WHITE paper

Programming Controllers: C *vs.* Relay Ladder Logic

By G. Young, Customer Support Manager

This paper examines two methods of programming embedded controllers and describes the benefits of using C-Language in industrial control applications.

The World of programmable controllers has become more sophisticated if not more complicated. No longer is it sufficient for a machine to endlessly perform the same series of actions. Systems now, in addition to performing repetitive tasks, must optimize procedures in order to maximize production while keeping basic statistics (a requirement of quality assurance). Moreover, many systems must interact with one or more third-party devices with interfaces that are alien to the controlling Programmable Logic Controller (PLC). Some systems must also interface with at least one enterprise-wide network that provides management with up-to-the-minute information.

Relay Ladder Logic (RLL) was designed to provide more flexible software simulation that would replace the relays, timers, and counters common to control systems of the 1960s and 1970s. RLL has been successful in not only achieving, but in surpassing its modest goals. RLL is a powerful tool when applied to an application that performs specific tasks. However, RLL cannot meet the multiple challenges of today's PLCs even though some RLLs have been retrofitted with features to handle more complicated application demands. At best, these enhancements are awkward and slow and are either difficult or nearly impossible to use. RLL is simply not suited for today's more complicated industrial problems.

C, on the other hand, was designed to be a general purpose programming language. Not only does C dominate the development of applications for personal and corporate computing, it dominates the development of embedded systems. Therefore, C is used to program everything from word processors and databases to microwaves and TVs. In essence, C is everywhere.

The reason for C's popularity is its general-purpose nature. Because C makes no assumptions about its environment or what types of tasks it will perform, programs written in C use resources much more efficiently than equivalent RLL systems.

Most PLCs provide the user with a fixed number of devices such as auxiliary relays, registers, and timers. Eventually, every RLL programmer encounters a situation where a fixed allocation of assets is not acceptable for an application. For example, a particular application may only require a few auxiliary relays and timers, but requires more registers than an RLL application can make available.

When programming in C, allocating memory is not a problem because the programmer is in control of how resources are allocated.

The most aggravating aspect of this scenario is that the PLC often has the needed memory, but has allocated that memory to unneeded auxiliary relays and timers. When programming in C, allocating memory is not a problem because the programmer is in control of how resources are allocated. As long as a programmer does not attempt to allocate more resources than available, C allows the programmer to decide how to use resources.

Another important advantage of C is its execution focus. RLL was written to simulate hardware's inherently parallel operation. Parallel operation is accomplished by executing every "rung" of the RLL program on each scan. While this achieves the desired results, it has one major drawback: scan times increase as the size and complexity of the program increases. When attempting to use RLL to perform complex tasks, scan times, which can increase to more than 10 milliseconds, are generally too slow for most applications.

C, however, has a program counter that starts at the function main (the first function executed in any C program). The programmer counter works its way through the code, directed by flow control statements written by the programmer. A processor running a C program is executing only one sequence of instructions at a time.

C can solve most programming problems and is supported on all major programming platforms.

While one sequence of instructions is executed, the rest of the program is idle, waiting to execute the next set of instructions. This single execution focus gives the programmer incredible power to decide which aspects of the application need the most intense attention and which only need to be checked periodically. Without this control, handling high-speed events (such as deployment of an airbag) would be impossible.

In today's more sophisticated realm of programmable controllers, it is well worth the time invested to become a proficient C programmer. C can solve most programming problems and is supported on all major programming platforms. Ultimately, a good C programmer can write programs that out perform those of an equally competent programmer using RLL.

C and RLL Characteristics

C Programming Language	Relay Ladder Logic
• Programming flexibility provides many means to solve a problem. So many, in fact, that a novice often gets overwhelmed by the possibilities.	• So restrictive that the simplest problems often seem to solve themselves.
• Ability to focus execution on a small section of code. Important for applications where certain functions are not done often enough or are skipped entirely.	• Executes every section with every scan. There is no chance of code not executing.
• Less intuitive for beginners and a little harder to learn than RLL.	• Can be learned relatively quickly.
• A useful set of basics can be learned quickly, but the more complicated aspects of the language can be quite confusing even to programmers with moderate experience.	• Designed to perform a single task, and performs that task well.