Contact and Coil

Nearly In Control

General Principles of PLC Programming

The PLC tutorials on this site focus on specific principles, but I'd like to point out general principles too. In fact, general principles of PLC programming are the same as PC programming, though the way we satisfy those principles can vary widely between those two domains.

Principle 1: Readability

This is number 1 because it trumps everything (short of functional correctness, of course). Many of the principles focus on making the code easy to change, but before you can change it you need to understand how it works at a deep level. The easier it is to understand, the easier it is to change, so readability drives most of my PLC programming tutorials and explanations. Also remember your audience. Readability means someone with only an electrical background and no C/C++/Java/C# experience should still be able to walk up and understand your logic. That is the entire point of PLCs.

Principle 2: Keep Things that Change Together Close Together

If you know that changing one piece of code will require you to change another piece of code, and you can't somehow combine them into a single piece of code, then at least put them next to each other. The more related they are, the closer they should be in your program. Do whatever you can to help your future self see the trap you've laid.

Principle 3: Once and Only Once

This is an ideal form of Principle 2. If you have an array that has 10 elements in it, and you need to reference the number of elements in lots of places, then declare a constant and use that everywhere. Then you only need to change it in one place.

However, don't get too carried away. Remember not to let this trump readability. I had a thought-provoking discussion with my colleague recently. All of our devices use millimeters, but we have to display our measurements in inches. We have lots of places where we multiply or divide by 25.4, which is the conversion factor between millimeters and inches. If you follow principle 3, then you should define a constant, e.g., MILLIMETERS_PER_INCH = 25.4 and use that everywhere. On the other hand, the conversion factor between

millimeters and inches is unlikely to change anytime soon, and even if it were, you could find all the instances of 25.4 in our program and replace them in less than 60 seconds with a search and replace tool. Plus the constant is just longer. Concise is good. Furthermore, the context in which it's used explains the value (because it's typically used like this: distance_mm = distance_inches * 25.4). For these reasons I'm OK, in this particular case, using 25.4 instead of a named constant.

My point is that these ideas aren't always black and white. Don't apply these blindly and assume you've done your job. Make sure you use your brain too.

Principle 4: Isolate Things That Change Separately

This is the corollary of Principle 2. Just because your system has three identical pumps now doesn't mean it'll have three identical pumps 5 years from now. You might feel like a genius because you made a function block to control those pumps, but when the feedback on one of them malfunctions and you need to go in and bypass that one feedback without messing up the feedback logic on the other pumps, you've just made your life more complicated. Plus, if it's the electrician that has to put that bypass in, how comfortable will they be modifying your function block vs. modifying a rung that only affects one pump? The best 2 am support call is the one you never get.

Principle 5: Use Patterns for Consistency

You are not the first person to program a PLC. Those who've come before you and learned through trial and error have settled on some useful patterns of ladder logic programming that perform specific functions in well understood ways. These are the nouns, verbs, and adjectives of our field. You can expect someone reading your logic to recognize these patterns quickly and understand what you're doing.

You'll also start coming up with patterns that are specific to your machine or facility. Patterns have the advantage that once you learn it, you understand it whenever you see it. Consistency is good.

Principle 6: Build a Domain-Specific Language

Whether you're programming a machine or a family of similar machines, you're likely to find that the same problems arise again and again. Use function blocks to create a short-hand notation for the nouns, verbs, and adjectives that are specific to the problems you're solving. Look for repeated logic. I don't mean repeated because the hardware repeats (because hardware changes) but look for cases where your *ideas* are repeated.

For a simple example, we have a lot of mechanical presses in our facility and we often want to know if a press is in a certain "window" such as from 90 to 180 degrees, or a more complicated test is from 350 to 10 degrees (because it goes through zero). I created a Window function to handle these simple tests and encapsulate the more complicated logic of when the window includes the 360 to 0 rollover point. The use of the function is readable, it's used widely, but unlikely to change in a way that might break all the places it's used, so it's a good candidate for a function block.

Another good candidate is a function block that logs an event to your plant-wide event logging system.

Conclusion

Remember, readability and correctness trump everything else. Being concise is good but not at the expense of being cryptic. A simple 3-rung pattern repeated 10 times is easier to understand than a single complicated 10-rung block, even if the latter is a third the size. Ask yourself at the outset, "what's likely to change?" and be honest with yourself. Let the answer guide your decisions. And above all, think, "why am I doing it this way?"

7

This entry was posted in Industrial Automation and tagged plc on October 23, 2017 [http://www.contactandcoil.com/automation/industrial-automation/general-principles-of-plc-programming/] .

4 thoughts on "General Principles of PLC Programming"

Jz
October 31, 2017 at 4:14 am

Thanks for the post, all very good points!
A bit of my experience – we have a "stupid" work ethic at our company. A lot of my colleagues (not everyone) don't care about readability at ALL! Incorrect variable names, messed up structure of program, very few comments (and a lot of them incorrect too), etc..
That is how they get the job done faster – you just throw things together without caring, debug most errors during commissioning and voila! And bosses love them for that.

Now I have to make changes in these programs, and it takes days, to get to know how everything works.

Rudolf
November 1, 2017 at 3:26 pm

Excellent article Scott! Thank you very much for taking the time, I really served a lot and especially the article on the patterns of ladder logic programming .. I have to learn on the fly and it is very rewarding to structure cor-

rectly the work that I like and I think that is how one develops more.

Thank you

Pingback: General Principles of PLC Programming | Contact and Coil – BrentHumphreys.net

Fakhri
November 4, 2020 at 1:29 am

Excellent
Principles.
