The Buzz on Embedded Development Productivity

Doug Johnson
Embedded Hardware/Firmware Engineer
Wilmington Research and Development Corp.
Originally Published: July 31, 2002
Republished: January 20, 2021

Foreword - WRD has since moved away from ProductMaker as a platform, but our new designs are based on a suite of pre-designed circuits and software, tailored to fit your application saving on time and physical space, all while reducing your recurring costs as much as possible. In this, the concept behind ProductMaker lives on in our embedded development philosophy, and as such this paper is as relevant as it was when it was first published.

If you would like to learn more about what we do, or look out for more articles from WRD, check out our social pages, or shoot us an email:

LinkedIn | Twitter | Email

Introduction - Any embedded development discussion is going to be a discussion about the "process": How does the programmer define the sequence of operation; how does the CPU interpret the definition for execution; what hardware architecture will be chosen; what preexisting aids, i.e. hardware/software tools are available; and how do you manage it all, and deliver a well-executed, properly operating target. All of this is what I mean by the “process” i.e., “what is used, and how it’s used”.

"The search for the Holy Grail" in embedded development is a search for a more productive process. Recently, the buzz about this issue has been about as subtle as a F-14 on full after-burner. Everyone is looking for a way to improve the process, and there has been a lot of press on the subject.

Historically, most process improvements have fallen into one of the following areas:

- Language: higher levels of abstraction such as C, C++, JAVA; industry specific such as Relay Ladder programming; ease of use for "normal" people such as BASIC or functional block (flow chart) type programming
- Target hardware: memory density; flash memory; higher throughput; lower cost; improved functionality (peripherals); increased integration; etc.
- Development tools: assemblers; compilers; emulators; simulators; debuggers; integrated development environments; etc.
- Methodology: this has to do with the definition, and implementation of methods shown to provide productivity gains.

A discussion on the virtue of these “improvements” is difficult because, among other things, the word improvement implies the “newer” thing is “better”. For example, I have almost never seen an article that says that C is better than C++, yet on small, cost constrained systems, this might likely be the case. Frequently people using the “latest” technology feel that the improvement is universal and should be applied universally. This is a source of some of the zeal that sneaks into discussions concerning language and processors. The truth is more complicated and requires a broader view of the objectives.

Recently there has been some buzz on a couple of "new ideas" that are significant: System on Chip (SoC); and Platform Development. (SoC implies a single chip implementation, platforms are more general and are not necessarily single chip) With both, you start with a known, pre-

determined hardware/software base. You then make changes and additions required for the intended application. The improvement in productivity comes from the fact that your starting point is pre-done and is (hopefully) trouble free. Also, as the platform is reused repeatedly, the developer gains familiarity (productivity). And finally, if the platform covers a wide application base, and is low enough in cost, it is possible to use the platform to cover an entire range of product offerings. This "single processor, single platform" approach provides a huge productivity opportunity, as it makes it possible for a single embedded engineer to design and support a wide range of products. (The alternative is having the developer support multiple products using multiple processors, a tall order, or having an embedded team, which is costly)

The “platform” concept is not new. Many will recognize that a PC (embedded or not), with any commercial OS, is a platform. The difference is that neither the PC hardware, nor most OS's, were originally designed for the purpose of embedded controls. These newest SoC and Platform architectures are being designed from the ground up for embedded use.

The WRD ProductMaker is a development platform designed specifically for the embedded controls market. Having spent an entire career designing embedded control systems, it is clear how important reusability and system architecture is.

The ProductMaker is a highly flexible hardware/firmware platform based on a powerful yet inexpensive 8 bit microprocessor (Z180), pre-engineered schematics (which cover most embedded control requirements), and an automated firmware configuration process which not only contains all peripheral drivers, task scheduler, floating point routines, & ISR's, keyboard scanning, LCD/LED displays, Rom monitor, etc., but integrates these into a seamless custom control engine that lets the system designed concentrate on the application instead of the integration details.

This platform is used extensively at WRD and is key to WRD's ability to provide high quality, high functionality, low-cost custom embedded controls, with short development times and low NRE.