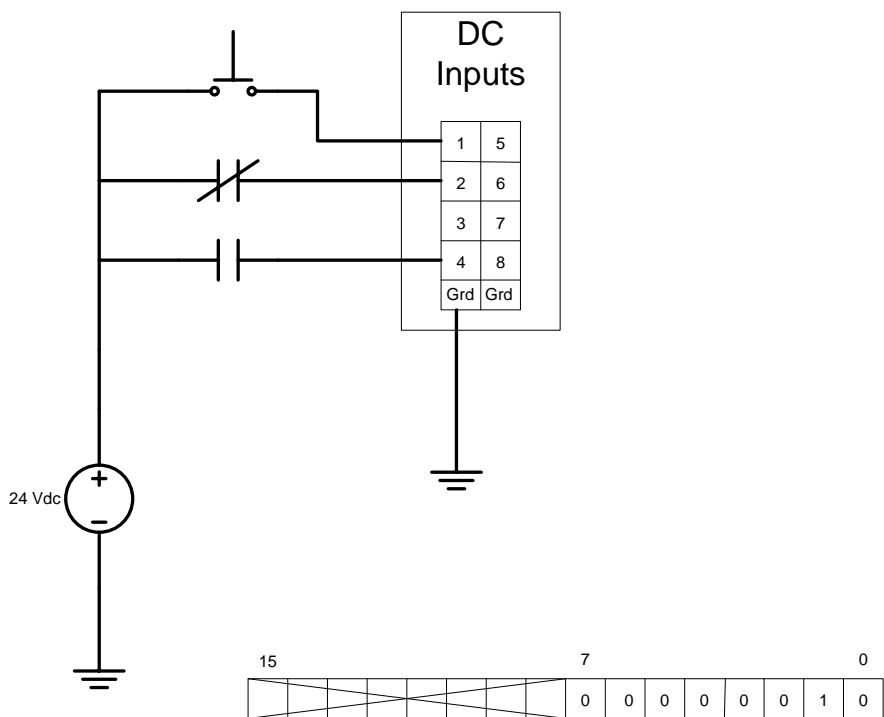


Figure 3. External Input Device Memory Mapping

module in this figure has three switches attached. Two are normally open and one is normally closed. A closed switch produces a logic 1 in the PLC memory map and an open switch results in a logic 0. The last eight bits of this memory word (high byte) are not available for addressing since the slot contains an eight input. PLC input external wiring requires a power supply with the correct current type (ac or dc) and voltage level for the input module to sense changes in state of the input devices. System designers should consult the I/O module instruction sheets for these values.

The CPU of a PLC can operate in two modes, run and program. In the run mode, the PLC operating system places the CPU into a while loop and performs the following actions.

- 1.) **Scan Inputs** - the processor scans the input memory locations and updates them as necessary.
- 2.) **Scan Program** - the PLC processor evaluates the logical statements and other actions specified in the program using the latest input values
- 3.) **Update Outputs**- the processor updates output memory using the latest program evaluation results.
- 4.) **Processor Housekeeping** - processor handles communication requests, updates watchdog timer, responds to debug requests.

The time required for a PLC program to complete the four steps listed is called the processor scan time. The scan time increases with the complexity of the program and the number on I/O points. As scan time increases rapidly changing input transitions may be lost. This can cause program malfunctions and are difficult to troubleshoot.

The programming mode of a PLC allows host computers running program development software to download new program code to the processor. The host and PLC must have an established communication link for a program transfer to take place. The lab PLCs communicate with host PCs using RS-232 serial connections and USB. It is also possible to communicate with PLC over TCP/IP networks. These devices can be part of the overall facility network or a stand-alone private network.

A program developer can monitor and debug a PLC program when it is communicating with the host computer or is part of a network to which both PLC and host belong. The PLC continues to run its existing program in the monitor mode. A copy of the program appears in the programming environment located on the host. This program reflects changes in PLC I/O by changes in program code colors. PLC variable values are available to programmers in the development software.

Programming A PLC Using Ladder Diagram Programming

Several methods exist for programming modern PLCs. The most intuitive is Ladder Diagram programming (LD). This programming method uses symbols that resemble the coil/contact schematic diagram symbols used to document electromechanical hard-wired ladder logic. Ladder Diagram programming is easy for someone with limited programming background to understand and can quickly implement simple sequential control functions. LD programming becomes cumbersome when a project required more advanced functions, such as robot motion control.

LD programming is graphical and has a set of rules for constructing ladder rungs. These rules differ depending on the manufacturer of the PLC hardware/software but the following list covers typical rules.

- 1.) All outputs are on the left-most location of a rung. No other symbol must appear after an output symbol on a rung.
- 2.) Inputs are on the right side of the rung
- 3.) Program logic flows down the ladder and from left to right
- 4.) The number of series contact symbols is limited
- 5.) The number of parallel contact symbols is limited
- 6.) A rung can have only one output

Figure 4 shows a valid ladder diagram rung from the Allen-Bradley Connected Components Workbench (CCW) development software. Note that all input symbols are on the left and

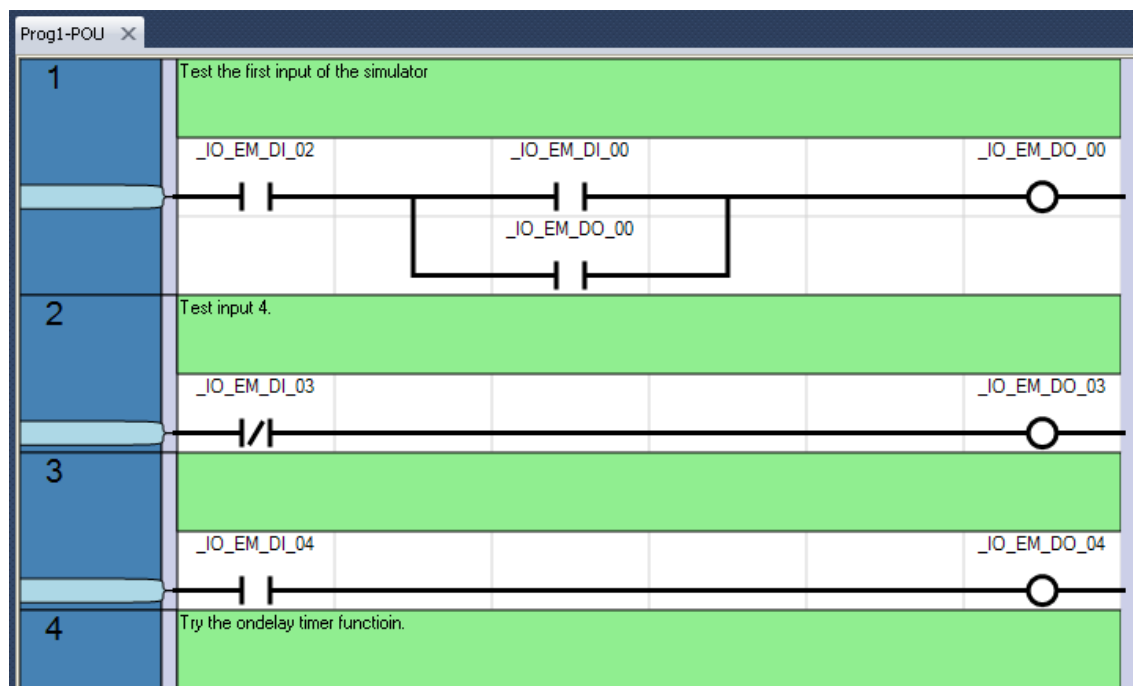





Figure 4. Connected Components Workbench Ladder Diagram Showing Program Structure.

outputs are on the right. Each instruction relates to a PLC memory address shown above the instruction.

The schematic symbols of the LD program represent instructions that test the condition of bits in the PLC's memory. Tables 1 and 2 define the operation of three basic logic symbols of ladder diagram programming and give their outputs. The tables are for both the Allen-Bradley CCW and RSLogix software. Table 1 gives the names for the symbol in each programming environment.

Table 1. Basic Ladder Program Instructions

LD Symbol	Connected Components Workbench	RSlogix
	Direct Contact (DC)	XIC (examine if closed)
	Reverse Contact (RC)	XIO (examine if open)
	Direct Coil (DCL)	OTE(output energize)

Ladder diagram programming uses logical continuity not electrical continuity to determine if an output instruction is executed. The DC/XIC and RC/XIO instructions test any bit in memory and return Boolean values based on the condition of the bit. The output instructions DCL/OTE set or reset bits based of the final Boolean value of a rung. Table 2 list the instructions and their Boolean and bit results.

Table 2. Instruction Logical Results

Input Bit	Input Instruction		Output Instructions	
	XIC/DC Output	XIO/RC Output	Rung Result	OTE/DC Output
1	TRUE	FALSE	TRUE	1
0	FALSE	TRUE	FALSE	0

Electromechanical ladder diagrams and the ladder diagram program do not have a one-to-one correspondence since LD programs operate on logical not electrical continuity. Figure 5 shows an electromechanical diagram for a motor starter control.

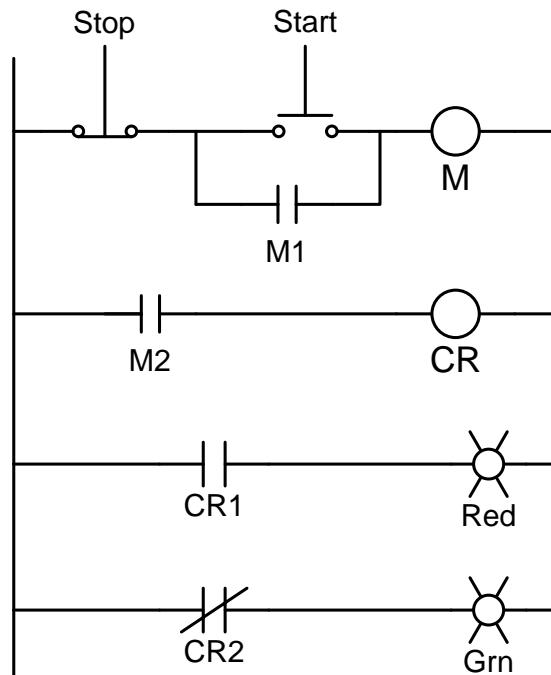


Figure 5. Electromechanical Motor Control Ladder diagram.

Two manually operated push buttons start and stop the motor by energizing the motor contactor coil M. The contact M1, which is mechanically linked to the contactor coil, maintains electrical continuity in the rung. The contact M2, also mechanically linked to M, energizes a control relay, CR that powers the correct indicator lights through the contacts CR1 and CR2. Figure 6 shows how the I/O devices connect to PLC I/O modules and how the memory maps read for the motor off condition. Notice that there is no CR relay in the PLC implementation. A bit in PLC memory evaluated using the DC/XIC and DCL/OTE instructions take its place.

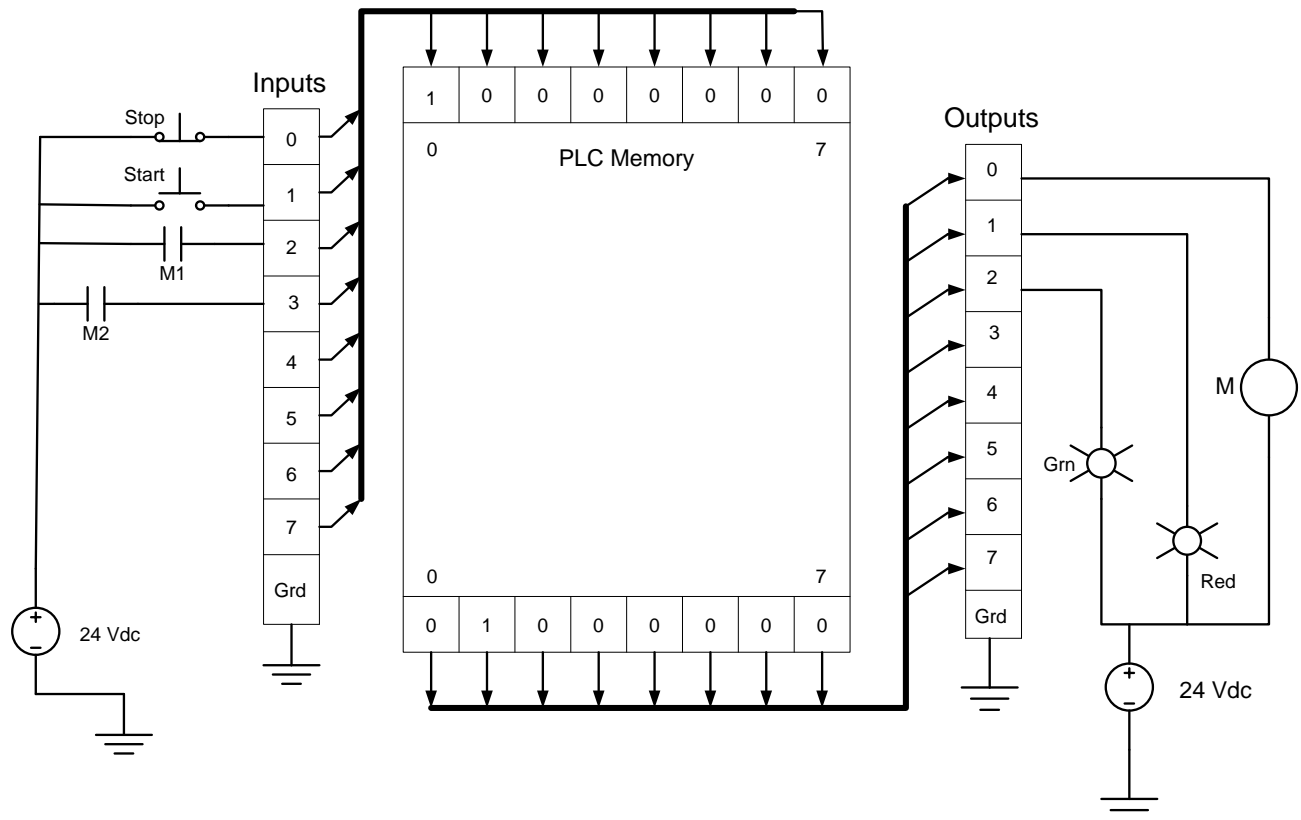


Figure 6. PLC External Connections For Motor Control Derived From Electromechanical Diagram.

The following Boolean equations describe the logic of the motor control in Figure 5.

$$\begin{aligned}\overline{STOP} \cdot (Start + M1) &= M \\ M2 &= CR \\ CR1 &= RLITE \quad \overline{CR2} = GLITE\end{aligned}$$

These equations are the building blocks of the ladder diagram program. Equation 1 shows that the output M is energized when the stop button is not depressed and either the start button is depressed or the M-coil is energized closing the M1 contact. Programming this into the PLC requires teach input device be connected to an input point and the correct address of the input be

applied to programming instructions. Figure 7 shows the completed PLC ladder diagram program developed using the CCW programming software. Rung 1 of the program handles the start/stop operation. The address `_IO_EM_DI_00` associates the 0-th input of the PLC to the DC

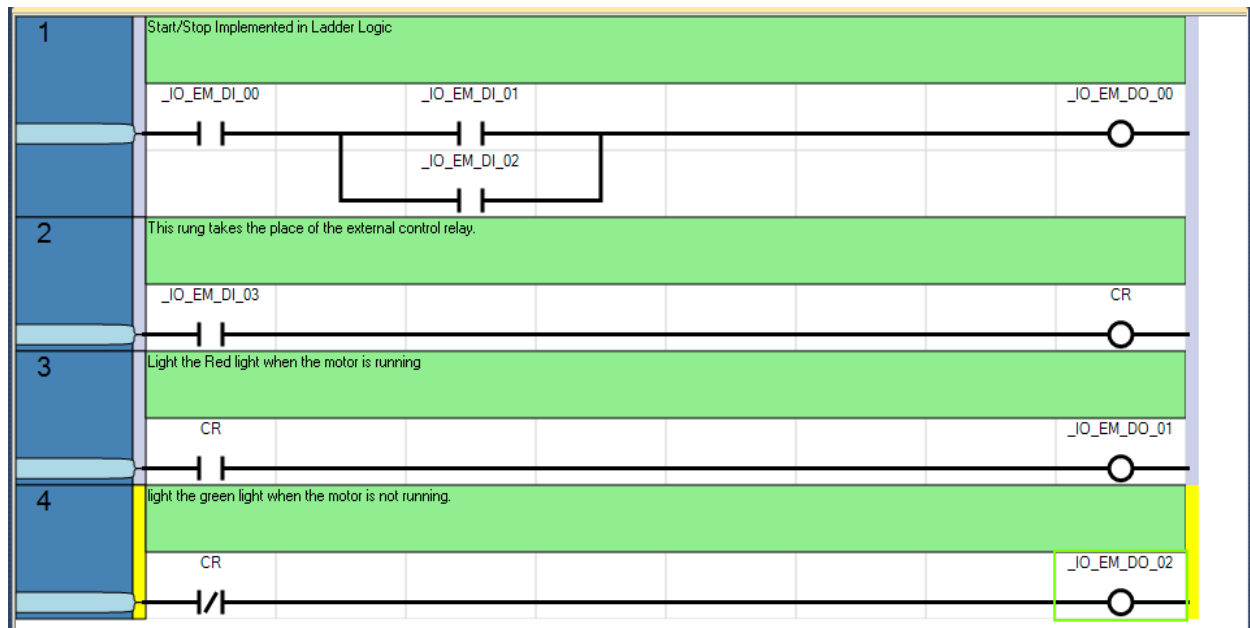


Figure 7. Ladder Diagram Implementation of a Motor Starter

instruction. This input connects to the normally-closed push button stop switch. The ladder diagram instruction is DC/XIC and not a RC/XIO that would match the normally closed symbol of the switch. The PLC rung uses logical continuity as shown in the Boolean equation. The rung must test the Stop switch to see if it is closed to complete logical continuity. The output instruction maps to the `_IO_EM_DO_00` memory location to energize the motor starter coil M. The input addresses `_IO_EM_DI_01` and `_IO_EM_DI_02` act on the bits associated with the start switch and the motor starter contact M1 respectively.

When the program runs, depressing the Start switch will cause the bit addressed at `_IO_EM_DI_01` by the DC instruction to return a TRUE. This is ANDed to the result of the DC instruction addressing the normally closed Stop switch causing the entire rung to return a TRUE to the DCL instruction. A TRUE rung result for the DCL instruction sets the bit at `_IO_EM_DO_00` energizing the motor starter coil. The mechanical linkage between the M coil and the M1 contact causes the DC instruction addressing `_IO_EM_DI_02` to return TRUE providing an alternate logic path when the Start button is released.

In rung 2, the DC instruction addressing `_IO_EM_DI_03` maps to the input point for contact M2 that is also mechanically linked to M. An energized M coil closes the contact and produces a

TRUE result from the DC instruction. The TRUE result causes the DCL instruction labeled CR to set a bit in memory. The CR variable is a user-defined Boolean variable that replaces the external relay coil and contacts of the electromechanical circuit shown in Figure 5.

Rungs 3 and 4 use the bit address CR along with DC and RC instructions to control indicator lamps. Setting the CR bit causes the DC instruction in rung 3 to return a TRUE. This causes the DCL instruction to set the bit in memory location `_IO_EM_DO_01` to 1 causing the red lamp to light. The RC instruction in rung 4 evaluates the logic 1 in the CR memory location and returns a FALSE. This causes the DCL instruction addressing `_IO_EM_DO_02` to reset the underlying bit de-energizing the green light. The PLC operating system continually scan the inputs for state changes, runs the program logic and updates output while the PLC is in the run mode.

Debugging Ladder Programs

Connected Component Workbench software includes a graphical debugger to help developers indentify logical errors in program execution. To use the debugger a PLC must have a program file downloaded to it and the PLC must be communicating with the host computer. Selecting **Debug** from the menu or pressing F5 places the software in debug mode.

The ladder logic diagram in CCW changes color when operating in the debug mode. Color changes on program rungs indentify the extent of logical continuity.. Depressing a physical input switch causes the instructions linked to that memory address to change color. When a program includes counters and timers, the debugger displays current values of their outputs. The debugger also supports variable watches. This allows the developer to monitor the values of program variables as the program runs.

The RSLogix software has a similar mode for monitoring the status of the memory locations and the program logic.

The Micro800 PLCs programming software does not include an emulator. Emulators are software tools that execute processor instructions on the host without downloading them to the processor. An emulator exists for the SLC 5/500 series processor. Developers can download this tool from Rockwell Automation free of charge.

Basic PLC Programming Procedure

Developing the PLC programs can be done on your own computers if you download the CCW software and install it. The latest release, Version 6.01 runs under the Windows 7 operating system. Version 4 only operates under Windows XP. Limited versions of the communications software RSLinx and the Ladder programming software RSLogix is also available for download

- 1.) View the demonstration videos in D2L the show how to navigate the development environment, communicate from a host PC to a PLC, develop and download a program.
- 2.) Locate a computer system with PLC programming software installed. The computers in D106 have RSLogix software and CCW software installed.
- 3.) Develop and debug PLC ladder programs specified below for the Micro800 series trainers.

Programming Assignments

Use the CCW software and Micro800 trainers to develop and demonstrate the following ladder diagram programs. Some of these programs can be combined into one program. Those that require previously allocated I/O points must be in a separate program.

- a.) Create a program that will cause outputs DO-0 and DO-1 to light when the switches DI-0 and DI-2 are actuated.
- b.) Create a program that will cause output DO-2 to light when either DI-5 or DI-3 are actuated.
- c.) Develop a program using DI-0 and DI-1 as inputs and DO-3 as an output that implements XOR logic with ladder instructions
- d.) Use user defined variables to create a ladder program that implements the following logic function:

$$Y = A \cdot B \cdot C + A \cdot \bar{B} \cdot \bar{C} + \bar{A} \cdot \bar{B} \cdot C$$

Where: A=DI-4

B=DI-5

C=DI-6 and Y=DO-4

Lab 6 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 6 technical background.
- 2.) Demonstrate working programs to the lab TA
- 3.) Submit pdf files of the working programs with a short written description of how each program operates.