Pergamon

Computers ind. Engng Vol. 33, Nos 1–2, pp. 421–424, 1997

0360-8352/97 $17.00 + 0.00

PII: S0360-8352(97)00127-7

Programmable Logic Controllers versus Personal Computers for Process Control

Agustín Rullán, Ph.D.

Industrial Engineering Department
University of Puerto Rico
P.O. Box 5000
Mayagüez, Puerto Rico 00681

ABSTRACT

It is proposed that Personal Computers (PCs) can be used effectively for the same industrial applications as Programmable Logic Controllers (PLCs). The basic concepts related to the operation of a PLC to emulate the behavior of a relay panel are explained. This is used to build a simple model for using standard PCs in the same applications as PLCs. This is demonstrated using a simple example of an automated process. Sample code in a standard high level language is presented that can be used as a template for future developments. Finally, the potential advantages and enhancements that can result in using PCs for process control are presented.

Keywords: Programmable Logic Controller, Personal Computer, Automation, Ladder Logic, Relay Panels

INTRODUCTION

Personal Computers (PCs) can be used effectively for the same applications as *Programmable Logic Controllers* (PLCs). A modern PLC is a computer-based device designed to control a process. It relates information coming from sensors that monitor the state of a process, with the status of some actuators that are capable of changing it. This relationship is established in most cases in Boolean logic. Typical sensors used in industrial applications include limit switches, proximity sensors, and other binary sensors. Actuators may include solenoids, motor starters, and other similar devices. Figure 1 shows a simplified model of an industrial application where a PLC might be used.

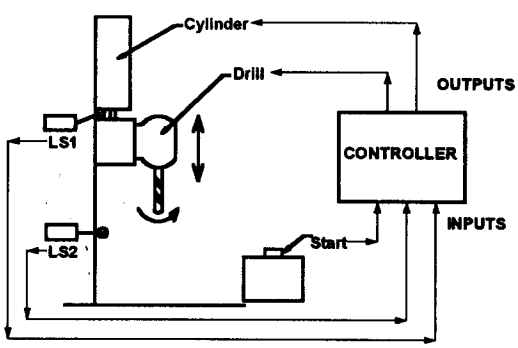

Fig. 1: Model of Simple Process

RELAY PANELS

PLCs were designed to replace *relay panels*[5]. These are custom made controllers dedicated to a particular application. They can be expensive for complex systems, cannot be easily reconfigured, are difficult to troubleshoot, consume lots of energy, have a relatively moderate speed of operation, and have low reliability. Relay panels are not the most suitable alternative for a moderate to complex industrial application where flexibility, ease of maintenance and troubleshooting are very important. On the other hand, they are relatively low-tech and are easy to understand by electricians and non-engineering personnel.

The electrical control circuits of relay panels are generally drawn using so-called *electric ladder diagrams*. They differ slightly from conventional wiring diagrams in that they do not show the physical arrangement of the components, but emphasize the function of each circuit. They are a set of parallel circuits that in essence represent a hardwired program that controls the sequence of operations in a given process. Being electrical circuits in parallel gives the advantage of solving all the control logic simultaneously and practically instantaneously. Figure 2 represents an electric ladder diagram for a relay panel that can be used to control the process presented in fig. 1.

PROGRAMMABLE LOGIC CONTROLLERS

PLCs are typically computer-based, solid-state, single-processor devices that emulate the behavior of an electric ladder diagram. Since they are sequential machines, to emulate the workings of parallel circuits that respond instantaneously, PLCs use an *input/output image table* and a *scanning cycle*.

An input/output image table is a memory structure that stores all the relevant information about the current scanning cycle. It can be subdivided in three basic parts: the input table, the output table, and the internal relay table. The current state of the inputs (sensors) is kept in the input table; the desired state of the outputs (actuators) is kept in the output table; and the state of the virtual control relays is kept in the internal relay table.

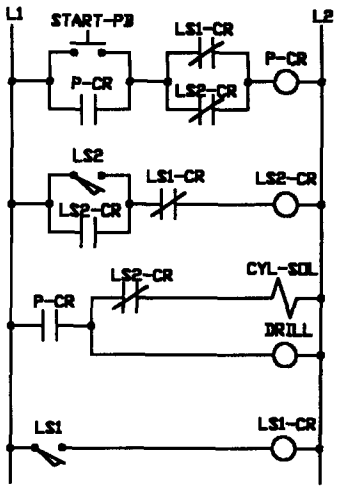

Fig. 2: Electric Ladder Diagram

When a program is being run in a PLC it is continuously executing a scanning cycle (fig. 3). The scanning cycle has two major parts (in an actual PLC it has other parts but two are relevant here for the sake of the discussion): the input/output scan, and the program scan. In the input/output scan the current state of the inputs is read from the input points and stored in the input table, and the desired state of the outputs (from the output table) is sent to the output points. The program scan solves the Boolean logic that relates the information in the input table, with that in the output and internal relay tables. Also, the information in the output and internal relay tables is updated during the program scan. In a PLC this Boolean logic is typically represented in a graphical language that looks very much like the electrical circuit that it emulates. This language is known as *ladder logic*. In fig. 4 there is a ladder logic program that can be used to control the process in fig. 1. Note the similarities with the electric ladder diagram in fig. 2. The advantage of the PLC scanning cycle scheme is that it allows multiple processes to be controlled concurrently as in a relay panel.

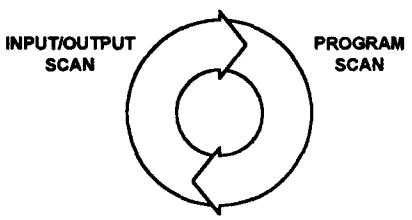

Fig. 3: PLC Scanning Cycle

PLCs overcome all of the relay panel shortcomings and currently are the most widely used industrial automation controllers. At the time of their introduction they were very successful because their language, ladder logic, was based on electric ladder diagrams which engineers and electricians of the time were already familiar with[5]. Nevertheless, PLCs have some shortcomings of their own that are evident when compared with other available technology. Some of the most limiting ones include: there is no industry-standard hardware or software platform, they may limit the programmer as to the control actions and manipulations that can be made, and offer relatively low computing power for the money.

PERSONAL COMPUTERS

Another technology that could be used for the same purpose is the standard personal computer (PC). The PC is becoming increasingly popular for process control. A PC-based controller model is proposed here which could replace PLCs in any industrial control application and would open up a world of possibilities in software development, standard components, and connectivity in general. This model includes an industry-standard PC running any modern operating system, a set of standard input/output modules equivalent to those found in a typical PLC, and an application developed in any high level programming language which will implement a PLC scanning cycle.

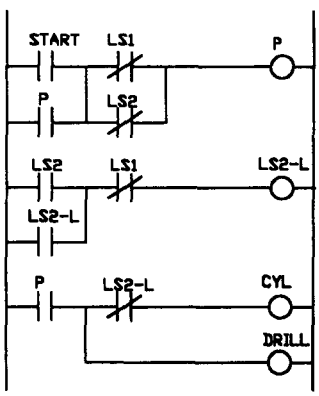

Fig. 4: PLC Ladder Logic

A SIMPLE EXAMPLE

The proposed model will be presented through a sample program coded in Turbo Pascal® for the control of the process shown in fig. 1. Assume that sensors are connected to the PC through input port hex address 3BD, and that actuators are connected to output port hex address 3BC, as shown in figs. 5 and 6 respectively.

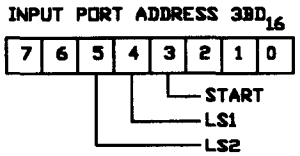

Fig. 5: Input Connections

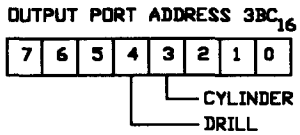

Fig. 6: Output Connections

The sample Pascal program is shown in fig. 7. The image tables: input, output, and internal relays, are implemented using Pascal Boolean variables. When any of these variables is TRUE, that represents an ON state; and when they are FALSE that represents an OFF state. When exchanging signals with the input and output points an ON state is represented by a logical "0" and an OFF state with a logical "1".

The main program is an emulation of the PLC scanning cycle presented in fig. 3. In this simplified example it is assumed that the PLC will be running its program until a key is pressed on the PC keyboard. Other device can be used if it is desired to use the PC keyboard for other more productive purpose.

The input/output scan is emulated using subroutine I_O_Scan. There, the 8 bit input port (fig. 5) is read and the status of the individual bits stored in the input table. Also the status of the output points from the output table is written to the 8 bit output port (see fig. 6). The Program_Scan subroutine is a direct translation of the ladder diagram in fig. 4. For every rung in the ladder logic there is an if-then-else statement. The Initialize and Finalize subroutines were added to handle the state of the output points and internal relays during the startup and shutdown of the PLC program.

```
Program PLC; Uses CRT,DOS;
Const on=true; off=false;
var {Image Tables}
{Input table}  start, ls1, ls2,
{Output table} cylinder, drill,
{Int.relays} process, ls2_1:boolean;

Procedure I_O_Scan;
    var I_O: byte;
begin
{read to Input table}
   start:=(port[$3bd] and $08)=0;
   ls1:=(port[$3bd] and $10)=0;
   ls2:=(port[$3bd] and $20)=0;
{write from Output table)}
  I_O:=$ff;
  if cylinder then I_O:=I_O and $f7;
  if drill then I_O:=I_O and $ef;
  port[$3bc]:=I_O
end;

Procedure Initialize;
begin
{Outputs}
    cylinder:=off; drill:=off;
{Internal Relays}
    ls2_1:=off; process:=off
end;

Procedure Finalize;
begin Initialize; I_O_Scan end;

Procedure Program_Scan;
begin
    if (start or process) and
       (not ls1 or not ls2)
            then process:=on
            else process:=off;
    if ls2 or ls2_1 and not ls1
      then ls2_1:=on
      else ls2_1:=off;
    if process then drill:=on
              else drill:=off;
    if process and not ls2_1
      then cylinder:=on
      else cylinder:=off
end;

begin {main - PLC Cycle Scan}
      Initialize;
      repeat
            I_O_Scan;
            Program_Scan
      until keypressed;
      Finalize
end.
```

Fig. 7: Sample Pascal Program

It should be noted that this is a very simplified example whose purpose is to provide a framework for control program development on a PC combining the programming paradigms of standard procedural high-level languages with the familiar PLC ladder logic. Although it cannot be fully demonstrated in this paper, it provides a simple way of simultaneous control of parallel processes without considering the intricacies of a particular operating system. It demonstrates that complex custom software does not have to be created to make this technology work.

The proposed model would be very easy to adapt and implement even in-house because it can still use ladder logic, so no new technological skills are necessary. Any typical PLC instructions can be added to the model very easily, including timers, counters, one-shots, and so on. Other custom PLC instructions

suited to a particular application can be included that use the features of high level languages, making it more powerful than standard PLC ladder logic.

ADVANTAGES OF PERSONAL COMPUTERS

The PC is a standard hardware/software platform. PCs improve at a rapid pace, become cheaper, and have more power than PLCs. Pentium® systems widely available today outperform even the fastest PLC by margins of 20:1 or more[2]. A new generation of PCs becomes available every six to nine months[1,2]. By contrast new generation of PLC hardware becomes available every two to three years[2]. PCs with at least 16Mbytes are commonplace, while PLCs still have memory in the order of Kbytes. The PC supports more standard peripherals[2] available from many vendors such as CD ROM drives, sound cards, voice recognition, networking facilities, and so on, at very affordable prices. The PC is available worldwide, on short notice, from many vendors[2].

It has been envisioned that the next generation industrial controls should provide an open architecture and a single software development environment[7]. The trend will be to move away from closed, proprietary systems[6] due to the advantages an open architecture provides[9]. It gives more flexibility since users do have to "marry" to a particular supplier[10]. There is easier access to the latest technology since numerous independent developers are continuously advancing the functionality and ease of use of *PC-compatible* hardware and software. Also, it provides cost effectiveness because its large market ensures a competitive environment and economies of scale that drive prices to their lowest level.

The PC can provide a totally integrated solution that incorporates the functions of the PLC, the man-machine interface, and the programming terminal. It can provide process simulation/emulation so that complete software development can be done independent of the hardware. Also, it can provide sophisticated troubleshooting and diagnostic tools, providing in-depth analysis of the state of the machine, possible causes for malfunction, and recommended remedies[2]. You can even run off-the-shelf Windows® software for data analysis[1] while the control system is running[2]. Standard Windows data exchange methods can be readily applied to move information between the control system and the rest of the enterprise[2].

PCs for industrial control might be successful today for reasons analogous to those for the PLCs at the time of their introduction[5]; engineers today are well versed in computer programming and technology. Many sites already have significant PC programming expertise on hand. Also, the typical PC language paradigms lend themselves more readily to flowcharting techniques and languages of recent development for control programming, such as Sequential Function Charts.

DISADVANTAGES OF PERSONAL COMPUTERS

Commercial-grade PCs are not normally designed to tolerate the shock, vibration, temperature, and electrical noise frequently found on the manufacturing floor[2]. Even though hardware that meets these environmental conditions is readily available as both PLCs and PCs[2,3,8], this may increase the cost of implementing PC-based control[4]. PCs may not be cost effective for applications with few I/O points (90% of the market[4]). Very cheap PLCs are available for that market. PCs are visualized as solutions for more complex systems with many I/O points and complex control strategies. PC-based input/output interfacing may be as expensive, and more difficult to implement than that for PLCs[4]. PLCs are faster to restart after a power failure. They also have better ability to retain data so they are easier to restart from where they left off[4].

REFERENCES

[1] Bassett, E. W., "Real-Time PC Control", *Industrial Computing*, vol. 15, no. 2, February 1995, pp 14-17.

[2] Gee, D., "The Benefits of Personal Computer Based Control Systems", *Proceedings of IPC '95 Conference*, May 8-11, 1995, published by ESD, pp 111-120.

[3] Gee, D., "PC or PLC: What's in a Name", *Industrial Computing*, vol. 14, no. 7, July 1995, pp 24-26.

[4] Hohmann, T., "Why PCs Won't Kill PLCs", *Industrial Computing*, vol. 15, no. 10, October 1996, pp 6-11.

[5] Levine, P. S., "The Programmable (Logic) Controller: Adapting in an Environment of Change", *Industrial Computing*, vol. 14, no. 3, March 1995, pp 14-18.

[6] Pollard, J. R., "Open Architecture for Control is Coming, but When?", *InTech*, vol. 43, no. 4, April 1996, pp 45-48.

[7] Sadre, Ahmad, "21st Century PC Control", *Industrial Computing*, vol. 14, no. 15, May 1995, pp16-17.

[8] Trent, B., " Industrial PCs: Tough Enough?", *Industrial Computing*, vol. 14, no. 15, May 1995, pp 22-25.

[9] Verling, Jay, "Open Architecture Boosts Options or Factory-Floor Workstations", *InTech*, vol 43, no. 10, October 1996, pp34-38.

[10] _____, "The State of the PLC", *InTech*, vol. 43, no. 4, April 1996, pp 41-43.