

Control System Programming Best Practices

Control system programming is both an art and a science. Utilizing multiple tools, languages, and functions allow engineers to produce a creative solution. However, due to the multiple methods and tools available, the road to success can vary greatly depending path chosen. The purpose of this article is to outline general, acceptable best programming practices across all control system platforms. The intent is not to define how to program specific functions, but to cover a level of programming content that provides an exceptional deliverable, consistent in approach and well planned.

Programming standards are methods of coding that have been declared acceptable and are typically defined and supported by control system vendors. Closely tied to programming standards are “best practices” which outline recommended methods and strategies to follow when applying and deploying programming standards. Best practices may also involve the addition of extra code segments and/or removal of redundant or dead code that are not used. Best practices are recommended ways of writing a segment of code, whereas programming standards are a specific set of rules to apply to coding style and techniques.

Define Your Structure

When developing a program from scratch or augmenting an existing program, you must step back from the details and define a structure. This provides a clear plan of how you will divide up the functions, variables, and provide a logical arrangement. Segmentation of functions allows for a design that can easily be expanded and followed by other programmers.

Start by developing a visual diagram of how the code will be structured. In some cases, you can clearly outline this structure within the actual code. This is comparable to building a legend table within a drawing and it should provide a map of how your program will be structured and/or any segmentation and development notes to clearly outline so the next programmer will be able to follow the same methods. This structure should include both the controller code as well as the HMI programming.

Document

Control System Programming Best Practices

Supporting documentation of functional requirements and/or specifications is essential to allow developing functions. Just as important are supporting documentation of the programming standards that will be used to achieve the objectives defined in the specifications. Most DCS have a standard library that outlines how each functional element will be configured, such as a valve, motor, PID, etc. that has supporting documentation. However, in other applications like PLCs, the structure does not exist and must be documented and developed.

The supporting documentation does not require a lengthy formal document but it must clearly outline the structure and programming standards that will be followed. A simple spreadsheet can be used to relay this information. However, the ideal location for this information is the actual program itself to ensure the standards are always coupled with the code and not misplaced.

Plan for Change

When developing a system, it is important to plan for future expansion, changes, and reductions. Therefore, when you are defining your structure, you should always be aware of potential changes and ensure that you do not paint yourself into a corner. Do not outline limited strategies that could potentially cause you to change your strategy. For example, if you segment your memory locations for each data type, be sure to have plenty of room for changes and/or expansions of the system in the years to come.

In addition, it is always a good idea to outline your structure on paper to visually identify how the code will be segmented and introduce some potential changes to test the organizational structure of your design. Planning for change is difficult because you don't know what you don't know. It is always recommended to review your concept with other more experienced engineers to ensure you are not overlooking the obvious. However, there is a balance here between planning for extra-scope future expansions and meeting the requirements. Allow for future expansion, but do not devote significant resources to out-of-scope development.

Know Your Resources

System resources are a vital part of what makes software run efficiently and the

Control System Programming Best Practices

last thing you want to do is overburden your system's capabilities, which could cause latency or even crash the system. This issue was more prevalent in years past, due to memory and processing speed limitations. Technology has advanced and you now have much greater capabilities within a system to provide a more structured and organized code. However, the resources are not unlimited and it is always good practice to understand where the limitations are and how your structure is impacting those resources.

This is accomplished by performing a loading test, which will help you get a feel for where your limitations are. To do this, create an application with extensive loading using additional structure and routines that are far outside of the existing scope that you are trying to achieve. Not doing this is irresponsible. You would be blindly adding structure and overhead processing requirements without knowing how much room you have to grow the application. A resource test may require multiple application expansions, but finding the limit is crucial for success.

Reuse Code

The easy part of control systems is that we continually reuse of the same functions to achieve the logic task at hand. You do not want to hard-code the same functions over and over, which would incur additional costs such as increased memory, processing speed, and extra time to make changes to an application that is programmed not utilizing function calls.

By separately coding common code, you also introduce an opportunity for error down the road because changes to one instance must be accurately copied to all other instances by hand – an invitation for human error. All functions that are used should be referenced from a single source library and called when needed. This will require for you to code the functions on a general level and, in some cases, you may want to include additional parameters or variable to help the function account for different or additional attributes. This will allow the management of these functions to be common instead of isolated by system requirements. Meaning, you do not want to create a library of existing functions and end up managing an unmanageable library.

Control System Programming Best Practices

Be Consistent

Consistency is a common element to success on many different levels. Programming is no different and nothing is more frustrating than reverse engineering a control system to follow a design concept and find inconsistent practices. This shows immaturity in a design and can cause multiple issues over the life-cycle of a control system.

In some cases, alternative methods may be discovered after the initial development and the decision to change course should be solely dependent upon system performance and additional functionality. If you have identified a more clever method that may have a “cool” factor to it, but it does not result in improved performance or save enough development time to cover all of the time previously spent in the application, then it’s generally best to stick with the original method. Moving forward you may decide to utilize it in the future, but to achieve good practices stay the course.

Use Your Tools

A good programmer should have a good understanding of all the tools that are accessible. If you don’t stay up to date with what’s currently available, you may be cheating yourself and potentially losing some work efficiency. Take the time to test the programming tools to ensure the programming and deploying utilize the most efficient methods available.

As an example, instead of embedding multiple “IF THEN” statements you should use “CASE” statements to provide more efficiency, as well provide a more readable structure. Reach out to other more experienced engineers and discuss alternative tools that you may not be familiar with and set up a quick test to gain the necessary experience.

For clarity, this is not a suggestion that you should experiment with multiple methods to achieve a singular goal within your application. This could interfere with and sacrifice the previous concept of being consistent. This simply means you should know your tools prior to designing your application to ensure that you can provide an adequate solution given the defined deliverable period.

Control System Programming Best Practices

Housekeeping

Good housekeeping within your program is always good practice. In some cases, you have to build functions or things to help manage the data and keep your work area clean. If you allow alternative work areas to pile up trash, you can cause confusion and cause a hindrance or force unforeseen consequences in your application. All shelving or play areas should be compartmentalized. Meaning, every application needs a defined area to take care of alternative functions or shelve items for later use, but when those items are complete they should be cleaned up. Good housekeeping practices are part of being organized, which promotes efficiency and professionalism.

Commenting on Code

Nothing is more frustrating than tracing code that is poorly commented or even worse inconsistently commented. Meaning, the terminology, tag naming, and description used within a program should be just as consistent as the code itself. The code itself explains how something is happening and the comment should reflect why this is happening.

Commenting should be performed at each level. Meaning, each function itself should have a brief comment, but each section of code should include a header comment to explain the breakdown. Then, the application as a whole should be commented on to help the person coming behind you follow your consistent train of thought. Comments should be concise, consistent, to the point, and written while you are programming and not as an afterthought.

One particularly valuable kind of comment is descriptions of why you did NOT code something in a particular way. If there is a standard way of coding something, and you chose to do it a different way (whether you tried it the standard way first or not), you should explain why you chose to do it differently in a comment. This is as much for yourself as anyone else – when you come back and look at your code a year later and ask “why did I do this?” The answer will be right there.

Log Your Changes

Keeping track of changes within an application is essential to maintaining it over

Control System Programming Best Practices

the life-cycle. Change logs should be incorporated within the application and preferably in the header section for overall changes. However, if changes are needed at the function layer, they should be reflected in the overall header as well as at the comment layer in the function itself.

A good change log includes a log entry or revision number. As well as the date and person making the changes. The description should outline the need for the change and explicitly define all aspects of the application that are affected. Lastly, the change log should be consistently listed in a reserved area of the application.

Summary

Programming methods and strategies are subjective to an extent, and the flexibility of using multiple techniques allows engineers a creative avenue to produce solutions. Being creative is a form of self-expression that must be balanced with good practices to ensure the solution fits the overall objectives of the end-user.

The end-user obviously has common goals shared with the developer but also has additional objectives such as usability, expansion capability, and the ability to troubleshoot and interject functions to support production on an as needed basis. Following best practices should be a main consideration when providing a final solution for the end-user to ensure that the programming deliverable follows acceptable industry practices.

These methods listed above provide the overall usability, flexibility, and maintainability to ensure both developer and end-user are provided with an efficient quality driven solution that can withstand the product lifecycle. Engineers should strive to deliver a higher standard of programming to allow our clients to be successful at manufacturing. Every project should consist of solid solutions that enables our businesses to thrive in the manufacturing environment.

Work with our engineering team to ensure your control system is programmed efficiently and is helping your operation reduce risk, improve quality, and increase efficiency in your processes.