I did a lot of PLC programming, and now do quite a bit of .NET programming. It's very dangerous to make the switch either way, because a lot of the skills that you think should be transferrable (patterns and such) lead you very far astray.

The biggest difference that I tell people is that PC program code should be written as if other programmers are the audience, but PLC programs (ladder logic) must be written as if maintenance people are the audience. Maintainance in most facilities (particularly manufacturing) frequently connect directly to PLCs and in online mode they can watch the code execute graphically to figure out what's wrong.

For instance, if an output isn't turning on, they'll type the output electrical device ID into the find function of the programming software, find that output coil, and start tracing back from there looking for issues. One of the frequent mistakes that some PLC programmers make is to "map" their I/O into a structure (in PLCs, these are called user-defined types), and they use a copy instruction to move all the inputs or outputs over to the structure at once. Makes sense from a PC programming perspective, but it makes the maintenance person want to kill you. Typically the programming software provides a cross reference feature where they can specify that output coil, and it will tell them everywhere in the program that it's used. If you use a copy instruction to move 10 words of I/O into a 10 word data structure, he's got to sit there and count bits to figure out which bit in the source of the copy maps to which bit on the destination side of the copy. True, comments can help, but there's a problem with that too... PLCs store the whole program and allow you to upload the program from it in an emergency if you need to troubleshoot and you don't have a copy of the original program. The problem is that for space reasons, the PLC doesn't store the comments. So if the line is down, it's costing $5000 per minute in downtime, and a guy runs out there with a laptop, he might have to do a quick upload without comments and try to troubleshoot it. Having those copy instructions in there, wasting 10 minutes of his time, just cost the company $50,000 in downtime. These are the things you have to be aware of when writing PLC programs.

Some other tips: some PLCs have support for FOR loops. **Never** use them. For the same reason above, they make the code very difficult to troubleshoot for a maintenance person. This is because if you have one piece of code in the PLC that gets scanned more than once per scan (like the contents of a loop), then when you go into online debugging mode, the software can't show you the values for each of 10 loops that executed this scan, so you really have no idea what value you're looking at. Then you have to write all this tricky code to pull the loop values for a specific loop index out into some other tags (variables) that you can monitor. That's just one more impedance to fixing the problem in an emergency. Using a subroutine more than once per scan suffers from the same problem.

Indirect addressing (what we would call Arrays) are very difficult for maintenance people to understand. It's generally OK to use them when you're dealing with recipe management (storing and retrieving values for how to build your part) but you should try to stay away from it in the control part of the program.

In PC programming, of course we seek to re-use code as much as possible. However, in PLCs and control systems, downtime is extremely expensive, and hardware is expensive. Memory is cheap, and actually PLC programmers are cheap. Therefore, it's expected that if you have 10 identical things on your machine (like conveyor drives or something) that you will have 10 different files (subroutines), one for each drive, and each drive will have its own variables associated with them: e.g. Drive1_Run, Drive2_Run, Drive3_Run, etc. This is going to feel very "wrong" to you when you come from a PC programming background, but this is all because of the points I've made above. When you're in a downtime situation, and someone says that Drive 3 isn't working, you crack open the laptop, go to the file for Drive 3 and you look at the Run output rung. You start troubleshooting from there, while the program is executing. There's no breakpoints (the program never stops).

Good luck on your endeavors. I wrote up some more insights from my years of programming PLCs, if you want to check them out.

edited Nov 30, 2010 at 19:50

answered Jul 22, 2009 at 20:56

Scott Whitlock
13.6k 7 65 113