Machine Control Using Coupled Computer/PLC Architecture

by Hamilton Woods, Accutrol, LLC

Abstract

For many years, options for machine control have included embedded computers and Programmable Logic Controllers (PLCs). More recently hybrid systems, such as Programmable Automation Controllers (PACs), have blurred the line between the traditional PLC and computers. There is a class of applications that require computational resources and device interfaces that are commonly available in an embedded computer, yet demand the determinism or uptime available in a PLC. A novel solution for this class of application is presented.

Introduction

To understand the current state of industrial automation, let’s look at the development of key technologies. The first programmable machine was possibly the Jacquard loom, completed in 1805 by Joseph-Marie Jacquard. Incidentally, “his machine aroused [such] bitter hostility among the silk weavers, who feared that its labor-saving capabilities would deprive them of jobs[, that] the weavers of Lyon not only burned machines that were put into production but attacked Jacquard as well. Eventually, the advantages of the loom brought about its general acceptance, so that by 1812 there were 11,000 in use in France.” [1] Henry Ford built what is considered the world’s first assembly line in 1913. Automation of production systems really began to take off with advances in electrical timers, micro-switches and protective relays during the 1930’s [2]. The first freely programmable computer (the Z1 mechanical computer) was completed in 1938 [3]. Introduction of the transistor in 1948 and the integrated circuit in 1958 [4] propelled developments in machine automation. The first PLC was introduced in 1968 by Dick Morley. In 1971 Intel created the first commercially available central processing unit (CPU) on a single integrated circuit chip: this innovation became known as the first microprocessor, the 4-bit Intel 4004 [5]. An entire industry, the general purpose computer, resulted from the development of the microprocessor. A general purpose computer that is used for machine control is referred to as an embedded computer.

“In the early 1970’s communications between PLCs started to develop. Modbus was introduced as the first industrial communications network. This communication network was based on a Master/Slave architecture using messaging to communicate between Modbus nodes.” [6]

During the decades that followed, advances in computers, PLCs, sensors and instruments allowed embedded computers and PLCs to automate increasingly complex operations. For machines that were produced in large volume, cost savings have been achieved by development of specialized embedded controllers. Specialized embedded controllers are not commonly used in systems that are produced in low volume, since development and modification is labor-intensive.

During the 1990s the distinction between a PLC and an embedded computer began to blur in devices that became known as Programmable Automation Controllers (PACs). As the choice of designation is left to the vendor, even the labels do not clearly distinguish the two. In general, a PAC is a hybrid between an embedded computer and a PLC. Typically, the features that distinguish a PAC from a PLC include ability to program with a higher level programming language, appearing as a computer with a file system on a network, access to multiple communications protocols, and off-the-shelf distributed processing. Some vendors have reclassified specific PLC models as PACs.

Argument

Availability of multiple categories of computing platforms (embedded computers and PLCs) is a blessing and a curse. Industrial processes come in all shapes and sizes (spectrum). There are many factors that play into the decision of which computing platform to implement for a particular application. PLCs tend to be used in systems that need to be highly reliable, deterministic, or quickly adapted to changes in the process that is being controlled. Embedded computers tend to be used in systems that are complex, that require intensive computations, that require local storage of large amounts of data, or that interface with instruments that require software drivers.

PLCs tend to be used for one class of application categories and embedded computers for another class of application categories. There are some application categories for which either platform is suitable. There are also application categories for which neither a PLC alone nor an embedded computer alone is well suited. These applications include complex systems that need to be highly reliable or deterministic.

For many years, we at Accutrol, LLC, implemented machine control using PLCs for simple systems and using DOS-based computers for systems that relied on a recipe of parameters to control a process or for systems that required specialty PC cards. Keep in mind that majority of the automation projects we have worked on are for small- and medium-sized manufacturers, for whom system cost is a significant factor influencing purchasing decisions. The DOS operating system provided a stable platform that allowed for software development in a high level language. Because DOS was not commonly used as a multi-tasking operating system, computer "crashes" were extremely rare events. Other stable operating systems have been available, even real-time operating systems. The high equipment cost for some of these systems and the large effort to implement best practices for a real-time application, put this option out of reach for some projects. One of the benefits of using a computer-based approach was that we were able to develop a single executable that could be used in a family of systems, configuring the software for a specific machine by an initialization file.

Eventually, DOS-based computers have become available from fewer sources and in limited stock. This prompted an examination into an appropriate replacement platform. Some of our customers were reluctant to consider Linux as an option. Windows-based computers did not satisfy the uptime requirements for industrial automation, the largest concern being the indeterminate state in which discrete outputs were left when a "crash" occurred. Not discovering any other reasonable approach, we began implementing systems, for which a computer would have been better suited, using PLCs. Configuring the software, which was developed for a family of systems, for a specific machine became more complicated. We relied on the capabilities of the operator interface panel (OIP) to store the PLC configuration file and recipe parameters. The OIP was also responsible for generating data logs. Connection to the plant network was accomplished through the OIP.

Maintenance of the software for these PLC-based systems was also complicated. Testing of any software upgrades, even if they did not modify the operation of PLC outputs, required access to the system on which it was to operate. Installation of the software on the target system required the customer to have proprietary PLC programming software and a special cable for connecting to the PLC.

Solution

In an attempt to use PLCs for the tasks for which they are well suited and computers for the tasks for which they are well suited, we decided to investigate the possibility of using a hybrid PLC/embedded computer platform to automate machine control. What we were looking for was a solution that would provide the greatest connectivity options and would also eliminate the opportunity for discrete outputs to be left in an indeterminate state in the event of a computer fault. Granted, there are real-time embedded controllers that can be used for these applications; but real-time embedded controllers tend to be relatively expensive and connectivity options are somewhat limited compared to off the shelf computers.

As Modbus is available for a great majority of PLCs, we chose to connect a computer to a PLC using Modbus. The computer would be responsible for commanding the PLC. The PLC would be responsible for direct inputs and outputs. For the simplest case, the PLC software performs negligible processing. The only processing is to determine if the connection between the computer and the PLC is still active and take action when the connection is not active. For this, we programmed the computer to send a heartbeat signal to the PLC. The heartbeat can be a frequent poll of the state of an ESTOP input. If the heartbeat signal is not present the PLC assumes that the computer or the cable between it and the PLC is not working properly, in which case the PLC executes a process to put all outputs in a safe state.

As we were determining the feasibility of this hybrid approach, we recognized numerous opportunities:

- PLC can provide interlocks for outputs,
- Processing performed by the PLC can be tailored to the application, relying on the PLC for actions that need to be deterministic,
- The PLC can be specified based on the I/O needed, with little regard to its processing limits,
- As the PLC software in the hybrid system almost always depends on PLC software elements that are common to all PLCs, development and testing can be somewhat decoupled from hardware construction,
- Any of a wide variety of high level programming languages can be used to develop the computer program (we use C#, Visual Basic.NET, and LabVIEW)
- Applications that require vision processing or significant numerical processing will require a computer anyway,
- Interfaces to instruments and other equipment are almost always easier in a high level language and example interface routines are often available from the internet,
- Operator login (authentication and authorization) are more straightforward in a high level programming language,
- Computer software components are much easier than PLC routines to reuse in future projects, especially when software needs to ported from a PLC of one manufacturer to a PLC of another manufacturer,
- Configuration parameters, recipe parameters and logged data can be stored on computer (developing text manipulation routines is much easier in high level programming languages available for computers),
- A computer monitor can provide OIP functions, especially using a touchscreen monitor
- Data can be shared with other computers on plant network using Ethernet and an ftp server resident on the computer, or using a USB memory stick when Ethernet is not available,

- For those cases when remote operation of the machine is needed (which should be rare), the computer can host a remote interface,

For our investigation we used Visual Studio.NET 2010 Express (available without cost from Microsoft) and NModbus, an open-source .NET dynamic link library (DLL) that provides an interface to devices that communicate using Modbus. The PLC we used was a Click PLC from AutomationDirect, a low-cost PLC whose programming software is available without charge. We interfaced to the PLC using Modbus over RS-485.

We quickly discovered that the Click PLC keeps track of the time since last transmission on the RS-485 port. Therefore, the heartbeat check only required that we monitor the time since last transmission and report lost heartbeat when that time exceeds a maximum allowable value.

The hybrid PLC/embedded computer approach performed as expected. We were able to monitor input (and output) states, control outputs, and monitor and update analog values. The only disappointment was that we hoped that the heartbeat signal (which we set to transmit every 200 milliseconds) would be transmitted to the PLC at least once per second. Although this was almost always the case, rarely the signal would be delayed beyond the one second boundary during CPU intensive processes. We had to set the maximum allowable value to 2 seconds. We relied on the Windows Forms timer, which is incredibly non-deterministic. When we need a more responsive indication of loss of heartbeat (remember, ESTOP is handled locally in the PLC and also in computer) we can implement a more deterministic timer in the computer software.

After we developed a working system, we developed a class (actually a Windows Form) that consists of components that are general to all applications. The details that are required for communicating with the PLC using Modbus (the device parameters and memory names and Modbus addresss) are read from configuration files. This allows the class to be copied from project to project with little or no modification.

We have developed software systems in such a way that they can be used in three different modes: ONLINE for use in a fully functional machine; OFFLINE for demonstration, training and testing without a PLC; and SIMULATION for demonstration, training and testing with a PLC but no functional machine. This allows portions of the software system to be modified and tested without hardware and, in some cases, without a PLC. Sections of the PLC software are dedicated to the SIMULATION mode, simulating proper operation of the machine without any hardware attached, which allows the computer software to be tested even without hardware. The software can be executed in OFFLINE or SIMULATION (even ONLINE, attached to the machine) mode from a laptop computer.

To date, we have used this hybrid approach in more than a dozen end-of-line testers that are in use around the US and internationally. The control of each of these testers is accomplished using a touchscreen panel computer connected to a Click PLC using Modbus over RS485. As customers need software features added (as long as there are no hardware interface changes), we are able to modify and test the software and then email the update to the customer. The update can then be installed on the machine over Ethernet or on a USB memory stick by copying the executable to the computer and restarting the software. The process of updating the PLC software, which occurs less frequently, still requires specific software and a PLC interface cable.

Conclusion

The automation industry is blessed with many options for computer control of machines. The requirements of different applications suggest the use of different control devices ranging from PLCs to PACs to specialized embedded controllers to embedded computers. For OEMs that standardize on a single computing platform, the specialized embedded controller is usually cost effective. For the system integrator, the choice of computing platform depends on many factors. Sometimes the choice is clear or is dictated by the customer. When the choice is not clear, it usually indicates that the control strategy has competing, often contradictory, requirements. For these applications, especially when cost is a factor, it is proposed that a combination PLC/embedded computer solution can be advantageous.

[1] "Joseph-Marie Jacquard | Biography - French Inventor." *Encyclopedia Britannica Online*. Encyclopedia Britannica, n.d. Web. 21 May 2015. <http://www.britannica.com/EBchecked/topic/299152/Joseph-Marie-Jacquard>.

[2] "History of Automated Assembly." *Brighthub Engineering*. Bright Hub, Inc., n.d. Web. 21 May 2015. <http://www.brighthubengineering.com/manufacturing-technology/126293-history-of-automation-in-manufacturing/>.

[3] "CHM Revolution." *Konrad Zuse*. Computer History Museum, n.d. Web. 21 May 2015. <http://www.computerhistory.org/revolution/birth-of-the-computer/4/84>.

[4] Bellis, Mary. "The History of Computers - Computer History Timeline." *About.com Inventors*. About.com, 05 Mar. 2014. Web. 21 May 2015. <http://inventors.about.com/library/blcoindex.htm>.

[5] "Intel Timeline: A History of Innovation." *Intel*. Intel Corporation, n.d. Web. 21 May 2015. <http://www.intel.com/content/www/us/en/history/historic-timeline.html>.

[6] "Programmable Logic Controller (PLC) History." *Programmable Logic Controller (PLC) History*. Automation Technologies, Inc., n.d. Web. 21 May 2015. <http://www.plcmentor.com/Articles/Newsletters/Programmable-Logic-Controller-PLC-History.aspx>.