The History of the PLC

as told to Howard Hendricks by Dick Morley

The following are some fables associated with the first ten years of the programmable controller business. These Fables may or may not have a basis of truth, but in general, they are the best that my Alzheimer-plagued memory can do at the moment. As has been often in other articles and reports, the startup of Modicon and the programmable controller industry as a whole is well documented. The programmable controller was detailed on New Year's Day, 1968, and from hence till now, a slow steady growth has allowed the manufacturing and process control industries to take advantage of applications-oriented software.

The early days however, were not as straightforward nor as simple. We had some real problems in the early days of convincing people that a box of software, albeit cased in cast iron, could do the same thing as 50 feet of cabinets, associated relays and wiring. The process was indeed difficult, and deserves some of the stories that I hope the reader will be regaled with as he proceeds onward through the tortuous swamp of my mind.

One of my earliest recommendations was that the programmable controller, according to my own system architecture specification, did not need to go fast because I felt as though speed was not a criteria because it would go as fast as we needed it to. The initial machine, which was never delivered, only had 125 words of memory, and speed was not a criteria as mentioned earlier. You can imagine what happened! First, we immediately ran out of memory, and second, the machine was much too slow to perform any function anywhere near the relay response time. Relay response times exist on the order of 1/60th of a second, and the topology formed by many cabinets full of relays transformed to code is significantly more than 125 words. We expanded the memory to 1K and thence to 4K. At 4K, it stood the test of time for quite a while. Initially, marketing and memory sizes were sold in 1K, 2K, 3K, (?) and 4K. the 3K was obviously the 4K version with constrained address so that field expansion to 4K could easily be done.

The question of speed, in part, was part of the early designs. No interrupts were necessary because the external signal conditions were directly written onto memory without any supervisory requirements or "operating system of the conventional type. This allowed the processor to pay attention to solving logic rather than housekeeping the I/O. As a result, of course, the processor had to have significantly more processing power than normally associated with this size computer; and secondly, the system had to be made to run fast.

We increased the memory size, as mentioned above, but to get it to run fast, we had to break up the machine into three distinct components. Initially, the programmable controller was conceived of a processor board and a memory, and that the algorithmic and logical manipulation would be done in software. This approach was painfully slow, both on the generic "store bought computers, and other items.

We did, however, manage to substantially speed up the machine by making a third major component. This was called the logic solver. A logic solver board solved the dominant algorithms associated with solving ladder logic without the intervention and classical software

approach of general-purpose processing. This meant that we ended up with three boards; memory, logic solver and processor. This single step allowed us to get the speed we needed in this application-specific computer to solve the perceptually simple problem of several cabinets full of relay wiring.

We had also assumed a modular approach to the programmable controller. In act, the name Modicon means MOdular DIgital CONtroller. The modularity, however, was soon abandoned because, as everyone knows, open architectures are no good. We instead had the marketing premise that a large footprint would contain within it the sets of problems we wished to solve. This meant that a buyer of programmable controllers could buy large numbers of the same units, and the software and hardware would be identical across a broad spectrum of applications in his factory. Service, maintenance and total life cost would be substantially lower than the perceived lower cost of an open architecture and modular expansion. Although at first, a supporter of the open architecture modular expansion, I soon became convinced by the marketplace, but this was folly.

We took one of our early units which was aimed at the machine tool industry because of my Bedford Associates consulting background, up to one of the early requesters of this equipment. This particular early requester was Byrant Chuck and Grinder in Springfield, Vermont. We took the machine up there, and it was heavy. This was the 084. The 084 was in the trunk of my old Pontiac, and since we needed help carrying it in, requested some of the people at Bryant to help us. We went out and opened the hood, and the first comment made by an outside viewer of the programmable controller said, "Thank God it’s not another pastel colored piece of sheet metal.

We can hypothesize from this particular comment that the ruggedness of the visual design was pleasing to him, and being human (as opposed to Martian), assumed that this same attitude went deep inside the construction of the machine in both the hardware and software. Indeed, this was the case, and the machine as a result, was built rugged, had no ON/OFF switch, had no fans, did not make any noise and had no wear out system.

To reminisce for a moment---in selecting the cores for the first memories, which in itself was a revolutionary step, we selected these cores and we applied Shannon’s Law. Shannon’s Law assumes that the signal-to-noise ratio is what makes signals good or bad. There are several ways to get the power from the signal-to-noise ratio; one is to code heavily, be triply redundant, and use lots and lots of error checking. There is another way, which is perfectly compatible with theory, which is to use lots of signal power in another domain. A nice switch, a car battery and a D-rated light bulb will work fairly well over a long time period.

Therefore, what we did was rather than going error checking, triply redundant and stuff, we got, and searched for and found high energy, large ferrite core memories that had lots on energy per bit. We still make the same assumption today. The energy per bit is extremely important---as Shannon’s theory said in his most famous 1948 paper, that the signal noise to power noise is what gives you transmission. the way we got signal power was to increase the energy per bit. This we felt was far more important than getting the energy per bit increased by means of doubly transmitting it. But I digress. Bryant Chuck and Grinder put it in, and liked the equipment so much that they never bought one. They in turn thought it was a good idea, and as many did at that time, tried to evolve their own.

One of our first major customers, however, was Landis in Landis, PA. We flew the equipment down in a private aircraft, and with apprehension because we were late (as usual), brought the equipment into Landis. In doing so, we tripped over the threshold. The equipment went KA-RASH onto the floor! Without much chagrin, we picked the equipment up, trundled it in. hooked it up, and low and behold, it worked quite well.

Now, Landis was pleased and surprised. They were pleased because it worked, but they were most pleasantly surprised---not because the equipment worked---but because the guys from Modicon fully expected the equipment to work in spite of it being dropped. In other words, the people from Modicon weren't nervous about the fact that it fell on the floor over the threshold.

Landis subsequently took and wrapped welding coils of wire around the machine to induce electro-magnetic noise to see if they could make it fail. We had them there! We used to test the programmable controllers with a Teslar coil that struck a quarter inch to half-inch arch anywhere on the system, and the programmable controller still had to continue to run. There was significant strangeness with respect to the programmable controller. For example, it had no ON/OFF switch. It had no means to load software. It had no fans. It ran cool. It could survive bad, physical and thermal environments. It was not computer industry standard. There were many things that were most difficult in the acceptance of the programmable controller, and early acceptance was most difficult indeed.

Our sales in the first four years were abysmal. Early innovators such as Landers and General Motors were, of course, heroes to our eyes, but they would buy small numbers of units and then test them in the field before they committed themselves later on. We had one customer in the utilities business that took them approximately six to seven years to make a decision to but the first one.

We never really sold any programmable controllers into the intended market which was machine tool control such as lathes, grinders and stuff, but we did, as luck would have it, stumble across the transfer line market which was and still is the mainstay, long-term market for the application of programmable controllers. Discreet parts manufacturing in an automatic environment, i.e., mass production, continues to be, and probably will be for the future, the mainstay of the programmable controller industry.

Some of the more interesting stories center around the personalities and experiences as opposed to the programmable controller. Modicon's third president (or fourth, if you count my two-week stint) was Don Kramer. When Don Kramer was chosen as president, we decided to go out and celebrate at the Lanum Club in Andover. At the time, we felt we should celebrate over both martinis and food. As we were leaving the shop for the Lanum Club, Don made the aside comment that "the place is dingy and needs a paint job. As we were leaving, I mentioned to Don that as president you have to change what you say, and not be very open---you have to be a little careful about what you say because employees, customers, and boards of directors tend to take what you say as truth. Rather than listen to the meaning, they listen to the literal statements, and one must be careful. We went over to the Lanum Club and had a nice glowing two hours of discussion, food, and drink. Coming back, as we entered the Modicon lobby, we noticed that there was scaffolding about and people were painting. We went over and asked Lou as to why these people are painting since, at the time, we don't have any money. Who ordered this paint job? And Lou looked Don Kramer straight in the eye, and said, "Why you did, Mr. Kramer. Nuff said.

As has been mentioned many times, your author, that's me---Dick Morley---is supposed to be the inventor of the programmable controller. This is at best, partially true. The thing that made the Modicon Company and the programmable controller really take off was not the 084, but the 184. The 184 was done in design cycle by Michael Greenberg, one of the best engineers I have ever met. He, and Lee Rousseau, president and marketeer, came up with a specification and a design that revolutionized the automation business. they built the 184 over the objections of yours truly. I was a purist and felt that all those bells and whistles and stuff weren't "pure, and somehow they were contaminating my "glorious design, Dead wrong again, Morley! they were specifically right on! the 184 was a walloping success, and it---not the 084, not the invention of

the programmable controller---but a product designed to meet the needs of the marketplace and the customer, called the 184, took off and made Modicon and the programmable controller the company and industry it is today. My compliments to the two chefs---Lee Rousseau and Mike Greenberg.

The issue of quality in programmable controllers is a story that is normally taken for granted. The gentle reader must remember that our engineering people came from the computer industry where reliability in those days was a phantom---a phantom of design, a phantom of cost. People felt that reliability was something other people did, and that if we only could deliver faster computers, even if they didn't work, everything would be fine.

When the programmable controller was designed, it was designed in to be reliable. We used lots of energy per information bit by utilizing D-rated components, large memory ferrite cores, relatively stable and large etchings on printed circuit boards, totally enclosed systems and conductive cooling. No fans were used, and outside air was not allowed to enter the system for fear of contamination and corrosion. Mentally, we had imagined the programmable controller being underneath a truck, in the open, and being driven around---driven around in Texas, driven around in Alaska. Under those circumstances, we anted it to survive. The other requirement was that it stood on a pole helping run an utility or a microwave station which was not climate controlled, and not serviced at all. Under those circumstances, would it work for the years that it was intended to be? Could it be walled in? Could it be bolted in a system that was expected to last 20 years?

The humorous side of this is though we did all those designs and very carefully tried to make this system as intrinsically reliable as we could, not by redundancy, but by building well. In other words, it was designed to be built, it was designed to be designed, and it was designed to be reliable. We, however, as engineers, didn't understand the accountants and manufacturing. those two have their grail, shipments by the end of the month. As far as we could ascertain at the time, shipments were made independent of quality and independent of whether or not the system ran.

In the early days of the programmable controller and Modicon, even though I wasn't a direct employee and an owner, I would give out my home phone number to many of our critical customers so that if they had a problem, they could call me directly. Several calls indicated that when we shipped near the end of the month, let's say October 34th, that the equipment would not run; and secondly, when they opened the box and took the machine apart, cards were missing, bolts were on the bottom of the cabinetry, and some of the cards were not fully inserted. In other words, to make the end of the month was much more important than to deliver equipment that ran. to put it mildly, we were pissed! How do we as engineers maintain quality without continual surveillance which is most difficult for the design and entrepreneurial mind set. What we did was specify and design "blue boxes. These were cabinetries that the system had to operate in and run continuously for a minimum of 24 hours, under load, and under varying conditions. The box was built out of plywood, but its primary intention was to heat cycle the programmable controller under various input/output loads. We also ran, as a specification, that a Tesla coil was to be used on the programmable controller, and that vibration and thumping with a hammer (rubber) would be part of the specification.

This may seem unscientific to many of you, but let us assume that you try to get your equipment to run while somebody purposely tries to destroy it with a rubber hammer or spark coil that he can put anywhere on the system. Remember, your intention is to make the processor stop. That combination significantly depressed those monthly shipments during the first period. As a result of that, however, the message got through. Not only did we build ovens and tests, and pay attention to heat and spark and RF emissions, we would run the system continuously even in the shipping crate to get the maximum number of pre-custom hours we could. It was important to us

that we found the mistakes and not the customer and his secondary customer.

The language itself, ladder lister, bears some discussion. This particular language was not the invention of Modicon. We hypothesize that the language is very old, and originated in Germany to describe relay circuitry. If one looks at ladder lister, it has been our technical community for so long, we somehow think those little symboligies actually look like relays. In fact, it,s a mnemonic form of rule-based language, very modern and very high level, but designed in a Darwinian fashion over a period of many decades.

The ladder logic construct, "If... Then... is a very powerful construct used today in expert systems and other rule-based languages. The symbology, allowing normally open and normally closed situations as well as parallel and serial representation, was used for many decades before the invention of the programmable controller. I have worked on machines where the number of C-size and D-size prints were hung in special racks, and would be up to three feet thick worth of documentation on those drawing sets.

The name ladder comes from the fact that on the right-hand of the drawing is one power rail and the left-hand side is the other power rail; and in between in a horizontal fashion, is the statement or sequential connection of logical elements which we call relays or relay logic. The initial 084 had only logic in its functionality, and as a result, was marginal. In other words, all we did was replace relays rather than enhance the functionality by a factor of ten which is the entrepreneurial rule. Immediately, of course, based on customer response and our own frustrations, we put thing in the ladder listing language such as addition, multiplication, subtraction, and other functionalities that went far beyond relay capability and entered the realm of mathematics and set theory. This was still not sufficient, however, and we needed some way to make a "call to a "subroutine using ladder lister symbology and representation.

A software engineer, Chuck Schelberg, and myself were in the conference room one day trying to ascertain how we could make a generic call to functionalities that far exceeded the relay symbology and representation, and came up with the "DX function. This function was a block function that would be an element on the ladder logic representation that could perform many functionalities including arrays, motor drive functions, servo functions, extended mathematical functions, PID loops, ad nauseam. We felt there would be an occasional representation and use of these functionalities, and that not much had to be done to the programmable controller other than to modify the software. Wrong again!

The first customer that took delivery of a programmable controller utilizing the DX function, had a capability to be predictable and operate in real time. The RUN light went out, and the time to execute a scan or complete transformation of the ladder logic went far beyond the time allowable. Every single line had a DX function on it. Again we learned that when you enhance functionality, people use it all. I have never designed a computer that had too much memory. I,ve only designed computers that have too little memory. The same thing applies to any other functionality. Conventional wisdom seems to think that price/performance depends on only one thing---price---when, in fact, my experience has been that the customer cares little about price.

This price/performance tirade being over, one of the lessons we learned is that the customer wants functionality over the entire life cycle cost installation of the job. the customer also wants ease of installation, to have some fun, and to be proud of the work he does. After he’s finished, he never wants to come back.. The equipment should work as installed and as based. At one time, the programmable controller meantime before failure in the field was 50,000 hours. This is far in excess of almost any other type of electronic or control equipment.

The concept of languages and high-level languages is important. The programmable controller,

as it evolved, began to request more and more power, and more and more memory. The memories continually went up as well as power. It is estimated that at one time, in the mid-1970s, that the programmable controller had the equivalent of two MIPS processor and 128 kilobytes of memory, which at that time was a significantly powered minicomputer capability. Why? High-level languages require power to run them. If we take the equivalent of the ladder lister statement "If... Then..., the high-level language as represented here, requires a substantial amount of interpretive compiler, if you will, generation of underlying code. In other words, this statement spawns significant underlying code that must be run quickly, reliably, and contain within it, all aspects of resource allocation and operations resource. The higher level the language, the more powerful the processor apparently has to be in order to run the language. Ladder lister is a high-level rule-based language which, until now, we haven't talked much about in these terms. Our customers treated the programmable controller as a box of relays, and well they should. Language theory is neither necessary not desirable for most of the customers to know. The customers, instead, understand their problem, and are indeed much smarter than the design engineers because the dimensions of their problem far exceed the relatively simple problem of designing a computer software system and language. Ladder lister requires high performance which is one of the reasons it has difficulty running on the personal computer even of today.

References

Send mail to rmi.info@barn.org for more information.
Please send mail to webmaster@barn.org regarding web site structure.

R. Morley Incorporated
586-3 Nashua Street, Suite 56
Milford, NH 03055-4992 USA
Tel: 603-878-4365 FAX: 603-878-4385