Development of a Relay Ladder Logic Programming and Simulation Tool with Graphical User Interface

Kevin M. Hubbard, Ph.D., CMfgE
University of Missouri-Rolla

I. Abstract

Two software packages have been developed in the Computer Integrated Manufacturing Laboratory of the University of Missouri-Rolla's Engineering Management Department. The first, LadderCAD, is a relay ladder logic programming environment. It features:

- "Point and Click" graphical user interface based ladder logic program construction.
- Forward Interpretation: The generation of an English-like representation of the control logic embodied by the ladder logic program.
- Backward Interpretation: The generation of a ladder logic program from English-like control logic statements.
- The use of assignable English device names in place of device addresses.
- The generation of PC based data acquisition card device control code.

The second, SIM, is a simulation package. Using this software package, ladder logic programs may be tested, verified, and debugged off-line. Through a graphical user interface, the user may actuate/deactuate input devices, and monitor the state of output devices, counters, timers, and mathematical instructions in real time, or in "step mode". Once the ladder logic program has been verified, it may be used to run physical equipment.

Students receiving instruction in relay ladder logic programming may use LadderCAD's forward and backward interpretation routines to gain a more thorough understanding of the process by which these programs are written and interpreted. LadderCAD and SIM may also be used in the instruction of digital data acquisition card programming.

II. Background

Programmable Logic Controllers (PLC's) are used extensively for logic and sequencing control in a wide variety of applications. The demand for these controllers will continue to increase during the foreseeable future[1]. As a result, a demand exists in industry for engineers proficient in their use and programming. To satisfy this demand, many Mechanical, Electrical, and Manufacturing Engineering curricula contain courses dealing with PLC programming and applications.

Although a variety of programming methods for these devices are in use, the most common PLC language is relay ladder logic, a symbolic language in which the PLC is programmed to emulate the activity of a set of logic circuits. Relay ladder logic offers many benefits, but suffers from a number of disadvantages as well. Chiefly, these disadvantages lie in the areas of:

- Programming Approach: The ladder logic programmer must take a "bottom up" approach to program development[2]. Ladder logic code is written at the lowest possible level. If a program overview is required, the generation of that overview involves additional activities on the part of the programmer. These documentation layers are not contained in the program itself, but instead are generated outside of the programming environment. One example of the program overview generation activity is that of the translation of a ladder logic program into a process diagram.

- Language Variation: Ladder logic is not standard across PLC brands, and is not portable from one brand of PLC to another. This issue is of particular importance to the PLC training provider in an academic setting. University PLC programming courses should, if possible, focus on PLC programming concepts, not on the specifics of the programming of a particular type of PLC. Hardware and software specificity in the training environment must be minimized in order to provide broadly based training which is applicable regardless of the PLC type being used.

- Simulation Considerations: Most ladder logic programming environments provide no overall graphical view of the entire process. Ladder logic environments typically provide powerful user interface troubleshooting tools such as input/output (I/O) forcing and graphic reduction of state (power flow), but unfortunately, unless the program being verified is very small (usually three or fewer rungs), the programmer can view only a small portion of the program on the screen. This difficulty forces the programmer to install "dummy rungs" in the program for troubleshooting purposes.

Because of these and other difficulties, ladder logic programs may be difficult to create, debug, verify and maintain. LadderCAD and its companion utility, SIM, address each of the problems outlined above.

Students often find ladder logic concepts difficult to grasp upon first exposure. Most first time ladder logic programmers have dealt with textual, structured languages such as C, FORTRAN, and BASIC, but the method of execution of programs written using these languages is quite different from that of the ladder logic program. What is needed, in order for these students to make the transition from textual languages to ladder logic, is an "example generator", to be used in conjunction with lecture and laboratory exercises. LadderCAD is such an example generator. With its Windows based graphical user interface, LadderCAD allows the user to easily construct ladder logic diagrams using drag-and-drop, cut-and-paste methods.

Once the ladder logic diagram has been drawn, it can be translated by LadderCAD into an English-like text file. This file contains a set of instructions based on the QuickBASIC programming language, and is known as a system state language (SSL) file. This file is easily read and intuitively understandable. An examination of the SSL file provides an understanding as to whether the desired control program result has been achieved. This understanding may then be verified through the use of the SIM software package. The student may also perform this procedure in reverse, by first writing the SSL file, and allowing LadderCAD to create, from that file, a ladder logic diagram. These translations (English-like text to ladder logic diagram, and vice versa) constitute an example generator which can be used to supplement the in-class, blackboard based examples provided by a PLC programming instructor. Novice programmers can use LadderCAD and SIM to perform "what-if" analyses in order to develop effective ladder logic programming methods, and experienced programmers may use these software packages to speed program development, aid in troubleshooting and debugging, and increase program maintainability.

The LadderCAD and SIM packages may be employed in PC based data acquisition card device control as well. Once the student has mastered the concepts of relay ladder logic programming, these packages may be used to generate control programs which make use of data acquisition cards for device control. These files are generated during the forward translation procedure.

III. Software Descriptions

LadderCAD is a user friendly ladder logic development tool. LadderCAD's ladder logic diagram editor, shown in Figure 1, employs a graphical user interface, which allows the user to drop symbols and device names into a form, thus building a ladder logic diagram. Each symbol in the form (normally open and closed switches, output instructions, math and comparison instructions, etc.) must have an accompanying address - the physical device, internal device, or variable to which the instruction refers. For example, it is meaningless to place a normally open switch in the form without specifying the physical device whose state is to be queried by that instruction.

LadderCAD's logic editor provides the user with a database of physical and internal devices from which to choose. This database cross references the English name of the device with its data acquisition card address and PLC address. The database is stored in an ASCII file, and is easily edited by the user.

LadderCAD performs forward interpretation, creating a man-readable system state language (SSL) code file, an executable run time module (RTM) code file, and a simulation (SIM) code file for use by the SIM package. Each of these three files is based on the ladder logic diagram drawn by the user.

System state language files possess the filename extension .SSL, and are man-readable. Instructions necessary for the execution of the file to control physical hardware are not included in this file, and devices are referenced by their user-assigned English names, not by PLC or data acquisition card addresses. This file contains English-like statements that may be reviewed by the user in order to determine whether the intent of the control logic has been achieved.

Run time module files are given the filename extension .RTM, and when executed, control the physical hardware referenced by the ladder logic diagram and system state language files. This control is achieved by way of a PC based data acquisition card. Devices are referenced using their data acquisition card I/O addresses. In order to accomplish I/O functions, the RTM file is appended to a "header" file named HEADER.BAS. This header contains subprograms which handle timers, counters, one-shot instructions, math and comparison instructions, and data acquisition card I/O. The header file is easily edited by the user so that various types of data acquisition cards may be employed.

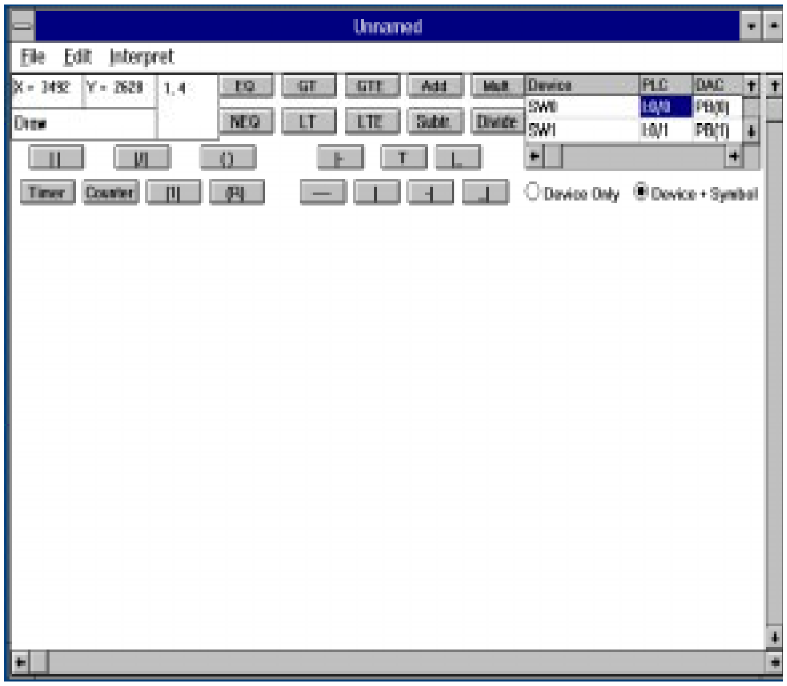

Figure 1: The LadderCAD Editor Form

Simulation files possess the filename extension .SIM, and contain instructions to be used by the SIM package. With the SIM package, the user can force, by way of on-screen command buttons, the state of inputs, and observe the state of outputs, timers, logical bits, counters, reset instructions, etc. The main screen of the SIM package is shown in Figure 2. Note that with both packages, device names are specified by the user. These device names are read from the ASCII device database at program startup.

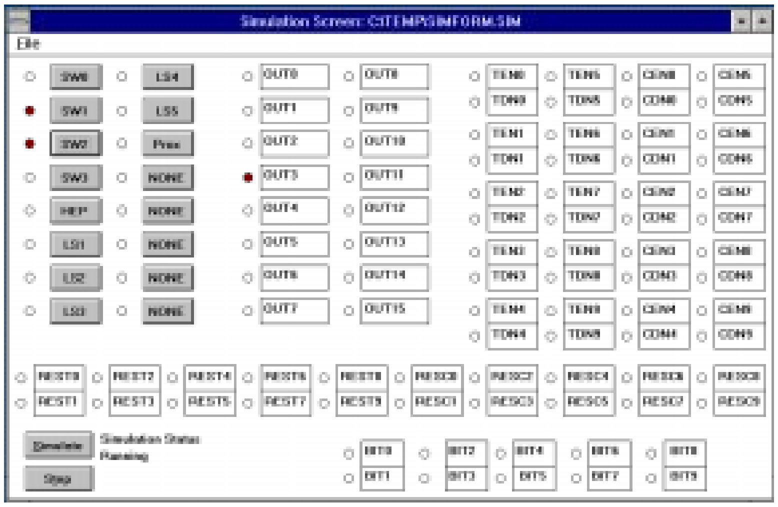

Figure 2: The Simulation Package Main Screen

IV. Relay Ladder Logic Instruction Using LadderCAD and SIM

Using LadderCAD, the student is encouraged to generate relay ladder logic programs by way of three steps. In the first step, the student poses and answers the question "what output devices are to be controlled by the program?". All output devices are listed in the device database, shown at the top right of the ladder logic editing form in Figure 1. In the second program generation step, the student poses and answers the question "what input devices are pertinent to the execution of the program?". As with output device names, all input device names are contained in LadderCAD's device database.

The third program generation step is the most complex. In this step, the student poses the question "under which system states is each output device to be actuated?". In other words, "under which conditions is each output to be actuated?". The student may write these conditions in the form of a system state language file, and cause LadderCAD to perform a backward interpretation, generating a ladder logic diagram. Once the student is proficient in this process, he/she may skip the system state language generation procedure, and proceed directly to the generation of a relay ladder logic diagram using LadderCAD's ladder logic editor. After the diagram is generated, a forward translation may be performed, in which SSL, RTM, and SIM files are generated. The student may then examine the system state language file in order to determine whether the ladder logic diagram embodies the desired control logic, and test program execution using the SIM package.

As an example[3], consider the equipment shown in Figure 3. When the pneumatic cylinder is in its fully retracted position, limit switch 6 is actuated, and when the cylinder is in the fully extended position, the hall effect proximity sensor is actuated. When solenoid 4 is energized, the pneumatic cylinder extends, and when solenoid 4 is deactuated, the cylinder retracts. A novice PLC programming student, asked to create a program to cause the pneumatic cylinder to cycle back and forth between its fully retracted and fully extended positions, might create the following system state language code. Line numbers have been added to this example to facilitate explanatory comments.

```
1  DOITAGAIN:
2  IF LS6 THEN
3  SOL4 = 1
4  ELSEIF SOL4 AND  NOT HEP THEN
5  SOL4 = 1
6  ELSE SOL4 = 0
7  END IF
8  GOTO DOITAGAIN
```

Line 1 contains a line label, "DOITAGAIN". Line 8 branches to this line label, causing the system state language file to appear to execute in an endless loop, imitating the execution of a relay ladder logic program. The other lines of the system state language code may be interpreted as shown below.

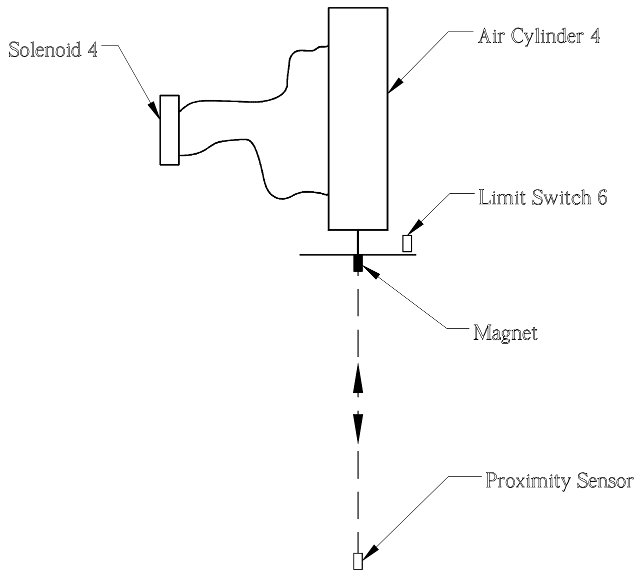

Figure 3: Example Equipment Setup

2 If limit switch 6 is on, then
3 actuate solenoid 4
4 or else, if solenoid 4 is actuated and the hall effect proximity sensor is not actuated, then
5 actuate solenoid 4
6 otherwise, deactuate solenoid 4
7 end of block if

LadderCAD's backward interpretation routine, using this system state language file, would generate the relay ladder logic diagram shown in Figure 4.

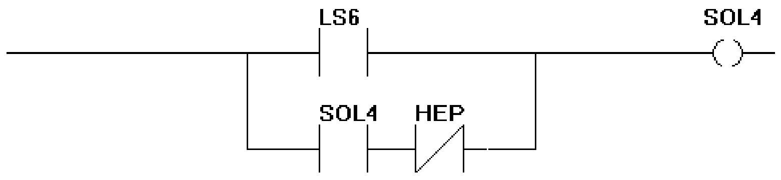

Figure 4: Example Relay Ladder Logic Program

During the forward interpretation process, LadderCAD generates three files: a system state language file, a simulation file, and a run time module written in QuickBASIC. The simulation file may be run by the SIM package to verify program execution, either in run mode (continuous execution) or in step mode (one program scan per user request). The SIM package is started, and the correct SIM file is loaded. At program startup, the SIM package retrieves the user assigned equipment names from the device database. Input device names are assigned to command buttons, and output device names are assigned to labels, as shown in Figure 2. Once the simulation is placed in run mode, the user may toggle the states of input devices using their command buttons, and observe the output states which result. Once program operation has been verified using the SIM package, the run time module file may be used to control physical equipment.

The run time module file may be used to operate physical equipment by way of a PC based data acquisition card. The run time module consists of a header file and user defined logic corresponding to the ladder diagram created with the LadderCAD package. The header file contains all instructions necessary for data acquisition card setup, as well as subprograms which handle timers, counters, etc. The user defined logic is appended to the header file, and its syntax is similar to that of the system state language file. The user defined logic appended to the header, however, references physical devices using their data acquisition card I/O addresses rather than user assigned device names. The run time module executes in a loop, allowing the user to exit program execution and disable all output devices by striking any key on the PC's keyboard.

An instructional equipment set consisting of motors, optical sensors, limit switches, proximity sensors, and other equipment has been assembled, as well. Using these components, students can design, assemble, and connect devices to accomplish tasks outlined by the instructor, and then generate programs to accomplish those tasks. This equipment is light weight and portable, and has been used both on-campus, and in courses in the Engineering Management department's on-site M.S. program at Boeing-St. Louis (formerly McDonnell Douglas Aerospace).

The LadderCAD and SIM packages may also be used for instruction in digital data acquisition card programming. The run time module output from the LadderCAD package is, in fact, a control program for PC based device control by way of a digital data acquisition card. Once the student is familiar with elementary ladder logic program construction, he/she may use LadderCAD as an "example generator" for this purpose, comparing his/her individually developed data acquisition card control programs against those generated by the LadderCAD package. It is also useful for the student to develop subprograms for the handling of one shot devices, timers, counters, etc., for comparison against corresponding subprograms contained in the header file. Ladder logic program construction is substantially different from procedural language device control program construction, and these differences may be highlighted in laboratory exercises, providing the student with valuable insight into the two device control methods.

At present, LadderCAD handles only digital control, as well as mathematical instructions (add, subtract, multiply and divide), mathematical comparison instructions (greater than, less than, greater than or equal to, less than or equal to, equal to, not equal to), timer on delay type timers, count up counters, one shot rising instructions, and logical bits. Work is under way to expand LadderCAD's capabilities to include analog control, and to incorporate advanced control algorithms such as potential, integral, and derivative control, as well as combinations of these algorithms.

V. Summary

Programmable logic controller programming instruction, and PC based data acquisition device control instruction at the University of Missouri-Rolla have been enhanced through the creation and use of the LadderCAD and SIM software packages. With these software packages, students are introduced simultaneously to both control methods. Through the use of forward and backward interpretation, and off line simulation, students gain a more thorough understanding of the intricacies of digital device control. This software, in conjunction with the instructional equipment set developed in the Engineering Management department's Computer Integrated Manufacturing Laboratory, have been used successfully both on-campus, and in the Engineering Management Department's M.S. program on-site at the McDonnell Aircraft and Missile Systems division of Boeing in St. Louis, Missouri.

VI. References

1. Programmable Logic Controllers: Operation, Interfacing, and Programming
   Job Den Otter
   Prentice-Hall, 1998

2. Use of Object Oriented Programming Methodology in Hierarchical Control Application Employing PLC's
   Master's Thesis
   Henry J. Johnston
   University of Missouri-Rolla, 1992

3. Development of a Relay Ladder Logic Programming/Debugging/Simulation Software Package With Graphical User Interface
   Ph.D. Dissertation
   Kevin M. Hubbard
   University of Missouri-Rolla, 1996

VII. Biographical Information

KEVIN M. HUBBARD is an assistant professor in the University of Missouri-Rolla's Engineering Management department. He also serves as the Manufacturing Engineering academic advisor, and as the Computer Integrated Manufacturing Laboratory Director. He teaches and conducts research in the areas of manufacturing processes and manufacturing system control and integration.